CS 357

Final Exam Review I

David Semeraro

University of Illinois at Urbana-Champaign

May 1, 2014

Definitions

- Significant digits are the numbe of digits beginning with the leftmost nonzero digit and ending with the rightmost corret digit, including final zeros that are exact.
- Absolute Error

$$|exact\ value - approximate\ \ value|$$

- Relative Error

$$\frac{|exact\ value - approximate\ value|}{|exact\ value|}$$

- **Accurate to *n* Decimal Places** means you can trust *n* digits to the right of the decimal
- **Accurate to *n* significant digits** means you can trust a total of *n* digits beginning with the leftmost nonzero digit.

definitions

- *Roundoff* occurs when digits in a decimal point $(0.3333\ldots)$ are lost (0.333) due to a limit on the memory available for storing one numerical value
- *Truncation* error occurs when discrete values are used to approximate a mathematical expression.

Taylor's Theorem

If the function f posesses continuous derivatives of orders $0, 1, 2, \cdots, (n+1)$ in a closed interval $I = [a, b]$, then for any c and x in I,

$$f(x) = \sum_{k=0}^{n} \frac{f^k(c)}{k!}(x-c)^k + E_{n+1}$$

Where the error term can be given in the form

$$E_{n+1} = \frac{f^{(n+1)}(\epsilon)}{(n+1)!}(x-c)^{n+1}$$

Alternating Series

If $a_1 \geqslant a_2 \geqslant a_3 \geqslant \cdots \geqslant a_n \geqslant \cdots \geqslant 0$ for all n and $\lim_{n\to\infty} a_n = 0$, then the alternating series:

$$a_1 - a_2 + a_3 - a_4 + \cdots$$

converges; that is,

$$\sum_{k=1}^{\infty} (-1)^{k-1} a_k = \lim_{n\to\infty} \sum_{k=1}^{n} (-1)^{k-1} a_k = \lim_{n\to\infty} S_n = S$$

Where S is the sum and S_n is the n^{th} partial sum. Moreover, for all n.

$$|S - S_n| \leqslant a_{n+1}$$

Computer representation

- A binary number x can be written:

$$x = \pm 0.b_1 b_2 b_3 \cdots \times 2^n$$

- Or

$$x = \pm q \times 2^m \quad (\frac{1}{2} \leqslant q < 1)$$

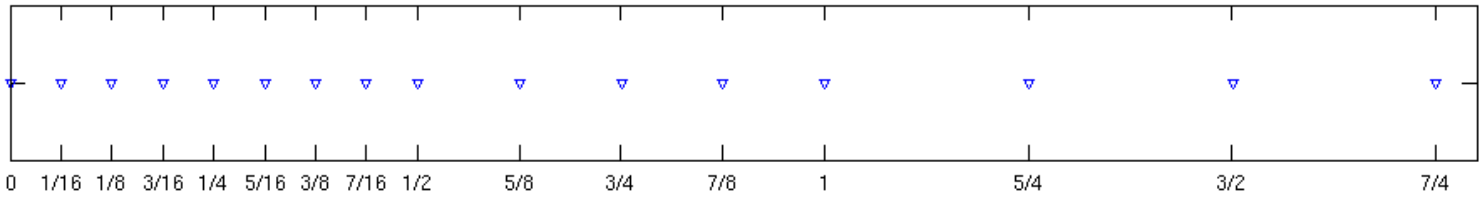
- q - normalized mantissa in $[\frac{1}{2}, 1)$
- m - exponent
- $b_1 \neq 0$

Computer representation

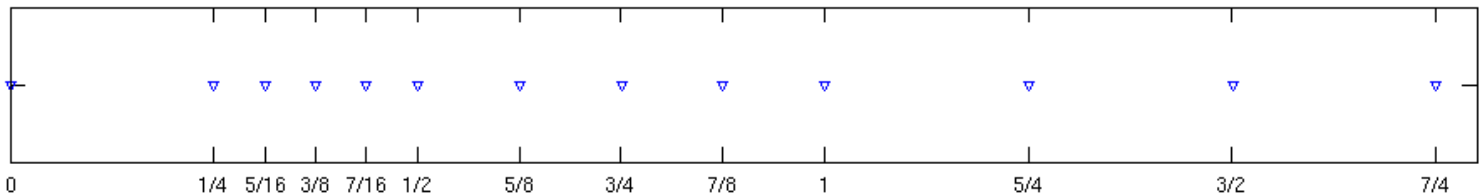
- Finite Word Length (number of bits per word)
  - Finite number of digits per number
  - Irrational numbers can not be represented
  - Numbers may be too big or too small
- 1 word per number in single precision
- 2 or more words per number in extended precision

Computer Representation

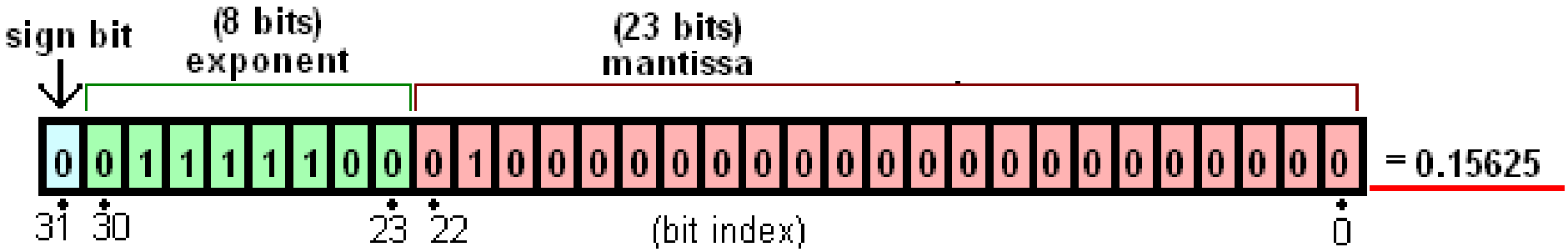
- *Machine numbers* are a discrete set.
- Consider $x = \mp(0.b_1b_2b_3) \times 2^{\pm m}$



- $x = \pm q \times 2^m; -1 \leqslant m \leqslant 1$
- m outside permissible range - *overflow or underflow*
- Numbers $< \frac{1}{16}$ underflow to zero.
- Numbers $> \frac{7}{4}$ overflow to machine infinity.
- Allowing only normalized numbers $(b_1 = 1)$ creates a *hole* at zero. $1/8, 1/16$, and $3/16$ are lost.



