TJTMI Issue 2, Spring 2023, pp. 29–83

From Rotations and Groups to Tensor Networks

E. Miles Stoudenmire, PhD
Research Scientist, Center for Computational Quantum Physics
Flatiron Institute, New York, NY 10010

Abstract: Even the simplest mathematical concepts can have surprisingly deep struc- ture and hidden connections to other topics. In this article we explore rotations of two- dimensional objects. After viewing rotations first concretely as small matrices, then more abstractly as members of a "group", we consider how rotations transform simple functions (x, y, $x^2 - y^2$, etc.) and their products. Transforming products of three or more functions is best captured by introducing multi-dimensional generalizations of vectors and matrices known as tensors and chaining these tensors together. The resulting object known as a tensor train (or matrix product state) is one of the most powerful computational tools for applications such as modeling fluid flows, understanding properties of metals and crystals, predicting financial markets, and simulating quantum computers.

1

Introduction

Sometimes in mathematics, ideas that initially seem simple may have much more to them than meets the eye. In this article, we will see that "rotations" are one such example.

The natural world has a surprisingly economical structure. The more we understand the laws of physics, the structure of atoms, or even patterns of language, the more we encounter recurring mathematical structures and connections between fields previously thought to be unrelated. The physicist Eugene Wigner remarked on this in his celebrated article "The Unreasonable Effectiveness of Mathematics in the Natural Sciences" [Wig90].

In this article we will explore the structure of **rotations**, specifically two-dimensional rotations. At first, there does not seem to be too much to say about them. An object turns by some angle then after 360 degrees is back to where it started, like a record on a turntable. But rotations turn out to be surprisingly rich. Imagine if the record was painted with multicolored stripes and we wanted to know how far it could rotate before the stripes repeated.

The structure grows deeper if we rotate multiple objects at once. Rotations of composite objects can be "classified" by taking sums and differences of the "rates" of rotations of the simpler objects forming them, like the "wheels within wheels" of the Talking Heads song "Slippery People" or the vision of the prophet Ezekiel.

We can try to make sense of the increasingly complex structure of rotations using vectors and matrices, but ultimately it becomes easier to introduce **tensors**—higher dimensional generalizations of vectors and matrices. To organize rotating objects, we will find that the tensors compose into a chain-like or tree-like structure known as a **tensor train** which is a structure that appeared in mathematics only in the last decade or so, but is now underpinning powerful computational techniques across scientific fields as wide-ranging as fluid dynamics, quantum physics, language modeling, and economics. Tackling the seemingly humble subject of rotations forces this more powerful structure into our hands.

2

Motivation

Rotations and their symmetries are studied by both mathematicians and physicists and have a number of real-world applications that range from quantum mechanics to machine learning. This section gives a brief overview.

Why would we want to know how collections of objects rotate? Rotational symmetries and other symmetries have become central to physical theories and our modern understanding of the natural world. The most profound example might be Einstein's relativity where symmetries between observers moving at different velocities connect the concepts of space and time.

Symmetry plays a central role in the physics of the quantum world too. Quantum mechanics explains the behavior of microscopic objects like atoms in terms of wave-like mathematical equations. Early experiments on pure gases consisting of a single type of atom showed that pumping energy into the gas only released light at specific, discrete wavelengths or colors. The reason for the particular colors observed can be traced back in part to the **rotational symmetries** of the electron waves surrounding the atomic nuclei. Later in this article we will encounter the idea of objects rotating under specific symmetry "representations" labeled by integers—the discrete atomic gas colors observed by early 20th century scientists depends on these integer labels (adapted to the case of three-dimensional rotations). Today similar quantum mechanical effects are used to build advanced "quantum computers" where the building blocks are puddles of electrons rotating in different directions [KYG+07, AAB+19]. And quantum effects can be used to understand atmospheres of distant planets and stars by distinguishing the discrete bands of light emitted by electrons orbiting different types of faraway atoms and molecules [RWH13].

At the more speculative end of physics, rotational symmetries have been proposed to explain the structure of spacetime itself, starting from only quantum mechanics, in a theory called **loop quantum gravity** [Per04]. At present, quantum mechanics and gravity have totally different starting points and conceptual foundations, and physicists hope to unify them under a single theoretical framework. Loop quantum gravity hypothesizes the existence of fundamental objects ("spins") obeying quantum mechanics, and argues that the mathematical structure describing all the possible rotationally symmetric combinations of the states of these spins has tantalizing parallels with parts of Einstein's general theory of relativity describing space, time, and gravity. At the end of this article we will encounter a powerful object—the F **symbol**—that explains how the properties of such a spacetime are actually independent of the microscopic details of how we assemble it from the constituent spins, since the quantum states can be consistently transformed into each other.

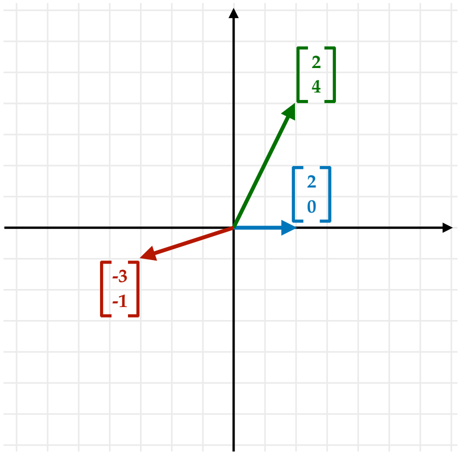
Another quite different field of study where combining functions obeying certain symmetries plays a major role is the field of **machine learning**. Currently a major topic in that field is the development of functions which are powerful enough to perform well on tasks such as image recognition while also transforming in precise ways under rotations and other symmetry groups [KLT18, BAO+20, DWL+21]. The tools we will outline below are essential for such constructions. It has been found that using machine learning models which respect symmetries can lead to superior learning performance, and is a crucial ingredient in the application of machine learning to scientific topics such as the behavior of fluids and gases, such as the behavior of air around the wing of a plane.

