Math 504 (Fall 2010) - Lecture 1

IEEE Double Precision Arithmetic and Operation Count

Emre Mengi
Department of Mathematics
Koç University, Istanbul
emengi@ku.edu.tr

Outline

- IEEE double precision arithmetic
- Performing floating point operations in IEEE standards
- Floating point operation count (*flop* count)

IEEE Double Precision Arithmetic

- 64 binary digits (bits) for each floating point number

$$f = \pm\,(1.b_1 b_2 \ldots b_{52})_2 \times 2^{(a_1 a_2 \ldots a_{11})_2}$$

IEEE Double Precision Arithmetic

- 64 binary digits (bits) for each floating point number

$$f = \pm\,(1.b_1 b_2 \ldots b_{52})_2 \times 2^{(a_1 a_2 \ldots a_{11})_2}$$

  - 52 bits for the significand (mantissa)
  - 11 bits for the exponent
  - 1 bit for the sign

IEEE Double Precision Arithmetic

- 64 binary digits (bits) for each floating point number

$$f = \pm\,(1.b_1 b_2 \ldots b_{52})_2 \times 2^{(a_1 a_2 \ldots a_{11})_2}$$

  - 52 bits for the significand (mantissa)
  - 11 bits for the exponent
  - 1 bit for the sign

e.g.

$$(1.\underbrace{1}_{b_1}0\ldots0\underbrace{1}_{b_{52}})_2 \times 2^{(00\ldots010)_2} = \left(1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-52}\right) \times 2^2$$

IEEE Double Precision Arithmetic

- 11 bits can be used to represent $2^{11} = 2048$ exponent values.

IEEE Double Precision Arithmetic

- 11 bits can be used to represent $2^{11} = 2048$ exponent values.
- $(00\ldots0)_2$ and $(11\ldots1)_2$ are reserved for special purposes.
  - $(11\ldots1)_2$ for ∞ and NaN (not a number *e.g.* $\infty - \infty$).

IEEE Double Precision Arithmetic

- 11 bits can be used to represent $2^{11} = 2048$ exponent values.
- $(00\ldots0)_2$ and $(11\ldots1)_2$ are reserved for special purposes.
  - $(11\ldots1)_2$ for ∞ and NaN (not a number *e.g.* $\infty - \infty$).
- The remaining 2046 exponent values represent any integer in $[-1022, 1023]$.

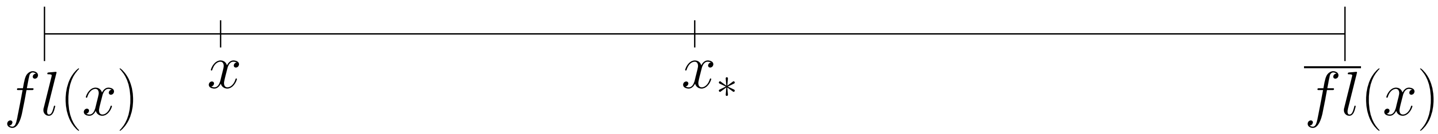
IEEE Double Precision Arithmetic

- 11 bits can be used to represent $2^{11} = 2048$ exponent values.
- $(00\ldots0)_2$ and $(11\ldots1)_2$ are reserved for special purposes.
  - $(11\ldots1)_2$ for ∞ and NaN (not a number *e.g.* $\infty - \infty$).
- The remaining 2046 exponent values represent any integer in $[-1022, 1023]$.
- Let x be any floating point number in double precision.

$$\begin{aligned}
-(1.11\ldots1)_2 \times 2^{1023} &\leq x \leq (1.11\ldots1)_2 2^{1023} \\
-((10.0\ldots0)_2 - (0.0\ldots1)_2) \times 2^{1023} &\leq x \leq ((10.0\ldots0)_2 - (0.0\ldots1)_2) \times 2^{1023} \\
\underbrace{-(2-2^{-52}) \times 2^{1023}}_{R_{\min}} &\leq x \leq \underbrace{(2-2^{-52}) \times 2^{1023}}_{R_{\max}} \approx 1.8 \times 10^{308}
\end{aligned}$$

