

4.2: Linear Equations

In physics, we are often called upon to solve **linear equations** of the form

$$\mathbf{A}\vec{x} = \vec{b}, \quad (4.2.1)$$

where \mathbf{A} is some $N \times N$ matrix, and both \vec{x} and \vec{b} are vectors for length N . Given \mathbf{A} and \vec{b} , the goal is to solve for \vec{x} .

It's an important and useful skill to recognize linear systems of equations when they arise in physics problems. Such equations can arise in many diverse contexts; we will give a couple of simple examples below.

Example 4.2.1

Suppose there is a set of N electrically charged point particles at positions $\{\vec{R}_0, \vec{R}_1, \dots, \vec{R}_{N-1}\}$. We do not know the value of the electric charges, but we are able to measure the electric potential at any point \vec{r} . The electric potential is given by

$$\phi(\vec{r}) = \sum_{j=0}^{N-1} \frac{q_j}{|\vec{r} - \vec{R}_j|}. \quad (4.2.2)$$

If we measure the potential at N positions, $\{\vec{r}_0, \vec{r}_1, \dots, \vec{r}_{N-1}\}$, how can the charges $\{q_0, \dots, q_{N-1}\}$ be deduced?

Solution

To do this, let us write the equation for the electric potential at point \vec{r}_i as:

$$\phi(\vec{r}_i) = \sum_{j=0}^{N-1} \left[\frac{1}{|\vec{r}_i - \vec{R}_j|} \right] q_j. \quad (4.2.3)$$

This has the form $\mathbf{A}\vec{x} = \vec{b}$, where $\mathbf{A}_{ij} \equiv \frac{1}{|\vec{r}_i - \vec{R}_j|}$, $\vec{b}_i \equiv \phi(\vec{r}_i)$, and the unknowns are $\vec{x}_j = q_j$.

Example 4.2.2

Linear systems of equations commonly appear in circuit theory. For example, consider the following parallel circuit of N power supplies and resistances:

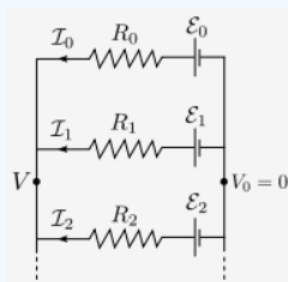


Figure 4.2.1

Assume the voltage on the right-hand side of the circuit is $V_0 = 0$. Given the resistances $\{R_0, \dots, R_{N-1}\}$ and the EMFs $\{\mathcal{E}_0, \dots, \mathcal{E}_{N-1}\}$, how do we find the left-hand voltage V and the currents $\{I_0, \dots, I_{N-1}\}$?

Solution

We follow the usual laws of circuit theory. Each branch of the parallel circuit obeys Ohm's law,

$$I_j R_j + V = \mathcal{E}_j. \quad (4.2.4)$$

Furthermore, the currents obey Kirchhoff's law (conservation of current), so

$$\sum_{j=0}^{N-1} I_j = 0. \quad (4.2.5)$$

We can combine these $N + 1$ equations into a matrix equation of the form $\mathbf{A} \vec{x} = \vec{b}$

$$\begin{bmatrix} R_0 & 0 & \cdots & 0 & 1 \\ 0 & R_1 & \cdots & 0 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & R_{N-1} & 1 \\ 1 & 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathcal{I}_0 \\ \mathcal{I}_1 \\ \vdots \\ \mathcal{I}_{N-1} \\ V \end{bmatrix} = \begin{bmatrix} \mathcal{E}_0 \\ \mathcal{E}_1 \\ \vdots \\ \mathcal{E}_{N-1} \\ 0 \end{bmatrix} \quad (4.2.6)$$

Here, the unknown vector \vec{x} consists of the N currents passing through the branches of the circuit, and the potential V .

4.2.1 Direct Solution

Faced with a system of linear equations, one's first instinct is usually to solve for \vec{x} by inverting the matrix \mathbf{A} :

$$\mathbf{A} \vec{x} = \vec{b} \quad \Rightarrow \quad \vec{x} = \mathbf{A}^{-1} \vec{b}. \quad (4.2.7)$$

Don't do this. It is mathematically correct, but numerically inefficient. As we'll see, computing the matrix inverse \mathbf{A}^{-1} , and then right-multiplying by \vec{b} , involves more steps than simply solving the equation directly

To solve a system of linear equations, use the `solve` function from the `scipy.linalg` module. (You will need to import `scipy.linalg` explicitly, because it is a submodule of `scipy` and does not get imported by our usual `from scipy import *` statement.) Here is an example:

```
>>> A = array([[1., 2., 3.], [2., 4., 0.], [1., 3., 9.]])
>>> b = array([6., 6., 9.])
>>>
>>> import scipy.linalg as lin
>>> x = lin.solve(A, b)
>>> x
array([ 9., -3., 1.])
```

We can verify that this is indeed the solution:

```
>>> dot(A, x)           # This should equal b.
array([ 6.,  6.,  9.])
```

The direct solver uses an algorithm known as Gaussian elimination, which we'll discuss in the next article. The runtime of Gaussian elimination is $O(N^3)$, where N is the size of the linear algebra problem.

The reason we avoid solving linear equations by inverting the matrix \mathbf{A} is that the matrix inverse is itself calculated using the Gaussian elimination algorithm! If you are going to use Gaussian elimination anyway, it is far better to apply the algorithm directly on the desired \mathbf{A} and b . Solving by calculating \mathbf{A}^{-1} involves about twice as many computational steps.

This page titled [4.2: Linear Equations](#) is shared under a [CC BY-SA 4.0](#) license and was authored, remixed, and/or curated by [Y. D. Chong](#) via [source content](#) that was edited to the style and standards of the LibreTexts platform.