3

Rotations in Two Dimensions

Next, we will introduce concepts called vectors and matrices. Then we will see how two-dimensional rotations can be understood by rotating two-dimensional vectors by an angle θ, and how rotations can be formalized by thinking of them as matrices that act on vectors to give rotated vectors.

To start understanding the math of rotations, the simplest setting is two dimensions. In one dimension (1D) you can only go left or right—rotations aren't too interesting in 1D—so two dimensions is really the best starting point. What kind of mathematical objects can be rotated? The traditional example is a **vector**, which is a directed arrow. Here are some examples of vectors in the two dimensional (2D) plane:

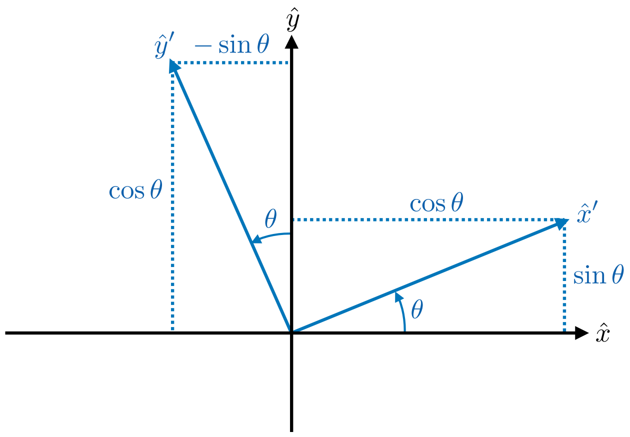

$\begin{bmatrix} 2 \\ 4 \end{bmatrix}$
$\begin{bmatrix} 2 \\ 0 \end{bmatrix}$
$\begin{bmatrix} -3 \\ -1 \end{bmatrix}$

The notation $\left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]$ stands for a vector extending from the origin $(0,0)$ to the point (x,y). We say the first entry is the vector's x coordinate and the second its y coordinate. What we want is a rule that tells us how a rotation acts on one of these vectors. Specifically, if we rotate a vector $\left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]$ by an angle θ, we want to know the new vector $\left[\begin{smallmatrix} x' \\ y' \end{smallmatrix}\right]$. We can think of a rotation as an operation R_θ that acts on one vector to give another, like this:

$$R_\theta \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} .$$

The kind of mathematical objects that multiplies a vector to give a new vector is a **matrix**. So R_θ is a matrix that we are looking to find.(For a review of matrix multiplication, please see Appendix A.)

We can find the entries of R_θ by a graphical argument, using a bit of trigonometry.



In the figure above, we have rotated the x and y axes by an angle θ and used trigonometry to work out the new coordinates x' and y'. More specifically, if $\hat{x} = \left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ is a vector of unit length along the x axis, and $\hat{y} = \left[\begin{smallmatrix}0\\1\end{smallmatrix}\right]$ is a unit vector along the y axis, then, from the above figure, we can see that acting R_θ on $\hat{x}$ maps it to a vector $\hat{x}'$ with entries:

$$\hat{x}' = \begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix} .$$

In other words

$$R_\theta\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}\cos\theta\\ \sin\theta\end{bmatrix} = \cos\theta\begin{bmatrix}1\\0\end{bmatrix} + \sin\theta\begin{bmatrix}0\\1\end{bmatrix} = \cos\theta\ \hat{x} + \sin\theta\ \hat{y} .$$

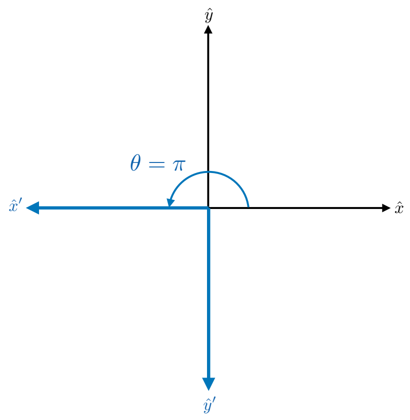
By similar reasoning about $\hat{y}$ and $\hat{y}'$ also shown in the figure,

$$R_\theta\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}-\sin\theta\\ \cos\theta\end{bmatrix} = -\sin\theta\ \hat{x} + \cos\theta\ \hat{y} .$$

This is now enough information to determine that R_θ is the matrix

$$R_\theta = \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix} . \tag{1}$$

(For more information about why knowing the action of R_θ on $\hat{x}$ and $\hat{y}$ determines this matrix, see the end of Appendix A.) Let's do an example to check our formula in Equation (1). If we take $\theta = \pi$ then we have the following rotation:



and if we use the matrix R_π from Equation (1), using $\cos \pi = -1$, $\sin \pi = 0$ we find:

$$R_\theta = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

so that

$$R_\theta \hat{x} = R_\theta \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \end{bmatrix} = -\hat{x}$$

$$R_\theta \hat{y} = R_\theta \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} = -\hat{y}$$

In other words the rotation R_π sends $\hat{x}$ and $\hat{y}$ to their negatives, which matches up exactly with the result shown in the figure above.

4

Rotations as a Group

As we will see in this section, rotations behave like "numbers" that can be "multiplied" together, but do not have to follow the usual rules of multiplication. This kind of generalization of numbers is called a group. Two-dimensional rotations form a group whose elements can be composed or multiplied by adding their angles modulo 2π, and multiplying the matrices that represent the rotations enacts the same multiplication rules as the abstract group elements.

The Notion of a Group

Let's go one rung up the ladder of abstraction for a moment. Rotations are a key example of a structure in mathematics called a **group**. A group is like an "alternate universe" of number-like objects that we can compose to make other objects in the group, but which are different from the usual numbers we use every day.

