Week 1

Vectors in Linear Algebra

1.1 Opening Remarks

1.1.1 Take Off

"Co-Pilot Roger Murdock (to Captain Clarence Oveur): We have clearance, Clarence.

Captain Oveur: Roger, Roger. What's our vector, Victor?"

From Airplane. Dir. David Zucker, Jim Abrahams, and Jerry Zucker. Perf. Robert Hays, Julie Hagerty, Leslie Nielsen, Robert Stack, Lloyd Bridges, Peter Graves, Kareem Abdul-Jabbar, and Lorna Patterson. Paramount Pictures, 1980. Film.

You can find a video clip by searching "What's our vector Victor?"

Vectors have direction and length. Vectors are commonly used in aviation where they are routinely provided by air traffic control to set the course of the plane, providing efficient paths that avoid weather and other aviation traffic as well as assist disoriented pilots.

Let's begin with vectors to set our course.

1.1.2 Outline Week 1

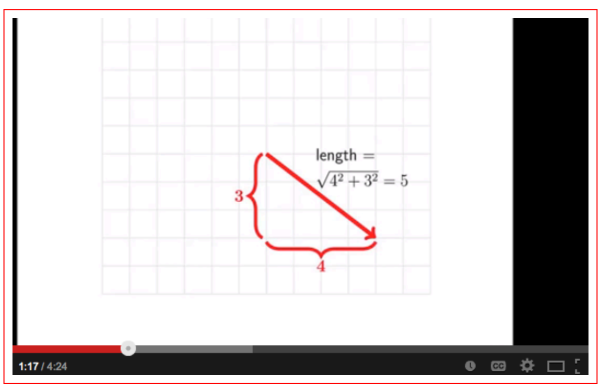
1.1.3 What You Will Learn

Upon completion of this unit, you should be able to

- Represent quantities that have a magnitude and a direction as vectors.
- Read, write, and interpret vector notations.
- Visualize vectors in $\mathbb{R}^2$.
- Perform the vector operations of scaling, addition, dot (inner) product.
- Reason and develop arguments about properties of vectors and operations defined on them.
- Compute the (Euclidean) length of a vector.
- Express the length of a vector in terms of the dot product of that vector with itself.
- Evaluate a vector function.
- Solve simple problems that can be represented with vectors.
- Create code for various vector operations and determine their cost functions in terms of the size of the vectors.
- Gain an awareness of how linear algebra software evolved over time and how our programming assignments fit into this (enrichment).
- Become aware of overflow and underflow in computer arithmetic (enrichment).

1.2 What is a Vector?

1.2.1 Notation



Definition

Definition 1.1 *We will call a one-dimensional array of n numbers a vector of size n:*

$$x = \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix}.$$

- This is an *ordered* array. The position in the array is important.

- We will call the ith number the ith *component* or *element*.

- We denote the ith component of x by χ_i. Here χ is the lower case Greek letter pronounced as "kı". (Learn more about our notational conventions in Section 1.7.1.)

  As a rule, we will use lower case letters to name vectors (e.g., $x, y, ...$). The "corresponding" Greek lower case letters are used to name their components.

- We start indexing at 0, as computer scientists do. Python, the language we will be using to implement our libraries, naturally starts indexing at 0 as well. Mathematicians and most physical scientists sometimes start indexing at 1, but we will not.

- Each number is, at least for now, a real number, which in math notation is written as $\chi_i \in \mathbb{R}$ (read: "ki sub i (is) in r" or "ki sub i is an element of the set of all real numbers").

- The *size* of the vector is n, the number of components. (Sometimes, people use the words "length" and "size" interchangeably. We will see that length also has another meaning and will try to be consistent.)

- We will write $x \in \mathbb{R}^n$ (read: "x" in "r" "n") to denote that x is a vector of size n with components in the real numbers, denoted by the symbol: $\mathbb{R}$. Thus, $\mathbb{R}^n$ denotes the set of all vectors of size n with components in $\mathbb{R}$. (Later we will talk about vectors with components that are complex valued.)

- A vector has a direction and a length:

  – Its direction is often visualized by drawing an arrow from the origin to the point $(\chi_0, \chi_1, \ldots, \chi_{n-1})$, but the arrow does not necessarily need to start at the origin.

  – Its *length* is given by the Euclidean length of this arrow,

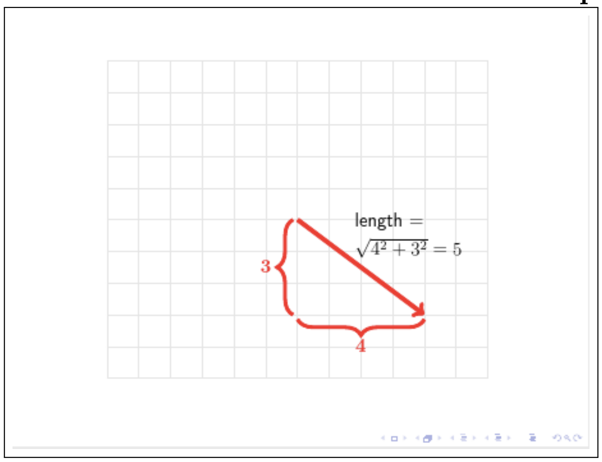
  $$\sqrt{\chi_0^2 + \chi_1^2 + \cdots + \chi_{n-1}^2},$$

  It is denoted by $\|x\|_2$ called the *two-norm*. Some people also call this the *magnitude* of the vector.

- A vector does *not* have a location. Sometimes we will show it starting at the origin, but that is only for convenience. It will often be more convenient to locate it elsewhere or to move it.

Examples

Example 1.2

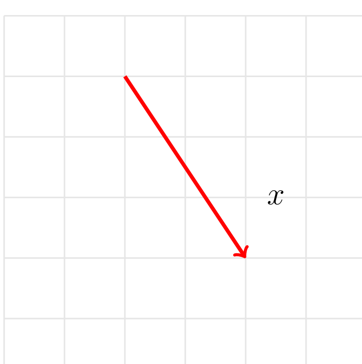


Consider $x = \begin{pmatrix} 4 \\ -3 \end{pmatrix}$. Then

- Components 4 and -3 are the first and second component, respectively.
- $\chi_0 = 4$, $\chi_1 = -3$ so that 4 is the component indexed with 0 and -3 the component indexed with 1.
- The vector is of size 2, so $x \in \mathbb{R}^2$.

Exercises

Homework 1.2.1.1 Consider the following picture:



(a) $x = \begin{pmatrix} -2 \\ -3 \end{pmatrix}$

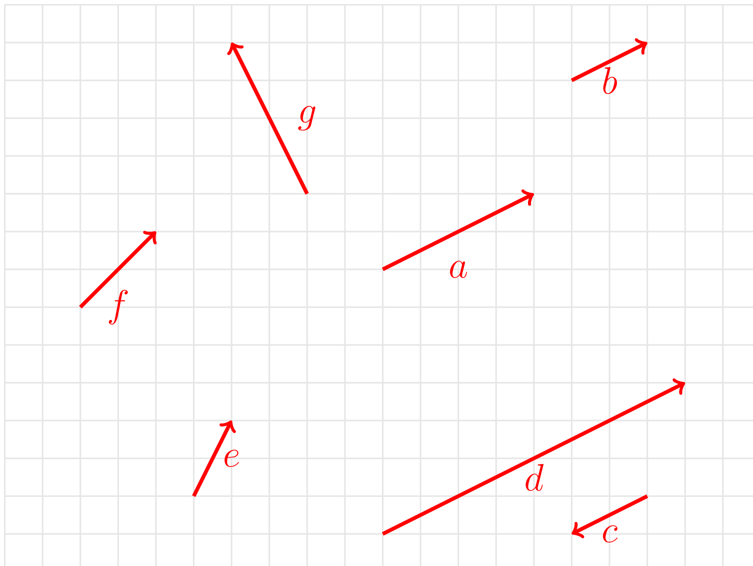
(b) $x = \begin{pmatrix} 3 \\ -2 \end{pmatrix}$

(c) $x = \begin{pmatrix} 2 \\ -3 \end{pmatrix}$

(d) $x = \begin{pmatrix} -3 \\ -2 \end{pmatrix}$

(e) None of the above

Homework 1.2.1.2 Consider the following picture:



(a) $a = \begin{pmatrix} \\ \\ \end{pmatrix}$ (b) $b = \begin{pmatrix} \\ \\ \end{pmatrix}$ (c) $c = \begin{pmatrix} \\ \\ \end{pmatrix}$ (d) $d = \begin{pmatrix} \\ \\ \end{pmatrix}$

(e) $e = \begin{pmatrix} \\ \\ \end{pmatrix}$ (f) $f = \begin{pmatrix} \\ \\ \end{pmatrix}$ (g) $g = \begin{pmatrix} \\ \\ \end{pmatrix}$

While a vector does not have a location, but has direction and length, vectors are often used to show the direction and length of movement from one location to another. For example, the vector from point $(1, -2)$ to point $(5, 1)$ is the vector $\begin{pmatrix} 4 \\ 3 \end{pmatrix}$. We might geometrically represent the vector $\begin{pmatrix} 4 \\ 3 \end{pmatrix}$ by an arrow from point $(1, -2)$ to point $(5, 1)$.

