2016-04-23

topics to be covered

- √Computer Arithmetic
- √Matrices and Vectors - Definitions
- √Gauss Elimination
- √LU factorization
- √Pivoting
- √Norms and condition numbers of matrices

notes1 - computer arithmetic

- signs can be represented as an extra digit 0 or 1
- hexadecimal system (base=16) digits used: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- (human) representation of reals: $x = \pm(x_I . x_F)_b = \pm(d_n d_{n-1} \cdots d_0 . d_{-1} d_{-2} \cdots)_b$, where x_I is the integral part, and x_F the fraction. The sign is represented by one digit, 0 or 1.
- A terminating decimal fraction may have an equivalent non-terminating binary fraction.
- computer representation of numbers: floating-point.
  - $x = (f)_b \times b^{(e)_b}$, t base-b digits precision
  - $f = \pm(. d_1 d_2 \cdots d_t)_b$ is mantissa (or significand). The absolute value of mantissa is in $[0, 1)$.
  - $e = \pm(c_{s-1} c_{s-2} \cdots c_0)_b$ is exponent or (characteristic).
    - The exponent is limited: $E_{min} \leq e \leq E_{max}, -E_{min} = E_{max} = (aa \cdots a)_b, a = b - 1$
  - normalised if $d_1 \neq 0, orelse d_1 = d_2 = \cdots = d_n = 0$
  - significant digits: digitis following and including the first non-zero digit of the mantissa.
  - The largest floating-point number (overflow level, OFL) is $N_{max} = (. aa \cdots a)_b \times b^{(aa \cdots a)_b}, a = b - 1$
  - underflow level, UFL, $N_{min} = (.100 \cdots 0)_b \times b^{-(aa \cdots a)_b}$ if normalized, or $N_{min} = (.00 \cdots 01)_b \times b^{-(aa \cdots a)_b}$
- Set of floating-point numbers. $R_b(t, s)$ denotes the set of all base b floating-point numbers that can be represented by t b-digits mantissa (incl. sign), and s b-digits exponent (incl. sign). Note: $R_b(t, s) \subset [-OFL, -UFL] \cup \{0\} \cup [UFL, OFL]$.
  - $R_b(t, s)$ is finite, R is infinite.
  - R is dense, while $R_b(t, s)$ is not.
- Approximation of real numbers by floating-point numbers: $fl(x) \in R_b(t, s)$
  - $x = (d_k \cdots d_0 . d_{-1} \cdots)_b = (. D_1 D_2 \cdots)_b \times b^{k+1}$
  - chopping: chop after digit t of mantissa
  - rounding (traditiona): chop after digit t, then round D_t up or down, depending on whether $D_{t+1} \geq b/2$ or $D_{t+1} < b/2$ respectively.
  - rounding (proper or perfect): same as traditional, except when $D_{t+1} = b/2$ and $D_{t+2} = D_{t+3} = \cdots = 0$ then round D_t up or down, to the nearest even.
- IEEE Standard, single precision (binary32), double precision (binary64) and quadruple precision (binary128)...
- Round-off error (representation error): difference between $fl(x)$ and x. $fl(x) = x(1 + \delta)$, where δ is the relative round-off error.

    - $|\delta| \leq b^{1-t}$ if normalized numbers and chopping are assumed
    - $|\delta| \leq \frac{1}{2} b^{1-t}$ if normalized numbers and rounding are assumed
    - an approximation $\hat{x}$ to x is said to be correct in r significant b-digits, if $|\frac{x-\hat{x}}{x}| \leq \frac{1}{2} b^{1-r} (= 5b^{-r} \text{ if } b = 10)$
- Absolute and relative errors
    - $\hat{x} - x$ and $\frac{\hat{x}-x}{x}$, we are ususally interested in the absolute value of relative errors
- computer arithmetic
    - operation $o \in \{+, -, \times, /\}$, computer's floating-point operation $\bar{o}$ corresponding to o:
    - $xoy \neq fl(x\bar{o}y)$, but $x\bar{o}y = fl(xoy)$, so the error of a computation is the round-off error of the correct result.
    - The phenomenon in which a non-zero number is added to another and the latter is left unchanged is often referred to as saturation.
    - $\bar{f}(x) = fl(f(x))$
- machine epsilon: the smallest (non-normalized) floating-point number ϵ_{mach} with the property $1 + \epsilon_{mach} > 1$ is mach-eps.
    - $\epsilon_{mach} = b^{1-t}$ if chopping
    - $\epsilon_{mach} = \frac{1}{2} b^{1-t}$ if traditional rounding and
    - $\frac{1}{2} b^{1-t} < \epsilon_{mach} \leq b^{1-t}$, $(\epsilon_{mach} = \frac{1}{2} b^{1-t} + b^{-t})$ if proper rounding
    - thus $-\epsilon_{mach} \leq \delta \leq \epsilon_{mach}$
    - also $0 < UFL < \epsilon_{mach} < OFL$
- error propagation: as soon as an error (round-off or by a floating-point operation) rises, it may then be amplified or reduced in subsequent operations.
    - adding nearly opposite or subtracting nearly equal numbers may result in having no correct digits at all. In some cases though, we are able to eliminate this cancellation phenomenon (often referred to as catastrophic cancellation)
    - conditioning of problems
        - the factor $\kappa_f = |\frac{xf'(x)}{f(x)}|$ is called (relative) condition number of $f(x)$ and is a measure of the relative sensitivity of the computation of $f(x)$ on relatively small changes in the input x, or in other words a measure of how the relative error in x propagates in $f(x)$.
        - a computation is called well-conditioned if relatively small changes in the input, produce relatively small changes in the output, otherwise it is called ill-conditioned.
        - the condition number of a function is a property inherent to the function itself, and not to the way the function is computed.
    - stability of algorithms
        - stability is a concept similar to conditioning, but it refers to a numerical algorithm, i.e., to the particular way a certain computation is carried out. A numerical algorithm is stable, if small changes in the algorithm input parameters have a small effect on the algorithm output, otherwise it is called unstable.
        - general consideration: mathematically equivalent expressions are not necessarily computationally equivalent.
        - how to avoid?
            - to avoid adding nearly opposite or subtracting nearly equal numbers
            - to minimize the number of operations
            - when adding several numbers, to add them from the smallest to proceeding to the largest
            - to be alert when adding numbers of very different scales
            - but no rule guarantees success
- forward and backward errors
    - $y - \hat{y}$ is forward error. may include initial data error in x, as well as propagation of error in

computations.
    - if all errors were due to computations, can view $\hat{y}$ as being the result of inexact computations on exact input $\hat{y} = \hat{f}(x)$
    - if all errors were due to initial data error, one can view $\hat{y}$ as beign the result of exact computations on inexact input $\hat{y} = f(\hat{x})$
    - So $\hat{f}(x) = f(\hat{x})$, if f^{-1} exists, then $\hat{x} = f^{-1}(\hat{f}(x))$
  - $x - \hat{x}$ is backward error
  - $|\text{relative forward error }| = \text{condition number} \times |\text{ relative backward error}|$
- truncation (discretization) and rounding errors
  - sometimes certain mathematical expressions are approxmiated by other (possibly more convenient for calculations) expressions. The latter expression can be viewed as algorithms for approximating the original expressions. The error in these approximations is referred to as trunction or discretization error.
  - in addition, the evaluation of approximate expressions is not performed in exact but in finite arithmetic. Therefore, it may involve additional error, referred to as rounding error.
  - together form the computational error
- total error
  - $$\begin{aligned}\text{total error} &= f(x) - \hat{g}(\hat{x}) = (f(x) - f(\hat{x})) + [(f(\hat{x}) - g(\hat{x})) + (g(\hat{x}) - \hat{g}(\hat{x}))] \\ &= (\text{propagated data error}) + [\text{computation error}] \\ &= (\text{propagated data error}) + [(\text{truncation error}) + (\text{rounding error})]\end{aligned}$$