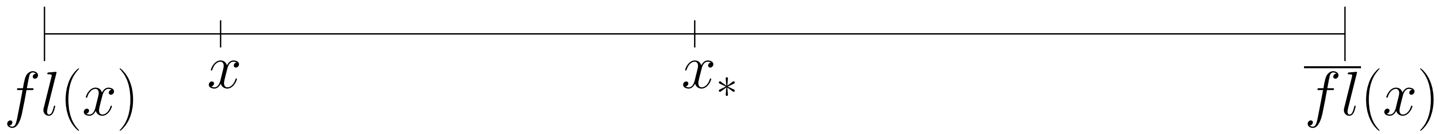
IEEE Double Precision Arithmetic

- ϵ_{mach} : machine precision (Unit round-off error)
  maximal relative error due to floating point representation



IEEE Double Precision Arithmetic

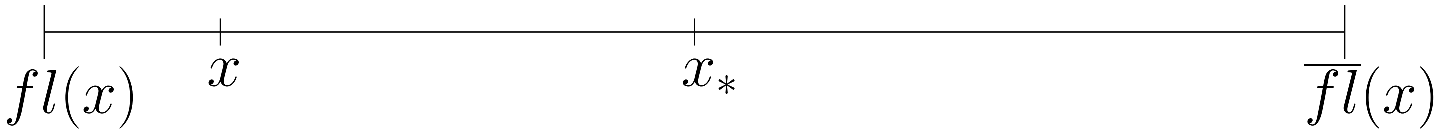
- ϵ_{mach} : machine precision (Unit round-off error)
  maximal relative error due to floating point representation



$$x = s \times 2^E \in (R_{\min}, R_{\max})$$

IEEE Double Precision Arithmetic

- ϵ_{mach} : machine precision (Unit round-off error)
  maximal relative error due to floating point representation

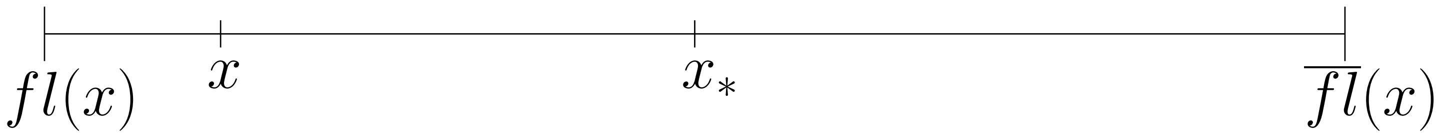


$x = s \times 2^E \in (R_{\min}, R_{\max})$

$fl(x) = \hat{s} \times 2^E$ (floating point number closest to x)

IEEE Double Precision Arithmetic

- ϵ_{mach} : machine precision (Unit round-off error)
  maximal relative error due to floating point representation



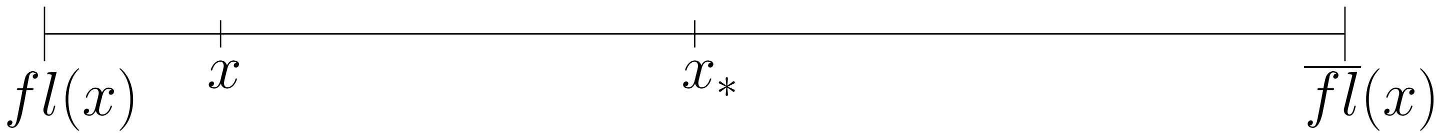
$$x = s \times 2^E \in (R_{\min}, R_{\max})$$

$fl(x) = \hat{s} \times 2^E$ (floating point number closest to x)

$$\overline{fl}(x) = (\hat{s} + 2^{-52}) \times 2^E$$

IEEE Double Precision Arithmetic

- ϵ_{mach} : machine precision (Unit round-off error)
  maximal relative error due to floating point representation



$$x = s \times 2^E \in (R_{\min}, R_{\max})$$

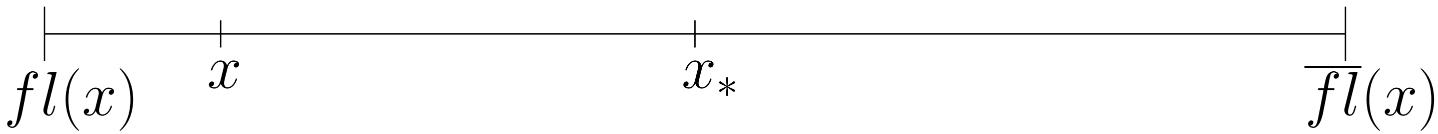
$fl(x) = \hat{s} \times 2^E$ (floating point number closest to x)

$$\overline{fl}(x) = (\hat{s} + 2^{-52}) \times 2^E$$

$$x_* = \frac{fl(x) + \overline{fl}(x)}{2} = \frac{\hat{s} \times 2^E + (\hat{s} + 2^{-52}) \times 2^E}{2} = \left(\hat{s} + 2^{-53}\right) \times 2^E$$

