MA5233 Computational Mathematics

Lecture 26: Summary

Simon Etter

2019/2020

Summary

Lecture 1: Machine numbers

- Binary representation of integers and floating-point numbers. See `integers()` and `floats_*()`.
- Integer arithmetic is exact up to over- and underflow. See `integers_overflow()`.
- Floating-point arithmetic involves rounding and violates important identities. See `floats_addition()`.

Summary

Lecture 2: Conditioning and stability

- Classification of errors:
  - Absolute ($|\tilde{x} - x|$) vs. relative ($\frac{|\tilde{x}-x|}{|x|}$).
  - Forward ($|\tilde{f}(x) - f(x)|$) vs. backward ($|\tilde{x} - x|$ with $f(\tilde{x}) = \tilde{f}(x)$).
- Condition number: $\kappa(f, x) := \frac{|f'(x)|}{|f(x)|}|x|$.
- Fundamental theorem of conditioning and stability:

  $$\text{forward error} \approx \text{condition number} \times \text{backward error}$$

- Composition theorem:
  If $\tilde{f}(x)$, $\tilde{g}(x)$ are forward-stable approximations of well-conditioned functions $f(x)$, $g(x)$, then $\tilde{f}\big(\tilde{g}(x)\big)$ is forward stable.
- Condition number of addition: $\kappa\big(+, (x, y)\big) = \frac{|x|+|y|}{|x+y|}$.
  Errors can blow up when subtracting two numbers $x \approx y$!
  This is precisely what happened in `floats_addition()`.
- Condition number of matrix multiplication: $\kappa(A) := \|A\| \, \|A^{-1}\|$.

Summary

Lecture 3: LU factorisation

- For every invertible matrix $A \in \mathbb{K}^{n\times n}$, there exist
  - a permutation matrix $P \in \mathbb{K}^{n\times n}$,
  - a lower-triangular matrix $L \in \mathbb{K}^{n\times n}$ with unit diagonal, and
  - an upper-triangular matrix $U \in \mathbb{K}^{n\times n}$

  such that $PA = LU$. The L, U are unique for fixed P.
- Standard algorithm for solving dense linear systems.
- You should know by heart the algorithms for computing the LU factorisation and solving triangular systems.

  Recommended exercise: try to implement methods yourself.
- Cost of LU facorisation: $\mathcal{O}(n^3)$. Cost of triangular system: $\mathcal{O}(n^2)$.
- Conditioning of linear systems:
  If $Ax = b$ and $(A + \Delta A)(x + \Delta x) = b + \Delta b$, then

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa(A) \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right) + \mathcal{O}\big(\kappa(A)^2\big).$$

Summary

Lecture 3: LU factorisation (continued)

- Stability of LU factorisation: Numerical solution $\tilde{x}$ to $Ax = b$ computed via LU factorisation satisfies

$$(A + \Delta A)\,\tilde{x} = b \qquad \text{where} \qquad \frac{\|\Delta A\|}{\|L\|\,\|U\|} \approx \mathcal{O}\big(\varepsilon_{\text{mach}}\big).$$

  Solving linear systems via LU is backward stable if $\|L\|\,\|U\| \approx \|A\|$.
- Combining conditioning and stability yields

$$\frac{\|\tilde{x} - x\|}{\|x\|} \approx \kappa(A)\,\frac{\|L\|\,\|U\|}{\|A\|}\,\mathcal{O}\big(\varepsilon_{\text{mach}}\big).$$

- No pivoting: $\|L\|, \|U\| = \infty$ is possible. However, no-pivoting LU factorisation is provably stable for diagonally dominant and symmetric positive definite matrices (see HW4).
- For both partial and complete pivoting, we have $\max_{ij} |L_{ij}| = 1$.
- Partial pivoting: $\max_{ij} |U_{ij}| = 2^{n-1} \max_{ij} |A_{ij}|$ is possible (see Wilkinson's matrix from HW2) but never occurs in practice.
- Complete pivoting: probably $\max_{ij} |U_{ij}| = \mathcal{O}(n) \max_{ij} |A_{ij}|$. Complete pivoting is never used in practice.

Summary

Lecture 4: QR factorisation

- Any $A \in \mathbb{K}^{m \times n}$ can be written as $A = QR$, where
  - $Q \in \mathbb{K}^{m \times m}$ is orthogonal ($Q^H Q = I$), and
  - $R \in \mathbb{K}^{m \times n}$ is upper triangular.

  Thin QR: $Q \in \mathbb{K}^{m \times n}$, $R \in \mathbb{K}^{n \times n}$.
- Provably backward-stable algorithm for solving linear systems (but rarely used for that purpose since LU factorisation is faster and usually achieves similar accuracy).
- Algorithm of choice for solving linear least-squares problem (see Lecture 9 and Krylov methods).
- Method of choice for determining orthogonal basis for space spanned by columns of A (used in eigenvalues algorithms).
- Two algorithms for computing QR factorisation: Gram-Schmidt and Householder (you may ignore Givens).
- You should be able to derive Gram-Schmidt and Householder QR algorithms. Recommended exercise: try to implement methods yourself.

