

# Vectors in Linear Algebra

Jirasak Sittigorn

Department of Computer Engineering  
Faculty of Engineering  
King Mongkut's Institute of Technology Ladkrabang

# Vector



- Notation
    - Definition
      - a one-dimensional array of  $n$  numbers : a vector of size  $n$

$$x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$$

**-n**: size of the vector

$$-\chi_i \in \mathbb{R}$$

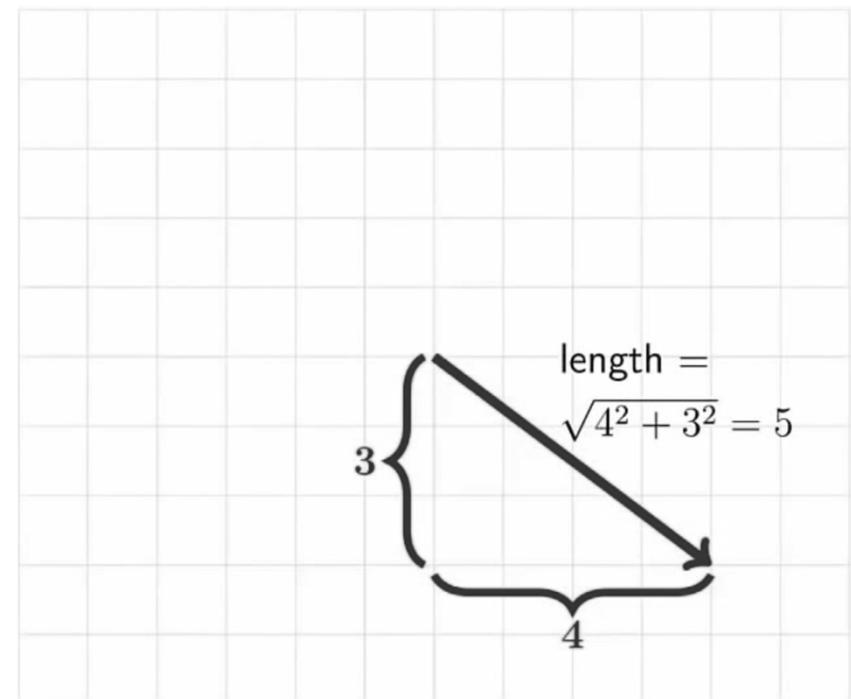
$$-x \in \mathbb{R}^n$$

# Vector

- A vector has a direction and a length:
  - Direction : draw an arrow from the origin to the point  $(x_0, x_1, \dots, x_{n-1})$
  - Its length is given by the Euclidean length of this arrow
$$\sqrt{x_0^2 + x_1^2 + \dots + x_{n-1}^2}$$
» It is denoted by  $\|x\|_2$  called the two-norm (the magnitude of the vector)
- A vector does not have a location

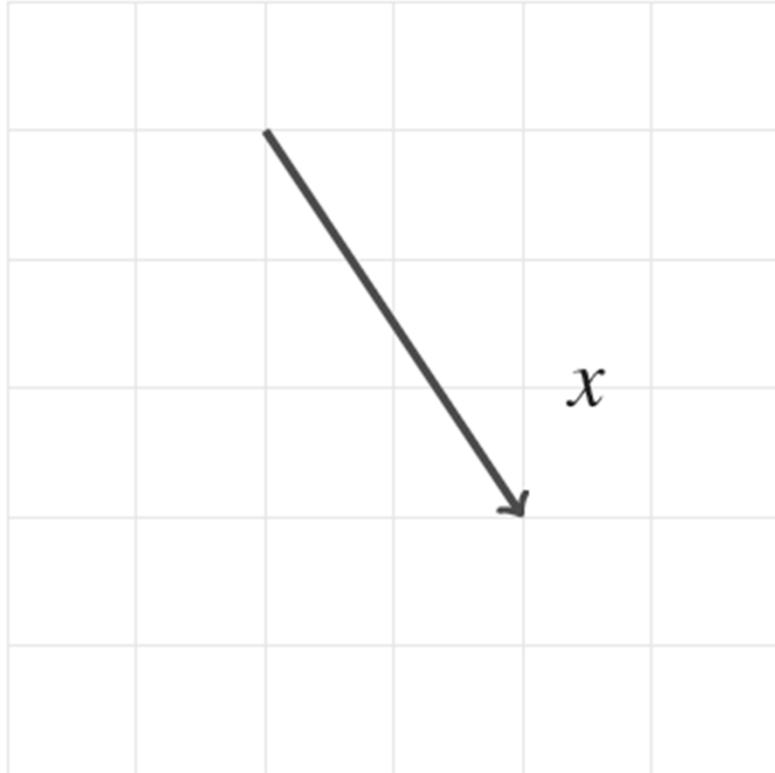
# Example

- Consider  $x = \begin{pmatrix} 4 \\ -3 \end{pmatrix}$ 
  - Component indexed with 0 :  $x_0 = 4$
  - Component indexed with 1 :  $x_1 = -3$
  - The vector is of size 2
    - $x \in \mathbb{R}^2$
  - Length (magnitude) = 5



