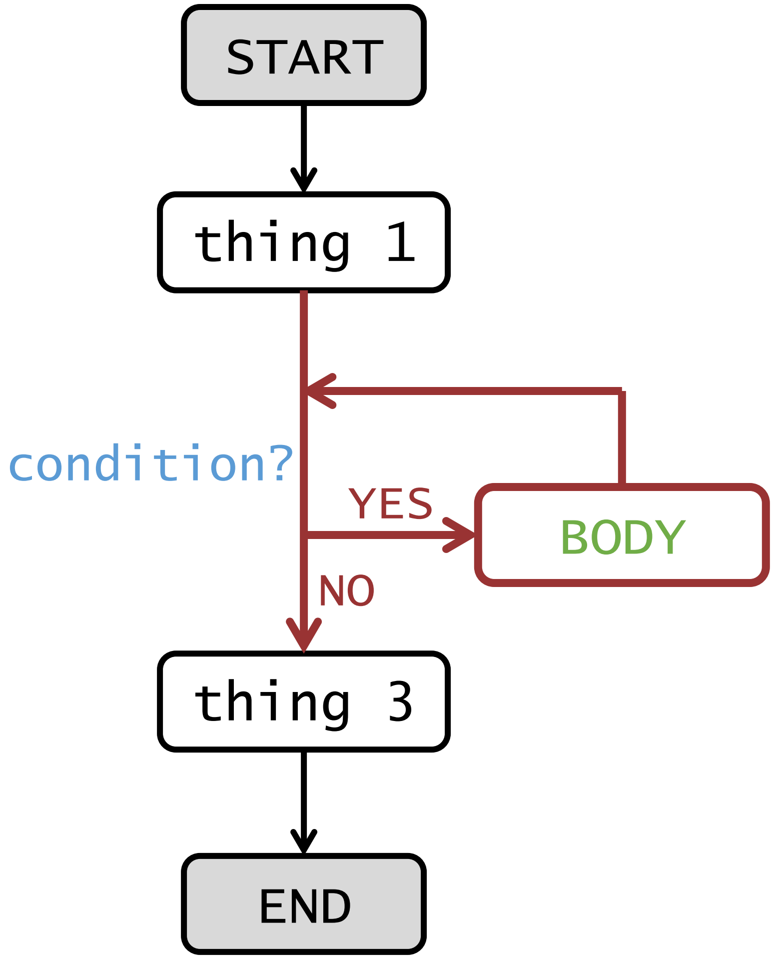
You can access these slides on the course Github:
https://github.com/natrask/ENM1050

ENGR 1050
Intro to Scientific Computation

Lecture 05 – Nonlinear solvers:
Newton, bisection, secant methods

Prof. Nat Trask
Mechanical Engineering & Applied Mechanics
University of Pennsylvania

Last time: `while` loops




```
while CONDITION:
    BODY
```

While `CONDITION` is `true`, BODY is executed repeatedly

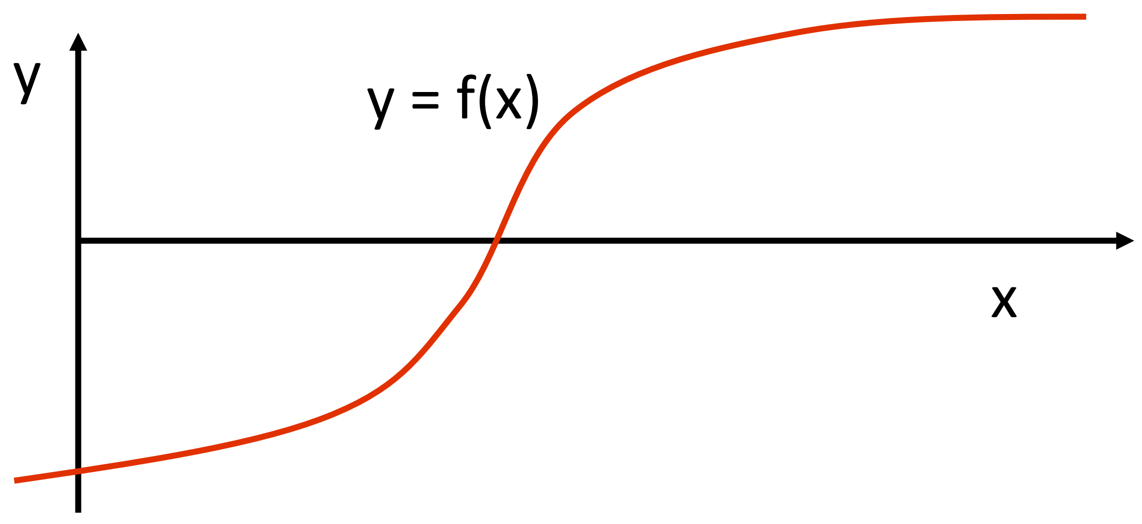
BODY executed as a block

`CONDITION` is checked only at the start of each block execution

Today: Applied while loops (aka finding zeros)

Finding the zero crossings of a function is a classical numerical problem with wide applications

Given a function f(x) our goal is to find values of x for which f(x) == 0



Optimization Problems

A wide range of applications in science and engineering take the form of optimization problems where the goal is to find the ‘best’ value for some parameter or set of parameters.

Consider for instance the problem of finding the right launch angle for your cannon to hit an intended target or the right price to charge for an iPhone to maximize profit.

Problems of this form can be viewed as finding the value that optimizes some given cost function, f.

AKA solve $f'(x) = 0$

What do we lose when we need numerical methods?

THE QUADRATIC FORMULA

If $ax^2 + bx + c = 0$ but $a \neq 0$

then

$$\boxed{x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}}$$

DISCRIMINANT

- $b^2 - 4ac > 0$ two real solutions
- $b^2 - 4ac = 0$ one real solutions
- $b^2 - 4ac < 0$ zero real solutions

What do we lose when we need numerical methods?

The solution of $ax^3+bx^2+cx+d=0$ is

$$
\begin{aligned}
x \quad = \quad & \sqrt[3]{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)+\sqrt{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)^2+\left(\frac{c}{3a}-\frac{b^2}{9a^2}\right)^3}} \\
+ \quad & \sqrt[3]{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)-\sqrt{\left(\frac{-b^3}{27a^3}+\frac{bc}{6a^2}-\frac{d}{2a}\right)^2+\left(\frac{c}{3a}-\frac{b^2}{9a^2}\right)^3}} \quad - \quad \frac{b}{3a}
\end{aligned}
$$

Cardano, 1545

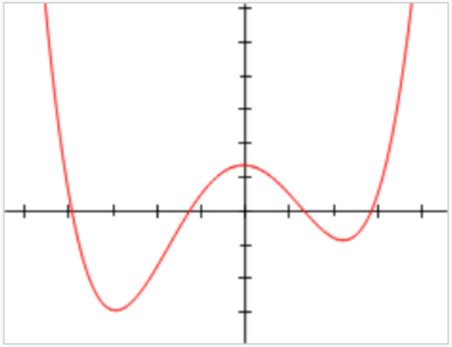
What do we lose when we need numerical methods?

In mathematics, a **quartic equation** is one which can be expressed as a *quartic function* equaling zero. The general form of a quartic equation is

$$ax^4 + bx^3 + cx^2 + dx + e = 0$$

where $a \neq 0$.

The **quartic** is the highest order polynomial equation that can be solved by radicals in the general case (i.e., one in which the coefficients can take any value).



Graph of a polynomial function of degree 4, with its 4 roots and 3 critical points.

History [edit]

Lodovico Ferrari is attributed with the discovery of the solution to the quartic in 1540, but since this solution, like all algebraic solutions of the quartic, requires the solution of a cubic to be found, it could not be published immediately.[1] The solution of the quartic was published together with that of the cubic by Ferrari's mentor Gerolamo Cardano in the book *Ars Magna* (1545).

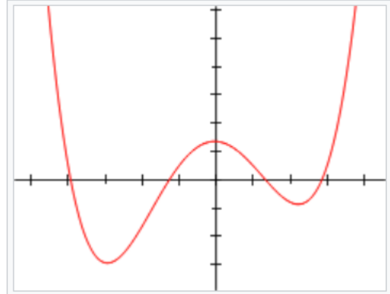
What do we lose when we need numerical methods?

In mathematics, a **quartic equation** is one which can be expressed as a *quartic function* equaling zero. The general form of a quartic equation is

$$ax^4 + bx^3 + cx^2 + dx + e = 0$$

where $a \neq 0$.

The **quartic** is the highest order polynomial equation that can be solved by radicals in the general case (i.e., one in which the coefficients can take any value).



Graph of a polynomial function of degree 4, with its 4 roots and 3 critical points.

History [edit]

Lodovico Ferrari is attributed with the discovery of the solution to the quartic in 1540, but since this solution, like all algebraic solutions of the quartic, requires the solution of a cubic to be found, it could not be published immediately.[1] The solution of the quartic was published together with that of the cubic by Ferrari's mentor Gerolamo Cardano in the book *Ars Magna* (1545).

$$r_1 = \frac{-a}{4} - \frac{1}{2}\sqrt{\frac{a^2}{4} - \frac{2b}{3} + \frac{2^{\frac{1}{3}}\left(b^2 - 3ac + 12d\right)}{3\left(2b^3 - 9abc + 27c^2 + 27a^2d - 72bd + \sqrt{-4\left(b^2 - 3ac + 12d\right)^3 + \left(2b^3 - 9abc + 27c^2 + 27a^2d - 72bd\right)^2}\right)^{\frac{1}{3}}} + \left(2b^3 - 9abc + 27c^2 + 27a^2d - 72bd + \cdots\right.}$$

$$r_2 = \frac{-a}{4} - \frac{1}{2}\sqrt{\frac{a^2}{4} - \frac{2b}{3} + \frac{2^{\frac{1}{3}}\left(b^2 - 3ac + 12d\right)}{3\left(2b^3 - 9abc + 27c^2 + 27a^2d - 72bd + \sqrt{-4\left(b^2 - 3ac + 12d\right)^3 + \left(2b^3 - 9abc + 27c^2 + 27a^2d - 72bd\right)^2}\right)^{\frac{1}{3}}} + \left(2b^3 - 9abc + 27c^2 + 27a^2d - 72bd + \cdots\right.}$$

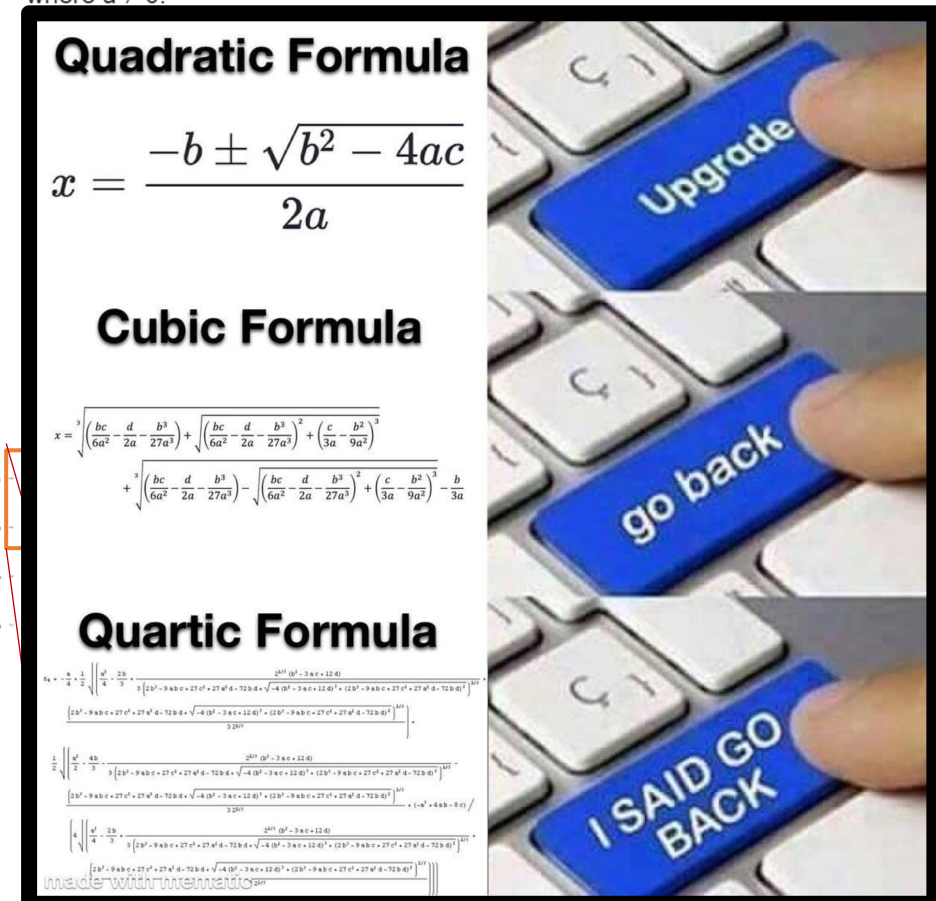
https://en.wikipedia.org/wiki/Quartic_equation#The_general_case