Single Precision



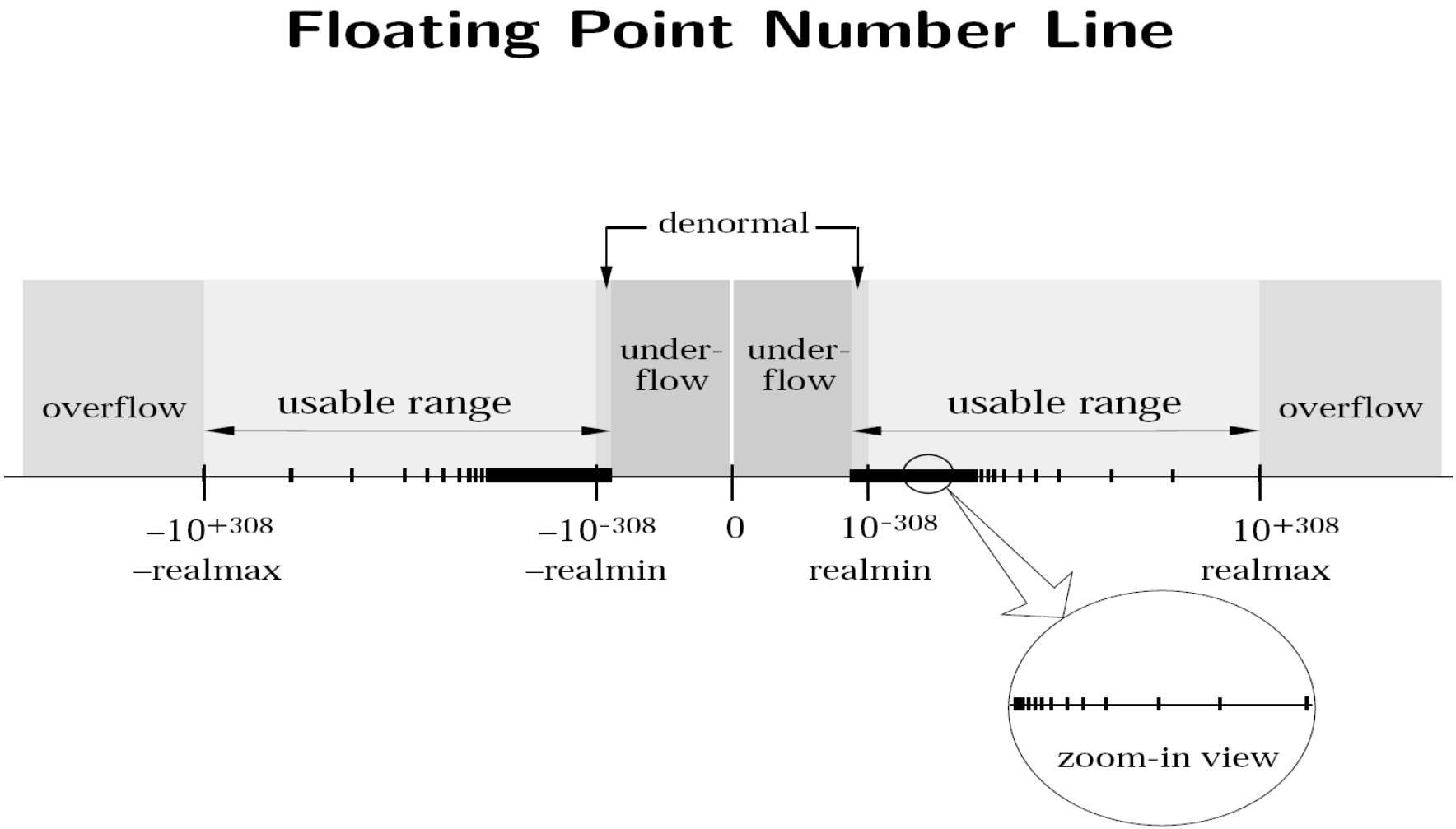
St

single precision floating point

$$(-1)^s \times 2^{c-127} \times (1.f)_2$$

- Bit 31 contains s, sign of mantissa.
- Bits 23-30 contain c in 2^{c-127}
- Bits 0-22 contains f from $(1.f)_2$

Number Line


Floating Point Number Line
denormal
overflow
usable range
under-
flow
under-
flow
usable range
overflow
-10^{+308}
$-$realmax
-10^{-308}
$-$realmin
0
10^{-308}
realmin
10^{+308}
realmax
zoom-in view

Rounding

- Machine epsilon ϵ, is the smallest machine number for which $1 + \epsilon \neq 1$.
- In single precision, $\epsilon = 2^{-23}$.
- the relative error in representing a normalized floating point number by a machine number using round to nearest is bounded by the *unit roundoff error u*.
- In single precision $u = 2^{-24}$

Solving a system

$$Ax = b$$

Three situations:

1. A is nonsingular: There exists a unique solution $x = A^{-1}b$
2. A is singular and $b \in Range(A)$: There are infinite solutions.
3. A is singular and $b \notin Range(A)$: There no solutions.

1. $A = \begin{bmatrix} 2 & 0 \\ 0 & 4 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ 8 \end{bmatrix}$, then $x = \begin{bmatrix} 1/2 \\ 2 \end{bmatrix}$.
2. $A = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, then infinitely many solutions. $x = \begin{bmatrix} 1/2 \\ \alpha \end{bmatrix}$.
3. $A = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$ $b = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, then no solutions.

Gaussian Elimination

Gaussian elimination is a mostly general method for solving square systems.

