Document: Math 226A (Fall 2011)
Professor: Freund
Latest Update: April 2, 2012
Author: Jeff Irion
http://www.math.ucdavis.edu/~jlirion

Contents

1 9-23-11

1.1 Announcements

- `http://www.math.ucdavis.edu/~freund/226A`
- Office Hours are Monday 12:30-2:30 at MSB 2140
- 5 homeworks
- Grading scheme
  - HW 50% (10% each)
  - Final 50% (open book, open notes)
    - 226B and 226C will have final projects
- No textbook, several reference books are listed on the course webpage

1.2 General Remarks

Definition 1.1. ***Numerical Analysis***

Numerical analysis is the study of algorithms for the problems of continuous mathematics (Trefethen).

Problem:

$$\text{Input} \;\to\; \text{Output: Solution(s) or No Solution}$$

Numerical Methods:

$$\text{(Approximate) Input} \;\to\; \text{(Approximate) Solution}$$

Sources of Errors:

1. Approximate input errors (cannot represent the data exactly, e.g. irrational numbers) $\Rightarrow$ *conditioning* of the problem. These problems are inherent to the problem.
2. Rounding errors $\Rightarrow$ *stability* of the algorithm
3. Approximation errors (i.e. terminating your algorithm when the result is "close enough") $\Rightarrow$ *convergence*

Example 1.2.

1. Systems of linear equations

$$Ax = b, \qquad A \in \mathbb{C}^{n\times n},\ b \in \mathbb{C}^n,\ A \text{ nonsingular}$$

   Input: $A, b \to$ Solution: $x = A^{-1}b$ (obtained via LU factorization = modern Gaussian elimination)

2. Systems of nonlinear equations

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \qquad \text{where } \mathbf{f}(\mathbf{x}) = \mathbf{Ax} - \mathbf{b}$$

   Input: $\mathbf{f} : D \to \mathbb{R}^n,\ D \subset \mathbb{R}^n$.
   Solutions: $\mathbf{x} \in D$ such that $\mathbf{f}(\mathbf{x}) = \mathbf{0}$.
   Standard method for solving: Newton's method

1.3 Introduction to LU Factorization

Remark 1.3. ***LU Factorization***

Given: (nonsingular) $A \in \mathbb{C}^{n\times n},\ b \in \mathbb{C}^n$.
Goal: solve $Ax = b$.
Gaussian elimination:

$$Ax = b \leftrightarrow Ux = c, \qquad \text{where } U = \begin{bmatrix} * & * & \dots & * \\ 0 & * & \dots & * \\ \vdots & \ddots & \ddots & * \\ 0 & \dots & 0 & * \end{bmatrix}$$

i.e. U is *upper-triangular*. How do we do this?

$$L_{n-1}\dots L_2L_1A = U \quad \Leftrightarrow \quad A = \underbrace{L_1^{-1}\dots L_{n-2}^{-1}L_{n-1}^{-1}}_{L}U$$

$$L = \begin{bmatrix} 1 & 0 & \dots & 0 \\ * & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & 0 \\ * & \dots & * & 1 \end{bmatrix}$$

i.e. L is *lower-triangular*.

Example 1.4. ***General LU Factorization Example***

$n = 4$

$$A = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \overset{L_1}{\rightarrow} \begin{bmatrix} * & * & * & * \\ 0 & \# & \# & \# \\ 0 & \# & \# & \# \\ 0 & \# & \# & \# \end{bmatrix} = L_1 A$$

$$\overset{L_2}{\rightarrow} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & \# & \# \\ 0 & 0 & \# & \# \end{bmatrix} = L_2 L_1 A$$

$$\overset{L_3}{\rightarrow} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & \# \end{bmatrix} = L_3 L_2 L_1 A = U$$

$*$s represent unchanged nonzero entries, #s represent changed nonzero entries.

The L_i are Frobenius matrices.

Definition 1.5. ***Frobenius matrix***

`http://en.wikipedia.org/wiki/Frobenius_matrix`

A *Frobenius matrix* is a square matrix with the following properties:

- all entries on the main diagonal are ones
- the entries below the main diagonal of at most one column are arbitrary
- every other entry is zero

Frobenius matrices are invertible. The following is an example:

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & a_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2} & 0 & \cdots & 1 \end{bmatrix}, \qquad A^{-1} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & -a_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & -a_{n2} & 0 & \cdots & 1 \end{bmatrix}$$

2 9-26-11

2.1 LU Factorization (Continued)

Example 2.1.

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

$$L_1 A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -4 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{bmatrix} A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 3 & 5 & 5 \\ 0 & 4 & 6 & 8 \end{bmatrix}$$

$$L_2 L_1 A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & -4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 3 & 5 & 5 \\ 0 & 4 & 6 & 8 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 2 & 4 \end{bmatrix}$$

$$L_3 L_2 L_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} L_2 L_1 A = \begin{bmatrix} 2 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 2 \end{bmatrix} = U \quad \text{upper-triangular}$$

$$A = LU$$

$$L = L_1^{-1} L_2^{-1} L_3^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 4 & 0 & 1 & 0 \\ 3 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 3 & 1 & 0 \\ 0 & 4 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 4 & 3 & 1 & 0 \\ 3 & 4 & 1 & 1 \end{bmatrix} \qquad \text{unit lower-triangular}$$

Remark 2.2. ***Convention (Matlab Notation)***

$$\mathbf{A} = [a_{jk}]_{j,k=1,2,\ldots,n} \in \mathbb{C}^{n \times n}$$

$$a_{j,k:n} = [a_{j,k} \; a_{j,k+1} \; \cdots \; a_{j,n}]$$

Remark 2.3. ***Algorithm: LU Factorization without pivoting***

Input: $\mathbf{A} \in \mathbb{C}^{n \times n}$
Output: L, U such that $A = LU$

Set $U = A,\ L = I$ ($n \times n$ identity matrix).

- for $k = 1, 2, \ldots, n-1$
  - for $j = k+1, k+2, \ldots, n$
    * $l_{j,k} = u_{j,k}/u_{k,k}$ (potential problem)
    * $u_{j,k:n} = u_{j,k:n} - l_{j,k}u_{k,k:n}$
  - end
- end

Output: $L, U \in \mathbb{C}^{n \times n}$ such that $A = LU$

Remark 2.4. ***Operation Count***

Additions: $\sum\limits_{k=1}^{n-1}(n-k)(n-k+1) = \sum\limits_{l=1}^{n-1} l(l+1) \approx \frac{n^3}{3}$ (where $l = n-k$)
Multiplications: $\approx \frac{n^3}{3}$
Divisions: $\approx n^2 \Rightarrow$ ignore (because it is a lower order term)
Total Work: $\approx \frac{2n^3}{3}$ flops

Definition 2.5. ***Flop***

A *flop* is a floating-point operation: addition, subtraction, multiplication, division, square root.

Remark 2.6. ***Solution of*** $Ax = b$

$$\underbrace{A}_{=LU} x = b$$
$$L\,\underbrace{Ux}_{=y} = b$$
$$Ly = b \tag{2.1}$$
$$Ux = y \tag{2.2}$$

$Ly = b$ is easily solved via forward substitution. $Ux = y$ is easily solved by backward substitution.

One triangular solve requires $\approx n^2$ flops.

2.2 Problems with LU Factorization without Pivoting

LU factorization without pivoting is unstable!

Example 2.7. ***Error from LU Factorization without Pivoting***

$$A = \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

With exact arithmetic:

$$L = \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 10^{-20} & 1 \\ 0 & 1 - 10^{20} \end{bmatrix}$$

Now solve a system:

$$\begin{aligned} Ly &= b \\ y &= \begin{bmatrix} 1 \\ -10^{20} \end{bmatrix} \\ Ux &= y \\ x &= \frac{1}{1 - 10^{-20}} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \\ &\approx \begin{bmatrix} -1 \\ 1 \end{bmatrix} \end{aligned}$$

In a computer:
Floating point arithmetic does everything accurately up to ≈ 16 digits.

$$\begin{aligned} \tilde{L} = L &= \begin{bmatrix} 1 & 0 \\ 10^{20} & 1 \end{bmatrix}, \quad \tilde{U} = \begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix} \\ \tilde{L}\tilde{U} &= \begin{bmatrix} 10^{-20} & 1 \\ 1 & 0 \end{bmatrix} \not\approx A \end{aligned}$$

If you use this to compute the answer, you will get a bogus answer.

3 9-28-11

3.1 Continued from 9-26-11...

Example 3.1. ***Continuing from last time...***

$$\begin{aligned}
\tilde{L}\tilde{y} &= b \\
\tilde{y} &= \begin{bmatrix} 1 \\ 10^{-20} \end{bmatrix} \\
\tilde{U}\tilde{x} &= \tilde{y} \\
\tilde{x} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \neq x
\end{aligned}$$

The culprit is 10^{20}.

3.2 Pivoting

$$\begin{aligned}
A &= \begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} \\
P &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} && \textit{(permutation matrix)} \\
PA &= \begin{bmatrix} 1 & 1 \\ 20^{-20} & 1 \end{bmatrix} \\
&= \underbrace{\begin{bmatrix} 1 & 0 \\ 10^{-20} & 1 \end{bmatrix}}_{=L} \underbrace{\begin{bmatrix} 1 & 1 \\ 0 & 1 - 10^{-20} \end{bmatrix}}_{=y}
\end{aligned}$$

Floating point arithmetic:

$$\begin{aligned}
\tilde{L} &= L \\
\tilde{U} &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \\
PAx &= Pb \\
LUx &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
\tilde{L}\tilde{y} &= Pb \\
\tilde{y} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
\tilde{U}\tilde{x} &= \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\
\tilde{x} &= \begin{bmatrix} -1 \\ 1 \end{bmatrix} \approx x = A^{-1}b
\end{aligned}$$

Every time you use Gaussian elimination you need to use pivoting.

Example 3.2. ***General Pivoting Example***

$n = 4,\ k = 2$ (4×4 matrix)

$$U = \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{bmatrix}$$

$$\xrightarrow{P_2} \begin{bmatrix} * & * & * & * \\ 0 & \# & \# & \# \\ 0 & * & * & * \\ 0 & \# & \# & \# \end{bmatrix} \qquad \text{exchange rows 2 \& 4}$$

$$\xrightarrow{L_2} \begin{bmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & \# & \# \\ 0 & 0 & \# & \# \end{bmatrix}$$

In general:

$$L_{n-1}P_{n-1}\cdots L_3L_2P_2L_1P_1A = U$$

If you don't need to pivot, then use $P_i = I$ (the identity).

Example 3.3. ***Specific Pivoting Example***

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

$$\underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{P_1} A = \begin{bmatrix} 8 & 7 & 9 & 5 \\ 4 & 3 & 3 & 1 \\ 2 & 1 & 1 & 0 \\ 6 & 7 & 9 & 8 \end{bmatrix}$$

$$L_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 & 0 \\ -\frac{1}{4} & 0 & 1 & 0 \\ -\frac{3}{4} & 0 & 0 & 1 \end{bmatrix}$$

$$L_1 P_1 A = \begin{bmatrix} 8 & 7 & 9 & 5 \\ 0 & \frac{7}{4} & \frac{9}{4} & \frac{17}{4} \\ 0 & -\frac{3}{4} & -\frac{5}{4} & -\frac{5}{4} \\ 0 & -\frac{1}{2} & -\frac{3}{2} & -\frac{3}{2} \end{bmatrix}$$

$$\vdots$$

$$U = \begin{bmatrix} 8 & 7 & 9 & 5 \\ 0 & \frac{7}{4} & \frac{9}{4} & \frac{17}{4} \\ 0 & 0 & -\frac{6}{7} & -\frac{2}{7} \\ 0 & 0 & 0 & \frac{2}{3} \end{bmatrix}$$

Example 3.4. ***Pivoting in Matrix Form***

$n = 4:$

$$L_3 P_3 \underbrace{I}_{=P_3^T P_3} L_2 P_2 L_1 \underbrace{I}_{=P_2^T P_3^T P_3 P_2} P_1 A = U$$

$$\underbrace{L_3}_{=L_3'} \underbrace{(P_3 L_2 P_3^T)}_{=L_2'} \underbrace{(P_3 P_2 L_1 P_2^T P_3^T)}_{=L_1'} \underbrace{(P_3 P_2 P_1)}_{=P} A = U$$

$$L_3' L_2' L_1' (PA) = U$$

L_1' ($\neq L_1^T$) has the same structure as L_1, with nontrivial elements in column 1 reordered.

$$L_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 & 0 \\ -\frac{1}{4} & 0 & 1 & 0 \\ -\frac{3}{4} & 0 & 0 & 1 \end{bmatrix}, \qquad L_1' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & 0 & 1 & 0 \\ * & 0 & 0 & 1 \end{bmatrix}$$

$$PA = \underbrace{(L_1')^{-1} (L_2')^{-1} (L_3')^{-1}}_{=L} U$$

General case:

$$PA = LU$$

where P is the permutation matrix and L and U are lower and upper triangular matrices, respectively.

Remark 3.5. ***Algorithm: LU factorization with partial pivoting***

Input: $A \in \mathbb{C}^{n \times n}$
Set $U = A,\ L = I,\ P = I.$

- for $k = 1, 2, \ldots, n-1$
  - choose $i \geq k$ such that $|u_{ik}| = \max\limits_{k \leq l \leq n} |u_{lk}|$
  - if $u_{lk} = 0$, stop: A is singular
  - $u_{k,k:n} \leftrightarrow u_{i,k:n}$ (interchange rows i and k)
  - $l_{k,1:k-1} \leftrightarrow l_{i,1:k-1}$
  - $p_{k,:} \leftrightarrow p_{i,:}$
  - for $j = k+1, k+2, \ldots, n$
    - $l_{jk} = u_{jk}/u_{kk}$
    - $u_{j,k:n} = u_{j,k:n} - l_{jk} u_{k,k:n}$
  - end (j)
- end (k)

Output: U, L, P such that

$$PA = LU$$

where $L = [l_{jk}]$ with $|l_{jk}| \leq 1$ for all $j > k$.

4 9-30-11

4.1 LU Factorization Recap

$$PA = LU$$
$$L = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{bmatrix}, \quad |l_{jk}| \leq 1$$

LU factorization with partial pivoting is stable in practice.

4.2 Newton's Method

Newton's method is not a finite method, meaning that you would have to run it infinitely many times in order for it to arrive at the answer.

Remark 4.1. ***Newton's Method Overview***

Given: $f : D \to \mathbb{R}^n,\ D \subset \mathbb{R}^n$

$$f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}, \quad f_j : D \to \mathbb{R}, \quad j = 1, 2, \ldots, n$$
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in D \subset \mathbb{R}^n, \quad x_j \in \mathbb{R}, \quad j = 1, 2, \ldots, n$$

Goal: Find $x \in D$ such that
$$f(\mathbf{x}) = \mathbf{0}.$$

Thus, we have n nonlinear equations for n unknowns.

Example 4.2. ***Newton's Method Example 1: Possible Scenarios***

$D = \mathbb{R}^n,\ f : \mathbb{R}^n \to \mathbb{R}^n,$

$$f(\mathbf{x}) = \mathbf{b} - \mathbf{A}\mathbf{x}, \qquad \mathbf{b} \in \mathbb{R}^n,\ \mathbf{A} \in \mathbb{R}^{n \times n}$$
$$f(\mathbf{x}) = 0 \ \Leftrightarrow\ \mathbf{A}\mathbf{x} = \mathbf{b}$$

There are 3 possibilities:

1. 1 solution
2. no solution
3. infinitely many solutions

Example 4.3. ***Newton's Method Example 2***

$f = f_\alpha : \mathbb{R} \to \mathbb{R},\ f_\alpha(x) = x^2 + \alpha,\ \alpha \in \mathbb{R}$ is a parameter. This is an upward facing parabola with minimum value α.

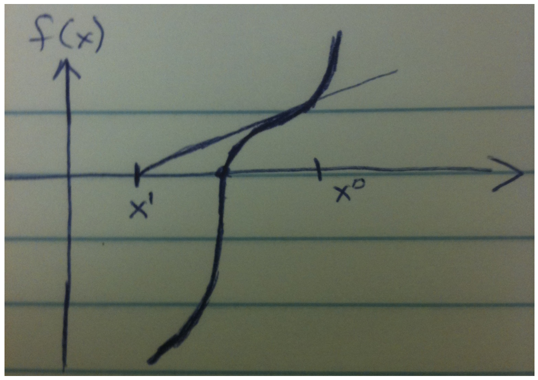
1. $f(x)$ has no solution if $\alpha > 0$
2. $f(x)$ has one solution if $\alpha = 0$ $(x = 0)$
3. $f(x)$ has two solutions if $\alpha < 0$ $(x = \pm\sqrt{-\alpha})$

Example 4.4. ***Newton's Method Example 3***

$n = 1,\ D = \mathbb{R}^+ = \{x \in \mathbb{R} \bigm| x > 0\},\ f(x) = \sin\frac{1}{x}$. Thus, f oscillates faster as $x \to 0$.

$$\begin{aligned} f(x) = 0 \quad &\Leftrightarrow \quad \frac{1}{x} = j\pi,\ j = 1, 2, \ldots \\ x = \frac{1}{j\pi} \end{aligned}$$

Remark 4.5. ***Newton's Method Algorithm for*** $n = 1$



x^0 = initial guess
$f(x^*) = 0$

1. Linearize $f(x)$ around x^0:

$$f(x) = f(x^0) + f'(x^0)(x - x^0) + O((x - x^0)^2) \equiv t(x)$$

2. Get new approximation x^1 by setting $t(x^1) = 0$ and solving for x^1:

$$x^1 = x^0 - \frac{f(x^0)}{f'(x^0)}$$

(provided $f'(x) \neq 0$).

3. Repeat with x^0 replaced by x^1. Iterative process:

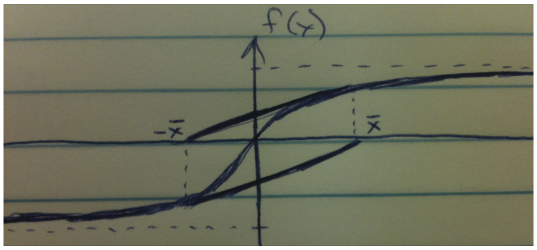
$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}, \quad k = 0, 1, 2, \ldots$$

Remark 4.6. ***Questions about Newton's Method***

1. Can we guarantee that $\lim_{k \to \infty} x^k = x^*$?
2. If yes, what is the speed of convergence?

Example 4.7. ***Failure of Newton's Method***

$f : \mathbb{R} \to \mathbb{R}$, $f(x) = \tan^{-1} x$, $f'(x) = \frac{1}{1+x^2}$.



f has a single zero: $x^* = 0$. We apply Newton's method. Choose $x^0 \in \mathbb{R}$.

$$x^{k+1} = x^k - \left(1 + (x^k)^2\right) \tan^{-1}(x^k), \quad k = 0, 1, 2, \ldots$$

Convergence?
Let $\overline{x}$ be the unique solution of

$$-\overline{x} = \overline{x} - (1 + \overline{x}^2) \tan^{-1} \overline{x}, \quad \overline{x} > 0$$
$$\overline{x} = 1.39174\ldots$$

where $\overline{x}$ is computed by using Newton's Method for

$$g(x) = 2x - (1 + x^2) \tan^{-1} x.$$

Thus, if $x^0 = \overline{x}$, then $x^k = (-1)^k \overline{x}$, $k = 0, 1, \ldots$
Similarly, if $x^0 = -\overline{x}$, then $x^{k+1} = (-1)^{k+1} \overline{x}$, $k = 0, 1, \ldots$
If $|x^0| < \overline{x}$, then $\lim_{k \to \infty} x^k = 0 = x^*$
If $|x^0| > \overline{x}$, then $\lim_{k \to \infty} x^k = \infty$

5 10-3-11

5.1 Newton's Method (Continued)

Remark 5.1. ***Newton's Method for $n \geq 1$***

$$f : D \to \mathbb{R}^n, \quad f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in D$$

Initial guess: $x^0 \in D$.

$$f(x) = f(x^0) + Df(x^0)(x - x^0) + O(\|x - x^0\|^2) \tag{5.1}$$

where

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} = \text{Euclidean norm.}$$

$$Df(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} = \left[\frac{\partial f_j}{\partial x_k}\right]_{j,k=1,2,\ldots,n} = \text{Jacobian of } f$$

Obtain a new approximation x^1 by setting (5.1) to 0:

$$\mathbf{0} = f(x^0) + Df(x^0)\underbrace{(x^1 - x^0)}_{\Delta x^0}$$

$$\begin{aligned} Df(x^0)\Delta x^0 &= -f(x^0) \\ x^1 &= x^0 + \Delta x^0. \end{aligned}$$

This is a system of n linear equations for n unknowns.