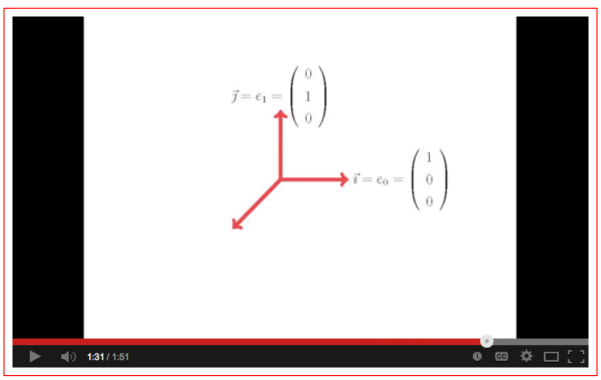
Homework 1.2.1.3 The vector represented geometrically in $\mathbb{R}^2$ by

- an arrow from point $(-1, 2)$ to point $(0, 0)$ can be written as $x = \begin{pmatrix} ? \\ ? \end{pmatrix}$.
- an arrow from point $(0, 0)$ to point $(-1, 2)$ can be written as $x = \begin{pmatrix} ? \\ ? \end{pmatrix}$.

The vector represented geometrically in $\mathbb{R}^3$ by

- an arrow from point $(-1, 2, 4)$ to point $(0, 0, 1)$ can be written as $x = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix}$.
- an arrow from point $(1, 0, 0)$ to point $(4, 2, -1)$ can be written as $x = \begin{pmatrix} ? \\ ? \\ ? \end{pmatrix}$.

1.2.2 Unit Basis Vectors



Definition

Definition 1.3 *An important set of vectors is the set of unit basis vectors given by*

$$e_j = \left(\begin{array}{c} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{array}\right) \begin{array}{l} \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} j \text{ zeroes} \\ \longleftarrow \text{ component indexed by } j \\ \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} n-j-1 \text{ zeroes} \end{array}$$

where the "1" appears as the component indexed by j. Thus, we get the set $\{e_0, e_1, \ldots, e_{n-1}\} \subset \mathbb{R}^n$ given by

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

In our presentations, any time you encounter the symbol e_j, it *always* refers to the unit basis vector with the "1" in the component indexed by j.

These vectors are also referred to as the **standard basis vectors**. Other terms used for these vectors are natural basis and canonical basis. Indeed, "unit basis vector" appears to be less commonly used. But we will use it anyway!

Homework 1.2.2.1 Which of the following is not a unit basis vector?

(a) $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$

(b) $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$

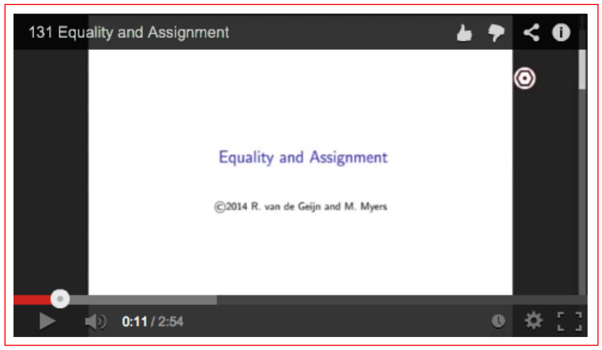
(c) $\begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}$

(d) $\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$

(e) None of the above.

1.3 Simple Vector Operations

1.3.1 Equality (=), Assignment (:=), and Copy



Definition

Definition 1.4 *Two vectors $x, y \in \mathbb{R}^n$ are equal if all their components are element-wise equal:*

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

This means that two vectors are equal if they point in the same direction and are of the same length. They don't, however, need to have the same location.

The *assignment* or *copy* operation assigns the content of one vector to another vector. In our mathematical notation, we will denote this by the symbol $:=$ (pronounce: *becomes*). After the assignment, the two vectors are equal to each other.

Algorithm

The following algorithm copies vector $x \in \mathbb{R}^n$ into vector $y \in \mathbb{R}^n$, performing the operation $y := x$:

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

for $i = 0, \ldots, n-1$
  $\psi_i := \chi_i$
endfor

Cost

(Notice: we will cost of various operations in more detail in the future.)

Copying one vector to another vector requires $2n$ memory operations (memops).

- The vector x of length n must be read, requiring n memops and
- the vector y must be written, which accounts for the other n memops.

Homework 1.3.1.1 Decide if the two vectors are equal.

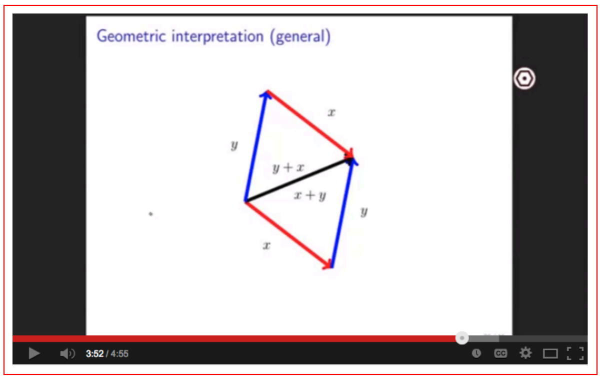
- The vector represented geometrically in $\mathbb{R}^2$ by an arrow from point $(-1, 2)$ to point $(0, 0)$ and the vector represented geometrically in $\mathbb{R}^2$ by an arrow from point $(1, -2)$ to point $(2, -1)$ are equal.

  True/False

- 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.

  True/False

1.3.2 Vector Addition (ADD)



Definition

Definition 1.5 *Vector addition $x + y$ (sum of vectors) is defined by*

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

In other words, the vectors are added element-wise, yielding a new vector of the same size.

Exercises

Homework 1.3.2.1 $\begin{pmatrix} -1 \\ 2 \end{pmatrix} + \begin{pmatrix} -3 \\ -2 \end{pmatrix} =$

Homework 1.3.2.2 $\begin{pmatrix} -3 \\ -2 \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \end{pmatrix} =$

Homework 1.3.2.3 For $x, y \in \mathbb{R}^n$,

$$x + y = y + x.$$

Always/Sometimes/Never

Homework 1.3.2.4 $\begin{pmatrix} -1 \\ 2 \end{pmatrix} + \left(\begin{pmatrix} -3 \\ -2 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right) =$

Homework 1.3.2.5 $\left(\begin{pmatrix} -1 \\ 2 \end{pmatrix} + \begin{pmatrix} -3 \\ -2 \end{pmatrix} \right) + \begin{pmatrix} 1 \\ 2 \end{pmatrix} =$

Homework 1.3.2.6 For $x, y, z \in \mathbb{R}^n$,

$$(x + y) + z = x + (y + z).$$

Always/Sometimes/Never

Homework 1.3.2.7 $\begin{pmatrix} -1 \\ 2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix} =$

Homework 1.3.2.8 For $x \in \mathbb{R}^n$,

$$x + 0 = x,$$

where 0 is the zero vector of appropriate size.

Always/Sometimes/Never

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 $z := x + y$:

$$\begin{pmatrix} \zeta_0 \\ \zeta_1 \\ \vdots \\ \zeta_{n-1} \end{pmatrix} := \begin{pmatrix} \chi_0 + \psi_0 \\ \chi_1 + \psi_1 \\ \vdots \\ \chi_{n-1} + \psi_{n-1} \end{pmatrix}.$$

for $i = 0, \ldots, n-1$

  $\zeta_i := \chi_i + \psi_i$

endfor

Cost

On a computer, real numbers are stored as floating point numbers, and real arithmetic is approximated with floating point arithmetic. Thus, we count floating point operations (flops): a multiplication or addition each cost one flop.

Vector addition requires $3n$ memops (x is read, y is read, and the resulting vector is written) and n flops (floating point additions).

For those who understand “Big-O” notation, the cost of the SCAL operation, which is seen in the next section, is $O(n)$. However, we tend to want to be more exact than just saying $O(n)$. To us, the coefficient in front of n is important.

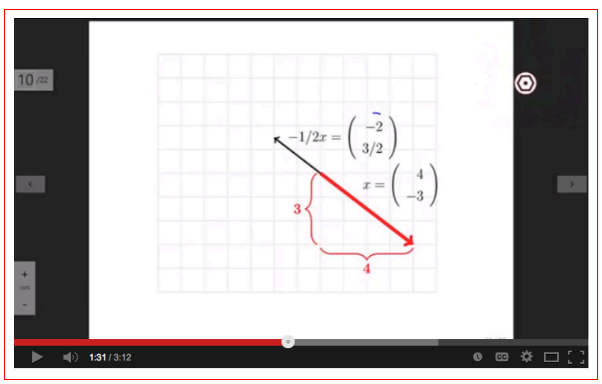
Vector addition in sports

View the following video and find out how the “parallelogram method” for vector addition is useful in sports:

http://www.scientificamerican.com/article.cfm?id=football-vectors

> Discussion: Can you find other examples of how vector addition is used in sports?