More specifically, a group is a set of objects g called the group elements. The group as a whole is called G. The group comes equipped with an operation[1] notated as " $\cdot$ " which composes two group elements to make another one:

$$g_1 \cdot g_2 = g_3$$

so that g_3 is also in G. Crucially, for G to be a group there has to be a unique group element called the **identity** notated e which has no effect on other group elements

$$g \cdot e = g$$

For every element a in G there also has to exist an element b in G that composes with a to go back to the identity e, in other words

$$a \cdot b = e.$$

One says that b is the inverse of a.

The most familiar example of a group is the set of real numbers that we use everyday for measuring things. Two real numbers r_1 and r_2 can be composed by multiplying them: $r_1 \cdot r_2 = r_3$. The identity of this group is the number 1 that is, $e = 1$. Every element a has a multiplicative inverse $b = 1/a$ such that $a \cdot b = a/a = 1$. The only exception is the number 0, so only the non-zero real numbers form a group under multiplication. (Real numbers are also a group under addition, and this double-group structure plus a few other properties makes the reals not only a group but a "field.")

[1] Technically, this operation must also be associative for G to be a valid group.

Composing Rotations

Now that we know what a group is, we can check that the two-dimensional rotations $\mathcal{R}_\theta$ are a group. Note that we are using the notation $\mathcal{R}_\theta$ to refer to an abstract object representing a rotation, versus R_θ which was a specific 2×2 matrix. To make rotations $\mathcal{R}_\theta$ into a group, we take the composition rule " $\cdot$ " to be that we add their angles. In other words

$$\mathcal{R}_\theta \cdot \mathcal{R}_\phi = \mathcal{R}_{\theta+\phi}.$$

Under this rule, the identity element must be $e = \mathcal{R}_0$ because

$$\mathcal{R}_\theta \cdot \mathcal{R}_0 = \mathcal{R}_{(\theta+0)} = \mathcal{R}_\theta.$$

The inverse of a group element $\mathcal{R}_\theta$ is given by $\mathcal{R}_{-\theta}$ which we can easily check because

$$\mathcal{R}_\theta \cdot \mathcal{R}_{-\theta} = \mathcal{R}_{(\theta-\theta)} = \mathcal{R}_0 = e.$$

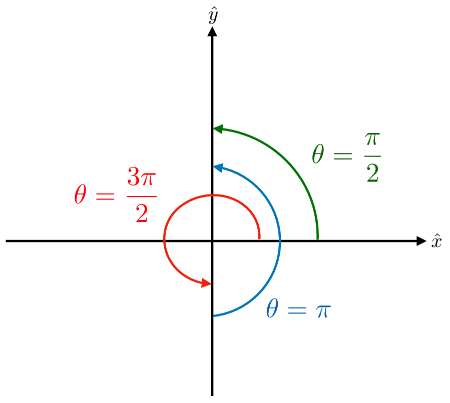
After doing all this work to formalize how rotations compose, only to find that their angles just add, could we have not just talked about regular addition of the angles and skipped the business about groups? A key point is that while small rotations add like regular numbers—their angles just adding together to make bigger angles—once the angles become too large the addition goes around the circle and results in small angles again. For example,

$$\mathcal{R}_\pi \cdot \mathcal{R}_\pi = \mathcal{R}_0$$

because rotating by π or 180 degrees two times is the same as no rotation at all. Another example is

$$\mathcal{R}_{3\pi/2} \cdot \mathcal{R}_{\pi} = \mathcal{R}_{\pi/2}$$

which we can visualize as



whereas if we just naively added the angles we would get $3\pi/2 + \pi = 5\pi/2$. But note that $5\pi/2 = \pi/2 + 2\pi$. So really the correct composition rule for rotations is that we must add the angles **modulo** 2π since a rotation by an angle $\theta + 2\pi$ is exactly the same as a rotation by just θ. That is why we say rotations are an example of a group. They mostly act like numbers in that they can be added together, but they are different from the usual numbers in that the addition is modulo 2π.

Representing the Rotation Group with Matrices

Recall how in the previous section we had worked out a concrete matrix for a rotation. How does the matrix form R_θ of a rotation connect to the more abstract concept of rotations forming a group? The answer is that group composition $\mathcal{R}_\theta \cdot \mathcal{R}_\phi$ can be enacted by mapping the group elements $\mathcal{R}_\theta$ to matrices R_θ and multiplying the matrices. To check this, we can roll up our sleeves and do some computation.

$$\mathcal{R}_\theta \cdot \mathcal{R}_\phi \to R_\theta R_\phi = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

$$= \begin{bmatrix} (\cos\theta\cos\phi - \sin\theta\sin\phi) & (-\cos\theta\sin\phi - \sin\theta\cos\phi) \\ (\sin\theta\cos\phi + \cos\theta\sin\phi) & (-\sin\theta\sin\phi + \cos\theta\cos\phi) \end{bmatrix} \tag{2}$$

$$= \begin{bmatrix} \cos(\theta+\phi) & -\sin(\theta+\phi) \\ \sin(\theta+\phi) & \cos(\theta+\phi) \end{bmatrix} \to \mathcal{R}_{\theta+\phi} \tag{3}$$

Going from Equation (2) to (3) above requires using the following "trigonometry identities" for adding angles which you may or may not remember from high school.

$$\cos\theta\cos\phi - \sin\theta\sin\phi = \cos(\theta+\phi)$$

$$\sin\theta\cos\phi + \cos\theta\sin\phi = \sin(\theta+\phi)$$

The upshot is that we can see from Equation (3) that the matrix corresponding to $R_{\theta+\phi}$ is exactly the matrix we get when we define $\mathcal{R}_\theta \cdot \mathcal{R}_\phi$ to be the rule that we multiply the corresponding matrices. This correspondence can be summarized by saying that the rotation matrices

$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

represent the two-dimensional rotation group (or form a **representation** of the group). They are matrices that behave in every way like the abstract group elements themselves. This includes the addition-modulo-2π property since $\cos(\theta + 2\pi) = \cos\theta$ and similarly for $\sin\theta$.

But are these the only matrices that behave like the group elements? We explore that question in the next section.