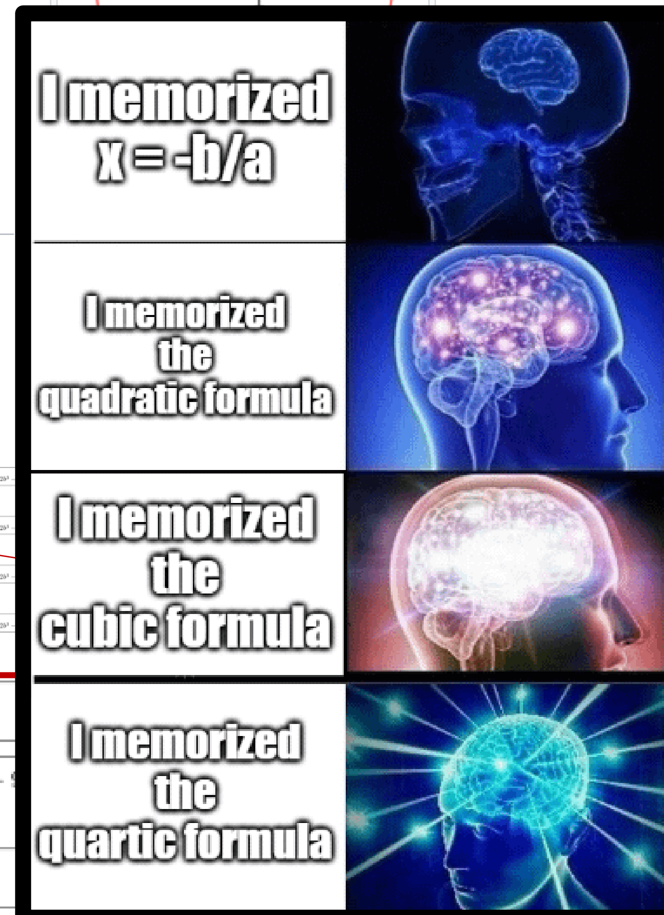
What do we lose when we need numerical methods?

In mathematics, a **quartic equation** is one which can be expressed as a *quartic function* equaling zero. The general form of a quartic equation is

$$ax^4 + bx^3 + cx^2 + dx + e = 0$$

where $a \neq 0$.



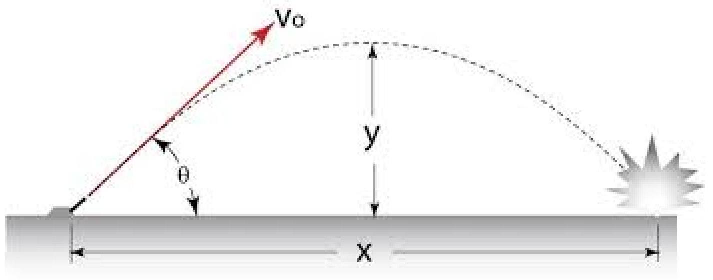


https://en.wikipedia.org/wiki/Quartic_equation#The_general_case

What do we lose when we need numerical methods?

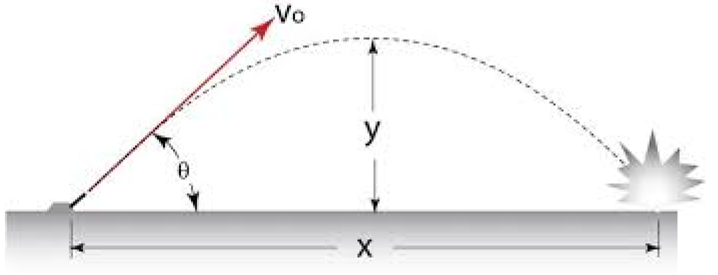
Deep learning is (often) minimizing the mismatch between a neural network and a dataset

Cannon Problem



You are launching a projectile with an initial velocity of $v_0 = 10$ m/s at an angle θ towards a target at x = 3 m away. What is the maximum height of the projectile?

Cannon Problem



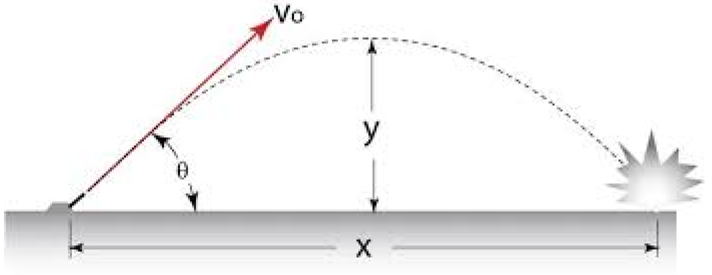
You are launching a projectile with an initial velocity of $v_0 = 10$ m/s at an angle θ towards a target at x = 3 m away. What is the maximum height of the projectile?

$$x(t) = v_0 \cos(\theta)\, t$$
$$y(t) = v_0 \sin(\theta)\, t - \frac{1}{2} g t^2$$
$$y'(t) = v_0 \sin(\theta) - gt$$
$$y'(t_f) = 0$$

$$\Rightarrow \quad t_f = \frac{v_0 \sin(\theta)}{g}$$

$$y(t_f) = \frac{v_0 \sin^2(\theta)}{g} - \frac{1}{2} g \left(\frac{v_0 \sin(\theta)}{g}\right)^2$$

Cannon Problem



You are launching a projectile with an initial velocity of $v_0 = 10$ m/s at an angle θ towards a target at x = 3 m away. What is the maximum height of the projectile?

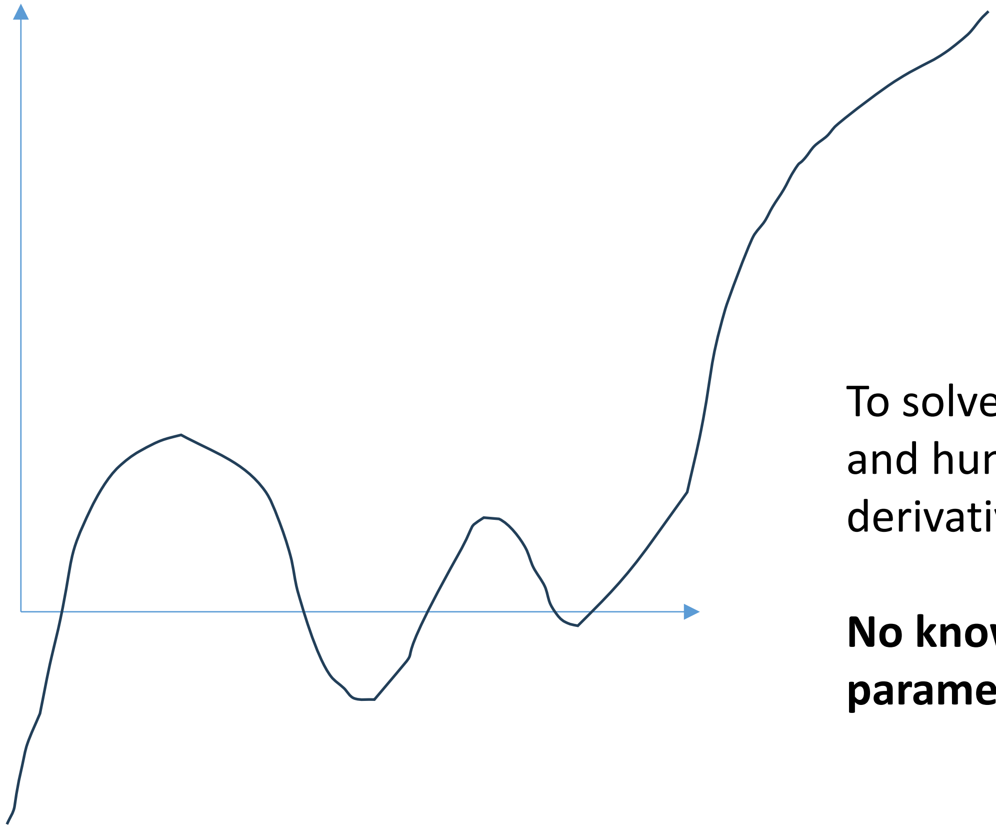
$$\boxed{\begin{aligned} x(t) &= v_0 \cos(\theta)\, t \\ y(t) &= v_0 \sin(\theta)\, t - \frac{1}{2} g t^2 \\ y'(t) &= v_0 \sin(\theta) - gt \\ y'(t_f) &= 0 \end{aligned}}$$

$$\Rightarrow \quad t_f = \frac{v_0 \sin(\theta)}{g}$$

$$\boxed{y(t_f) = \frac{v_0 \sin^2(\theta)}{g} - \left(\frac{v_0 \sin(\theta)}{2}\right)^2}$$

- Linear scaling with initial velocity
- How height varies with angle

What do we lose when we need numerical methods?

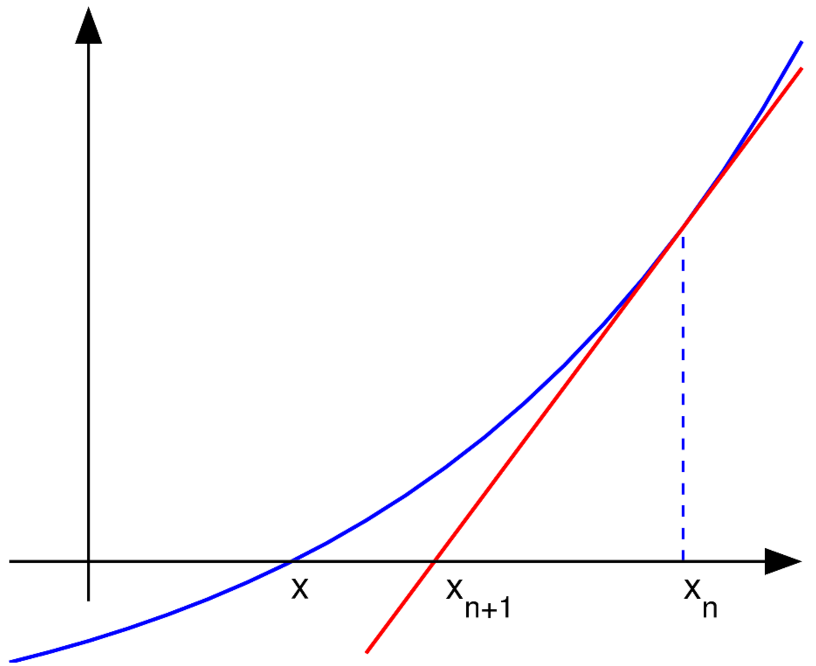


To solve numerically, we will guess a solution and hunt for where it crosses zero, using derivative information to “chase downhill”

No knowledge about dependence on parameters in the equation!!!

Newton's Method

Newton's method for root finding uses the derivative of the function. At each iteration we approximate the function using its derivative, predict where the zero will be and jump to that location.



What is the equation of the red line, call it $f(x)$?