We will work with systems in their matrix form, such as

$$\begin{aligned} x_1 + 3x_2 + 5x_3 &= 4 \\ 9x_1 + 7x_2 + 8x_3 &= 6 \\ 3x_1 + 2x_2 + 7x_3 &= 1, \end{aligned}$$

in its equivalent matrix form,

$$\begin{bmatrix} 1 & 3 & 5 \\ 9 & 7 & 8 \\ 3 & 2 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 1 \end{bmatrix}.$$

Triangular Systems

The generic lower and upper triangular matrices are

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & & 0 \\ \vdots & & \ddots & \vdots \\ l_{n1} & & \cdots & l_{nn} \end{bmatrix}$$

and

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & & u_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & u_{nn} \end{bmatrix}$$

The triangular systems

$$Ly = b \qquad Ux = c$$

are easily solved by **forward substitution** and **backward substitution**, respectively

Pivoting Strategies

Partial Pivoting: Exchange only rows

- Exchanging rows does not affect the order of the x_i
- For increased numerical stability, make sure the largest possible pivot element is used. This requires searching in the partial column below the pivot element.
- Partial pivoting is usually sufficient.

More Pivoting Strategies

Full (or Complete) Pivoting: Exchange *both* rows and columns

- Column exchange requires changing the order of the x_i
- For increased numerical stability, make sure the largest possible pivot element is used. This requires searching in the pivot row, *and* in all rows below the pivot row, starting the pivot column.
- Full pivoting is less susceptible to roundoff, but the increase in stability comes at a cost of more complex programming (not a problem if you use a library routine) and an increase in work associated with searching and data movement.

Scaled Partial Pivoting

We simulate full pivoting by using a scale with partial pivoting.

- pick pivot element as the largest **relative** entry in the column (relative to the other entries in the row)
- do not swap, just keep track of the order of the pivot rows
- call this vector $\ell = [\ell_1, \ldots, \ell_n]$.

SPP Process

1. Determine a scale vector **s**. For each row

$$s_i = \max_{1 \leqslant j \leqslant n} |a_{ij}|$$

2. initialize $\ell = [\ell_1, \ldots, \ell_n] = [1, \ldots, n]$.
3. select row j to be the row with the largest ratio

$$\frac{|a_{\ell_i 1}|}{s_{\ell_i}} \qquad 1 \leqslant i \leqslant n$$

4. swap ℓ_j with ℓ_1 in ℓ
5. Now we need $n-1$ multipliers for the first column:

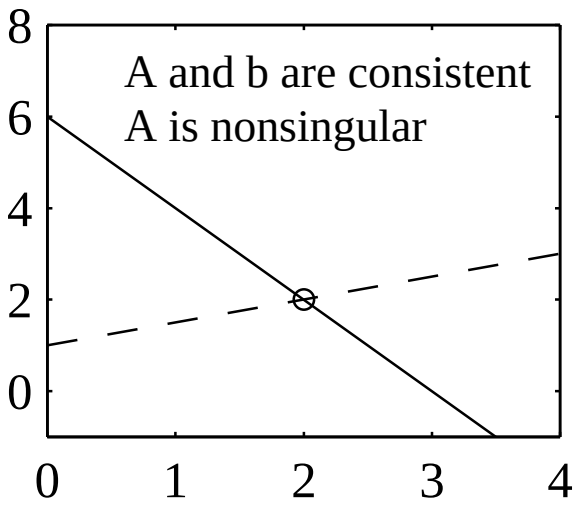
$$m_1 = \frac{a_{\ell_i 1}}{a_{\ell_1 1}}$$

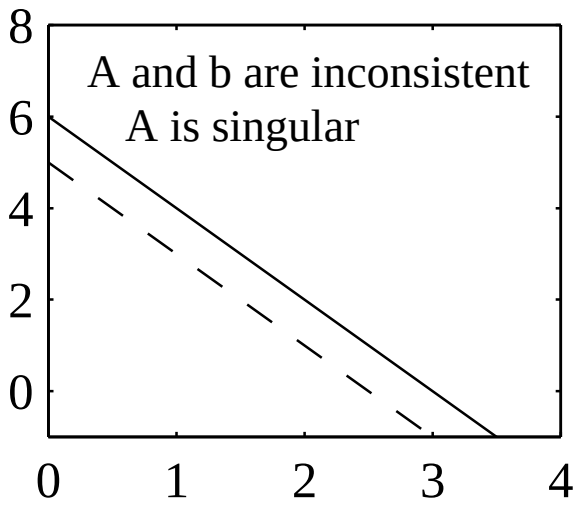
6. So the index to the rows are being swapped, NOT the actual row vectors which would be expensive
7. finally use the multiplier m_1 times row ℓ_1 to subtract from rows ℓ_i for $2 \leqslant i \leqslant n$

