UNIVERSIDAD DEL CEMA
Buenos Aires
Argentina

Serie
DOCUMENTOS DE TRABAJO

Área: Matemática y Economía

Intuitive Mathematical Economics Series. Linear Functions, their Matrix Form and the Geometry of Linear Systems of Equations

Sergio Pernice

Diciembre 2019
Nro. 707

https://ucema.edu.ar/publicaciones/doc_trabajo.php
UCEMA: Av. Córdoba 374, C1054AAP Buenos Aires, Argentina
ISSN 1668-4575 (impreso), ISSN 1668-4583 (en línea)
Editor: Jorge M. Streb; asistente editorial: Valeria Dowding <jae@cema.edu.ar>

Intuitive Mathematical Economics Series
Linear Functions, Their Matrix Form and The Geometry of Linear Systems of Equations

Sergio A. Pernice[1]

Universidad del CEMA
Av. Córdoba 374, Buenos Aires, 1054, Argentina

Abstract

Matrices, their products, linear systems, and the underlying geometric ideas are presented in an intuitive and practical way for economics students and other students of the social sciences. Python Jupyter notebooks are used to present examples that enforce the geometric ideas.

Keywords: Linear systems, Matrices.

1 Introduction

Continuing with the objective of the "Intuitive Mathematical Economics Series" of presenting to economics students, and in general students of the social science, the mathematics they need in an intuitive way, in this article we present linear functions, their matrix form, and linear systems of equations in an intuitive, geometric way.

The article first presents linear and affine functions. Their matrix form is presented simply as a short way of encapsulating linear functions. Presented in this way, matrix-vector ad matrix-matrix multiplication become natural. We also present systems of linear equations and their matrix form. Then we introduce the "column view" of matrices, which make the geometry of linear systems transparent in any dimension. This is the main purpose of the article, allowing a very intuitive approach to linear equations. Finally we present some examples with Jupyter notebooks.

[1] sp@ucema.edu.ar

2 Linear and Affine Functions

Definition 2.1. A scalar function $f : \mathbb{R}^n \to \mathbb{R}$ is *affine* if it has the form:

$$f\left[\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\right] = a_1x_1 + a_2x_2 + \cdots + a_nx_n + b$$

where $a_1, \cdots, a_n, b \in \mathbb{R}$. Note that $f[\mathbf{0}] = b$, where $\mathbf{0}$ is the vector with all its elements equal to zero.

Definition 2.2. A scalar function $f : \mathbb{R}^n \to \mathbb{R}$ is *linear* if it has the form:

$$f\left[\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\right] = a_1x_1 + a_2x_2 + \cdots + a_nx_n \quad (b = 0)$$

where $a_1, \cdots, a_n \in \mathbb{R}$. Note that $f[\mathbf{0}] = 0$.

Definition 2.3. A vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ is *affine* if it has the form:

$$f\left[\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\right] = \begin{pmatrix} f_1[\mathbf{x}] \\ f_2[\mathbf{x}] \\ \vdots \\ f_m[\mathbf{x}] \end{pmatrix}$$

where we are using bold letters to denote vectors, as in $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$. The functions $f_i[\mathbf{x}]$, $i = 1, \cdots, m$ are affine functions.

Definition 2.4. A vector-valued function $f : \mathbb{R}^n \to \mathbb{R}^m$ is *linear* if it has the form as in definition 2.3 but where all the functions $f_i[\mathbf{x}]$ are linear.

Note that a property of linear functions is that $f[\mathbf{0}] = \mathbf{0}$, i.e. it transforms the null vector into a null vector. However note that the first zero vector $\mathbf{0}$ lives in $\mathbb{R}^n$ and the second in $\mathbb{R}^m$.