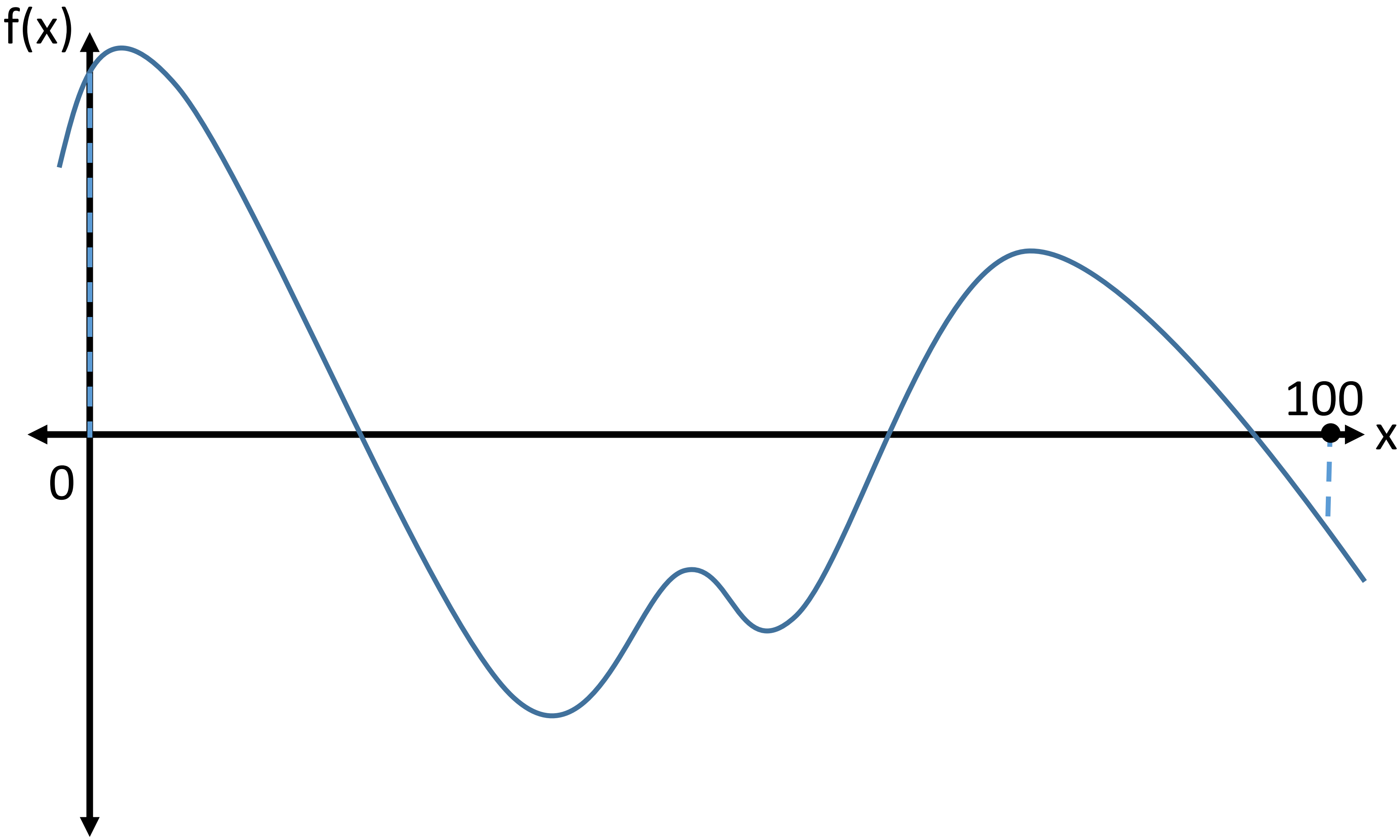
Taylor series approximation $f(x) \approx f(x_n) + f'(x_n)(x - x_n)$

At what x is $f(x) = 0$? $0 = f(x_n) + f'(x_n)(x - x_n)$

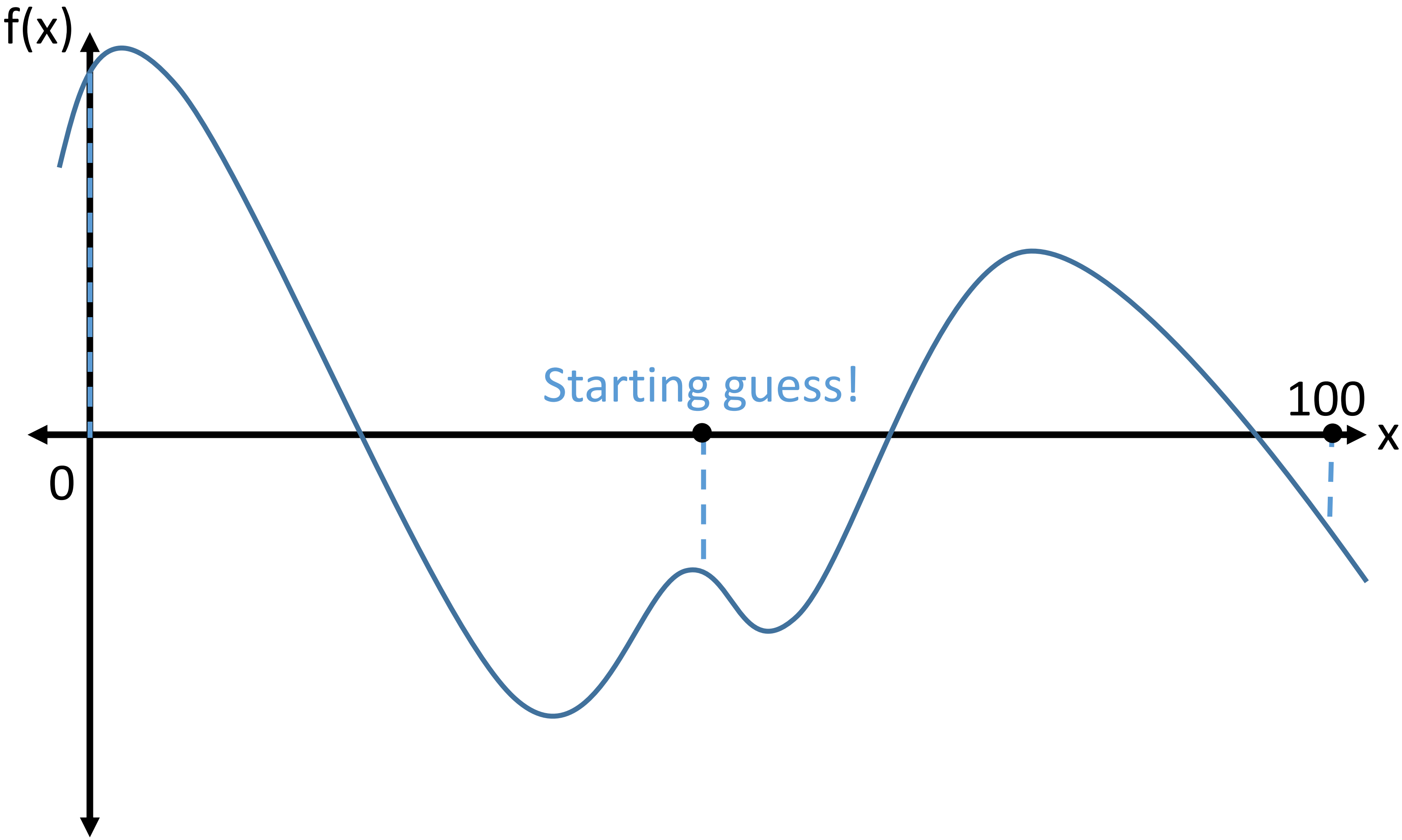
This is the new guess x_{n+1} $\boxed{x = x_n - \frac{f(x_n)}{f'(x_n)}}$

Basic Idea: x_{n+1} is closer to the root than x_n

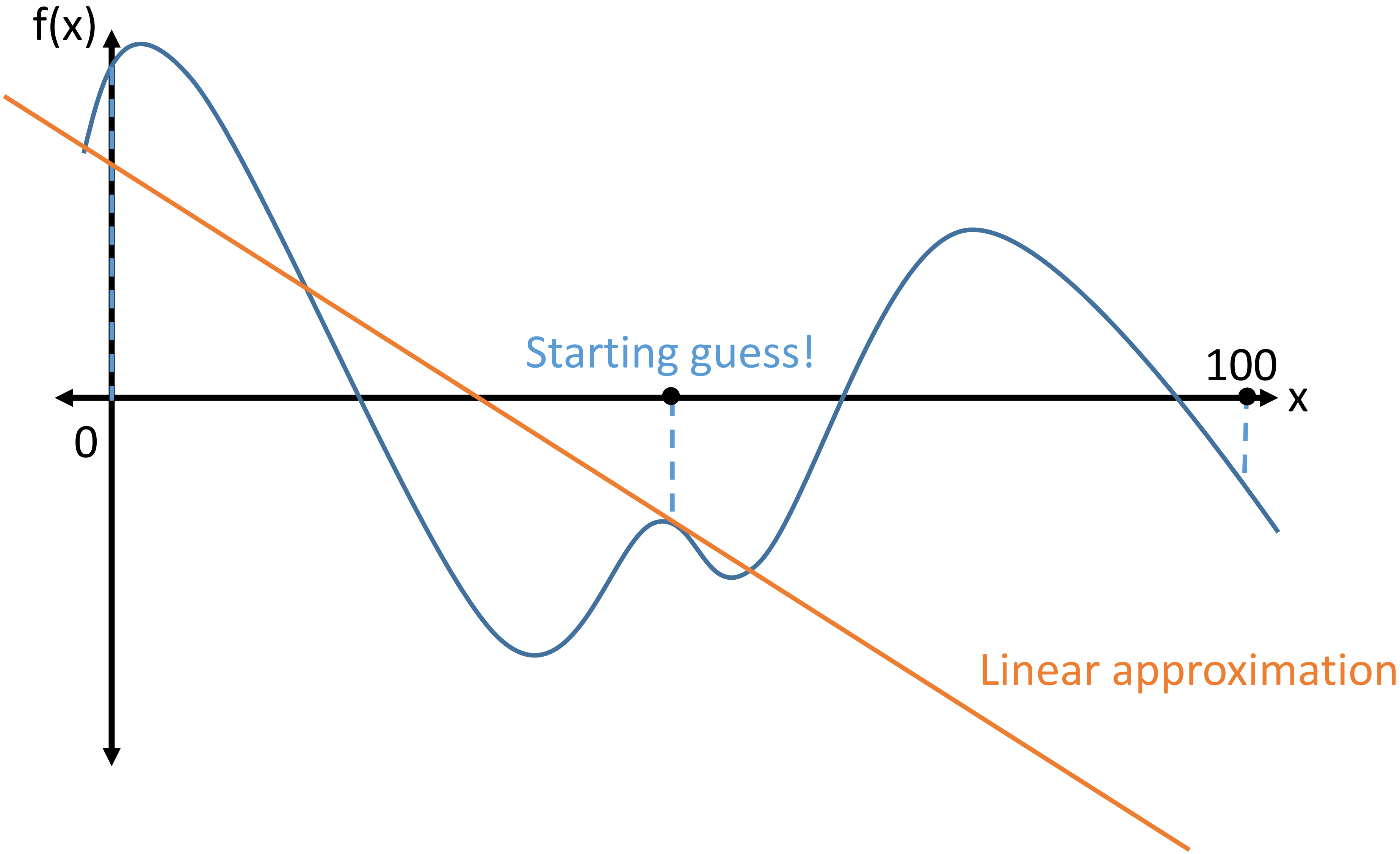
Newton's Method!



