

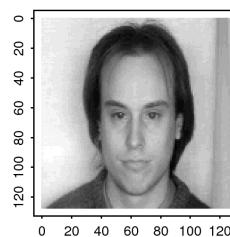
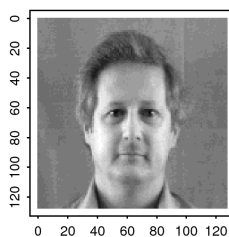
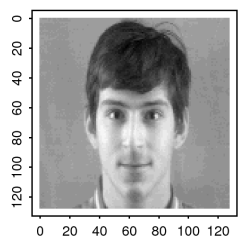
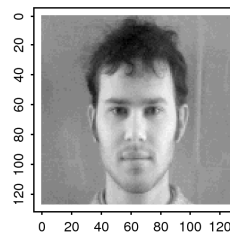
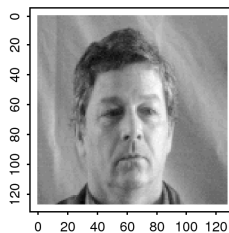
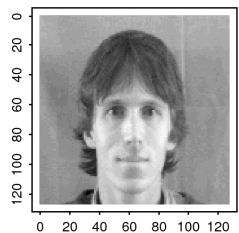
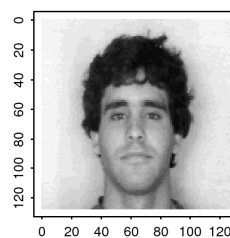
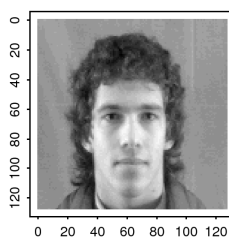
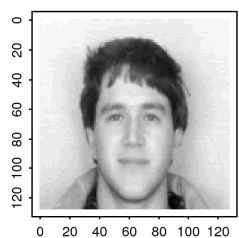
Continuation of Examples

LECTURE 02

Example 2: Face Image Database

- Provided by Prof. L. Sirovich (Mt Sinai S.M.)
- Consists of 143 faces of male caucasian students (and some faculty) at Brown Univ, without glasses, mustache, beard
- Each face is a gray-scale image with 128×128 pixels
- Horizontal dilation was applied so that the pupils are placed on two fixed points.

Below, 9 out of 143 are displayed:



Note that each image of size 128×128 pixels can be represented as a matrix of 128 rows and 128 columns or a vector of length $128 \times 128 = 16384$. In fact, in Julia

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,128} \\ \vdots & \vdots & & \vdots \\ x_{128,1} & x_{128,2} & \dots & x_{128,128} \end{bmatrix} \in \mathbb{R}^{128 \times 128}$$

then $X[:,]$ represents a vector of length 16384 as follows:

$$X[:,] \leftrightarrow \begin{bmatrix} x_{1,1} \\ x_{128,1} \\ x_{1,2} \\ x_{128,2} \\ \vdots \\ x_{1,128} \\ \vdots \\ x_{128,128} \end{bmatrix}$$

By default, a vector is a column vector

Now, let's pick one face image, and consider its representations using two different bases:

- 1) The standard basis $\{e_1, \dots, e_n\}$
- 2) The **wavelet** basis $\{w_1, \dots, w_n\}$

$$n = 128^2 = 16384$$

→ its relatives are used in JPEG2000!

