MA633L-Numerical Analysis

Lecture 44 : Final Conclusions

Panchatcharam Mariappan[1]

[1]Associate Professor
Department of Mathematics and Statistics
IIT Tirupati, Tirupati

April 23, 2026

भारतीय प्रौद्योगिकी संस्थान तिरुपति
IIT
TIRUPATI

A Bird's-eye View of the Course

Over the semester, we built six pillars of Numerical Analysis:

- **Module 1:** Preliminaries, Errors and Floating Point Arithmetic
- **Module 2:** Numerical Interpolation
- **Module 3:** Nonlinear Equations
- **Module 4:** Numerical Linear Algebra
- **Module 5:** Numerical Integration
- **Module 6:** Numerical Differential Equations

Today we revisit the *key conclusions* from each topic—what to remember, when to use, and what can go wrong.

Module 1: Errors & Floating Point Arithmetic

Asymptotic Notations and Master Theorem

- $\mathcal{O}, \Omega, \Theta$ bound algorithmic cost up to constants and for large n; o and ω are strict.
- **Master Theorem.** For $T(n) = aT(n/b) + f(n)$, compare $f(n)$ with $n^{\log_b a}$:
  - regularly-polynomially smaller $\Rightarrow T = \Theta(n^{\log_b a})$,
  - same order $\Rightarrow T = \Theta(n^{\log_b a} \log n)$,
  - polynomially larger (with regularity) $\Rightarrow T = \Theta(f(n))$.
- **Take-away:** Before coding a method, estimate its complexity—it tells you whether doubling n doubles, quadruples, or explodes the cost.

Order of Convergence

- A sequence $x_n \to x^*$ has order p and asymptotic constant C if

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^p} = C, \quad C > 0.$$

- $p = 1$: linear; $p = 2$: quadratic; $1 < p < 2$: superlinear.
- **Conclusion:** Order of convergence is the single most important descriptor of an iterative method—a quadratic method doubles correct digits every step.

Errors, Machine Epsilon, Floating Point

- Three error sources: **truncation** (method), **round-off** (machine), **data** (input).
- Machine epsilon ϵ_{mach} is the smallest $\epsilon > 0$ with $\mathrm{fl}(1+\epsilon) \neq 1$. For IEEE-754 double: $\epsilon_{mach} \approx 2.22 \times 10^{-16}$.
- Relative error of representation is bounded by $\epsilon_{mach}/2$.
- **Dangers:** subtractive cancellation, overflow/underflow, summing small to large.

Golden rule
Total error $\downarrow$ as $h \downarrow$ only up to a point—beyond it, round-off takes over. Choose h at the valley of the error curve.

Module 2: Numerical Interpolation

Newton's Forward / Backward / Divided Differences

- Newton form: $P_n(x) = \sum_{k=0}^{n} f[x_0, \ldots, x_k] \prod_{i=0}^{k-1} (x - x_i)$.
- **Forward** for equispaced nodes, interpolating near the top of the table; **backward** near the bottom; **divided differences** for arbitrary nodes.
- Adding a new data point costs only one new column in the difference table.

Conclusion

Newton's form is the **preferred practical** form: cheap updates, progressive-degree evaluation, and diagnostic use of higher differences to detect data errors.

Lagrange Interpolation

- $P_n(x) = \sum_{k=0}^{n} f(x_k)\, L_k(x)$, with $L_k(x) = \prod_{j \neq k} \dfrac{x - x_j}{x_k - x_j}$.
- Same unique polynomial as Newton's—only the basis is different.
- Elegant for proofs and for formulas (quadrature weights, FE shape functions), but unstable to evaluate in the naive form. Use the barycentric variant in practice.

Error in Interpolation & Runge Phenomenon

- If $f \in C^{n+1}[a,b]$, then for some $\xi(x) \in (a,b)$,

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i).$$

- Raising the degree does not always reduce the error—**Runge's phenomenon** on $\frac{1}{1+25x^2}$ with equispaced nodes.
- **Remedy:** Chebyshev nodes minimise $\| \prod (x - x_i) \|_\infty$.

Osculating & Spline Interpolation

- **Hermite / osculating** polynomials match both f and its derivatives at nodes—useful when slopes are known.
- **Splines** use low-degree pieces (typically cubic) joined with continuity of S, S', S''.
- Cubic splines avoid Runge oscillations, have small interpolation error $\mathcal{O}(h^4)$ for smooth f, and are the de-facto tool for smooth curves through many points.

Take-away
For few points: Newton/Lagrange. For many points / smooth curves: splines.