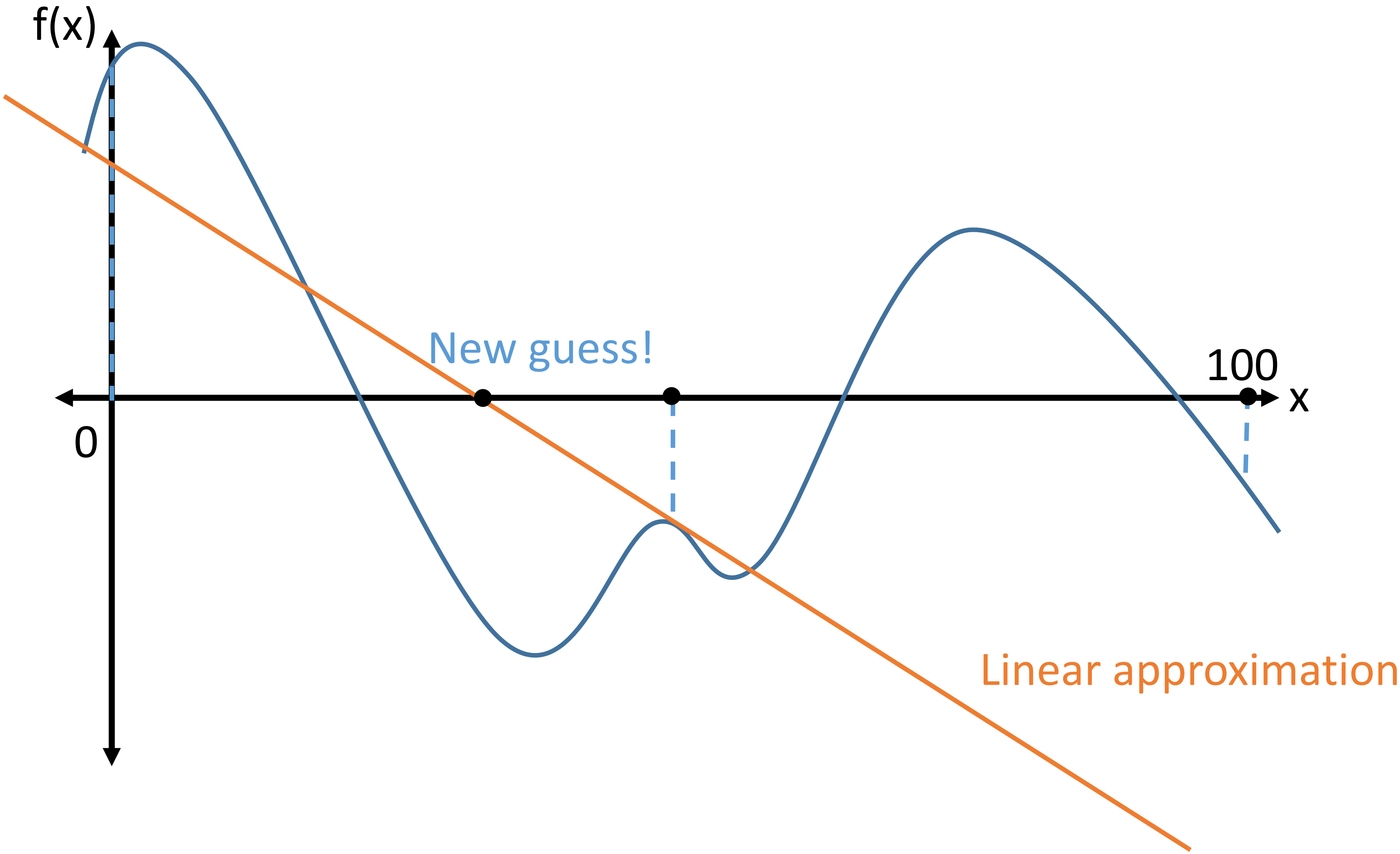
Newton's Method!



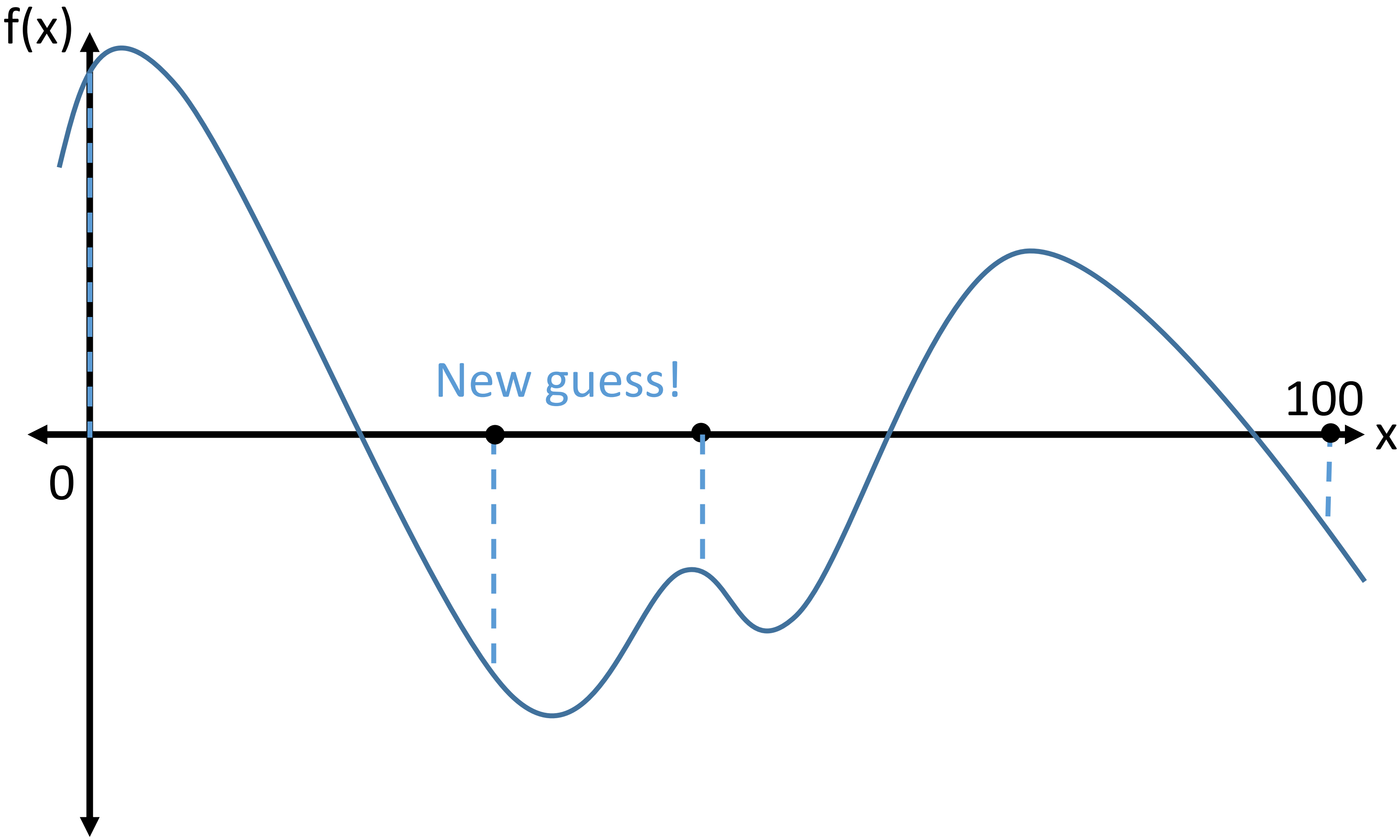
Newton's Method!



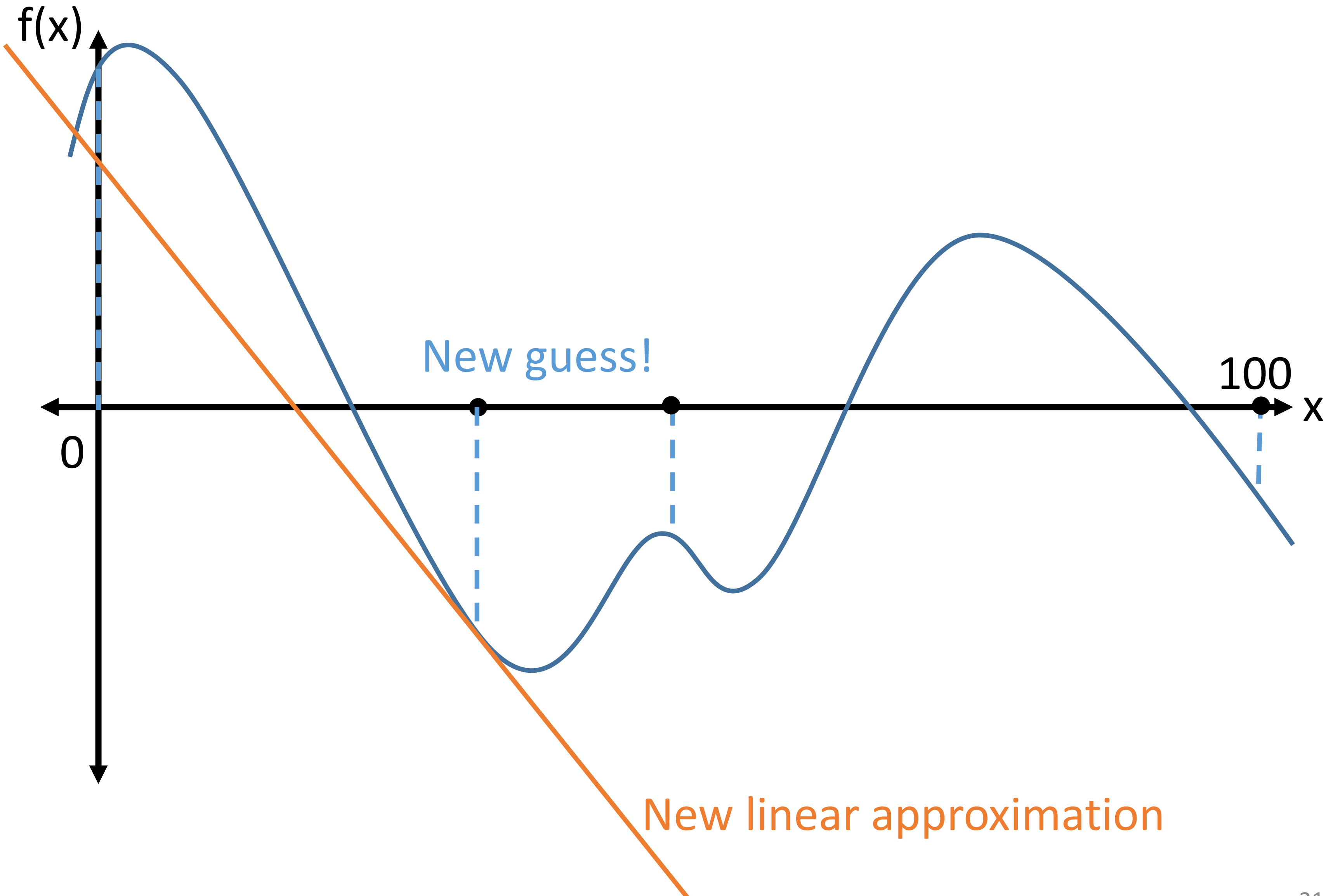
Newton's Method!