Remark 5.2. ***Newton's Method Algorithm***

Choose $x^0 \in D$.

- for $k = 1, 2, \ldots$ (open-ended), do:
  - if $Df(x^k)$ is singular, stop.
  - else, solve $Df(x^k)\Delta x^k = -f(x^k)$ for Δx^k
  - set $x^{k+1} = x^k + \Delta x^k$
  - if $x^{k+1} \notin D$, stop.
  - check for convergence: if $f(x^{k+1}) \approx 0$, stop.
- end (k)

Remark 5.3.

1. Need to solve a linear system at each k step $\Rightarrow$ LU factorization of $A = Df(x^k)$
2. Newton's method is *affine-invariant*:
   Let $M \in \mathbb{R}^{n\times n}$ be a fixed nonsingular matrix, and let $g : D \to \mathbb{R}^n$, $g(x) := Mf(x)$ for all $x \in D$. Thus, $f(x) = 0 \;\Leftrightarrow\; Mf(x) = \mathbf{0} \;\Leftrightarrow\; g(x) = 0$. Then Newton's method applied to f and g results in the same iterates $x^k,\ k = 0, 1, 2, \ldots$.

Proof. (That Newton's method is affine-invariant)

$$
\begin{aligned}
g(x) &= Mf(x) \\
Dg(x^k) &= MDf(x^k) \\
Dg(x^k)\Delta x^k &= -g(x^k) \quad \Leftrightarrow \quad Df(x^k)\Delta x^k = -f(x^k)
\end{aligned}
$$

□

5.2 Convergence of Newton's Method

Proposition 5.4. ***Convergence***

Let x^* denote a zero of f, i.e. $f(x^*) = \mathbf{0}$. If $Df(x^*)$ is nonsingular and x^0 is "close" to x^*, then

$$\lim_{k\to\infty} x^k = x^*$$

and the speed of convergence is quadratic, i.e.

$$\|x^{k+1} - x^k\| \le C\|x^k - x^*\|^2$$

Definition 5.5. ***Quadratic Convergence***

The number of correct digits roughly doubles in each iteration.

Theorem 5.6. ***Convergence of Newton's Method, Affine-Invariant Version***

Assumptions:

- $D \subset \mathbb{R}$ is convex
- $f : D \to \mathbb{R}^n$ is C^1
- $x^0 \in D$ and $Df(x^0)$ is nonsingular
- Df satisfies the *affine-invariant Lipschitz condition*:

$$\|(Df(x^0))^{-1}Df(y) - Df(x)\| \leq \gamma\|y - x\|$$

Here $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ is a vector norm and $\|M\| := \max_{\|x\|=1} \|Mx\|$ is the associated matrix norm, which satisfies:

1. $\|Mx\| \leq \|M\|\|x\|$
2. $\|I\| = 1$
3. $\|MN\| \leq \|M\|\|N\|$

6 10-5-11

6.1 Newton's Method Convergence Theorem

Theorem 6.1.

Given:

- $f : D \to \mathbb{R}^n$
- $x^0 \in D$ and $Df(x^0)$ is nonsingular
- $\|Df(x^0)^{-1}(Df(y) - Df(x))\| \leq \gamma\|y - x\|$ for all $x, y \in D$ (bigger $\gamma \Rightarrow$ more nonlinear)
- $\|Df(x^0)^{-1}f(x^0)\| \leq \alpha$ for some $\alpha > 0$
- $h := \alpha\gamma < \frac{1}{2}$

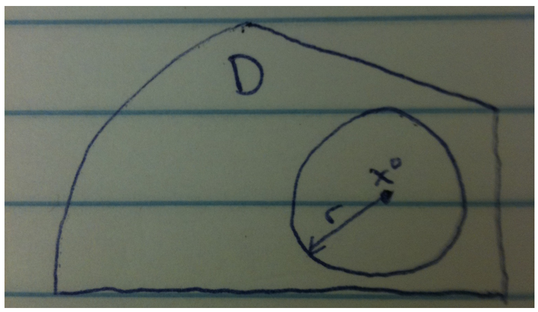


Figure 1: $S_r(x^0) := \{x \in \mathbb{R}^n \mid \|x - x^0\| < r\},\ \overline{S_r(x^0)} := \{x \in \mathbb{R}^n \mid \|x - x^0\| \leq r\}$

- $\overline{S_r(x^0)} \subset D,\ r = \frac{1-\sqrt{1-2h}}{\gamma} > 0$

Claims:

1. All Newton iterates x^k satisfy $x^k \in S_r(x^0),\ Df(x^k)$ is nonsingular
2. The iterates x^k converge quadratically to a zero of f, $x^* := \lim_{k\to\infty} x^k$, $x^* \in \overline{S_r(x^0)}$
3. x^* is the <u>only</u> zero of f in $D \cap S_R(x^0),\ R := \frac{1+\sqrt{1-2h}}{\gamma}$

Note:

$$\|x^1 - x^0\| = \| - Df(x^0)^{-1}f(x^0)\| \leq \alpha = \frac{h}{\gamma}$$

Proof. (of parts 1 and 2 of the theorem)
We define two scalar sequences (h_k) and (γ_k) as

$$h_k := \begin{cases} h & k = 0 \\ \frac{1}{2}\frac{h_{k-1}^2}{(1-h_{k-1})^2} & k > 0 \end{cases}$$

$$\gamma_k := \begin{cases} \gamma & k = 0 \\ \frac{\gamma_{k-1}}{1-h_{k-1}} & k > 0 \end{cases}$$

It is easy to verify that for all $k = 0, 1, \ldots$ we have

$$\lim_{j\to\infty} h_j = 0 < h_{k+1} < h_k < \frac{1}{2}$$

$$\gamma_k < \gamma_{k+1} < \frac{\gamma}{\sqrt{1-2h}} = \lim_{j\to\infty} \gamma_j$$

2 Properties:

$$\frac{h_k}{\gamma_k} \le \frac{1}{2^k}\alpha$$

$$\frac{h_0}{\gamma_0} + \frac{h_1}{\gamma_1} + \cdots + \frac{h_{k-1}}{\gamma_{k-1}} = \frac{1}{\gamma} - \frac{1}{\gamma_k} < r = \frac{1-\sqrt{1-2h}}{\gamma}$$

Claim: For $k = 0, 1, \ldots,$ $x^k \in S_r(x^0)$, $Df(x^k)$ is nonsingular, and

$$\|Df(x^k)^{-1}(Df(y) - Df(x))\| \le \gamma_k \|y - x\|, \qquad x, y \in D.$$

We will also prove that

$$\|x^{k+1} - x^k\| \le \frac{h_k}{\gamma_k} \le \frac{1}{2^k}\alpha.$$

Proof by induction on k:
$k = 0 \Rightarrow$ satisfied (by our assumptions).
$k \ge 1$:

$$\begin{aligned}
\|x^k - x^0\| &\le \underbrace{\|x^k - x^{k-1}\|}_{\le \frac{h_{k-1}}{\gamma_{k-1}}} + \underbrace{\|x^{k-1} - x^{k-2}\|}_{\le \frac{h_{k-2}}{\gamma_{k-2}}} + \cdots + \underbrace{\|x^1 - x^0\|}_{\le \frac{h_0}{\gamma_0}} \\
&\le \frac{h_0}{\gamma_0} + \frac{h_1}{\gamma_1} + \cdots + \frac{h_{k-1}}{\gamma_{k-1}} \\
&< r
\end{aligned}$$

Thus, $x^k \in S_r(x^0)$.

$$Df(x^k) = \underbrace{Df(x^{k-1})}_{\text{nonsingular}} \Big(\underbrace{I - Df(x^{k-1})^{-1}(Df(x^{k-1}) - Df(x^k))}_{:=A} \Big) \tag{6.1}$$

$$\|A\| = \|Df(x^{k-1})^{-1}(Df(x^k) - Df(x^{k-1}))\| \le \gamma_{k-1} \underbrace{\|x^k - x^{k-1}\|}_{\le \frac{h_{k-1}}{\gamma_{k-1}}}$$

$$\le h_{k-1} < \frac{1}{2} < 1$$

Banach Lemma: If $\|A\| < 1$, then $I - A$ is nonsingular and

$$\|(I-A)^{-1}\| \le \frac{1}{1 - h_{k-1}}.$$

Thus, by (6.1), $Df(x^k)$ is nonsingular and

$$\begin{aligned}
\|Df(x^k)^{-1} \Bigg[\underbrace{Df(x^{k-1}) Df(x^{k-1})^{-1}}_{I} \Bigg] Df(y) - Df(x)\| &\le \| \underbrace{Df(x^k)^{-1} Df(x^{k-1})}_{(I-A)^{-1}} \| \|Df(x^{k-1})^{-1}(Df(y) - Df(x))\| \\
&\le \frac{1}{1 - h_{k-1}} \gamma_{k-1} \|y - x\| \\
&= \gamma_k \|y - x\|, \qquad x, y \in D.
\end{aligned}$$

$$
\begin{aligned}
x^{k+1} - x^k &= -Df(x^k)^{-1}\left(f(x^k)\underbrace{-f(x^{k-1}) - Df(x^{k-1})(x^k - x^{k-1})}_{=0}\right)\\
&= -Df(x^k)^{-1}\int_0^1 \Big(Df(x^{k-1} + t(x^k - x^{k-1})) - Df(x^{k-1})\Big)\,(x^k - x^{k-1})\,dt\\
\|x^{k+1} - x^k\| &\leq \int_0^1 \underbrace{\Big\|Df(x^k)^{-1}(Df(x^{k-1} + t(x^k - x^{k-1})) - Df(x^{k-1}))\Big\|}\,\|x^k - x^{k-1}\|\,dt\\
&\leq \gamma_k \int_0^1 \Big\|\cancel{x^{k-1}} + t(x^k - x^{k-1}) - \cancel{x^{k-1}}\Big\|\,\|x^k - x^{k-1}\|\,dt\\
&= \gamma_k \|x^k - x^{k-1}\|^2 \int_0^1 t\,dt\\
&= \frac{\gamma_k}{2}\underbrace{\|x^k - x^{k-1}\|^2}_{\leq\left(\frac{h_{k-1}}{\gamma_{k-1}}\right)^2} \leq \frac{\gamma_h}{2}\frac{h_{k-1}^2}{\gamma_{k-1}^2}\\
&= \frac{h_k}{\gamma_k}
\end{aligned}
$$

To be continued...

7 10-7-11

7.1 Continued from 10-5-11...

$$
\begin{aligned}
x^k &\in S_r(x^0) \\
\|Df(x^k)^{-1}(Df(y) - Df(x))\| &\leq \gamma_k \|y - x\|, \qquad x, y \in D \\
\|x^{k+1} - x^k\| &\leq \frac{h_k}{\gamma_k} \leq \frac{\alpha}{2^k}
\end{aligned}
$$

$f : D \to \mathbb{R}^n,\ f(x^*) = 0.$

Claim: (x^k) is a Cauchy sequence.
Let $m > k$.

$$
\begin{aligned}
\|x^m - x^k\| &\leq \|x^m - x^{m-1}\| + \|x^{m-1} - x^{m-2}\| + \cdots + \|x^{k+1} - x^k\| \\
&\leq \frac{\alpha}{2^{m-1}} + \frac{\alpha}{2^{m-2}} + \cdots + \frac{\alpha}{2^k} \\
&\leq \frac{\alpha}{2^k}\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots + \frac{1}{2^{m-k-1}}\right) \\
&\leq \frac{\alpha}{2^k}\left(\underbrace{1 + \frac{1}{2} + \frac{1}{4} + \cdots}_{=2}\right) \\
&\leq \frac{\alpha}{2^{k-1}}
\end{aligned}
$$

Thus, (x^k) is Cauchy and therefore convergent:

$$
\lim_{k\to\infty} x^k = x^*, \qquad x^* \in \overline{S_r(x^0)}.
$$

Claim: x^* is a zero of f.

$$
\begin{aligned}
f(x^k) &= -Df(x^k)(x^{k+1} - x^k) \qquad (7.1) \\
\|f(x^k)\| &\leq \underbrace{\|Df(x^k)\|}_{\leq \max_{x\in\overline{S_r(x^0)}} \|Df(x)\| := M} \underbrace{\|x^{k+1} - x^k\|}_{\leq \frac{\alpha}{2^k}} \\
&\leq \frac{\alpha M}{2^k}
\end{aligned}
$$

Thus,

$$
0 = \lim_{k\to\infty} f(x^k) = f(\lim_{k\to\infty} x^k) = f(x^*)
$$

Claim: the convergence is quadratic.

$$
\begin{aligned}
x^{k+1} &= x^k - Df(x^k)^{-1}f(x^k) && \text{by (7.1)}\\
x^{k+1} - x^* &= x^k - x^* - Df(x^k)^{-1}(f(x^k) - \underbrace{f(x^*)}_{=0})\\
&= \int_0^1 Df(x^k)^{-1}(Df(x^k) - Df(x^* + t(x^k - x^*)))(x^k - x^*)\, dt\\
\|x^{k+1} - x^*\| &\le \|x^k - x^*\| \cdot \int_0^1 \|Df(x^k)^{-1}(Df(x^k) - Df(x^* + t(x^k - x^*)))\|\, dt\\
&\le \|x^k - x^*\|\gamma_k \int_0^1 \underbrace{\|x^k - x^* - t(x^k - x^*)\|}_{=(1-t)\|x^k - x^*\|}\, dt\\
&= \|x^k - x^*\|^2 \gamma_k \int_0^1 (1-t)\, dt\\
&= \frac{\gamma_k}{2}\|x^k - x^*\|^2\\
&\le \frac{1}{2}\frac{\gamma}{\sqrt{1-2h}}\|x^k - x^*\|^2
\end{aligned}
$$

Note:

$$
\frac{d}{dt} f(x^* + t(x^k - x^*)) = Df(x^* + t(x^k - x^*))(x^k - x^*)
$$

Parameterizing in terms of t is the reason why we need complexity. □

7.2 Use of Newton in Practice

Remark 7.1. ***Monotonicity Test (Preliminary)***

Newton iterates: $x^{k+1} = x^k - \underbrace{Df(x^k)^{-1}f(x^k)}_{=\Delta x^k}$.

If all goes well: $x^k \to x^*$, $f(x^*) = 0$.

Check for progress: $\|f(x^{k+1})\| \le \theta \|f(x^k)\|$ for some $\theta < 1$. Note that this check is not affine-invariant!

Instead:

$$
\|Df(x^k)^{-1}f(x^{k+1})\| \le \theta \underbrace{\|Df(x^k)^{-1}f(x^k)\|}_{=\|\Delta x^k\|}
$$

In addition to

$$
Df(x^k)\Delta x^k = -f(x^k) \tag{7.2}
$$

we also need to solve a second system:

$$
Df(x^k)\overline{\Delta} x^{k+1} = -f(x^{k+1})
$$

Additional cost: $O(n^2)$ flop if LU factorization is used to solve (7.2).

Remark 7.2. ***Monotonicity Test (Refined)***

- Compute $\overline{\Delta}x^{k+1}$ in addition to Δx^k.
- Check if $\|\overline{\Delta}x^{k+1}\| \leq \theta\|\Delta x^k\|$ for some $\theta < 1$. (Typical value is $\theta = \frac{1}{2}$)
- If not satisfied, use Newton's method with damping

7.3 Newton's Method with Damping

Remark 7.3. ***Newton's Method: Standard vs. Damped***

Standard: $x^{k+1} = x^k + \Delta x^k$
Damped: $x^{k+1} = x^{k+1}(\lambda_k) = x^k + \lambda_k \Delta x^k$, where $0 < \lambda_k \leq 1$ for some damping factor.

What is a suitable strategy for selecting λ_k?

8 10-10-11

8.1 Newton's Method with Damping (Continued)

Remark 8.1. ***Goal of Damping***

$$\|Df(x^k)^{-1}f(x^{k+1})\| \leq \theta\|Df(x^k)^{-1}f(x^k)\| \qquad \text{for some } \theta < 1 \tag{8.1}$$

This can always be satisfied if λ_k is chosen small enough.

In the following,

$$\|\mathbf{v}\| := \sqrt{\mathbf{v}^T\mathbf{v}} = \text{ Euclidean norm.}$$

Proposition 8.2.

If $f(x^k) \neq \mathbf{0}$, then the Newton increment

$$x^k = Df(x^k)^{-1}f(x^k)$$

is a <u>descent direction</u> for the function

$$\phi(x) := \frac{1}{2}\|Df(x^k)^{-1}f(x)\|^2.$$

at x^k, i.e.

$$\frac{d}{d\lambda}\phi(x^k + \lambda\delta x^k)\big|_{\lambda=0} < 0.$$

Corollary 8.3.

If $\lambda_k > 0$ is chosen small enough, then

$$x^{k+1} = x^k + \lambda_k\Delta x^k$$

satisfies (8.1). This implies that

$$\phi(x^{k+1}) \leq \theta^2\phi(x^k).$$

Proof. (of Proposition 8.2)
Note: $f(x^k) \neq 0 \;\Rightarrow\; \Delta x^k \neq 0$.

$$\begin{aligned}
\frac{d}{d\lambda}\phi(x^k + \lambda\Delta x^k)\big|_{\lambda=0} &= D_x\phi(x)\big|_{x=x^k}\Delta x^k \\
D_x\phi(x) &= (Df(x^k)^{-1}f(x))^T Df(x^k)^{-1}Df(x) \\
&= -(\Delta x^k)^T\Delta x^k \\
&= -\|\Delta x^k\|^2 \\
&< 0
\end{aligned}$$

$\square$

Remark 8.4. ***Practical Strategy for Selecting λ***

Determine $0 < \lambda_k \leq 1$ such that the monotonicity test is satisfied with $\theta = 1 - \frac{\lambda_k}{2}$. If possible, use $\lambda_k = 1$ (which implies that $\theta = \frac{1}{2}$).

<u>Procedure</u>

- for $\lambda_k = 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots, \lambda_{\min}$
  - Solve $Df(x^k)\overline{\Delta}x^{k+1}(\lambda_k) = -f(x^k + \lambda_k \Delta x^k)$ and check if
  $$\|\overline{\Delta}x^{k+1}(\lambda_k)\| \leq \left(1 - \frac{\lambda_k}{2}\right) \|\Delta x^k\| \tag{8.2}$$
  - If (8.2) is satisfied, use λ_k and set $x^{k+1} = x^k + \lambda_k \Delta x^k$
  - In the next Newton step, use $\lambda_{k+1} = \min\{1, 2\lambda_k\}$ as the first value for checking (8.2).

8.2 Conditioning

<u>Problem:</u> Input $\to$ Output (or No Solution)

<u>Abstract Formulation:</u> $X \to Y$, $x \in X$ is the input, $y = F(x) \in Y$ is the corresponding output.

