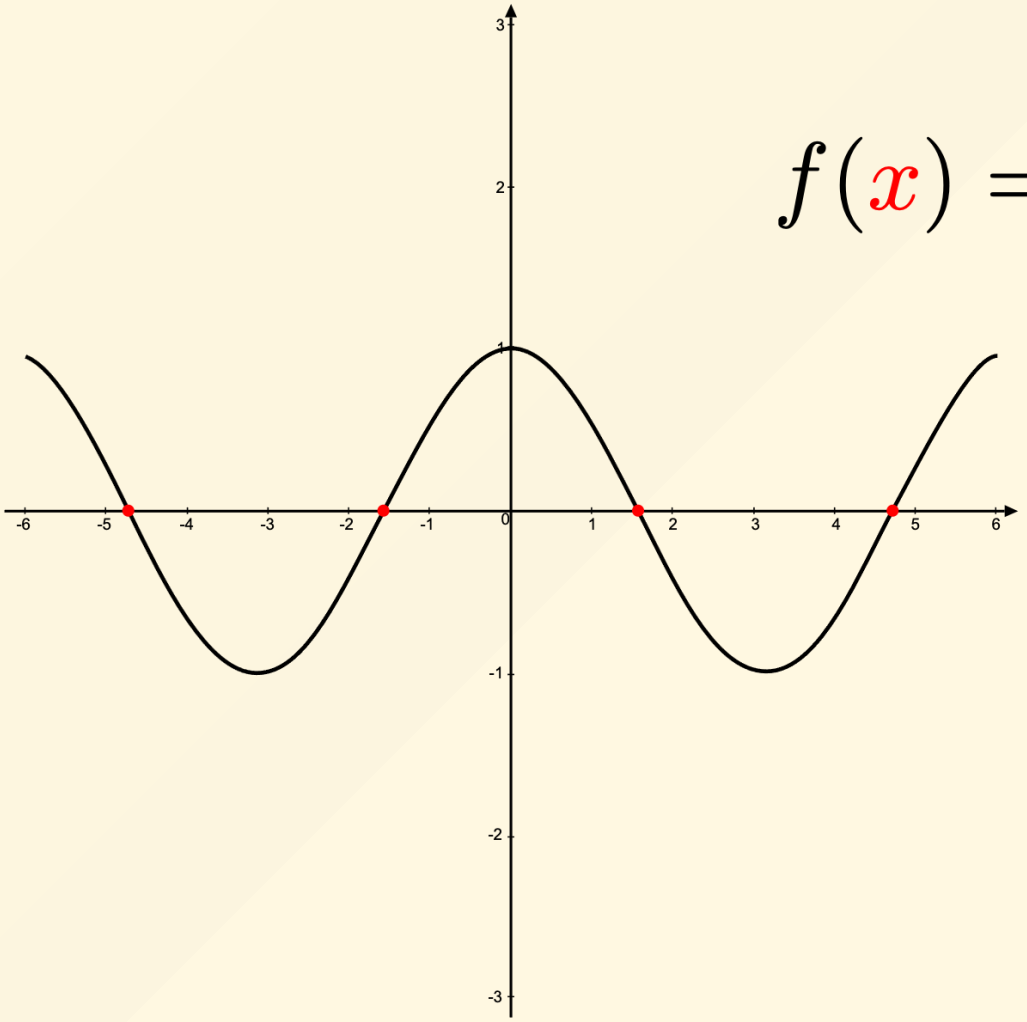
Design

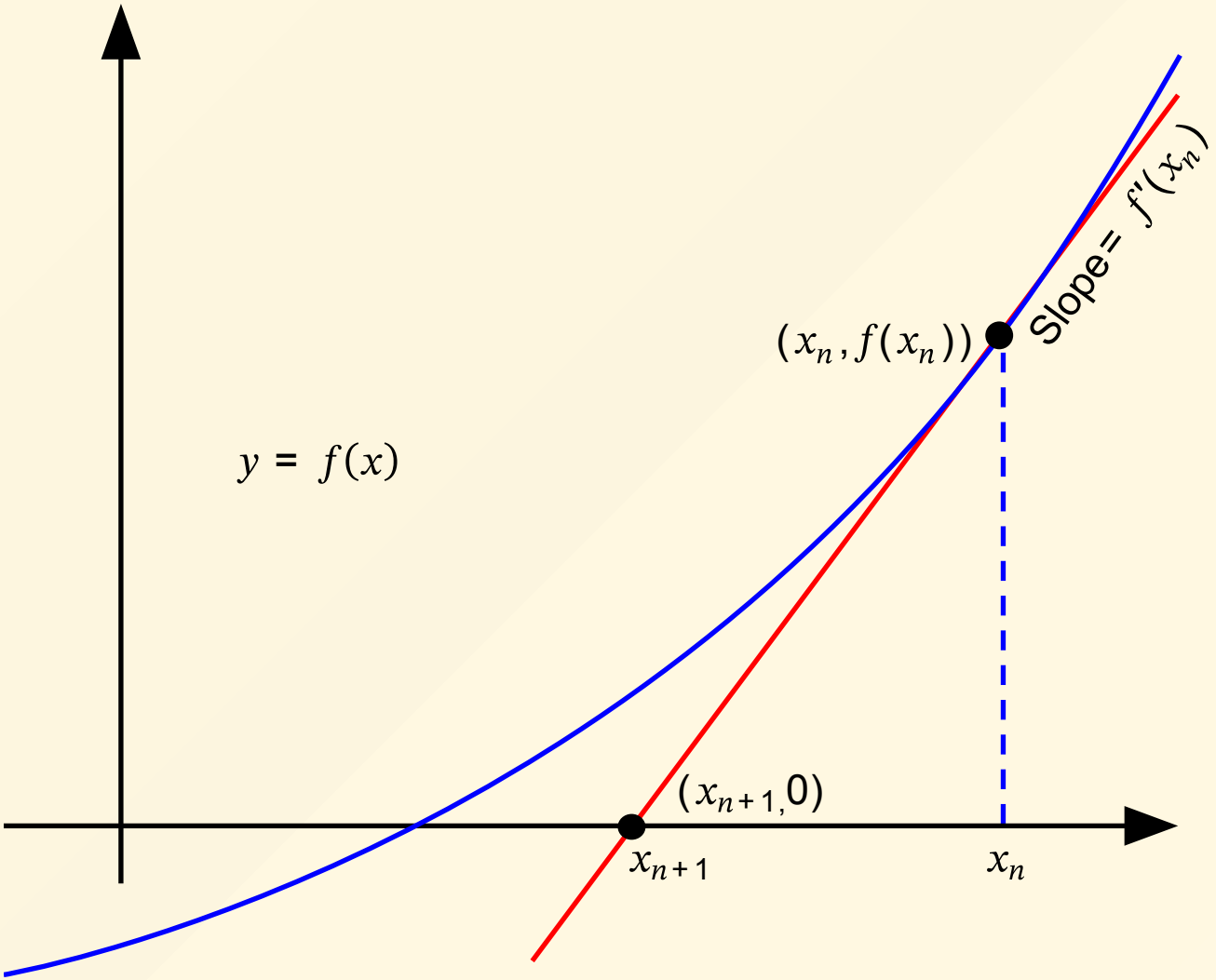
Object-Oriented Programming with C++