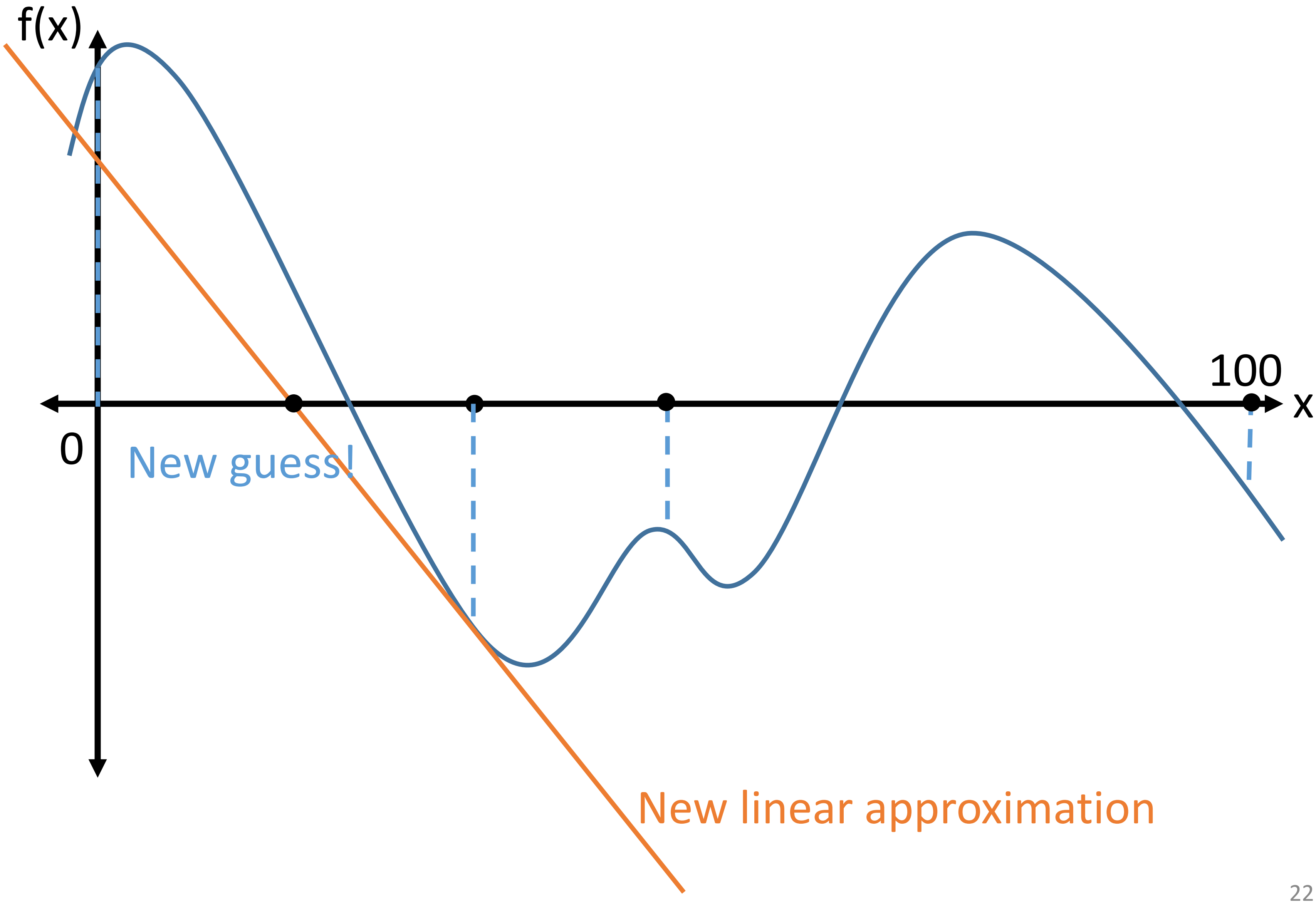
Newton's Method!



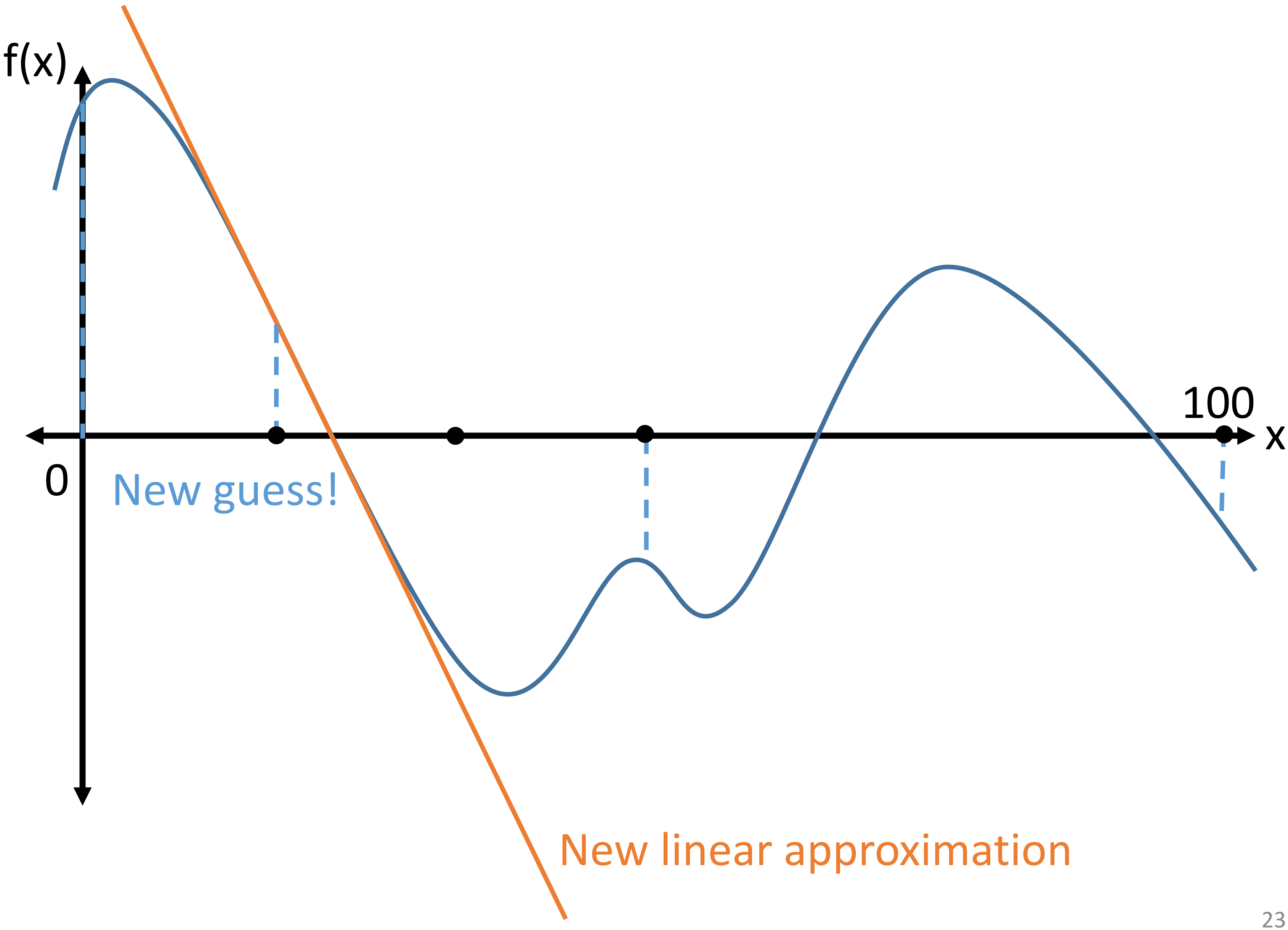
Newton’s Method!



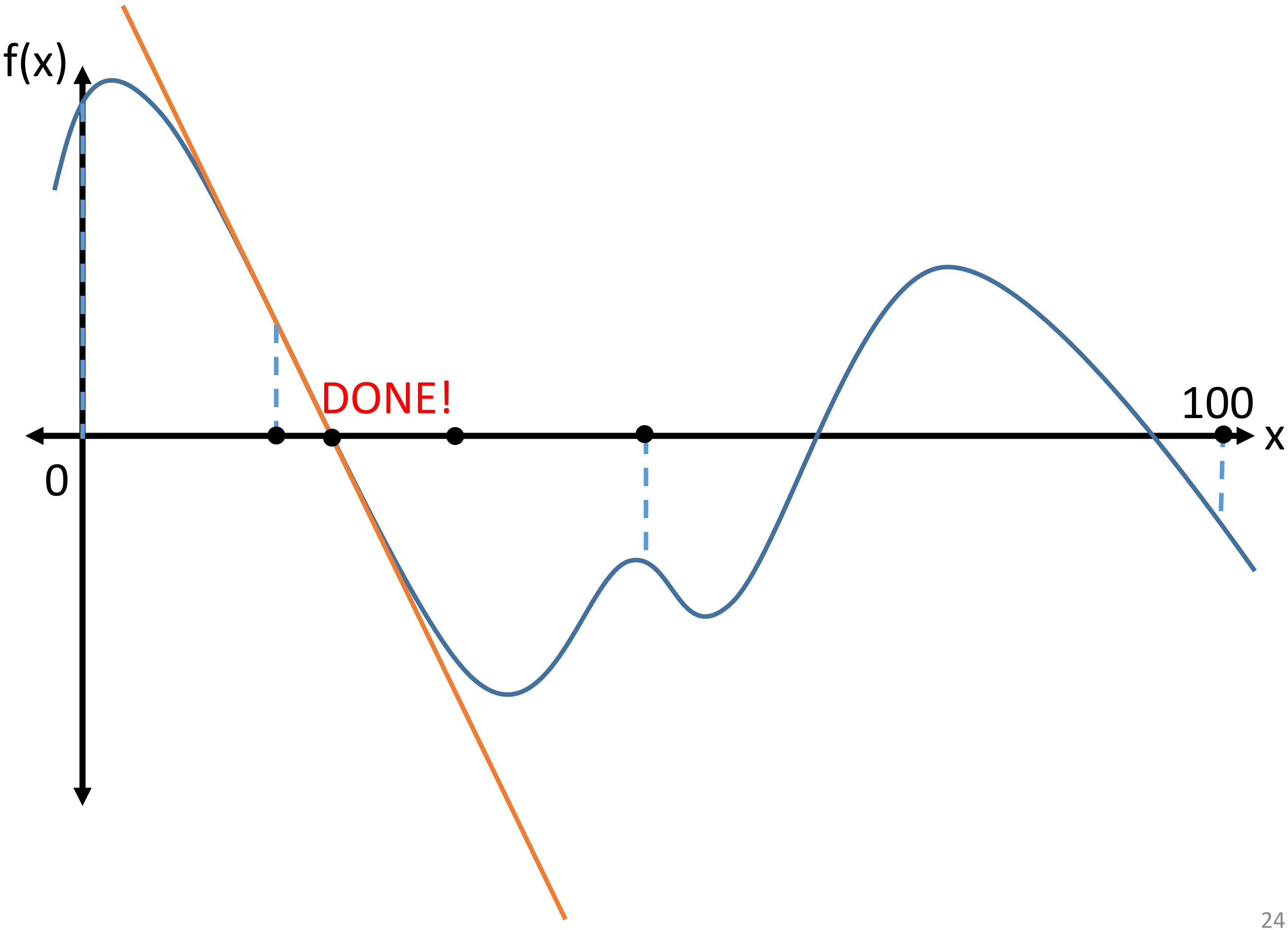
Newton's Method!



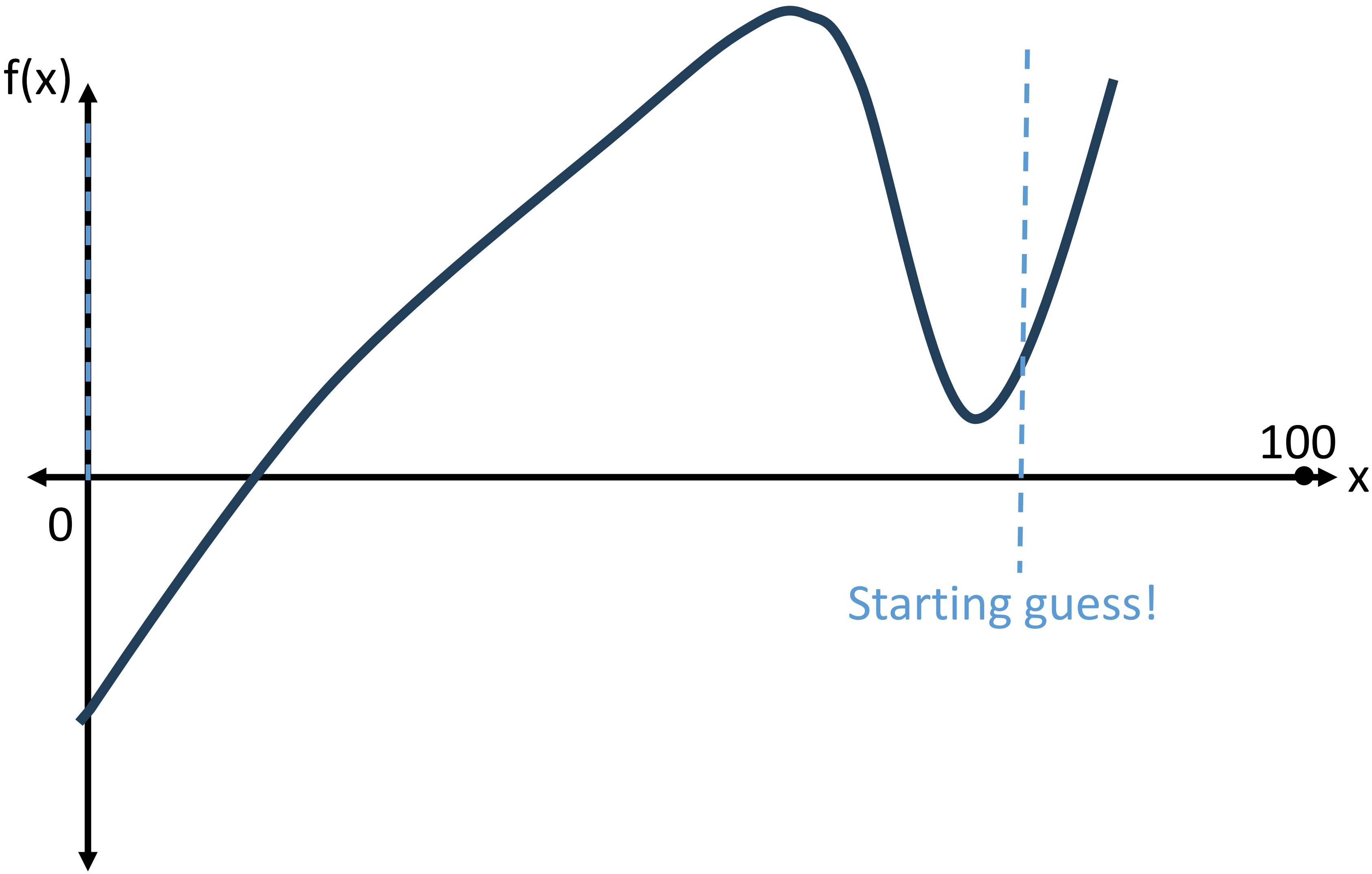
Newton's Method!