- Taylor's Theorem
  - setup: integer $k \geq 1, a \in \mathbb{R}, f : \mathbb{R} \to \mathbb{R}$ is $k+1$ times differentiable on open interval and continuous on the closed interval between a and $x \in \mathbb{R}$, then $f(x) = t_k(x) + R_{k+1}(x) \approx t_k(x)$ where
    - $t_k(x) = f(a) + f'(a)(x-a) + \frac{f''(x)}{2!}(x-a)^2 + \cdots + \frac{f^{(k)}(a)}{k!}(x-a)^k$ is called Taylor polynomial of degree k
    - $R_{k+1}(x) \equiv \frac{f^{(k+1)}(\xi)}{(k+1)!}(x-a)^{k+1}$ is remainder, which is also the truncation error
  - alternative representation:
    $f(x+h) = f(x) + f'(x)h + f''(a)\frac{h^2}{2!} + \cdots + f^{(k)}(a)\frac{h^k}{k!} + f^{(k+1)}(\xi)\frac{h^{k+1}}{(k+1)!}$
  - typical analytic functions and respective Taylor's series
    - $e^x = 1 + x + \frac{x^2}{2!} + \cdots = \sum_{k=0}^{\infty} \frac{x^k}{k!}$
    - $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$
    - $\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots = \sum_{k=1}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!}$
    - $\log(x) = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3!} - \cdots = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{(x-1)^k}{k}$
    - $\log(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots = \sum_{k=1}^{\infty} (-1)^{k+1} \frac{x^k}{k}$
  - mostly used technique approximating functions by Taylor polynomials: Study for which k the remainder term (in abs. value) becomes bounded by the tolerance, for all x and ξ of interest. Then keep adding more terms until the difference in the resulting approximation to the function is small enough (or until it makes no difference, i.e. sum to saturation).

notes2def - Matrices and Vectors - Definitions

- matrix, order (size), square, row, column, equal, vector, sum/product of matrices
- properties, zero (null), identity, inverse, invertible (non-singular), singular (non-invertible)
- transpose, symmetic, orthogonal ($A^T A = I$), conjugate transpose, Hermitian, unitary, normal
- permutation matrix, elementary permutation, linearly independent, inner product, orthogonal, orthonormal

- diagonal, (strict) lower triangular, (strict) upper triangular, unit lower triangular, unit upper triangular, dense, density, sparse, sparsity
- banded matrix, full bandwidth, symmetrically banded, semi-bandwidth, tridiagonal matrix, pentadiagonal matrix, (row) diagonally dominant
- positive definite, non-negative matrix, monotone matrix, M-matrix, determinant
- system of m linear equations with n unknowns
- column (row) rank, rank, null space/kernel, nullity, range space

notes2ge - Gauss Elimination

- Gauss elimination (GE): general technique of transforming a given linear system to another mathematically equivalent to the first.
- 3 mainrow operations:
  - scalar multiplication of a row ρ_i with nonzero scalar $\rho_i \leftarrow \kappa\rho_i, \kappa \neq 0$
  - addition of rows $\rho_i \leftarrow \kappa\rho_i + \lambda\rho_j, \kappa \neq 0, \lambda \neq 0$
  - permutation of rows $\rho_i \leftrightarrow \rho_j$
- basically we want to transform the matrix (linear system) into upper triangular matrix.
- each step we pivot on a diagonal element ($a_{11}, a_{22}, \ldots$)
- How many operations are required?
  - about $\sum_{k=1}^{n-1}(n-k)^2 = \sum_{k=1}^{n-1} k^2 = \frac{n(n-1)(2n-1)}{6} \approx \frac{n^3}{3}$ pairs of additions and mult. (flops)
  - $\frac{n^2}{2}$ divisions
  - since divisions are much less than additions and multiplications, often ignore them in the operation counts
  - if matrix has particular properties, such as symmetry, bandedness, etc, then requires fewer operations
- Cost of solving a general linear system by Gauss elimination
  - Gauss elimination: $\frac{n^3}{3}$ pairs of additions and mult. (flops) and $\frac{n^2}{2}$ divisions
  - simultaneous processing of the RHS vector: $\frac{n^2}{2}$ pairs of additions and multiplications (flops)
  - back substitutions: $\frac{n^2}{2}$ pairs of additions and multiplications (flops) and n divisions
  - total: $\frac{n^3}{3} + \frac{2n^2}{2} flopsand$\frac{n^2}{2}+n$ divisions

notes2lu - LU factorization

- apply GE to A so we get an upper triangular matrix U.
- L is a unit lower triangular matrix (with diagonal 1s).
- $A = LU$
- elementary Gauss transformation, never stored individually.
  - unit lower triangular matrix
  - non-zero elements are 1's on the diagonal, and the elements of one column below the diagonal
- solution of a general linear system using the LU factorization, related cost
  - $Ax = b$, apply GE on A and obtain L and U
  - then the solution is reduced to the solutions of $Ly = b$ and $Ux = y$, so one forward and one backward substitution are required
  - cost
    - LU/GE: $\frac{n^3}{3}$ pairs of additions and multiplications (flops), and $\frac{n^2}{2}$ divisions
    - forward substitution: $\frac{n^2}{2}$ pairs of additions and mult. (flops) (the n divisions are not needed, since L has 1s on the main diagonal)

        - backward substitution: $\frac{n^2}{2}$ pairs of flops, and n divisions
        - total: $\frac{n^3}{3} + n^2$ flops and $\frac{n^2}{2} + n$ divisions
        - cost for solving m linear systems of size $n \times n$ with the same matrix: $\frac{n^3}{3} + mn^2$ flops and $\frac{n^2}{2} + mn$ divisions
            - apply GE/LU once, store the L and U factors, then apply a pair of f/s and b/s for each right-hand side vector
- properties of LU
    - L, U factors of the LU decomposition of a given A are unique.
    - symmetric matrices can reduce the work of LU factorization to $\frac{n^3}{6}$ flops.
    - the solution of an (l, u)-banded linear system by GE/LU and f/b/s requires $nlu + n(l + u)$ flops
    - the cost of computing the inverse of a matrix is n^3 flops

notes2piv - Pivoting

- when large number appears in L, U, could lead to instability in GE.
- pivoting in GE is a technique according to which rows (or columns or both) are interchanged, so that zero or very small in absolute value denominators in multipliers are avoided.
    - row pivoting: reorder rows of the matrix ($P_r A = LU, P_r$ permutation matrix)
    - column pivoting: reorder columns of the matrix ($AP_c = LU, P_c$ permutation matrix)
    - partial pivoting: row or column pivoting
    - complete pivoting: reorder both rows and columns of the matrix ($P_r A P_c = LU$)
    - symmetic pivoting: reorder both rows and columns of the matrix, but apply the same reordering to both rows and columns ($PAP^T = LU$)
- the most common one is row pivoting
- procedure: at the kth GE step, before the multipliers at column k, rows $k + 1, \cdots, n$ are computed ,a search along the kth column from row k to row n is performed, to identify the largest in abosulte value element. This element becomes the *pivot*. Assume the pivot belongs to row s, if $s \neq k$, rows k and s are interchanged. $ipiv(k) = s$.
- example: $Ax = b$ where,

$$k = 1, \left[\begin{array}{cccc:c} 1 & -2 & -4 & -3 & 2 \\ 2 & 0 & -1 & 2 & -1 \\ -1 & 2 & 2 & -1 & 4 \\ 3 & 0 & -3 & 6 & 9 \end{array}\right] \rightarrow \left[\begin{array}{cccc:c} 3 & 0 & -3 & 6 & 9 \\ 2 & 0 & -1 & 2 & -1 \\ -1 & 2 & 2 & -1 & 4 \\ 1 & -2 & -4 & -3 & 2 \end{array}\right], P_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$ipiv = [4, \cdot, \cdot]$$

$$\begin{aligned} \rho_2 &\leftarrow \rho_2 - 2/3\rho_1 \\ \rho_3 &\leftarrow \rho_3 + 1/3\rho_1 \\ \rho_4 &\leftarrow \rho_4 - 1/3\rho_1 \end{aligned}$$