Root-finding algorithm

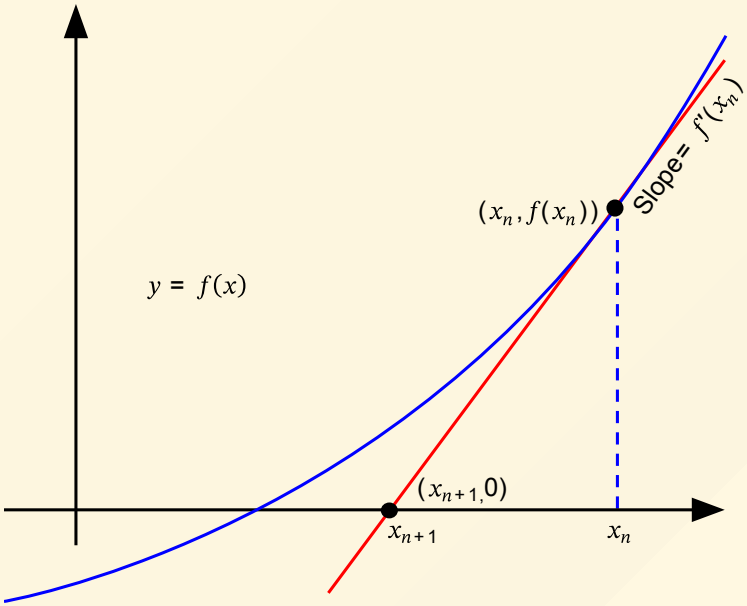


$$f(x) = 0$$

Newton's method



Newton's method



The slope is:

$$f'(x_n) = \frac{f(x_n) - 0}{x_n - x_{n+1}}.$$

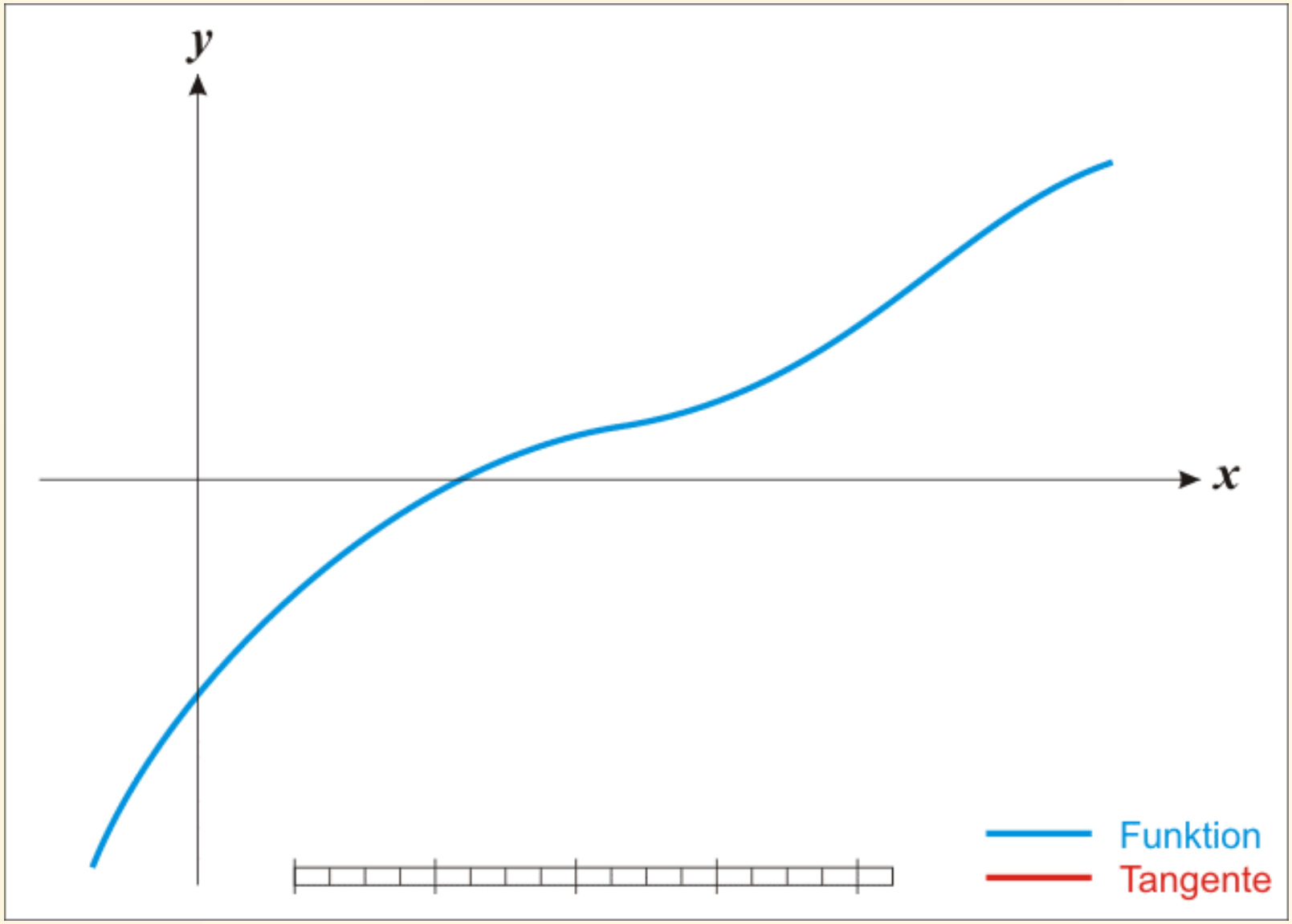
Solving for x_{n+1} gives:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Newton's method