IEEE Double Precision Arithmetic

- ϵ_{mach} : machine precision (Unit round-off error)
  maximal relative error due to floating point representation



$$x = s \times 2^E \in (R_{\min}, R_{\max})$$

$fl(x) = \hat{s} \times 2^E$ (floating point number closest to x)

$$\overline{fl}(x) = (\hat{s} + 2^{-52}) \times 2^E$$

$$x_* = \frac{fl(x)+\overline{fl}(x)}{2} = \frac{\hat{s}\times 2^E + (\hat{s}+2^{-52})\times 2^E}{2} = \left(\hat{s} + 2^{-53}\right) \times 2^E$$

- Relative error

$$\frac{|x - fl(x)|}{|x|} \leq \frac{|x_* - fl(x)|}{|x_*|} = \frac{2^{-53} \times 2^E}{s \times 2^E} \leq \underbrace{2^{-53}}_{\epsilon_{mach}} \approx 10^{-16} \quad (|s| \geq 1)$$

IEEE Double Precision Arithmetic

- Smallest non-zero number in absolute value

IEEE Double Precision Arithmetic

- Smallest non-zero number in absolute value
  - When $(a_1a_2\ldots a_{11})_2 = 0$ the floating point number is in the (subnormalized) form

$$(0.b_1\ldots b_{52})_2 \times 2^{-1022}$$

IEEE Double Precision Arithmetic

- Smallest non-zero number in absolute value
  - When $(a_1 a_2 \ldots a_{11})_2 = 0$ the floating point number is in the (subnormalized) form

$$(0.b_1 \ldots b_{52})_2 \times 2^{-1022}$$

  - The smallest number

$$(0.0 \ldots 01)_2 \times 2^{-1022} = 2^{-52} \times 2^{-1022} = 2^{-1074} \approx 4.94 \times 10^{-324}$$

Performing Floating Point Operations in IEEE Standards

- Floating point operations or flops ($\oplus, \otimes, \ominus, \oslash$) in single or double precision
  - IEEE standards require the flops to satisfy

$$x \oplus y = fl(x + y)$$
$$x \ominus y = fl(x - y)$$
$$x \otimes y = fl(x \times y)$$
$$x \oslash y = fl(x/y)$$

where x and y are floating point numbers.

Performing Floating Point Operations in IEEE Standards

- Floating point operations or flops ($\oplus, \otimes, \ominus, \oslash$) in single or double precision
  - IEEE standards require the flops to satisfy

$$x \oplus y = fl(x+y)$$
$$x \ominus y = fl(x-y)$$
$$x \otimes y = fl(x \times y)$$
$$x \oslash y = fl(x/y)$$

where x and y are floating point numbers.

e.g.

In single precision $1 \oplus 2^{-23} = 1 + 2^{-23}$, but $1 \oplus 2^{-24} = 1$

(Note: In single precision 23 and 8 bits are reserved for mantissa and exponent.)

Performing Floating Point Operations in IEEE Standards

- Floating point operations or flops ($\oplus, \otimes, \ominus, \oslash$) in single or double precision
  - IEEE standards require the flops to satisfy

$$x \oplus y = fl(x + y)$$
$$x \ominus y = fl(x - y)$$
$$x \otimes y = fl(x \times y)$$
$$x \oslash y = fl(x/y)$$

    where x and y are floating point numbers.

e.g.
In single precision $1 \oplus 2^{-23} = 1 + 2^{-23}$, but $1 \oplus 2^{-24} = 1$
(Note: In single precision 23 and 8 bits are reserved for mantissa and exponent.)