The following examples show affine, linear and general nonlinear functions:

Example 2.1.

$$f\left[\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\right] = \begin{pmatrix} 3x_1 + 0.2x_2 - x_3 \\ x_3 - x_1 + 7 \\ x_1 - x_2 - x_3 - 1 \end{pmatrix} \qquad \text{is affine}$$

Example 2.2.

$$f\left[\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\right] = \begin{pmatrix} 2x_2 - x_3 \\ x_3 - x_1 \\ x_1 - x_2 - x_3 \end{pmatrix} \qquad \text{is linear}$$

Example 2.3.

$$f\left[\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\right] = \begin{pmatrix} 2x_2 - x_3 \\ x_3 - x_1 \\ x_1 - x_2 - e^{x_3} \end{pmatrix} \qquad \text{is neither linear nor affine}$$

Note that with the above definition, for functions $f : \mathbb{R} \to \mathbb{R}$, a function like $f(x) = ax + b$, that normally would be called linear, is in fact an affine function. It is linear only for $b = 0$.

Consider a general linear function. From definition 2.4, a general linear function $f : \mathbb{R}^n \to \mathbb{R}^m$ can be written as:

$$f\left[\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\right] = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{pmatrix} \tag{2.1}$$

Linear functions like (2.1) can be seen as *defining* of a product between a matrix and a vector:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \equiv \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{pmatrix} \tag{2.2}$$

where on the left hand side we have a matrix A, with elements a_{ij}, $i = 1, \cdots, m$, $j = 1, \cdots, n$. The element a_{ij} corresponds to the number in the i^{th} row and the j^{th} column of matrix A. A is said to have "dimensions" $m \times n$, also called "shape" (m, n).

In a more compact notation, the left hand side of equation (2.2) can be written as $A\mathbf{x}$, and the product between the matrix A of dimensions $m \times n$ and the vector $\mathbf{x} \in \mathbb{R}^s$, as defined by (2.2), only makes sense if $s = n$. The result is a vector in $\mathbb{R}^m$, which can also be seen as a matrix of dimension $m \times 1$:

$$\underset{m\times n}{A}\ \underset{n\times 1}{\mathbf{x}} = \underset{m\times 1}{\mathbf{y}} \tag{2.3}$$

The matrix times vector multiplication can be expressed in terms of their components as

$$(A\mathbf{x})_i = \sum_{j=1}^{n} a_{ij}x_j \tag{2.4}$$

In a more compact notation, using the "Einstein summation convention", it can be written as:

$$(A\mathbf{x})_i = a_{ij}x_j \tag{2.5}$$

where the convention is that equal indices (the index "j" in a_{ij} and in x_j) is summed over.

Equation (2.2) indicates that every linear function $f : \mathbb{R}^n \to \mathbb{R}^m$ has associated an $m \times n$ matrix and vice versa. As already mentioned, it also defines the matrix-vector product.

The composition of two linear functions $f : \mathbb{R}^n \to \mathbb{R}^m$, with associated $m \times n$ matrix F and $g : \mathbb{R}^m \to \mathbb{R}^s$, with associated $s \times m$ matrix G is defined as:

$$g\left(f\left(\mathbf{x}\right)\right) = g\left(F\mathbf{x}\right) = G\left(F\mathbf{x}\right) \tag{2.6}$$

where in the first equality $F\mathbf{x}$ represents the vector in $\mathbb{R}^m$ that results from the product of the $m \times n$ matrix F with the vector $\mathbf{x} \in \mathbb{R}^n$, and the second equality represents the product of the $s \times m$ matrix G and the vector $F\mathbf{x} \in \mathbb{R}^m$.

In components form, as in (2.4), the composition (3.2) is:

$$g\left(f\left(\mathbf{x}\right)\right)_i = g\left(\sum_{j=1}^{n} F_{kj}x_j\right)_i = \sum_{k=1}^{m} G_{ik}\left(\sum_{j=1}^{n} F_{kj}x_j\right) = \sum_{j=1}^{n}\left(\sum_{k=1}^{m} G_{ik}F_{kj}\right)x_j = \left(\left(GF\right)\mathbf{x}\right)_i \tag{2.7}$$