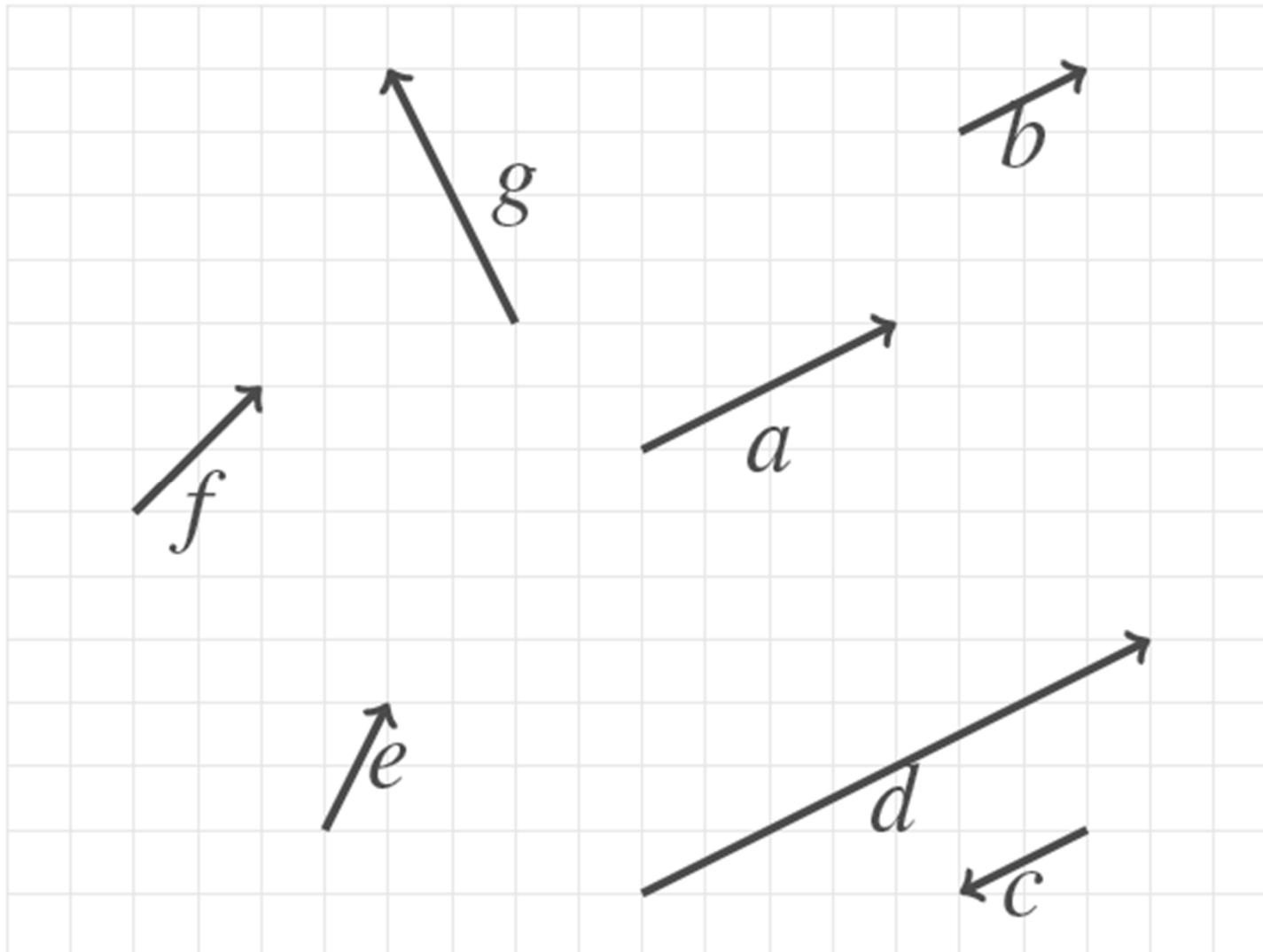
# Exercises

- Consider the following picture



# Exercises

- Consider the following picture



# Exercises



- Write each of the following as a vector:
  - The vector represented geometrically in  $\mathbb{R}^2$  by an arrow from point  $(-1, 2)$  to point  $(0, 0)$
  - The vector represented geometrically in  $\mathbb{R}^2$  by an arrow from point  $(0, 0)$  to point  $(-1, 2)$ .
  - The vector represented geometrically in  $\mathbb{R}^3$  by an arrow from point  $(-1, 2, 4)$  to point  $(0, 0, 1)$ .
  - The vector represented geometrically in  $\mathbb{R}^3$  by an arrow from point  $(1, 0, 0)$  to point  $(4, 2, -1)$ .

# Vector

- Unit Basis Vectors

- Definition

- An important set of vectors is the set of unit basis vectors given by

$$e_j = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \quad \begin{array}{l} \left. \right\} j \text{ zeros} \\ \leftarrow \text{component indexed by } j \\ \left. \right\} n - j - 1 \text{ zeros} \end{array}$$

- where the “1” appears as the component indexed by  $j$ .

# Vector

- The set  $\{e_0, e_1, \dots, e_{n-1}\} \subset \mathbb{R}^n$  given by

$$e_0 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, e_1 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \dots, e_{n-1} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

– Unit basis vectors of  $x \in \mathbb{R}^2$  ?

– Unit basis vectors of  $x \in \mathbb{R}^3$  ?

# Exercises

- Which of the following is not a unit basis vector?

- $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$

- $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

- $\begin{pmatrix} \frac{\sqrt{2}}{2} \\ \frac{2}{2} \\ \frac{\sqrt{2}}{2} \\ \frac{2}{2} \end{pmatrix}$

- $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

- None of these are unit basis vectors.

# Simple Vector Operations

- Equality ( $=$ ), Assignment ( $\mathbf{:=}$ ), and Copy
- Vector Addition (ADD)
- Scaling (SCAL)
- Vector Subtraction

# Simple Vector Operations

- Equality ( $=$ )

$$x = \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix} \text{ and } y = \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix}$$

## – Definition

- Two vectors  $x, y \in \mathbb{R}^n$  are equal if all their components are element-wise equal:

$x = y \text{ if and only if } \chi_i = \psi_i, \text{ for all } 0 \leq i < n$

# Simple Vector Operations

- Assignment ( $\coloneqq$ ) or Copy
  - Algorithm
    - The following algorithm copies vector  $x \in \mathbb{R}^n$  into vector  $y \in \mathbb{R}^n$ , performing the operation  $y \coloneqq x$

$$\begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} \coloneqq \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix}$$

```
for i = 0, ..., n - 1
     $\psi_i \coloneqq \chi_i$ 
endfor
```

# Exercises

- Decide if the two vectors are equal.
  - The vector represented geometrically in  $\mathbb{R}^2$  by an arrow from point  $(-1,2)$  to point  $(0,0)$  and the vector represented geometrically in  $\mathbb{R}^2$  by an arrow from point  $(1,-2)$  to point  $(2,-1)$  are equal.
  - The vector represented geometrically in  $\mathbb{R}^3$  by an arrow from point  $(1,-1,2)$  to point  $(0,0,0)$  and the vector represented geometrically in  $\mathbb{R}^3$  by an arrow from point  $(1,1,-2)$  to point  $(0,2,-4)$  are equal.

# Simple Vector Operations

- Vector Addition (ADD)

- Definition

- Vector addition  $x + y$  (sum of vectors) is defined by

$$x + y = \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix} + \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} = \begin{pmatrix} \chi_0 + \psi_0 \\ \chi_1 + \psi_1 \\ \vdots \\ \chi_{n-1} + \psi_{n-1} \end{pmatrix}$$

# Exercises

- $\begin{pmatrix} -1 \\ 2 \end{pmatrix} + \begin{pmatrix} -3 \\ -2 \end{pmatrix} =$
- $\begin{pmatrix} -3 \\ -2 \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \end{pmatrix} =$
- For  $x, y \in \mathbb{R}^n$ ,  $x + y = y + x$ 
  - Always
  - Sometime
  - Never

# Exercises

- $\begin{pmatrix} -1 \\ 2 \end{pmatrix} + \left( \begin{pmatrix} -3 \\ -2 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right) =$
- $\left( \begin{pmatrix} -1 \\ 2 \end{pmatrix} + \begin{pmatrix} -3 \\ -2 \end{pmatrix} \right) + \begin{pmatrix} 1 \\ 2 \end{pmatrix} =$
- For  $x, y, z \in \mathbb{R}^n$ ,  $(x + y) + z = x + (y + z)$ 
  - Always
  - Sometime
  - Never