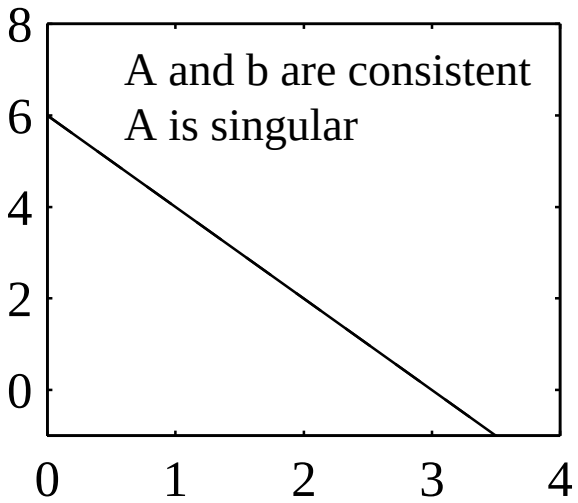
SPP Process continued

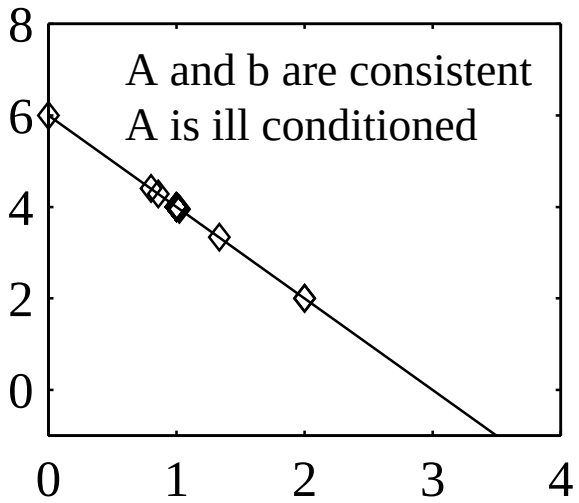
1. For the second column in forward elimination, we select row j that yields the largest ratio of
$$\frac{|a_{\ell_i,2}|}{s_{\ell_i}} \qquad 2 \leqslant i \leqslant n$$
2. swap ℓ_j with ℓ_2 in ℓ
3. Now we need $n-2$ multipliers for the second column:
$$m_2 = \frac{a_{\ell_i,2}}{a_{\ell_2 2}}$$
4. finally use the multiplier m_2 times row ℓ_2 to subtract from rows ℓ_i for $3 \leqslant i \leqslant n$
5. the process continues for row k
6. note: scale factors are *not* updated

Geometric Interpretation of Singularity









Norms

Vectors:

$$\|x\|_p = \left(|x_1|^p + |x_2|^p + \ldots + |x_n|^p\right)^{1/p}$$

$$\|x\|_1 = |x_1| + |x_2| + \ldots + |x_n| = \sum_{i=1}^{n} |x_i|$$

$$\|x\|_\infty = \max\left(|x_1|, |x_2|, \ldots, |x_n|\right) = \max_i \left(|x_i|\right)$$

Matrices:

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}$$

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}$$

$$\|A\|_1 = \max_{1 \leqslant j \leqslant n} \sum_{i=1}^{m} |a_{ij}|$$

$$\|A\|_\infty = \max_{1 \leqslant i \leqslant m} \sum_{j=1}^{n} |a_{ij}|$$

Effect of Perturbations to b

Perturb b with δb such that

$$\frac{\|\delta b\|}{\|b\|} \ll 1,$$

The perturbed system is

$$A(x + \delta x_b) = b + \delta b$$

The perturbations satisfy

$$A\delta x_b = \delta b$$

Analysis shows (see next two slides for proof) that

$$\frac{\|\delta x_b\|}{\|x\|} \leqslant \|A\|\|A^{-1}\|\frac{\|\delta b\|}{\|b\|}$$

Thus, the effect of the perturbation is small *only if* $\|A\|\|A^{-1}\|$ is small.

$$\frac{\|\delta x_b\|}{\|x\|} \ll 1 \quad \text{only if} \quad \|A\|\|A^{-1}\| \sim 1$$

Effect of Perturbations to A

Perturb A with δA such that

$$\frac{\|\delta A\|}{\|A\|} \ll 1,$$

The perturbed system is

$$(A + \delta A)(x + \delta x_A) = b$$

Analysis shows that

$$\frac{\|\delta x_A\|}{\|x + \delta x_A\|} \leqslant \|A\|\|A^{-1}\| \frac{\|\delta A\|}{\|A\|}$$

Thus, the effect of the perturbation is small *only if* $\|A\|\|A^{-1}\|$ is small.

$$\frac{\|\delta x_A\|}{\|x + \delta x_A\|} \ll 1 \quad \text{only if} \quad \|A\|\|A^{-1}\| \sim 1$$

Condition number of A

The **condition number**

$$\kappa(A) \equiv \|A\|\|A^{-1}\|$$

indicates the sensitivity of the solution to perturbations in A and b. The condition number can be measured with any p-norm.
The condition number is always in the range

$$1 \leqslant \kappa(A) \leqslant \infty$$

- $\kappa(A)$ *is a mathematical property of* A
- *Any algorithm will produce a solution that is sensitive to perturbations in* A *and* b *if* $\kappa(A)$ *is large.*
- *In exact math a matrix is either singular or non-singular.* $\kappa(A) = \infty$ *for a singular matrix*
- $\kappa(A)$ *indicates how close* A *is to being numerically singular.*
- *A matrix with large* κ *is said to be* ***ill-conditioned***

The Residual

Let $\hat{x}$ be the numerical solution to $Ax = b$. $\hat{x} \neq x$ (x is the exact solution) because of roundoff.
The **residual** measures how close $\hat{x}$ is to satisfying the original equation

$$r = b - A\hat{x}$$

It is not hard to show that

$$\frac{\|\hat{x} - x\|}{\|\hat{x}\|} \leqslant \kappa(A) \frac{\|r\|}{\|b\|}$$

Small $\|r\|$ does not guarantee a small $\|\hat{x} - x\|$.
If $\kappa(A)$ is large the $\hat{x}$ returned by Gaussian elimination and back substitution (or any other solution method) is *not* guaranteed to be anywhere near the true solution to $Ax = b$.

The Residual

Let $\hat{x}$ be the numerical solution to $Ax = b$. $\hat{x} \neq x$ (x is the exact solution) because of roundoff.
The **residual** measures how close $\hat{x}$ is to satisfying the original equation

$$r = b - A\hat{x}$$

It is not hard to show that

$$\frac{\|\hat{x} - x\|}{\|\hat{x}\|} \leqslant \kappa(A) \frac{\|r\|}{\|b\|}$$

Small $\|r\|$ does not guarantee a small $\|\hat{x} - x\|$.
If $\kappa(A)$ is large the $\hat{x}$ returned by Gaussian elimination and back substitution (or any other solution method) is *not* guaranteed to be anywhere near the true solution to $Ax = b$.