Let us pause to understand every term in these equalities. The left hand side indicates that we are going to compute the i^{th} component of the vector that results from the composition of the function g with the function f applied to the vector $\mathbf{x}$. In the second term the reader should be warned that the notation may be misunderstood as indicating that g is only a function of the component k of the vector $f(\mathbf{x})$. This is most definitively not what it means. g is a function of the whole vector $f(\mathbf{x})$, whose k^{th} component is explicitly shown, as computed with the matrix-vector product between the matrix F associated with the linear function f and the vector $\mathbf{x}$. The third term computes g of this resulting vector by multiplying it by the matrix G associated with the linear transformation g. The fourth term uses the distributive property of the finite sums and products to interchange the sums in j and k. In this fourth term, inside the parenthesis, appears the expression

$$\sum_{k=1}^{m} G_{ik}F_{kj} \equiv \underset{s\times n}{\left(\underset{s\times m}{G}\ \underset{m\times n}{F}\right)_{ij}} \tag{2.8}$$

This expression *defines* the product between two matrices G and F. It is important to note that this product rule follows directly from the notion of a composition of linear transformations, together with the fact that linear transformations are one to one related to matrices.

3 Vectors as Matrices, and Matrix Transpose

With the notion of matrix-to-matrix product defined in (2.8), vectors can be seen as a special case of matrices. A general matrix A has dimensions $m \times n$, where m and n are positive integers. A vector can be seen as a matrix with dimensions $n \times 1$.

Given a matrix A with dimensions $m \times n$ and elements a_{ij}, the matrix *transpose* A^T, is a matrix with dimensions $n \times m$ with elements $a^T_{ij} = a_{ji}$, i.e., with columns and rows interchanged.

Example 3.1.

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}, \qquad A^T = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

```
In [11]:  A = np.array([[1,2],[3,4],[5,6]])
          AT = A.transpose()
          print(A)
          print(A.shape)
          print(AT)
          print(AT.shape)
```

```
[[1 2]
 [3 4]
 [5 6]]
(3, 2)
[[1 3 5]
 [2 4 6]]
(2, 3)
```

Figure 1: Python numpy array as a matrix and its transpose, with their respective dimensionalities.

In Python, with the library numpy we have.

Note that the dimensionality of the matrix is given by the ".shape" operation.

Now, looking at the (column) vector $\mathbf{x}$ as an $n \times 1$ matrix, its transpose must be a $1 \times n$ matrix, and is known as a "row" vector:

Example 3.2.

$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \qquad \mathbf{x}^T = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

In terms of the transpose of a vector, the scalar product between two vectors can be seen as a product of two matrices as in (2.8). Indeed:

$$\mathbf{x} \cdot \mathbf{y} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \tag{3.1}$$

$$= \underbrace{x_1 y_1 + x_2 y_2 + \cdots + x_n y_n}_{1\times 1} \tag{3.2}$$

$$= \underbrace{(x_1, x_2, \cdots, x_n)}_{1\times n} \underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{n\times 1} \tag{3.3}$$

see Figure (3).

```
In [5]:
x = np.array([[1],[2],[3]])
xT = x.transpose()
print(x)
print(x.shape)
print(xT)
print(xT.shape)
```

```
[[1]
 [2]
 [3]]
(3, 1)
[[1 2 3]]
(1, 3)
```

Figure 2: A column vector as an $n \times 1$ matrix and its transpose, a *row* vector, or a $1 \times n$ matrix.

```
In [14]:
x = np.array([[1],[2]])
y = np.array([[3],[4]])
print(np.vdot(x,y))
print(np.dot(x.transpose(),y))
```

```
11
[[11]]
```

Figure 3: In Python, the scalar product between two vectors is done with the numpy function **vdot**. We can also take the scalar product as a matrix product we use the numpy function **dot**. In this case we have to transpose the first vector. Note also the type difference.

4 Linear Transformations, Systems of Equations and Their Matrix Form

As we saw in (2.1), a linear function is a mapping between vectors in $\mathbb{R}^n$ into vectors in $\mathbb{R}^m$, and every linear transformation has its matrix form (2.2).

