Introduction to Numerical Computing

Foundations of numerical methods

NC 01	Why computers use numerical approximations
Core idea	Transform a mathematical problem into a sequence of computable operations
Today	Models, algorithms, accuracy, efficiency and a first worked example

Learning outcomes

By the end of this lecture, students should be able to...

01	Define numerical computing and distinguish exact from approximate solutions.
02	Identify common classes of numerical problems and suitable methods.
03	Explain error, tolerance, convergence and computational cost.
04	Trace a simple numerical algorithm and interpret its result.

What is numerical computing?

Mathematics + algorithms + finite-precision computation

Mathematical problem	Find a number, function or model satisfying given conditions.
Numerical method	A systematic procedure for obtaining an approximate numerical solution.
Numerical computing	Implement the procedure on a computer and assess the reliability of its result.

Exact versus numerical solutions

Two complementary approaches

Analytical solution	Uses symbolic manipulation to obtain an exact expression. Example: $x^2 = 2$ gives $x = \pm\sqrt{2}$.
Numerical solution	Produces a value such as the positive root $x \approx 1.414214$.
Why approximate?	Some equations have no simple closed-form solution; even exact expressions often require numerical evaluation.

Why do we need numerical methods?

Problems grow beyond convenient hand calculations

Nonlinear models	Equations such as $\cos(x) - x = 0$ are not easily solved by elementary algebra.
Large systems	Thousands of simultaneous equations arise in engineering and scientific simulation.
Measured data	Observations may be discrete, noisy or incomplete; interpolation and fitting help extract information.

The numerical computing workflow

A repeatable process from question to result

1. Model	State the mathematical problem, assumptions and input data.
2. Choose & implement	Select a method, define a stopping rule and translate it into an algorithm.
3. Compute & validate	Run the method, inspect residuals and errors, and check whether the result is meaningful.
4. Refine	Change the step size, precision or method if the accuracy is insufficient.

Major classes of numerical problems

The roadmap for this course

Equations	Find roots of f(x) = 0 or solve a system Ax = b.
Data approximation	Estimate values between observations or fit a model to data.
Numerical calculus	Approximate derivatives, integrals and differential-equation solutions.
Advanced computation	Estimate eigenvalues and minimize objective functions.

Accuracy, precision and error

A numerical answer is useful only with an accuracy assessment

Absolute error	$E = \lvert x - \hat{x} \rvert$, where x is the exact value and $\hat{x}$ is its approximation.
Relative error	$E_r = \lvert x - \hat{x} \rvert / \lvert x \rvert$, provided $x \neq 0$. Percentage error = $100\,E_r$.
Precision versus accuracy	Precision concerns representation or repeatability; accuracy concerns closeness to the true value.

Worked example: approximating √2

Compare 1.41 with the reference value 1.414213562...

Absolute error	$\lvert 1.414213562... - 1.41 \rvert \approx 0.004213562$
Relative error	$0.004213562... / 1.414213562... \approx 0.00297944$
Percentage error	$100 \times 0.00297944 \approx 0.297944\%$.
Interpretation	1.41 is close to √2, but whether it is sufficient depends on the application tolerance.

A first iterative method

Compute the positive square root of a positive number a

Starting point	Choose $a > 0$ and an initial estimate $x_0 > 0$.
Update rule	$x_{k+1} = (x_k + a / x_k) / 2$
Stopping rule	Stop when $\lvert x_{k+1} - x_k \rvert < \varepsilon$, or when the iteration limit is reached.
Note	This is Newton’s method applied to $x^2 - a = 0$; its derivation comes later.

Trace the iteration: a = 2

Start at $x_0 = 1$; tolerance $\varepsilon = 0.000001$

k = 1	$x_1 = (1 + 2/1)/2 = 1.500000000$
k = 2	$x_2 = (1.5 + 2/1.5)/2 \approx 1.416666667$
k = 3	$x_3 \approx 1.414215686$; change ≈ 0.002450980
k = 4, 5	$x_4 \approx 1.414213562$; $x_5 \approx 1.414213562$. Step 5 meets the tolerance.

Algorithm: square root

Pseudocode with a finite iteration limit

Initialize	Input $a > 0$, $x > 0$, $\varepsilon > 0$ and maximum iterations N.
Repeat up to N times	$y \leftarrow (x + a/x)/2$; if $\lvert y - x \rvert < \varepsilon$, return y with convergence flag.
Update	$x \leftarrow y$; continue until stopping condition or iteration limit.
If limit reached	Return latest estimate with a warning; check residual $\lvert x^2 - a \rvert$.

Python implementation

A compact implementation of the same algorithm

```
def sqrt_iter(a, x=1.0, tol=1e-6, max_iter=100):
    if a <= 0 or x <= 0 or tol <= 0 or max_iter < 1:
        raise ValueError("positive inputs required")
```

```
    for k in range(max_iter):
        y = (x + a/x) / 2
        if abs(y-x) < tol:
            return y, True
        x = y
```

```
    return x, False
# sqrt_iter(2) → (1.414213562373095, True)
```

How do we judge a numerical method?

Accuracy alone is not enough

Convergence	Do successive approximations approach the desired solution?
Stability	How sensitive is the algorithm to perturbations and rounding errors?
Efficiency	How many operations, iterations and how much memory are required?
Validation	Check a residual, compare with a known answer, or repeat with tighter tolerance.

Where numerical computing is used

Examples across disciplines

Engineering	Structural simulation, heat transfer and circuit analysis.
Computer science	Machine learning optimization, computer graphics and scientific computing.
Science and economics	Population models, weather prediction and estimation from data.

Practice questions

Test your understanding before the next lecture

Q1 Distinguish an analytical solution from a numerical solution.

Q2 Approximate π by 3.14. Calculate absolute and percentage errors.

Q3 For $a = 9$ and $x_0 = 2$, compute the first two square-root iterations.

Q4 Why should an iterative algorithm include a maximum iteration limit?

Solutions and recap

Key takeaways from NC 01

A1	An analytical solution is symbolic or exact; a numerical solution is a computed approximation.
A2	Absolute error ≈ 0.00159265; percentage error ≈ 0.0507%.
A3	$x_1 = 3.25$; $x_2 = (3.25 + 9/3.25)/2 \approx 3.009615385$.
A4	It prevents an endless loop if the tolerance is not met; the algorithm must report failure to converge.

Next lecture: floating-point arithmetic

NC 02 — Number representation and floating-point arithmetic

Remember	Model → method → computation → validation.
Coming next	Binary representation, finite precision, rounding and floating-point limitations.
Suggested reading	Burden, Faires & Burden, Numerical Analysis; Chapra & Canale, Numerical Methods for Engineers.