Summary

Lecture 4: QR factorisation (continued)

	Gram-Schmidt	Householder
Factorisation:	thin	fat
Vector space:	abstract	$\mathbb{K}^n$
Orthogonality of Q:	may fail	$\mathcal{O}\left(\varepsilon_{\mathsf{mach}}\right)$
Backward stable:	yes (for MGS)	yes
Cost:	$\mathcal{O}(mn^2)$	$\mathcal{O}(mn^2)$

Summary

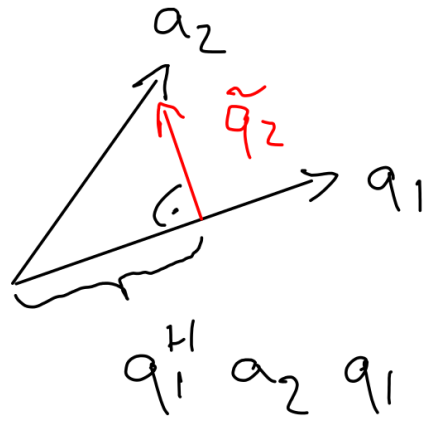
Lecture 4: QR factorisation (continued)

Key formulae for Gram-Schmidt:

$$\tilde{q}_{k+1} := a_{k+1} - \sum_{\ell=1}^{k} q_\ell^H a_{k+1}\, q_\ell$$

$$\Longleftrightarrow$$

$$a_{k+1} = \underbrace{\|\tilde{q}_{k+1}\|_2}_{R[k+1,k+1]}\, q_{k+1} + \sum_{\ell=1}^{k} \underbrace{q_\ell^H a_{k+1}}_{R[\ell,k+1]}\, q_\ell$$



Classical Gram-Schmidt:

$$\tilde{q}_{k+1}^{(0)} := a_{k+1}, \qquad \tilde{q}_{k+1}^{(\ell+1)} := \tilde{q}_{k+1}^{\ell} - q_\ell^H a_{k+1}\, q_\ell$$

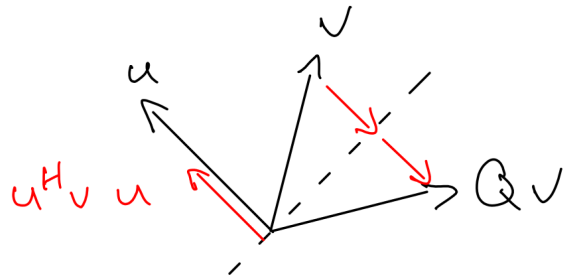
Modified Gram-Schmidt:

$$\tilde{q}_{k+1}^{(0)} := a_{k+1}, \qquad \tilde{q}_{k+1}^{(\ell+1)} := \tilde{q}_{k+1}^{(\ell)} - q_\ell^H \tilde{q}_{k+1}^{(\ell)}\, q_\ell$$

Summary

Lecture 4: QR factorisation (continued)

Householder reflector: $Q := I - 2uu^H$ with $\|u\|_2 = 1$.



Householder QR factorisation:

- Choose Q_1 such that $Q_1 a_1 = -\operatorname{sign}(a_{11}) \|a_1\| e_1$.

- Use this idea iteratively, starting from $R_0 := A$.

$$\begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \\ x & x & x \end{pmatrix}_{R_0} \xrightarrow{R_1 = Q_1 R_0} \begin{pmatrix} x & x & x \\ & x & x \\ & x & x \\ & x & x \end{pmatrix}_{R_1} \xrightarrow{R_2 = Q_2 R_1} \begin{pmatrix} x & x & x \\ & x & x \\ & & x \\ & & x \end{pmatrix}_{R_2} \xrightarrow{R_3 = Q_3 R_2} \begin{pmatrix} x & x & x \\ & x & x \\ & & x \\ & & \end{pmatrix}_{R_3}$$

- Final factorisation: $A = Q_1 Q_2 Q_3 R_3$.

Summary

Lecture 5: Poisson equation

$$\begin{cases} -\Delta u(x) = f(x) & \forall x \in \Omega, \\ u(x) = 0 & \forall x \in \partial\Omega. \end{cases}$$

Finite difference discretisation in 1D: $-\Delta_n u_n = f_n$, or

$$(n+1)^2 \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \begin{pmatrix} u\left(\frac{1}{n+1}\right) \\ \vdots \\ u\left(\frac{n}{n+1}\right) \end{pmatrix} = \begin{pmatrix} f\left(\frac{1}{n+1}\right) \\ \vdots \\ f\left(\frac{n}{n+1}\right) \end{pmatrix}.$$

Consistency and stability implies convergence theorem:

$$\|u - u_n\| \leq \|\Delta_n^{-1}\| \, \|\Delta_n u + f_n\|.$$

- Estimate $\|\Delta_n^{-1}\|$ using Fourier theory (we will return to this later).
- Estimate $\|\Delta_n u + f_n\|$ using Taylor series.

Summary

Lecture 6: Sparse matrices

Sparse vector formats:

- Coordinate list (`i`,`j`,`v`): convenient.
- Compressed sparse column (CSC) (`p`,`i`,`v`): more storage-efficient.

Kronecker product:

$$A \otimes B := \begin{pmatrix} A[1,1]\,B & \cdots & A[1,n]\,B \\ \vdots & \ddots & \vdots \\ A[n,1]\,B & \cdots & A[n,n]\,B \end{pmatrix}$$

- Useful identities:
  - $(A \otimes B)(a \otimes b) = (Aa) \otimes (Bb)$
  - $\big(A \otimes B\big)\,\mathrm{vec}(C) = \mathrm{vec}\big(BCA^T\big)$
- 2d Laplacian matrix: $\Delta_n^{(2)} = \Delta_n^{(1)} \otimes I + I \otimes \Delta_n^{(1)}$.
  Eigenpairs of $\Delta_n^{(2)}$: $\quad \lambda_{\ell_1,\ell_2} := \lambda_{\ell_1} + \lambda_{\ell_2}, \quad u_{\ell_1,\ell_2} := u_{\ell_1} \otimes u_{\ell_2}$.