1.3.3 Scaling (SCAL)



Definition

Definition 1.6 *Multiplying vector x by scalar α yields a new vector, αx, in the same direction as x, but scaled by a factor α. Scaling a vector by α means each of its components, χ_i, is multiplied by α:*

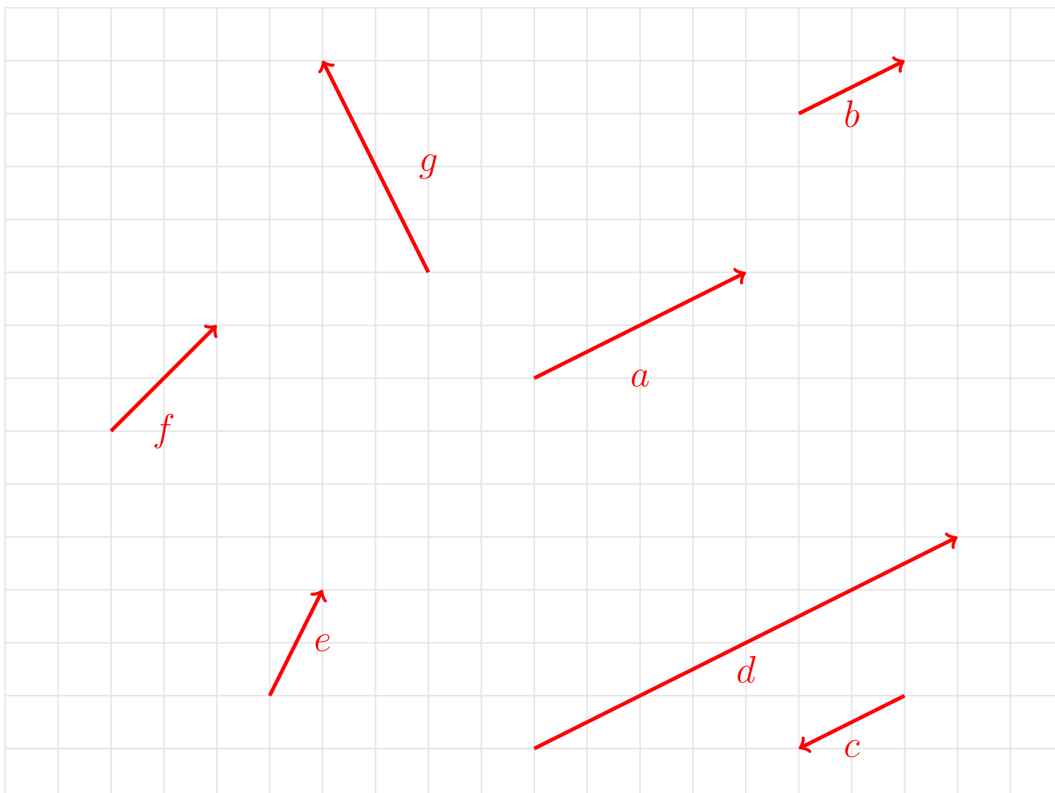
$$\alpha x = \alpha \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix} = \begin{pmatrix} \alpha\chi_0 \\ \alpha\chi_1 \\ \vdots \\ \alpha\chi_{n-1} \end{pmatrix}.$$

Exercises

Homework 1.3.3.1 $\left(\begin{pmatrix} -1 \\ 2 \end{pmatrix} + \begin{pmatrix} -1 \\ 2 \end{pmatrix} \right) + \begin{pmatrix} -1 \\ 2 \end{pmatrix} =$

Homework 1.3.3.2 $3 \begin{pmatrix} -1 \\ 2 \end{pmatrix} =$

Homework 1.3.3.3 Consider the following picture:



Which vector equals $2a$?; $1/2a$? ; and $-1/2a$?

Algorithm

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

$$\begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix} := \begin{pmatrix} \alpha\chi_0 \\ \alpha\chi_1 \\ \vdots \\ \alpha\chi_{n-1} \end{pmatrix}.$$

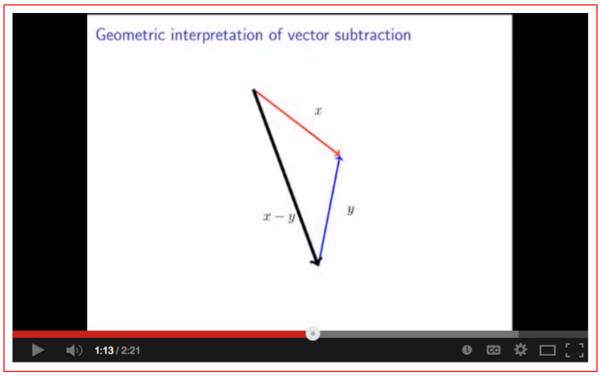
for $i = 0, \ldots, n-1$
    $\chi_i := \alpha \chi_i$
endfor

Cost

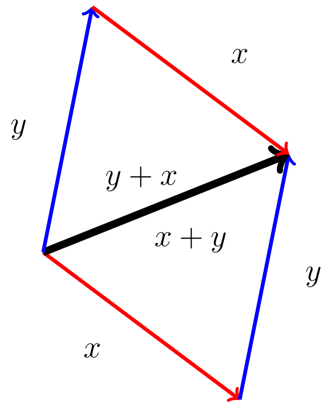
Scaling a vector requires n flops and $2n+1$ memops. Here, α is only brought in from memory once and kept in a register for reuse. To fully understand this, you need to know a little bit about computer architecture.

"Among friends" we will simply say that the cost is $2n$ memops since the one extra memory operation (to bring α in from memory) is negligible.

1.3.4 Vector Subtraction



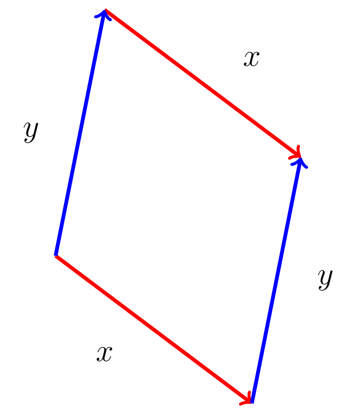
Recall the geometric interpretation for adding two vectors, $x, y \in \mathbb{R}^n$:



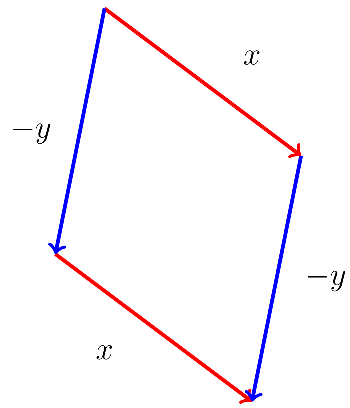
Subtracting y from x is defined as

$$x - y = x + (-y).$$

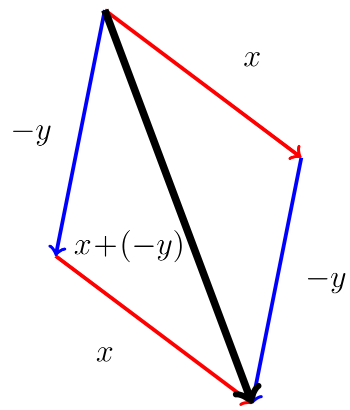
We learned in the last unit that $-y$ is the same as $(-1)y$ which is the same as pointing y in the opposite direction, while keeping it's length the same. This allows us to take the parallelogram that we used to illustrate vector addition



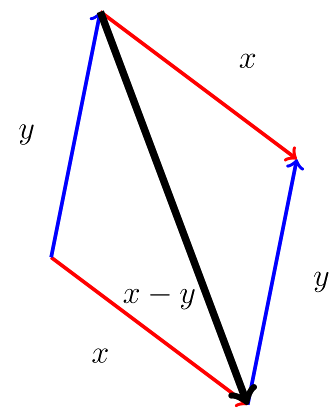
and change it into the equivalent picture



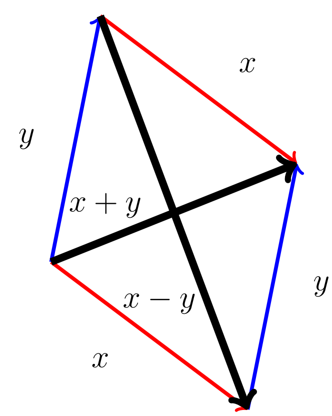
Since we know how to add two vectors, we can now illustrate $x + (-y)$:



Which then means that $x - y$ can be illustrated by



Finally, we note that the parallelogram can be used to simulaneously illustrate vector addition and subtraction:



(Obviously, you need to be careful to point the vectors in the right direction.)

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:

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

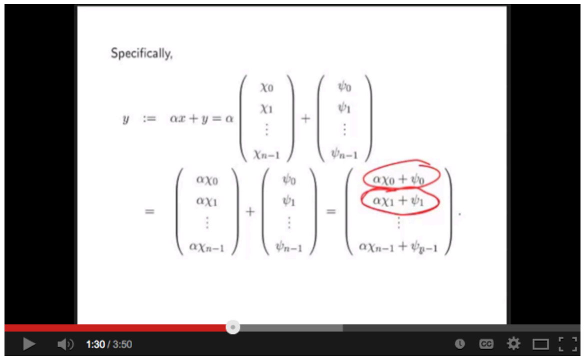
Homework 1.3.4.1 For $x \in \mathbb{R}^n$,

$$x - x = 0.$$

Always/Sometimes/Never

1.4 Advanced Vector Operations

1.4.1 Scaled Vector Addition (AXPY)



Definition

Definition 1.7 *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 $y := \alpha x + y$:*

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

It is often referred to as the AXPY operation, which stands for **a**lpha times **x** **p**lus **y**. We emphasize that it is typically used in situations where the output vector overwrites the input vector y.