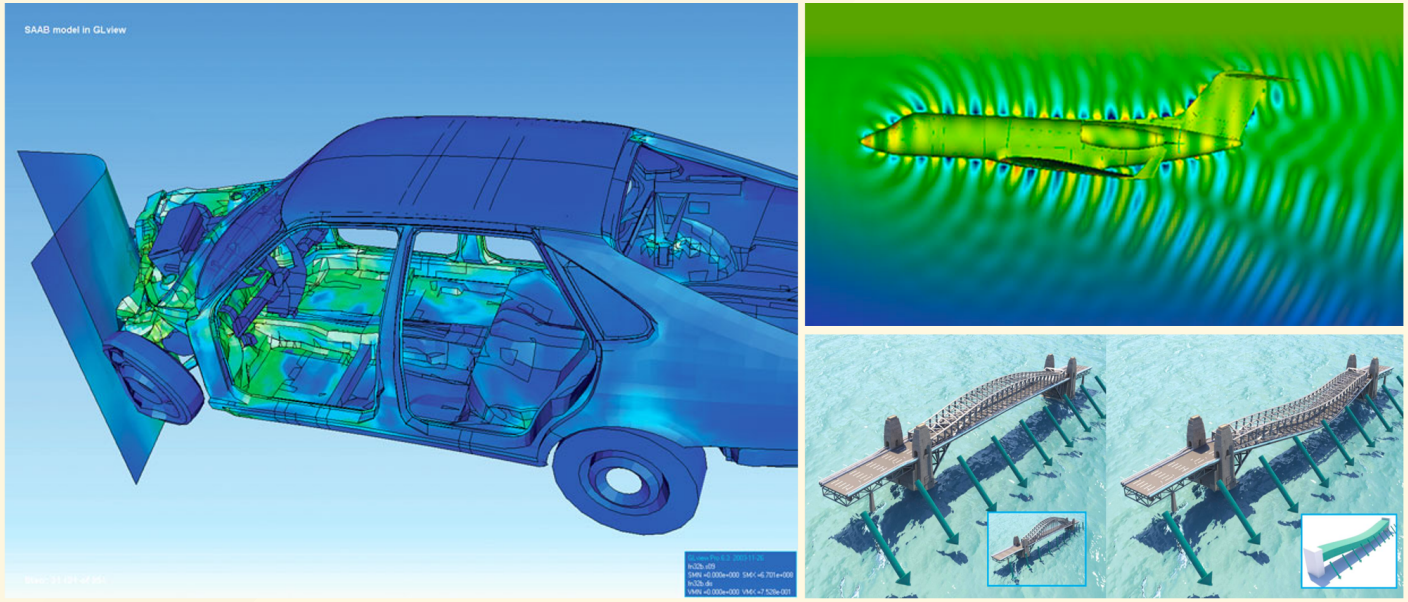
- Just a few *iterations*.

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


y
x
Funktion
Tangente

Newton's method

- Very fast: quadratic convergence
- Scientific computing



Newton's method

- Fast inverse square root

John Carmack

Quake III Arena (1999)

Designing classes

- Write classes in a way that they are
  - understandable
  - maintainable
  - reusable

Class design: what to do?

- How many types of class do we need?
- When to define a class?
- What kind of interface/data in a class?
- Shall we construct inheritance to promote interface and code reuse?
- Which function should be virtual to support dynamic binding in run-time?

Code quality

- Two important concepts for quality of code:
  - Coupling
  - Cohesion

Cohesion

- Cohesion refers to the number and diversity of tasks that a single unit is responsible for.
- If each unit is responsible for one single logical task, we say it has high cohesion.
- Cohesion applies to classes and methods.
- We aim for high cohesion.

High cohesion

- High cohesion makes it easier to:
  - Understand what a class or method does
  - Use descriptive names
  - Reuse classes or methods

Cohesion of methods/classes

- A method should be responsible for one and only one well-defined task.
- A class should represent one single, well-defined entity.

Coupling

> *If X changes, how much code in Y must be changed?*

- Coupling refers to links between separate units of a program.
- If two classes depend closely on many details of each other, we say they are tightly coupled.
- We aim for loose coupling.

Loose coupling

- Loose coupling makes it possible to:
  - Understand one class without reading others
  - Change one class without affecting others
  - Improves maintainability

Techniques to loose

- call-back
- message mechanism
- interface abstraction
- ...

Code duplication

- Code duplication:
  - is an indicator of bad design
  - makes maintenance harder
  - leads to severe errors

Software changes

- Software is not like a novel that is written once and then remains unchanged.
- Software is *extended*, *corrected*, *maintained*, *ported*, *adapted*, ...
- The work is done by different people over time (often decades).

Change or die

- There are only two options for software:
  - continuously maintained
  - dies
- Software that cannot be maintained will be thrown away.

Thinking ahead

- When designing a class, we try to think what changes are likely to be made in the future.
- We aim to make those changes easy.

Design guidelines

- A method is too long if it does more then one logical task.
- A class is too complex if it represents more than one logical entity.
- Note: these are guidelines – they still leave much open to the designer.

SOLID principles

- In OO programming, the term SOLID represents five design principles intended to make software designs more *understandable*, *flexible* and *maintainable*.

SOLID principles

- **S**ingle responsibility principle
- **O**pen/closed principle

> “ software entities … should be open for extension, but closed for modification. ”

SOLID principles

- **L**iskov substitution principle

  > “ objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program. ”

SOLID principles

- **I**nterface segregation principle
- **D**ependency inversion principle

> “ one should "depend upon abstractions, [not] concretions." ”

Review

- Programs are continuously changed.
- It is important to make this change possible.
- Quality of code requires much more than just performing correct at one time.
- Code must be readily understandable and maintainable.

Review

- Good quality code avoids duplication, displays high cohesion, low coupling.
- Coding style (commenting, naming, layout, etc.) is also important.
- There is a big difference in the amount of work required to change poorly-structured and well-structured code.