$$\rightarrow \left[\begin{array}{ccccc:c} 3 & 0 & -3 & 6 & : & 9 \\ 2/3 & 0 & 1 & -2 & : & -7 \\ -1/3 & 2 & 1 & 1 & : & 7 \\ 1/3 & -2 & -3 & -5 & : & -1 \end{array}\right]$$

$$k = 2, \left[\begin{array}{ccccc:c} 3 & 0 & -3 & 6 & : & 9 \\ 2/3 & 0 & 1 & -2 & : & -7 \\ -1/3 & 2 & 1 & 1 & : & 7 \\ 1/3 & -2 & -3 & -5 & : & -1 \end{array}\right] \rightarrow \left[\begin{array}{ccccc:c} 3 & 0 & -3 & 6 & : & 9 \\ -1/3 & 2 & 1 & 1 & : & 7 \\ 2/3 & 0 & 1 & -2 & : & -7 \\ 1/3 & -2 & -3 & -5 & : & -1 \end{array}\right], P_2 = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right]$$

$$ipiv = [4, 3, \cdot]$$

$$\rho_3 \leftarrow \rho_3 - 0\rho_2$$
$$\rho_4 \leftarrow \rho_4 + 1\rho_2$$

$$\rightarrow \left[\begin{array}{ccccc:c} 3 & 0 & -3 & 6 & : & 9 \\ -1/3 & 2 & 1 & 1 & : & 7 \\ 2/3 & 0 & 1 & -2 & : & -7 \\ 1/3 & -1 & -2 & -4 & : & 6 \end{array}\right]$$

$$k = 3, \left[\begin{array}{ccccc:c} 3 & 0 & -3 & 6 & : & 9 \\ -1/3 & 2 & 1 & 1 & : & 7 \\ 2/3 & 0 & 1 & -2 & : & -7 \\ 1/3 & -1 & -2 & -4 & : & 6 \end{array}\right] \rightarrow \left[\begin{array}{ccccc:c} 3 & 0 & -3 & 6 & : & 9 \\ -1/3 & 2 & 1 & 1 & : & 7 \\ 1/3 & -1 & -2 & -4 & : & 6 \\ 2/3 & 0 & 1 & -2 & : & -7 \end{array}\right], P_3 = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right]$$

$$ipiv = [4, 3, 4]$$

$$\rho_4 \leftarrow \rho_4 + \frac{1}{2}\rho_3$$

$$\rightarrow \left[\begin{array}{ccccc:c} 3 & 0 & -3 & 6 & : & 9 \\ -1/3 & 2 & 1 & 1 & : & 7 \\ 1/3 & -1 & -2 & -4 & : & 6 \\ 2/3 & 0 & -1/2 & -4 & : & -4 \end{array}\right]$$

note: $A = LU$ no longer holds, but we have $PA = LU$ where $P = P_3P_2P_1$.

- there are 2 ways of solving $Ax = b$ on GE with pivoting.
    - first applies GEpiv to A and b simultaneously, and obtains an upper triangular matrix U and a transformed vector $c = b^{(n-1)}$ such that $Ax = b$ is equivalent to $Ux = c$, then applies back substitution to $Ux = c$ to compute x. In this case, the multipliers are computed, but do not need to be stored. Note that, when GEpiv is applied to A and b, both the row permutations and the elimination operations are applied to both A and b. The permutation matrix P does not need to be stored for the solution process.
    - second applies GEpiv to A, and obtains the L and U factors and the permutation matrix P, such that $PA = LU$, then applies f/s to $Lc = Pb$ to compute an intermediate vector c, and then applies b/s to $Ux = c$ to compute x. In this case, the multipliers are computed and stored in strictly lower triangular part of A. The permutation matrix P is not explicitly stored, but the vector $ipiv$ is, and from that the relevant information can be extended.
    - two ways are mathematically equivalent and involve the same computational cost.
    - however, when we need to solve several linear systems with the same matrix and different rhs vectors, we adopt the second way. apply GE/LU once, store the L and U and $ipiv$, then apply row interchanges and a pair of f/s and b/s to each rhs vector.
    - cost for solving m linear systems of size $n \times n$ with the same matrix: $\frac{n^3}{3} + m(\frac{n^2}{2} + \frac{n^2}{2}) = \frac{n^3}{3} + mn^2$ flops, $\frac{n^2}{2} + mn$ divisions, and $\frac{n^2}{2}$ comparisons.

- scaled partial pivoting: scale each equation so that the largest element in each row is equal to 1, then apply GEpiv...
    - cost: requires $n(n-1) \approx n^2$ comparisons and equal number of divisions in addition to the flops and comparisons required by the partial pivoting (no scaling). bottom line is that requires approximately the same amount of work as the non-pivoting algorithm ($n^3/3$)
- complete pivoting: requires $n^/3$ comparisons in addition to the flops required by the no-pivoting algorithm. (so twice in total)
    - accurate but expensive, so not commonly used!

notes2rm - Norms and condition numbers of matrices

- norm properties ($x, y \in S$):
    - $\|x\| \geq 0$, $\|x\| = 0$ iff $x = 0 \in S$
    - $\|\alpha x\| = \mid \alpha \mid \|x\|$ for all scalars α
    - $\|x + y\| \leq \|x\| + \|y\|$ (triangle or Minowski's inequality)
- vector norms
    - p-norm or Hölder norm: $\|x\|_p \equiv (\sum_{i=1}^n \mid x_i \mid^p)^{1/p}$
    - max (infinity norm): $\|x\|_\infty \equiv \max_{i=1}^n \{\mid x_i \mid\}$
    - Euclidean norm (lenth, L2, two-norm): $\|x\|_2 \equiv (\sum_{i=1}^n x_i^2)^{1/2}$
    - one-norm: $\|x\|_1 \equiv \sum_{i=1}^n \mid x_i \mid$
- matrix norm
    - p-norm (or induced or Hölder norm), $p = 1, 2, \infty$: $\|A\|_p \equiv \max_{x \neq 0} \{ \frac{\|Ax\|_p}{\|x\|_p} \}$
    - more properties:
        - $\|Ax\| \leq \|A\| \|x\|$
        - $\|I\| = 1$
        - $\|AB\| \leq \|A\| \|B\|$

- condition number of a non-singular matrix is $\kappa_a(A) = \|A\|_a \|A^{-1}\|_a$, the condition number of a singular matrix is ∞
  - condition number measures the relative sensitivity of the solution x of $Ax = b$ to relative changes in A and b.
  - $$\frac{\|x - \hat{x}\|_a}{\|x\|_a} \leq \kappa_a(A) \left(\frac{\|b - \hat{b}\|_a}{\|b\|_a} + \frac{\|A - \hat{A}\|_a}{\|A\|_a} + \frac{\|r\|_a}{\|b\|_a} \right)$$

    $r = \hat{b} - \hat{A}\hat{x}$ is the residual corresponding to the computed solution
  - $\kappa_a(A) \geq 1$ (amplification factor), $\kappa_a(I) = 1$
  - Well-conditioned matrix: $\kappa_a(A) \approx 1$
  - Ill-conditioned matrix: $\kappa_a(A) \gg 1$, $\kappa_a(A) \approx \epsilon_{mach}^{-1}$
  - geometrically, 2-dimensional space,two lines are parallel, the system has no solution, if they cross each other, has a unique solution. If they overlap, the system has infinitely many solutions. If they cross each other at a small angle (almost parallel lines), the system is ill-conditioned, that is, a small error in the input may give rise to a large error in the output.
  - notes:
    - $\frac{1}{n} \kappa_2(A) \leq \kappa_1(A) \leq n\kappa_2(A)$
    - $\frac{1}{n} \kappa_\infty(A) \leq \kappa_2(A) \leq n\kappa_\infty(A)$
    - $\frac{1}{n}^2 \kappa_1(A) \leq \kappa_\infty(A) \leq n^2\kappa_1(A)$