# Exercises

- $\begin{pmatrix} -1 \\ 2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} =$
- For  $x \in \mathbb{R}^n$ ,  $x + 0 = x$ , where 0 is zero vector of appropriate size
  - Always
  - Sometime
  - Never

# Simple Vector Operations

- Vector Addition (ADD)

- Algorithm

- The following algorithm assigns the sum of vectors  $x$  and  $y$  (of size  $n$  and stored in arrays  $x$  and  $y$ ) to vector  $z$  (of size  $n$  and stored in array  $z$ ), computing  $z := x + y$

$$\begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \vdots \\ \zeta_{n-1} \end{pmatrix} := \begin{pmatrix} x_0 + \psi_0 \\ x_1 + \psi_1 \\ \vdots \\ x_{n-1} + \psi_{n-1} \end{pmatrix}$$

```
for i = 0, ..., n - 1  
     $\zeta_i := x_i + \psi_i$   
endfor
```

# Simple Vector Operations

- **Scaling (SCAL)**

- Definition

- Multiplying vector  $x$  by scalar  $\alpha$  yields a new vector,  $\alpha x$ , in the same direction as  $x$ , but scaled by a factor  $\alpha$ .

Scaling a vector by  $\alpha$  means each of its components,  $x_i$  is multiplied by  $\alpha$

$$\alpha x = \alpha \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} = \begin{pmatrix} \alpha x_0 \\ \alpha x_1 \\ \vdots \\ \alpha x_{n-1} \end{pmatrix}$$

# Exercises

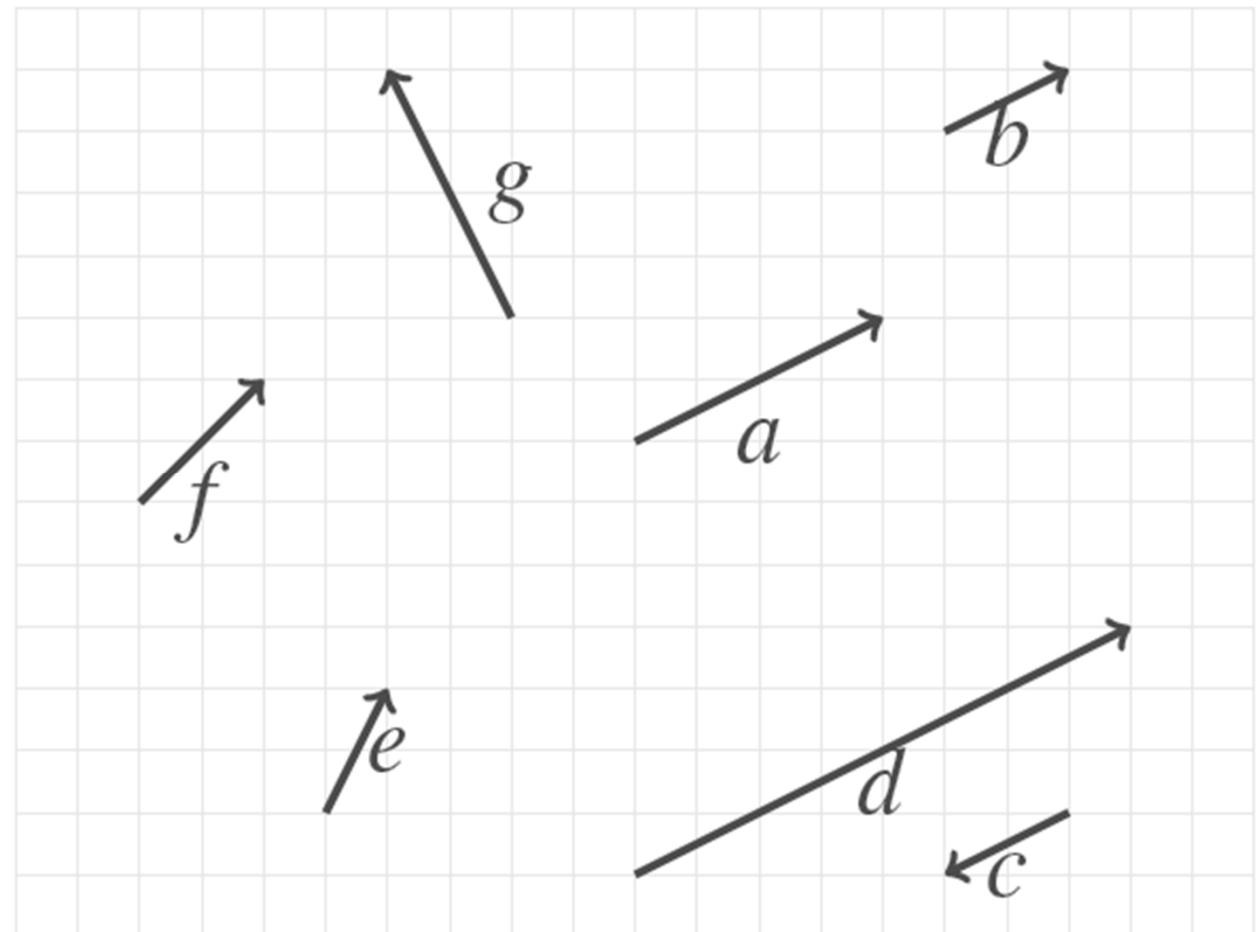
- $\left( \begin{pmatrix} -1 \\ 2 \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \end{pmatrix} \right) + \begin{pmatrix} -1 \\ 2 \end{pmatrix} =$
- $3 \begin{pmatrix} -1 \\ 2 \end{pmatrix} =$

# Exercises

- Consider the following picture

- Which vector equals

- $2a$ ?:
    - $(1/2)a$ ?:
    - $-(1/2)a$ ?



# Simple Vector Operations

- **Scaling (SCAL)**

- Algorithm

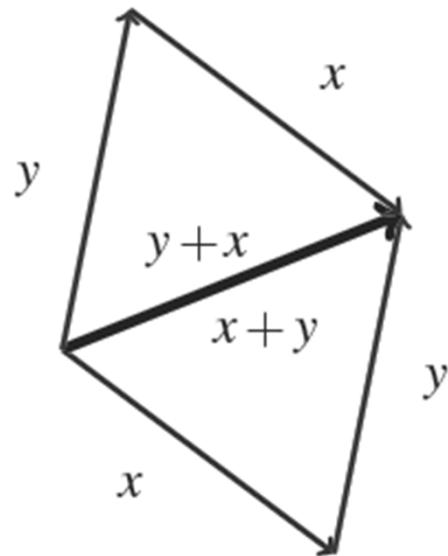
- The following algorithm scales a vector  $x \in \mathbb{R}^n$  by  $\alpha$ , overwriting  $x$  with the result  $\alpha x$

$$\begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \vdots \\ \zeta_{n-1} \end{pmatrix} := \begin{pmatrix} \alpha x_0 \\ \alpha x_1 \\ \vdots \\ \alpha x_{n-1} \end{pmatrix}$$

```
for i = 0, ..., n - 1
     $\zeta_i := \alpha x_i$ 
endfor
```