Module 3: Nonlinear Equations

Bracketing: Bisection & Regula-Falsi

- Guaranteed convergence if f is continuous and changes sign on $[a, b]$.
- **Bisection:** linearly convergent, $|e_{n+1}| \leq \frac{1}{2}|e_n|$; error after n steps $\leq (b-a)/2^n$.
- **Regula-Falsi:** usually faster than bisection but may stall on one-sided convergence; modified variants fix this.

When to use

Reliable root-enclosure when a bracket is available and derivatives are expensive or unavailable.

Open Methods: Fixed Point, Newton, Secant

- **Fixed point** $x = g(x)$: converges linearly iff $|g'(x^*)| < 1$ locally; rate $= |g'(x^*)|$.
- **Newton–Raphson:** $x_{n+1} = x_n - f(x_n)/f'(x_n)$; quadratic convergence for simple roots, but only locally, and requires f'.
- **Secant:** replaces f' by a finite-difference; order $\phi = (1+\sqrt{5})/2 \approx 1.618$; two previous iterates.
- **Muller:** quadratic interpolation through three points $\Rightarrow$ can find complex roots; used for polynomial roots.

Roots & Acceleration

- For multiple roots Newton's method drops to linear convergence; **modified Newton** $x_{n+1} = x_n - m\, f/f'$ (with multiplicity m) or $u(x) = f(x)/f'(x)$ restores quadratic order.
- **Aitken's** Δ^2 and **Steffensen's** method accelerate linearly convergent sequences to quadratic at little extra cost.

Conclusion

No method is universally best. **Bracket** $\Rightarrow$ safe. **Newton** $\Rightarrow$ fastest when the derivative is available and you start close. **Secant** $\Rightarrow$ best of both in most practical problems.

Module 4: Numerical Linear Algebra

Gaussian Elimination, LU & Pivoting

- Gaussian elimination $\equiv$ LU decomposition: $A = LU$ (when no pivoting needed), or $PA = LU$.
- Cost: $\mathcal{O}(n^3)$; once LU is known, each extra RHS is $\mathcal{O}(n^2)$.
- **Partial pivoting** is essential for numerical stability; **complete pivoting** offers stronger guarantees at higher cost.
- **Variants:** Doolittle, Crout, Cholesky ($A = LL^T$ for SPD, half the cost).

Norms & Condition Number

- Vector norms: $\|x\|_1, \|x\|_2, \|x\|_\infty$. All are equivalent on $\mathbb{R}^n$, but behave differently numerically.
- Matrix norm (induced): $\|A\| = \sup_{x \neq 0} \|Ax\|/\|x\|$; columns/row-sums for 1 and ∞, largest singular value for 2.
- **Condition number** $\kappa(A) = \|A\|\|A^{-1}\|$: measures sensitivity of $Ax = b$ to perturbations.
- Rule of thumb: if $\kappa(A) \approx 10^k$, expect loss of about k digits of accuracy.

Iterative Methods: Jacobi, Gauss–Seidel, SOR, CG

- Splitting $A = M - N$, iterate $Mx_{k+1} = Nx_k + b$. Convergence iff $\rho(M^{-1}N) < 1$.
- **Jacobi** and **Gauss–Seidel** converge for strictly diagonally dominant or SPD matrices; GS uses latest updates.
- **SOR** with optimal $\omega \in (0, 2)$ can dramatically speed up GS.
- **Conjugate Gradient** is the gold standard for large, sparse, SPD systems; convergence in $\leq n$ steps (exact), and much faster with a good preconditioner.

When to iterate

When n is large and A is sparse/structured—direct methods then cost too much in flops and memory.

QR, SVD & Power Method

- **QR decomposition** ($A = QR$, Q orthogonal): backbone of least-squares (stable) and of the QR-eigenvalue algorithm.
- **SVD** $A = U\Sigma V^T$: the decomposition—gives rank, condition number (σ_1/σ_r), pseudoinverse, and the best low-rank approximation (Eckart–Young).
- **Power method** finds the dominant eigenvalue/vector, cheap but slow; inverse power + shifts give any eigenpair.

Module 5: Numerical Integration

Newton–Cotes: Trapezoidal, Simpson's 1/3 & 3/8

- Replace f by an interpolating polynomial on $[a, b]$ and integrate.
- **Trapezoidal rule:** exact for degree 1; error $-\frac{(b-a)^3}{12} f''(\xi)$.
- **Simpson's 1/3:** exact for degree 3 (bonus degree!); error $-\frac{(b-a)^5}{2880} f^{(4)}(\xi)$.
- **Simpson's 3/8:** exact for degree 3; error comparable, requires multiple of 3 intervals.
- **Open Newton–Cotes** (midpoint etc.) useful when endpoint values unavailable / singular.