2016-04-25

topics to be covered

- √Nonlinear equation and solvers
- √Interpolation 1
- √Interpolation 2
- √Piecewise polynomial interpolation

notes4nl - Nonlinear equation and solvers

- general nonlinear systems: not necessarily square, the function f has m components and n variables, $f : \mathbb{R}^n \to \mathbb{R}^m$. Then the arising system has m and n unknowns, i.e. it's $m \times n$.
- fixed points, contractive functions
  - a point x is called fixed point of the function $g(x)$ when $x = g(x)$
  - a function $g : \mathbb{R}^n \to \mathbb{R}^n$ is called contractive in a set $S \subset \mathbb{R}^n$ if there exists a constant $0 \leq \lambda < 1$ such that $\|g(x) - g(z)\| \leq \lambda \|x - z\|, \forall x, z \in S$
  - if g is differentiable in S, and $n = 1$, the relation is $|g'(x)| \leq \lambda, \forall x \in S$. Then finding roots for f is equivalent to finding a fixed point of g.
  - existence and uniqueness of root and fixed point
    - Intermediate Value Theorem (IVP): If f is continuous in $[a, b]$ then for all (scalar) $\gamma \in (f(a), f(b))$, there exists $\xi \in (a, b)$ such that $f(\xi) = \gamma$
    - Mean Value Theorem (MVP): If f is continuous in $[a, b]$ and differentiable in (a, b), then there exists $\xi \in (a, b)$ such that $f'(\xi) = \frac{f(b) - f(a)}{b - a}$
  - theorems for existence and uniqueness of root and fixed point
    - Theorem 1 (Bolzano) Existence of a root of $f(x)$: If $f(x)$ is continuous in $[a, b], f(a) \cdot f(b) < 0$, then there exists at least one root of $f(x)$ in (a, b).
    - Theorem 2 Existence of a fixed point of $g(x)$: If $g(x)$ is continous $I = [a, b], g(x) \in I, \forall x \in I$ (i.e. $g(x)$ maps I to itself), then there exists at least one fixed point of $g(x)$ in I.
    - Theorem 3 Uniqueness of a root of $f(x)$: If $f(x)$ is differentiable in $I = (a, b), f'(x) \neq 0, \forall x \in I$, there exists a root of $f(x)$ in I, then the root is unique in I.
    - Theorem 4 Existence and uniqueness of a fixed point of $g(x)$: If g is contraction in $I = [a, b], g(x) \in I, \forall x \in I$ (i.e. $g(x)$ maps I to itself), then there exists a unique fixed point x^* of $g(x)$ in I.
- nonlinear solver, iterative method, stopping criterion
- the bisection method:
  - requires only one function evaluation per iteration
  - interval shrinks a half each time
  - always converges
  - converges linearly. slower than Newton's and slower than secant, but is useful for computing initial guesses for either Newton's or the secant method

- if several roots exist in $[L, R]$, no guarantee which the method converges to
- fixed-point (functional) iteration methods
  - Theorem 4b (extension of Thm 4) Convergence of fixed-point iteration $x^{(k+1)} = g(x^{(k)})$: If g is continuous in $I = [a, b]$ with constant λ, $g(x) \in I, \forall x \in I$, then there exists a unique fixed point x^* of g in I, $\forall x^{(0)} \in I$, the iteration scheme $x^{(k+1)} = g(x^{(k)})$ converges to $x^*, \mid x^{(k)} - x^* \mid \leq \lambda^k \mid x^{(0)} - x^* \mid \leq \lambda^k \max\{x^{(0)} - a, b - x^{(0)}\}$
  - Theorem 5 Convergence of fixed-point iteration $x^{(k+1)} = g(x^{(k)})$: If $g(x)$ has a fixed point x^*, $g(x)$ has continuous derivative in an open interval containing x^*, $|g'(x^*)| < 1$, then there exists an open interval I that contains x^*, i.e. $I = (x^* - r^*, x^* + r^*), r^* > 0$, such that $\forall x^{(0)} \in I$, the iteration scheme $x^{(k+1)} = g(x^{(k)})$ converges to x^*
- rate of convergence of fixed-point iteration
  - Theorem 6: If g has a fixed point α, $x^{(k+1)} = g(x^{(k)})$ converges to α, $g \in \mathbb{C}^\beta$ near α, $g'(\alpha) = g''(\alpha) = \cdots = g^{(\beta-1)}(\alpha) = 0$ and $g^{(\beta)}(\alpha) \neq 0$, then the rate of convergence of $x^{(k+1)} = g(x^{(k)}$ is β and the asymptotic error constant is $\frac{1}{\beta!} \mid g^{(\beta)}(\alpha)|$
- Newton's method
  - $$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$
  - if we define $g(x) = x - \frac{f(x)}{f'(x)}$ then Newton's iteration becomes $x^{(k+1)} = g(x^{(k)})$, which takes the form of a fixed-point iteration.
  - note that f must be differentiable and $f'(x^{(k)} \neq 0$
  - does not always converge. may converge when started at a certain initial guess, but may diverge if started at another initial guess. However, always converges if f is twice differentiable and $x^{(0)}$ is chosen close enough to the root
  - when Newton's method converges, usually converges quadratically (rate 2).
  - Newton's method converges slowly close to a multiple root
  - if f has several roots, no guarantee as to which of the roots Newton's converges to
- Modified Newton's:
  - $$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(0)})}$$
  - converges much slower, need more iterations
- the secant method
  - $$x^{(k+1)} = x^{(k)} - f(x^{(k)}) \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}$$
  - the secant iteration is a two-step scheme. thus requires 2 initial guesses.
  - not always converges, one function evaluation eac htime
  - when converges, rate $\beta = 1.618$ which is the positive root of $x^2 - x - 1 = 0$
- Newton's method for systems of nonlinear equations
  - $f : \mathbb{R}^n \to \mathbb{R}^n$ be a differentiable function and let $\bar{f}(\bar{x}) = \bar{0}$ be the system to be solved.
  - Jacobian matrix J, $(J(\bar{x}))_{ij} = \frac{\delta f_i}{\delta x_j}$
  - $\bar{x}^{(k+1)} = \bar{x}^{(k)} - J^{-1}(\bar{x}^{(k)})\bar{f}(\bar{x}^{(k)})$
  - Newton's method for systems requires the evaluation of a function of n variables (f) and the

computation of the Jacobian matrix J at each iteration. The computation of J requires n^2 function evaluations $(\delta f_i/\delta x_j)$ each of n variables
    - futhermore, requires the solution of an $n \times n$ linear system at each iteration, costs about $n^3/3$ flops.
    - does not always converge, if converges then quadratically
- final comparison

method property	bisection	newton	secant
guaranteed convergence	y	n	n
convergence rate	1	2	1.618
no. of func. evals per it	1	2 (f, f')	1
requirements for application	continuity, change of sign	differentiability, $f'(x^{(k)} \neq 0$	continuity, $f(x^{(k)}) \neq f(x^{(k-1)})$
no. of initial guesses	2	1	2
fixed point iter. method	n	y	n
extendable to nonlin. systems	no	Newton's	Broyden's
no. of func. evals per it	/	$n^2 + n$	n
flops per iter	/	$O(n^3)$	$O(n^2)$

notes5int - Interpolation 1

- def: say $g(x)$ interpolates $f(x)$ at points $x_i, i = 0, \cdots, n$ if $g(x_i) = f(x_i)$, then g is called the interpolant or interpolating function of the function f at points x_i. when $g(x)$ interpolates $f(x)$ at certain points, then $f(x)$ interpolates $g(x)$ at the same points.
    - interpolation is not the only type of approximation possible. other types like least squares approximation is also possible
    - given data, no unique interpolation (unless given some restraints)
- Weierstrass Thm: If $f(x)$ is a continuous function on $[a, b]$ then for every $\epsilon < 0$ there exists a polynomial $p_n(x)$ of degree $n = n(\epsilon)$ s.t. $\max_{x \in [a,b]} \mid f(x) - p_n(x) \mid < \epsilon$
- polynomial interpolation with monomial basis
    - e.g., data $(0, 5), (-1, 7), (2, 13)$, so $n = 2$, degree at most 2.
    - $p_2(x) = a_0 + a_1 x + a_2 x^2$:
        - $p_2(x_0) = f_0 \implies a_0 + a_1 x_0 + a_2 x_0^2 = f_0 \implies a_0 + a_1 \cdot 0 + a_2 \cdot 0 = 5$
        - $p_2(x_1) = f_1 \implies a_0 + a_1 x_1 + a_2 x_1^2 = f_1 \implies a_0 + a_1 \cdot (-1) + a_2 \cdot 1 = 7$
        - $p_2(x_2) = f_2 \implies a_0 + a_1 x_2 + a_2 x_2^2 = f_2 \implies a_0 + a_1 \cdot 2 + a_2 \cdot 4 = 13$
- the function `polyfit(xi, yi, n)` finds the coefficients of a polynomial of degree at most `n` that fits the data vectors `xi` and `yi`. In general, the func constructs the least squares approximation of the data. When `n` is set to the number of data minus 1, then the approximation reduces to interpolation.