# Simple Vector Operations

- **Vector Subtraction**
  - Recall the geometric interpretation for adding two vectors,  $x, y \in \mathbb{R}^n$



- Subtracting  $y$  from  $x$  is defined as
$$x - y = x + (-y)$$

# Simple Vector Operations

- Now computing  $x - y$  when  $x, y \in \mathbb{R}^n$  is a simple matter of subtracting components of  $y$  off the corresponding components of  $x$

$$x - y = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} - \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} = \begin{pmatrix} x_0 - \psi_0 \\ x_1 - \psi_1 \\ \vdots \\ x_{n-1} - \psi_{n-1} \end{pmatrix}$$

# Exercises

- For  $x \in \mathbb{R}^n$ ,  $x - x = 0$ 
  - Always
  - Sometime
  - Never
- For  $x, y \in \mathbb{R}^n$ ,  $x - y = y - x$ 
  - Always
  - Sometime
  - Never

# Advanced Vector Operations

- Scaled Vector Addition (AXPY)
- Linear Combinations of Vectors
- Dot or Inner Product (DOT)
- Vector Length (NORM2)
- Vector Functions
- Vector Functions that Map a Vector to a Vector

# Advanced Vector Operations

- Scaled Vector Addition (AXPY)

- Definition

- One of the most commonly encountered operations when implementing more complex linear algebra operations is the scaled vector addition, which (given  $x, y \in \mathbb{R}^n$ ) computes  $y := \alpha x + y$

$$\begin{aligned}\alpha x + y &= \alpha \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} + \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} \\ &= \begin{pmatrix} \alpha x_0 \\ \alpha x_1 \\ \vdots \\ \alpha x_{n-1} \end{pmatrix} + \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} = \begin{pmatrix} \alpha x_0 + \psi_0 \\ \alpha x_1 + \psi_1 \\ \vdots \\ \alpha x_{n-1} + \psi_{n-1} \end{pmatrix}\end{aligned}$$

# Advanced Vector Operations

## — Algorithm

- Obviously, one could copy  $x$  into another vector, scale it by  $a$ , and then add it to  $y$ . Usually, however, vector  $y$  is simply updated one element at a time:

$$\begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} := \begin{pmatrix} \alpha x_0 + \psi_0 \\ \alpha x_1 + \psi_1 \\ \vdots \\ \alpha x_{n-1} + \psi_{n-1} \end{pmatrix}$$

```
for i = 0, ..., n - 1  
     $\psi_i := \alpha x_i + \psi_i$   
endfor
```

# Advanced Vector Operations

- Linear Combinations of Vectors

- Definition

- Let  $u, v \in \mathbb{R}^m$  and  $\alpha, \beta \in \mathbb{R}$ . Then  $\alpha u + \beta v$  is said to be a linear combination of vectors  $u$  and  $v$ .

$$\begin{aligned}\alpha u + \beta v &= \alpha \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{m-1} \end{pmatrix} + \beta \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{m-1} \end{pmatrix} \\ &= \begin{pmatrix} \alpha u_0 \\ \alpha u_1 \\ \vdots \\ \alpha u_{m-1} \end{pmatrix} + \beta \begin{pmatrix} \beta v_0 \\ \beta v_1 \\ \vdots \\ \beta v_{m-1} \end{pmatrix} = \begin{pmatrix} \alpha u_0 + \beta v_0 \\ \alpha u_1 + \beta v_1 \\ \vdots \\ \alpha u_{m-1} + \beta v_{m-1} \end{pmatrix}\end{aligned}$$

# Exercises

$$\bullet \quad 3 \begin{pmatrix} 2 \\ 4 \\ -1 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} =$$

$$\bullet \quad -3 \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + 2 \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + 4 \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} =$$

$$\bullet \quad \alpha \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + \gamma \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix}$$
$$\alpha = \qquad \beta = \qquad \gamma =$$

# Advanced Vector Operations

- Algorithm

- Given  $v_0, v_1, \dots, v_{n-1} \in \mathbb{R}^m$  and  $\chi_0, \chi_1, \dots, \chi_{n-1} \in \mathbb{R}$  the linear combination
$$w = \chi_0 v_0 + \chi_1 v_1 + \dots + \chi_{n-1} v_{n-1}$$
- can be computed by first setting the result vector  $w$  to the zero vector of size  $m$ , and then performing  $n$  AXPY operations

```
w = 0      (the zero vector of size m)
for j = 0, ..., n - 1
    w :=  $\chi_j v_j + w$ 
endfor
```

# An important example

- Given any  $x \in \mathbb{R}^n$  with  $x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}$ , this vector can always be written as the linear combination of the unit basis vectors given by

$$x = \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix} = x_0 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} + x_1 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix} + \cdots + x_{n-1} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$
$$= x_0 e_0 + x_1 e_1 + \cdots + x_{n-1} e_{n-1} = \sum_{i=0}^{n-1} x_i e_i$$

- Shortly, this will become really important as we make the connection between linear combinations of vectors, linear transformations, and matrices.

# Advanced Vector Operations

- Dot or Inner Product (DOT)
  - Definition
    - The other commonly encountered operation is the dot (inner) product. It is defined by

$$\begin{aligned} \text{dot}(x, y) &= \sum_{i=0}^{n-1} x_i \psi_i \\ &= x_0 \psi_0 + x_1 \psi_1 + \cdots + x_{n-1} \psi_{n-1} \end{aligned}$$

# Advanced Vector Operations

- Dot or Inner Product (DOT)
  - Alternative notation

$$\begin{aligned}x^T y = \text{dot}(x, y) &= \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}^T \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} \\&= (x_0 \quad x_1 \quad \cdots \quad x_{n-1}) \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} \\&= x_0\psi_0 + x_1\psi_1 + \cdots + x_{n-1}\psi_{n-1}\end{aligned}$$

# Exercises

$$\bullet \begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix}^T \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} =$$

$$\bullet \begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix}^T \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} =$$

$$\bullet \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix} =$$

• For  $x, y \in \mathbb{R}^n, x^T y = y^T x$

- Always
- Sometime
- Never

# Exercises

$$\cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}^T \left( \begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \right) =$$

$$\cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} =$$

$$\cdot \left( \begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \right)^T \begin{pmatrix} 1 \\ 0 \\ 0 \\ 2 \end{pmatrix} =$$