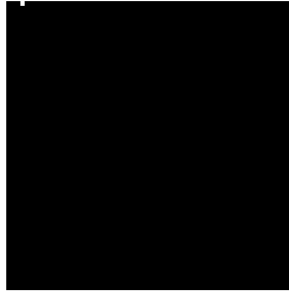
LECTURE 02

1) The standard basis

Original



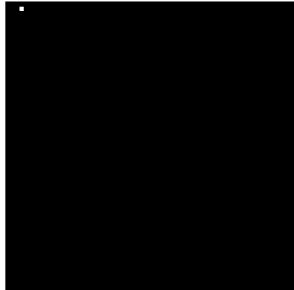
Basis #1



Basis #2



Basis #3



Approx with 1311 terms



Residual



2) The wavelet basis

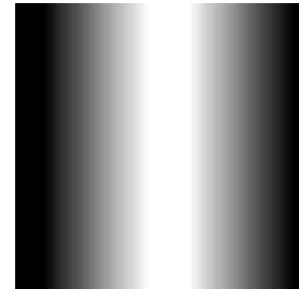
Original



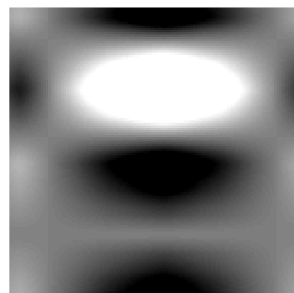
Basis #1



Basis #2



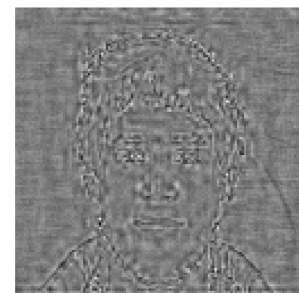
Basis #3



Approx with 1311 terms



Residual



Example 3: Term - Document Matrices for Search

This example is from Berry & Browne: "Understanding Search Engines", 2005.

Consider a set of documents consisting of book titles, and a set of words (or terms) as follows:

Terms	Documents
T1: Bab(y,ies,y's)	D1: <u>Infant</u> & <u>Toddler</u> First Aid
T2: Child(ren's)	D2: <u>Babies</u> & <u>Children's</u> Room (For Your <u>Home</u>)
T3: Guide	D3: <u>Child</u> <u>Safety</u> at <u>Home</u>
T4: Health	D4: Your <u>Baby's</u> <u>Health</u> and <u>Safety</u> : From <u>Infant</u> to <u>Toddler</u>
T5: Home	D5: <u>Baby</u> <u>Proofing</u> Basics
T6: Infant	D6: Your <u>Guide</u> to Easy Rust <u>Proofing</u>
T7: Proofing	D7: Beanie <u>Babies</u> Collector's <u>Guide</u>
T8: Safety	
T9: Toddler	

Then, consider the following term - document matrix of size 9×7 .

The 9×7 term-by-document matrix before normalization, where the element $\hat{a}_{ij}$ is the number of times term i appears in document title j :

$$\text{Terms} \left\{ \begin{array}{l} T2 \\ T7 \end{array} \right. \hat{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \leftarrow \text{rather sparse!}$$

D2 D3 D5 D6 documents

Now suppose we want to retrieve books on "Child Proofing"

T2 T7

Then this can be represented by a query vector:

$$q = [0 \underset{T2}{1} 0 0 0 0 \underset{T7}{1} 0 0]^T$$

Try to match q with each column vector (i.e., a document).

⇒ No exact hit, but the close ones are D2, D3, D5, D6

Generalizing this to:

{ terms = the English dictionary
documents = the entire webpages

the term-document matrix

becomes huge!!

est. in 2017

300,000 × 47,000,000,000

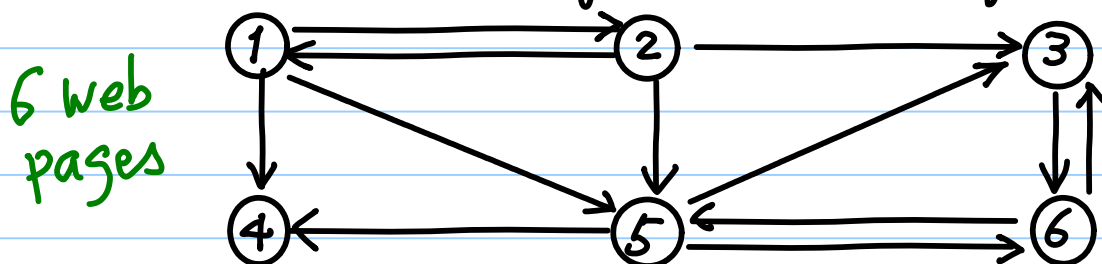
or even larger now!