notes5int2 - Interpolation 2

- polynomial interpolation with Lagrange basis
  - interpolates the data $(x_i, f_i), i = 0, \cdots, n$. instead of using monomials $1, x, x^2, \cdots, x^n$ as basis functions ,we use the Larange basis functions $l_0^{(n)}(x), l_1^{(n)}(x), l_2^{(n)}(x), \cdots, l_n^{(n)}(x)$ defined by:
    - $$l_j^{(n)}(x) \equiv \frac{\Pi_{i=0, j\neq j}^{n}(x - x_i)}{\Pi_{i=0, i\neq j}^{n}(x_j - x_i)}$$

      having the property $l_j^{(n)}(x_i) = 1$ if $i = j$, 0 o.w.
    - e.g. data $(0, 5), (-1, 7), (2, 13)$, so $n = 2$, degree at most 2.
    - $$\begin{aligned} p_2(x) &= f_0 l_0(x) + f_1 l_1(x) + f_2 l_2(x) = 5l_0(x) + 7l_1(x) + 13l_2(x) \\ &= 5\,\frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + 7\,\frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + 13\,\frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \\ &= 5\,\frac{(x + 1)(x - 2)}{(0 + 1)(0 - 2)} + 7\,\frac{(x - 0)(x - 2)}{(-1 - 0)(-1 - 2)} + 13\,\frac{(x - 0)(x + 1)}{(2 - 0)(2 + 1)} \\ &= -5\,\frac{(x + 1)(x - 2)}{2} + 7\,\frac{x(x - 2)}{3} + 13\,\frac{x(x + 1)}{6} \end{aligned}$$
- existence and uniqueness of the interpolating polynomial
  - Theorem: Given data $(x_i, f_i), i = 0, \cdots, n$ with x_i being distinct, there exists a unique polynomial of degree at most n interpolating the data.
- polynomial interpolation with Newton's basis
  - $p_1(x) = f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0)$
  - e.g. data $(0, 5), (-1, 7), (2, 13)$, so $n = 2$, degree at most 2.
  - (NDD table) $\begin{bmatrix} 0 & 5 & & \\ & & -2 & \\ -1 & 7 & & 2 \\ & & 2 & \\ 2 & 13 & & \end{bmatrix}$, note: -2=(5-7)/(0--1)
  - $p_2(x) = 5 - 2(x - 0) + 2(x - 0)(x + 1)$
- the comparison of 3 bases
  - monomials: $1, x, x^2, \cdots, x^n$
  - $p_n(x)$ is efficiently evaluated (Horner's rule) and easy to differentiate and integrate
  - computing the coefficients a_i requires solving a linear system $Ba = f$ of size $n + 1$ where B is dense, and ill-conditioned for large n
  - adding a piece of data requires recomputation of all a_i's
  - Lagrange basis
  - $p_n(x)$ is messy to compute and even messier to differentiate and integrate
  - computing the coefficients a_i is trivial: $a_i = f_i$
  - adding a piece of data requires updating all $l_j(x)$'s
  - Newton basis
  - $p_n(x)$ is efficiently evaluated but messy to differentiate and integrate
  - computing a_i is simple, with NDD, is equivalent to solving a triangular linear system (f/s) of size

$n+1$
    - adding a piece of data is easy
- error in poly interpolation
    - spread of a set of points $\{s_1, s_2, \cdots, s_N\}$ is $spr\{s_1, \cdots, s_N\}$ is the smallest interval containing all s_i's. The open spread is the largest open interval which is a subset of $spr\{s_1, s_2, \cdots, s_N\}$, denoted as $ospr\{s_1, \cdots, s_N\}$. If ordered, then $spr\{s_i\} = [s_1, s_N], ospr\{s_i\} = (s_1, s_N)$
    - Theorem (error in poly interpo): If $f(x)$ has $n+1$ continuous derivatives and $p_n(x)$ is the polynomial of degree at most n interpolating f at distinct points $x_0, x_1, \cdots, x_n$, then for any x,
    - $$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \Pi_{j=0}^{n}(x - x_j)$$

      where ξ is an unknown point in $ospr\{x_0, x_1, \cdots, x_n, x\}$ that depends on x
    - remarks
        - if f happens to be a poly of degree at most n, since $f^{(n+1)} = 0$, the error is 0
        - the error of the interpolant is 0 on the data points x_i
        - the error formula consists of 2 main parts, one is $f^{(n+1)}(\xi)$, the other is $W(x) = \Pi_{j=0}^{n}(x - x_j) = (x - x_0)(x - x_1)\cdots(x - x_n)$
        - ξ is unknown, so $f^{(n+1)}(\xi)$ is unknown
        - the value of $W(x)$ can be calculated for a particular x
- pitfalls of poly interpo
    - good approx of $f(x)$ of nth degree, need a fairly large n, but high degree always exhibit an oscillatory behaviour towards the endpoints.
    - high degree poly construction leads to ill-conditioned computations
    - also often sensitive to small perturbations of the coefficients so small error ...
    - higher cost than $O(n^2)$
    - piecewise poly interpolation is a good alternative, that resolves the above problems:
        - avoids oscillatory behaviour for large n
        - guarantees that the error decreases as n increases
        - often leads to well-conditioned matrices
        - requires $O(n)$ flops for construction and $O(1)$ flops for evaluation per data point

notes5pp - Piecewise polynomial interpolation

- def: let $\Delta = \{a = x_0, x_1, \cdots, x_n = b\}$ be a set of distinct points, called knots or nodes or breakpoints or gridpoints, paritiioning $I = [a, b]$ into n subintervals.
- a piecewise polynomial (pp) $p(x)$ of degree N w.r.t. the knots $x_i, i = 0, \cdots, n$, is a poly of degree (at most) N on each interval $(x_{i-1}, x_i), i = 1, \cdots, n$. A polynomial of degree N is always a pp of degree N but the opposite is not always true.
- A pp $p(x)$ of degree N w.r.t. the knots x_i is called a spline (of degree N w.r.t the knots x_i) if $p(x)$ is continuous on the knots and has continuous derivatives up to some order on the knots
- A pp $p(x)$ of degree N w.r.t. Δ is always written as a function with n branches
- error of the linear spline interpolant: it can be shown that the error of the linear spline interpolant $L(x)$ of a function $f(x) \in \mathbb{C}^2$ satisfies
    - $$|f(x) - L(x)| \leq \frac{1}{8} \max_{x \in [a,b]} |f''(x)| \max_{i=1,\cdots,n} (x_i - x_{i-1})^2$$

for all $x \in [a, b]$
    - if we doulbe n (twice as many data points), the error bound decreases by a factor of 4.
- linear splines have angles, not as smooth and visually pleasing as we would like., use cubic splines
- error of the cubic spline interpolant: it can be shown that the error of the clamped cubic spline interpolant $C(x)$ of a function $f(x) \in \mathbb{C}^4$ satisfies
    - $$|f(x) - C(x)| \leq \frac{5}{384} \max_{x \in [a,b]} |f^{(4)}(x)| \max_{i=1,\cdots,n} (x_i - x_{i-1})^4$$

      for all $x \in [a, b]$
    - double n, the error bound decreases by a facotr of 16.