Composite Rules & Romberg Integration

- Subdividing $[a, b]$ into n subintervals and summing gives composite rules:
  - Composite trapezoidal: error $= \mathcal{O}(h^2)$,
  - Composite Simpson's 1/3: error $= \mathcal{O}(h^4)$.
- **Romberg integration:** repeated Richardson extrapolation on composite trapezoidal; each level eliminates the next error term—$\mathcal{O}(h^2), \mathcal{O}(h^4), \mathcal{O}(h^6), \ldots$.

Conclusion

Romberg turns a weak rule (trapezoid) into a spectacularly strong one—at almost no extra cost.

Adaptive Quadrature & Gaussian Quadrature

- **Adaptive:** subdivide only where the local error estimate exceeds a tolerance—efficient for integrands with peaks/boundary layers.
- **Gaussian quadrature:** n cleverly chosen nodes $\Rightarrow$ exact for polynomials of degree $\leq 2n-1$ (twice the Newton–Cotes degree).
- Nodes = roots of orthogonal polynomials (Legendre, Chebyshev, Laguerre, Hermite for different weight functions and domains).

Take-away

For smooth integrands on a fixed interval: Gauss. For general or spiky integrands: adaptive Simpson/Gauss–Kronrod.

Module 6: Numerical Differential Equations

Euler & Runge–Kutta Methods

- **Euler (forward):** $y_{n+1} = y_n + h\, f(t_n, y_n)$; simple, order 1, conditionally stable.
- **Backward Euler:** implicit, A-stable—good for stiff problems.
- **RK4:** order 4, four f-evaluations per step, the workhorse of non-stiff IVPs.
- Embedded pairs (RK45, Dormand–Prince) give cheap error estimates for step-size control.

Multi-Step Methods & Stiffness

- **Adams–Bashforth** (explicit) and **Adams–Moulton** (implicit) reuse previous f-values $\Rightarrow$ cheap high-order steps once initialised.
- Predictor–corrector pairs (AB–AM) balance speed and stability.
- **Dahlquist barriers:** order, stability, and explicitness cannot all be maximised.
- **Stiff problems:** need implicit methods (BDF, implicit RK). Symptom—explicit methods need tiny steps unrelated to accuracy.

BVPs: Shooting & Finite Differences

- **Shooting:** convert BVP $\rightarrow$ IVP + a root-finding problem on the missing initial slope (Newton on the shot).
- **Finite differences:** replace derivatives by differences on a grid $\Rightarrow$ linear (or nonlinear) system—preferred for stiff/linear BVPs and for PDEs.
- **Consistency + Stability** $\Leftrightarrow$ **Convergence** (Lax equivalence for linear problems).

Overarching Principles

Ten Lessons to Carry Forward

1. Every algorithm has a **cost**, an **error** and a **stability** profile—know all three.
2. **Conditioning** is a property of the problem; **stability** is a property of the algorithm.
3. Higher order $\neq$ higher accuracy when data is noisy or step sizes are impractical.
4. Prefer **structured / sparse** representations; exploit symmetry and positive-definiteness.
5. Always **validate** with a simpler method or an analytical sub-case.
6. Know when to switch from **direct** to **iterative**, from **explicit** to **implicit**.
7. For smooth problems, use **spectral-flavoured** methods (Gauss, Chebyshev) where possible.
8. Estimate errors a posteriori—residuals, Richardson, embedded pairs.
9. Respect machine epsilon—cancellation is silent and deadly.
10. Write readable, reproducible code; a clever one-liner you cannot debug is worthless.

What to Read Next

- **Burden & Faires**, Numerical Analysis—the canonical classroom reference.
- **Trefethen & Bau**, Numerical Linear Algebra—a masterpiece on Module 4.
- **Quarteroni, Sacco & Saleri**, Numerical Mathematics—modern, MATLAB-oriented.
- **Hairer, Nørsett & Wanner**, Solving ODEs I & II—the deep dive on Module 6.
- **Higham**, Accuracy and Stability of Numerical Algorithms—error analysis bible.

A Closing Thought

"The purpose of computing is insight, not numbers."

— Richard Hamming

Numerical analysis is the craft of turning **continuous mathematics** into **reliable computation**.

What you carry forward is not a list of formulas, but a **way of thinking** about approximation, error, and trust in your answers.

Thanks

Doubts and Suggestions

panch.m@iittp.ac.in

MA633L-Numerical Analysis

Lecture 44 : Final Conclusions

Panchatcharam Mariappan[1]

[1]Associate Professor
Department of Mathematics and Statistics
IIT Tirupati, Tirupati

April 23, 2026

भारतीय प्रौद्योगिकी संस्थान तिरुपति
IIT
TIRUPATI