Summary

Lecture 7: Sparse LU factorisation

- Graph $G(A) := \big(V(A), E(A)\big)$ defined by

$$V(A) := \{1, \ldots, n\}, \qquad E(A) := \big\{j \to i \mid A[i,j] \neq 0\big\}.$$

- Key results:

$$\begin{aligned} A^p[i,j] \neq 0 &\iff \exists \text{ path } j \to i \text{ of length } p \\ A^{-1}[i,j] \neq 0 &\iff \exists \text{ path } j \to i \\ (L+U)[i,j] \neq 0 &\iff \exists \text{ fill path } j \to i \end{aligned}$$

- Vertex orders to reduce fill-in: nested dissection, approximate minimum degree (AMD).
- Complexity of LU factorisation for partial differential equations:

	$d = 1$	$d = 2$	$d = 3$
Runtime:	$\mathcal{O}(N)$	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N^2)$
Memory:	$\mathcal{O}(N)$	$\mathcal{O}(N \log(N))$	$\mathcal{O}(N^{4/3})$

Summary

Lecture 8: Fast Fourier transform

- Sine matrix $S_n \in \mathbb{R}^{n \times n}$, given by $(S_n)_{k\ell} := \sin\left(\pi \frac{k\ell}{n+1}\right)$.
- Key identities: $S_n^2 = \frac{n+1}{2} I$ and $\Delta_n S_n = S_n \Lambda_n$ with

$$(\Lambda_n)_{k\ell} = (n+1)^2 \left(2\cos\left(\pi \tfrac{\ell}{n+1}\right) - 2\right) \delta_{k\ell}.$$

  Recommended exercise: prove these identities.

- Important algorithmic result:
  $S_n v$ can be computed in $\mathcal{O}\big(n \log(n)\big)$ FLOP rather than $\mathcal{O}\big(n^2\big)$!
- Corollary: $-\Delta u = f$ on $[0,1]^d$ can be solved with $\mathcal{O}\big(n^d \log(n)\big)$ FLOP in any dimension d.
- Sine matrix is closely related to Fourier matrix $(F_n)_{k\ell} := \exp\big(2\pi\iota \frac{k\ell}{n}\big)$.
  $S_n^2 = \frac{n+1}{2} I$ can be derived from $F_n^H F_n = nI$.

Summary

Lecture 9: Linear least squares

$$\arg\min_x \|Ax - b\|_2 = R^{-1} Q^H b$$

where $QR = A$ is the thin QR factorisation of $A \in \mathbb{K}^{m \times n}$.

Lectures 10-12: Krylov subspace methods

Approximate $x = A^{-1} b$ by

$$\tilde{x} := p_n(A)\, b \qquad \text{where} \qquad p_n := \arg\min_{p_n \in \mathcal{P}_n} \left\| \left(A p_n(A) - I \right) b \right\|$$

- GMRES: A arbitrary, norm $\|r\|_2 = \sqrt{r^T r}$.
- MinRes: A symmetric, norm $\|r\|_2 = \sqrt{r^T r}$.
- Conjugate gradients: A symmetric, norm $\|r\|_{A^{-1}} = \sqrt{r^T A^{-1} r}$.

Summary

Lectures 10-12: Krylov subspace methods (continued)

Implementation:

- Krylov minimisation problem is equivalent to

$$\tilde{x} := V_n y \quad \text{where} \quad y = \arg\min \|AV_n y - b\|, \quad V_n = \begin{pmatrix} b & Ab & \dots & A^n b \end{pmatrix}.$$

- Arnoldi iteration: use Gram-Schmidt to determine orthogonal matrix Q_n such that $\operatorname{span}\{q_1, \dots, q_n\} = \operatorname{span}\{b, Ab, \dots, A^{n-1}b\}$.
- Arnoldi relation: $AQ_n = Q_{n+1}H_n$ with Hessenberg $H_n \in \mathbb{K}^{(n+1)\times n}$. Allows us to rewrite $\|AQ_n y - b\|_2 = \left\|H_n y - \|b\|_2 e_1\right\|_2$ which is crucial to making Krylov methods numerically stable and fast.
- If A is symmetric, Arnoldi simplifies to Lanczos.
- Arnoldi/Lanczos iteration dominates cost of Krylov methods. These costs are n matrix-vector products and $\mathcal{O}\left(Nn^p\right)$ other FLOP, where $p = 2$ for general A and $p = 1$ for symmetric A.
  (N denotes dimension of A)
- GMRES requires storage for full Q_n. For MinRes and CG, it is possible to rearrange algorithms such that only a fixed number of vectors have to be stored.