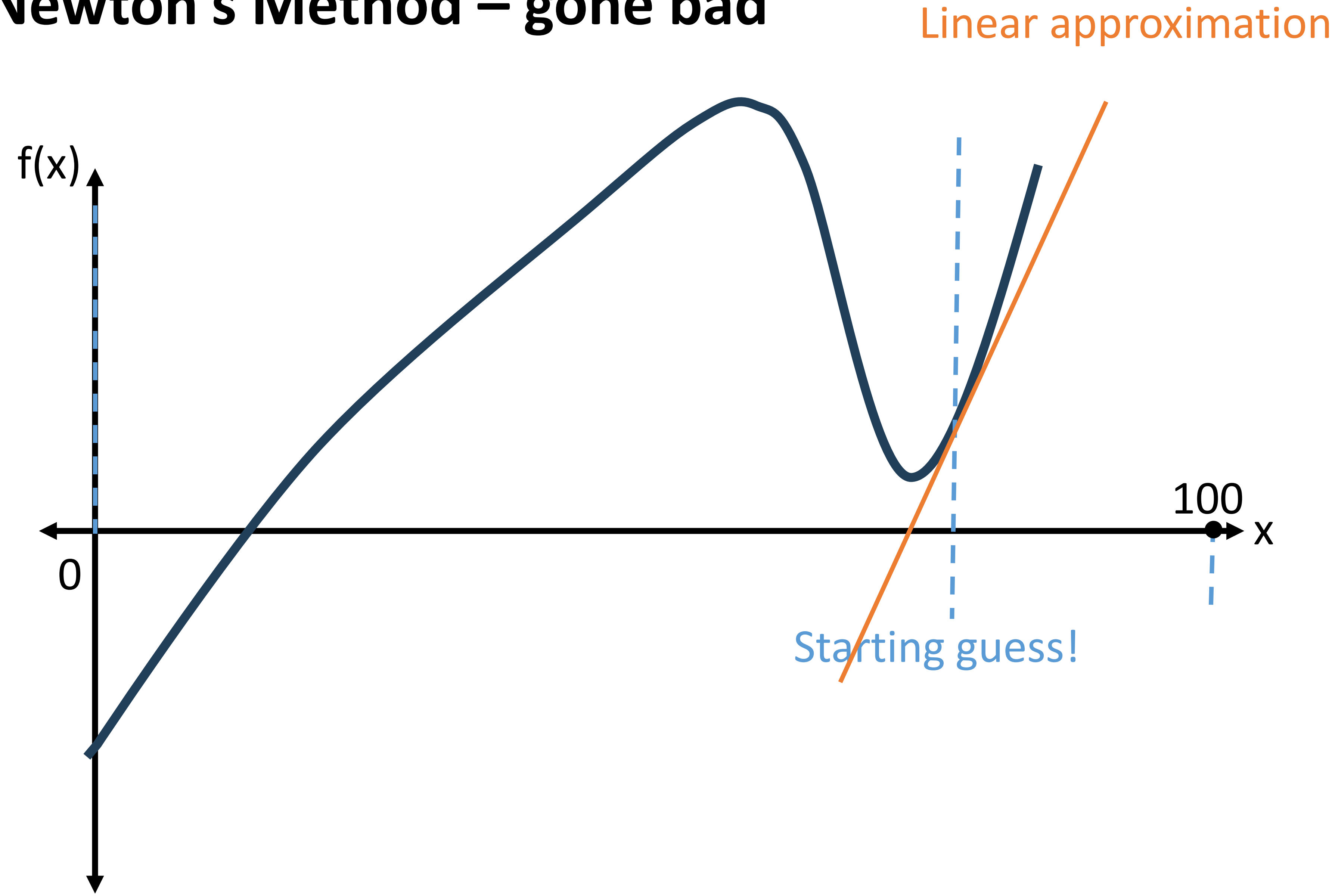
Newton’s Method!