A way to view this mapping, specially insightful when $m = n$ (square matrices), but useful even when $m \neq n$, is to think about the linear mapping as a *transformation* of the input vector into the output vector. So in the equation

$$\mathbf{y} = A\mathbf{x} \tag{4.1}$$

$\mathbf{y}$ is viewed as the (output) vector in which the matrix A transforms the (input) vector $\mathbf{x}$.

If the matrix A and vector $\mathbf{y}$ are known, but the vector $\mathbf{x}$ is not, equation (4.1) defines a system of linear equations. From the "transformation" perspective, solving the equation amounts to

finding, if it exists, the vector (there may be more than one) $\mathbf{x}$ such that when transformed by A gives you back the known vector $\mathbf{y}$. Let us consider an example:

Example 4.1. Consider the equations

$$x + y = 3$$
$$x - y = -1$$

Fig. **??** is the picture we tend to have in our mind for a system of equations like this.

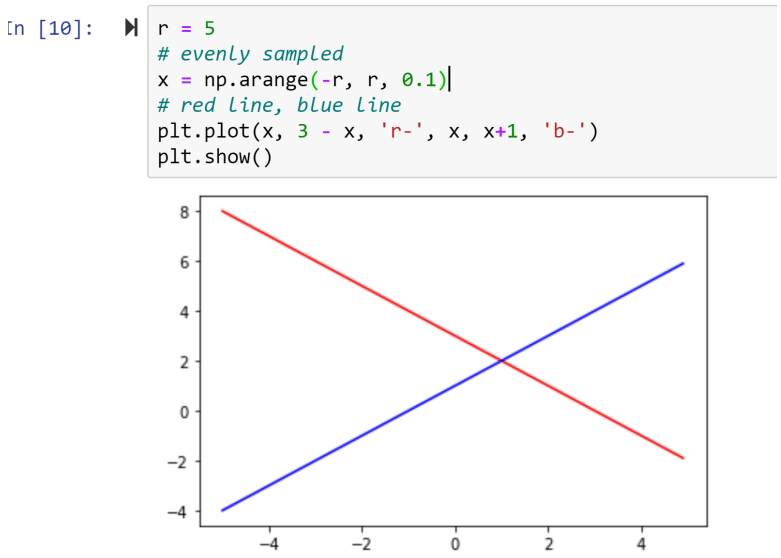

Figure 4

Equation $x + y = 3$ defines the straight line $y = 3 - x$ in the (x, y) plane, and equation $x - y = -1$ the straight line $y = 1 + x$. Solving the system of equations amounts to finding the point of intersection between these two lines, which happens to be at $x = 1, y = 2$.

But this system can also be seen, in a matrix form, as:

$$A\mathbf{x} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x + y \\ x - y \end{pmatrix} = \begin{pmatrix} 3 \\ -1 \end{pmatrix} = \mathbf{y}$$

in this way, the matrix A transforms the unknown vector $\mathbf{x}$ into the vector $\mathbf{y} = (3, -1)^T$. And solving the system must amount to somehow doing the *inverse* transformation to $\mathbf{y}$.

5 Matrix-Vector Product, the Rows View, and Column View

A matrix A can be seen as a collection of rows $[a_{i1}, a_{i2}, \cdots, a_{in}]$. It can also be seen as a collection of columns $\begin{pmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{pmatrix}$. Each one of these views open their own "vistas".

Let us start with the view of a matrix as a collection of rows:

$$A = \begin{pmatrix} \text{—}\mathbf{r_1}\text{—} \\ \text{—}\mathbf{r_2}\text{—} \\ \vdots \\ \text{—}\mathbf{r_m}\text{—} \end{pmatrix} \tag{5.1}$$

where the row vectors are $\mathbf{r_i} = (a_{i1}, a_{i2}, \cdots, a_{in})$. In terms of row vectors the matrix-vector product (2.2) is

$$A\mathbf{x} = \begin{pmatrix} \text{—}\mathbf{r_1}\text{—} \\ \text{—}\mathbf{r_2}\text{—} \\ \vdots \\ \text{—}\mathbf{r_m}\text{—} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \mathbf{r_1} \cdot \mathbf{x} \\ \mathbf{r_2} \cdot \mathbf{x} \\ \vdots \\ \mathbf{r_n} \cdot \mathbf{x} \end{pmatrix} \tag{5.2}$$

As described in section 3, the row vectors are $1 \times n$ matrices, and the vector $\mathbf{x}$ is an $n \times 1$ matrix, therefore the product $\mathbf{r_i} \cdot \mathbf{x}$ is 1×1 (scalar).