2016-04-26

notessum - Summary

computer arithmetic

- Simplified form for presenting a floating-point number x in **base** b: $x = (f)_b \times b^{(e)_b}$
- $f = \pm(d_1 d_2 \cdots d_t)_b$ **mantissa** (or **significant**); $0 \leq f < 1$.
- $e = \pm(c_{s-1}\, c_{s-2} \cdots c_0)_b$ integer **exponent** (or **characteristic**); $E_{min} \leq e \leq E_{max}$
- A computer in which numbers are represented as above is said to have t base-b digits **precision**. The exponent goveners the **range** of representable numbers.
- *Term*: **normalized** mantissa, **significant** digits, OFL (overflow level, N_{max}), UFL (underflow level, N_{min}), **overflow**, **underflow**
- The real numbers that are exactly representable as floating-point numbers are discrete, belong to $[-N_{max}, -N_{min}] \cup \{0\} \cup [N_{min}, N_{max}]$, and are denser towards 0.
- For the rest of the real numbers:
  - those in $[-N_{max}, -N_{min}] \cup [N_{min}, N_{max}]$ are rounded/chopped (x represented as $fl(x)$);
  - those in $(-\infty, -N_{max}) \cup (N_{max}, \infty)$ overflow;
  - those in $(-N_{min}, 0) \cup (0, N_{min})$ underflow.
- **Relative round-off error (representation error)**: $\delta = \frac{fl(x)-x}{x}$
- Bound for δ: $|\delta| \leq \frac{1}{2} b^{1-t}$, i.e. $fl(x)$ is correct in t significant b-digits.
- *Terms*: **absolute** error, **relative** error
- All computer operations are designed so that **the error of a computation is the round-off error of the correct result.**
- **Machine epsilon** ϵ_{mach} : The smallest (non-normalised) floating-point number with the property $1 + \epsilon_{mach} > 1$.
- Bounds for ϵ_{mach} and δ:
  - $\frac{1}{2} b^{1-t} \leq \epsilon_{mach} \leq b^{1-t}$
  - $-\epsilon_{mach} \leq \delta \leq \epsilon_{mach}$
  - $0 < UFL < \epsilon_{mach} < OFL$.
- **Saturation**: The phenomenon in which a non-zero number is added to another and the latter is left unchanged.
- **Catastrophic cancellation**: The phenomenon in which adding nearly opposite (or subtracting nearly equal) numbers results in having no (or very few) correct digits (i.e. the past errors in the nearly opposite or nearly equal numbers, propagate from the least significant digits to the most significant ones).
- **Condition number** of a function: $\kappa_f(x) = |\frac{xf'(x)}{f(x)}|$; measure of how the error in x propagates in $f(x)$ (error propagation in computation)
  - | relative forward error | ≈ condition number × |relative backward error|
  - $$\frac{y - \hat{y}}{y} \approx \kappa_f \frac{x - \hat{x}}{x}, y = f(x)$$

    - It does not change by re-writing a certain computation in an equivalent way.
- **Stability** refers to the error propagation in a numerical algorithm, i.e. the particular way a certain computation is carried out, and may change if the computation is performed by a different algorithm.

matrices, solving square linear systems

- *Matrix terminology:* triangular (upper/lower), unit triangular (upper/lower), tridiagonal, (*l*,*u*)-banded, permutation, elementary Gauss transformation, orthogonal, positive definite, diagonally dominant, symmetric
- *Important algorithms:* Gauss elimination, Gauss elimination with partial pivoting (applicable to all matrices ,gives L unit lower triangular, U upper triangular and P permutaion, such that $PA = LU$), back substituion, forward substitution
- *Important flops counts:*

matrix	**LU**	**LU piv(part)**	**LU piv compl.**	**f/s**	**b/s**	**sol of m sys.**	**inverse**
general $n \times n$	$\frac{n^3}{3}$	$\frac{n^3}{3}$	$\frac{2n^3}{3}$	$\frac{n^2}{2}$	$\frac{n^2}{2}$	$\frac{n^3}{3} + mn^2$	n^3
(*l*,*u*)-banded	nlu	$2nlu$	-	ln	un	$2nlu + m(l+u)n$	$n^2(l+u)$
symmetric	$\frac{n^3}{6}$	-	-	-	-	-	-

- General $n \times m$ matrix times $m \times k$ matrix (or vector for $k = 1$): nmk

norms, condition numbers

- Definition of a norm: three properties
    - $\|x\| \geq 0, \|x\| = 0$ iff $x = 0 \in S$
    - $\|\alpha x\| = \mid \alpha \mid \|x\|$ for all scalars α
    - $\|x + y\| \leq \|x\| + \|y\|$ (triangle or Minowski's inequality)
- Common *vector norms*:
    - max (infinity) $\|x\|_\infty \equiv \max_{i=1}^n \{\mid x_i \mid\}$
    - Euclidean (2-norm) $\|x\|_2 \equiv \sqrt{(x,x)} \equiv (\sum_{i=1}^n x_i^2)^{1/2}$
    - one-norm $\|x\|_1 \equiv \sum_{i=1}^n \mid x_i \mid$
- For any vector $x \in \mathbb{R}^n$, we have $\|x\|_\infty \leq \|x\|_2 \leq \|x\|_1$.
- Comman *matrix norms*:
    - p-norms $\|A\|_p \equiv \max_{x \neq 0} \{\frac{\|Ax\|_p}{\|x\|_p}\}, p = 1, 2, \infty$
    - row norms $\|A\|_\infty = \max_{i=1}^n \{\sum_{j=1}^n \mid a_{ij} \mid\}$
    - col norms $\|A\|_1 = \max_{j=1}^n \{\sum_{i=1}^n \mid a_{ij} \mid\}$
- For the p-norms, another three properties hold
    - $\|Ax\| \leq \|A\|\|x\|$
    - $\|I\| = 1$
    - $\|AB\| \leq \|A\|\|B\|$
- **Condition number** of a non-singular matrix A: $\kappa_a(A) = \|A\|_a \|A^{-1}\|_a$. It is a measure of the relative sensitivity of the solution x of $Ax = b$ to relative changes in A and b:

- $$\frac{\|x-\hat{x}\|_a}{\|x\|_a} \leq \kappa_a(A)\left(\frac{\|b-\hat{b}\|_a}{\|b\|_a}+\frac{\|A-\hat{A}\|_a}{\|A\|_a}+\frac{\|r\|_a}{\|b\|_a}\right)$$

nonlinear equations -- terms and concepts

- General form: $f(x)=0$
- **Multiplicity** of root: If $f \in \mathbb{C}^m$ and $f(x^*)=0, f'(x^*)=0, \cdots, f^{(m-1)}(x^*)=0$, but $f^{(m)}(x^*) \neq 0$, for some $m \geq 1$, then x^* is a root of multiplicity m.
- General form of a $n \times n$ system of nonlinear equations: $f: \mathbb{R}^n \to \mathbb{R}^n, \bar{f}(\bar{x})=\bar{0}$
- **Jacobian** matrix: $(J(\bar{x}))_{ij} = \frac{df_i}{dx_j}(\bar{x})$
- **Fixed point** x of function $g(x)$: $x=g(x)$.
- **Contraction** in set $S \subset \mathbb{R}^n$: a function $g: \mathbb{R}^n \to \mathbb{R}^n$ for which there is constant λ, with $0 \leq \lambda < 1$, such that
  - $$\|g(x)-g(z)\| \leq \lambda \|x-z\|, \forall x, z \in S. \ (1)$$
- 4 theorems about existence and uniqueness or roots or fixed pts.
  - Theorem 1 (Bolzano) Existence of a root of $f(x)$: If $f(x)$ is continuous in $[a,b], f(a) \cdot f(b) < 0$, then there exists at least one root of $f(x)$ in (a,b).
  - Theorem 2 Existence of a fixed point of $g(x)$: If $g(x)$ is continous $I=[a,b]$, $g(x) \in I, \forall x \in I$ (i.e. $g(x)$ maps I to itself), then there exists at least one fixed point of $g(x)$ in I.
  - Theorem 3 Uniqueness of a root of $f(x)$: If $f(x)$ is differentiable in $I=(a,b), f'(x) \neq 0, \forall x \in I$, there exists a root of $f(x)$ in I, then the root is unique in I.
  - Theorem 4 Existence and uniqueness of a fixed point of $g(x)$: If g is contraction in $I=[a,b]$, $g(x) \in I, \forall x \in I$ (i.e. $g(x)$ maps I to itself), then there exists a unique fixed point x^* of $g(x)$ in I.