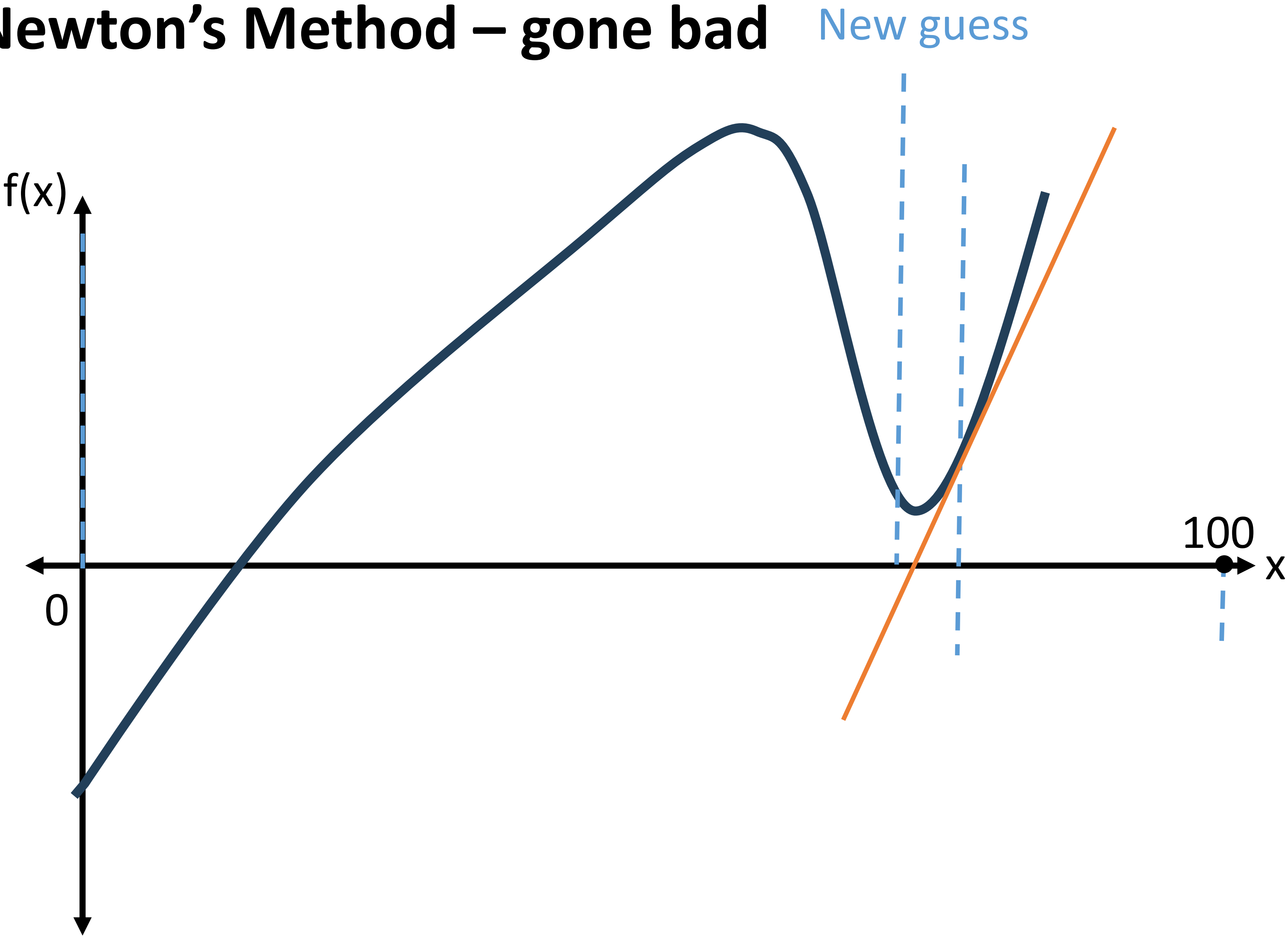
Newton's Method – gone bad



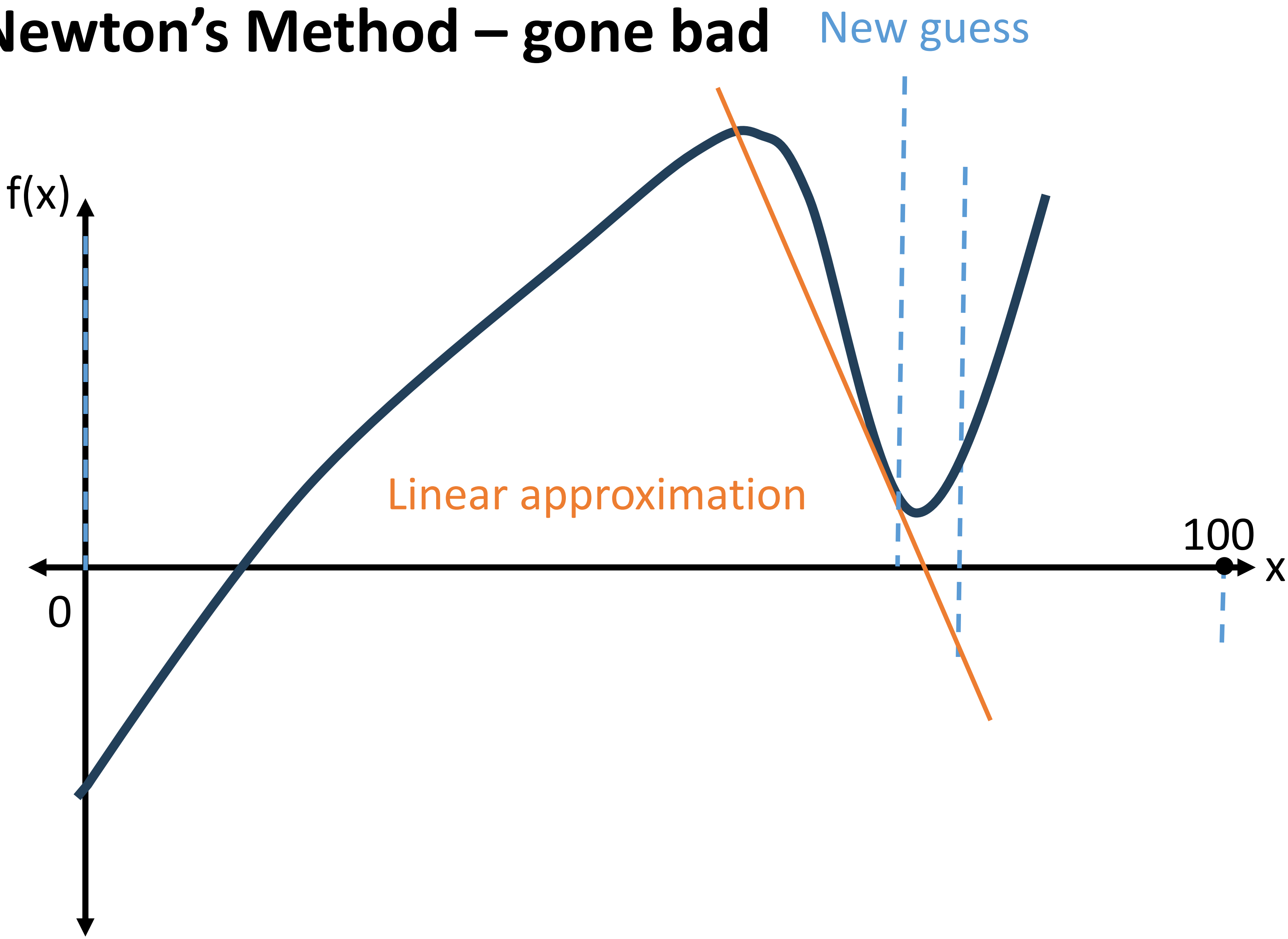
Newton's Method – gone bad



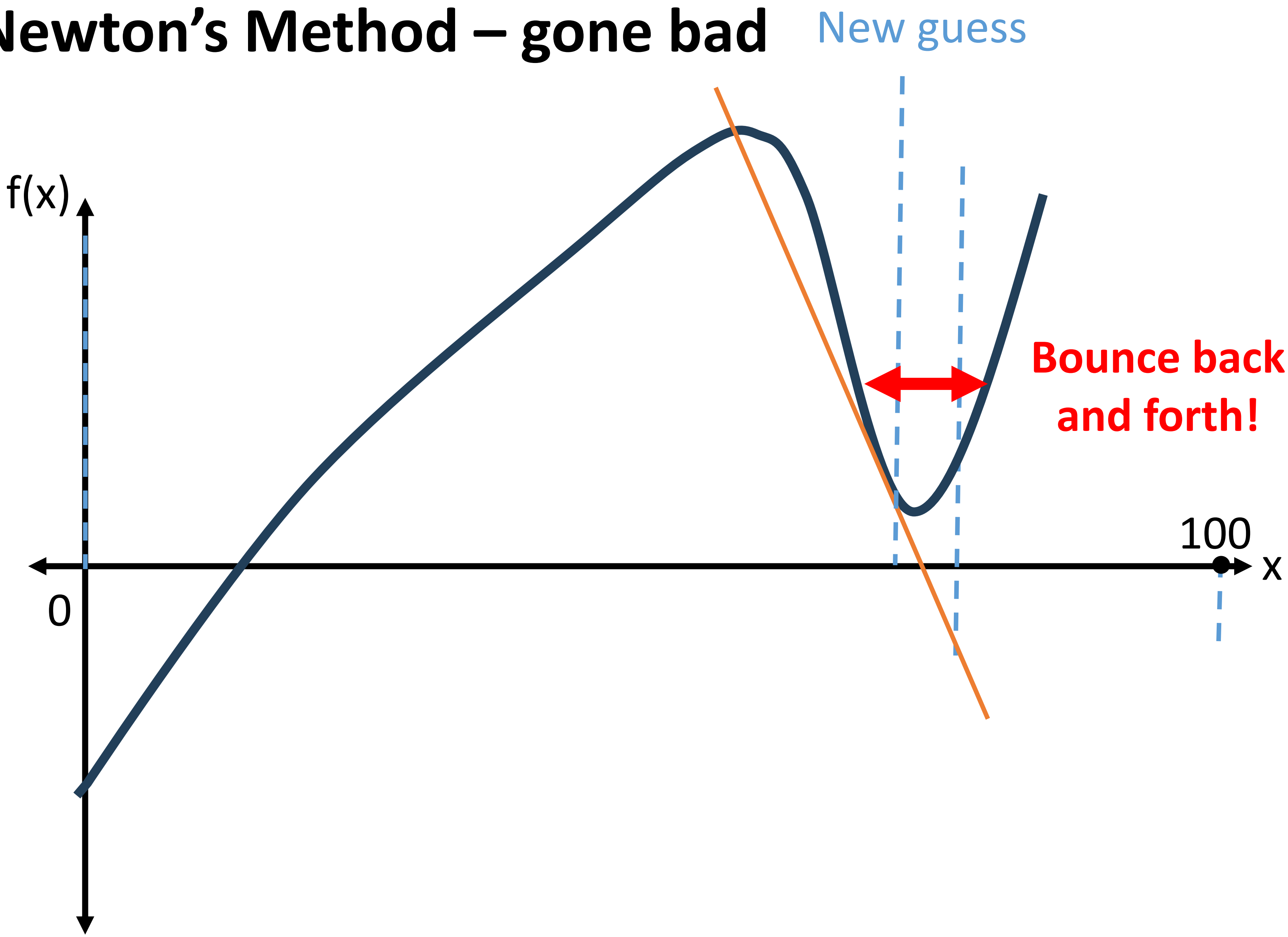
Newton's Method – gone bad



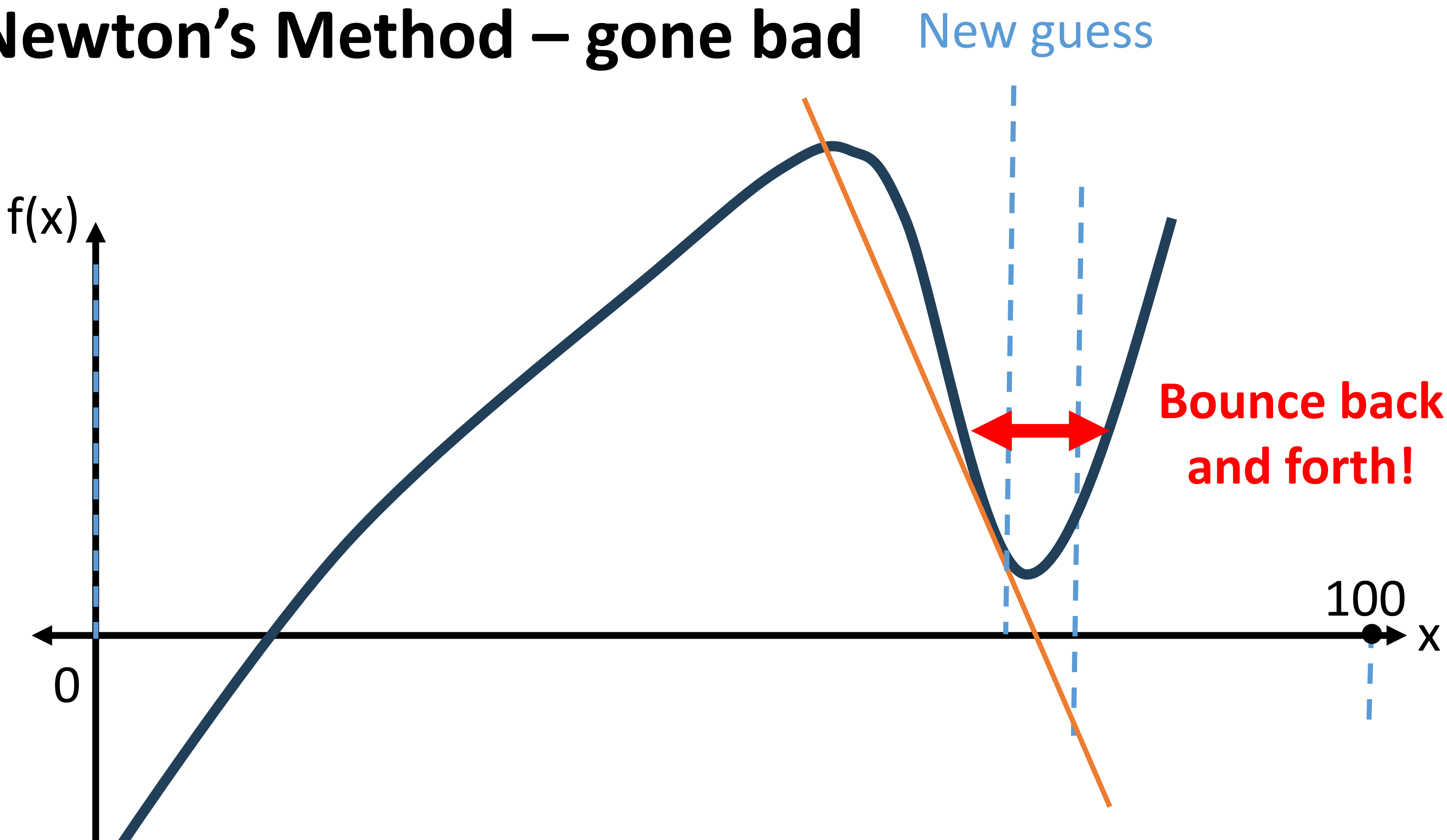
Newton's Method – gone bad



Newton's Method – gone bad



Newton’s Method – gone bad



Method only guaranteed to work for a good enough initial guess

Translating into code

Newton's method in pseudocode

Input

- A function f(x) and its derivative f'(x)
- Initial value, x_0 that is close to a zero crossing

```
x = x₀ # initial guess
while (guess is still far)
    x = x - (f(x) / f'(x)) # update estimate for x
```

Termination

Note that we need to tell our program when to stop.

This is typically done by terminating when:

- the function is “almost zero” or “near some value”

```
while (|f(x)| > tol):
    # iterate
```

- the error in independent variable is “almost zero”

```
while (|x-x*| > tol):
    # iterate
```

This tolerance will depend upon the function you are working with and how much info you have.

Newton's method in pseudocode

Input

- A function f(x) and its derivative f'(x)
- Initial value, x_0 that is close to a zero crossing

```
x = x0 # initial guess
while (|f(x)| > tolerance)
    x = x – (f(x) / f'(x)) # update estimate for x
```

Newton’s method in pseudocode

Input

- A function f(x) and its derivative f’(x)
- Initial value, x_0 that is close to a zero crossing

This part for safety

```
x = x0 # initial guess
while (|f(x)| > tolerance and n_iter < max_iter)
   x = x – (f(x) / f’(x)) # update estimate for x
```

Newton's Method

Pros

- Converges quickly
- Only requires one initial guess of root location

Cons

- Requires function and derivative
- Can diverge from the solution in some cases

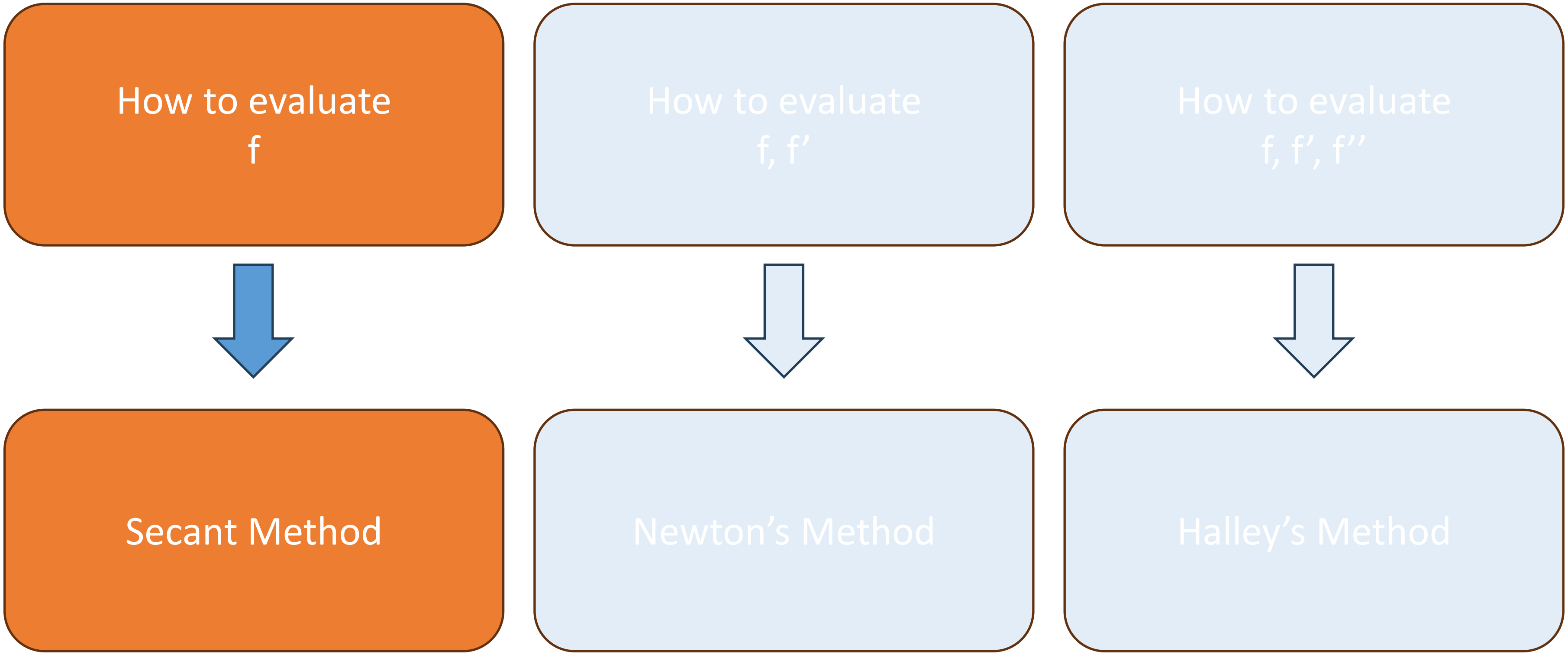
Picking a gradient based optimizer

What do we know? What are examples of sensors that pick up some of these but not others?