Algorithm

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

$$\begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} := \begin{pmatrix} \alpha\chi_0 + \psi_0 \\ \alpha\chi_1 + \psi_1 \\ \vdots \\ \alpha\chi_{n-1} + \psi_{n-1} \end{pmatrix}.$$

for $i = 0, \ldots, n-1$

  $\psi_i := \alpha\chi_i + \psi_i$

endfor

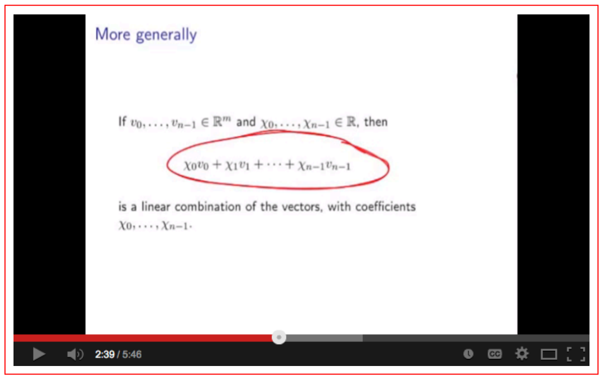
Cost

In Section 1.3 for many of the operations we discuss the cost in terms of memory operations (memops) and floating point operations (flops). This is discussed in the text, but not the videos. The reason for this is that we will talk about the cost of various operations later in a larger context, and include these discussions here more for completely.

Homework 1.4.1.1 What is the cost of an axpy operation?

- How many memops?
- How many flops?

1.4.2 Linear Combinations of Vectors



Discussion

There are few concepts in linear algebra more fundamental than linear combination of vectors.

Definition

Definition 1.8 *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:*

$$\alpha u + \beta v = \alpha \begin{pmatrix} \upsilon_0 \\ \upsilon_1 \\ \vdots \\ \upsilon_{m-1} \end{pmatrix} + \beta \begin{pmatrix} \nu_0 \\ \nu_1 \\ \vdots \\ \nu_{m-1} \end{pmatrix} = \begin{pmatrix} \alpha\upsilon_0 \\ \alpha\upsilon_1 \\ \vdots \\ \alpha\upsilon_{m-1} \end{pmatrix} + \begin{pmatrix} \beta\nu_0 \\ \beta\nu_1 \\ \vdots \\ \beta\nu_{m-1} \end{pmatrix} = \begin{pmatrix} \alpha\upsilon_0 + \beta\nu_0 \\ \alpha\upsilon_1 + \beta\nu_1 \\ \vdots \\ \alpha\upsilon_{m-1} + \beta\nu_{m-1} \end{pmatrix}.$$

The scalars α and β are the coefficients used in the linear combination.

More generally, if $v_0, \ldots, v_{n-1} \in \mathbb{R}^m$ are n vectors and $\chi_0, \ldots, \chi_{n-1} \in \mathbb{R}$ are n scalars, then $\chi_0 v_0 + \chi_1 v_1 + \cdots + \chi_{n-1} v_{n-1}$ is a linear combination of the vectors, with coefficients $\chi_0, \ldots, \chi_{n-1}$.

We will often use the summation notation to more concisely write such a linear combination:

$$\chi_0 v_0 + \chi_1 v_1 + \cdots + \chi_{n-1} v_{n-1} = \sum_{j=0}^{n-1} \chi_j v_j.$$

Homework 1.4.2.1

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

Homework 1.4.2.2

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

Homework 1.4.2.3 Find α, β, γ such that

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

$\alpha =$ $\beta =$ $\gamma =$

Algorithm

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

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

The axpy operation computed $y := \alpha x + y$. In our algorithm, χ_j takes the place of α, v_j the place of x, and w the place of y.

Cost

We noted that computing $w = \chi_0 v_0 + \chi_1 v_1 + \cdots \chi_{n-1} v_{n-1}$ can be implementated as n AXPY operations. This suggests that the cost is n times the cost of an AXPY operation with vectors of size m: $n \times (2m) = 2mn$ flops and (approximately) $n \times (3m)$ memops.

However, one can actually do better. The vector w is updated repeatedly. If this vector stays in the L1 cache of a computer, then it needs not be repeatedly loaded from memory, and the cost becomes m memops (to load w into the cache) and then for each AXPY operation (approximately) m memops (to read v_j (ignoring the cost of reading χ_j). Then, once w has been completely updated, it can be written back to memory. So, the total cost related to accessing memory becomes $m + n \times m + m = (n+2)m \approx mn$ memops.

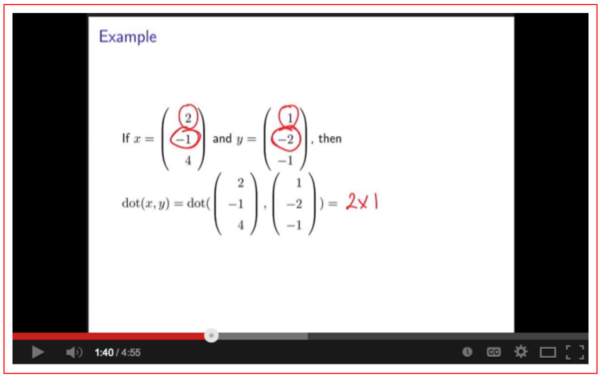
An important example

Example 1.9 Given any $x \in \mathbb{R}^n$ with $x = \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix}$, this vector can always be written as the linear combination of the unit basis vectors given by

$$
\begin{aligned}
x &= \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix} = \chi_0 \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} + \chi_1 \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix} + \cdots + \chi_{n-1} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \\
&= \chi_0 e_0 + \chi_1 e_1 + \cdots + \chi_{n-1} e_{n-1} = \sum_{i=0}^{n-1} \chi_i e_i.
\end{aligned}
$$

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

1.4.3 Dot or Inner Product (DOT)



Definition

The other commonly encountered operation is the dot (inner) product. It is defined by

$$\mathrm{dot}(x, y) = \sum_{i=0}^{n-1} \chi_i \psi_i = \chi_0 \psi_0 + \chi_1 \psi_1 + \cdots + \chi_{n-1} \psi_{n-1}.$$

Alternative notation

We will often write

$$\begin{aligned} x^T y &= \text{dot}(x, y) = \begin{pmatrix} \chi_0 \\ \chi_1 \\ \vdots \\ \chi_{n-1} \end{pmatrix}^T \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} \\ &= \begin{pmatrix} \chi_0 & \chi_1 & \cdots & \chi_{n-1} \end{pmatrix} \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{pmatrix} \\ &= \chi_0\psi_0 + \chi_1\psi_1 + \cdots + \chi_{n-1}\psi_{n-1} \end{aligned}$$

for reasons that will become clear later in the course.

Exercises

Homework 1.4.3.1 $\begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix}^T \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} =$

Homework 1.4.3.2 $\begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix}^T \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} =$

Homework 1.4.3.3 $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix} =$

Homework 1.4.3.4 For $x, y \in \mathbb{R}^n$,

$$x^T y = y^T x.$$

Always/Sometimes/Never

Homework 1.4.3.5 $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}^T \left(\begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \right) =$

Homework 1.4.3.6 $\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}^T \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} =$

Homework 1.4.3.7 $\left(\begin{pmatrix} 2 \\ 5 \\ -6 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \right)^T \begin{pmatrix} 1 \\ 0 \\ 0 \\ 2 \end{pmatrix} =$

Homework 1.4.3.8 For $x, y, z \in \mathbb{R}^n$,

$$x^T (y + z) = x^T y + x^T z.$$

Always/Sometimes/Never

Homework 1.4.3.9 For $x, y, z \in \mathbb{R}^n$,

$$(x + y)^T z = x^T z + y^T z.$$

Always/Sometimes/Never

Homework 1.4.3.10 For $x, y \in \mathbb{R}^n$,

$$(x + y)^T (x + y) = x^T x + 2x^T y + y^T y.$$

Always/Sometimes/Never

Homework 1.4.3.11 Let $x, y \in \mathbb{R}^n$. When $x^T y = 0$, x or y is a zero vector.

Always/Sometimes/Never

Homework 1.4.3.12 For $x \in \mathbb{R}^n$,

$$e_i^T x = x^T e_i = \chi_i,$$

where χ_i equals the ith component of x.

Always/Sometimes/Never

Algorithm

An algorithm for the DOT operation is given by

$\alpha := 0$

for $i = 0, \ldots, n-1$

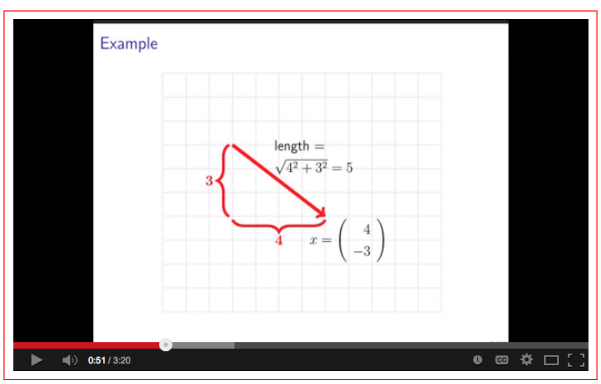
  $\alpha := \chi_i \psi_i + \alpha$

endfor

Cost

Homework 1.4.3.13 What is the cost of a dot product with vectors of size n?

1.4.4 Vector Length (NORM2)



Definition

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

$$\|x\|_2 = \sqrt{\chi_0^2 + \chi_1^2 + \cdots + \chi_{n-1}^2} = \sqrt{\sum_{i=0}^{n-1} \chi_i^2}.$$

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

Homework 1.4.4.1 Compute the lengths of the following vectors:

$$\text{(a)} \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}; \text{(b)} \begin{pmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{pmatrix}; \text{(c)} \begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix}; \text{(d)} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Homework 1.4.4.2

$$\|x\|_2 < 0$$

Always/Sometimes/Never

Homework 1.4.4.3 If x is a unit vector then x is a unit basis vector.