(Optional) Advanced Information About the 2D Rotation Group

As a brief detour into terminology which you may see in mathematics or physics literature, note that the group of 2D rotations is important enough to have an official name. This name is $SO(2)$, which stands for the **special orthogonal group in two dimensions**. The term **special** refers to the matrices representing the group elements, and means that each of these matrices has a determinant of $+1$. Among other things, this means the matrices are "pure" rotations and do not include a reflection across one of the axes. The term **orthogonal** says that the matrices making up the group have rows and columns that are orthogonal to one another. We can understand intuitively why the columns are orthogonal using the fact that the columns are what matrices map $\hat{x}$ and $\hat{y}$ onto. The vectors $\hat{x}$ and $\hat{y}$ start out with a 90 degree angle between them, and rotations cannot change this angle.

5

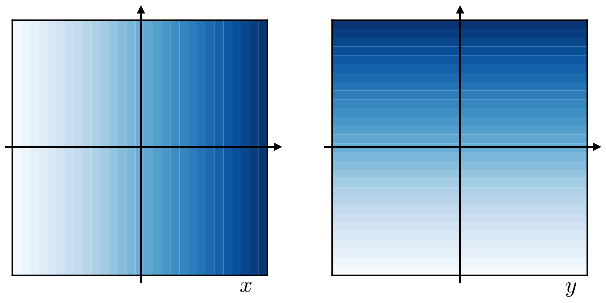
Higher Representations

In this section, we will explore how certain functions transform under rotations. Some transform similarly to vectors $\hat{x}$ and $\hat{y}$, but others transform differently with faster rates of rotation. Objects (such as collections of functions) that rotate into each other under faster rotation rates form "higher" representations of the rotation group. This section will show how the matrices which rotate them have entries like $\cos(n\theta)$, $\sin(n\theta)$ where n is the rate of rotation.

Rotating Functions

In the previous section we saw that rotations taken together form a group and that each rotation is in one-to-one correspondence with or *represented* by a 2×2 matrix R_θ. What's more, multiplying the R_θ matrices together gives the same result as composing the corresponding *abstract* group elements $\mathcal{R}_\theta$ labeled by angles. Is that the end of the story, that 2D rotations really "are" these 2×2 matrices which rotate 2-dimensional vectors and that's all there is to it?

To see that the story is actually richer, let's consider how *functions* behave under rotations. Two of the simplest functions we can write down in two dimensions are $f(x, y) = x$ and $g(x, y) = y$ which, at each point (x, y), tell us either how large the x coordinate is or how large the y coordinate is. We will simply denote each function by x and y, respectively, and we can visualize these functions as follows,



where in the plots above, the darker color indicates a larger (more positive value) of the function and the lighter color indicates a smaller (more negative) value of the function. The function x looks like a linear ramp increasing from left to right, and y a linear ramp increasing from bottom to top.

From the figures, we can see that evidently y is the function you get if you rotate x by 90 degrees ($\frac{\pi}{2}$ radians), in other words

$$\mathcal{R}_{\frac{\pi}{2}}\, x = y. \tag{4}$$

Likewise, if you rotate x by 180 degrees (π radians) then

$$\mathcal{R}_{\pi}\, x = -x. \tag{5}$$

In fact for more general rotations, the function x transforms exactly the same way as the vector $\hat{x}$,

$$\mathcal{R}_\theta \, x = \cos\theta \, x + \sin\theta \, y \tag{6}$$

and similarly the function y transforms like $\hat{y}$,

$$\mathcal{R}_\theta \, y = -\sin\theta \, x + \cos\theta \, y. \tag{7}$$

For a longer discussion of why the above general transformations (6) and (7) hold, see Appendix B for a derivation. We can see that Equations (4) and (5) are special cases of these formulas by plugging in either $\theta = \frac{\pi}{2}$ or $\theta = \pi$.

So much for the functions x and y—while it's interesting that they transform just like the vectors $\hat{x}$ and $\hat{y}$, it's not too interesting. What about more complicated functions?

Rotating Harmonic Polynomials of Degree Two

An important class of functions whose rotation properties we might want to know are the **harmonic polynomials**. These are functions like xy, $x^3 - 2xy^2$, and so on which have the property that their second derivative is zero. Physically, for a function to have zero second derivative means it describes the density of a fluid, or some other "stuff," where the amount of stuff going into any small region is the same as the amount leaving the region. (If you aren't familiar with the notion of a second derivative, which is a concept from calculus, it's not at all required for what follows—instead you can think of harmonic polynomials such as xy or $x^3 - 2xy^2$ as a special set of polynomials tabulated by scientists for having certain desirable properties.) The harmonic polynomials play a central role in physics because they are solutions to Laplace's equation, a well-known equation governing the steady-state distribution of heat in a material, and to Maxwell's equations describing the behavior of electric and magnetic fields.

Let's consider harmonic polynomials of degree two (meaning the total of all the exponents in any given term sums to 2), specifically the functions

$$x^2 - y^2$$
$$2xy.$$

Any other harmonic polynomial of degree two can be written as a linear combination of these two functions. (A "linear combination" is a phrase used to refer to a sum or difference of these two functions, after possibly multiplying one or both of them by a real number. The process of taking a second derivative distributes over addition and multiplication by scalars, so linear combinations of harmonic polynomials are still harmonic.) The extra factor of 2 in front of the second function will be important, because we will find that certain rotations turn $(x^2 - y^2)$ into $(2xy)$ and not just (xy).

How do these functions transform under the same rotations, $\frac{\pi}{2}$ and π, we considered for x and y above? To work out the transformation properties, we can "distribute" the rotation operation onto the individual x and y factors like this:

$$\mathcal{R}_\theta(x^2 - y^2) = (\mathcal{R}_\theta x)(\mathcal{R}_\theta x) - (\mathcal{R}_\theta y)(\mathcal{R}_\theta y)$$

(More correctly, we are not really "distributing" here but using the coordinate substitution approach laid out in Appendix B.)