Note that since the row vectors $\mathbf{r_i}$ are n dimensional, if $m > n$, at least $m - n$ of them must be a linear combination of the other n.

Equation (5.2) represents the row view of a matrix-vector product. But we can also see a matrix as a collection of columns:

$$A = \begin{pmatrix} | & | & & | \\ \mathbf{c_1} & \mathbf{c_2} & \cdots & \mathbf{c_n} \\ | & | & & | \end{pmatrix} \tag{5.3}$$

where each column $\mathbf{c_i}$ is an $m \times 1$ (column) vector $\in \mathbb{R}^m$ with elements:

$$\mathbf{c_i} = \begin{pmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{mi} \end{pmatrix} \tag{5.4}$$

Now, look at equation (2.2) again, specifically, the column vector in the right hand side. Notice that it can be written as a linear combination of the column vectors of the matrix A with

coefficients equal to the components of $\mathbf{x}$:

$$A\mathbf{x} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \equiv \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{pmatrix} = x_1\mathbf{c_1} + x_2\mathbf{c_2} + \cdots + x_n\mathbf{c_n} \tag{5.5}$$

this view of the matrix-vector multiplication is specially illuminating when transforming a system of equations into its matrix form, as in example 4.1.

A system of equations in its matrix form is

$$A\mathbf{x} = \mathbf{y} \tag{5.6}$$

where, again, A and $\mathbf{y}$ are known, and the vector $\mathbf{x}$ is unknown.

The right hand side of (5.5) indicates that $A\mathbf{x}$ is a linear combination of the n columns vectors $\mathbf{c_i}$ of A. Each one of these columns vectors live in $\mathbb{R}^m$, and $\mathbf{y}$ also lives in $\mathbb{R}^m$.

Just form these observations, remembering that A is an $m \times n$ matrix, we can arrive at very important high level conclusions:

1. **If $m > n$, the chances are that the system will have no solution.**

   In high school, if $m > n$, we would say that the system has *more equations than unknowns*, and therefore, in general, there will be no solution. This statement is typically supplemented with examples.

   Consider the case of $m = 3$ equations and $n = 2$ unknowns. Each equation represents a line in the (x, y) plane, and since there are 3 equations ($m = 3$), there are 3 straight lines. It is geometrically intuitive that in the plane, in general, 3 straight lines will not have any point belonging to the 3 of them.

   Two lines, in general, will have a point of intersection (unless they are parallel, in which case they either don't share any point, or they share infinitely many of them if they are the same line). But if we have three lines, each pair of them will in general have a point of intersection, but these 3 points will in general be different. So there will be no solution.

   The problem with this line of reasoning is that when m and n are large, our intuition doesn't work that well. It would be nice to have a way of looking the system of equations that makes our intuition almost independent of the values of m and n. This is the role the matrix form of the system of equations, and the column view of matrix-vector multiplication, is going to play.

   As equation (5.5) shows, the left hand side of equation (5.6), $A\mathbf{x}$, is a linear combination, with unknown coefficients, of the columns vectors of the matrix A. If we think for a moment about the unknown coefficients as arbitrary, the left hand side of (5.6) can be any vector in the subspace of $\mathbb{R}^m$ spanned by the n column vectors. Solving the equation means finding the particular linear combination that makes the vector on the left hand side equal to the known vector $\mathbf{y}$ on the right hand side.

But if $m > n$, even assuming that all the columns vectors are linearly independent, they will span at most a subspace of dimension n of $\mathbb{R}^m$. This means that, unless the vector $\mathbf{y}$ happens to lie in this same subspace, the system of equations won't have any solution!