TRUE/FALSE

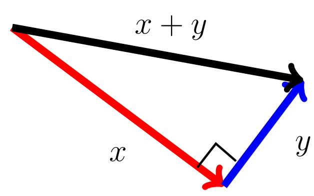
Homework 1.4.4.4 If x is a unit basis vector then x is a unit vector.

TRUE/FALSE

Homework 1.4.4.5 If x and y are perpendicular (orthogonal) then $x^T y = 0$.

TRUE/FALSE

Hint: Consider the picture

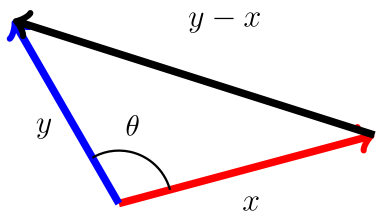


Homework 1.4.4.6 Let $x, y \in \mathbb{R}^n$ be nonzero vectors and let the angle between them equal θ. Then

$$\cos \theta = \frac{x^T y}{\|x\|_2 \|y\|_2}.$$

Always/Sometimes/Never

Hint: Consider the picture and the "Law of Cosines" that you learned in high school. (Or look up this law!)



Homework 1.4.4.7 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

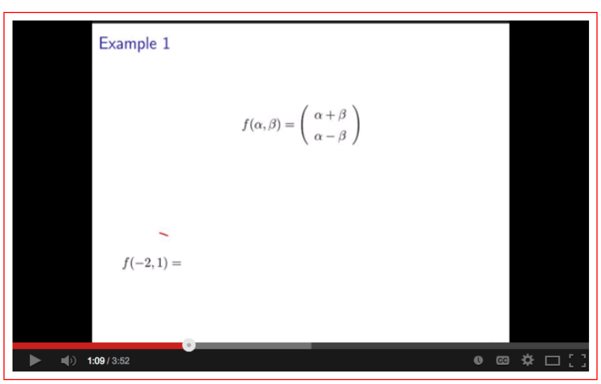
Algorithm

Clearly, $\|x\|_2 = \sqrt{x^T x}$, so that the DOT operation can be used to compute this length.

Cost

If computed with a dot product, it requires approximately n memops and $2n$ flops.

1.4.5 Vector Functions



Last week, we saw a number of examples where a function, f, takes in one or more scalars and/or vectors, and outputs a vector (where a scalar can be thought of as a special case of a vector, with unit size). These are all examples of vector-valued functions (or vector functions for short).

Definition

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

Examples

Example 1.10

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

Example 1.11

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

Example 1.12 *The* AXPY *and* DOT *vector functions are other functions that we have already encountered.*

Example 1.13

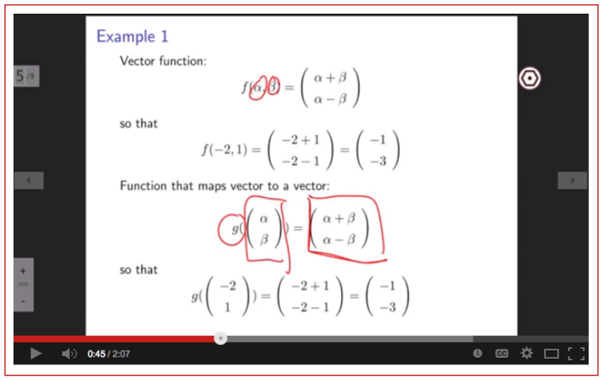
$$f(\alpha, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}) = \begin{pmatrix} \chi_0+\chi_1 \\ \chi_1+\chi_2 \end{pmatrix} \quad \text{so that} \quad f(\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}) = \begin{pmatrix} 1+2 \\ 2+3 \end{pmatrix} = \begin{pmatrix} 3 \\ 5 \end{pmatrix}.$$

Practice

Try this! If $f(\alpha, \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}) = \begin{pmatrix} \chi_0 + \alpha \\ \chi_1 + \alpha \\ \chi_2 + \alpha \end{pmatrix}$, find

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

1.4.6 Vector Functions that Map a Vector to a Vector



Now, we can talk about such functions in general as being a function from one vector to another vector. After all, we can take all inputs, make one vector with the separate inputs as the elements or subvectors of that vector, and make that the input for a new function that has the same net effect.

Example 1.14 *Instead of*

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

we can define

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

Example 1.15 *Instead of*

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

we can define

$$g(\begin{pmatrix} \alpha \\ \begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix} \end{pmatrix}) = g(\begin{pmatrix} \alpha \\ \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}) = \begin{pmatrix} \chi_0 + \alpha \\ \chi_1 + \alpha \\ \chi_2 + \alpha \end{pmatrix} \quad \text{so that} \quad g(\begin{pmatrix} -2 \\ 1 \\ 2 \\ 3 \end{pmatrix}) = \begin{pmatrix} 1 + (-2) \\ 2 + (-2) \\ 3 + (-2) \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}.$$

The bottom line is that we can focus on vector functions that map a vector of size n into a vector of size m, which is written as

$$f : \mathbb{R}^n \to \mathbb{R}^m.$$

Exercises

Homework 1.4.6.1 If $f(\begin{pmatrix} \chi_0 \\ \chi_1 \\ \chi_2 \end{pmatrix}) = \begin{pmatrix} \chi_0 + 1 \\ \chi_1 + 2 \\ \chi_2 + 3 \end{pmatrix}$, evaluate

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

Homework 1.4.6.2 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

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

Homework 1.4.6.3 If $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, then

$$f(0) = 0.$$

Always/Sometimes/Never

Homework 1.4.6.4 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/Sometimes/Never

Homework 1.4.6.5 If $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and $x, y \in \mathbb{R}^n$, then

$$f(x + y) = f(x) + f(y).$$

Always/Sometimes/Never

1.5 LAFF Package Development: Vectors

1.5.1 Starting the Package

In this course, we will explore and use a rudimentary dense linear algebra software library. The hope is that by linking the abstractions in linear algebra to abstractions (functions) in software, a deeper understanding of the material will be the result.

We have chosen to use Python as our language. However, the resulting code can be easily translated into other languages. For example, as part of our FLAME research project we developed a library called `libflame`. Even though we coded it in the C programming language, it still closely resembles the Python code that you will write and the library that you will use.

You learn how to implement your code with iPython notebooks. In "Week 0" of this course, you installed all the software needed to use iPython notebooks. This provides an interactive environment in which we can intersperse text with executable code.

A library of vector-vector routines

The functionality of the Python functions you will write is also part of the "laff" library of routines. What this means will become obvious in subsequent units.

Here is a table of all the vector functions, and the routines that implement them, that you will be able to use in future weeks is given in the following table:

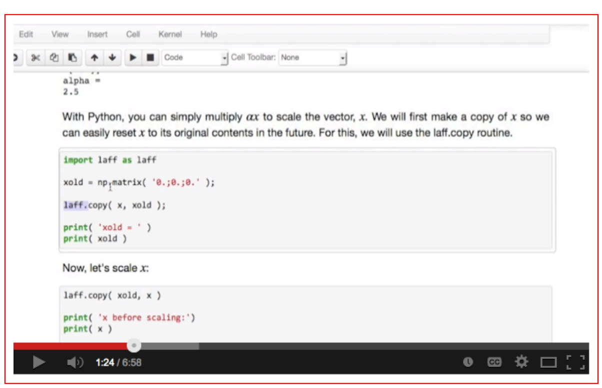
Operation Abbrev.	Definition	Function	Approx. cost	
			flops	memops
Vector-vector operations				
Copy (COPY)	$y := x$	`laff.copy( x, y )`	0	$2n$
Vector scaling (SCAL)	$x := \alpha x$	`laff.scal( alpha, x )`	n	$2n$
Scaled addition (AXPY)	$y := \alpha x + y$	`laff.axpy( alpha, x, y )`	$2n$	$3n$
Dot product (DOT)	$\alpha := x^T y$	`alpha = laff.dot( x, y )`	$2n$	$2n$
Length (NORM2)	$\alpha := \|x\|_2$	`alpha = laff.norm2( x )`	$2n$	n

Next, let's dive right in! We'll walk you through it in the next units.

1.5.2 A Copy Routine (copy)

> **Homework 1.5.2.6** Do the IPython Notebook exercise
>
> 1.5.2 Implementing a copy routine
>
> as described in the video for this unit.

1.5.3 A Routine that Scales a Vector (scal)

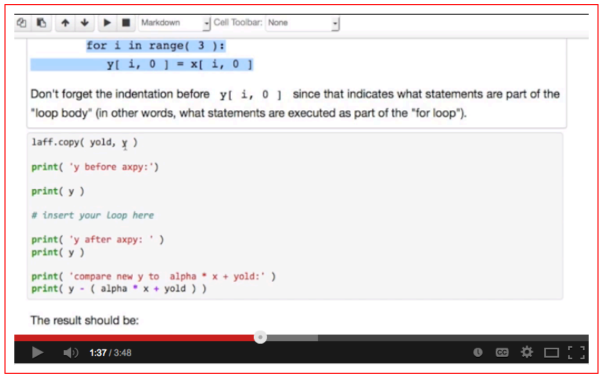


Homework 1.5.3.7 Do the IPython Notebook exercise

1.5.3 Implementing a routine that scales a vector

as described in the video for this unit.