Example 8.5. ***General Problem We Are Considering***

Solution of $Ay = b$, where $A \in \mathbb{C}^{n\times n}$ is nonsingular, $b \in \mathbb{C}^n$.

$$\begin{aligned} X &= \left\{x = (A, b) \;\middle|\; A \in \mathbb{C}^{n\times n} \text{ is nonsingular, } b \in \mathbb{C}^n\right\} \\ Y &= \mathbb{C}^n \\ F(x) = F(A, b) &:= A^{-1}b = y \in \mathbb{C}^n \end{aligned}$$

Definition 8.6. ***Conditioning of a Problem***

The impact of perturbation: $x \to x + \delta x$, δx is small. In other words, we are solving a slightly wrong problem and we want to know the effect on the output.

Perturbation of input: $\delta x (= x + \delta x - x)$
Perturbation of output: $\delta F(x) (= F(x + \delta x) - F(x))$

The *condition number* of a problem, $\kappa = \kappa(x)$, is a measure of the sensitivity of the output to small changes in the input:
$$\frac{\|\delta F(x)\|}{\|x\|} \leq \kappa(x) \frac{\|\delta x\|}{\|x\|} \qquad \text{for small } \|\delta x\|$$

9 10-12-11

9.1 Conditioning (Continued)

We have $F : X \to Y,\ x + \delta x \to F(x) + \delta F(x)$.

$$\frac{\|\delta F(x)\|}{\|F(x)\|} \le \kappa(x)\frac{\|\delta x\|}{\|x\|} \qquad \text{for small } \|\delta x\|$$
$$\kappa(x) \le \frac{\|\delta F(x)\|}{\|F(x)\|}\frac{\|x\|}{\|\delta x\|}$$

Here, $\|\cdot\|$ are suitably chosen norms in X and Y.

Definition 9.1. *$\kappa(x)$*

$$\kappa(x) := \lim_{\substack{\delta\to 0\\ \|\delta x\|\le\delta}} \sup \frac{\|\delta F(x)\|}{\|F(x)\|}\frac{\|x\|}{\|\delta x\|}$$
$$\overset{\text{shorthand}}{=} \sup_{\delta x} \frac{\|\delta F(x)\|}{\|\delta x\|}\frac{\|x\|}{\|F(x)\|}$$

Definition 9.2. ***Ill-Conditioned, Well-Conditioned***

A problem is said to be *ill-conditioned* if $\kappa(x) \gg 1$ and *well-conditioned* otherwise.

How do we compute the condition number?

Remark 9.3. ***Conditioning Number for F Differentiable***

If $X \subset \mathbb{R}^n,\ Y \subset \mathbb{R}^m$ and $F : X \to Y$ is differentiable, then

$$\kappa(x) = \|DF(x)\|\frac{\|x\|}{\|F(x)\|}.$$

(Note: $\frac{\|\delta F(x)\|}{\|\delta x\|}$ is kind of like a difference quotient.)

Example 9.4.

Evaluation of $y = F(x) = \sqrt{x},\ x > 0$.

$$\begin{aligned} F'(x) &= \frac{1}{2}\frac{1}{\sqrt{x}} \\ \kappa(x) &= |F'(x)|\frac{|x|}{|F(x)|} = \frac{1}{2}\cdot\frac{1}{\sqrt{x}}\cdot\frac{x}{\sqrt{x}} \\ &= \frac{1}{2} \end{aligned}$$

So this is well-conditioned for all $x > 0$.

Example 9.5.

Computing $y = F(x) = x_1 - x_2$ for $x = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T \in \mathbb{R}^2$.
$F : \mathbb{R}^2 \to \mathbb{R}$. F is differentiable:

$$DF = \begin{bmatrix} 1 & -1 \end{bmatrix}$$

We compute:

$$\kappa(x) = \|DF(x)\|_2 \frac{\|x\|_2}{|F(x)|} = \sqrt{2}\frac{\sqrt{x_1^2 + x_2^2}}{|x_1 - x_2|}$$

So this problem is ill-conditioned if $x_1 \approx x_2,\ x_1, x_2 \not\approx 0$.

The reason why $F(x) = x_1 - x_2$ is ill-conditioned is that small perturbations of x_1 and x_2 get amplified:

$$\begin{aligned} F(x + \delta x) = (x_1 + \delta x_1) - (x_2 + \delta x_2) &= \underbrace{x_1 - x_2}_{\approx 0} + \delta x_1 - \delta x_2 \\ &\approx \delta x_1 - \delta x_2 \end{aligned}$$

This is called "loss of significant digits."

Remark 9.6.

Conditioning is only a property of the problem to be solved, not of a specific algorithm for obtaining its solution.

To compute a condition number in Matlab, use `cond(A)`.

9.2 Floating-Point Numbers

Problem: How do we represent $x \in \mathbb{R}$ on a computer?

Finite Storage: irrational numbers like $\sqrt{3} = 1.73205\ldots$ or even very large integers cannot be represented exactly!

Base of the representation: $\beta \geq 2$

- $\beta = 2$: binary representation
- $\beta = 10$: decimal representation

If $x \geq 0$ is an integer,

$$\begin{aligned} x &= b_n\beta^n + b_{n-1}\beta^{n-1} + \cdots + b_1\beta + b_0 \\ &=: (b_n b_{n-1} \cdots b_1 b_0)_\beta \end{aligned}$$

where the b_i's are integers such that $0 \leq b_i \leq \beta - 1$.

Example 9.7. ***53 in Base 10 and Base 2***

Let $x = 53$.

$$\begin{aligned} x &= 5 \cdot 10^1 + 3 \cdot 10^0 = (53)_{10} \\ &= 32 + 16 + 4 + 1 = 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 \\ &= (110101)_2 \quad \Rightarrow \text{ 6 bits} \end{aligned}$$

Example 9.8. ***How many numbers can we store?***

Suppose we use 52 bits to store x in a binary representation. How many numbers can we represent exactly?

$$x = (b_{51}b_{50} \cdots b_1 b_0)_2$$

Answer: $x = 0, 1, 2, \ldots, 2^{52} - 1$.

In general we need to use floating-point representation, even for integers.

9.3 IEEE Floating-Point Standard

Base $\beta = 2$. Any $x \in \mathbb{R}$, $x \neq 0$, can be represented in the binary form

$$x = \pm(1.b_1 b_2 \cdots) \times 2^p$$

where $b_1, b_2, \cdots \in \{0, 1\}$ and p is an integer.

10 10-14-11

10.1 Quick Review of the IEEE Standard

$\beta = 2,\ x \in \mathbb{R},\ x \neq 0$

$$x = (1.b_1\ b_2\ \ldots) \times 2^p$$

where $b_1, b_2, \ldots \in \{0, 1\}$ and p is an integer.

Example 10.1. ***41.7 in the IEEE standard***

$$\begin{aligned} \frac{41}{2} &= 20 \text{ remainder } 1 \\ \frac{20}{2} &= 10 \text{ remainder } 0 \\ \frac{10}{2} &= 5 \text{ remainder } 0 \\ \frac{5}{2} &= 2 \text{ remainder } 1 \\ \frac{2}{2} &= 1 \text{ remainder } 0 \\ \frac{1}{2} &= 0 \text{ remainder } 1 \end{aligned} \qquad \begin{aligned} 2 \times 0.7 &= 0.4 \text{ remainder } 1 \\ 2 \times 0.4 &= 0.8 \text{ remainder } 0 \\ 2 \times 0.8 &= 0.6 \text{ remainder } 1 \\ 2 \times 0.6 &= 0.2 \text{ remainder } 1 \\ 2 \times 0.2 &= 0.4 \text{ remainder } 0 \end{aligned}$$

Thus,

$$\begin{aligned} 41.7 &= (101001.1\overline{0110})_2 \\ &= +(1.01001100110\ldots) \times 2^5 \end{aligned}$$

This is its *normalized binary representation*.

10.2 Normalized IEEE Floating-Point Numbers

$$\underbrace{\pm}_{\text{sign}}(1.\underbrace{b_1\ b_2\ \ldots\ b_N}_{\text{mantissa}}) \times 2^{\overbrace{p}^{\text{exponent}}}$$

where $b_1, b_2, \ldots \in \{0, 1\}$ and p can be represented with M bits.

Precision	sign	N (length of mantissa)	M (exponent)	total bits	total bytes
Single	1	23	8	32	4
Double	1	52	11	64	8
Long Double	1	64	15	80	10

From now on we stick with double precision.

A double floating-point number looks like:

$$\pm(1.b_1\ b_2\ \ldots\ b_{52}) \times 2^p$$

where p is an integer in $[-1022, 1023]$.

Some numbers:

$$1 := +1.\underbrace{0\ 0\ \ldots\ 0}_{52} \times 2^0$$
$$1 + 2^{-52} := +1.\underbrace{0\ 0\ \ldots\ 0}_{51}\ 1 \times 2^0$$

10.3 Machine Precision

$1+2^{-52}$ is the smallest floating-point number that is still larger than 1. To check this, we loop `for n=0,1,2,...`
:

$$1 + 2^{-n} \overset{?}{>} 1$$

It should stop for $n = 53$.

Definition 10.2. ***Machine Precision***

Machine precision (or *machine epsilon*) is a measure of precision. Its value is

$$\epsilon_{\text{mach}} := 2^{-52} = (1 + 2^{-52} - 1)$$

(`eps` in Matlab)
Note:

$$\epsilon_{\text{mach}} = 2.220\ldots \times 10^{-16}$$

which is why double precision has ≈ 16 significant digits.

10.4 Floating-Point Representation

$$x \in \mathbb{R} \xrightarrow{\text{round to nearest}} fl(x) = \text{ floating-point number closest to } x$$

Remark 10.3. ***Chopping vs. Rounding to Nearest***

Consider

$$x = \pm 1.b_1\ b_2\ \cdots\ b_{52}\ b_{53}\ \cdots \times 2^p$$

- Chopping:
  - $x \to \pm 1.b_1\ b_2\ \cdots\ b_{52} \times 2^p$
- Round down: same as chopping
- Round up: 1 is added to b_{52}
- Rounding to Nearest:
  - If $b_{53} = 0$, round down
  - If $b_{53} = 1$ and $b_j = 1$ for at least one of $j > 53$, round up
  - If $b_{53} = 1$ and $b_j = 0$ for all $j > 53$:
    - round up if $b_{52} = 1$
    - round down if $b_{52} = 0$

Definition 10.4. ***Relative Error***

For $x \neq 0$, the *relative error* of $fl(x)$ is

$$\left|\frac{fl(x)-x}{x}\right| \leq \frac{1}{2}\epsilon_{\text{mach}} = 2^{-53} = 1.11\ldots \times 10^{-16}$$

Example 10.5. ***Representation of 41.7 with Rounding***

$$41.7 = +1.\underbrace{010011}_{6}\,\overbrace{\underbrace{0110}_{4}\,\underbrace{0110}_{4}\cdots\underbrace{0110}_{4}}^{11}\,0\underbrace{\;1\;}_{b_{52}}|10\;0110\;\ldots\;\times 2^5$$

$$fl(41.7) = +1.010011\;0110\;0110\;\ldots\;0110\;01 \times 2^5$$

11 10-17-11

11.1 Representation of the Exponent p

Double precision uses 11 bits:

$$e_{10}2^{10} + e_9 2^9 + \cdots + e_1 2^1 + e_0 2^0, \qquad e_0, e_1, \ldots, e_{10} \in \{0, 1\}$$

We can represent all integers from 0 to $2^{11} - 1 = 2047$. BUT p is written in the shifted form:

$$p = \underbrace{-1023}_{\text{fixed}} + e_{10}2^{10} + e_9 2^9 + \cdots + e_1 2^1 + e_0 2^0$$

Thus, p ranges from -1023 to 1024.

11.2 Machine Representation

s	$e_0 \quad e_1 \quad \cdots \quad e_{10}$	$b_1 \quad b_2 \quad \cdots \quad b_{52}$

$$\begin{array}{rl} x \neq \pm 0, \ \pm\infty : & s \ e_0 \ e_1 \ \ldots \ e_{10} \mid b_1 \ b_2 \ \ldots \ b_{52} \\ x = \pm 0 : & s \ 0 \ 0 \ \ldots \ 0 \mid 0 \ 0 \ \ldots \ 0 \\ x = \pm\infty : & s \ 1 \ 1 \ \ldots \ 1 \mid 0 \ 0 \ \ldots \ 0 \end{array}$$

Undefined Cases:

$$\begin{array}{rll} \text{NaN:} & s \ 1 \ 1 \ \ldots \ 1 \mid b_1 \ b_2 \ \ldots \ b_{52}, & \text{at least one } b_j = 1 \\ \underbrace{\text{subnormal floating point numbers:}}_{<2^{-1022}} & s \ 0 \ 0 \ \ldots \ 0 \mid b_1 \ b_2 \ \ldots \ b_{52}, & \text{at least one } b_j = 1 \\ 2^{-1022}: & 1 \ 1 \ 0 \ \ldots \ 0 \mid 0 \ 0 \ \ldots \ 0 & \end{array}$$

11.3 Floating Point Arithmetic

Example 11.1. ***41.7 $\oplus$ 10.425***

$$\begin{aligned} x &= fl(41.7) = 1.010011\ 0110\ 0110\ \ldots\ 0110\ 01 \times 2^5 \\ y &= fl(\underbrace{10.425}_{=41.7/4}) = 1.010011\ 0110\ 0110\ \ldots\ 0110\ 01 \times 2^3 \end{aligned}$$

Addition of x and y in floating-point arithmetic:

$$\begin{aligned} x &= +1.0100110110\ldots011001 \mid \qquad \times 2^5 \\ y &= +0.010100110110\ldots0110 \mid 01 \ \times 2^5 \\ x + y &= 1.101000010\ldots00000 \mid 01 \ \times 2^5 \\ \underbrace{fl(x+y)}_{=x \oplus y \neq x+y} &= 1.101000010\ldots00000 \mid \qquad \times 2^5 \end{aligned}$$

Exact Arithmetic	floating-point arithmetic
$x + y$	$x \oplus y$
$x - y$	$x \ominus y$
$x \times y$	$x \otimes y$
x/y	$x \oslash y$

For all floating-point numbers x, y:

$$x \oplus y = (x+y)(1+\epsilon) \qquad \text{where } |\epsilon| \le \frac{1}{2}\epsilon_{\text{mach}} = 2^{-53}$$

$$x \ominus y = (x-y)(1+\epsilon) \qquad \text{where } |\epsilon| \le \frac{1}{2}\epsilon_{\text{mach}}$$

$$x \otimes y = (x \times y)(1+\epsilon) \qquad \text{where } |\epsilon| \le \frac{1}{2}\epsilon_{\text{mach}}$$

$$(y \neq 0) \qquad x \oslash y = (x/y)(1+\epsilon) \qquad \text{where } |\epsilon| \le \frac{1}{2}\epsilon_{\text{mach}}$$

Remark 11.2.

The relative error of the floating-point implementations of the four basic arithmetic operations is bounded by

$$\frac{1}{2}\epsilon_{\text{mach}} = 2^{-53} = 1.11 \ \ldots \ \times 10^{-16}.$$

11.4 Catastrophic Effects of Round-Off Errors

Loss of Significant Digits

Occurs in subtraction of nearly equal numbers, i.e.

$$x - y, \qquad \text{where } x \approx y.$$

Example 11.3. ***Relative Error of $x - y$ when $x \approx y$***

$$x = 1 + 2^{-52} + 2^{-53} + 2^{-54} \qquad = +1.0\ldots01 \mid 110\ldots \times 2^0$$

$$y = 1 + 2^{-54} \qquad = +1.0\ldots00 \mid 010\ldots \times 2^0$$

Exact arithmetic:

$$x - y = 2^{-52} + 2^{-53} = 3 \times 2^{-53}$$

Floating-point arithmetic:

$$\begin{aligned} fl(x) &= +1.0\ldots010 \mid \times 2^0 \\ fl(y) &= +1.0\ldots000 \mid \times 2^0 \\ fl(x) - fl(y) &= +0.0\ldots010 \mid \times 2^0 = +1.0\ldots0 \times 2^{-51} \end{aligned}$$

Relative error:

$$\left| \frac{2^{-51} - 3 \times 2^{-53}}{3 \times 2^{-53}} \right| = \frac{1}{3}$$

12 10-19-11

12.1 Loss of Significant Digits

$$x - y, \qquad x \approx y$$

In practice, this problem can often be avoided easily.

Example 12.1. ***Trouble Is Avoidable***

Evaluation of

$$f(x) = \frac{1-\cos x}{\sin^2 x} \qquad \text{for } 0 < x < \pi$$

Problem: $\cos x \approx 1$ for $x \approx 0$.

$$\begin{aligned} f(x) &= \frac{1-\cos x}{\sin^2 x} \cdot \frac{1+\cos x}{1+\cos x} = \frac{1-\cos^2 x}{\sin^2 x} \cdot \frac{1}{1+\cos x} \\ &= \frac{1}{1+\cos x} \end{aligned}$$

But $\cos x \approx -1$ for $x \approx \pi$.
$\Rightarrow$ use

$$f(x) = \begin{cases} \frac{1}{1+\cos x} & 0 < x \le \frac{\pi}{2} \\ \frac{1-\cos x}{\sin^2 x} & \frac{\pi}{2} < x < \pi \end{cases}$$

Example 12.2. ***Huge Intermediate Quantities***

$$f(x) = \frac{1}{x}\left(\frac{\frac{1}{x}}{\frac{1}{x}-1} - 1\right), \qquad 0 < x < 1$$

Suppose we want to evaluate this at $x = 2^{-54}$.
In exact arithmetic:

$$\begin{aligned} y = f(2^{-54}) &= 2^{54}\left(\frac{2^{54}}{2^{54}-1} - 1\right) \\ &= \frac{1}{1-2^{-54}} \\ &\approx 1 \end{aligned}$$

In floating-point arithmetic:

$$\begin{aligned} \tilde{y} &= 2^{54} \otimes \left((2^{54} \oslash \underbrace{(2^{54} \ominus 1)}) \ominus 1\right) \\ &= 2^{54} \otimes \left(\underbrace{(2^{54} \oslash 2^{54}}) \ominus 1\right) \\ &= 2^{54} \otimes (\underbrace{1 \ominus 1}) \\ &= 2^{54} \otimes 0 \\ &= 0 \end{aligned}$$

$$\text{Relative Error } = \left|\frac{\tilde{y}-y}{y}\right| = 1$$

The problem is the huge intermediate quantity 2^{54}.
<u>Remedy:</u>

$$f(x) = \frac{1}{x}\left(\frac{1}{\frac{1}{x}-1}\right) = \frac{1}{1-x}$$

In floating-point arithmetic:

$$\begin{aligned} \tilde{\tilde{y}} &= 1 \oslash (1 \ominus 2^{-54}) = 1 \oslash 1 \\ &= 1 \end{aligned}$$

$$\text{Relative Error } = \left|\frac{\tilde{\tilde{y}}-y}{y}\right| = \left|\frac{1 - \frac{1}{1-2^{-54}}}{\frac{1}{1-2^{-54}}}\right| = 2^{-54} < \epsilon_{\text{mach}}$$

12.2 Stability

Abstract formulation of a problem:

$$\text{Input } x \in X \underbrace{\xrightarrow{F}}_{F:X\to Y} F(x) = y \in Y, \ y \text{ is the output}$$

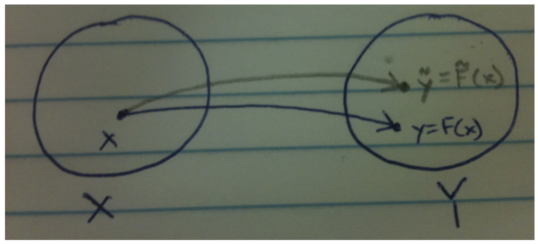
An algorithm in floating-point arithmetic:

$$\tilde{F} : X \to Y$$

$\tilde{F}$ captures all errors:

- $x \to fl(x)$
- round-off errors of floating-point arithmetic
- possible approximation error (e.g. stopping Newton's method)

Exact solution of the problem: $y = F(x)$
Computed solution: $\tilde{y} = \tilde{F}(x)$



Relative error of the computed solution:

$$\frac{\|\tilde{y} - y\|}{\|y\|} = \frac{\|\tilde{F}(x) - F(x)\|}{\|F(x)\|}$$

Ideally we would like to have

$$\frac{\|\tilde{F}(x) - F(x)\|}{\|F(x)\|} = O(\epsilon_{\text{mach}}) \quad \text{for all } x \in X \tag{12.1}$$

Recall: $g(\epsilon) = O(\epsilon) \;\Leftrightarrow\;$ there exists a constant C such that $\|g(\epsilon)\| \leq C|\epsilon|$ for $\epsilon \to 0$.