Summary

Lectures 10-12: Krylov subspace methods (continued)

Convergence:

- Assume A has eigendecomposition $A = V\Lambda V^{-1}$. Then

$$\|A\tilde{x} - b\|_2 \leq \kappa(V)\,\|b\|_2 \min_{q_n \in \mathcal{P}_n} \max_{\lambda_k} \frac{|q_n(\lambda_k)|}{|q(0)|}.$$

- Bounding last factor is hard in general, but we have the important special case

$$\min_{q_n \in \mathcal{P}_n} \max_{x \in [1,\kappa]} \frac{|q_n(x)|}{|q_n(0)|} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^n.$$

- Initial guess: solve $A\,\Delta x = b - Ax_0$ and set $x = x_0 + \Delta x$.
- Preconditioning: solve $P^{-1}Ax = P^{-1}b$ with P such that eigenvalues of $P^{-1}A$ are "nice" and $P^{-1}v$ is cheap to evaluate.

Summary

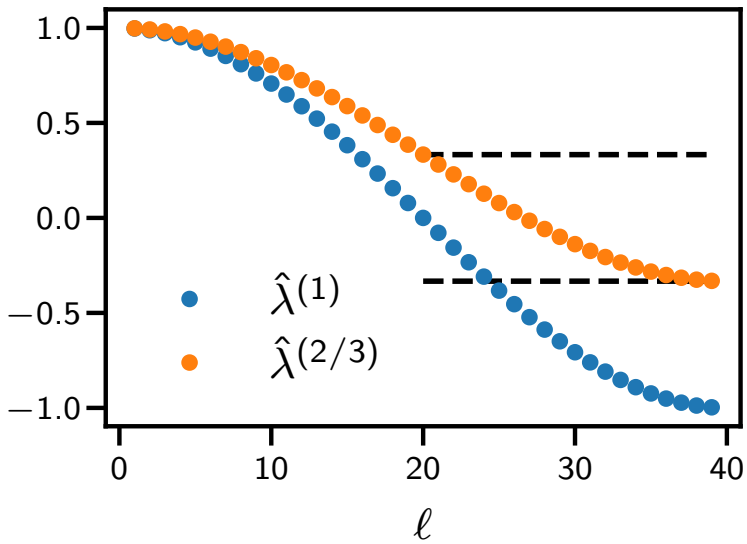
Lecture 13: Multigrid

- Another algorithm for solving $Ax = b$, tailored particularly for linear systems arising from partial differential equations.
- Relaxed Jacobi iteration and associated error recursion:

$$x^{(k+1)} = (1-\theta)\, x^{(k)} + \theta\, D^{-1}\left(b - (A-D)\, x^{(k)}\right)$$

$$x^{(k)} - x = \underbrace{\Big((1-\theta)\, I - \theta\, D^{-1}\,(A-D)\Big)}_{R_\theta}\,(x^{(k-1)} - x),$$

- Eigenvalues of R_θ:



Summary

Lecture 13: Multigrid (continued)

- Coarse-grid correction:

$$x^{(k+1)} = x^{(k)} + \tfrac{1}{2} P (A')^{-1} P^T (b - A x^{(k)})$$

- Multigrid idea: use relaxed Jacobi (or other "smoother") to eliminate high-frequency errors, then use coarse-grid correction to recursively eliminate low-frequency errors.
- Multigrid solves the localisation problem: every entry of $x = A^{-1} b$ depends on all entries of b.
  - Sparse LU tracks this dependence explicitly which is expensive.
  - Krylov and Jacobi propagate information locally, which requires many steps.
  - Multigrid splits information into components and propagates each component with the appropriate step size.

Summary

Lecture 14: Polynomial approximation

- Given $f : [-1, 1] \to \mathbb{R}$, find $p := \arg\min_{p \in \mathcal{P}_n} \|f - p\|_{[-1,1]}$.
- Theoretical foundation for much of numerical linear algebra and numerical analysis.
- Best approximation: existence and uniqueness is guaranteed, and $p := \arg\min_{p \in \mathcal{P}_n} \|f - p\|_{[-1,1]}$ if and only if $f(x) - p(x)$ equioscillates in at least $n + 2$ points.
- Interpolation: existence and uniqueness is guaranteed, and we have the error estimate

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

- Interpolation in Chebyshev points is well-conditioned and we have

$$\|f - p_{\text{best}}\|_{[-1,1]} \leq 2 \left(1 + \log(n+1)/\pi\right) \|f - p_{\text{chebint}}\|.$$

- Interpolation in equispaced points is ill-conditioned and may diverge.
- Lagrange polynomials: $\ell_j(x) := \prod_{i \neq j} \frac{x - x_i}{x_j - x_i}$.

  Key property: $\ell_j(x_i) = \delta_{ij}$.

Summary

Lecture 14: Polynomial approximation (continued)

- Chebyshev polynomials:

$$T_0(x) = 1, \qquad T_1(x) = x, \qquad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

  Chebyshev polynomials are related to Joukowsky map

$$\phi(z) := \frac{z + z^{-1}}{2}, \qquad \phi_{\pm}^{-1}(x) := x \pm \sqrt{x^2 - 1}$$

  Key property: $T_n(x)$ equioscillates in $n+1$ points on $[-1, 1]$.

- Legendre polynomials:

$$L_0(x) = 1, \quad L_1(x) = x,$$
$$(n+1)\, L_{n+1}(x) = (2n+1)\, x\, L_n(x) - n\, L_{n-1}(x).$$

  Key property: $\displaystyle\int_{-1}^{1} L_n(x)\, L_m(x)\, dx = \frac{2}{2n+1}\, \delta_{mn}$ (orthogonality).

- Three-term recurrence relations are related to Gram-Schmidt.