1.5.4 A Scaled Vector Addition Routine (axpy)

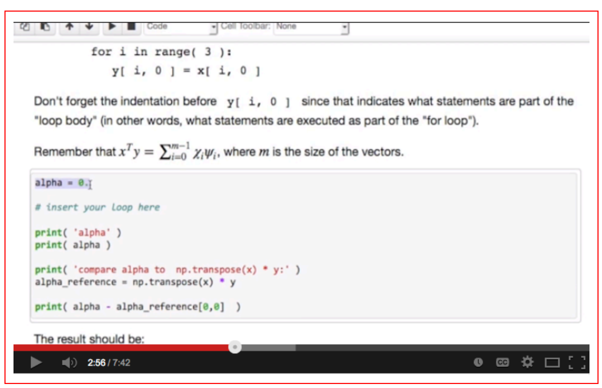


Homework 1.5.4.8 Do the IPython Notebook exercise

1.5.4 Implementing an axpy routine

as described in the video for this unit.

1.5.5 An Inner Product Routine (dot)

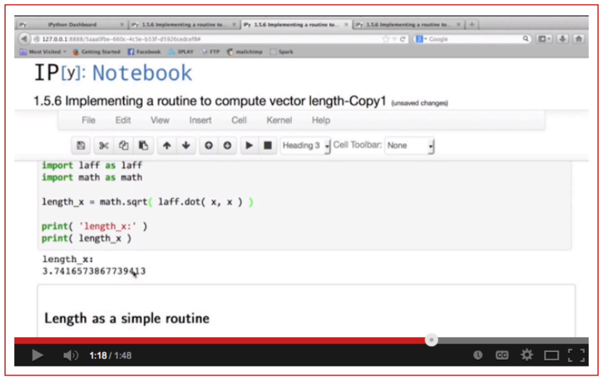


Homework 1.5.5.9 Do the IPython Notebook exercise

1.5.5 Implementing a dot routine

as described in the video for this unit.

1.5.6 A Vector Length Routine (norm2)



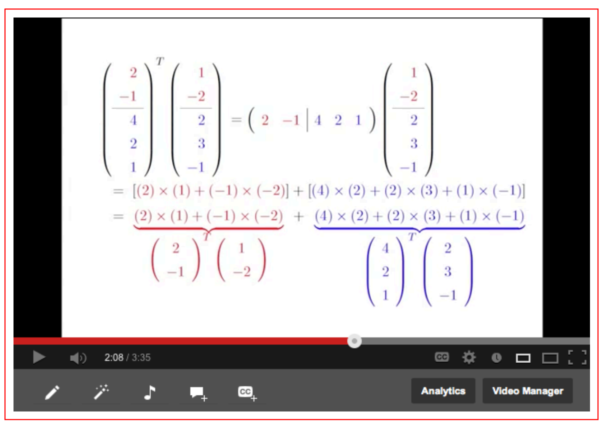
Homework 1.5.6.10 Do the IPython Notebook exercise

1.5.6 Implementing a routine to compute vector length

as described in the video for this unit.

1.6 Slicing and Dicing

1.6.1 Slicing and Dicing: Dot Product



In the video, we justify the following theorem:

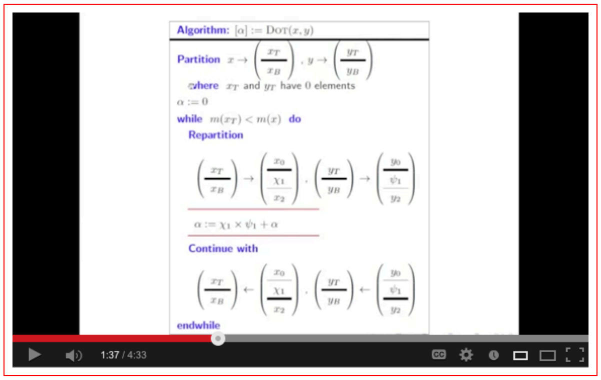
Theorem 1.16 *Let $x, y \in \mathbb{R}^n$ and partition (Slice and Dice) these vectors as*

$$x = \left(\begin{array}{c} x_0 \\ \hline x_1 \\ \hline \vdots \\ \hline x_{N-1} \end{array}\right) \quad \text{and} \quad y = \left(\begin{array}{c} y_0 \\ \hline y_1 \\ \hline \vdots \\ \hline y_{N-1} \end{array}\right),$$

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

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

1.6.2 Algorithms with Slicing and Redicing: Dot Product



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

Partition $x \rightarrow \left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right)$, $y \rightarrow \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right)$

  where x_T and y_T have 0 elements

$\alpha := 0$

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

  Repartition

$$\left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right) \rightarrow \left(\begin{array}{c} x_0 \\ \hline \chi_1 \\ x_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ y_2 \end{array}\right)$$

  where χ_1 has 1 row, ψ_1 has 1 row

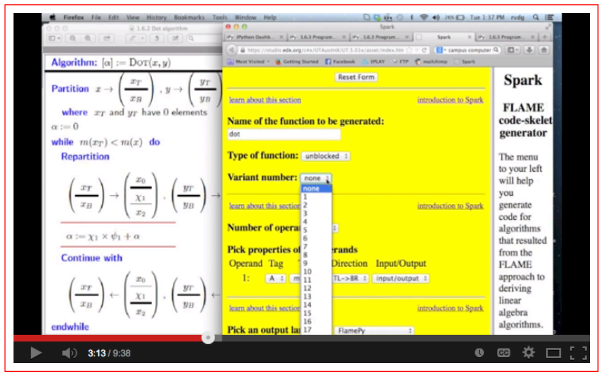
$\alpha := \chi_1 \times \psi_1 + \alpha$

  Continue with

$$\left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right) \leftarrow \left(\begin{array}{c} x_0 \\ \chi_1 \\ \hline x_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \leftarrow \left(\begin{array}{c} y_0 \\ \psi_1 \\ \hline y_2 \end{array}\right)$$

endwhile

1.6.3 Coding with Slicing and Redicing: Dot Product

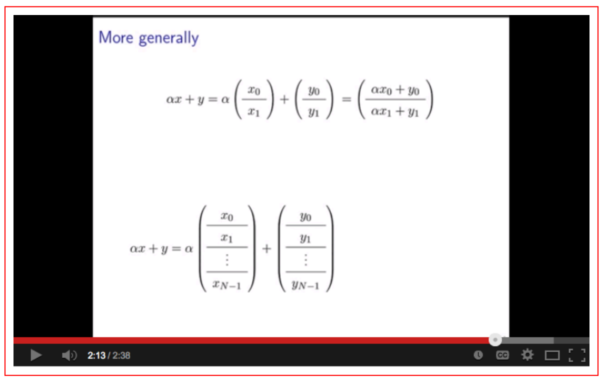


Homework 1.6.3.11 Do the IPython Notebook exercise

1.6.3 Programming without indices (dot product)

as described in the video for this unit.

1.6.4 Slicing and Dicing: axpy



In the video, we justify the following theorem:

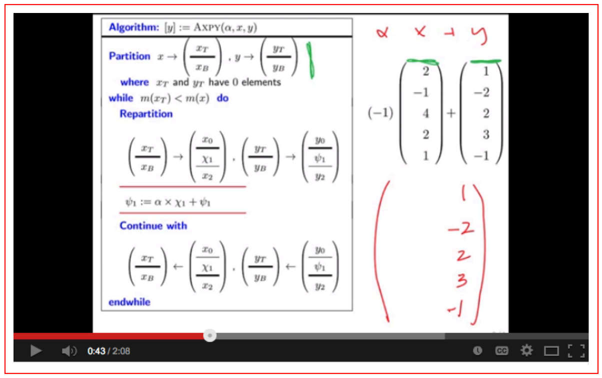
Theorem 1.17 *Let $\alpha \in \mathbb{R}$, $x, y \in \mathbb{R}^n$, and partition (Slice and Dice) these vectors as*

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

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

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

1.6.5 Algorithms with Slicing and Redicing: axpy



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

Partition $x \rightarrow \left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right)$, $y \rightarrow \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right)$

where x_T and y_T have 0 elements

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

Repartition

$$\left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right) \rightarrow \left(\begin{array}{c} x_0 \\ \hline \chi_1 \\ \hline x_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \rightarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$$

where χ_1 has 1 row, ψ_1 has 1 row

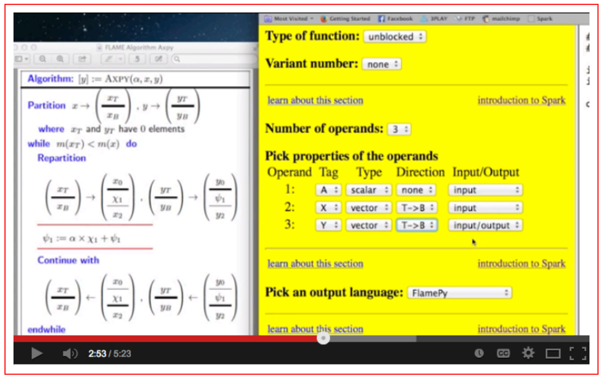
$$\psi_1 := \alpha \times \chi_1 + \psi_1$$

Continue with

$$\left(\begin{array}{c} x_T \\ \hline x_B \end{array}\right) \leftarrow \left(\begin{array}{c} x_0 \\ \hline \chi_1 \\ \hline x_2 \end{array}\right), \left(\begin{array}{c} y_T \\ \hline y_B \end{array}\right) \leftarrow \left(\begin{array}{c} y_0 \\ \hline \psi_1 \\ \hline y_2 \end{array}\right)$$

endwhile

1.6.6 Coding with Slicing and Redicing: axpy



Homework 1.6.6.12 Do the IPython Notebook exercise

1.6.6 Programming without indices (axpy)

as described in the video for this unit.