# Exercises

- For  $x, y, z \in \mathbb{R}^n$ ,  $x^T(y + z) = x^Ty + x^Tz$ 
  - Always / Sometime / Never
- For  $x, y, z \in \mathbb{R}^n$ ,  $(x + y)^Tz = x^Tz + x^Tz$ 
  - Always / Sometime / Never
- For  $x, y, z \in \mathbb{R}^n$ ,  $(x + y)^T(x + y) = x^Tx + 2x^Ty + y^Ty$ 
  - Always / Sometime / Never
- For  $x, y, z \in \mathbb{R}^n$ . When  $x^Ty = 0$ ,  $x$  or  $y$  is zero vector.
  - Always / Sometime / Never
- For  $x \in \mathbb{R}^n$ ,  $e_i^T x = x^T e_i = \chi_i$ , where  $\chi_i$  equals the  $i$ th component of  $x$ .
  - Always / Sometime / Never

# Advanced Vector Operations

- Dot or Inner Product (DOT)
  - Algorithm

```
 $\alpha \coloneqq 0$ 
for  $i = 0, \dots, n - 1$ 
     $\alpha \coloneqq \chi_j \psi_j + \alpha$ 
endfor
```

# Advanced Vector Operations

- Vector Length (NORM2)

- Definition

- Let  $x \in \mathbb{R}^n$ . Then the (Euclidean) length of a vector  $x$  (the two-norm) is given by

$$\|x\|_2 = \sqrt{x_0^2 + x_1^2 + \cdots + x_{n-1}^2} = \sqrt{\sum_{i=0}^{n-1} x_i^2}$$

- Here  $\|x\|_2$  notation stands for “the two norm of  $x$ ”, which is another way of saying “the length of  $x$ ”

- A vector of length one is said to be a unit vector.

# Exercises

- Compute the lengths of the following vectors

- $\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} =$

- $\begin{pmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{pmatrix} =$

- $\begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} =$

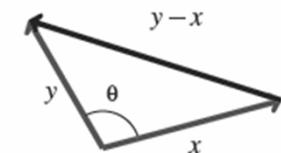
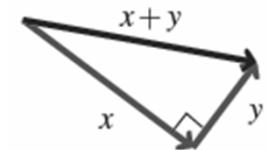
- $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} =$

# Exercises

- Let  $x \in \mathbb{R}^n$ . The length of  $x$  is less than zero:  
 $\|x\|_2 < 0$ .
  - Always / Sometime / Never
- If  $x$  is a unit vector then  $x$  is a unit basis vector.
  - TRUE / FALSE
- If  $x$  is a unit basis vector then  $x$  is a unit vector.
  - TRUE / FALSE

# Exercises

- If  $x$  and  $y$  are perpendicular (orthogonal) then  $x^T y = 0$ .
  - TRUE / FALSE
- Let  $x, y \in \mathbb{R}^n$  be nonzero vectors and let the angle between them equal  $\theta$ . Then  $\cos\theta = \frac{x^T y}{\|x\|_2 \|y\|_2}$ 
  - Always / Sometime / Never
- Let  $x, y \in \mathbb{R}^n$  be nonzero vectors. Then  $x^T y = 0$  if and only if  $x$  and  $y$  are orthogonal (perpendicular).
  - TRUE / FALSE



# Advanced Vector Operations

- Vector Length (NORM2)
  - Algorithm
    - Clearly,  $\|x\|_2 = \sqrt{x^T x}$ , so that the DOT operation can be used to compute this length

# Advanced Vector Operations

- Vector Functions
  - Definition
    - A vector(-valued) function is a mathematical functions of one or more scalars and/or vectors whose output is a vector.

# Example

- $f(\alpha, \beta) = \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix}$  so that  $f(-2, 1) = \begin{pmatrix} -2 + 1 \\ -2 - 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -3 \end{pmatrix}$
- $f\left(\alpha, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) = \begin{pmatrix} \chi_0 + \alpha \\ \chi_1 + \alpha \\ \chi_2 + \alpha \end{pmatrix}$  so that  $f\left(-2, \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}\right) =$   
 $\begin{pmatrix} 1 + (-2) \\ 2 + (-2) \\ 3 + (-2) \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$
- The AXPY and DOT vector functions are other functions that we have already encountered
- $f\left(\alpha, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) = \begin{pmatrix} \chi_0 + \chi_1 \\ \chi_1 + \chi_2 \end{pmatrix}$  so that  $f\left(\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}\right) = \begin{pmatrix} 1 + 2 \\ 2 + 3 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$

# Exercises

- If  $f\left(\alpha, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) = \begin{pmatrix} \chi_0 + \alpha \\ \chi_1 + \alpha \\ \chi_2 + \alpha \end{pmatrix}$ , find
  - $\alpha f\left(\beta, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$
  - $f\left(\beta, \alpha \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$
  - $f\left(\alpha, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix} + \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}\right) =$
  - $f\left(\alpha, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) + f\left(\alpha, \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}\right) =$
- $f\left(1, \begin{pmatrix} 6 \\ 2 \\ 3 \end{pmatrix}\right) =$
- $f\left(\alpha, \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}\right) =$
- $f\left(0, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$
- $f\left(\beta, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$

# Advanced Vector Operations

- Vector Functions that Map a Vector to a Vector
  - Example

$$f(\alpha, \beta) = \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix} \text{ so that } f(-2, 1) = \begin{pmatrix} -2 + 1 \\ -2 - 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -3 \end{pmatrix}$$

We can define

$$g\left(\begin{pmatrix} \alpha \\ \beta \end{pmatrix}\right) = \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix} \text{ so that } g\left(\begin{pmatrix} -2 \\ 1 \end{pmatrix}\right) = \begin{pmatrix} -2 + 1 \\ -2 - 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -3 \end{pmatrix}$$

– Example

$$f\left(\alpha, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) = \begin{pmatrix} \chi_0 + \alpha \\ \chi_1 + \alpha \\ \chi_2 + \alpha \end{pmatrix} \text{ so that } f\left(-2, \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}\right) = \begin{pmatrix} 1 + (-2) \\ 2 + (-2) \\ 3 + (-2) \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}$$

We can define

$$g\left(\begin{pmatrix} \alpha \\ \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) = g\left(\begin{pmatrix} \alpha \\ \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) = \begin{pmatrix} \chi_0 + \alpha \\ \chi_1 + \alpha \\ \chi_2 + \alpha \end{pmatrix} \text{ so that } g\left(\begin{pmatrix} -2 \\ 1 \\ 2 \\ 3 \end{pmatrix}\right) = \begin{pmatrix} -2 + 1 \\ -2 - 1 \\ -2 - 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -3 \end{pmatrix}$$

# Exercises

- If  $f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) = \begin{pmatrix} \chi_0 + 1 \\ \chi_1 + 2 \\ \chi_2 + 3 \end{pmatrix}$ , evaluate

$$f\left(\begin{pmatrix} 6 \\ 2 \\ 3 \end{pmatrix}\right) =$$

$$f\left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}\right) =$$

$$f\left(2\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$$

$$2f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$$

- $f\left(\alpha\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$

- $\alpha f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$

- $f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix} + \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}\right) =$