If the condition number is 1, then (12.1) is possible. If the condition number is big, i.e. the problem is ill-conditioned, then (12.1) is unrealistic.

Instead...

Definition 12.3. ***Stable***

The "algorithm" $\tilde{F} : X \to Y$ for solving the problem is *stable* if for each $x \in X$, there exists an $\tilde{x} \in X$ such that

$$(1) \qquad \frac{\|\tilde{F}(x) - F(\tilde{x})\|}{\|F(\tilde{x})\|} = O(\epsilon_{\text{mach}})$$

$$(2) \qquad \frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_{\text{mach}})$$

Meaning of Stable
A stable algorithm gives nearly the right solution to nearly the right problem.

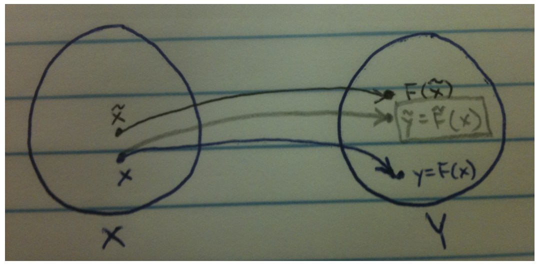


Figure 2: $\tilde{y} = \tilde{F}(x)$ is the computed solution.

This is a realistic goal.

13 10-21-11

13.1 Backward Stability

Definition 13.1. ***Backward Stability***

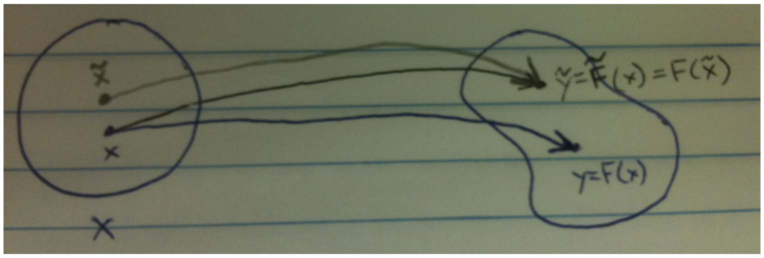
A stronger form of stability:
The "algorithm" $\tilde{F} : X \to Y$ for solving the problem $F : X \to Y$ is said to be *backward stable* if for each input there exists an $\tilde{x} \in X$ such that

$$\tilde{F}(x) = F(\tilde{x})$$

and

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_{\text{mach}})$$

Meaning of Backward Stability:
A backward stable algorithm gives the right solution to nearly the right problem.



Example 13.2. ***Backward Stability of Floating-Point Subtraction***

$$y = F(x) = x_1 - x_2, \qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2$$

$$\begin{aligned}
\tilde{y} = \tilde{F}(x) &= fl(x_1) \ominus fl(x_2) \\
&= [x_1(1+\epsilon_1) - x_2(1+\epsilon_2)](1+\epsilon_3) \qquad \text{where } |\epsilon_1|, |\epsilon_2|, |\epsilon_3| \le \frac{1}{2}\epsilon_{\text{mach}} \\
&= \underbrace{x_1(1+\epsilon_1)(1+\epsilon_3)}_{=:\tilde{x}_1} - \underbrace{x_2(1+\epsilon_2)(1+\epsilon_3)}_{=:\tilde{x}_2} \\
&= F(\tilde{x}) \\
\left|\frac{\tilde{x}_1 - x_1}{x_1}\right| &= |(1+\epsilon_1)(1+\epsilon_3) - 1| = |\epsilon_1 + \epsilon_3 + \epsilon_1\epsilon_3| \\
&\le |\epsilon_1| + |\epsilon_3| + |\epsilon_1||\epsilon_3| = O(\epsilon_{\text{mach}}) \\
\left|\frac{\tilde{x}_2 - x_2}{x_2}\right| &= O(\epsilon_{\text{mach}}) \qquad \text{by similar analysis}
\end{aligned}$$

The floating point algorithm for subtraction is backward stable.

Remark 13.3. ***Backward Stability of Floating-Point Addition, Multiplication, and Division***

Similar to Example 13.2, we can show that addition, multiplication, and division are backward stable.

Example 13.4. ***Stable, but not Backward Stable***

$$
\begin{aligned}
y &= F(x) = x + 1, \quad x \in \mathbb{R} \\
\tilde{y} &= \tilde{F}(x) = fl(x) \oplus 1 = [x(1+\epsilon_1)+1](1+\epsilon_2) \qquad \text{where } |\epsilon_1|, |\epsilon_2| \le \frac{1}{2}\epsilon_{\text{mach}} \\
&= \underbrace{x(1+\epsilon_1)(1+\epsilon_2)+\epsilon_2}_{=:\tilde{x}} + 1 = \tilde{x} + 1 = F(\tilde{x}) \left|\frac{\tilde{x}-x}{x}\right| \quad = \left|\frac{x(1+\epsilon_1)(1+\epsilon_2)+\epsilon_2 - x}{x}\right| \\
&= \left|\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2 + \frac{\epsilon_2}{x}\right| = O(\epsilon_{\text{mach}}) + O\left(\frac{\epsilon_{\text{mach}}}{|x|}\right) \\
&\ne O(\epsilon_{\text{mach}}) \quad \text{if } x \approx 0
\end{aligned}
$$

Thus, this algorithm is not backward stable. Is it stable?

$$
\begin{aligned}
F(x) &= x + 1 \\
\tilde{F}(x) &= \underbrace{x(1+\epsilon_1)}_{=:\tilde{x}}(1+\epsilon_2) + \epsilon_2 + 1 \\
\tilde{F}(\tilde{x}) - F(\tilde{x}) &= (\tilde{x}+1)(1+\epsilon_2) - (\tilde{x}+1) = (\tilde{x}+1)\epsilon_2 \\
&= F(\tilde{x})\epsilon_2 \\
|\tilde{F}(x) - F(\tilde{x})| &= |F(\tilde{x})||\epsilon_2| = |F(\tilde{x})|O(\epsilon_{\text{mach}}) \\
\tilde{x} - x &= x\epsilon_1 \\
|\tilde{x} - x| &= |x|O(\epsilon_{\text{mach}})
\end{aligned}
$$

Thus, it is stable.

13.2 Accuracy of Backward Stable Algorithms

Theorem 13.5.

Let the problem $F : X \to Y$ have condition number $\kappa = \kappa(x)$, $x \in X$, and let $\tilde{F} : X \to Y$ be a backward stable algorithm for solving the problem. Then:

$$
\underbrace{\frac{\|\tilde{F}(x) - F(x)\|}{\|F(x)\|}}_{=\frac{\|\tilde{y}-y\|}{\|y\|}} = O(\kappa(x)\epsilon_{\text{mach}}) \qquad \text{for all } x \in X.
$$

Thus, if the condition number is really high, then even a backward stable algorithm cannot save you.

14 10-24-11

14.1 Backward Stability (Continued)

Theorem 14.1.

$$F : X \to Y, \quad \kappa(x)$$
$$\tilde{F} : X \to Y \quad \text{backward stable}$$

$$\frac{\|\tilde{F}(x) - F(x)\|}{\|F(x)\|} = O(\kappa(x)\epsilon_{\text{mach}}) \quad \forall\, x \in X$$

Proof. Let $x \in X$ and let $\tilde{F}$ be backward stable. Then

$$\tilde{F}(x) = F(\tilde{x}) \qquad \text{for some } \tilde{x} \in X \text{ with } \frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_{\text{mach}}).$$

Recall:

$$\kappa(x) = \sup_{\delta x} \frac{\|\delta F(x)\|}{\|\delta x\|} \cdot \frac{\|x\|}{\|F(x)\|}$$

Then

$$\begin{aligned}
\frac{\|\tilde{F}(x) - F(x)\|}{\|F(x)\|} &= \frac{\|F(\tilde{x}) - F(x)\|}{\|F(x)\|} && \text{(backward stability)} \\
&= \frac{\|\delta F(x)\|}{\|F(x)\|} && (\delta x := \tilde{x} - x) \\
&= \left(\frac{\|\delta F(x)\|}{\|\delta x\|} \cdot \frac{\|x\|}{\|F(x)\|} \right) \frac{\|\tilde{x} - x\|}{\|x\|} \\
&\le (\kappa(x) + \underbrace{o(1)}_{\to 0 \text{ as } \epsilon_{\text{mach}} \to 0}) \underbrace{\frac{\|\tilde{x} - x\|}{\|x\|}}_{= O(\epsilon_{\text{mach}})} \\
&= O(\kappa(x)\epsilon_{\text{mach}})
\end{aligned}$$

□

14.2 Norms

Definition 14.2. ***Norm***

A *norm* on $\mathbb{C}^n$ is a function

$$\| \cdot \| : \mathbb{C}^n \to \mathbb{R}$$

such that for all $x, y \in \mathbb{C}^n$, $\alpha \in \mathbb{C}$:

1. $\|x\| \geq 0$, and $\|x\| = 0$ if and only if $x = 0$
2. $\|x + y\| \leq \|x\| + \|y\|$
3. $\|\alpha x\| = |\alpha|\|x\|$

Example 14.3. ***Some Norms***

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$
$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$$
$$\|x\|_\infty = \max_{1\le i\le n} |x_i|$$

Example 14.4. ***Vec Norm***

Let $A \in \mathbb{C}^{n\times n}$. $\mathrm{vec}(A) := [a_{11}\ a_{21}\ \cdots\ a_{n1}\ a_{12}\ \cdots\ a_{nn}]^T \in \mathbb{C}^{n^2}$. $\|A\| := \|\mathrm{vec}(A)\|$, where $\|\cdot\| : \mathbb{C}^{n^2} \to \mathbb{R}$.

Example 14.5. ***Frobenius Norm***

$$\|A\|_F = \|\mathrm{vec}(A)\|_2 = \sqrt{\sum_{j,k=1}^{n} |a_{jk}|^2}$$

14.3 Matrix Norms

Definition 14.6. ***Matrix Norm***

Any *matrix norm* satisfies:

1. $\|A\| \ge 0$, with $\|A\| = 0$ if and only if $A = \mathbf{0}$
2. $\|A + B\| \le \|A\| + \|B\|$
3. $\|\alpha A\| = |\alpha|\|A\|$

Definition 14.7. ***Induced Matrix Norm***

Often more useful are *induced matrix norms*:

Let $\|\cdot\| : \mathbb{C}^n \to \mathbb{R}$ be a matrix norm on $\mathbb{C}^n$. Set

$$\operatorname{lub}(A) = \operatorname{lub}_{\|\cdot\|}(A) := \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1} \|Ax\|$$

It is easy to see that $\operatorname{lub}(A)$ is indeed a matrix norm.

Note: $\|Ax\| \operatorname{lub}_{\|\cdot\|}(A)\|x\|$ for all $A \in \mathbb{C}^{n\times n},\ x \in \mathbb{R}^n$.

Norm on $\mathbb{C}^n$	Induced Matrix Norm
$\|\cdot\|_1$	$\|A\|_1 := \operatorname{lub}_{\|\cdot\|}(A) = \max_{1\le k\le n} \sum_{j=1}^{n} \lvert a_{jk}\rvert$ (max column sum)
$\|\cdot\|_\infty$	$\|A\|_\infty := \operatorname{lub}_{\|\cdot\|_\infty}(A) = \max_{1\le j\le n} \sum_{k=1}^{n} \lvert a_{jk}\rvert$ (max row sum)
$\|\cdot\|_2$	$\|A\|_2 := \operatorname{lub}_{\|\cdot\|}(A) = \max_{\|x\|_2=1} \|Ax\|_2 = \sigma_{\max}(A) =$ the largest *singular value* of A

$$\sigma_{\max} = \sqrt{\lambda_{\max}(A^H A)}$$

Theorem 14.8. ***lub Norms Are Submultiplicative***

lub norms are *submultiplicative*:

$$\operatorname{lub}(AB) \le \operatorname{lub}(A)\operatorname{lub}(B) \qquad \forall\ A, B \in \mathbb{C}^{n\times n}$$

Proof.

$$\operatorname{lub}(AB) = \max_{x\neq 0} \frac{\|ABx\|}{\|x\|}$$

For $x \neq 0$, $y := Bx \neq 0$

$$\frac{\|ABx\|}{\|x\|} = \frac{\|ABx\|}{\|Bx\|}\cdot\frac{\|Bx\|}{\|x\|} = \frac{\|Ay\|}{\|y\|}\frac{\|Bx\|}{\|x\|}$$
$$\le \operatorname{lub}(A)\operatorname{lub}(B)$$

□

15 10-26-11

15.1 Conditioning of $Ax = b$

Let $A \in \mathbb{C}^{n\times n}$ be nonsingular.
Let $b \in \mathbb{C}^n$ be fixed, and consider small perturbations δA of A.

$$\left.\begin{array}{c} Ax = b \\ (A+\delta A)(x+\delta x) = b \end{array}\right\} \quad \Rightarrow \quad A(\delta x) + (\delta A)x + \underbrace{(\delta A)(\delta x)}_{\approx 0} = 0$$

$$A(\delta x) \approx -(\delta A)x \quad \Rightarrow \quad \delta x \approx -A^{-1}(\delta A)x$$

Let $\|\cdot\|$ be any vector norm on $\mathbb{C}^n$ and $\|\cdot\|$ be the associated lub norm on $\mathbb{C}^{n\times n}$.

$$\begin{aligned} \|\delta x\| &\approx \|A^{-1}(\delta A)x\| \leq \|A^{-1}(\delta A)\|\|x\| \\ &\leq \|A^{-1}\|\|\delta A\|\|x\| \\ \frac{\|\delta x\|}{\|x\|}\frac{\|A\|}{\|\delta A\|} &\lessapprox \|A\|\|A^{-1}\| =: \kappa(A)\big(= \kappa_{\|\cdot\|}(A)\big) \end{aligned}$$

Recall our definition of $\kappa(x)$:

$$\sup_{\delta A} \frac{\|\delta x\|}{\|x\|}\frac{\|A\|}{\|\delta A\|} = \kappa(A) = \|A\|\|A^{-1}\|$$

Theorem 15.1.

Let $b \in \mathbb{C}^n$ be fixed. Let $\|\cdot\|$ be a vector norm on $\mathbb{C}^n$ and let $\|\cdot\|$ be the associated lub norm. Let A be nonsingular. The conditioning number of solving $Ax = b$ is given by

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

Corollary 15.2.

$\kappa(A) \geq 1$ for all A (since we are using lub norms).

Proof.

$$\begin{aligned} I &= AA^{-1} \\ \|I\| &= \|AA^{-1}\| \\ &\leq \|A\|\|A^{-1}\| \\ &= \kappa(A) \\ \|I\| &= \max_{\|x\|=1} \|Ix\| = 1 \end{aligned}$$

□

15.2 Stability of LU Factorization

Let $A \in \mathbb{C}^{n \times n}$ be nonsingular. Assume that A has an LU factorization: $A = LU$.

Then: for all "sufficiently small" ϵ_{mach}, LU factorization without partial pivoting in floating-point arithmetic encounters no zero pivots. The algorithm will run to completion, computing a lower-triangular matrix $\tilde{L}$ and an upper-triangular matrix $\tilde{U}$ such that

$$\tilde{L}\tilde{U} = A + \delta A \qquad \text{and} \qquad \frac{\|\delta A\|}{\|L\| \cdot \|U\|} = O(\epsilon_{\text{mach}}).$$

If $\|L\| \cdot \|U\| = O(\|A\|)$, then

$$\frac{\|\delta A\|}{\|A\|} = O(\epsilon_{\text{mach}})$$

and LU factorization without pivoting is backward stable.

However, for general nonsingular $A \in \mathbb{C}^{n \times n}$,

$$\|L\| \cdot \|U\| \neq O(\|A\|)$$

In fact, without pivoting $\|L\|$ can be arbitrarily large (see the 2×2 example we did).

$\Rightarrow$ LU factorization without pivoting is not backward stable (and not even stable).

15.3 LU Factorization with Partial Pivoting

Partial pivoting ensures that $|l_{jk}| \leq 1$ for all $j, k \Rightarrow \|L\| = O(1)$. But what can we say about $\|U\|$?

Define the so-called growth factor.

Definition 15.3. ***Growth Factor***

$$\rho = \rho(A) = \frac{\max\limits_{j,k=1,2,\ldots,n} |u_{jk}|}{\max\limits_{j,k=1,2,\ldots,n} |a_{jk}|}$$

Theorem 15.4.

Let $A \in \mathbb{C}^{n \times n}$ be a nonsingular matrix. Then LU factorization with partial pivoting in floating-point arithmetic generates matrices $\tilde{L}$, $\tilde{U}$, and $\tilde{P}$ such that

$$\tilde{L}\tilde{U} = \tilde{P}A + \delta A$$

and $\frac{\|\delta A\|}{\|A\|} = O(\rho\epsilon_{\text{mach}})$, where $\rho = \rho(A)$ is the growth factor.

Consequence:
If $\rho(A)$ is bounded by a constant for all nonsingular matrices $A \in \mathbb{C}^{n \times n}$, then

$$\frac{\|\delta A\|}{\|A\|} = O(\epsilon_{\text{mach}})$$

and thus, LU factorization with partial pivoting is backward stable.

This is indeed the case. It can be shown that

$$\rho(A) \leq 2^{n-1} \qquad \text{for all nonsingular matrices } A \in \mathbb{C}^{n\times n}$$

and the inequality is sharp. However, in practice, $\rho(A)$ does not approach this bound.

16 10-28-11

16.1 Backward Stability of LU Factorization with Partial Pivoting (Continued)

Example 16.1. ***Large*** $\rho(A)$

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ -1 & 1 & 0 & \cdots & 0 & 1 \\ -1 & -1 & 1 & \ddots & 0 & 1 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & -1 & 1 \end{bmatrix} \in \mathbb{R}^{n\times n}$$

$$A \to \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 0 & \cdots & 0 & 2 \\ 0 & -1 & 1 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 & 2 \\ 0 & -1 & -1 & \ddots & 1 & 2 \\ 0 & -1 & -1 & \cdots & -1 & 2 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 0 & \cdots & 0 & 2 \\ 0 & 0 & 1 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 & 2^{n-3} \\ 0 & 0 & 0 & \ddots & 1 & 2^{n-2} \\ 0 & 0 & 0 & \cdots & 0 & 2^{n-1} \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ -1 & -1 & 1 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 & 0 \\ -1 & -1 & -1 & \ddots & 1 & 0 \\ -1 & -1 & -1 & \cdots & -1 & 1 \end{bmatrix}$$

$$\rho(A) = 2^{n-1}$$

Example 16.2. ***Large $\rho(A)$ (Continued)***

For nonsingular matrices $A \in \mathbb{C}^{n\times n}$ of all sizes $n \geq 1$, LU factorization with partial pivoting is not backward stable!

Reason: $\frac{\|\delta A\|}{\|A\|} = O(\rho(A)\epsilon_{\text{mach}}) \neq O(\epsilon_{\text{mach}})$, since $\rho(A) = 2^{n-1} \to \infty$ as $n \to \infty$.

Good News: In practice, matrices with large $\rho(A)$ have never occurred!