1.7 Enrichment

1.7.1 Learn the Greek Alphabet

In this course, we try to use the letters and symbols we use in a very consistent way, to help communication. As a general rule

- Lowercase Greek letters (α, β, 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.

Typically, if we use a given uppercase letter for a matrix, then we use the corresponding lower case letter for its columns (which can be thought of as vectors) and the corresponding lower case Greek letter for the elements in the matrix. Similarly, as we have already seen in previous sections, if we start with a given letter to denote a vector, then we use the corresponding lower case Greek letter for its elements.

Table 1.1 lists how we will use the various letters.

1.7.2 Other Norms

A norm is a function, in our case of a vector in $\mathbb{R}^n$, that maps every vector to a nonnegative real number. The simplest example is the absolute value of a real number: Given $\alpha \in \mathbb{R}$, the absolute value of α, often written as $|\alpha|$, equals the magnitude of α:

$$|\alpha| = \begin{cases} \alpha & \text{if } \alpha \geq 0 \\ -\alpha & \text{otherwise.} \end{cases}$$

Notice that only $\alpha = 0$ has the property that $|\alpha| = 0$ and that $|\alpha + \beta| \leq |\alpha| + |\beta|$, which is known as the *triangle inequality*.

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

Figure 1.1: Correspondence between letters used for matrices (uppercase Roman),vectors (lowercase Roman), and the symbols used to denote their scalar entries (lowercase Greek letters).

Similarly, one can find functions, called norms, that measure the magnitude of vectors. One example is the (Euclidean) length of a vector, which we call the 2-norm: for $x \in \mathbb{R}^n$,

$$\|x\|_2 = \sqrt{\sum_{i=0}^{n-1} \chi_i^2}.$$

Clearly, $\|x\|_2 = 0$ if and only if $x = 0$ (the vector of all zeroes). Also, for $x, y \in \mathbb{R}^n$, one can show that $\|x + y\|_2 \leq \|x\|_2 + \|y\|_2$.

A function $\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$ 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).

The 2-norm (Euclidean length) is a norm.

Are there other norms? The answer is yes:

- The taxi-cab norm, also known as the 1-norm:

  $$\|x\|_1 = \sum_{i=0}^{n-1} |\chi_i|.$$

  It is sometimes called the taxi-cab norm because it is the distance, in blocks, that a taxi would need to drive in a city like New York, where the streets are laid out like a grid.

- For $1 \leq p \leq \infty$, the p-norm:

  $$\|x\|_p = \sqrt[p]{\sum_{i=0}^{n-1} |\chi_i|^p} = \left(\sum_{i=0}^{n-1} |\chi_i|^p \right)^{1/p}.$$

  Notice that the 1-norm and the 2-norm are special cases.

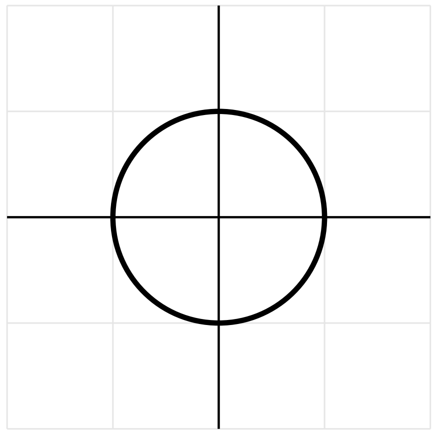
- The ∞-norm:

  $$\|x\|_\infty = \lim_{p \to \infty} \sqrt[p]{\sum_{i=0}^{n-1} |\chi_i|^p} = \max_{i=0}^{n-1} |\chi_i|.$$

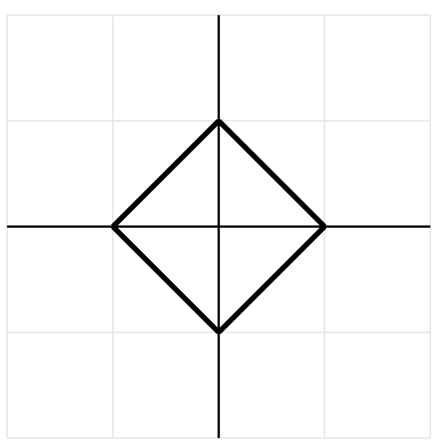
The bottom line is that there are many ways of measuring the length of a vector. In this course, we will only be concerned with the 2-norm.

We will not prove that these are norms, since that, in part, requires one to prove the triangle inequality and then, in turn, requires a theorem known as the Cauchy-Schwarz inequality. Those interested in seeing proofs related to the results in this unit are encouraged to investigate norms further.

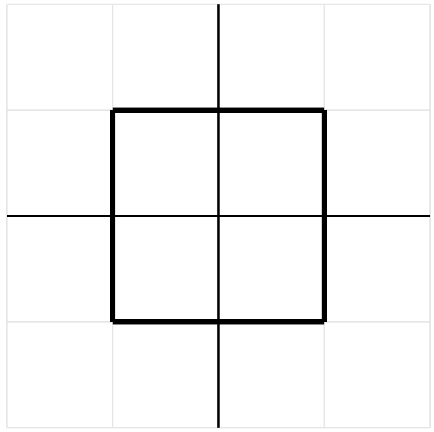
Example 1.18 The vectors with norm equal to one are often of special interest. Below we plot the points to which vectors x with $\|x\|_2 = 1$ point (when those vectors start at the origin, $(0,0)$). (E.g., the vector $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ points to the point $(1,0)$ and that vector has 2-norm equal to one, hence the point is one of the points to be plotted.)



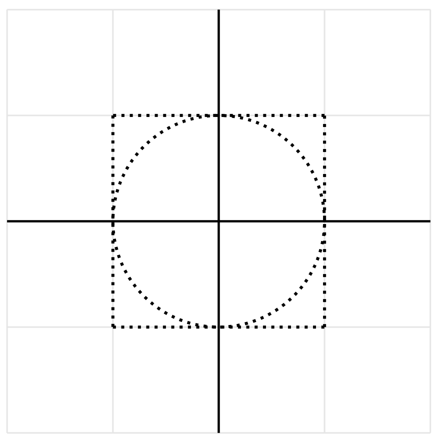
Example 1.19 Similarly, below we plot all points to which vectors x with $\|x\|_1 = 1$ point (starting at the origin).



Example 1.20 Similarly, below we plot all points to which vectors x with $\|x\|_\infty = 1$ point.



Example 1.21 Now consider all points to which vectors x with $\|x\|_p = 1$ point, when $2 < p < \infty$. These form a curve somewhere between the ones corresponding to $\|x\|_2 = 1$ and $\|x\|_\infty = 1$:



1.7.3 Overflow and Underflow

A detailed discussion of how real numbers are actually stored in a computer (approximations called floating point numbers) goes beyond the scope of this course. We will periodically expose some relevant properties of floating point numbers througout the course.

What is import right now is that there is a largest (in magnitude) number that can be stored and a smallest (in magnitude) number not equal to zero, that can be stored. Try to store a number larger in magnitude than this largest number, and you cause what is called an *overflow*. This is often stored as a "Not-A-Number" (NAN). Try to store a number not equal to zero and smaller in magnitude than this smallest number, and you cause what is called an *underflow*. An underflow is often set to zero.

Let us focus on overflow. The problem with computing the length (2-norm) of a vector is that it equals the square root of the sum of the squares of the components. While the answer may not cause an overflow, intermediate results when squaring components could. Specifically, any component greater in magnitude than the square root of the largest number that can be stored will overflow when squared.

The solution is to exploit the following observation: Let $\alpha > 0$. Then

$$\|x\|_2 = \sqrt{\sum_{i=0}^{n-1} \chi_i^2} = \sqrt{\sum_{i=0}^{n-1} \left(\alpha^2 \frac{\chi_i}{\alpha}\right)^2} = \sqrt{\alpha^2 \sum_{i=0}^{n-1} \left(\frac{\chi_i}{\alpha}\right)^2} = \alpha \sqrt{\left(\frac{1}{\alpha}x\right)^T \left(\frac{1}{\alpha}x\right)}$$

Now, we can use the following algorithm to compute the length of vector x:

- Choose $\alpha = \max_{i=0}^{n-1} |\chi_i|$.
- Scale $x := x/\alpha$.
- Compute $\|x\|_2 = \alpha\sqrt{x^T x}$.

Notice that no overflow for intermediate results (when squaring) will happen because all elements are of magnitude less than or equal to one. Similarly, only values that are very small relative to the final results will underflow because at least one of the components of x/α equals one.

1.7.4 A Bit of History

The functions that you developed as part of your LAFF library are very similar in functionality to Fortran routines known as the (level-1) Basic Linear Algebra Subprograms (BLAS) that are commonly used in scientific computing libraries. These were first proposed in the 1970s and were used in the development of one of the first linear algebra libraries, LINPACK. Classic references for that work are

- C. Lawson, R. Hanson, D. Kincaid, and F. Krogh, "Basic Linear Algebra Subprograms for Fortran Usage," *ACM Transactions on Mathematical Software*, 5 (1979) 305–325.
- J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, *LINPACK Users' Guide*, SIAM, Philadelphia, 1979.