- $f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) + f\left(\begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}\right) =$

# Exercises

- If  $f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) = \begin{pmatrix} \chi_0 \\ \chi_0 + \chi_1 \\ \chi_0 + \chi_1 + \chi_2 \end{pmatrix}$ , evaluate
  - $f\left(\alpha \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$
  - $\alpha f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$
  - $f\left(\begin{pmatrix} 6 \\ 2 \\ 3 \end{pmatrix}\right) =$
  - $f\left(\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}\right) =$
  - $f\left(2 \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$
  - $2f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) =$
- $f\left(\begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}\right) =$
- $f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix} + \begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}\right) =$
- $f\left(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}\right) + f\left(\begin{pmatrix} \psi_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}\right) =$

# Exercises

- If  $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ , then  $f(0) = 0$ 
  - Always / Sometime / Never
- If  $f: \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $\lambda \in \mathbb{R}$  and  $x \in \mathbb{R}^n$ , then  $f(\lambda x) = \lambda f(x)$ 
  - Always / Sometime / Never

# LAFF Package Development: Vectors

- Starting the Package
- A Copy Routine (`copy`)
- A Routine that Scales a Vector (`scal`)
- A Scaled Vector Addition Routine (`axpy`)
- An Inner Product Routine (`dot`)
- A Vector Length Routine (`norm2`)

# LAFF Package Development: Vectors

- Starting the Package

Operation Abbrev.	Definition	Function	M-script intrinsic	Approx. cost	
				flops	memops
<b>Vector-vector operations</b>					
Copy (COPY)	$y := x$	<code>y = laff_copy( x, y )</code>	<code>y = x</code>	0	$2n$
Vector scaling (SCAL)	$x := \alpha x$	<code>x = laff_scal( alpha, x )</code>	<code>x = alpha * x</code>	$n$	$2n$
Scaled addition (AXPY)	$y := \alpha x + y$	<code>y = laff_axpy( alpha, x, y )</code>	<code>y = alpha * x + y</code>	$2n$	$3n$
Dot product (DOT)	$\alpha := x^T y$	<code>alpha = laff_dot( x, y )</code>	<code>alpha = x' * y</code>	$2n$	$2n$
Length (NORM2)	$\alpha := \ x\ _2$	<code>alpha = laff_norm2( x )</code>	<code>alpha = norm2( x )</code>	$2n$	$n$

# LAFF Package Development: Vectors

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad x = \begin{pmatrix} 1 \\ 4 \\ 7 \end{pmatrix}$$

- In M-script (MATLAB), this can be entered as

$A = [1 2 3; 4 5 6; 7 8 9]$

- Now, a (column) vector can be entered as

$x = [1; 4; 7]$

- and is hence stored just like a matrix with only one column. Similarly, a row vector can be entered as

$x = [1 4 7]$

# LAFF Package Development: Vectors

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad x = \begin{pmatrix} 1 \\ 4 \\ 7 \end{pmatrix}$$

A = [1 2 3

4 5 6

7 8 9

]

x = [1

4

7

]

A(2,1) => 4

# LAFF Package Development: Vectors

```
>> S = sin(180)          >> [S, C, T] = trigon(180)  
  
S =  
-0.8012  
  
>> C = cos(180)  
  
C =  
-0.5985  
  
>> T = tan(180)  
  
T =  
1.3387
```

S =  
1.2246e-16

C =  
-1

T =  
-1.2246e-16

```
deg2rad.m  × +  
1 - [-] function [rad] = deg2rad(deg)  
2 - | rad = deg * pi / 180;  
3 - | end
```

```
trigon.m  × +  
1 - [-] function [x, y, z] = trigon(d)  
2 - | x = deg2rad(d);  
3 - | y = sin(r);  
4 - | z = tan(r);  
5 - | end
```

# LAFF Package Development: Vectors

- A Copy Routine (`copy`)
  - Implement the function `laff_copy` that copies a vector into another vector. The function is defined as
$$\text{function } [\text{y\_out}] = \text{laff\_copy}(\text{x}, \text{y})$$
  - Where
    - $\text{x}$  and  $\text{y}$  must each be either an  $n \times 1$  array (column vector) or a  $1 \times n$  array (row vector);
    - $\text{y\_out}$  must be the same kind of vector as  $\text{y}$  (in other words, if  $\text{y}$  is a column vector, so is  $\text{y\_out}$  and if  $\text{y}$  is a row vector, so is  $\text{y\_out}$ ).
    - The function should “transpose” the vector if  $\text{x}$  and  $\text{y}$  do not have the same “shape” (if one is a column vector and the other one is a row vector).
    - If  $\text{x}$  and/or  $\text{y}$  are not vectors or if the size of (row or column) vector  $\text{x}$  does not match the size of (row or column) vector  $\text{y}$ , the output should be ‘FAILED’.

# LAFF Package Development: Vectors

- A Routine that Scales a Vector (scal)
  - Implement the function laff scal that scales a vector  $x$  by a scalar  $a$ . The function is defined as

```
function [ x_out ] = laff_scal( alpha, x )
```
  - where
    - $x$  must be either an  $n \times 1$  array (column vector) or a  $1 \times n$  array (row vector);
    - $x_{\text{out}}$  must be the same kind of vector as  $x$ ; and
    - If  $x$  or  $\alpha$  are not a (row or column) vector and scalar, respectively, the output should be 'FAILED'.

# LAFF Package Development: Vectors

- A Scaled Vector Addition Routine (axpy)
  - Implement the function `laff_axpy` that computes  $\alpha x + y$  given scalar  $\alpha$  and vectors  $x$  and  $y$ . The function is defined as
$$\text{function } [y\_out] = \text{laff\_axpy}(\text{alpha}, x, y)$$
  - where
    - $x$  and  $y$  must each be either an  $n \times 1$  array (column vector) or a  $1 \times n$  array (row vector);
    - $y_{\text{out}}$  must be the same kind of vector as  $y$ ; and
    - If  $x$  and/or  $y$  are not vectors or if the size of (row or column) vector  $x$  does not match the size of (row or column) vector  $y$ , the output should be 'FAILED'.
    - If  $\text{alpha}$  is not a scalar, the output should be 'FAILED'.

# LAFF Package Development: Vectors

- An Inner Product Routine (dot)
  - Implement the function `laff_dot` that computes the dot product of vectors  $x$  and  $y$ . The function is defined as

```
function [ alpha ] = laff_dot( x, y )
```
  - where
    - $x$  and  $y$  must each be either an  $n \times 1$  array (column vector) or a  $1 \times n$  array (row vector);
    - If  $x$  and/or  $y$  are not vectors or if the size of (row or column) vector  $x$  does not match the size of (row or column) vector  $y$ , the output should be 'FAILED'.