Rules of Thumb

- Applying Gaussian elimination with partial pivoting and back substitution to $Ax = b$ yields a numerical solution $\hat{x}$ such that the residual vector $r = b - A\hat{x}$ is small *even if* the $\kappa(A)$ is large.
- If A and b are stored to machine precision ε_m, the numerical solution to $Ax = b$ by any variant of Gaussian elimination is correct to d digits where

$$d = |\log_{10}(\varepsilon_m)| - \log_{10}(\kappa(A))$$

Rules of Thumb

$$d = |\log_{10}(\varepsilon_m)| - \log_{10}(\kappa(A))$$

Example:
NUMPY computations have $\varepsilon_m \approx 2.2 \times 10^{-16}$. For a system with $\kappa(A) \sim 10^{10}$ the elements of the solution vector will have

$$\begin{aligned} d &= |\log_{10}(2.2 \times 10^{-16})| - \log_{10}(10^{10}) \\ &= 16 - 10 \\ &= 6 \end{aligned}$$

correct (decimal) digits

Summary of Limits to Numerical Solution of $Ax = b$

1. $\kappa(A)$ indicates how close A is to being numerically singular
2. If $\kappa(A)$ is "large", A is **ill-conditioned** and *even the best* numerical algorithms will produce a solution, $\hat{x}$ that cannot be guaranteed to be close to the true solution, x
3. In practice, Gaussian elimination with partial pivoting and back substitution produces a solution with a small residual

$$r = b - A\hat{x}$$

even if $\kappa(A)$ is large.

Tridiagonal

A tridiagonal matrix A

$$\begin{bmatrix} d_1 & c_1 & & & & & \\ a_1 & d_2 & c_2 & & & & \\ & a_2 & d_3 & c_3 & & & \\ & & \cdots & \cdots & \cdots & & \\ & & & a_{i-1} & d_i & c_i & \\ & & & & \cdots & \cdots & \cdots \\ & & & & \cdots & \cdots & \cdots \\ & & & & & a_{n-1} & d_n \end{bmatrix}$$

- storage is saved by not saving zeros
- only $n + 2(n-1) = 3n - 2$ places are needed to store the matrix versus n^2 for the whole system
- can operations be saved? yes!

LL^T: Cholesky Factorization

- A must be symmetric and positive definite (SPD)
- A is Positive Definite (PD) if for all $x \neq 0$ the following holds

$$x^T A x > 0$$

- Positive definite gives us an all positive D in $A = LDL^T$
  - Let $x = L^{-1}e_i$, where e_i is the i-th column of I
- L becomes $LD^{1/2}$
- $A = LL^T$, i.e. $L = U^T$
  - Half as many flops as LU!
  - Only calculate L not U

Why SPD?

A matrix is Positive Definite (PD) if for all $x \neq 0$ the following holds

$$x^T Ax > 0$$

- For SPD matrices, use the Cholesky factorization, $A = LL^T$
- Cholesky Factorization
  - Requires no pivoting
  - Requires one half as many flops as LU factorization, that is only calculate L not L and U.
  - Cholesky will be more than *twice* as fast as LU because no pivoting means no data movement
- Use SCIPY's built-in `scipy.linalg.cholesky()` function for routine work

CSR

$$A = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 \\ 3 & 4 & 0 & 5 & 0 \\ 6 & 0 & 7 & 8 & 9 \\ 0 & 0 & 10 & 11 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{bmatrix}$$

$$\begin{array}{lllllllllllll} AA & = & [& 1.0 & 2.0 & 3.0 & 4.0 & 5.0 & 6.0 & 7.0 & 8.0 & 9.0 & 10.0 & 11.0 & 12.0 &] \\ JA & = & [& 1 & 4 & 1 & 2 & 4 & 1 & 3 & 4 & 5 & 3 & 4 & 5 &] \\ IA & = & [& 1 & 3 & 6 & 10 & 12 & 13 &] & & & & & & \end{array}$$

- Length of AA and JA is nnz; length of IA is $n+1$
- $IA(j)$ gives the index (offset) to the beginning of row j in AA and JA (one origin due to Fortran)
- no structure, fast row access, slow column access
- related: CSC, MSR

Rootfinding

Goals:

- Find roots to equations
- Compare usability of different methods
- Compare convergence properties of different methods

1. bracketing methods
2. Bisection Method
3. Newton's Method
4. Secant Method

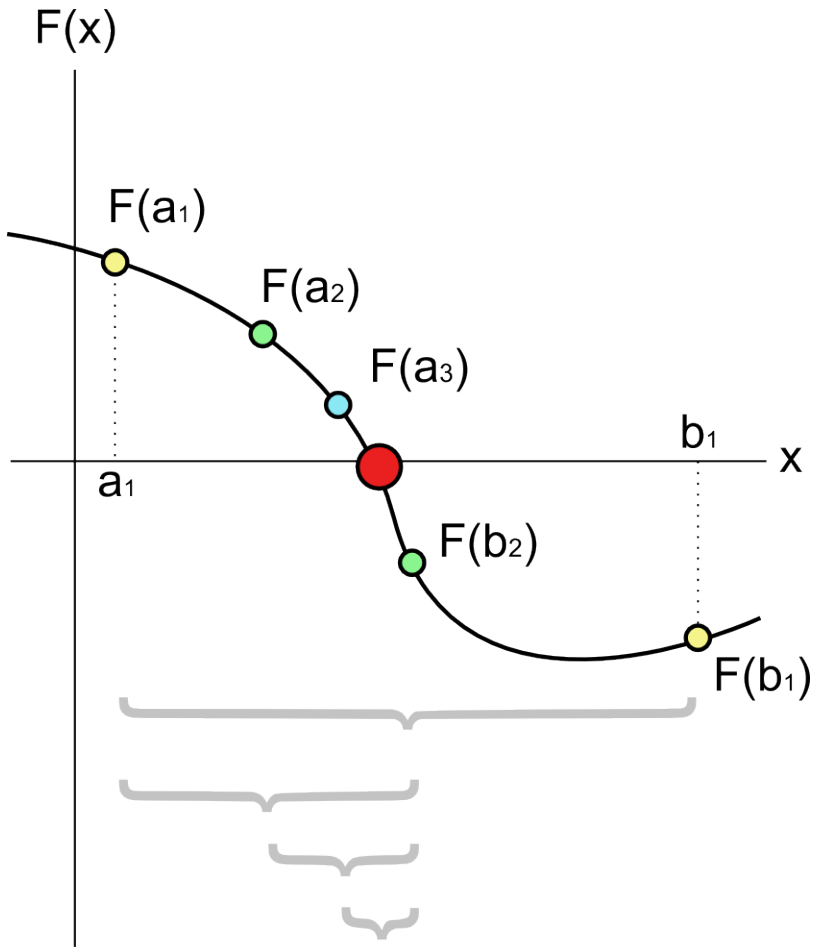
Bisection

For the bracket interval $[a, b]$ the midpoint is

$$x_m = \frac{1}{2}(a + b)$$

idea:

1. split bracket in half
2. select the bracket that has the root
3. goto step 1



Analysis of Bisection

Let δ_n be the size of the bracketing interval at the n^{th} stage of bisection. Then

$$\delta_0 = b - a = \text{initial bracketing interval}$$

$$\delta_1 = \frac{1}{2}\delta_0$$

$$\delta_2 = \frac{1}{2}\delta_1 = \frac{1}{4}\delta_0$$

$$\vdots$$

$$\delta_n = \left(\frac{1}{2}\right)^n \delta_0$$

$$\Longrightarrow \qquad \frac{\delta_n}{\delta_0} = \left(\frac{1}{2}\right)^n = 2^{-n}$$

or $$n = \log_2\left(\frac{\delta_n}{\delta_0}\right)$$

Convergence Criteria

An automatic root-finding procedure needs to monitor progress toward the root and stop when current guess is close enough to the desired root.

- Convergence checking will avoid searching to unnecessary accuracy.
- Check how closeness of successive approximations

$$|x_k - x_{k-1}| < \delta_x$$

- Check how close $f(x)$ is to zero at the current guess.

$$|f(x_k)| < \delta_f$$

- Which one you use depends on the problem being solved

Convergence rate of a root finding iteration

- Let $e_n = x^* - x_n$ be the error.
- In general, a sequence is said to converge with rate r if

$$\lim_{k \to \infty} \frac{|e_{n+1}|}{|e_n|^r} = C$$

Special Cases:

- If $r = 1$ and $C < 1$, then the rate is *linear*
- If $r = 2$ and $C > 0$, then the rate is *quadratic*
- If $r = 3$ and $C > 0$, then the rate is *cubic*

Example

Convergence Rate

1. $10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}...$
2. $10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}...$
3. $10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}...$
4. $10^{-2}, 10^{-6}, 10^{-18}, ...$

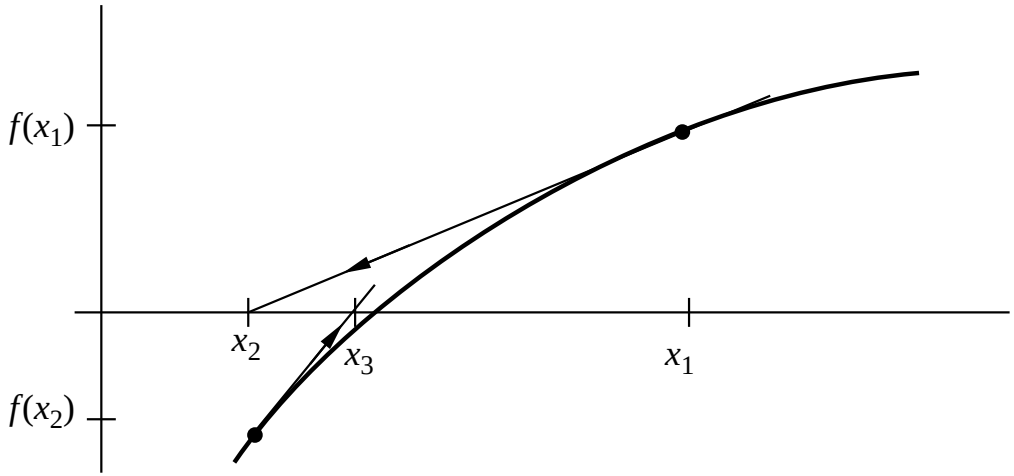
Example

Convergence Rate

1. $10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$... (linear with $C = 10^{-1}$)
2. $10^{-2}, 10^{-4}, 10^{-6}, 10^{-8}$... (linear with $C = 10^{-2}$)
3. $10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}$...(quadratic)
4. $10^{-2}, 10^{-6}, 10^{-18}$, ... (cubic)

- Linear: Adds one digit of accuracy at each step
- Quadratic: Doubles the number of digits at each step

Newton's Method



For a current guess x_k, use $f(x_k)$ and the slope $f'(x_k)$ to predict where $f(x)$ crosses the x axis.

Newton's Method

Goal is to find x such that $f(x) = 0$.
Set $f(x_{k+1}) = 0$ and solve for x_{k+1}

$$0 = f(x_k) + (x_{k+1} - x_k) f'(x_k)$$

or, solving for x_{k+1}

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}}$$

Newton's Method: Convergence

Recall

Convergence of a method is said to be of order r if there is a constant $C > 0$ such that

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^r} = C$$

Newton's method is of order 2 (quadratic) when $f'(x_*) \neq 0$. For ξ between x_k and x_*

$$f(x_*) = f(x_k) + (x_* - x_k)f'(x_k) + \frac{1}{2}(x_* - x_k)^2 f''(\xi) = 0$$

So

$$\frac{f(x_k)}{f'(x_k)} + x_* - x_k + \frac{(x_* - x_k)^2}{2}\frac{f''(\xi)}{f'(x_k)} = 0$$