Summary

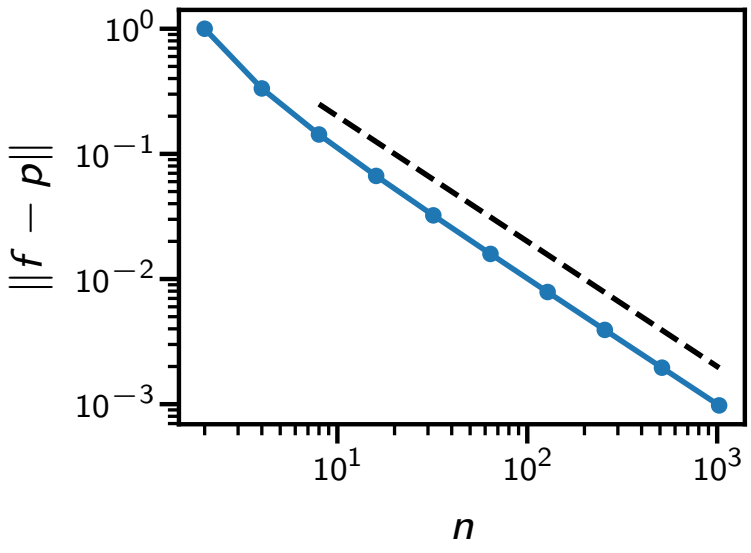
Lecture 14: Polynomial approximation (continued)

If $f \in C^{k-1}([-1,1])$ and $f \in C^k([-1,1])$, then we have algebraic convergence of order k,

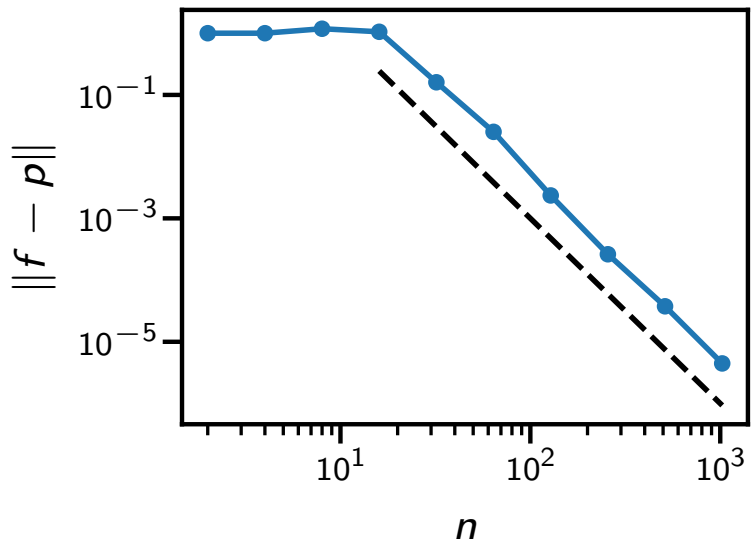
$$\min_{p \in \mathcal{P}_n} \|f - p\|_{[-1,1]} \leq C \, \|f^{(k)}\|_{[-1,1]} \, n^{-k}.$$

Example:

$f_1(x) = |x| \to e_n = \mathcal{O}(n^{-1})$



$f_2(x) = |\sin(4\pi x)|^3 \to e_n = \mathcal{O}(n^{-3})$



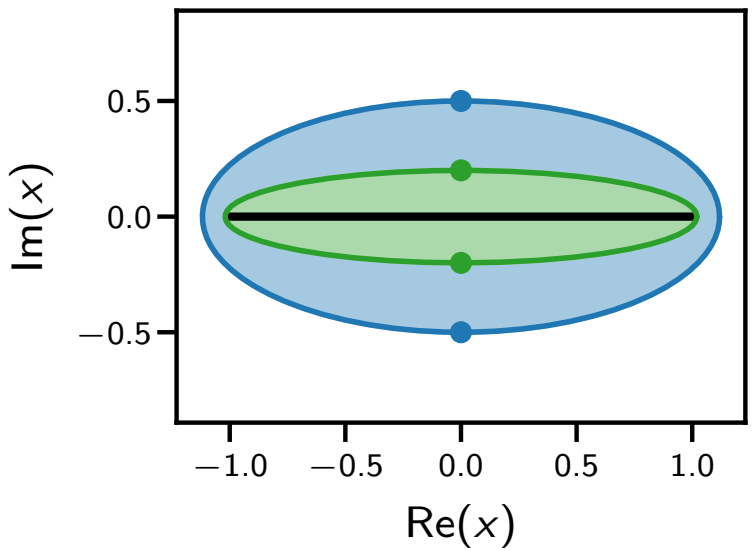
Summary

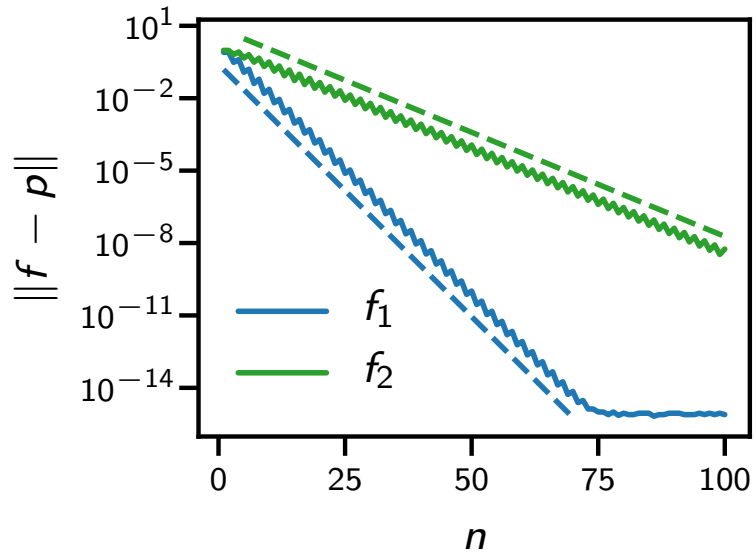
Lecture 14: Polynomial approximation (continued)