First considering $\theta = \frac{\pi}{2}$, and using our previous results for how x and y transform, we find

$$\begin{aligned}\mathcal{R}_{\frac{\pi}{2}}(x^2 - y^2) &= (\mathcal{R}_{\frac{\pi}{2}} x)(\mathcal{R}_{\frac{\pi}{2}} x) - (\mathcal{R}_{\frac{\pi}{2}} y)(\mathcal{R}_{\frac{\pi}{2}} y) \\ &= (y)(y) - (-x)(-x) \\ &= -(x^2 - y^2).\end{aligned}$$

Likewise we find that

$$\begin{aligned}\mathcal{R}_{\frac{\pi}{2}}(2xy) &= 2(\mathcal{R}_{\frac{\pi}{2}} x)(\mathcal{R}_{\frac{\pi}{2}} y) \\ &= 2(y)(-x) \\ &= -(2xy).\end{aligned}$$

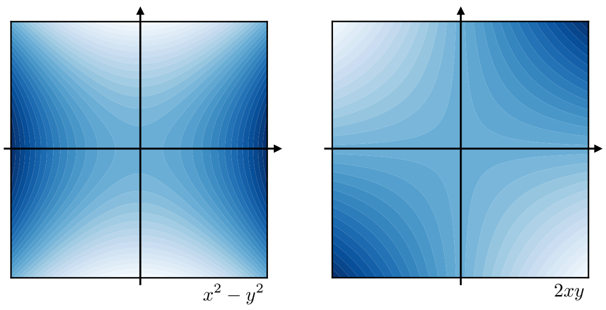
Under a 90-degree rotation both of these polynomials transform into their negative. This is quite different from how x and y transform. Recall that x and y, as well as vectors like $\hat{x}$ or $\hat{y}$ (or actually *any* 2D vector of coordinates), only transformed into their negative after a 180-degree rotation. So the degree-two harmonic polynomials must have a different rotation rule from x and y.

Consider also a rotation by π, where we find by a similar computation that

$$\mathcal{R}_\pi(x^2 - y^2) = +(x^2 - y^2)$$
$$\mathcal{R}_\pi(2xy) = +(2xy).$$

Under a 180-degree rotation these functions are already back to their original value, whereas for vectors and functions like x and y this would only happen after 360-degree (2π) rotation. Apparently the functions $x^2 - y^2$ and $2xy$ rotate *twice as quickly* as the functions x and y.

To gain an intuition of why the functions $x^2 - y^2$ and $2xy$ rotate into each other and make a full rotation after just an angle of 180 degrees, it is helpful to plot them:



where again the darker color indicates a larger (more positive value) of the function and the lighter color indicates a smaller (more negative) value of the function.

By inspecting the plots above, one can see that the functions have the same shape, but one is just rotated by 45 degrees to the other. Furthermore, they have a kind of an "airplane propeller" shape to them that has more symmetry (twice as much, in a sense) as the functions x and y we plotted at the beginning of this section.

To formalize the rotation properties of $x^2 - y^2$ and $2xy$ we can put them into a vector like this

$$\begin{bmatrix} x^2 - y^2 \\ 2xy \end{bmatrix}$$

and find the matrix which appropriately rotates this vector. The answer one finds for this matrix is

$$R_\theta^{[2]} = \begin{bmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \cos(2\theta) \end{bmatrix}.$$

This matrix can be worked out by applying a rotation to $x^2 - y^2$ and xy with general θ, distributing the rotation over individual factors of x and y as in the examples above, and using trigonometry identities to simplify and collect terms. It is a straightforward but tedious calculation and is outlined in Appendix C. Note the superscript "[2]" we have put on $R_\theta^{[2]}$ to distinguish it from $R_\theta^{[1]} = R_\theta$, where

$$R_\theta^{[1]} = R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

was the matrix we found for rotating vectors in previous sections. We can see that $R_\theta^{[2]}$ only depends on 2θ and not on θ alone, which directly says that $R_\theta^{[2]}$ carries out rotations twice as quickly as $R_\theta^{[1]}$.

What have found here is a second *representation* of the 2D rotation group that is distinct from the $R_\theta^{[1]}$ representation. We have found another set of matrices $R_\theta^{[2]}$ that obey all of the abstract properties of the rotation group, but are different from the $R_\theta^{[1]}$ matrices in that they rotate the vectors or functions they act on twice as quickly. The $R^{[2]}$ representation is called a "higher" representation than $R^{[1]}$ to indicate that objects rotate more quickly under it (and also because the integer $n = 2$ "labeling" or "classifying" this representation is larger than the $n = 1$ label of $R^{[1]}$).

We see a more interesting story is developing for 2D rotations. Apparently there are multiple kinds of objects that can be rotated in a 2D plane. Some rotate at a rate of 1, others at a rate of 2. Are there objects that rotate at a rate of 3, 4, or 5? How about a rate of 0?

6

Combining Representations

Now we will describe how starting from functions like x_1, y_1, x_2, and y_2, which transform at a rate of 1, we can form special combinations that rotate under well-defined rates of 2 and 0. In this section, we will show that we can organize these combined functions into a combined representation made out of matrices of single-rate representations. We will also see that the functions with different rates do not mix together.

Rotating Combinations of Functions

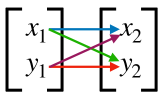
Now that we have seen that there is more than one way mathematical objects can transform under 2D rotations and that there are **higher** representations of the 2D rotation group, it's interesting to ask if there is a systematic way to build these higher representations out of lower ones. Namely, how do functions which transform according to one representation transform after they are combined by being multiplied together?