numerical methods for solving nonlinear equations

- General nonlinear solver
  - guess $x^{(0)}$ (and possibly $x^{(-1)}$ or more)
  - for $k=1, \cdots,$ maxit
    - compute $x^{(k)}$ using previous approximations and information from f
    - if stopping criterion satisfied exit, endif
  - endfor

rate of convergence of sequences / iterative methods

- Sequence $x^{(0)}, x^{(1)}, \cdots$ converges to x^*, with rate of convergence β and asymptotic error constant C:
  - $$\lim_{k \to \infty} \frac{\|x^{(k+1)}-x^*\|}{\|x^{(k)}-x^*\|^\beta} = C$$
- See Theorem 6 about rate of convergence of fixed-pt iteration methods.

numerical methods for solving nonlinear equations

- *Bisection* method

    - Bisection is applicable when f is continuous and there are two points L and R ($L < R$), such that $f(L)f(R) < 0$. Bisection chooses $M = L + (R - L)/2$ as next approximation and continuous to $[L, M]$ if $f(L)f(M) < 0$, or to $[M, R]$ if $f(M)f(R) < 0$. Bisection always converges (when applicable) and is of first order. Requires one function evaluation per iteration.
- *Newton's* method
    - Newton's is applicable if f is differentiable and $f'(x^{(k)}) \neq 0$:
        - $$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$$
    - Converges if f is twice differentiable and if $x^{(0)}$ *close enough* to root. When it converges, it is usually of second order. Requires one function and one derivative evaluation per iteration.
- General form of *fixed-point iteration*
    - Assume we have found (or been given) function $g(x)$ such that
        - $$f(x) = 0 \iff x = g(x)$$
    - Then, the associated fixed-point iteration method computes
        - $$x^{(k+1)} = g(x^{(k)})$$
    - Newton's is a fixed-point iteration method with $g(x) = x - f(x)/f'(x)$.
    - 3 theorems convergence (and rate) of fixed-point iteration methods
        - Theorem 4b (extension of Thm 4) Convergence of fixed-point iteration $x^{(k+1)} = g(x^{(k)})$: If g is continuous in $I = [a, b]$ with constant λ, $g(x) \in I, \forall x \in I$, then there exists a unique fixed point x^* of g in I, $\forall x^{(0)} \in I$, the iteration scheme $x^{(k+1)} = g(x^{(k)})$ converges to $x^*, \mid x^{(k)} - x^* \mid \leq \lambda^k \mid x^{(0)} - x^* \mid \leq \lambda^k \max\{x^{(0)} - a, b - x^{(0)}\}$
        - Theorem 5 Convergence of fixed-point iteration $x^{(k+1)} = g(x^{(k)})$: If $g(x)$ has a fixed point x^*, $g(x)$ has continuous derivative in an open interval containing x^*, $|g'(x^*)| < 1$, then there exists an open interval I that contains x^*, i.e. $I = (x^* - r^*, x^* + r^*), r^* > 0$, such that $\forall x^{(0)} \in I$, the iteration scheme $x^{(k+1)} = g(x^{(k)})$ converges to x^*
        - Theorem 6: If g has a fixed point α, $x^{(k+1)} = g(x^{(k)})$ converges to α, $g \in \mathbb{C}^\beta$ near α, $g'(\alpha) = g''(\alpha) = \cdots = g^{(\beta-1)}(\alpha) = 0$ and $g^{(\beta)}(\alpha) \neq 0$, then the rate of convergence of $x^{(k+1)} = g(x^{(k)}$ is β and the asymptotic error constant is $\frac{1}{\beta!} \mid g^{(\beta)}(\alpha)|$
- *Secant* method
    - Secant is applicable if f is continuous and $f(x^{(k)}) \neq f(x^{(k-1)})$:
        - $$x^{(k+1)} = x^{(k)} - f(x^{(k)}) \frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}$$
    - Secant does not always converge; when it converges it is of order approximately 1.6. Requires one function evaluation per iteration.
- *Newton's* for systems
    - Newton's is applicable if all components of f are differentiable (w.r.t. each variable) and $J(\bar{x}^{(k)})$ is nonsingular:
        - $$\bar{x}^{(k+1)} = \bar{x}^{(k)} - J^{-1}(\bar{x}^{(k)})\bar{f}(\bar{x}^{(k)})$$

polynomial interpolation

- *Generic interpolation problem*: Given data $(x_i, y_i), i = 0, \cdots, n$, find a function (possibly a polynomial) $p(x)$, that interpolates the data, i.e. $p(x_i) = y_i, i = 0, \cdots, n$.
- *Existence and uniqueness theorem*: Given data $(x_i, y_i), i = 0, \cdots, n$, with x_i distinct, there exists a unique polynomial $p_n(x)$ of degree n or less that interpolates the data.
- *Polynomial interpolation error formula*: Let $p_n(x)$ be the polynomial of degree n or less that interpolates f at $x_i, i = 0, \cdots, n$. If f has $n + 1$ continuous derivatives, then, for any x,
  - $$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \Pi_{i=0}^{n}(x - x_i),$$
  - for some $\xi \in \text{ospr}\{x, x_0, x_1, \cdots, x_n\}$, that depends on x.
  - Notes: If $a = \min_i\{x_i\}$ and $b = \max_i\{x_i\}$, then $\text{ospr}\{x_0, x_1, \cdots, x_n\} = (a, b)$.
  - If $a \leq x \leq b$, $\text{ospr}\{x, x_0, x_1, \cdots, x_n\} = (a, b)$.
  - If $x < a$, $\text{ospr}\{x, x_0, x_1, \cdots, x_n\} = (x, b)$.
  - If $b < x$, $\text{ospr}\{x, x_0, x_1, \cdots, x_n\} = (a, x)$.

construction of polynomial interpolants

- *Properly posed polynomial interpolation problem*: Given data $(x_i, y_i), i = 0, \cdots, n$, with x_i distinct, find the (unique) polynomial $p_n(x)$ of degree n or less that interpolates the data.
- Three ways of constructing:
  - (1) Monomials, Vandermonde matrix: easy for small n, very inefficient and inaccurate (unstable) for large n, construction requires the solution of an $(n + 1) \times (n + 1)$ dense and ill-conditioned linear system, easy to evaluate (Horner's), differentiate and intergrate.
  - (2) Lagrange basis functions, explicit formula: often convenient for mathematical manipulation, easy to construct, not very stable, reasonably easy to evaluate, hard to differentiate and integrate.
  - (3) Newton's basis functions, recursive algorithm, NDD table: reasonably efficient to construct, stable, easy to evaluate (Horner's), relatively hard to differentiate and integrate, adding data is easy.

problems with polynomial interpolation

- Polynomial interpolants of high degree
  - oscillate a lot close to the endpoints;
  - lead to ill-conditioned computations;
  - are sensitive to small perturbations of the coefficients;
  - are costly to construct (NDD table takes $O(n^2)$ time for n data points) and evaluate (the evaluation on one point takes $O(n)$ time, while the number of evaluation points is often much larger than the number of data points).
- Polynomial interpolants do not guarantee that the error decreases as the number of data increases.
- Chebyshev data points minimize the $\Pi_{i=0}^{n}(x - x_i)$ part of the error among all choices of $n + 1$ data points in a given interval, and partly only address the oscillatory behaviour problem.
- Piecewise polynomial interpolation is a good alternative, that resolves all the above problems.