How to deal with such a huge matrix for search?

⇒ We'll learn how later in this course.

Example 4: Link Graph Matrix of webpages

Consider a very small set of webpages



This graph means that

① has outlinks to ②, ④, ⑤
(pointers)

① has an inlink from ②

② has ---

A link graph matrix $P = (P_{ij})$ is defined by

$$P_{ij} = \begin{cases} \text{nonzero} & \text{if } \exists (j) \rightarrow (i) \\ 0 & \text{otherwise} \end{cases}$$

there exists

The nonzero const. is determined by the normalization

$$\sum_{i=1}^n P_{ij} = 1 \quad \text{for } 1 \leq j \leq n$$

So, in this small example, we have

$$P = \begin{bmatrix} 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & \frac{1}{3} & 0 \end{bmatrix} //$$

One possible measure of importance of a webpage (i)

= how many other important pages have outlinks to (i)

- This can be solved by the eigenvalue decomposition of P

- In reality, $n > \text{billions!}$

- Depending on the literature, P_n^T is denoted as " P ", i.e.,
 $\sum_{j=1}^n P_{ij} = 1, \quad 1 \leq i \leq n$

$\Rightarrow$ We'll discuss link graph matrices in detail later in this course.

Floating Point Computations

★ Floating Point Numbers

On a digital computer, one can only use a finite number of bits to represent a real number, e.g., $\sqrt{2}$, π , e , etc.

Hence, the floating point representation was invented and standardized.

⇒ IEEE 754 Standard.

The idealized floating point system is a discrete subset $F \subset \mathbb{R}$

$$F := \{0\} \cup \{x \in \mathbb{R} \mid x = \pm 1.d_1 d_2 \dots d_t \times \beta^e,$$

$$d_j \in \{0, 1, \dots, \beta-1\}, 1 \leq j \leq t$$

$$t \geq 1, t \in \mathbb{N},$$

$$\beta \geq 2, \beta \in \mathbb{N},$$

$$e \in \mathbb{Z} \}$$

$$d_1 d_2 \dots d_t = \text{mantissa (or fraction)}$$

$$\beta = \text{base (or radix)}, \text{ typically } \beta = 2.$$

$$e = \text{exponent}, t = \text{precision}.$$

$$1.d_1 d_2 \dots d_t \times \beta^e \text{ in base } \beta$$

$$= \left(\frac{1}{\beta^0} + \frac{d_1}{\beta^1} + \frac{d_2}{\beta^2} + \dots + \frac{d_t}{\beta^t} \right) \times \beta^e \text{ in } \underline{\text{base 10}}$$

LECTURE 02

In C++,
they are
float, double, long double.

Basic f.p. types for a real number:

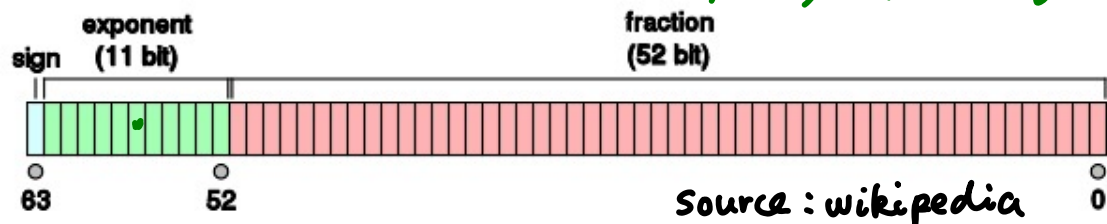
Single precision f.p. number = 32 bits

Double " " = 64 bits

Quadruple " " = 128 bits

Julia Default!

⇒ 4, 8, 16 bytes



How about t ?