Remark 16.3. ***Backward Stability of LU Factorization***

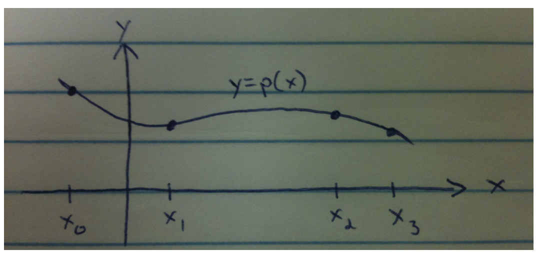
LU factorization with partial pivoting is backward stable in practice!

16.2 Interpolation

Problem: Given $n+1$ data points

$$(x_j, y_j), \quad j = 0, 1, \ldots, n \tag{16.1}$$

where $x_j, y_j \in \mathbb{R}$ and $x_0 < x_1 < x_2 < \cdots < x_n$.



Find a "simple" function p that interpolates the data.

16.2.1 Polynomial Interpolation

Remark 16.4. ***Polynomial Notation***

$$\begin{aligned} P_n \in \prod_n :&= \left\{P(x) = c_0 + c_1 x + \cdots + c_n x^n \;\middle|\; c_0, c_1, \ldots, c_n \in \mathbb{R}\right\} \\ &= \text{ set of all (real) polynomials of degree} \leq n \end{aligned}$$

Theorem 16.5.

There is a unique $P_n \in \prod_n$ that interpolates the data (16.1):

$$P_n(x) = \sum_{j=0}^{n} y_j \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x - x_k}{x_j - x_k}$$

(*Lagrange interpolation formula*)

Proof.

$$P_n(x_l) = \sum_{j=0}^{n} y_j \prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x_l - x_k}{x_j - x_k}$$

$$\prod_{\substack{k=0 \\ k \neq j}}^{n} \frac{x_l - x_k}{x_j - x_k} = \begin{cases} 0 & l \neq j \\ 1 & l = j \end{cases}$$

$$P_n(x_l) = y_l \qquad \forall\ l$$

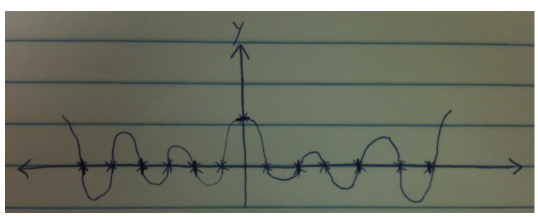
<u>P_n is unique:</u> Let $Q \in \prod_n$ be another interpolating polynomial. Then

$$D(x) = P_n(x) - Q(x) \in \prod_n$$

$$D(x_j) = P_n(x_j) - Q(x_j) = y_j - y_j = 0, \qquad j = 0, 1, \ldots, n$$

Thus, D has at least $n+1$ zeros. Since D is a polynomial of degree n, this implies that $D = 0 \ \Rightarrow\ P_n = Q$. $\square$

<u>But...</u>
Polynomials are usually not <u>flexible</u> enough.



<u>Remedy:</u> piecewise polynomials $\Rightarrow$ *splines*

16.2.2 Splines

<u>Given:</u> data points $(x_0, y_0),\ (x_1, y_1),\ (x_2, y_2),\ \ldots,\ (x_n, y_n)$, where $a := x_0 < x_1 < x_2 < \cdots < x_n =: b$ and $n \geq 1$.

Definition 16.6. ***Spline***

A function $S : [a, b] \to \mathbb{R}$ is called a *spline* of degree $k - 1$ (order k) if

(a) $S \in C^{k-2}[a, b]$

(b) On each interval $[x_{j-1}, x_j],\ j = 1, 2, \ldots, n,\ S(x) \in \prod_{k-1}$

17 10-31-11

$$
\begin{aligned}
a = x_0 < x_1 < x_2 < \ldots < x_n = b \\
S \in C^{k-2}[a,b] \\
S(x) \in \prod_{k-1}, \quad x_{j-1} \le x_j
\end{aligned}
$$

$k \ge 2$, k order, $k-1$ degree

17.1 Working with Splines

Example 17.1. ***Linear & Cubic Splines***

$k = 2 \;\Rightarrow$ linear spline
$k = 4, \;\Rightarrow$ cubic spline

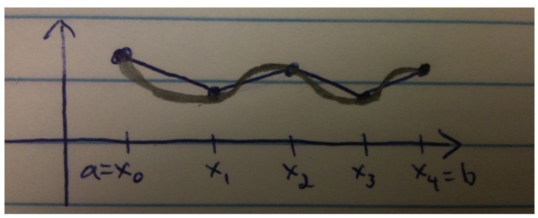


Figure 3: Linear spline (pen) and cubic spline (pencil).

Remark 17.2. ***Counting Degrees of Freedom***

On each $[x_{j-1}, x_j]$: $S = S_j \in \prod_{k-1} \;\Rightarrow\; k$ coefficients.
$\Rightarrow$ The total number of parameters $= nk$ coefficients.

$S \in C^{k-2}$. For each x_j, $j = 1, 2, \ldots, n-1$, $S_j^{(i)}(x_j) = S^{(i)}(x_j - 0) = S^{(i)}(x_j + 0) = S_{j+1}^{(i)}(x_j)$, $i = 0, 1, \ldots, k-2$. $\Rightarrow$ This is a total of $(n-1)(k-1)$ conditions.
Use a spline of degree $k-1$ to interpolate the data (x_0, y_0), (x_1, y_1), $\ldots$, (x_n, y_n):

$$S(x_j) = y_j, \quad j = 0, 1, \ldots, n$$

$\Rightarrow$ Another $n+1$ conditions.

$$
\begin{aligned}
\#\text{ coefficients} - \#\text{ conditions} &= nk - (n-1)(k-1) - (n-1) \\
&= k - 2
\end{aligned}
$$

Thus, we need to impose $k-2$ more conditions in order to get a unique spline.

Example 17.3.

For $k = 2$, there is a unique linear spline that interpolates the data.
For $k = 4$, we need 2 additional conditions to have a unique cubic spline that interpolates the data.

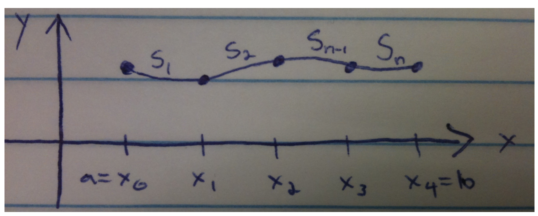
17.1.1 Cubic Splines

$k = 4$, $S \in C^2[a, b]$,

$$S(x) = S_j(x) \in \prod_3 \quad \text{on } [x_{j-1}, x_j],\ j = 1, 2, \ldots, n.$$

Remark 17.4. ***Additional 2 Conditions***

1. Natural spline: $S''(a) = S_1''(a) = 0,\ S''(b) = S_n''(b) = 0.$
2. Not-a-knot condition: we need that $n \geq 3$. $S_1'''(x_1) = S_2'''(x_1),\ S_{n-1}'''(x_{n-1}) = S_n'''(x_{n-1})$.



   Recall: $S_1, S_2 \in \prod_3$, $S_1, S_2 \in C^2$, $S_1^{(i)}(x_1) = S_2^i(x_2)$ for $i = 0, 1, 2, 3$. Thus, $S_1 = S_2$ for all $x_0 \leq x \leq x_2$, i.e. x_1 is not really a knot. Similarly, x_{n-1} is not really a knot.
   $\Rightarrow$ This condition is the default for Matlab's `spline` function.
3. Periodic spline: Assume that the data is periodic, i.e. $y_0 = y_n$ $(S_1(a) = S_n(b))$.

$$S_1'(a) = S_n'(b)$$
$$S_1''(a) = S_n''(b)$$

4. Clamped spline: $S_1'(a) = v_0,\ S_n'(b) = v_n$, where $v_0, v_n \in \mathbb{R}$ are given.

Theorem 17.5.

For each of the 4 cases in Remark 17.4, there exists a unique cubic spline that interpolates the data.

Remark 17.6.

Cubic splines are the "smoothest" interpolants among all interpolating functions $f \in C^2[a, b]$.

$$\|f\|_2 = \left(\int_a^b |f(x)|^2 \, dx \right)^{1/2}$$

Theorem 17.7.

Let S be a cubic spline that interpolates the data. Let $f \in C^2[a,b]$ be any function that also interpolates the data: $f(x_j) = y_j,\ j = 0, 1, \ldots, n$. Assume that

$$S''(x)[f'(x) - S'(x)]\Big|_{x=a}^{x=b} = 0.$$

Then $\|S''\|_2 \leq \|f''\|_2$.

Remark 17.8. ***Comment on Theorem 17.7***

Theorem 17.7 does not apply for the not-a-knot condition because the assumption does not hold, but it does apply for the other 3.

18 11-2-11

18.1 Proof of Theorem 17.7

Proof.

$$\begin{aligned}
\int_a^b S''(x)[f''(x) - S''(x)]\,dx &\overset{\text{IBP}}{=} \underbrace{S''(f' - S')\big|_a^b}_{=0} - \int_a^b S'''(x)[f'(x) - S'(x)]\,dx \\
&= -\sum_{j=1}^n c_j \int_{x_{j-1}}^{x_j} f'(x) - S'(x)\,dx \qquad\qquad S'''(x) = c_j = \text{ constant on } [x_{j-1}, x_j] \\
&= -\sum_{j=1}^n c_j [f(x) - S(x)]\big|_{x_{j-1}}^{x_j} \\
&= -\sum_{j=1}^n c_j (\cancel{y_j} - \cancel{y_j} - \cancel{y_{j-1}} + \cancel{y_{j+1}}) \\
&= 0
\end{aligned}$$

Next,

$$\begin{aligned}
f''(x) &= S''(x) + [f''(x) - S''(x)] \\
[f''(x)]^2 &= [S''(x)]^2 + 2S''(x)[f''(x) - S''(x)] + [f''(x) - S''(x)]^2 \\
\|f''\|_2^2 &= \underbrace{\int_a^b [S''(x)]^2\,dx}_{=\|S''\|_2} + 2\underbrace{\int_a^b S''(x)[f''(x) - S''(x)]\,dx}_{=0} + \underbrace{\int_a^b [f''(x) - S''(x)]^2\,dx}_{\geq 0} \\
\|f''\| &\geq \|S''\|_2
\end{aligned}$$

□

18.2 Construction of an Interpolating Cubic Spline

Constructing splines boils down to solving a tridiagonal linear system ($\Rightarrow$ you never have to pivot). We will only cover the natural cubic spline; the other endpoint conditions are similar.

Recall:

$$\begin{aligned}
a = x_0 < x_1 < x_2 < \cdots < x_n = b \\
S(x_j) = y_j, \qquad j = 0, 1, \ldots, n \\
S(x) = S_j(x) = \alpha_j + \beta_j(x - x_{j-1}) + \gamma_j(x - x_{j-1})^2 + \delta_j(x - x_{j-1})^3 \qquad \text{on } [x_{j-1}, x_j],\ j = 1, 2, \ldots, n
\end{aligned}$$

Conditions:

1. $S_j(x_{j-1}) = y_{j-1},\ S_j(x_j) = y_j$ for $j = 1, 2, \ldots, n$
2. $S_j'(x_j) = S_{j+1}'(x_j),\ j = 1, 2, \ldots, n-1$
3. $S_j''(x_j) = S_{j+1}''(x_j),\ j = 1, 2, \ldots, n-1$
4. $S_1''(x_0) = 0,\ S_n''(x_n) = 0$ (natural spline condition)

From these conditions, we get:

1. $\alpha_j = y_{j-1}$ and

$$\begin{aligned} y_j - y_{j-1} &= \beta_j(x_j - x_{j-1}) + \gamma_j(x_j - x_{j-1})^2 + \delta_j(x_j - x_{j-1})^3 \\ \Delta_j : &= y_j - y_{j-1}, \qquad \xi_j := x_j - x_{j-1} \qquad \text{(Notation)} \end{aligned} \tag{18.1}$$

2. $\beta_j + 2\gamma_j\xi_j + 3\delta_j\xi_j^2 = \beta_{j+1},\ j = 1, 2, \ldots, n-1$

3. $2\gamma_j + 6\delta_j\xi_j = 2\gamma_{j+1},\ j = 1, 2, \ldots, n$

4. $2\gamma_1 = 0,\ 2\gamma_n + 6\delta_n n\xi_n = 0 \Leftrightarrow$
   $\gamma = 0,\ \gamma_{n+1} = 0$ where $\gamma_{n+1} = \gamma_n + 3\delta_n\xi_n$

3.
$$\delta_j = \frac{\gamma_{j+1} - \gamma_j}{3\xi_j}$$

1.
$$\beta_j = \frac{\Delta_j}{\xi_j} - \gamma_j\xi_j - \frac{\xi_j}{3}(\gamma_{j+1} - \gamma_j) = \frac{\Delta_j}{\xi_j} - \frac{\xi_j}{3}(\gamma_{j+1} + 2\gamma_j)$$

Insert the newest (3) and (1) into (2):

$$\xi_j\gamma_j + 2(\xi_j + \xi_{j+1})\gamma_{j+1} + \xi_{j+2}\gamma_{j+2} = 3\left(\frac{\Delta_{j+1}}{\xi_{j+1}} - \frac{\Delta_j}{\xi_j}\right), \quad j = 1, 2, \ldots, n-1$$

(where $\gamma_1 = 0,\ \gamma_{n+1} = 0$).

$$\begin{bmatrix} 2(\xi_1+\xi_2) & \xi_2 & & & & & \\ \xi_2 & 2(\xi_2+\xi_3) & \xi_3 & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \xi_{n-2} & 2(\xi_{n-2}+\xi_{n-1}) & \xi_{n-1} \\ & & & & & \xi_{n-1} & 2(\xi_{n-1}+\xi_n) \end{bmatrix} \begin{bmatrix} \gamma_2 \\ \gamma_3 \\ \vdots \\ \gamma_n \end{bmatrix} = \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_{n-1} \end{bmatrix}$$

where

$$t_j := 3\left(\frac{\Delta_{j+1}}{\xi_{j+1}} - \frac{\Delta_j}{\xi_j}\right), \qquad j = 1, 2, \ldots, n-1.$$

Remark 18.1.

The coefficient matrix of this system is tridiagonal, symmetric, and strictly diagonally dominant:

$$2(\xi_j + \xi_{j+1}) > \xi_j + \xi_{j+1} \qquad \text{for all } j.$$

$\Rightarrow$ There exists a unique solution $\gamma_2, \gamma_3, \ldots, \gamma_n$.
Furthermore, no pivoting is needed. The system can be solved in $O(n)$ flops.

19 11-4-11

19.1 B-splines

Given: $\Delta = \{x_0, x_1, \dots, x_n\}$, where $a = x_0 < x_1 < \cdots < x_n = b$ and $n \geq 1$ (so that $a < b$). Recall that for the spline S of degree $k-1$ (order k): $S \in C^{k-2}[a,b]$ and $S \in \prod_{k-1}$ on $[x_{j-1}, x_j]$, $j = 1, 2, \dots, n$.

Definition 19.1. $\mathcal{S}_{k,\Delta}$

$$\mathcal{S}_{k,\Delta} = \text{ the set of all splines of order } k,\ k \geq 2.$$

It is easy to see that $\mathcal{S}_{k,\Delta}$ is a real vector space:

$$S, \tilde{S} \in \mathcal{S}_{k,\Delta},\ \alpha \in \mathbb{R} \ \Rightarrow\ S + \tilde{S} \in \mathcal{S}_{k,\Delta},\ \alpha S \in \mathcal{S}_{k,\Delta}.$$

Also, $\prod_{k-1} \subset \mathcal{S}_{k,\Delta}$.

Proposition 19.2. *Dimension of* $\mathcal{S}_{k,\Delta}$

- On $[x_0, x_1]$: k coefficients
- On $[x_1, x_2]$: k coefficients and $k-1$ conditions $\Rightarrow$ i.e. only 1 coefficient
- Similarly for $[x_{j-1}, x_j]$, $j = 1, 2, 3, \dots, n$.

So $\dim(\mathcal{S}_{k,\Delta}) = k + n - 1$.

19.1.1 B-spline Basis (1)

We consider this kind of basis composed of monomials – that is, $p(x) = x^l$, $l = 0, 1, 2, \dots$ – and truncated powers of degree $k-1$:

$$f_{x_j}(x) = (x - x_{j-1})_+^{k-1} := (\max\{x - x_j, 0\})^{k-1} = \begin{cases} (x - x_j)^{k-1} & x \geq x_j \\ 0 & x < x_j \end{cases}$$

$$\frac{d^l}{dx^l}(x - x_j)_+^{k-1} = (k-1)(k-2)\dots(k-1-l)(x - x_j)_+^{k-1-l}$$

So $(x - x_j)_+^{k-1} \in C^{k-2}[a,b]$, $(x - x_j)_+^{k-1} \in \mathcal{S}_{k,\Delta}$, and

$$\frac{d^{k-1}}{dx^{k-1}}(x - x_j)_+^{k-1} = \begin{cases} 0 & x < x_j \\ (k-1)! & x > x_j \end{cases}$$

Theorem 19.3.

The set

$$B = \{1, x, x^2, \dots, x^{k-1}, (x - x_1)_+^{k-1}, (x - x_2)_+^{k-1}, \dots, (x - x_{n-1})_+^{k-1}\}$$

is a basis for $\mathcal{S}_{k,\Delta}$.

Proof. We only need to show that the $k+n-1$ functions in B are linearly independent. Let

$$S(x)=\sum_{l=0}^{k-1}\alpha_l x^l+\sum_{i=1}^{n-1}\beta_i(x-x_i)_+^{k-1}=0$$

for all $a\le x\le b$. For each $j=1,2,\dots,n-1$,

$$\begin{aligned}0&=S^{(k-1)}(x_j+0)-S^{(k-1)}(x_j-0)\\&=\beta_j\left(\frac{d^{k-1}}{dx^{k-1}}(x-x_j)_+^{k-1}\big|_{x=x_j+0}-\frac{d^{k-1}}{dx^{k-1}}(x-x_j)_+^{k-1}\big|_{x=x_j-0}\right)\\&=\beta_j(k-1)!.\end{aligned}$$

Thus, $\beta_j=0$ for $j=1,2,\dots,n-1$, and

$$S(x)=\sum_{l=0}^{k-1}\alpha_l x^l=0$$

for all $a\le x\le b$. So $\alpha_l=0$ for $l=0,1,\dots,k-1$, and thus the functions in B are linearly independent. □

However, the basis B is useless in practice:

- Ill-conditioned
- Global support. $\operatorname{supp} x^l=[a,b]$, $\operatorname{supp}(x-x_j)_+^n-1=[x_j,b]$. It is more efficient to have a basis function with local support only.

20 11-7-11

20.1 Homework Comments

Problem 3:

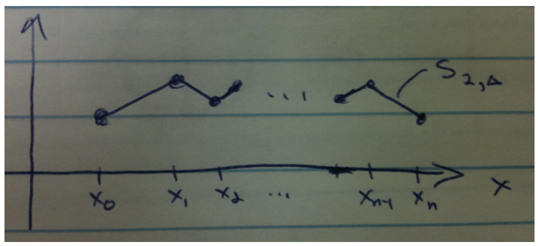
$$\frac{\|x-y\|}{\|x\|}$$

$Ax = b$, $\underline{b \neq 0}$.

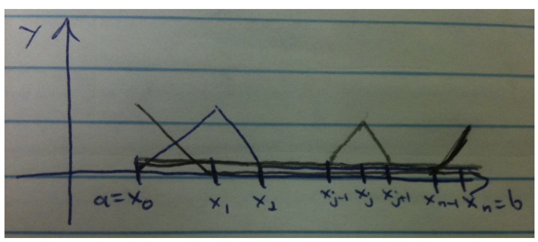
20.2 B-Splines

B-splines are basis functions for $\mathcal{S}_{k,\Delta}$ with desirable properties.