If f is analytic on Bernstein ellipse $E(r)$, then we have exponential convergence with rate r,

$$\min_{p \in \mathcal{P}_n} \|f - p\|_{[-1,1]} \leq C \, \|f\|_{E(r)} \, r^{-n}.$$

Example: $f_1(x) = \dfrac{1}{1+4x^2}$, $f_2(x) = \dfrac{1}{1+25x^2}$.



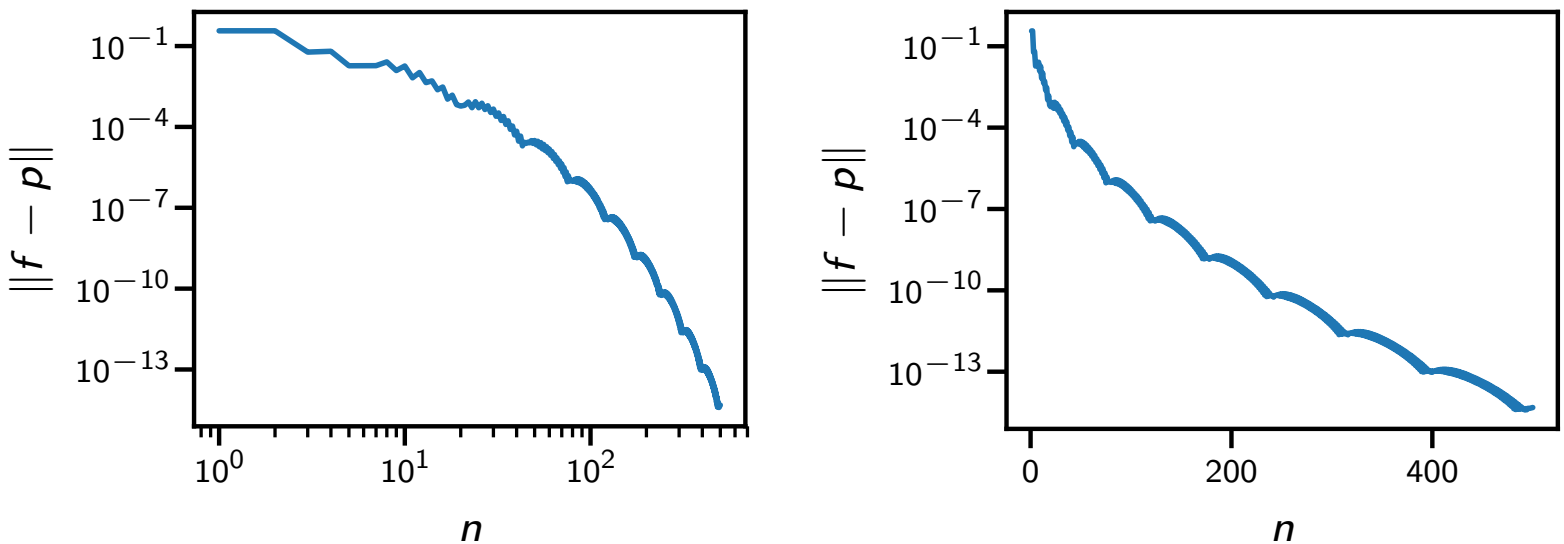


Summary

Lecture 14: Polynomial approximation (continued)

If $f \in C^\infty([-1,1])$ but not analytic: polynomial approximation converges superalgebraically but supexponentially.

Example: $f(x) := \exp\left(-\frac{1}{|x|}\right)$.



Summary

Lecture 15: Quadrature

- Quadrature rule: $\sum_{k=1}^{n} f(x_k)\, w_k \approx \int_a^b f(x)\, dx$.
- Recipe for constructing quadrature rules: choose x_k, then determine w_k such that quadrature rule is exact for $f \in \mathcal{P}_{n-1}$.
  Recommended exercise: write down linear system for w_k.
- Gauss quadrature: choose x_k as roots of Legendre polynomial $L_n(x)$.
  Then quadrule will be exact for $f \in \mathcal{P}_{2n-1}$.
  Gauss quadrature is optimal: $\nexists$ quadrule exact for all $f \in \mathcal{P}_{2n}$.
  Recommended exercise: prove exactness and optimality of Gauss quadrature.
- Convergence theory for quadrature follows from convergence theory for polynomial interpolation.
- Composite quadrature: local quadrule exact on $\mathcal{P}_d \implies$ error in m-interval composite quadrule is $\mathcal{O}\left(m^{-d-1}\right)$, assuming $f \in C^{d+1}$.