In double precision $1 \oplus 2^{-52} = 1 + 2^{-52}$, but $1 \oplus 2^{-53} = 1$

Performing Floating Point Operations in IEEE Standards

- Floating point operations or flops ($\oplus, \otimes, \ominus, \oslash$) in single or double precision
  - IEEE standards require the flops to satisfy

$$x \oplus y = fl(x+y)$$
$$x \ominus y = fl(x-y)$$
$$x \otimes y = fl(x \times y)$$
$$x \oslash y = fl(x/y)$$

    where x and y are floating point numbers.

e.g.
In single precision $1 \oplus 2^{-23} = 1 + 2^{-23}$, but $1 \oplus 2^{-24} = 1$
(Note: In single precision 23 and 8 bits are reserved for mantissa and exponent.)

In double precision $1 \oplus 2^{-52} = 1 + 2^{-52}$, but $1 \oplus 2^{-53} = 1$

In double precision

$$\begin{aligned}(1+2^{-52}) \otimes (2+2^{-51}) &= fl(2+2^{-51}+2^{-51}+2^{-103}) \\ &= fl((1+2^{-52}+2^{-52}+2^{-104}) \times 2) = 2(1+2^{-51})\end{aligned}$$

Floating Point Operation Count

- Efficiency of an algorithm is determined by the total $\#$ of $\oplus, \otimes, \ominus, \oslash$ required.

Floating Point Operation Count

- Efficiency of an algorithm is determined by the total $\#$ of $\oplus, \otimes, \ominus, \oslash$ required.
- Crudeness in flop count

Floating Point Operation Count

- Efficiency of an algorithm is determined by the total $\#$ of $\oplus, \otimes, \ominus, \oslash$ required.
- Crudeness in flop count
  - Time required for data transfers is ignored.

Floating Point Operation Count

- Efficiency of an algorithm is determined by the total $\#$ of $\oplus, \otimes, \ominus, \oslash$ required.
- Crudeness in flop count
  - Time required for data transfers is ignored.
  - All of the operations $\oplus, \otimes, \ominus, \oslash$ are considered of same computational difficulty. In reality $\otimes, \oslash$ are more expensive.

Floating Point Operation Count

- Inner (or dot) product : Let $f : \mathbf{R}^n \rightarrow \mathbf{R}$ be defined as

$$f(x) = a_1x_1 + a_2x_2 + \ldots a_nx_n = a^Tx$$

where $a = \begin{bmatrix} a_1 & \ldots & a_n \end{bmatrix}^T \in \mathbf{R}^n$ and $x = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}^T \in \mathbf{R}^n$.

  - Pseudocode to compute $f(x)$

    $f \leftarrow 0$

    for $j = 1, n$ **do**

    $\underbrace{f \leftarrow f + a_jx_j}_{2\ flops}$

    end for

    Return f

  - Total flop count : 2 flops per iteration for $j = 1, \ldots, n$

$$Total\ \#\ of\ flops\ = \sum_{j=1}^{n} 2 = 2n$$

Floating Point Operation Count

- Matrix-vector product : Let $g : \mathbf{R}^n \to \mathbf{R}^m$ be defined as

$$g(x) = Ax = x_1 A_1 + x_2 A_2 + \cdots + x_n A_n$$

where $A = \begin{bmatrix} A_1 & \ldots & A_n \end{bmatrix}^T$ is an $m \times n$ real matrix with $A_1, \ldots, A_n \in \mathbf{R}^m$ and $x = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}^T \in \mathbf{R}^n$.

Floating Point Operation Count

- Matrix-vector product : Let g : $\mathbf{R}^n \to \mathbf{R}^m$ be defined as

$$g(x) = Ax = x_1 A_1 + x_2 A_2 + \cdots + x_n A_n$$

where $A = \begin{bmatrix} A_1 & \ldots & A_n \end{bmatrix}^T$ is an $m \times n$ real matrix with $A_1, \ldots, A_n \in \mathbf{R}^m$ and $x = \begin{bmatrix} x_1 & \ldots & x_n \end{bmatrix}^T \in \mathbf{R}^n$.

e.g.

$$\begin{bmatrix} 2 & 1 & -2 \\ 1 & 0 & -1 \\ 3 & -1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix} = 2 \begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix} - 2 \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} + 1 \begin{bmatrix} -2 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 10 \end{bmatrix}$$

Floating Point Operation Count

- Pseudocode to compute $g(x) = Ax$

  Given an $m \times n$ real matrix A and $x \in \mathbf{R}^n$.