piecewise polynomial interpolation

- Piecewise polynomial of degree k defined w.r.t. knots $x_i, i = 0, \cdots, n$:

- $$p(x) = \begin{cases} a_{01} + a_{11}x + a_{21}x^2 + \cdots + a_{k1}x^k \text{ for } x_0 \leq x < x_1 \\ a_{02} + a_{12}x + a_{22}x^2 + \cdots + a_{k2}x^k \text{ for } x_1 \leq x < x_2 \\ \cdots \cdots \\ a_{0n} + a_{1n}x + a_{2n}x^2 + \cdots + a_{kn}x^k \text{ for } x_{n-1} \leq x < x_n \end{cases}$$

- There are $n(k+1)$ coefficients -- free parameters, degrees of freedom -- in the representation of p. Piecewise polynomials may not be continuous or may not have continuous derivatives of some order on the knots.
- By imposing continuity conditions on the interior knots, some coefficients are no longer free parameters.
- If we impose continuity conditions up to derivatives of order $k-1$, we get splines.
- Splines have $n(k+1) - (n-1)k = n + k$coefficients -- free parameters, degrees of freedom.
- Example: **Linear splines** are piecewise polynomial of degree 1 which are $\mathbb{C}^0$ continuous on the knots $x_i, i = 1, \cdots, n-1$, and have $n+1$ degrees of freedom.
- Example: **Cubic splines** are piecewise polynomial of degree 3 which are $\mathbb{C}^2$ continous on the knots $x_i, i = 1, \cdots, n-1$, and have $n+3$ degrees of freedom.

linear splines

- A **linear splines** $L(x)$ takes the form:
  - $$L(x) = \begin{cases} L_1(x) & \equiv a_{01} + a_{11}x \text{ for } x_0 \leq x < x_1 \\ L_2(x) & \equiv a_{02} + a_{12}x \text{ for } x_1 \leq x < x_2 \\ & \cdots \cdots \\ L_n(x) & \equiv a_{0n} + a_{1n}x \text{ for } x_{n-1} \leq x < x_n \end{cases}$$
  - and satisfies the $n-1$ *continuity conditions*
  - $$L_i(x_i) = L_{i+1}(x_i), i = 1, \cdots, n-1. (A)$$
- The **linear spline interpolant** $L(x)$ of data $(x_i, y_i), i = 0, \cdots, n$, is the linear spline (i.e. a pp of degree 1 satisfying (A)), satisfying the *interpolation conditions*
  - $$L(x_i) = y_i, i = 0, \cdots, n. (B)$$
- The linear spline interpolant $L(x)$ of data $(x_i, y_i), i = 0, \cdots, n$, is given by
  - $$L(x) = \left\{ y_{i-1} \frac{x - x_i}{x_{i-1} - x_i} + y_i \frac{x - x_{i-1}}{x_i - x_{i-1}} \right. \text{ for } x_{i-1} \leq x \leq x_i, i = 1, \cdots, n. (C)$$
- Thus, $L(x)$ is given by a ready-to-evaluate formula. (No construction cost.) The evaluation of a linear spline requires $O(1)$ time for one evaluation point.

cubic splines

- A **cubic spline** $C(x)$ takes the form:
  - $$C(x) = \begin{cases} C_1(x) & \equiv a_{01} + a_{11}x + a_{21}x^2 + a_{31}x^3 \text{ for } x_0 \leq x < x_1 \\ C_2(x) & \equiv a_{02} + a_{12}x + a_{22}x^2 + a_{32}x^3 \text{ for } x_1 \leq x < x_2 \\ & \cdots \cdots \\ C_n(x) & \equiv a_{0n} + a_{1n}x + a_{2n}x^2 + a_{3n}x^3 \text{ for } x_{n-1} \leq x < x_n \end{cases}$$

    - and satisfies the $3(n-1)$ *continuity conditions*
    - $$C_i(x_i) = C_{i+1}(x_i), i = 1, \cdots, n-1, (0a)$$
      $$C_i'(x_i) = C_{i+1}'(x_i), i = 1, \cdots, n-1, (0b)$$
      $$C_i''(x_i) = C_{i+1}''(x_i), i = 1, \cdots, n-1, (0c)$$
- A **cubic spline interpolant** $C(x)$ of data $(x_i, y_i, i = 0, \cdots, n,$ is a cubic spline (i.e. a pp of degree 3 satisfying (0a)-(0b)-(0c)), satisfying the *interpolation conditions*
    - $$C(x_i) = y_i, i = 0, \cdots, n,$$
    - and a set of two other conditions, referred to as *end-conditions.*
- Typical sets of end-conditions
    - *Clamped (derivative) end-contions:*
        - $$C'(x_i) = y_i', i = 0, n, (2a)$$
        - if $y_i', i = 0, n,$ are given.
    - *Approximate clamped (derivative) end-conditions:*
        - $$C'(x_i) = \text{ approximation to } y_i', i = 0, n, (2b)$$
        - if $y_i', i = 0, n,$ are not given (the approximations to y_i' can be formed by cubic polynomial interpolation based on 4 endpoint values).
    - *Natural end-conditions:*
        - $$C''(x_i) = 0, i = 0, n. (2c)$$
    - *Not-a-knot end-conditions:*
        - $$C''' \text{ continuous on } x_i, i = 1, n-1. (2d)$$
    - These are equivalently written as
        - $$C_i'''(x_i) = C_{i+1}'''(x_i), i = 1, n-1.$$
    - Note: only **one** of these sets (pairs) of end-conditions is satisfied.
- Note: There is no ready-to-evaluate (no cost) formula for cubic spline interpolants, such as (C) for linear spline interpolants.
- It can be shown that the computation to construct a cubic spline interpolant (i.e. to compute the coefficients $a_{ij}, j = 0, 1, 2, 3, i = 1, \cdots, n,$ of $C(x)$) given data $(x_i, y_i), i = 0, \cdots, n,$ is equivalent to solving an $(n+1) \times (n+1)$ tridiagonal symmetric linear system, i.e. $O(n)$. This linear system is usually well-conditioned.
- The evaluation of a cubic spline requires $O(1)$ time for one evluation point.

spine interpolation error bounds

- Linear spline interpolant error bound: Let $L(x)$ be the linear spline w.r.t $x_i, i = 0, \cdots n,$ interpolating f at $x_i, i = 0, \cdots, n.$ If $f(x) \in \mathbb{C}^2$, then for $x \in [x_0, x_n]$,

- $$|f(x) - L(x)| \leq \frac{1}{8} \max_{x_0 \leq x \leq x_n} |f^{(2)}(x)| \max_{i=1}^{n} (x_i - x_{i-1})^2$$

.

- The linear spline error decreases by a factor of approximately 4, when n doubles. Linear spline interpolation is of order 2.
- Cubic spline interpolant error bound: Let $C(x)$ be the (clamped) cubic spline w.r.t. $x_i, i = 0, \cdots, n$, interpolating f at $x_i, i = 0, \cdots, n$. If $f(x) \in \mathbb{C}^4$, then for $x \in [x_0, x_n]$,
  - $$|f(x) - C(x)| \leq \frac{5}{384} \max_{x_0 \leq x \leq x_n} |f^{(4)}| \max_{i=1}^{n} (x_i - x_{i-1})^4.$$
- The cubic spline error decreases by a factor of approximately 16, when n doubles. Cubic spline interpolation is of order 4.
- Piecewise polynomial interpolants guarantee that the error decreases as the number of data increases.
- Piecewise polynomial interpolants do not (normally) suffer from oscillatory behaviour.
- Notice the different use of term "order" (and "rate") for convergence of spline interpolants and for convergence of iterative methods for nonlinear equations.