# LAFF Package Development: Vectors

- A Vector Length Routine (`norm2`)
  - Implement the function `laff_norm2` that computes the length of vector  $x$ . The function is defined as

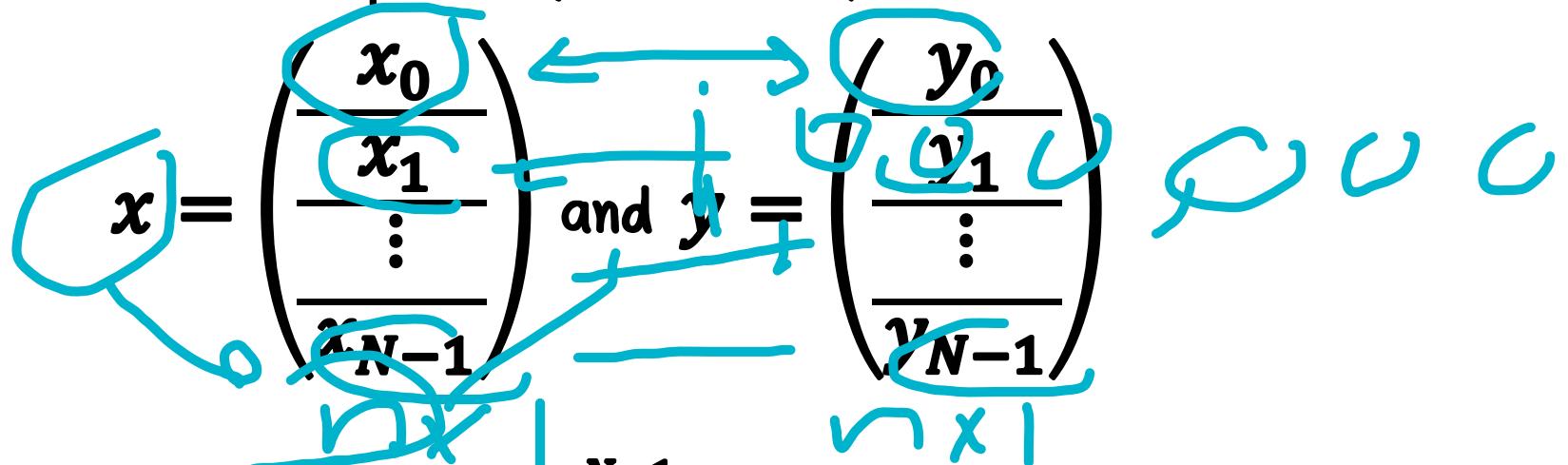
```
function [ alpha ] = laff_norm2( x )
```
  - where
    - $x$  is an  $n \times 1$  array (column vector) or a  $1 \times n$  array (row vector);
    - If  $x$  is not a vector the output should be 'FAILED'.

# Slicing and Dicing

- Slicing and Dicing: Dot Product

- Theorem

- Let  $x \in \mathbb{R}^n$  and partition (Slice and Dice) these vectors as



- Where  $x_i, y_i \in \mathbb{R}^{n_i}$  with  $\sum_{i=0}^{N-1} n_i = n$ . Then

$$x^T y = x_0^T y_0 + x_1^T y_1 + \dots + x_{N-1}^T y_{N-1} = \sum_{i=0}^{N-1} x_i^T y_i$$

# Slicing and Dicing

## • Algorithms with Slicing and Redicing: Dot Product

Algorithm:  $[\alpha] := \text{DOT}(x, y)$

Partition  $x \rightarrow \begin{pmatrix} x_T \\ x_B \end{pmatrix}$ ,  $y \rightarrow \begin{pmatrix} y_T \\ y_B \end{pmatrix}$

where  $x_T$  and  $y_T$  have 0 elements

$\alpha := 0$

while  $m(x_T) < m(x)$  do

Repartition

$$\begin{pmatrix} x_T \\ x_B \end{pmatrix} \rightarrow \begin{pmatrix} x_0 \\ \chi_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_T \\ y_B \end{pmatrix} \rightarrow \begin{pmatrix} y_0 \\ \psi_1 \\ y_2 \end{pmatrix}$$

where  $\chi_1$  has 1 row,  $\psi_1$  has 1 row

$\alpha := \chi_1 \times \psi_1 + \alpha$

Continue with

$$\begin{pmatrix} x_T \\ x_B \end{pmatrix} \leftarrow \begin{pmatrix} x_0 \\ \chi_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_T \\ y_B \end{pmatrix} \leftarrow \begin{pmatrix} y_0 \\ \psi_1 \\ y_2 \end{pmatrix}$$

endwhile

# Slicing and Dicing

- Coding with Slicing and Redicing: Dot Product
  - Follow along with the video to implement the routine  
`Dot_unb(x, y )`
  - The “Spark webpage” can be found at <http://edx-org-utaustinx.s3.amazonaws.com/UT501x/Spark/index.html>

# Slicing and Dicing

- Slicing and Dicing: axpy
  - Theorem
    - Let  $\alpha \in \mathbb{R}$ ,  $x, y \in \mathbb{R}^n$  and partition (Slice and Dice) these vectors as

$$x = \begin{pmatrix} \underline{x_0} \\ \underline{x_1} \\ \vdots \\ \underline{x_{N-1}} \end{pmatrix} \text{ and } y = \begin{pmatrix} \underline{y_0} \\ \underline{y_1} \\ \vdots \\ \underline{y_{N-1}} \end{pmatrix}$$

- Where  $x_i, y_i \in \mathbb{R}^{n'}$  with  $\sum_{i=0}^{N-1} n_i = n$ . Then

$$\alpha x + y = \alpha \begin{pmatrix} \underline{x_0} \\ \underline{x_1} \\ \vdots \\ \underline{x_{N-1}} \end{pmatrix} + \begin{pmatrix} \underline{y_0} \\ \underline{y_1} \\ \vdots \\ \underline{y_{N-1}} \end{pmatrix} = \begin{pmatrix} \underline{\alpha x_0 + y_0} \\ \underline{\alpha x_1 + y_1} \\ \vdots \\ \underline{\alpha x_{N-1} + y_{N-1}} \end{pmatrix}$$

# Slicing and Dicing

- Algorithms with Slicing and Redicing: axpy

Algorithm:  $[y] := \text{AXPY}(\alpha, x, y)$

Partition  $x \rightarrow \begin{pmatrix} x_T \\ x_B \end{pmatrix}$ ,  $y \rightarrow \begin{pmatrix} y_T \\ y_B \end{pmatrix}$

where  $x_T$  and  $y_T$  have 0 elements

while  $m(x_T) < m(x)$  do

Repartition

$$\begin{pmatrix} x_T \\ x_B \end{pmatrix} \rightarrow \begin{pmatrix} x_0 \\ \chi_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_T \\ y_B \end{pmatrix} \rightarrow \begin{pmatrix} y_0 \\ \Psi_1 \\ y_2 \end{pmatrix}$$

where  $\chi_1$  has 1 row,  $\Psi_1$  has 1 row

---

$$\Psi_1 := \alpha \times \chi_1 + \Psi_1$$

---

Continue with

$$\begin{pmatrix} x_T \\ x_B \end{pmatrix} \leftarrow \begin{pmatrix} x_0 \\ \chi_1 \\ x_2 \end{pmatrix}, \begin{pmatrix} y_T \\ y_B \end{pmatrix} \leftarrow \begin{pmatrix} y_0 \\ \Psi_1 \\ y_2 \end{pmatrix}$$

endwhile

# Slicing and Dicing

- Coding with Slicing and Redicing: axpy

- Implement the routine

Axpy\_unb( alpha, x, y ).