To motivate why someone would care about a question like this, an actual application of functions of this type is in the field of **data science** or **machine learning** where a major goal is to program computers to perform automated tasks on images such as photographs. Various parts or "features" of images can be filtered into different channels and organized by their symmetries, which can have benefits like removing noise from images (since noise would have very little symmetry) or classifying images irrespective of their orientation. In a bit more detail, a grayscale image can be viewed as a function $f(x, y)$ that takes the coordinates of a pixel as input, and outputs the brightness of that pixel. This "image function" can be then written as a sum of other functions, such as x, y, x^2, y^2, etc. If one wants to keep only information about the image that does not depend on which way you rotate the image, this corresponds to selecting only functions in the sum which also do not change under rotations.

Another motivation is the field of chemistry. If you remember in high school learning to count electron shells of atoms by saying "$1s^2, 2s^2, 2p^6, \ldots$" you were actually using group representation theory whether you knew it or not. Electron shell symmetry is at the root of the field of chemistry and explains why the periodic table has the structure it does, why some materials are metals, and why the noble gases do no react with other elements. (Though it's important to note electron shell structure is based on 3D rotations, versus the 2D rotations we are studying here.)

In this section we are seeking a more systematic approach to building functions that transform under rotations according to higher-order representations of the rotation group. To get started, first observe that all degree-one polynomials involving the variables x and y (polynomials where x and y only appear to the first power) can be built by taking linear combinations of the functions x and y themselves. For example, we could make the linear combinations $3x + 2y$ or $-5x + 10y$.

Similar "building blocks" for all degree two polynomials can be assembled in the following way: make two vectors containing x_1 and y_1 or x_2 and y_2 then "multiply" the entries of these vectors in all possible ways.



$x_1x_2 \quad x_1y_2 \quad y_1x_2 \quad y_1y_2$

By taking combinations of these, such as $-3x_1x_2 + 7x_1y_2 + 4y_1y_2$, we can obtain every degree-two polynomial made of products of the variables x_1, x_2, y_1, and y_2.

Here we are being informal about the definition of what it means to multiply two vectors, but intuitively it means just what the figure shows above—take each element of the first vector and multiply it by each element of the second vector. Note that the entries of each vector in the product above are functions such as x_1 and y_1 where the subscripts 1 or 2 say which vector they came from. We can either think of this as a notational convenience or we can think of building higher-dimensional functions that take two sets of coordinates as input (the input being (x_1, y_1, x_2, y_2)).

Next, we can find special linear combinations of these degree-two building blocks that rotate by a well-defined rate, such as at a rate of 1 or a rate of 2. Two of these combinations are analogous to the functions we studied in the previous section. They are

$$f_1 = x_1x_2 - y_1y_2$$

$$f_2 = x_1y_2 + y_1x_2$$

where we are now giving them the names f_1 and f_2 to help keep track of things. (It would be more correct to write $f_1(x_1, y_1, x_2, y_2)$ and $f_2(x_1, y_1, x_2, y_2)$ but we will leave out the arguments when they can be understood from the context.) If we erase the subscripts on the variables ($x_1 \to x$, $x_2 \to x$, etc.), notice how we get the functions $x^2 - y^2$ and $2xy$ which are the ones we studied previously. Recall from before that they rotate into each other under 2D rotations and do so at a rate of 2, because after an angle of just π they already returned back to their original values.

What our more systematic approach lets us do now is look for the remaining functions we can make out of the products x_1x_2, x_1y_2, y_1x_2, and y_1y_2. We have four products of variables and have found two functions, so only two more "independent" functions remain. They are

$$f_3 = x_1x_2 + y_1y_2$$

$$f_4 = x_1y_2 - y_1x_2.$$

By independent, we mean that by making linear combinations of f_1, f_2, f_3, f_4 we can get any degree-two polynomial. For example, $x_1x_2 = f_1 + f_3$. Let's check how the new functions f_3 and f_4 we haven't studied yet rotate using the techniques of the previous section, checking a few specific angles.

First the function $f_3 = x_1x_2 + y_1y_2$ rotates under an angle $\frac{\pi}{2}$ as

$$\begin{aligned}\mathcal{R}_{\frac{\pi}{2}}(x_1x_2 + y_1y_2) &= (R_{\frac{\pi}{2}}x_1)(R_{\frac{\pi}{2}}x_2) + (R_{\frac{\pi}{2}}y_1)(R_{\frac{\pi}{2}}y_2)\\ &= (y_1)(y_2) + (-x_1)(-x_2)\\ &= x_1x_2 + y_1y_2.\end{aligned}$$

It *doesn't change at all* under this rotation. We can also check the angle π:

$$\begin{aligned}\mathcal{R}_{\pi}(x_1x_2 + y_1y_2) &= (R_{\pi}x_1)(R_{\pi}x_2) + (R_{\pi}y_1)(R_{\pi}y_2)\\ &= (-x_1)(-x_2) + (-y_1)(-y_2)\\ &= x_1x_2 + y_1y_2\end{aligned}$$

and we find no change under this angle either.

If we do a more general calculation involving an arbitrary angle and using various trigonometry identities—sparing you the details—we will find a similar result for any angle whatsoever. The function $f_3 = x_1x_2 + y_1y_2$ is *invariant* under rotations. It always transforms back into itself. The function $f_4 = x_1y_2 - y_1x_2$ behaves in a similar way. For example, rotating it by the angle $\frac{\pi}{2}$ gives

$$\begin{aligned}\mathcal{R}_{\frac{\pi}{2}}(x_1y_2 - y_1x_2) &= (R_{\frac{\pi}{2}}x_1)(R_{\frac{\pi}{2}}y_2) - (R_{\frac{\pi}{2}}y_1)(R_{\frac{\pi}{2}}x_2)\\ &= (y_1)(-x_2) - (-x_1)(y_2)\\ &= x_1y_2 - y_1x_2\end{aligned}$$

which shows that f_4 rotates back to itself unchanged. This rotational invariance can be shown to hold for any choice of angle using some tedious but straightforward trig.