How to evaluate f	How to evaluate f, f’	How to evaluate f, f’, f’’

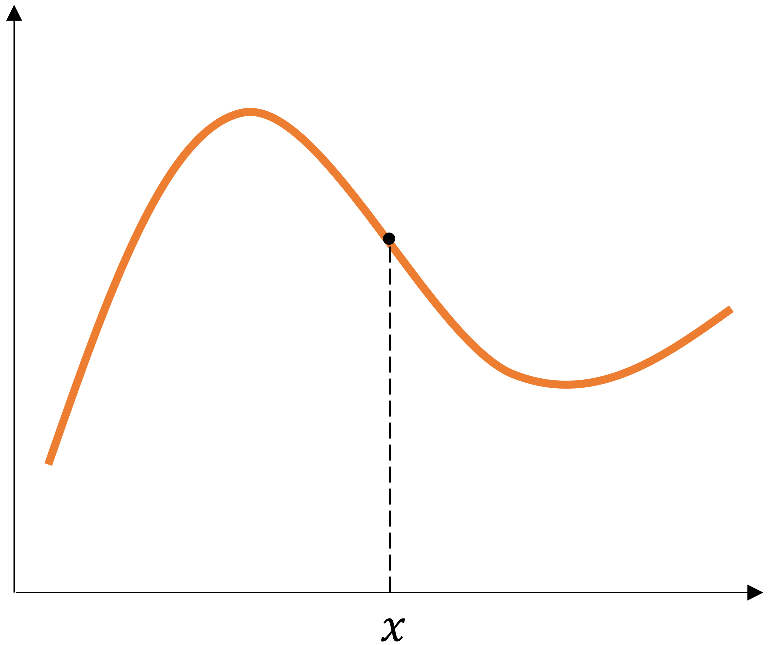
Other methods

What do we know? What are examples of sensors that pick up some of these but not others?

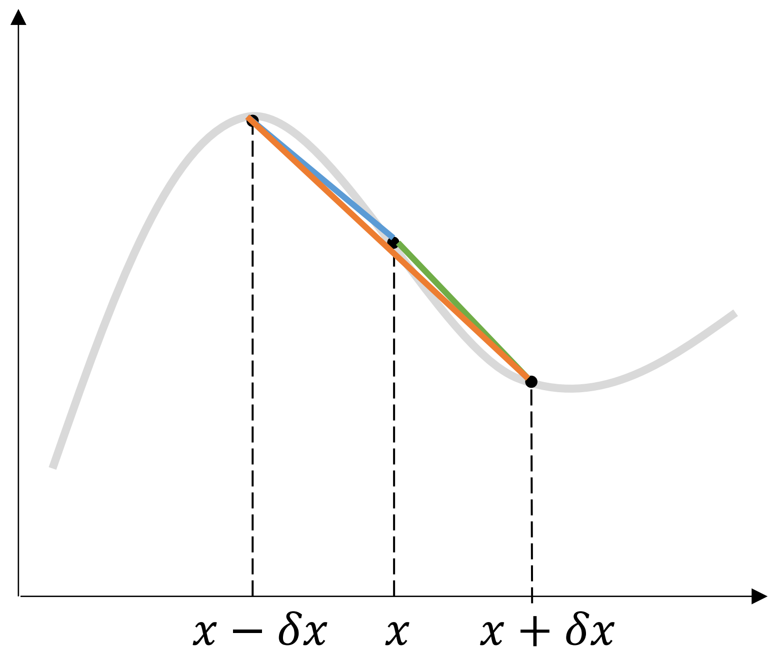


What if we don’t know the derivative?

What is the slope at x?



What if we don't know the derivative?



What is the slope at x?

Forward difference

$$\frac{f(x+\delta x) - f(x)}{\delta x}$$

Backward difference

$$\frac{f(x) - f(x-\delta x)}{\delta x}$$

Central difference

$$\frac{f(x+\delta x) - f(x-\delta x)}{2\,\delta x}$$

What if we don't know the 2nd derivative?

Forward difference

$$\frac{f(x+\delta x) - f(x)}{\delta x}$$

Backward difference

$$\frac{f(x) - f(x-\delta x)}{\delta x}$$

Central difference

$$\frac{f(x+\delta x) - f(x-\delta x)}{2\,\delta x}$$

Use the forward and backward difference formula to approximate the central difference approximation of f''

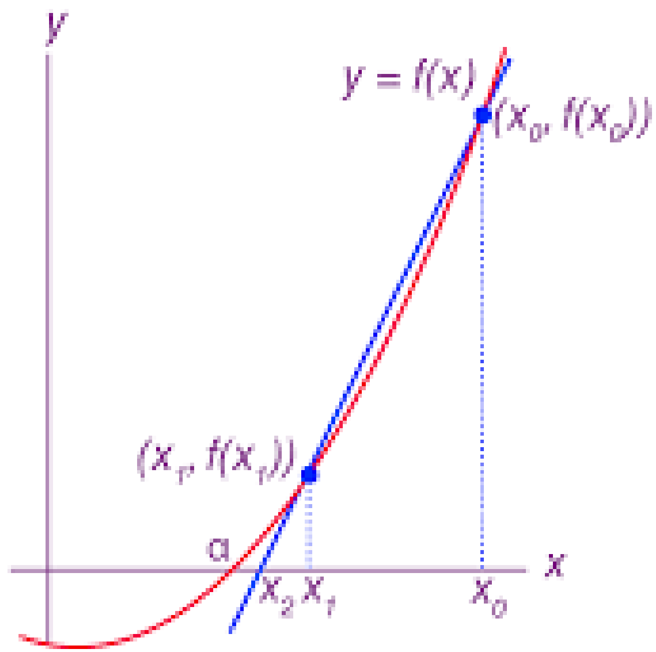
$f''(x)$ =

$$\frac{f'(x+\delta x) - f'(x-\delta x)}{2\,\delta x} =$$

$$\frac{\frac{f(x+\delta x) - f(x)}{dx} - \frac{f(x) - f(x-\delta x)}{\delta x}}{\delta x} = \frac{f(x+\delta x) - 2f(x) + f(x-\delta x)}{(\delta x)^2}$$

Secant Method

Newton's method for root finding uses the derivative of the function. If we can't have that, we use a finite difference to approximate it instead.



What is the equation of the red line, call it $f(x)$?

Taylor series approximation $\quad f(x) \approx f(x_n) + f'(x_n)(x - x_n)$

At what x is $f(x) = 0$? $\quad 0 = f(x_n) + f'(x_n)(x - x_n)$

This is the new guess x_{n+1}

$$x = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$$

Basic Idea: x_{n+1} is closer to the root than x_n

Other methods

What do we know? What are examples of sensors that pick up some of these but not others?

How to evaluate f	How to evaluate f, f’	How to evaluate f, f’, f’’
↓	↓	↓
Secant Method	Newton’s Method	Halley’s Method

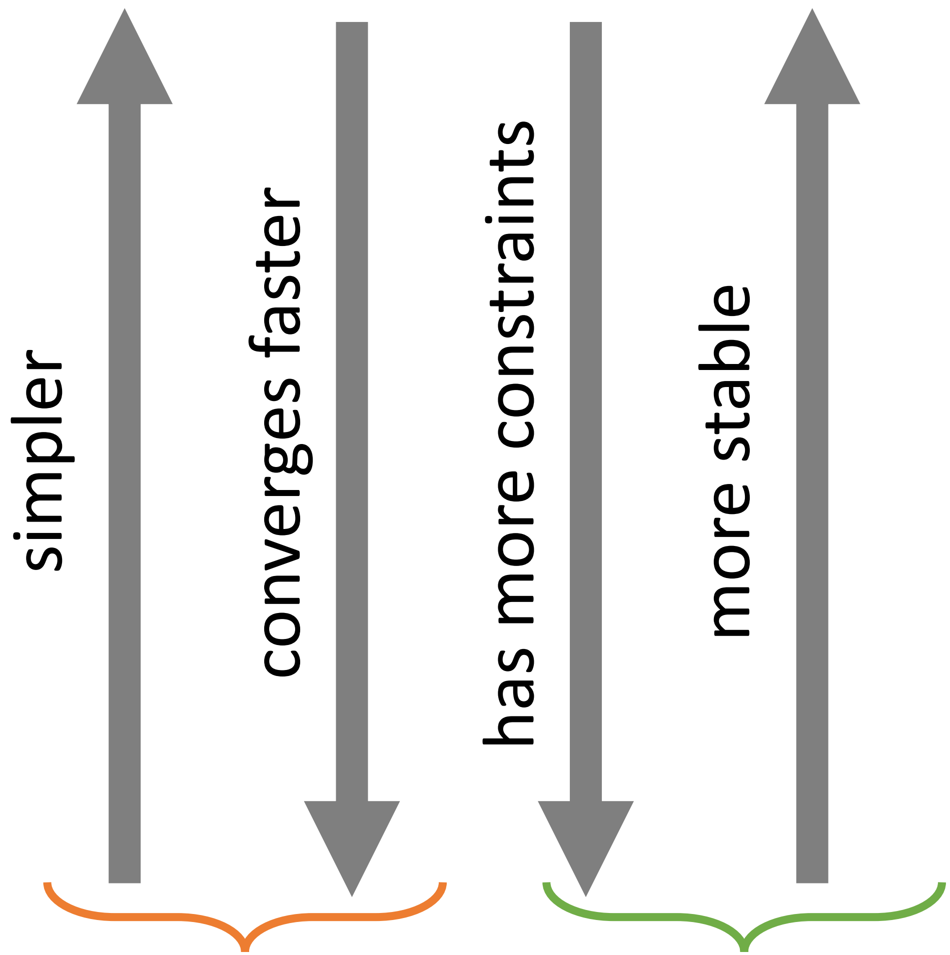
Higher Order Methods (Halley's Method)

Consider the function

$$g(x) = \frac{f(x)}{\sqrt{f'(x)}}$$

1. Any root of $f(x)$ where $f'(x) \neq 0$ is a root of $g(x)$
2. Any root of $g(x)$ is a root of $f(x)$ as long as $f'(x) \neq \pm\infty$ at that x
3. $g'(x) = \sqrt{f'(x)} - \frac{f(x)f''(x)}{2\left(f'(x)\right)^{3/2}}$
4. Use Newton's method on g

Choosing a Solution Method



Bisection

Newton

Halley

Total computation time is a combination of # iterations and computation per iteration

Convergence depends on how well-behaved your function is

In-Class 06: Nonlinear solvers

Do this with a partner.

Turn in as a pair on Canvas.

Tips for pair programming:

- Switch off who is typing.
- The person who is not typing should:
  - Make comments or suggest potential solutions
  - Be "devil's advocate": what are potential issues with what is being typed
  - Suggest other things to explore

At-Home: Complete in-class **(first real computing will be harder!)**