Moreover, the chances that a vector of $\mathbb{R}^m$, chosen at random, will lie in a pre-specified subspace of dimension $n < m$ is zero (think of choosing m real numbers at random, what is the probability that n of them will be zero?)

Of course, it could be the case that the vector $\mathbf{y}$, for some particular reason specific to the problem at hand, does lie in the subspace spanned by the column vectors of A, in which case there will be a solution, but in this case it will be interesting to understand why it is so.

The point is that the matrix form of the system of equations, together with the column vector view of the matrix-vector product, does provide a clear intuition about why and when a system of equations will have solutions.

2. **If $m < n$, the chances are that the system will have infinite solutions.**

   Suppose, to fix the numbers, that $m = 2$ and $n = 3$, then the matrix in the left hand side of equation (5.6) has three 2-D column vectors. Unless two vectors happen to be proportional to each other, in general, every pair of vectors will span $\mathbb{R}^2$. Making one of the three components of $\mathbf{x}$ equal to zero, say x_3, equation (5.6) becomes $A\mathbf{x} = x_1\mathbf{c_1} + x_2\mathbf{c_2} = \mathbf{y}$. But since by hypothesis the vectors $\mathbf{c_1}$ and $\mathbf{c_2}$ are linearly independent, then we are guaranteed that there will be unique values of x_{1_3} and x_{2_3} such that

   $$x_{1_3}\mathbf{c_1} + x_{2_3}\mathbf{c_2} = \mathbf{y} \tag{5.7}$$

   Similarly, making $x_1 = 0$, there will be unique values of x_{2_1} and x_{3_1} such that $x_{2_1}\mathbf{c_2} + x_{3_1}\mathbf{c_3} = \mathbf{y}$.

   $$x_{2_1}\mathbf{c_2} + x_{3_1}\mathbf{c_3} = \mathbf{y} \tag{5.8}$$

   and unique values of x_{3_2} and x_{1_2} such that

   $$x_{3_2}\mathbf{c_3} + x_{1_2}\mathbf{c_1} = \mathbf{y} \tag{5.9}$$

   Consider for example the pair of equations (5.7) and (5.8). Multiplying (5.7) by an arbitrary constant z, (5.8) by $1 - z$ and adding, we get:

   $$zx_{1_3}\mathbf{c_1} + \left(zx_{2_3} + (1-z)\,x_{2_1}\right)\mathbf{c_2} + (1-z)x_{3_1}\mathbf{c_3} = \mathbf{y} \tag{5.10}$$

   where again, this equation holds for an arbitrary constant z, so there are infinite solutions. And similarly if we choose the pair of equations (5.8) and (5.9), or (5.7) and (5.9).

3. **If $m = n$, the chances are that the system will have a unique solution.** After the discussion above this should be trivial to the reader.

6 Python Exercises

6.1 General Case, Unique Solution

Let us try to solve the equation $A\mathbf{x} = \mathbf{y}$, where A is a 3×3 matrix and $\mathbf{y}$ a 3×1 vector, both generated at random with components in the range $[-1, 1]$, See Figure 5.

```
In [22]: A = 2*(np.random.rand(3,3)-0.5)
         print(A)
         y = 2*(np.random.rand(3,1)-0.5)
         print(y)

[[-0.11304645  0.39438475  0.21708227]
 [-0.61713756  0.31015962 -0.96353159]
 [-0.23231527  0.01745998  0.08209802]]
[[ 0.45507238]
 [ 0.84694019]
 [-0.38359962]]
```

Figure 5: A and $\mathbf{y}$ are pseudo-randomly generated.

Since A is generated at random (pseudo-random), its columns will in general be linearly independent, and therefore they will span the whole 3-D space. So, no mater what $\mathbf{y}$ is (also generated at random), there will in general be a unique linear combination of the column vectors of A equal to $\mathbf{y}$, therefore there will in general be a unique solution to the equation, that we find by using the function **LA.solve**(), see Figure 6.

```
In [23]: x = LA.solve(A, y)
         print(x)

[[ 1.42679722]
 [ 2.16593339]
 [-1.09564158]]
```

Figure 6: Unique solution to the equation $A\mathbf{x} = \mathbf{y}$. "LA" is used as a nickname for the "linalg" numpy library. One should type "from numpy import linalg as LA" at the beginning of the Jupyter notebook.