Combining Representations

What happens if we combine f_1, f_2, f_3, f_4 together into a big vector and perform a rotation? How does this vector of functions transform? Starting again with a concrete example, let's take the angle $\theta = \frac{\pi}{2}$. We have already argued above that under a $\frac{\pi}{2}$ rotation,

$$\mathcal{R}_{\frac{\pi}{2}} f_1 = -f_1$$

$$\mathcal{R}_{\frac{\pi}{2}} f_2 = -f_2$$

while

$$\mathcal{R}_{\frac{\pi}{2}} f_3 = f_3$$

$$\mathcal{R}_{\frac{\pi}{2}} f_4 = f_4.$$

Putting these results together gives the matrix equation

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} -f_1 \\ -f_2 \\ f_3 \\ f_4 \end{bmatrix}.$$

A more interesting case is the angle $\frac{\pi}{4}$. From the rotation properties of the factors x_1, y_1, x_2, y_2 individually, one can show that

$$\mathcal{R}_{\frac{\pi}{4}} f_1 = f_2$$

$$\mathcal{R}_{\frac{\pi}{4}} f_2 = -f_1$$

while

$$\mathcal{R}_{\frac{\pi}{4}} f_3 = f_3$$

$$\mathcal{R}_{\frac{\pi}{4}} f_4 = f_4$$

where again recall f_3 and f_4 transform into themselves no matter the angle. The matrix version of this result is

$$\begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} f_2 \\ -f_1 \\ f_3 \\ f_4 \end{bmatrix}.$$

Notice how compared to the previous angle, the upper-left corner of the matrix is different while the lower-right corner is the same. By inspecting these matrix examples we can conjecture that the general matrix expression for how $[f_1\ f_2\ f_3\ f_4]$ transforms is going to look like this:

$$\begin{bmatrix} \cos 2\theta & -\sin 2\theta & 0 & 0 \\ \sin 2\theta & \cos 2\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} f_1' \\ f_2' \\ f_3' \\ f_4' \end{bmatrix}$$

We already knew that f_1 and f_2 transform into each other by the $R_\theta^{[2]}$ matrix representation, and in this section we've argued that f_3 and f_4 always transform into themselves. We can call this new "trivial" representation $R_\theta^{[0]}$. As a matrix, it is just the matrix

$$R_\theta^{[0]} = \begin{bmatrix} 1 \end{bmatrix}.$$

So the whole matrix for transforming $[f_1\ f_2\ f_3\ f_4]$ can be written schematically as

$$\begin{bmatrix} R_\theta^{[2]} & 0 & 0 \\ 0 & R_\theta^{[0]} & 0 \\ 0 & 0 & R_\theta^{[0]} \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} f_1' \\ f_2' \\ f_3' \\ f_4' \end{bmatrix}$$

where it is understood that $R_\theta^{[2]}$ a 2×2 "block" of the matrix, while $R_\theta^{[0]}$ is a 1×1 block, and the zeros can stand for whole blocks of zeros.

Section Summary

As we conclude this section, let's reflect on what has happened here. We started with simpler functions x_1, y_1, x_2, y_2 that individually transform under rotations according to the $R_\theta^{[1]}$ matrix (the rate 1 representation). After multiplying them together in all possible ways, we were able to find certain groupings (or linear combinations) of these simpler functions which transform according to the $R_\theta^{[2]}$ and $R_\theta^{[0]}$ representations. Just multiplying two of the original functions, say x_1 and y_2, would give a function $x_1 y_2$ that transforms in a messy, unclear way. But specific linear combinations of the functions transform among themselves in a clean way given by a certain "rotation rate" of 0 or 2. A vector of these special linear combinations is transformed by a matrix which has various representation matrices such as $R_\theta^{[2]}$ or $R_\theta^{[0]}$ down the diagonal of the matrix. The fact that these are on the diagonal means the functions transforming according to each representation do not mix together when rotated.

Mathematicians often summarize the story above by writing the expression $1 \otimes 1 = 2 \oplus 0 \oplus 0$ which means, "The product of two 1 representations can be decomposed into one copy of the 2 representation and two copies of the 0 representation." We started out with two sets of functions that rotate at a rate of 1, and by taking sums and products of them arrived at functions that rotate at a rate of 0 or 2. We can conjecture that multiplying functions with rotation rates m and m' leads to functions with rates of $m - m'$ and $m + m'$.

Looking ahead, what would happen if we took a product of **three** sets of functions, each transforming under the 1 representation? What would be the new representations we could form out of such products? There is a more general structure underlying this story and for that we will need to introduce some better notation and motivate the idea of a **tensor**.

7

Tensors and Tensor Diagrams

This section will introduce tensors, which are higher-dimensional generalizations of vectors and matrices. It is very useful to notate tensors with three or more indices as shapes with lines emanating from them, where the lines are the tensor indices. Connecting an index shared between a pair of tensors means that index is summed over, and the tensors are said be to contracted together. Multiple indices can be summed over simultaneously.

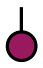
Before we tackle the last topic of this article, which is how three or more rotatable functions combine, we need to upgrade our notation. While vectors and matrices can be written as columns or squares of numbers, we will soon encounter **tensors** which in the same notation would have to be written as a three-dimensional *cube* of numbers. Even worse, higher-order tensors would be four-dimensional or higher "hypercubes" of numbers, so we would have a very hard time writing those.

The kind of notation we are looking for would treat vectors, matrices, and tensors on the same footing, and spare us from having to write down every element inside of a vector or matrix unless we want to. Fortunately just such a notation was created by the physicist and mathematician Roger Penrose in his efforts to understand how Einstein's theory of gravity might emerge from the laws of quantum mechanics [Pen71]. Penrose diagram notation or **tensor diagrams** will be very helpful to us in what follows.

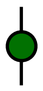
Depicting Tensors (First Rule of Tensor Diagrams)

The first rule of diagram notation is that *vectors, matrices, and tensors are shapes with lines coming out of them.* Each line refers to an **index** which enumerates the entry of the tensor we wish to access or refer to. Vectors have one index, and matrices have two indices, as in the following figures.