Example 20.1. $k = 2$ ***(Linear Splines)***



B-splines for $\mathcal{S}_{2,\Delta}$ are the <u>hat functions</u>.



$$S(x) = \sum_{i=1}^{n+1} y_{i-1} N_{i2}(x)$$

$$S(x_{i-1}) = y_{i-1}, \quad i = 1, 2, \ldots, n+1$$

<u>Note:</u> the end points x_0 and x_n are special: for $k = 2$, we count x_0 and x_n "twice."

For general $k \geq 2$:

$$\overbrace{\underbrace{x_0}_{\tau_1=\tau_2=\ldots=\tau_k}}^{k \text{ times}} < \underbrace{x_1}_{\tau_{k+1}} < \underbrace{x_2}_{\tau_{k+2}} < \ldots < \underbrace{x_{n-1}}_{\tau_{k+n-1}} < \overbrace{\underbrace{x_n}_{\tau_{k+n}=\tau_{k+n+1}=\ldots=\tau_{2k+n-1}}}^{k \text{ times}}$$

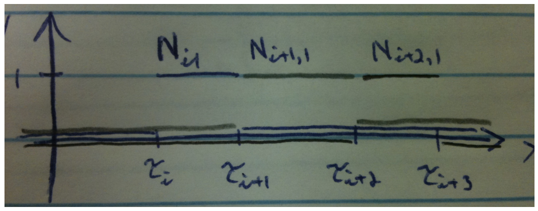
20.3 Construction of B-Splines of Order k (≥ 2)

$$\begin{aligned} \tau_1 \leq \tau_2 \leq \ldots \leq \tau_l, &\qquad \tau_1, \tau_2, \ldots, \tau_l \in \mathbb{R} \\ l = 2k + n - 1 & \\ l - k = k + n - 1 & \\ = \dim \mathcal{S}_{k,\Delta} & \end{aligned} \tag{20.1}$$

Definition 20.2.

The B-splines N_{ik} of order k (≥ 2) (associated with (20.1)), $i = 1, 2\ \ldots, l-k$, are defined recursively as follows:

$$N_{i1}(x) := \begin{cases} 1 & \tau_i \leq x < \tau_{i+1} \\ 0 & \text{otherwise} \end{cases}, \qquad i = 1, 2, \ldots, l-1$$



For $j = 2, 3, \ldots, k$:

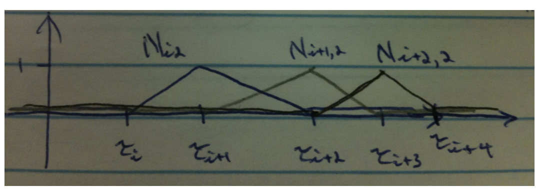
$$N_{ij}(x) := \frac{x - \tau_i}{\tau_{i+j-1} - \tau_i} N_{i,j-1}(x) + \frac{\tau_{i+j} - x}{\tau_{i+j} - \tau_{i+1}} N_{i+1,j-1}(x), \qquad i = 1, 2, \ldots, l-j \tag{20.2}$$

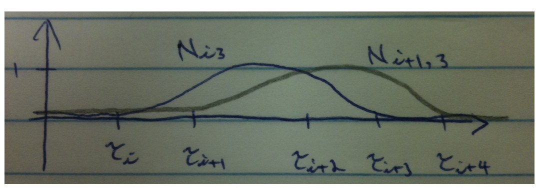
Conventions:

1. $\frac{0}{0} = 0$ if $\tau_{i+j} - \tau_i = 0$ or $\tau_{i+j} - \tau_{i+1} = 0$. For example, if $\tau_i = \tau_{i+1} = \ldots = \tau_{i+k}$.

$$N_{i,j-1}(x) \equiv 0$$
$$N_{i+1,j-1}(x) \equiv 0$$
$$N_{ij}(x) \equiv 0$$

2. $N_{ik}(\tau_l) := \lim\limits_{x \to \tau_l^-} N_{ik}(x)$





Note:

$$N'_{i4}(\tau_i) = N''_{i4}(\tau_i) = 0$$
$$N'_{i4}(\tau_{i+4}) = N''_{i4}(\tau_{i+4}) = 0$$

Properties

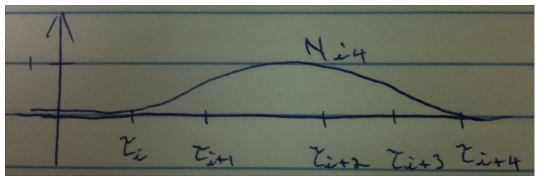
(a) $\operatorname{supp} N_{ik} \subset [\tau_i, \tau_{i+k}]$

(b) $N_{ik}(x) \geq 0$ for all $x \in [\tau_n, \tau_l]$

(c) $\sum_{i=1}^{l-k} N_{ik}(x) = 1$ for all $x \in [\tau_1, \tau_l]$, i.e. the N_{ik}'s form a partition of unity on $[\tau_1, \tau_l]$

(d) $N_{ik} \in C^{k-2}[\tau_1, \tau_l]$

(e) $N_{ik} \in \prod_{k-1}$ on $[\tau_j, \tau_{j+1}]$, $j = 1, 2, \ldots, l-1$



21 11-9-11

21.1 B-Splines (Continued)

$$\tau_1 \le \tau_2 \le \cdots \le \tau_l \tag{21.1}$$

$$\text{For } j = 2, 3, \ldots, k: \qquad N_{ij}(x) = \cdots N_{i,\gamma-1}(x) + \cdots N_{i+1,j-1}(x), \qquad i = 1, 2, \ldots, l-j \tag{21.2}$$

$$\begin{array}{ccccc}
N_{11} \searrow & & & & \\
 & N_{12} \searrow & & & \\
N_{21} \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & N_{13} \searrow & & \\
 & N_{22} \nearrow & & \ddots & \\
N_{31} \nearrow & & \vdots & & N_{1k} \\
 & \vdots & & & \vdots \\
\vdots & \vdots & \vdots & & N_{l-k,k} \\
 & N_{l-3,2} \searrow & & \iddots & \\
N_{l-2,1} \searrow & & N_{l-3,3} \nearrow & & \\
 & N_{l-2,2} \nearrow & & & \\
N_{l-1,1} \nearrow & & & &
\end{array}$$

21.1.1 Efficient Evaluation for $\tau_1 \le x < \tau_l$

Determine j such that $\tau_j \le x < \tau_{j+1}$ $(1 \le j < l,\ j$ is unique).

$$\begin{aligned} N_{j1}(x) &= 1 \\ N_{ji}(x) &= 0 \quad \text{if } i \ne j \end{aligned}$$

$$\begin{array}{ccccc}
 & & & N_{1,j}(x) & \text{(column } j\text{)} \\
 & & \iddots & & \\
 & & N_{j-2,3}(x) \nearrow & & \\
 & N_{j-1,2}(x) \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & \\
1 = N_{j1}(x) \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & N_{j-1,3}(x) & & \\
 & N_{j,2}(x) \begin{smallmatrix}\nearrow\\ \searrow\end{smallmatrix} & & & \\
 & & N_{j,3}(x) \searrow & & \\
 & & & \ddots & \\
 & & & N_{j,l-j}(x) & \text{(column } l-j\text{)}
\end{array}$$

21.1.2 Back to $\mathcal{S}_{k,\Delta}$

Choose the τ_j's (21.1) as follows:

$$\underbrace{a}_{\substack{=\tau_1=\tau_2=\cdots=\tau_k \\ k \text{ times}}} = x_0 < \underbrace{x_1}_{=\tau_{k+1}} < \underbrace{x_2}_{=\tau_{k+2}} < \cdots < \underbrace{x_{n-1}}_{=\tau_{k+n-1}} < x_n = \underbrace{b}_{\substack{=\tau_{k+n}=\tau_{k+n+1}=\cdots=\tau_{n+2k-1} \\ k \text{ times}}}$$

Set $l := n + 2k - 1$ ($\Rightarrow l - k = n + k - 1 = \dim \mathcal{S}_{k,\Delta}$).

Corresponding B-splines of order k:

$$N_{1k}, N_{2k}, \ldots, N_{n+k-1,k}$$

Theorem 21.1.

The B-splines N_{ik}, $i = 1, 2, \ldots, n + k - 1$ form a basis of $\mathcal{S}_{k,\Delta}$.

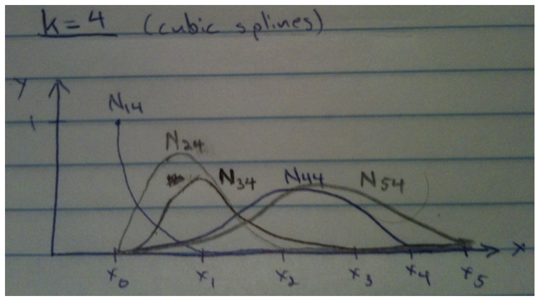


Figure 4: $\operatorname{supp} N_{i4} \subseteq [\tau_i, \tau_{i+4}]$. Also, $\sum_i N_{i4}(x) = 1$.

Notes:

1. The representation
$$S(x) = \sum_{i=1}^{n+k-1} \gamma_i N_{ik}(x)$$
of any $S \in \mathcal{S}_{k,\Delta}$ is well-conditioned.
2. The coefficients γ_i, $i = 1, 2, \ldots, n + k - 1$ are called the *de Boor points* of S.
3. Since $N_{ik}(x) \geq 0$ and $\sum_{i=1}^{n+k-1} N_i(x) = 1$ for all $a \leq x \leq b$, the $S(x) = \sum_{i=1}^{n+k-1} \gamma_i N_{ik}(x)$ is a convex combination of the de Boor points $\gamma_1, \gamma_2, \ldots, \gamma_{l-k}$ for any fixed $x \in [a, b]$.

Example 21.2. ***Cubic Spline with Not-A-Knot Condition***

$k = 4,\ n \geq 3.$

$$\underbrace{x_0}_{=\tau_1=\tau_2=\tau_3=\tau_4} < x_1 < \underbrace{x_2}_{=\tau_5} < \cdots < \underbrace{x_{n-2}}_{=\tau_{n+1}} < x_{n-1} < \underbrace{x_n}_{=\tau_{n+2}=\tau_{n+3}=\tau_{n+4}=\tau_{n+5}}$$

$l = n + 5$. Corresponding B-splines $N_{14}, N_{24}, \ldots, N_{n+1,4}$.

$$S(x) = \sum_{i=1}^{n+1} \gamma_i N_{i4}(x)$$

<u>Interpolating Conditions:</u>

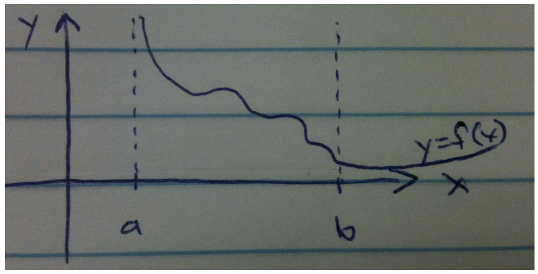
$$S(x_{j-1}) = y_{j-1}, \quad j = 1, \ldots, n+1$$

$$M \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{n+1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

where $M = [m_{ji}]_{j,i=1,2,\ldots,n+1}$ with $m_{ji} = N_{i4}(x_{j-1})$.

22 11-14-11

22.1 Numerical Integration



Problem:
Evaluate

$$I = \int_a^b f(x)\,dx$$

where $f:(a,b)\to\mathbb{R}$ is integrable on $[a,b]$.

Quadrature Rules

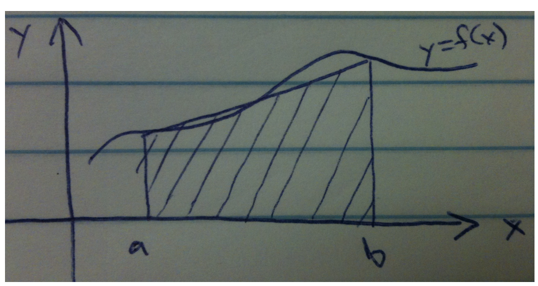
$$I \approx \sum_{i=1}^{n} w_i f(x_i)$$

where $a \le x_1 < x_2 < \cdots < x_n \le b$ and $w_i \in \mathbb{R}$.

$$R_n := \sum_{i=1}^{n} w_i f(x_i) - I = (\underline{\text{remainder}} \text{ or } \underline{\text{error}})$$

22.2 Examples of Quadrature Rules

1. Trapezoidal Rule



$$I \approx (b-a)\frac{f(a)+f(b)}{2} = \frac{b-a}{2}f(a) + \frac{b-a}{2}f(b)$$

$n=2,\ x_1=a,\ x_2=b,\ w_1=w_2=\frac{b-a}{2}$.

$$R_2 = (b-a)^3\frac{1}{12}f^{(2)}(\xi) \qquad \text{for some } \xi\in(a,b)$$

Thus, the trapezoid rule is exact for polynomials of degree ≤ 1.

2. Simpson's Rule

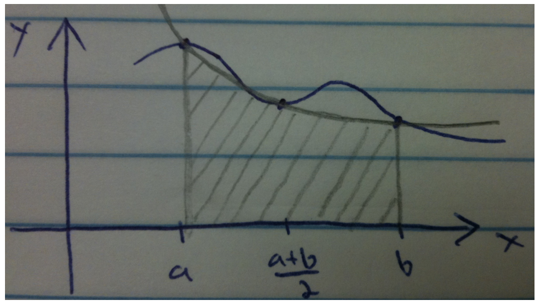


Figure 5: Interpolate the 3 points with a quadratic function.

$$I \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

$n = 3,\ x_1 = a,\ x_2 = \frac{a+b}{2},\ x_3 = b,\ w_1 = w_3 = \frac{b-a}{6},\ w_2 = \frac{2}{3}(b-a).$

$$R_3 := (b-a)^5 \frac{1}{2880} f^{(4)}(\xi) \qquad \text{for some } \xi \in (a,b)$$

Thus, Simpson's rule is exact for polynomials of degree ≤ 3. Where does this extra degree of accuracy come from?

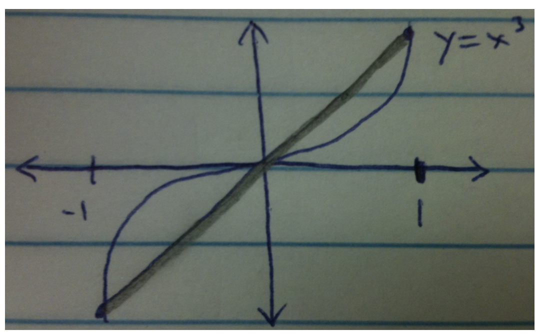
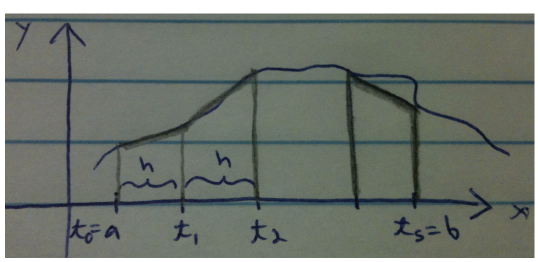


Figure 6: Simpson's rule for $y = x^3$ on $[-1, 1]$ is exact.

3. Compound trapezoidal rule. Let $s \geq 1$ be an integer. Set $t_i = a + ih$, $i = 0, 1, \ldots, s$, where $h := \frac{b-a}{s}$.



$$
\begin{aligned}
I = \int_a^b f(x)\,dx &= \sum_{i=0}^{s-1} \int_{t_i}^{t_{i+1}} f(x)\,dx \\
&\approx \sum_{i=0}^{s-1} \frac{h}{2}[f(t_i) + f(t_{i+1})] \\
&= \frac{h}{2}\left[f(a) + 2f(a+h) + 2f(a+2h) + \cdots + 2f(b-h) + f(b)\right] \\
&= h\left[\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \cdots + f(b-h) + \frac{1}{2}f(b)\right] \\
&=: T_{s+1}
\end{aligned}
$$

where $s+1$ is the number of points used. Error:

$$
\begin{aligned}
R_{s+1} = T_{s+1} - I &= sh^3 \frac{1}{12} f^{(2)}(\xi) \qquad \text{for some } \xi \in (a,b) \\
&= (b-a)\frac{1}{12}h^2 f^{(2)}(\xi)
\end{aligned}
$$

T_{s+1} involves $s+1$ points ($h = \frac{b-a}{s}$).
T_{2s+1} involves $2s+1$ points ($\hat{h} = \frac{b-a}{2s} = \frac{h}{2}$).

$$
\begin{aligned}
T_{2s+1} &= \underbrace{\hat{h}}_{\frac{h}{2}} \left[\frac{1}{2}f(a) + f(a+\hat{h}) + \underbrace{f(a+2\hat{h})}_{f(a+h)} + \cdots + \underbrace{f(b-2\hat{h})}_{f(b-h)} + f(b-\hat{h}) + \frac{1}{2}f(b)\right] \\
&= \frac{1}{2}T_{s+1} + \hat{h}\left[f(a+\hat{h}) + f(a+3\hat{h}) + \cdots f(b-3\hat{h}) + f(b-\hat{h})\right]
\end{aligned}
$$

Consequence: Once we have T_{s+1}, the approximation T_{s+1} can be obtained with s additional function evaluations! We can also estimate the error of T_{2s+1}:

$$
\begin{aligned}
R_{s+1} = T_{s+1} - I &= (b-a)\frac{1}{12}h^2 f^{(2)}(\xi) \\
R_{2s+1} = T_{2s+1} - I &= (b-a)\frac{1}{12}\frac{h^2}{4} f^{(2)}(\hat{\xi})
\end{aligned}
$$

Assume that $f^{(2)}(\xi) \approx f^{(2)}(\hat{\xi})$. Then

$$
R_{2s+1} \approx \frac{1}{4}R_{s+1}.
$$

But:

$$
\begin{aligned}
|T_{2s+1} - T_{s+1}| &= |R_{2s+1} - R_{s+1}| \\
&= \frac{3}{4}|R_{s+1}|
\end{aligned}
$$

So an error estimate for $T_{s+1} \approx |R_{s+1}| \approx \frac{4}{3}|T_{2s+1} - T_{s+1}|$. Approximate integral:

$$
I \approx T_{2s+1}.
$$

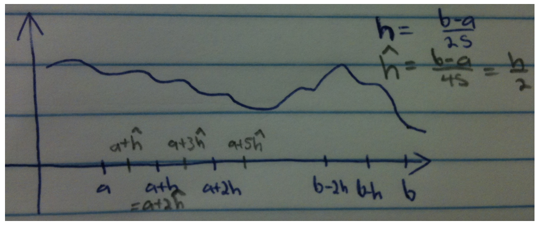
Conservative error estimate:

$$
\frac{4}{3}|T_{2s+1} - T_{s+1}|.
$$

23 11-16-11

23.1 Examples of Quadratures (Continued)

4. Compound Simpson's Rule. $s \geq 1,\ h = \frac{b-a}{2s}$



$$
\begin{aligned}
I &= \int_a^b f(x)\,dx = \sum_{i=0}^{s-1} \int_{a+2ih}^{a+2(i+1)h} f(x)\,dx \\
&\approx \frac{h}{3}\Big[\underbrace{f(a) + 4f(a+h) + f(a+2h)} + \underbrace{f(a+2h) + 4f(a+3h) + f(a+4h)} + \cdots + f(b-2h) \\
&\quad + \underbrace{f(b-2h) + 4f(b-h) + f(b)}\Big] \\
&\approx \frac{h}{3}\left[f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) + 2f(a+4h) + \cdots + 2f(b-2h) + 4f(b-h) + f(b)\right] \\
&= S_{2s+1}
\end{aligned}
$$

This uses $2s+1$ points. Remainder:

$$R_{2s+1} = S_{2s+1} - I = (b-a)\frac{h^4}{180} f^{(4)}(\xi) \qquad \text{for some } \xi \in (a,b)$$