  $g \leftarrow 0$ (where $g \in \mathbf{R}^n$)

  for $j = 1, n$ **do**

  $$\underbrace{g \leftarrow g + x_j A_j}_{2m\ flops}$$

  end for

  Return g

- Above $g + x_j A_j$ requires m addition and m multiplication for each j.

- Total flop count : $2m$ flops per iteration for $j = 1, \ldots, n$

$$Total\ \#\ of\ flops\ = \sum_{j=1}^{n} 2m = 2mn$$

Floating Point Operation Count

- Inner product view of the matrix-vector product $g(x) = Ax$.

$$g(x) = \begin{bmatrix} \bar{A}_1 x \\ \bar{A}_2 x \\ \vdots \\ \bar{A}_m x \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{nn}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix} \text{ where } A = \begin{bmatrix} \bar{A}_1 \\ \bar{A}_2 \\ \vdots \\ \bar{A}_m \end{bmatrix}$$

and $\bar{A}_1, \ldots, \bar{A}_m$ are the rows of A and a_{ij} is the entry of A at the ith row and jth column.

Floating Point Operation Count

- Inner product view of the matrix-vector product $g(x) = Ax$.

$$g(x) = \begin{bmatrix} \bar{A}_1 x \\ \bar{A}_2 x \\ \vdots \\ \bar{A}_m x \end{bmatrix} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{nn}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix} \text{ where } A = \begin{bmatrix} \bar{A}_1 \\ \bar{A}_2 \\ \vdots \\ \bar{A}_m \end{bmatrix}$$

and $\bar{A}_1, \ldots, \bar{A}_m$ are the rows of A and a_{ij} is the entry of A at the ith row and jth column.

e.g.

$$\begin{bmatrix} 2 & 1 & -2 \\ 1 & 0 & -1 \\ 3 & -1 & 2 \end{bmatrix} \begin{bmatrix} 2 \\ -2 \\ 1 \end{bmatrix} = \begin{bmatrix} (2)(2) + (1)(-2) + (-2)(1) \\ (1)(2) + (0)(-2) + (-1)(1) \\ (3)(2) + (-1)(-2) + (2)(1) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 10 \end{bmatrix}$$

Floating Point Operation Count

- Pseudocode to compute $g(x) = Ax$ exploiting the inner-product view

  Given an $m \times n$ real matrix A and $x \in \mathbf{R}^n$.

  $g \leftarrow 0$ (where $g \in \mathbf{R}^n$)

  for $i = 1, m$ **do**

      for $j = 1, n$ **do**

  $$\underbrace{g_i \leftarrow g_i + a_{ij}x_j}_{2\ flops}$$

      end for

  end for

  Return g

- Total flop count : 2 flops per iteration for each $j = 1, \ldots, n$ and $i = 1, \ldots, m$

$$Total\ \#\ of\ flops\ = \sum_{i=1}^{m} \sum_{j=1}^{n} 2 = \sum_{i=1}^{m} 2n = 2mn$$

Floating Point Operation Count

- Matrix-matrix product : Given an $n \times p$ matrix A and a $p \times m$ matrix X. The product $B = AX$ is an $n \times m$ matrix and defined such that

$$b_{ij} = \bar{A}_i X_j = \sum_{k=1}^{p} a_{ik} x_{kj}$$

where $\bar{A}_i$ is the ith row of A, X_j is the jth column of X and b_{ij}, a_{ij}, x_{ij} denote the (i, j)-entry of B, A and X,respectively.

Floating Point Operation Count

- Matrix-matrix product : Given an $n \times p$ matrix A and a $p \times m$ matrix X. The product $B = AX$ is an $n \times m$ matrix and defined such that

$$b_{ij} = \bar{A}_i X_j = \sum_{k=1}^{p} a_{ik} x_{kj}$$

where $\bar{A}_i$ is the ith row of A, X_j is the jth column of X and b_{ij}, a_{ij}, x_{ij} denote the (i,j)-entry of B, A and X,respectively.

e.g.

$$\begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} 2(-1)+1(1) & 2(1)+1(-2) \\ 1(-1)+0(1) & 1(1)+0(-2) \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ -1 & 1 \end{bmatrix}$$

Floating Point Operation Count

- Pseudocode to compute the product $B = AX$

  Given $n \times p$ and $p \times m$ matrices A and X.