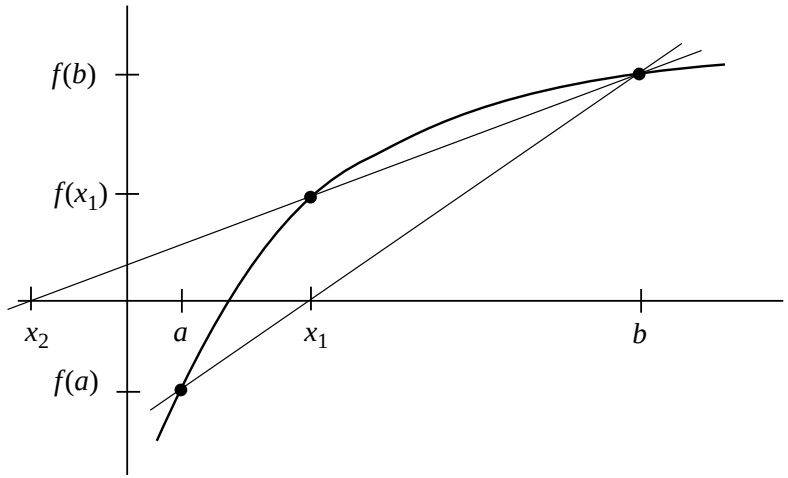
Then

$$x_* - x_{k+1} + \frac{1}{2}(x_* - x_k)^2 \frac{f''(\xi)}{f'(x_k)} = 0$$

Thus

$$\frac{|x_* - x_{k+1}|}{|x_* - x_k|^2} = \frac{1}{2}\frac{f''(\xi)}{f'(x_k)}$$

Secant Method



Given two guesses x_{k-1} and x_k, the next guess at the root is where the line through $f(x_{k-1})$ and $f(x_k)$ crosses the x axis.

Secant Method

Two versions of this formula are (equivalent in exact math)

$$x_{k+1} = x_k - f(x_k)\left[\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\right] \qquad (\star)$$

and

$$x_{k+1} = \frac{f(x_k)x_{k-1} - f(x_{k-1})x_k}{f(x_k) - f(x_{k-1})} \qquad (\star\star)$$

Equation $(\star)$ is better since it is of the form $x_{k+1} = x_k + \Delta$. Even if Δ is inaccurate the change in the estimate of the root will be small at convergence because $f(x_k)$ will also be small.
Equation $(\star\star)$ is susceptible to catastrophic cancellation:

- $f(x_k) \to f(x_{k-1})$ as convergence approaches, so cancellation error in denominator can be large.

Summary

- Plot $f(x)$ before searching for roots
- Bracketing finds coarse interval containing roots and singularities
- Bisection is robust, but converges slowly
- Newton's Method
  - Requires $f(x)$ and $f'(x)$.
  - Iterates are not confined to initial bracket.
  - Converges rapidly ($r = 2$).
  - Diverges if $f'(x) \approx 0$ is encountered.
- Secant Method
  - Uses $f(x)$ values to approximate $f'(x)$.
  - Iterates are not confined to initial bracket.
  - Converges almost as rapidly as Newton's method ($r \approx 1.62$).
  - Diverges if $f'(x) \approx 0$ is encountered.

Interpolation: Introduction

Given $n+1$ distinct points $x_0, \ldots, x_n$, and values $y_0, \ldots, y_n$, find a polynomial $p(x)$ of degree n so that

$$p(x_i) = y_i \quad i = 0, \ldots, n$$

- A polynomial of degree n has $n+1$ degrees-of-freedom:

$$p(x) = a_0 + a_1 x + \cdots + a_n x^n$$

- $n+1$ constraints determine the polynomial uniquely:

$$p(x_i) = y_i, \quad i = 0, \ldots, n$$

Theorem (page 128 6thEd)

If points $x_0, \ldots, x_n$ are distinct, then for arbitrary $y_0, \ldots, y_n$, there is a *unique* polynomial $p(x)$ of degree at most n such that $p(x_i) = y_i$ for $i = 0, \ldots, n$.

Monomials

Obvious attempt: try picking

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

So for each x_i we have

$$p(x_i) = a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_n x_i^n = y_i$$

OR

$$\begin{aligned}
a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= y_0 \\
a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= y_1 \\
a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_n x_2^n &= y_2 \\
a_0 + a_1 x_3 + a_2 x_3^2 + \cdots + a_n x_3^n &= y_3 \\
&\vdots \\
a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= y_n
\end{aligned}$$

Monomial: The problem

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ & & & \vdots & \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Question

- Is this a “good” system to solve?

Lagrange

The general Lagrange form is

$$\ell_k(x) = \prod_{i=0, i \neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

The resulting interpolating polynomial is

$$p(x) = \sum_{k=0}^{n} \ell_k(x) y_k$$

Example

Find the equation of the parabola passing through the points (1,6), (-1,0), and (2,12)

$x_0 = 1, x_1 = -1, x_2 = 2; \qquad y_0 = 6, y_1 = 0, y_2 = 12;$

$$\begin{aligned}
\ell_0(x) &= \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x+1)(x-2)}{(2)(-1)} \\
\ell_1(x) &= \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-1)(x-2)}{(-2)(-3)} \\
\ell_2(x) &= \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-1)(x+1)}{(1)(3)}
\end{aligned}$$

$$\begin{aligned}
p_2(x) &= y_0\ell_0(x) + y_1\ell_1(x) + y_2\ell_2(x) \\
&= -3 \times (x+1)(x-2) + 0 \times \frac{1}{6}(x-1)(x-2) \\
&\quad +4 \times (x-1)(x+1) \\
&= (x+1)[4(x-1) - 3(x-2)] \\
&= (x+1)(x+2)
\end{aligned}$$

Newton Polynomials

- Newton Polynomials are of the form

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) + \ldots$$

- The basis used is thus

function	order
1	0
$x - x_0$	1
$(x - x_0)(x - x_1)$	2
$(x - x_0)(x - x_1)(x - x_2)$	3

- More stable that monomials
- More computationally efficient (nested iteration) than using Lagrange and shifted monomials

Newton Polynomials using Divided Differences

Consider the data

x_0	x_1	x_2
y_0	y_1	y_2

We want to find a_0, a_1, and a_2 in the following polynomial so that it fits the data:

$$p_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

Matching the data gives three equations to determine our three unknowns a_i:

$$\begin{aligned}
&\text{at } x_0\text{: } y_0 = a_0 + 0 + 0 \\
&\text{at } x_1\text{: } y_1 = a_0 + a_1(x_1 - x_0) + 0 \\
&\text{at } x_2\text{: } y_2 = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1)
\end{aligned}$$