If we double s:

$$R_{4s+1} \approx \frac{1}{16} R_{2s+1}$$

Approximate integral: $I \approx S_{4s+1}$
Conservative error estimate: $\frac{16}{15}|S_{4s+1} - S_{2s+1}|$

<u>Efficient Implementation</u>

$$S_{2s+1} = \frac{h}{3}\left[f(a) + 4\sum_{j=1}^{s} f\big(a + (2j-1)h\big) + f(b) + 2\sum_{k=1}^{s-1}\sum f(a+2kh)\right], \qquad h = \frac{b-a}{2s}$$

$$= \frac{h}{3}\left[\underbrace{f(a) + 2\sum_{i=1}^{2s-1} f(a+ih) + f(b)}_{=:A_{2s+1}} + \underbrace{2\sum_{j=1}^{s} f\big(a+(2j-1)h\big)}_{=:B_s}\right]$$

$$S_{4s+1} = \frac{h}{6}\left[A_{4s+1} + B_{2s}\right]$$

where $$A_{4s+1} = f(a) + 2\sum_{i=1}^{4s-1} f\left(a + i\frac{h}{2}\right) + f(b)$$

and $$B_{2s} = 2\sum_{j=1}^{2s} f\left(a + (2j-1)\frac{h}{2}\right)$$

We get A_{4s+1} for free because

$$
\begin{aligned}
A_{4s+1} &= \underbrace{f(a) + 2\sum_{\substack{k=1 \\ (2k=i)}}^{2s-1} f(a+kh) + f(b)}_{=A_{2s+1}} + 2\sum_{\substack{j=1 \\ (2j-1=i)}}^{2s} f\left(a + (2j-1)\frac{h}{2}\right) \\
&= A_{2s+1} + B_{2s}
\end{aligned}
$$

<u>Consequence:</u> To obtain S_{4s+1} from $S_{2s+1} = \frac{b-a}{6s}(A_{2s+1} + B_s)$, we only need to compute

$$
B_{2s} = 2\sum_{j=1}^{2s} f\big(a + (2j-1)\hat{h}\big), \qquad \hat{h} = \frac{h}{2}
$$

and set

$$
\begin{aligned}
A_{4s+1} &= A_{2s+1} + B_{2s} \\
S_{4s+1} &= \frac{b-a}{12s}(A_{4s+1} + B_{2s})
\end{aligned}
$$

5. Gaussian Integration. So far all the formulas we've seen have been of the form

$$
I = \int_a^b f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{23.1}
$$

This expression has $2n$ degrees of freedom, so we hope to be able to integrate exactly polynomials of degree $\leq 2n-1$. For Gaussian integration, we choose $a < x_1 < x_2 < \cdots < x_n < b$ to be the zeros of the nth *orthogonal polynomial* $p_n(x) = x^n + \cdots \in \prod_n$ defined by

$$
\int_a^b p(x)p_n(x)\,dx = 0 \qquad \text{for all } p \in \prod_n
$$

($a=-1,\ b=1,\ p_n(x) = \gamma_n \cdot (n\text{th Legendre polynomial})$.)
The w_i's are chosen such that (23.1) is exact for all $f \in \prod_{n-1}$, i.e.

$$
\int_a^b f(x)\,dx = \sum_{i=1}^{n} w_i f(x_i), \quad f \in \prod_n .
$$

(For example: $f(x) = x^j,\ j = 0, 1, \ldots, n-1$.)
In this case, we get a linear system:

$$
\begin{bmatrix}
1 & 1 & \cdots & 1 \\
x_1 & x_2 & \cdots & x_n \\
x_1^2 & x_2^2 & \cdots & x_n^2 \\
\vdots & \vdots & \ddots & \vdots \\
x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1}
\end{bmatrix}
\begin{bmatrix}
w_1 \\ w_2 \\ \vdots \\ w_{n-1}
\end{bmatrix}
=
\begin{bmatrix}
b-a \\ \frac{1}{2}(b^2-a^2) \\ \vdots \\ \frac{1}{n}(b^n-a^n)
\end{bmatrix}
$$

24 11-18-11

24.1 Gaussian Integration (Continued)

$$I = \int_a^b f(x)\,dx \approx \sum_{i=1}^n w_i f(x_i) \tag{24.1}$$

Choose the x_i as the zeros of $f(x)$. $a < x_1 < x_2 < \cdots < x_n < b$. The w_i's are chosen so that

$$\int_a^b f(x)\,dx = \sum_{i=1}^n w_i f(x_i)$$

for all $f \in \prod_{n-1}$.

Theorem 24.1.

(24.1) is exact for all $p \in \prod_{2n-1}$.

Proof. Let $p \in \prod_{2n-1}$. $0 = \int_a^b p_n(x)q(x)\,dx$ for all $q \in \prod_{n-1}$. Polynomial division:

$$p(x) = p_n(x)q(x) + r(x) \qquad \text{where } q, r \in \prod_{n-1}$$

$$\begin{aligned}
\int_a^b p(x)\,dx &= \underbrace{\int_a^b p_n(x)q(x)\,dx}_{\substack{=0 \text{ b/c} \\ p_n = n\text{th orthogonal polynomial}}} + \int_a^b r(x)\,dx \\
&= \sum_{i=1}^n w_i r(x_i) \\
&= \sum_{i=1}^n w_i \underbrace{[\overbrace{p_n(x_i)}^{=0} q(x_i) + r(x_i)]}_{=p(x_i)} \\
&= \sum_{i=1}^n w_i p(x_i)
\end{aligned}$$

□

24.1.1 Pros and Cons of Gaussian Integration

Pros

- Optimal Accuracy
- Can be used for functions with singularities at $x = a$ or $x = b$ (since $a < x_1 < x_2 < \cdots < x_n < b$)

Cons

- When n is increased, old values $f(x_1),\ f(x_2),\ \ldots,\ f(x_n)$ cannot be reused

24.1.2 Gauss-Kronrod Rules

Given: Gaussian rule with n points,

$$G_n = \sum_{i=1}^{n} w_i f(x_i).$$

Gauss-Kronrod:

$$K_{2n+1} = \sum_{i=1}^{n} a_i f(x_i) + \sum_{j=1}^{n+1} b_j f(y_j)$$

where the y_j's are chosen as the zeros of the polynomial $q_{n+1}(x) = x^{n+1} + \cdots \in \prod_{n+1}$ satisfying

$$\int_a^b p_n(x)p(x)q_{n+1}(x)\,dx = 0 \qquad \text{for all } p \in \prod_n.$$

Thus, q_{n+1} is the $(n+1)$st orthogonal polynomial with respect to the "inner product" $(p,q) := \int\limits_a^b p(x)q(x)p_n(x)\,dx$.

The remaining parameters $a_i,\ i = 1, 2, \ldots, n$ and $b_j,\ j = 1, 2, \ldots, n+1$ are determined such that

$$\int_a^b p(x)\,dx = \sum_{i=1}^{n} a_i p(x_i) + \sum_{j=1}^{n+1} b_j p(y_j) \qquad \text{for all } p \in \prod_{2n}.$$

Theorem 24.2.

K_{2n+1} is exact for all $p \in \prod_{3n+1}$.

Proof. For $p \in \prod_{3n+1}$:

$$p = (p_n q_{n+1})t + r, \qquad r \in \prod_{2n},\ t \in \prod_n$$

$$\begin{aligned}
\int_a^b p(x)\,dx &= \underbrace{\int_a^b p_n(x)q_{n+1}(x)t(x)\,dx}_{=0} + \int_a^b r(x)\,dx \\
&= \sum_{i=1}^{n} a_i r(x_i) + \sum_{j=1}^{n+1} b_j r(x_i) \\
&= \sum_{i=1}^{n} a_i p(x_i) + \sum_{j=1}^{n+1} b_j p(y_j)
\end{aligned}$$

□

24.1.3 Practical Use

Pair (G_n, K_{2n+1}). Approximate the integral:

$$K_{2n+1} \approx \int_a^b f(x)\,dx.$$

Heuristic error estimate:

$$(200|G_n - K_{2n+1}|)^{3/2}.$$

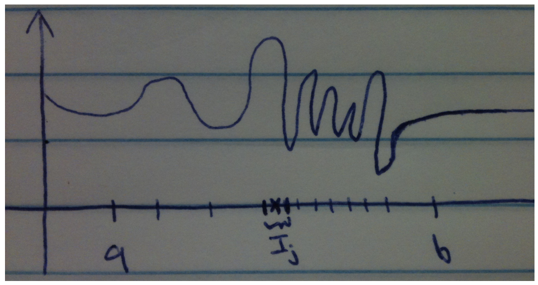
If we start with n reasonably large (e.g. $n = 10$), then K_{2n+1} will probably be accurate enough. However, if we need to double n then we have to start from scratch.

25 11-21-11

25.1 Corrections to the Homework

- 2(d) is trimmed down.
- use underscores for function names, not hyphens

25.2 Adaptive Quadrature



Idea: Use more function values $f(x_j)$ where f "varies more."

Ingredients

- Quadrature rule Q with error estimate E.
- Tolerance $\epsilon > 0$.

Goal: Approximate the integral $Q_{[a,b]}$ such that

$$\left| Q_{[a,b]} - \int_a^b f(x)\,dx \right| \overset{<}{\approx} E_{[a,b]} < \epsilon.$$

25.2.1 Basic Adaptive Procedure

At every state:

$$[a,b] = I_1 \cup I_2 \cup \cdots \cup I_l$$

where $I_j = [a_j, b_j],\ h_j = b_j - a_j$. There is no overlap $\Rightarrow\ h_1 + h_2 + \cdots + h_l = b - a$.

$$Q_j \approx \int_{I_j} f(x)\,dx, \qquad \left| Q_j - \int_{I_j} f(x)\,dx \right| \overset{<}{\approx} E_j$$

Initialization

- $l = 1,\ I_1 = [a,b],\ h_1 = b - a$. Apply your quadrature rule Q to $I_1\ \to\ Q_1, E_1$.

General Step:

- If $E_j < \frac{h_j}{b-a}\epsilon$ for all $j = 1, 2, \ldots, l$, stop. $Q_{[a,b]} = Q_1 + Q_2 + \cdots + Q_l$, with error estimate

$$E_{[a,b]} = E_1 + E_2 + \cdots + E_l < \frac{\epsilon}{b-a} \sum_{j=1}^{l} h_j = \epsilon$$

- Otherwise, for all $I_j = [a_j, b_j]$ with $E_j \geq \frac{h_j}{b-a}\epsilon$, set

$$I_j := \left[a_j, \frac{a_j + b_j}{2}\right], \quad I_{l+1} := \left[\frac{a_j + b_j}{2}, b_j\right]$$
$$h_j := h_{l+1} = \frac{b_j - a_j}{2}, \quad l := l + 1$$

and apply Q to I_j and $I_l \;\rightarrow\; Q_j, E_j$ and Q_l, E_l.

Note: For an actual algorithm, various safeguards are needed, e.g. $h_j \geq h_{\min} > 0$.

25.3 Eigenvalue Problems

Definition 25.1. ***Eigenvalue, Eigenvector***

Let $A \in \mathbb{C}^{n \times n}$. A number $\lambda \in \mathbb{C}$ is called an *eigenvalue* of A if there exists a nonzero $x \in \mathbb{C}^n$, $x \neq \mathbf{0}$, such that

$$Ax = \lambda x.$$

Such an x is called an *eigenvector* of A.

Definition 25.2. ***Spectrum***

The set

$$\Lambda(A) := \left\{\lambda \in \mathbb{C} \;\middle|\; \lambda \text{ is an eigenvalue of } A\right\}$$

is called the *spectrum* of A.

$$Ax = \lambda x, \qquad x \neq \mathbf{0}$$
$$\Rightarrow \quad (\lambda I - A)x = \mathbf{0}, \qquad x \neq \mathbf{0}$$
$$\Rightarrow \qquad \text{the matrix } \lambda I - A \text{ is singular}$$
$$\Rightarrow \det(\lambda I - A) = 0$$

Definition 25.3. ***Characterisitic Polynomial***

The polynomial

$$p(z) = p_A(z) := \det(zI - A) = z_n + \alpha_{n-1}z^{n-1} + \cdots + \alpha_1 z + \alpha_0$$

is called the *characterisitic polynomial* of A.

Notes:

1. The eigenvalues of A are the zeros of the characterisitic polynomial, p_A. In particular, $A \in \mathbb{C}^{n \times n}$ has n eigenvalues, but they are not necessarily distinct.

2. For $n \geq 5$, any algorithm for computing eigenvalues of general $A \in \mathbb{C}^{n \times n}$ has to be iterative!

    - Even if this was not true, working with polynomials is not great.

26 11-23-11

26.1 Eigenvalue Problems (Continued)

3. Any polynomial $p(z) = z^n + \alpha_{n-1}z_{n-1} + \alpha_{n-2}z^{n-2} + \cdots + \alpha_1 z + \alpha_0$ is the characteristic polynomial of a matrix $A \in \mathbb{C}^{n\times n}$. For example,

$$A = \begin{bmatrix} 0 & 0 & \cdots & 0 & -\alpha_0 \\ 1 & \ddots & \ddots & \vdots & -\alpha_1 \\ 0 & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & \ddots & 0 & -\alpha_{n-2} \\ 0 & \cdots & 0 & 1 & -\alpha_{n-1} \end{bmatrix}$$

has the characteristic polynomial

$$\det(zI - A) = \det \begin{bmatrix} z & & & 0 & \alpha_0 \\ -1 & z & & & \alpha_1 \\ & -1 & \ddots & & \vdots \\ & & \ddots & z & \alpha_{n-2} \\ 0 & & & -1 & z+\alpha_{n-1} \end{bmatrix} = z^n + \alpha_{n-1}^{n-1} + \cdots + \alpha_1 z + \alpha_0.$$

For some classes of matrices, the eigenvalues can be found trivially. For example, an upper triangular matrix:

$$A = \begin{bmatrix} a_{11} & * & \cdots & * \\ & a_{22} & \ddots & \vdots \\ & & \ddots & * \\ 0 & & & a_{nn} \end{bmatrix}, \qquad \Lambda(A) = \{a_{11}, a_{22}, \ldots, a_{nn}\}$$

26.2 Computation of Eigenvalues

26.2.1 Bad Ideas

1. Forming p_A and computing $\lambda \in \Lambda(A)$ as zeros of p_A.

2. Attempting to compute the *Jordan canonical form* of A:

$$X^{-1}AX = \begin{bmatrix} J_1 & & & 0 \\ & J_2 & & \\ & & \ddots & \\ 0 & & & J_l \end{bmatrix},$$

where each

$$J_i = \begin{bmatrix} \lambda_i & 1 & & & 0 \\ & \lambda_i & 1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 1 \\ 0 & & & & \lambda_i \end{bmatrix}, \quad i = 1, 2, \ldots, l$$

are *Jordan blocks*.

```
[J,X] = jordan(A)
```

26.2.2 Better Ideas

1. Use the *Schur factorization* of A.
2. Use unitary *similarity transformations* to first transform A to a "simpler form.

Definition 26.1. ***Similarity Transformations***

Let $A \in \mathbb{C}^{n \times n}$. Let $U \in \mathbb{C}^{n \times n}$ be nonsingular. The map

$$A \mapsto U^{-1}AU$$

is called a *similarity transformation* of A.

Lemma 26.2.

$$\Lambda(A) = \Lambda(U^{-1}AU)$$

Proof.

$$\begin{aligned} Ax &= \lambda x, \qquad x \neq \mathbf{0} \\ U^{-1}AU(\underbrace{U^{-1}x}_{\tilde{x}}) &= \lambda(\underbrace{U^{-1}x}_{\tilde{x}}), \qquad \tilde{x} \neq 0 \\ U^{-1}AU\tilde{x} &= \lambda \tilde{x}, \qquad \tilde{x} \neq 0 \end{aligned}$$

$\square$

Use matrices U that are easy to invert and that are numerically well-behaved:

Definition 26.3. ***Unitary***

$U \in \mathbb{C}^{n \times n}$ is said to be *unitary* if

$$U^H U = I$$

$(U = [u_{jk}],\ U^H = [\overline{u_{kj}}] = (\overline{U})^T.)$

Notes:

1. U unitary $\Rightarrow U$ is nonsingular and $U^{-1} = U^H$.
2. U unitary $\Rightarrow U^H$ is unitary and $(U^H)^{-1} = (U^H)^H = U$.
3. U unitary $\Rightarrow \|Ux\|_2 = \|x\|_2$ for all $x \in \mathbb{C}^{n \times n}$. This is because $\|Ux\|_2^2 = (Ux)^H Ux = x^H U^H U x = x^H x = \|x\|_2^2$ and $\|U\|_2 = \max\limits_{x \neq \mathbf{0}} \frac{\|Ux\|_2}{\|x\|_2} = 1$.

4. Unitary matrices U have the best possible Euclidean condition numbers:

$$\kappa_2(U) = \underbrace{\|U\|_2}_{=1} \underbrace{\| \overbrace{U^{-1}}^{=U^H} \|_2}_{=1} = 1.$$

26.3 Unitary Similarity Transformations

$$A \mapsto U^{-1}AU = U^H AU =: T$$

where U is unitary.

Question: What is the "simplest" T we can achieve?

Theorem 26.4. ***Schur***

For any $A \in \mathbb{C}^{n \times n}$, there exists a unitary $U \in \mathbb{C}^{n \times n}$ such that

$$U^H AU = T = \begin{bmatrix} t_{11} & * & \cdots & * \\ & t_{22} & \ddots & \vdots \\ & & \ddots & * \\ 0 & & & t_{nn} \end{bmatrix}$$

is upper-triangular. In particular, $\Lambda(A) = \Lambda(T) = \{t_{11}, t_{22}, \ldots, t_{nn}\}$.

We will prove this by induction on n. For $n = 1$: $A = [a],\ U = [1],\ T = [a]$.

27 11-28-11

Office hours Wednesday 12:30-2:30.