- The “Spark webpage” can be found at <http://edx-org-utaustinx.s3.amazonaws.com/UT501x/Spark/index.html>

# Enrichment

- Learn the Greek Alphabet

- Lowercase Greek letters ( $\alpha, \beta$ , etc.) are used for scalars.
- Lowercase (Roman) letters ( $a, b$ , etc) are used for vectors.
- Uppercase (Roman) letters ( $A, B$ , etc) are used for matrices.

Exceptions include the letters  $i, j, k, l, m$ , and  $n$ , which are typically used for integers.

# Enrichment

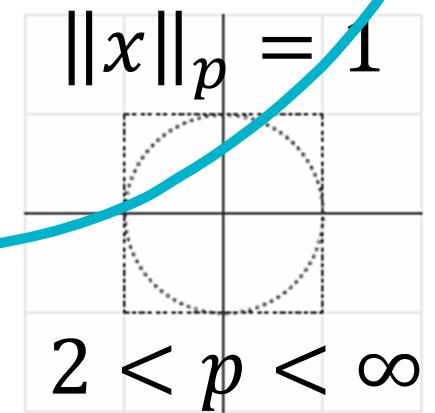
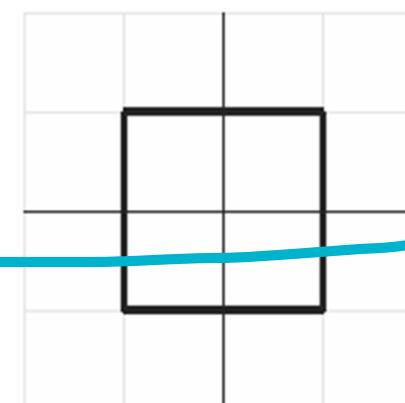
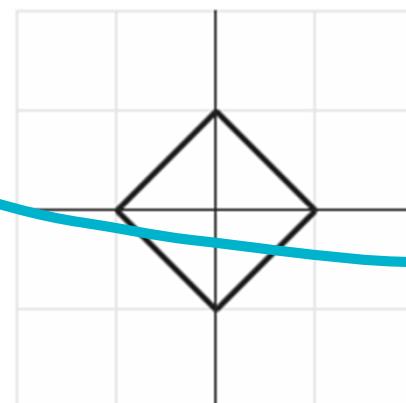
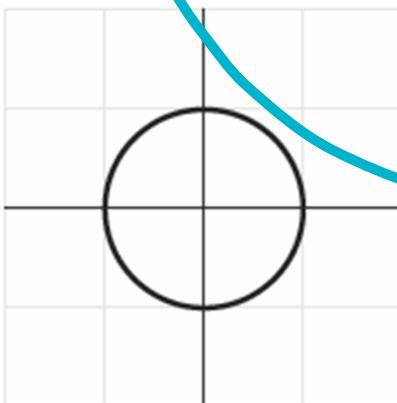
Matrix	Vector	Scalar			Note
		Symbol	\LaTeX	Code	
$A$	$a$	$\alpha$	\alpha	alpha	
$B$	$b$	$\beta$	\beta	beta	
$C$	$c$	$\gamma$	\gamma	gamma	
$D$	$d$	$\delta$	\delta	delta	
$E$	$e$	$\epsilon$	\epsilon	epsilon	$e_j = j$ th unit basis vector.
$F$	$f$	$\phi$	\phi	phi	
$G$	$g$	$\xi$	\xi	xi	
$H$	$h$	$\eta$	\eta	eta	
$I$					Used for identity matrix.
$K$	$k$	$\kappa$	\kappa	kappa	
$L$	$l$	$\lambda$	\lambda	lambda	
$M$	$m$	$\mu$	\mu	mu	$m(\cdot)$ = row dimension.
$N$	$n$	$\nu$	\nu	nu	$\nu$ is shared with V. $n(\cdot)$ = column dimension.
$P$	$p$	$\pi$	\pi	pi	
$Q$	$q$	$\theta$	\theta	theta	
$R$	$r$	$\rho$	\rho	rho	
$S$	$s$	$\sigma$	\sigma	sigma	
$T$	$t$	$\tau$	\tau	tau	
$U$	$u$	$\upsilon$	\upsilon	upsilon	
$V$	$v$	$\nu$	\nu	nu	$\nu$ shared with N.
$W$	$w$	$\omega$	\omega	omega	
$X$	$x$	$\chi$	\chi	chi	
$Y$	$y$	$\psi$	\psi	psi	
$Z$	$z$	$\zeta$	\zeta	zeta	

# Enrichment

- $|\alpha|$  : the magnitude of  $\alpha$   
$$|\alpha| = \begin{cases} \alpha & \text{if } \alpha \geq 0 \\ -\alpha & \text{otherwise} \end{cases}$$
- $\|x\|_2$  (2-norms) : the magnitude of vectors  $x$   
$$\|x\|_2 = \sqrt{\sum_{i=0}^{n-1} x_i^2}$$
- Function  $\|\cdot\|: \mathbb{R}^n \rightarrow \mathbb{R}$  is a norm if and only if the following properties hold for all  $x, y \in \mathbb{R}^n$ :
  - $\|x\| \geq 0$ ; and
  - $\|x\| = 0$  if and only if  $x = 0$ ; and
  - $\|x + y\| \leq \|x\| + \|y\|$  (the triangle inequality).

# Example

- The vectors with norm equal to one are often of special interest. Below we plot the points to which vectors  $x$  with  $\|x\|_2 = 1$  point (when those vectors start at the origin,  $(0,0)$ ). (E.g., the vector  $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$  points to the point  $(1,0)$  and that vector has 2-norm equal to one, hence the point is one of the points to be plotted.)
- Similarly, below we plot all points to which vectors  $x$  with  $\|x\|_1 = 1$  point (starting at the origin)
- Similarly, below we plot all points to which vectors  $x$  with  $\|x\|_\infty = 1$  point



# Questions and Answers