6.2 Case Where There is No Solution in General

Let us now generate a 3×3 matrix in which, by construction, one of the column vectors is a linear combination of the other two, but such that when we "look at" the matrix, this linear

dependence in not apparent to the "naked eye". We can do this by generating two 3×1 vectors at random, with values in the range $[-1, 1]$, see Figure 7. Note that we are subtracting a number

```
In [16]:
# c1 y c2 are two 3x1 vectors with random elements in the range [-1,1].
c1 = 2*(np.random.rand(3,1)-0.5)
c2 = 2*(np.random.rand(3,1)-0.5)
print(c1)
print(c2)
```

```
[[-0.92982617]
 [ 0.73660111]
 [-0.66442981]]
[[-0.14741923]
 [ 0.63063999]
 [ 0.49993865]]
```

Figure 7: $\mathbf{c_1}$ and $\mathbf{c_2}$ are generated at random.

to a vector. How is this possible? Search from "numpy broadcasting". It subtracts a vector of the same dimensions with all elements equal to 0.5.

Since $\mathbf{c_1}$ and $\mathbf{c_2}$ were generated at random (pseudo-random) the chances that they are linearly dependent is vanishingly small. Still we check it by taking the scalar product and dividing by their norm. We see that this is 0.2422..., see Figure 8. Since it is not equal to 1 or -1, they are

```
In [17]:
print(np.dot(np.transpose(c1),c2)/(LA.norm(c1)*LA.norm(c2)))
# The scalar product between c1 y c2 is c1.c2 = |c1||c2|cos(theta), therefore
# c1.c2/(|c1||c2|) = cos(theta). So if this number in NOT 1 nor -1,
# we know that the two vectors c1 y c2 are NOT linearly dependent.
```

```
[[0.24220675]]
```

Figure 8: $\cos\theta_{12} = \mathbf{c_1} \cdot \mathbf{c_2}/(|\mathbf{c_1}||\mathbf{c_2}|) = 0.2422...$, indicating that they are linearly independent.

linearly independent.

Let us generate $\mathbf{c_3}$ as a linear combination of $\mathbf{c_1}$ and $\mathbf{c_2}$ with pseudo random coefficients, so this linear dependency is not apparent to the "naked eye". With $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{c_3}$ we create the matrix A by concatenating the 3 vectors, see Figure 9.

```
[19]:
d1 = 2*(np.random.rand()-0.5) # d1 y d2 are scalars in the range [-1,1]
d2 = 2*(np.random.rand()-0.5)
c3 = d1*c1 + d2*c2  # c3, by construction, is linearly dependent of c1 y c2.
# However this is NOT obvious to the naked eye.
A = np.concatenate((c1, c2, c3), axis=1) # The matrix A has columns c1, c2 y c3
print(A.shape)
print(A)
```

```
(3, 3)
[[-0.92982617 -0.14741923  0.63417197]
 [ 0.73660111  0.63063999  0.0075318 ]
 [-0.66442981  0.49993865  1.05380532]]
```

Figure 9: A has columns $\mathbf{c_1}$, $\mathbf{c_2}$ and $\mathbf{c_3}$. $\mathbf{c_3}$ is a linear combination of $\mathbf{c_1}$ and $\mathbf{c_2}$.

We are going to try to solve the equation $A\mathbf{x} = \mathbf{y}$, for a vector $\mathbf{y}$ generated at random. Before looking at what happens, try to have the intuition for it.

The intuition goes as follows: we know that the left hand side of the equation, $A\mathbf{x}$, as (5.5) indicates, is a linear combination, with unknown coefficients x_1, x_2 and x_3, of the column vectors of A. And we explicitly constructed the matrix A so that the third column vector is a linear combination of the other two. Therefore, the left hand side of the equation $A\mathbf{x}$, is an arbitrary vector in the two dimensional subspace spanned by the first two columns of A.