27.1 Proof of Schur's Theorem

Recall:

$$U^H A U = T = \text{upper-triangular}$$

Proof. By induction. For $n = 1$, it is trivial. Assume it is true for $1, \ldots, n-1$. Let $\lambda \in \Lambda(A)$ with eigenvector x, $\|x\|_2 = 1$. Choose a unitary matrix

$$U_1 = \begin{bmatrix} x & * & * & \cdots & * \end{bmatrix} \in \mathbb{C}^{n \times n}$$

with x as the first column. Then:

$$\begin{aligned} AU_1 &= \begin{bmatrix} \lambda x & * & * & \cdots & * \end{bmatrix} \\ U_1^H A U_1 &= \begin{bmatrix} \lambda & * & \cdots & * \\ 0 & & & \\ \vdots & & \tilde{A} & \\ 0 & & & \end{bmatrix}, \qquad \text{where } \tilde{A} \in \mathbb{C}^{n-1 \times n-1} \end{aligned}$$

Induction hypothesis:

$$\tilde{U}^H \tilde{A} \tilde{U} = \tilde{T} = \begin{bmatrix} * & * & \cdots & * \\ & * & \cdots & * \\ & & \ddots & \vdots \\ 0 & & & * \end{bmatrix}, \qquad \tilde{U} \in \mathbb{C}^{n-1 \times n-1} \text{ is unitary}$$

Set

$$U := U_1 \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \tilde{U} & \\ 0 & & & \end{bmatrix} \in \mathbb{C}^{n \times n}$$

Then U is unitary and

$$\begin{aligned} U^H A U &= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \tilde{U}^H & \\ 0 & & & \end{bmatrix} U_1^H A U_1 \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \tilde{U} & \\ 0 & & & \end{bmatrix} \\ &= \begin{bmatrix} \lambda & * & \cdots & * \\ 0 & & & \\ \vdots & & \tilde{U}^H \tilde{A} \tilde{U} & \\ 0 & & & \end{bmatrix} =: T = \text{upper-triangular} \end{aligned}$$

$\square$

27.2 Two Simple Unitary Matrices: Householder Reflectors

Definition 27.1. ***Householder Reflectors***

$Q \in \mathbb{C}^{n\times n}$ of the form $Q = I - 2vv^H$, where $v \in \mathbb{C}^n$ with $\|v\|_2 = 1$, is a *Householder reflector* matrix.
Q is *Hermitian*:

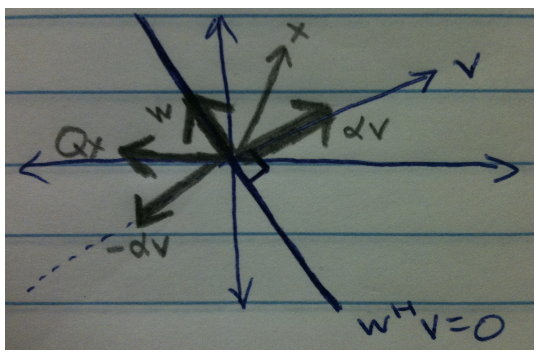
$$Q^H = I - 2(vv^H)^H = Q.$$

Q is unitary:

$$Q^H Q = Q^2 = I - 4vv^H + 4v \underbrace{v^H v}_{=\|v\|_2^2=1} v^H = I.$$

For any $x \in \mathbb{C}^n$:

$$Qx = \text{reflection of } x \text{ w.r.t. the } (n-1)\text{-dimensional plane } \{w \in \mathbb{C}^n \mid w^H v = 0\}$$



Proof.

$$\begin{aligned}
x &= \alpha v + w, \qquad \text{where } \alpha \in \mathbb{C},\ w^H v = 0 \\
Qx &= (I - 2vv^H)(\alpha v + w) \\
&= \alpha v + w - wv \underbrace{v^H \alpha v}_{=\alpha v^H H = \alpha} - 2v \underbrace{v^H w}_{=0} \\
&= -\alpha v + w
\end{aligned}$$

$\square$

Given: $x \in \mathbb{C}^n$.
We can construct a Householder reflector $Q = I - 2vv^H$ such that

$$\begin{aligned}
Qx &= \gamma e_1 \qquad \text{for some } \gamma \in \mathbb{C} \text{ with } |\gamma| = \|x\|_2 \\
x &= \alpha v + w, \qquad w^H v = 0 \\
Qx &= -\alpha v + w = \gamma e_1 \\
x - \gamma e_1 &= \alpha v + w - \gamma e_1 = 2\alpha v
\end{aligned}$$

Set

$$\tilde{v} := x - \gamma e_1 = \begin{bmatrix} x_1 - \gamma \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{and} \quad v := \frac{\tilde{v}}{\|\tilde{v}\|_2}$$

Numerically best choice of γ:

$$\gamma := -(\operatorname{sgn} x_1)\|x\|_2$$

where

$$\operatorname{sgn} x_1 := \begin{cases} \frac{x_1}{\|x_1\|} & x \neq 0 \\ 1 & x_1 = 0 \end{cases}.$$

Summary:

$$\begin{aligned} Q &= I - 2vv^H, \qquad \text{where } v = \frac{\tilde{v}}{\|\tilde{v}\|_2} \\ \tilde{v} &= x + (\operatorname{sgn} x_1)\|x\|_2 e_1 \\ &= \begin{bmatrix} x_1 + (\operatorname{sgn} x_1)\|x\|_2 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \\ Qx &= \gamma e_1, \qquad \text{where } \gamma = -(\operatorname{sgn} x_1)\|x\|_2 \end{aligned}$$

27.2.1 Reduction of A to Hessenberg Form

Definition 27.2. ***Hessenberg Matrix***
http://en.wikipedia.org/wiki/Hessenberg_matrix

An *upper Hessenberg matrix* has zero entries below the first subdiagonal, and a *lower Hessenberg matrix* has zero entries above the first superdiagonal. For example:

$$\begin{bmatrix} 1 & 4 & 2 & 3 \\ 3 & 4 & 1 & 7 \\ 0 & 2 & 3 & 4 \\ 0 & 0 & 1 & 3 \end{bmatrix}$$

is upper Hessenberg.

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 5 & 2 & 3 & 0 \\ 3 & 4 & 3 & 7 \\ 5 & 6 & 1 & 1 \end{bmatrix}$$

is lower Hessenberg.

Schur factorization:

$$U^H A U = T = \begin{bmatrix} * & \cdots & * \\ & \ddots & \vdots \\ 0 & & * \end{bmatrix}$$

Next "best" thing:

$$U^H A U = H = \begin{bmatrix} * & * & \cdots & \cdots & * \\ * & * & \ddots & & \vdots \\ & * & \ddots & \ddots & \vdots \\ & & \ddots & \ddots & * \\ 0 & & & * & * \end{bmatrix}$$

Can be achieved with $n-2$ Householder reflections

$$U = U_1 U_2 \ldots U_{n-2}, \qquad \text{where } U_j = \begin{bmatrix} I_j & 0 \\ 0 & Q_{n-j} \end{bmatrix}, \quad Q_{n-j} \in \mathbb{C}^{n-j \times n-j}, \text{ Householder reflector}$$

Example 27.3. $n = 5$

$$A = \begin{bmatrix} a_{11} & * \\ * & * \end{bmatrix}$$

where $x \in \mathbb{C}^4,\ Q_1 \in \mathbb{C}^{4\times 4},\ Q_1 x = \gamma_1 e_1,\ U_1 = \begin{bmatrix} 1 & 0 \\ 0 & Q_1 \end{bmatrix} = U_1^H.$

$$\begin{aligned}
U_1^H A U_1 = U_1 A U_1 = (U_1 A) U_1 &= \begin{bmatrix} a_{11} & * \\ \gamma_1 e_1 & * \end{bmatrix} U_1 \\
&= \begin{bmatrix} a_{11} & \# & \# & \# & \# \\ \gamma_1 & \# & \# & \# & \# \\ 0 & \# & \# & \# & \# \\ 0 & \# & \# & \# & \# \\ 0 & \# & \# & \# & \# \end{bmatrix}
\end{aligned}$$

$$U_3^H U_2^H U_1^H A U_1 U_2 U_3 = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

28 11-30-11

28.1 Reduction of A to Hessenberg Form (Continued)

$$U^H A U = H, \qquad U = U_1 U_2 \dots U_{n-2}, \; A \in \mathbb{C}^{n \times n}$$

Notes

1. $\Lambda(A) = \Lambda(H)$. Eigenvectors transform as follows:

$$Ax = \lambda x, \; x \neq 0 \;\Leftrightarrow\; \underbrace{U^H A U}_{H} U^H x = \lambda U^H x$$

$$y = U^H x \;\Leftrightarrow\; Hy = \lambda y, \; y \neq 0$$

   Here $y = U^H x$ and $x = Uy = U_1 U_2 \dots U_{n-2} y$,

$$U_j = \begin{bmatrix} I_j & 0 \\ 0 & Q_{n-j} \end{bmatrix}, \quad Q_{n-j} = I - 2 v_{n-j} v_{n-j}^H, \; v_{n-j} \in \mathbb{C}^{n-j}$$

2. Reduction of $A \in \mathbb{C}^{n \times n}$ to upper-Hessenberg form requires $\sim \frac{10}{3} n^3$ flops.

Left to do: Compute the eigenvalues of H.

28.2 The QR Algorithm

Definition 28.1. ***QR Factorization***

QR factorization of $M \in \mathbb{C}^{n \times n}$:

$$M = QR, \quad \text{where } Q \in \mathbb{C}^{n \times n} \text{ is unitary and } R \text{ is upper-triangular}$$

This is done via Gram-Schmidt.

Remark 28.2. ***QR Algorithm (for eigenvalue computations)***

Input: upper-Hessenberg matrix $H \in \mathbb{C}^{n \times n}$

- Set $A^{(0)} := H$
- For $k = 1, 2, \dots$
  - Choose a suitable "shift" $\mu_k \in \mathbb{C}$
  - Compute a QR factorization:

$$A^{(k-1)} - \mu_k I = Q^{(k)} R^{(k)}$$

  - Set $A^{(k)} := R^{(k)} Q^{(k)} + \mu_k I$
- end (k)

Lemma 28.3.

1. $A^{(k)} = (Q^{(k)})^H A^{(k-1)} Q^{(k)},\ k = 1, 2, \ldots$
2. $\Lambda(H) = \Lambda(A^{(k)}),\ k = 0, 1, 2, \ldots$

Proof.

2. By (1), $A^{(k)}, A^{(k-1)}, A^{(k-2)}, \ldots, A^{(1)}, A^{(0)} = H$ are all similar.

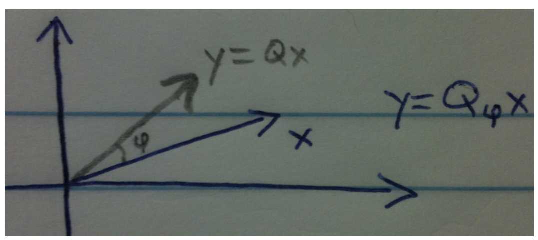
1. $$A^{(k)} = R^{(k)}Q^{(k)} + \mu_k I = (Q^{(k)})^H \underbrace{(Q^{(k)}R^{(k)} + \mu_k I)}_{=A^{(k-1)}} Q^{(k)}$$

□

Note: All the matrices $A^{(k)}$ are upper-Hessenberg!

28.3 Two Simple Unitary Matrices: Givens Rotations

28.3.1 2×2 Case



$$Q = \begin{bmatrix} c & s \\ -\bar{s} & \bar{c} \end{bmatrix} \in \mathbb{C}^{2\times 2}, \quad \text{where } |c|^2 + |s|^2 = 1$$

Q is unitary:

$$Q^H Q = \begin{bmatrix} \bar{c} & -s \\ \bar{s} & c \end{bmatrix} \begin{bmatrix} c & s \\ -\bar{s} & \bar{c} \end{bmatrix} = \begin{bmatrix} |c|^2 + |s|^2 & 0 \\ 0 & |c|^2 + |s|^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Householder:

$$Q \begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix} = \begin{bmatrix} * \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Givens:

$$Q \begin{bmatrix} * \\ * \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$

Use of Q: to "zero out" the entry x_2 of any given $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{C}^2,\ x \neq 0$.

Indeed: Set

$$c = e^{i\alpha} \frac{\overline{x_1}}{\|x\|_2}, \quad s = e^{i\alpha} \frac{\overline{x_2}}{\|x\|_2}, \qquad \text{where } \alpha \in \mathbb{C}$$

Then

$$|c|^2 + |s|^2 = \frac{|x_1|^2 + |x_2|^2}{\|x\|_2^2} = 1$$

$$Q\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \frac{1}{\|x\|_2}\begin{bmatrix} e^{i\alpha}\overline{x_1} & e^{i\alpha}\overline{x_2} \\ -e^{-i\alpha}x_2 & e^{-i\alpha}x_1 \end{bmatrix}\begin{bmatrix} x_2 \\ x_2 \end{bmatrix} = \begin{bmatrix} e^{i\alpha}\|x\|_2 \\ 0 \end{bmatrix}$$

28.3.2 General Case

$$Q = \begin{bmatrix} 1 & & & & & & & & & 0 \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & c & & & s & & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & & -\overline{s} & & & \overline{c} & & & \\ & & & & & & & 1 & & \\ & & & & & & & & \ddots & \\ 0 & & & & & & & & & 1 \end{bmatrix}$$

Notes:

1. Left-multiplication with Q only changes the jth and kth rows.
2. Right-multiplication with Q only changes the jth and kth columns.
3. Q is a Givens rotation $\Rightarrow Q^H$ is a Givens rotation:

$$Q^H = \begin{bmatrix} \overline{c} & -s \\ \overline{s} & c \end{bmatrix}$$

28.3.3 Use in QR Algorithm

Example 28.4. $n = 5$

$$A^{(k-1)} - \mu_k I = \begin{bmatrix} * & * & * & * & * \\ * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

$$\text{Choose } Q_1 = \begin{bmatrix} c_1 & s_1 & & & \\ -\overline{s_1} & \overline{c_1} & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \text{ such that } Q_1^H(A^{(k-1)} - \mu_k^T) = \begin{bmatrix} \# & \# & \# & \# & \# \\ 0 & \# & \# & \# & \# \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

$$\text{Choose } Q_2 = \begin{bmatrix} 1 & & & & \\ & c_1 & s_1 & & \\ & -\overline{s_1} & \overline{c_1} & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix} \text{ such that } Q_2^H Q_1^H(A^{(k-1)} - \mu_k I) = \begin{bmatrix} \# & \# & \# & \# & \# \\ 0 & \# & \# & \# & \# \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

$$\vdots$$

$$Q_4^H Q_3^H Q_2^H Q_1^H(A^{(k-1)} - \mu_k I) = \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & \# & \# \\ 0 & 0 & 0 & 0 & \# \end{bmatrix} =: R^{(k)}$$

29 12-2-11

29.1 Comments on the Final

Friday 6-8 p.m.

- Office hours: Monday 12:30-2:30 p.m. and Wednesday 2-4 p.m.
- Open book & notes

29.2 QR Factorization (Continued)

For general $n \geq 2$
Convert A to a Hessenberg matrix and feed this to the QR algorithm: $A \to H = A^{(0)}$.

$$A^{(k-1)} - \mu_k I = Q^{(k)} R^{(k)} \qquad \text{where } Q^{(k)} = \underbrace{Q_1 Q_2 \dots Q_{n-1}}_{n-1 \text{ Givens rotations}}$$

$$A^{(k)} = R^{(k)} Q^{(k)} + \mu_k I \qquad \text{is upper-Hessenberg}$$

29.3 Convergence of the QR Algorithm

Let $\Lambda(A) = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$ and assume that $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. (Special case: $\mu_k = 0$ for all k.) Then

$$A^{(k)} = \begin{bmatrix} * & * & \cdots & \cdots & * \\ a_{21}^{(k)} & * & \cdots & \cdots & * \\ & a_{32}^{(k)} & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ 0 & & & a_{n,n-1}^{(k)} & * \end{bmatrix} \xrightarrow{k\to\infty} R = \begin{bmatrix} r_{11} & * & \cdots & \cdots & * \\ 0 & r_{22} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & * \\ 0 & \cdots & \cdots & 0 & r_{nn} \end{bmatrix}$$

where $\Lambda(R) = \{r_{11}, r_{22}, \dots, r_{nn}\} = \Lambda(A)$.

Speed of convergence: If $|\lambda_n| > 0$, then

$$a_{j,j-1}^{(k)} = O\left(\left|\frac{\lambda_0}{\lambda_{j-1}}\right|^k\right), \quad j = 2, 3, \dots, \quad k = 1, 2, \dots$$

$\Rightarrow$ slow convergence if $|\lambda_j| \overset{<}{\approx} |\lambda_{j-1}|$. Shifts μ_k are used to speed up convergence. Suppose $\mu_k = \mu =$ constant and $|\lambda_1 - \mu| > |\lambda_2 - \mu| > \cdots > |\lambda_n - \mu| > 0$. Speed of convergence:

$$a_{j,j-1}^{(k)} = O\left(\left|\frac{\lambda_j - \mu}{\lambda_{j-1} - \mu}\right|^k\right), \quad j = 2, 3, \dots, n, \quad k = 1, 2, \dots$$

$\Rightarrow$ $a_{j,j-1}^{(k)} \xrightarrow{k\to\infty} 0$ fast if $\mu \approx \lambda_j$.

29.4 Strategy for Choosing μ_k

At the beginning of the kth iteration of the QR algorithm,

$$A^{(k-1)} = [a_{ij}^{(k-1)}] = \begin{bmatrix} * & \cdots & \cdots & \cdots & * \\ a_{21}^{(k)} & * & & & \vdots \\ 0 & \ddots & \ddots & & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n,n-1}^{(k)} & * \end{bmatrix}, \quad j = 2, 3, \dots, n, \quad \text{check if } a_{j,j-1}^{(k-1)} \approx 0.$$

$a^{(k)}_{j-1,j-1}$
$a^{(k-1)}_{j,j-1}$ $a^{(k)}_{jj}$

$$\left|a^{k-1}_{j,j-1}\right| \leq \epsilon \left(\left|a^{(k-1)}_{j-1,j-1}\right| + \left|a^{(k-1)}_{jj}\right|\right), \quad \text{where } \epsilon = O(e_{ps})$$

Set $a^{(k-1)}_{j,j-1} = 0$.

29.4.1 3 Cases

1. Case 1: $j = n$

$$A^{(k-1)} = \left[\begin{array}{ccccc|c} * & * & \cdots & \cdots & * & * \\ * & \ddots & \ddots & & \vdots & \vdots \\ & \ddots & \ddots & \ddots & \vdots & \vdots \\ & & \ddots & \ddots & * & \vdots \\ & & & * & * & * \\ \hline 0 & \cdots & \cdots & 0 & 0 & a^{(k-1)}_{nn} \end{array}\right] = \left[\begin{array}{c|c} \tilde{A}^{(k-1)} & * \\ \hline 0 & a^{(k-1)}_{nn} \end{array}\right], \quad \tilde{A}^{(k-1)} \in \mathbb{C}^{n-1\times n-1}$$

$\Rightarrow\ a^{(k)}_{nn} \in \Lambda(A^{(k-1)} = \Lambda(A)$.
Accept $a^{(k)}_{nn}$ as an eigenvalue of A and continue the QR algorithm on $\tilde{A}^{(k-1)}$. Typical choice: $\mu_k = a^{(k-1)}_{n-1,n-1}$.

2. Case 2: $j = 2$

$$A^{(k-1)} = \left[\begin{array}{c|cccc} a^{(k-1)}_{11} & * & \cdots & \cdots & * \\ \hline 0 & & & & \\ 0 & & & & \\ \vdots & & \tilde{A}^{(k-1)} & & \\ 0 & & & & \end{array}\right]$$

$a^{(k-1)}_{11}$ is an eigenvalue of A.

3. Case 3: $2 < j < n$

$$A^{(k-1)} = \left[\begin{array}{ccccc|ccccc} * & * & \cdots & \cdots & * & * & \cdots & \cdots & \cdots & * \\ * & \ddots & \ddots & & \vdots & \vdots & & & & \vdots \\ & \ddots & \ddots & \ddots & \vdots & \vdots & & & & \vdots \\ & & \ddots & \ddots & * & \vdots & & & & \vdots \\ & & & * & * & * & \cdots & \cdots & \cdots & * \\ \hline & & & & 0 & * & * & \cdots & \cdots & * \\ & & & & & * & \ddots & \ddots & & \vdots \\ & & 0 & & & & \ddots & \ddots & \ddots & \vdots \\ & & & & & & & \ddots & \ddots & * \\ & & & & & 0 & & & * & * \end{array}\right] = \left[\begin{array}{c|c} A^{(k-1)}_1 & * \\ \hline 0 & A^{(k-1)}_2 \end{array}\right]$$

$\Lambda(A) = \Lambda(A^{(k-1)}) = \Lambda(A^{(k-1)}_1) \cup \Lambda(A^{(k-1)}_2)$. Continue QR on $A^{(k-1)}_1 \in \mathbb{C}^{j-1\times j-1}$ and $A^{(k-1)}_2 \in \mathbb{C}^{n-j+1\times n-j+1}$.

Typical flop count for such a <u>practical</u> QR algorithm:

- $\sim 10n^3$ if only the eigenvalues are computed
- $\sim 27n^3$ if eigenvalues and eigenvectors are computed

A Algorithms

Algorithm	Operation Count	Page
Triangular Solve	n^2	
LU Factorization without pivoting	$\frac{2n^3}{3}$	8
LU Factorization with partial pivoting		14
Newton's Method ($n = 1$)		17
Newton's Method		19
QR Algorithm	$10n^3$ ($27n^3$ with eigenvectors)	85
$A \in \mathbb{C}^{n \times n} \to$ upper-Hessenberg	$\frac{10n^3}{3}$	

Index