  $B \leftarrow 0$

  for $i = 1, n$ **do**

      for $j = 1, m$ **do**

          for $k = 1, p$ **do**

  $$\underbrace{b_{ij} \leftarrow b_{ij} + a_{ik}x_{kj}}_{2\ flops}$$

          end for

      end for

  end for

  Return g

- Total flop count : 2 flops per iteration for each $k = 1, \ldots, p$, $j = 1, \ldots, m$ and $i = 1, \ldots, n$

$$Total\ \#\ of\ flops\ = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{p} 2 = 2nmp$$

Floating Point Operation Count

- Big-O notation

Floating Point Operation Count

- Big-O notation
  - The inner product $a^T x$ requires $2n = O(n)$ flops (linear $\#$ of flops).

Floating Point Operation Count

- Big-O notation
  - The inner product $a^T x$ requires $2n = O(n)$ flops (linear $\#$ of flops).
  - The matrix-vector product Ax for a square matrix A (with $m = n$) requires $2n^2 = O(n^2)$ flops (quadratic $\#$ of flops).

Floating Point Operation Count

- Big-O notation
  - The inner product $a^T x$ requires $2n = O(n)$ flops (linear $\#$ of flops).
  - The matrix-vector product Ax for a square matrix A (with $m = n$) requires $2n^2 = O(n^2)$ flops (quadratic $\#$ of flops).
  - The matrix-matrix product AX for square $n \times n$ matrices A and X (with $m = n = p$) requires $2n^3 = O(n^3)$ flops (cubic $\#$ of flops).

Floating Point Operation Count

- Big-O notation
  - The inner product $a^T x$ requires $2n = O(n)$ flops (linear # of flops).
  - The matrix-vector product Ax for a square matrix A (with $m = n$) requires $2n^2 = O(n^2)$ flops (quadratic # of flops).
  - The matrix-matrix product AX for square $n \times n$ matrices A and X (with $m = n = p$) requires $2n^3 = O(n^3)$ flops (cubic # of flops).

- The notation $g(n) = O(f(n))$ means asymptotically $f(n)$ scaled up to a constant grows at least as fast as $g(n)$, *i.e.*

> $g(n) = O(f(n))$ if there exists an n_0 and c such that
>
> $$g(n) \leq cf(n) \text{ for all } n \geq n_0$$

Floating Point Operation Count

- Big-O notation
  - The inner product $a^T x$ requires $2n = O(n)$ flops (linear # of flops).
  - The matrix-vector product Ax for a square matrix A (with $m = n$) requires $2n^2 = O(n^2)$ flops (quadratic # of flops).
  - The matrix-matrix product AX for square $n \times n$ matrices A and X (with $m = n = p$) requires $2n^3 = O(n^3)$ flops (cubic # of flops).

- The notation $g(n) = O(f(n))$ means asymptotically $f(n)$ scaled up to a constant grows at least as fast as $g(n)$, *i.e.*

> $g(n) = O(f(n))$ if there exists an n_0 and c such that
>
> $$g(n) \leq cf(n) \text{ for all } n \geq n_0$$

Examples:
$2n = O(n)$ as well as $2n = O(n^2)$ and $2n = O(n^3)$

Floating Point Operation Count

- Big-O notation
  - The inner product $a^T x$ requires $2n = O(n)$ flops (linear $\#$ of flops).
  - The matrix-vector product Ax for a square matrix A (with $m = n$) requires $2n^2 = O(n^2)$ flops (quadratic $\#$ of flops).
  - The matrix-matrix product AX for square $n \times n$ matrices A and X (with $m = n = p$) requires $2n^3 = O(n^3)$ flops (cubic $\#$ of flops).

- The notation $g(n) = O(f(n))$ means asymptotically $f(n)$ scaled up to a constant grows at least as fast as $g(n)$, *i.e.*

> $g(n) = O(f(n))$ if there exists an n_0 and c such that
>
> $$g(n) \leq cf(n) \text{ for all } n \geq n_0$$

Examples:

$2n = O(n)$ as well as $2n = O(n^2)$ and $2n = O(n^3)$

$2n^2 = O(n^2)$ as well as $2n^2 = O(n^3)$, but $2n^2$ is not $O(n)$.

Next Lecture

- Orthogonality (Trefethen&Bau, Lecture 2)
- Norms (Trefethen&Bau, Lecture 3)