Summary

Lectures 16-17: Ordinary differential equations

- Given $f : \mathbb{R}^n \to \mathbb{R}^n$, find $y : [0, T] \to \mathbb{R}^n$ such that $\dot{y} = f(y)$.
- Solution can be computed by recursively applying quadrature to

$$y(t) = \int_0^t f\big(y(\tau)\big)\, d\tau.$$

- s-stage Runge-Kutta/single-step schemes:
  - Split $[0, T]$ into m intervals $\big([t_{k-1}, t_k]\big)_{k=1}^m$.
  - On each interval, use s-point quadrature rule.
    (More precisely, s denotes the number of $f(y)$ evaluations.)
- Runge-Kutta step corresponds to numerical time propagator $\tilde{\Phi}_t : y(0) \mapsto \tilde{y}(t)$ which approximates exact time propagator Φ_t.
- Consistency and stability imply convergence, where
  - Consistency: $\|\tilde{\Phi}_t(y) - \Phi_t(y)\| = \mathcal{O}\big(t^{p+1}\big)$.
  - Stability: $\|\tilde{\Phi}_t(y_1) - \tilde{\Phi}_t(y_2)\| \leq (1 + \tilde{L}t)\, \|y_1 - y_2\|$
  - Convergence: $\|\tilde{y}(T) - y(T)\| = \mathcal{O}\big(m^{-p}\big)$.
- Consistency can be shown via Taylor expansion.
- Stability follows from Lipschitz-continuity of f and triangle ineq.

Summary

Lectures 16-17: Ordinary differential equations (continued)

- Butcher tableau for Runge-Kutta methods: $\left(\begin{array}{c|c} x & V \\ \hline & w^T \end{array}\right)$
- RK scheme is called *explicit* if V is strictly lower triangular, and *implicit* otherwise. Implicit schemes require solving linear systems of the form $y_1 = y_0 + f(y_1)$ (or more complicated) for y_1.
- Single Runge-Kutta applied to $\dot{y} = \lambda y$ yields $y(t) = R(\lambda t)\, y(0)$ where $R(z)$ is the *stability function* of the RK scheme.
- There is an explicit formula for $R(z)$ in terms of V and w.
- Stability domain: $\left\{ z \in \mathbb{C} \mid |R(z)| \leq 1 \right\}$.
- Stability domain is always bounded for explicit schemes, but may be unbounded for implicit schemes.
- Implicit schemes solve the *separation of time scales* problem, see HW6 and Lecture 22 (Time-dependent PDEs).

Summary

Lectures 19: Theory of PDEs

- $L^2([a,b]) := \{f : [a,b] \to \mathbb{R} \mid \|f\|_{L^2([a,b])} < \infty\}$,

$$\langle f, g\rangle_{L^2([a,b])} := \int_a^b f(x)\, g(x)\, dx, \qquad \|f\|_{L^2([a,b])} := \sqrt{\int_a^b f(x)^2\, dx}.$$

- Weak derivative: $f' \in L^2([a,b])$ is weak derivative of $f \in L^2([a,b])$ if for all $v \in C^\infty([a,b])$ with $v(a) = v(b) = 0$ we have

$$\int_a^b f(x)\, v'(x)\, dx = -\int_a^b f'(x)\, v(x)\, dx.$$

- Sobolev space

$$H^k([a,b]) := \{f \in L^2([a,b]) \mid f \text{ has } k \text{ weak derivatives}\},$$

$$\langle f, g\rangle_{H^k([a,b])} := \sum_{\ell=0}^{k} \langle f^{(\ell)}, g^{(\ell)}\rangle_{L^2([a,b])},$$

$$\|f\|_{H^k([a,b])} := \sqrt{\sum_{\ell=0}^{k} \|f^{(\ell)}\|^2_{L^2([a,b])}}.$$

Summary

Lectures 19: Theory of PDEs (continued)

- Sobolev space $H_0^k([a,b])$ with zero boundary conditions.
- L^2, H^k and H_0^k are *Hilbert spaces*.
- Weak formulation of Poisson equation:
  Find $u \in H_0^1([a,b])$ such that for all $v \in H_0^1([a,b])$ we have

$$\int_0^1 u'(x)\, v'(x)\, dx = \int_0^1 f(x)\, v(x)\, dx.$$

- Lax-Milgram: $a(u,v) = b(v)$ for all $v \in V$ has unique solution $u \in V$ if
  - $a(u,v)$ is bounded: $\exists A > 0$ such that $|a(u,v)| \leq A\, \|u\|_V\, \|v\|_V$.
  - $a(u,v)$ is coercive: $\exists c > 0$ such that $a(v,v) \geq c\|v\|_V^2$.
  - $b(v)$ is bounded: $\exists B > 0$ such that $|b(v)| \leq B\, \|v\|_V$.
- Poincaré ineq.: $\|v\|_{L^2([0,1])} \leq C\, \|v'\|_{L^2([0,1])}$ for all $v \in H_0^1([0,1])$.
  Needed to show coercivity of Poisson equation.