Divided Differences

the easy way: example

x	$f[\cdot]$	$f[\cdot,\cdot]$	$f[\cdot,\cdot,\cdot]$	$f[\cdot,\cdot,\cdot,\cdot]$
1	3			
		$\frac{1}{2}$		
$\frac{3}{2}$	$\frac{13}{4}$		$\frac{1}{3}$	
		$\frac{1}{6}$		-2
0	3		$-\frac{5}{3}$	
		$-\frac{2}{3}$		
2	$\frac{5}{3}$			

The coefficients are readily available and we arrive at

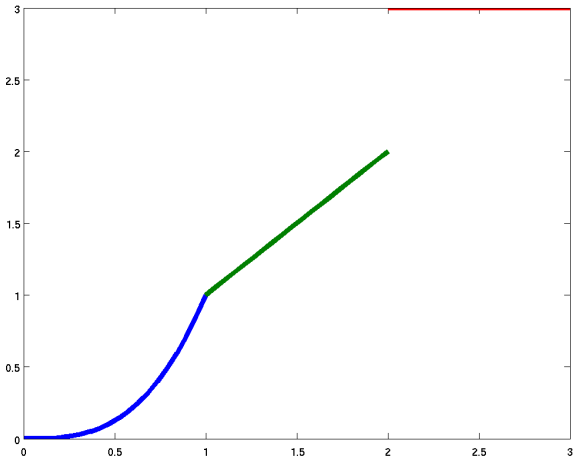
$$p_2(x) = 3 + \frac{1}{2}(x-1) + \frac{1}{3}(x-1)(x-\frac{3}{2}) - 2(x-1)(x-\frac{3}{2})x$$

Piecewise Polynomial

A function $f(x)$ is considered a piecewise polynomial on $[a, b]$ if there exists a (finite) partition P of $[a, b]$ such that $f(x)$ is a polynomial on each $[t_i, t_{i+1}] \in P$.

Example

$$f(x) = \begin{cases} x^3 & x \in [0, 1] \\ x & x \in (1, 2) \\ 3 & x \in [2, 3] \end{cases}$$



degree 1 spline

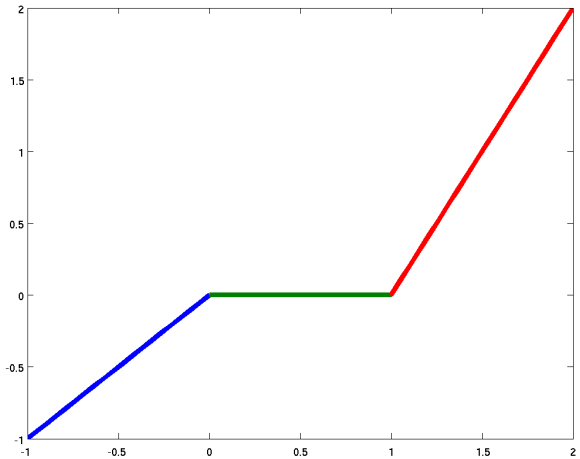
definition

A function $S(x)$ is a spline of degree 1 if:

1. The domain of $S(x)$ is an interval $[a, b]$
2. $S(x)$ is continuous on $[a, b]$
3. There is a partition $a = t_0 < t_1 < \cdots < t_n = b$ such that $S(x)$ is linear on each subinterval $[t_i, t_{i+1}]$.

Example

$$S(x) = \begin{cases} x & x \in [-1, 0] \\ 1 & x \in (0, 1) \\ 2x - 2 & x \in [1, 2] \end{cases}$$



degree 2 splines

definition

A function $S(x)$ is a spline of degree 2 if:

1. The domain of $S(x)$ is an interval $[a, b]$
2. $S(x)$ is continuous on $[a, b]$
3. $S'(x)$ is continuous on $[a, b]$
4. There is a partition $a = t_0 < t_1 < \cdots < t_n = b$ such that $S(x)$ is quadratic on each subinterval $[t_i, t_{i+1}]$.

degree 3 spline: cubic spline

definition

A function $S(x)$ is a spline of degree 3 if:

1. The domain of $S(x)$ is an interval $[a, b]$
2. $S(x)$ is continuous on $[a, b]$
3. $S'(x)$ is continuous on $[a, b]$
4. $S''(x)$ is continuous on $[a, b]$
5. There is a partition $a = t_0 < t_1 < \cdots < t_n = b$ such that $S(x)$ is cubic on each subinterval $[t_i, t_{i+1}]$.

degree 3 spline: cubic spline

In each intervale $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ knots, 4 unknowns per interval
- $4n$ unknowns

degree 3 spline: cubic spline

In each intervale $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ knots, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity

degree 3 spline: cubic spline

In each intervale $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ knots, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity
- $n-1$ constraints by continuity of $S'(x)$

degree 3 spline: cubic spline

In each intervale $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ knots, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity
- $n-1$ constraints by continuity of $S'(x)$
- $n-1$ constraints by continuity of $S''(x)$

degree 3 spline: cubic spline

In each intervale $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ knots, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity
- $n-1$ constraints by continuity of $S'(x)$
- $n-1$ constraints by continuity of $S''(x)$
- $4n-2$ total constraints

degree 3 spline: cubic spline

In each intervale $[t_i, t_{i+1}]$, $S(x)$ looks like

$$S_i(x) = a_{0,i} + a_{1,i}x + a_{2,i}x^2 + a_{3,i}x^3$$

- n intervals, $n+1$ knots, 4 unknowns per interval
- $4n$ unknowns
- $2n$ constraints by continuity
- $n-1$ constraints by continuity of $S'(x)$
- $n-1$ constraints by continuity of $S''(x)$
- $4n-2$ total constraints
- This leaves 2 extra degrees of freedom. The cubic spline is not yet unique!

degree 3 spline: cubic spline

Some options:

- natural cubic spline: $S''(t_0) = S''(t_n) = 0$
- fixed-slope: $S'(t_0) = a$, $S'(t_n) = b$
- not-a-knot: $S'''(x)$ continuous at t_1 and t_{n-1}
- periodic: S' and S'' are periodic at the ends