On the other hand, the right hand side of the equation, namely, $\mathbf{y}$, is a 3-D vector generated at random. The chances that it will lie in the subspace generated by the first two columns of A is virtually zero. Therefore the equation will not have a solution. Let us see what Python tells us by using the function **LA.solve**(), see Figure 10.

```
[20]: y = 2*(np.random.rand(3,1)-0.5)
      print(y)
      x = LA.solve(A, y)
      print(x)
```

```
[[-0.91795445]
 [ 0.60899666]
 [-0.77515903]]
[[-1.57052523e+14]
 [ 1.85675401e+14]
 [-1.87109218e+14]]
```

Figure 10: We are trying, and failing, to solve the equation $A\mathbf{x} = \mathbf{y}$. In general, the 3-D vector $\mathbf{y}$ will not lie in the two-dimensional subspace spanned by the column vectors of A.

The output vector, with components of the order of 10^{14}, is the way Python has of telling us that there is no solution (the details of why this is so will be explained in a different article).

If, instead of generating $\mathbf{y}$ at random and unconstrained (apart from the fact that the components lie in the segment $[-1, 1]$), we generate it as a random linear combination of $\mathbf{c_1}$ and $\mathbf{c_2}$, we are guaranteed that it will lie in the subspace of these two vectors, and the equation will have a unique solution even though in general it does not, see Figure 11.

These results, that are not obvious at all looking at the numerical components, are trivially expected with the column geometric view of the matrix A.

7 Conclusions

As advertised in the Introduction, in this article we presented matrices as a compact away of writing linear functions, this makes matrix-vector and matrix-matrix multiplication natural. Then after presenting systems of linear equations in their matrix form, we showed that looking at a matrix as collection of column vectors, the geometry of linear systems becomes transparent in

```
In [21]: f1 = 2*(np.random.rand()-0.5) # f1 y f2 are scalars in the range [-1,1]
         f2 = 2*(np.random.rand()-0.5)
         y = f1*c1 + f2*c2
         print(y)
         x = LA.solve(A, y)
         print(x)

[[ 0.50557939]
 [-0.01573705]
 [ 0.81451282]]
[[-0.82307115]
 [ 0.93869615]
 [-0.19135478]]
```

Figure 11: When $\mathbf{y}$ does lie in the two-dimensional subspace spanned by the column vectors of A, there is a unique solution.

any dimension. Finally we tried to make this intuition apparent by presenting some examples with Jupiter notebooks.

To summarize the results, the rules that encode the solutions to the system of linear equations $A\mathbf{x} = \mathbf{y}$, when viewed from the perspective of the column space of the $m \times n$ matrix A are extremely simple:

1. The left hand side of the equation, $A\mathbf{x}$, for all possible values of $\mathbf{x}$, correspond to the subspace of dimension $r \leq m$ of $\mathbb{R}^m$ spanned by the column vectors of A. If $\mathbf{y}$ lies in this subspace, then there is at least a solution, if not, then there isn't any.
   (a) If $r = m$, then there is always at least one solution.
   (b) If $r < m$, then in general there is no solution, but if $\mathbf{y}$ happens to lie in this r-dimensional subspace, then there is at least one solution.
2. Consider now the case where $\mathbf{y}$ does lie in the r dimensional ($r \leq m$) subspace spanned by the n column vectors of A, so that we guaranteed at least one solution.
   (a) If $n = r$, then there is one and only one solution, because $\mathbf{y}$ has a unique linear expansion in terms of the n linearly independent column vectors.
   (b) If $n > r$, then $n - r$ column vectors are linearly dependent of the other r, therefore there is redundancy in the column vectors of A and there are infinite solutions.
      i. If the i^{th} column of A happens to be the zero vector $\mathbf{0}$, this is a particular case of linear dependence and the same happens, as the i^{th} component of $\mathbf{x}$, x_i, can take any value.