Summary

Lecture 20: Galerkin's method

- Approximate solution of

$$\text{Find } u \in V \text{ such that } a(u, v) = b(v) \text{ for all } v \in V.$$

  by

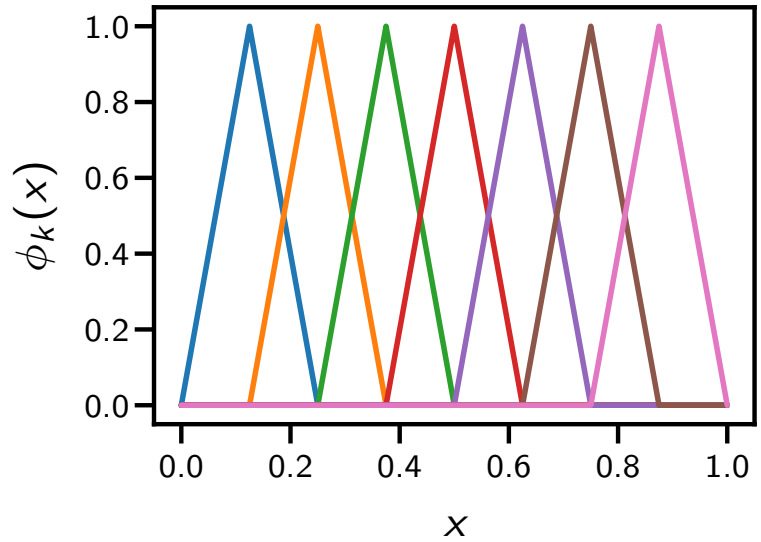
$$\text{Find } u_n \in V_n \text{ such that } a(u_n, v_n) = b(v_n) \text{ for all } v_n \in V_n. \qquad (1)$$

  where V_n is some finite-dimensional subspace of V.
- Galerkin problem (1) can be translated into a linear system.
- Céa's lemma: $\|u - u_n\|_V \leq \frac{A}{c} \inf_{v_n \in V_n} \|u - v_n\|_V$.
- Aubin-Nitsche lemma: usually implies that error converges with an additional power in L^2-norm compared to H^1-norm.

Summary

Lecture 21: Finite element method

- Galerkin's method using subspace of piecewise linear functions.
- Hat basis functions:



- Convergence theory follows from Céa and Aubin-Nitsche lemmas, but is a bit technical.
- You should know how to implement the finite element method in one dimension.

Summary

Lecture 22: Time-dependent PDEs

- Algorithms are a combination of FEM/FD and Runge-Kutta.
- Combination of $\mathcal{O}\left(n^{-p}\right)$-method in space and $\mathcal{O}\left(m^{-q}\right)$-method in time yields $\mathcal{O}\left(n^{-p} + m^{-q}\right)$ error.
- Explicit time-stepping schemes introduce $\Delta t \leq C\Delta x^2$ constraint.
  Power 2 in Δx^2 is a consequence of linear FEM/FD discretisation and may differ for other spatial discretisation schemes.

Lecture 23: Eigenvalues

- All eigenvalue solvers must be iterative.
- Single eigenpairs can be computed using power method.
- All eigenpairs can be computed by combining simultaneous iteration with a finite preprocessing step based on Householder reflectors. This preprocessing step requires $\mathcal{O}\left(n^3\right)$ FLOP and dominates the runtime of the overall algorithm.

Summary

Lecture 24: Monte Carlo

- Approximate $\mathbb{E}[F]$ by $Q := \sum_{k=1}^{N} F_k$.
- Error estimate is provided by central limit theorem, which says that for large N, Q is approximately normally distributed with $\mathbb{E}[Q] = \mathbb{E}[F]$ and $\mathrm{Var}[Q] = \frac{\mathrm{Var}[F]}{N}$, assuming F_k are independent.
- Central limit theorem implies $\mathcal{O}(N^{-1/2})$ convergence both for expected error and confidence interval.
- Transformation, rejection and importance sampling theorems.

  Recommended exercise: prove these results.

Summary

Lecture 25: Markov Chain Monte Carlo

- Markov chains: advanced technique for sampling from complicated distributions and/or distributions which are only known up to normalising constant.
- Markov property:

$$P(X_{k+1} = x_{k+1} \mid X_k = x_k, \ldots, X_0 = x_0) = P(X_{k+1} = x_{k+1} \mid X_k = x_k).$$

- In practice, we generate Markov chain by sampling

$$X_0 \sim p_0(X_0), \qquad X_{k+1} \mid X_k \sim p(X_{k+1} \mid X_k).$$

- Stationary distribution: $s(x_{k+1}) = \int p(x_{k+1} \mid x_k)\, s(x_k)\, dx_k$
- Stationary distribution is unique and $\lim_{k\to\infty} P(X_k = x) = s(x)$ under suitable conditions.
- Detailed balance: $s(x)$ is stationary distributions if (but not only if)

$$p(x_{k+1} \mid x_k)\, s(x_k) = p(x_k \mid x_{k+1})\, s(x_{k+1}).$$

Summary

Lecture 25: Markov Chain Monte Carlo (continued)

- Metropolis step: algorithm for generating $X_{k+1} \mid X_k$ such that detailed balance is satisfied for a given target stationary distribution.
- Two steps of Metropolis algorithm:
  - Propose arbitrary step $\tilde{X}_{k+1} \mid X_k$.
  - Accept step with carefully chosen probability which depends on target and proposal distributions.

  Recommended exercise: prove that Metropolis step satisfies detailed balance.
- Central limit theorem does not apply to Marky chain Monte Carlo due to correlated samples. Roughly speaking, error e_N behaves like

$$e_N = \mathcal{O}\left(\sqrt{\frac{\mathrm{Var}[F]\, N_{\mathrm{corr}}}{N - N_{\mathrm{start}}}}\right).$$

  - Burn-in time N_{start}: number of steps required for Markov chain to "forget" initial distribution $p_0(x_0)$.
  - Autocorrelation time N_{corr}: number of steps required such that $X_{k+N_{\mathrm{corr}}}$ becomes independent of X_k.

Summary

Conclusion

Thank you for being an excellent class, and best of luck for the exam!