The style of coding that we introduce in Section **??** is at the core of our FLAME project and was first published in

- John A. Gunnels, Fred G. Gustavson, Greg M. Henry, and Robert A. van de Geijn, "FLAME: Formal Linear Algebra Methods Environment," *ACM Transactions on Mathematical Software*, 27 (2001) 422–455.
- Paolo Bientinesi, Enrique S. Quintana-Orti, and Robert A. van de Geijn, "Representing linear algebra algorithms in code: the FLAME application program interfaces," *ACM Transactions on Mathematical Software*, 31 (2005) 27–59.

1.8 Wrap Up

1.8.1 Homework

Homework 1.8.1.1 Let

$$x = \begin{pmatrix} 2 \\ -1 \end{pmatrix}, \quad y = d\begin{pmatrix} \alpha \\ \beta - \alpha \end{pmatrix}, \quad \text{and} \quad x = y.$$

Indicate which of the following must be true:

(a) $\alpha = 2$

(b) $\beta = (\beta - \alpha) + \alpha = (-1) + 2 = 1$

(c) $\beta - \alpha = -1$

(d) $\beta - 2 = -1$

(e) $x = 2e_0 - e_1$

Homework 1.8.1.2 You are hiking in the woods. To begin, you walk 3 miles south, and then you walk 6 miles east. Give a vector (with units miles) that represents the displacement from your starting point.

Homework 1.8.1.3 You are going out for a Sunday drive. Unfortunately, closed roads keep you from going where you originally wanted to go. You drive 10 miles east, then 7 miles south, then 2 miles west, and finally 1 mile north. Give a displacement vector, x, (in miles) to represent your journey and determine how far you are from your starting point.

$$x = \begin{pmatrix} \quad \end{pmatrix}.$$

The distance from the starting point (length of the displacement vector) is _____.

Homework 1.8.1.4 In 2012, we went on a quest to share our research in linear algebra. Below are some displacement vectors to describe legs of this journey using longitude and latitude. For example, we began our trip in Austin, TX and landed in San Jose, CA. The displacement vector for this leg of the trip was $\begin{pmatrix} 7^\circ\ 05' \\ 24^\circ\ 09' \end{pmatrix}$, since Austin has coordinates $30^\circ\ 15'$ N(orth) $97^\circ\ 45'$ W(est) and San Joses are $37^\circ\ 20'$ N $121^\circ\ 54'$ W. Notice that for convenience, we extend the notion of vectors so that the components include units as well as real numbers. We could have changed each entry to real numbers with, for example, units of minutes (60 minutes $(')= 1$ degree$(^\circ)$.)
The following is a table of cities and their coordinates.

City	Coordinates	City	Coordinates
London	$00^\circ\ 08'$ E, $51^\circ\ 30'$ N	Austin	$-97^\circ\ 45'$ E, $30^\circ\ 15'$ N
Pisa	$10^\circ\ 21'$ E, $43^\circ\ 43'$ N	Brussels	$04^\circ\ 21'$ E, $50^\circ\ 51'$ N
Valencia	$00^\circ\ 23'$ E, $39^\circ\ 28'$ N	Darmstadt	$08^\circ\ 39'$ E, $49^\circ\ 52'$ N
Zürich	$08^\circ\ 33'$ E, $47^\circ\ 22'$ N	Krakow	$19^\circ\ 56'$ E, $50^\circ\ 4'$ N

Determine the order in which cities were visited, starting in Austin, given that the legs of the trip (given in order) had the following displacement vectors:

$$\begin{pmatrix} 102^\circ\ 06' \\ 20^\circ\ 36' \end{pmatrix} \rightarrow \begin{pmatrix} 04^\circ\ 18' \\ -00^\circ\ 59' \end{pmatrix} \rightarrow \begin{pmatrix} -00^\circ\ 06' \\ -02^\circ\ 30' \end{pmatrix} \rightarrow \begin{pmatrix} 01^\circ\ 48' \\ -03^\circ\ 39' \end{pmatrix} \rightarrow$$

$$\begin{pmatrix} 09^\circ\ 35' \\ 06^\circ\ 21' \end{pmatrix} \rightarrow \begin{pmatrix} -19^\circ\ 56' \\ 01^\circ\ 26' \end{pmatrix} \rightarrow \begin{pmatrix} 00^\circ\ 15' \\ -12^\circ\ 02' \end{pmatrix} \rightarrow \begin{pmatrix} -98^\circ\ 08' \\ -09^\circ\ 13' \end{pmatrix}$$

Homework 1.8.1.5 These days, high performance computers are called clusters and consist of many compute nodes, connected via a communication network. Each node of the cluster is basically equipped with a central processing unit (CPU), memory chips, a hard disk, and a network card. The nodes can be monitored for average power consumption (via power sensors) and application activity.
A system administrator monitors the power consumption of a node of such a cluster for an application that executes for two hours. This yields the following data:

Component	Average power (W)	Time in use (in hours)	Fraction of time in use
CPU	90	1.4	0.7
Memory	30	1.2	0.6
Disk	10	0.6	0.3
Network	15	0.2	0.1
Sensors	5	2.0	1.0

What is the total energy consumed by this node? Note: energy = power × time.
Answer:
Now, let's set this up as two vectors, x and y. The first records the power consumption for each of the components and the other for the total time that each of the components is in use:

$$x = \begin{pmatrix} 90 \\ 30 \\ 10 \\ 15 \\ 5 \end{pmatrix} \quad \text{and} \quad y = 2 \begin{pmatrix} 0.7 \\ 0.6 \\ 0.3 \\ 0.1 \\ 1.0 \end{pmatrix}.$$

Compute $x^T y$.
Answer:
Think: How do the two ways of computing the answer relate?
Energy is usually reported in KWh (thousands of Watt-hours), but that is not relevant to the question.
Answer:

1.8.2 Summary of Vector Operations

Vector scaling	$\alpha x = \begin{pmatrix} \alpha\chi_0 \\ \alpha\chi_1 \\ \vdots \\ \alpha\chi_{n-1} \end{pmatrix}$
Vector addition	$x + y = \begin{pmatrix} \chi_0 + \psi_0 \\ \chi_1 + \psi_1 \\ \vdots \\ \chi_{n-1} + \psi_{n-1} \end{pmatrix}$
Vector subtraction	$x - y = \begin{pmatrix} \chi_0 - \psi_0 \\ \chi_1 - \psi_1 \\ \vdots \\ \chi_{n-1} - \psi_{n-1} \end{pmatrix}$
AXPY	$\alpha x + y = \begin{pmatrix} \alpha\chi_0 + \psi_0 \\ \alpha\chi_1 + \psi_1 \\ \vdots \\ \alpha\chi_{n-1} + \psi_{n-1} \end{pmatrix}$
dot (inner) product	$x^T y = \sum_{i=0}^{n-1} \chi_i \psi_i$
vector length	$\|x\|_2 = \sqrt{x^T y} = \sqrt{\sum_{i=0}^{n-1} \chi_i \psi_i}$

1.8.3 Summary of the Properties of Vector Operations

Vector Addition

- Is commutative. That is, for all vectors $x, y \in \mathbb{R}^n, x + y = y + x$.
- Is associative. That is, for all vectors $x, y, z \in \mathbb{R}^n, (x + y) + z = x + (y + z)$.
- Has the zero vector as an identity.
- For all vectors $x \in \mathbb{R}^n, x + 0 = 0 + x = x$ where 0 is the vector of size n with 0 for each component.
- Has an inverse, x. That is $x + (x) = 0$.

The Dot Product of Vectors

- Is commutative. That is, for all vectors $x, y \in \mathbb{R}^n, x^T y = y^T x$.
- Distributes over vector addition. That is, for all vectors $x, y, z \in \mathbb{R}^n, x^T (y + z) = x^T y + x^T z$ and $(x + y)^T z = x^T z + y^T z$.

Partitioned vector operations

For (sub)vectors of appropriate size

- $$\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix} + \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix} = \begin{pmatrix} x_0 + y_0 \\ x_1 + y_1 \\ \vdots \\ x_{N-1} + y_{N-1} \end{pmatrix}.$$
- $$\begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{pmatrix}^T \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{pmatrix} = x_0^T y_0 + x_1^T y_1 + \cdots + x_{N-1}^T y_{N-1} = \sum_{i=0}^{N-1} x_i^T y_i.$$

Other Properties

- For $x, y \in \mathbb{R}^n$, $(x+y)^T(x+y) = x^T x + 2x^T y + y^T y$.
- For $x, y \in \mathbb{R}^n$, $x^T y = 0$ if and only if x and y are orthogonal.
- Let $x, y \in \mathbb{R}^n$ be nonzero vectors and let the angle between them equal . Then $cos() = x^T y \|x\|_2 \|y\|_2$.
- For $x \in \mathbb{R}^n$, $x^T e_i = e_i^T x = \chi_i$ where χ_i equals the ith component of x.

1.8.4 Summary of the Routines for Vector Operations

Operation Abbrev.	Definition	Function	Approx. cost	
			flops	memops
Vector-vector operations				
Copy (COPY)	$y := x$	`laff.copy( x, y )`	0	$2n$
Vector scaling (SCAL)	$x := \alpha x$	`laff.scal( alpha, x )`	n	$2n$
Scaled addition (AXPY)	$y := \alpha x + y$	`laff.axpy( alpha, x, y )`	$2n$	$3n$
Dot product (DOT)	$\alpha := x^T y$	`alpha = laff.dot( x, y )`	$2n$	$2n$
Length (NORM2)	$\alpha := \|x\|_2$	`alpha = laff.norm2( x )`	$2n$	n