In IEEE 754 Standard,

$$t = \begin{cases} 24 & \text{for single precision} \\ 53 & \text{double " } \\ 113 & \text{quadruple " } \end{cases}$$

$\mu := \frac{1}{2} \beta^{1-t}$ is called
the machine epsilon
or the unit round-off
of the floating point system,
often denoted by ϵ_{mach} .

$$\begin{aligned} \text{Single Precision } \mu &= 2^{-24} \approx 5.96 \times 10^{-8} \\ \text{Double Precision } \mu &= 2^{-53} \approx 1.11 \times 10^{-16} \\ \text{Quadruple Precision } \mu &= 2^{-113} \approx 9.63 \times 10^{-35} \end{aligned}$$

How about e ?

In IEEE 754 Standard,

$$\begin{cases} \text{single precision} : -126 \leq e \leq 127 \\ \text{double} \quad \quad \quad : -1022 \leq e \leq 1023 \\ \text{quadruple} \quad \quad : -16382 \leq e \leq 16383 \end{cases}$$

Hence the range of representable positive numbers are roughly:

$$\begin{cases} \text{single prec.} : 10^{-38} \sim 10^{+38} \\ \text{double} \quad \quad : 10^{-308} \sim 10^{+308} \\ \text{quadruple} \quad : 10^{-4932} \sim 10^{+4932} \end{cases}$$

Now, we have:

$$(*) \quad \forall x \in \mathbb{R}, \exists x' \in \mathbb{F} \text{ s.t. } |x - x'| \leq \mu |x|$$

Let's define the following function:

$$fl : \mathbb{R} \rightarrow \mathbb{F}$$

a fun giving the closest f.p.
approximation to a given real num.
= rounded equivalent in $\mathbb{F}$.

So, the statement $(*)$ is equivalent to
 $\forall x \in \mathbb{R}, \exists \varepsilon \in \mathbb{R}$ with $|\varepsilon| \leq \mu$ s.t.
 $fl(x) = x(1 + \varepsilon)$

Why is this so?

$$\odot \quad |x - fl(x)| \leq \mu |x|$$

$$\Leftrightarrow -\mu |x| \leq x - fl(x) \leq \mu |x|$$

$$\Leftrightarrow x - \mu |x| \leq fl(x) \leq x + \mu |x|$$

$$\Leftrightarrow fl(x) = x + \varepsilon x \text{ for } \exists \varepsilon \text{ with } |\varepsilon| \leq \mu$$

In essence, on any computer,
for any real number x , our best
hope is $\left| \frac{x - fl(x)}{x} \right| \leq \mu = \varepsilon_{mach}$

= relative error

$\approx 10^{-16}$
in Julia

`eps()`

↑ a function

Floating Point Arithmetic

Basic arithmetic operations

in $\mathbb{R} \Rightarrow +, -, \times, \div$

in $\mathbb{F} \Rightarrow \oplus, \ominus, \otimes, \oslash$

Ideal goal of computer arithmetic:

$$\forall x, y \in \mathbb{F}, \quad x \otimes y = fl(x \circ y)$$

$\circ = +, -, \times, \text{ or } \div$ operations in $\mathbb{F}$ operations in $\mathbb{R}$

From the above discussion, we have

$$(*) \quad x \otimes y = (x \circ y)(1 + \varepsilon), \quad \exists |\varepsilon| < \mu$$

If a computer truncates (instead of rounds) the intermediate result $x \circ y$, then in (★) μ should be replaced by 2μ .

Nick Trefethen (Oxford) calls (★)
"The Fundamental Axiom of
Floating Point Arithmetic."

★ Floating Point Operations (FLOPS)

FLOPS = flops = flop/s
= f.p. operations/second
= a measure of a computer's
performance

flop = a unit of one arithmetic
operation on a computer

Ex. The statement in a code

$y = y + a * x;$

How many flops does this require?

$\Rightarrow 2 \text{ flops}$

as of June 2016, it is
Sunway Taihualight (China)
Sep. 2025
El Capitan
(LLNL,
USA)

Note: The world fastest computer can perform
 $\geq 93 \text{ PetaFLOPS} = 93 \times 10^{15} \text{ FLOPS!}$

$1,742 \text{ exaFLOPS} = 1.742 \times 10^{18} \text{ FLOPS!}$