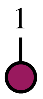
$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \longleftrightarrow$$



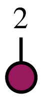
$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \longleftrightarrow$$



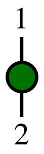
Setting each of the index lines to a specific value retrieves the corresponding element, for example.



$= v_1$

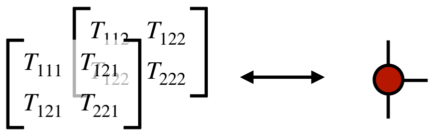


$= v_2$

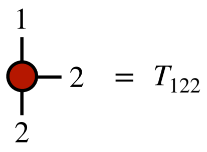


$= M_{12}$

A third-order tensor has three indices and is a cube of numbers or, in diagram notation, a shape with three lines coming out of it.



We can likewise refer to a specific element of this tensor by writing the following.



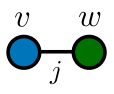
You can see above that notating a third-order tensor as two superposed matrices making a cube would be rather unwieldy, so we will use tensor diagrams as much as we can from now on.

When using diagrams to represent vectors, matrices, or tensors, it doesn't matter which way the lines come out of the tensors, whether upward, downward, to the right, etc.. (In cases where it would be ambiguous, such as distinguishing the row versus column of a matrix, one can adopt a "house" convention or use an asymmetric shape for the tensor.) The key thing is the number of index lines and how they connect to other lines. Note also that vectors and matrices are just considered tensors of order one (vectors) or order two (matrices). So we will use the term "tensor" to refer to a tensor of any order, including a vector or a matrix.

Contracting Tensors (Second Rule of Tensor Diagrams)

The second rule of tensor diagram notation is that *connecting two lines means a sum is performed over those indices*. This kind of connected sum is often referred to as **contracting** two tensors together. Some examples can help clarify what this means.

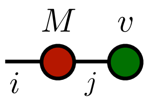
The simplest example of a tensor contraction is summing two vectors v and w on their shared index



$$= \sum_j v_j w_j = v \cdot w$$

where the symbol Σ_j means "sum over all the values j takes." For example if the vectors are $v = \left[\begin{smallmatrix} v_1 \\ v_2 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 1 \\ 2 \end{smallmatrix}\right]$ and $w = \left[\begin{smallmatrix} w_1 \\ w_2 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 3 \\ 4 \end{smallmatrix}\right]$ then the sum above is $v \cdot w = v_1 w_1 + v_2 w_2 = 1 \cdot 3 + 2 \cdot 4 = 11$. This particular operation involving two vectors is so important is has its own name: the dot product or **inner product** of two vectors. Note how the diagram for this operation has no left over index lines sticking out—this tells us that the result must be a number or a "scalar" (or an order-zero tensor, if you will).

Another common operation that can be viewed as a tensor contraction is multiplying a matrix times a vector. We can notate this as

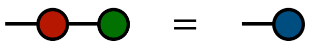


$$= \sum_j M_{ij} v_j$$

If the matrix is $M = \left[\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right]$ and the vector is $v = \left[\begin{smallmatrix} 1 \\ -1 \end{smallmatrix}\right]$ then the above notation is showing the operation

$$Mv = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 1-2 \\ 3-4 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

which we can see results in a vector. For matrix-vector multiplication, the traditional notation of numbers in boxes works perfectly well, and is especially helpful when we care about the details about what specific numbers or variables are in the matrix and vector. But the diagram notation has its own benefits. Just by inspecting the diagram

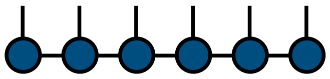


we can deduce that a matrix-vector product involves a sum over one index, while the other index is "free." The result is an object with one free index, so it is a vector.

It is a graphical proof that the product of a matrix times a vector is another vector, a fact commonly taught in an undergraduate linear algebra course.

The Usefulness of Tensor Diagrams

While diagram notation is fun, is it really necessary? One way to answer this question is to show a more "serious" tensor diagram that encodes a structure we will encounter by the end of the next section. Here it is:



Looking at this diagram, it is straightforward to reason about which tensor is contracted with which other tensors and on which indices. In contrast, the same expression written in tradition mathematical notation would be

$$\sum_{\alpha_1,\alpha_2,\alpha_3,\alpha_4,\alpha_5} A^{[1]s_1}_{\alpha_1} A^{[2]s_2}_{\alpha_1\alpha_2} A^{[3]s_3}_{\alpha_2\alpha_3} A^{[4]s_4}_{\alpha_3\alpha_4} A^{[5]s_5}_{\alpha_4\alpha_5} A^{[6]s_6}_{\alpha_5}$$

which I would prefer not to have to write every time compared to the simple diagram above. While the traditional notation has some advantages, it is much more laborious to write and read. Meanwhile, additional details, like the names of the indices, can be optionally included in the diagram notation when they are helpful, but they can also be left out to provide clarity without losing any key information.

For more reading about tensor diagrams, a nice introduction with more examples can be found on the Math3ma Blog [Bra19]. To see how tensor diagrams are used in the wild—mainly in the quantum physics literature so far, though sometimes in the applied math literature—good review articles include references [Orú14] and [BC17].

8

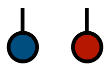
Combining Functions with Tensor Networks

Now that we have a basic understanding of tensors and tensor diagrams, we can use them in this section to help us combine three or more rotatable functions in a principled way. By doing so, the mathematics will naturally connect to two concepts that will be familiar to many physicists, namely "Clebsch-Gordan coefficients" and the "F symbol." At the end of this section, we will see that the F symbol is especially relevant to recent progress in both condensed matter physics and quantum computation.

Notating Combinations of Functions with Tensor Diagrams

Armed with the concept of tensors and tensor diagram notation, we can revisit the story from two sections earlier about combining two sets of functions which transform under rotations in a well-defined way (specific rate of rotation), finding that we can make new combinations that also transform in a well-defined, but different, way. Let's show this process as a tensor diagram. The first ingredient will be two sets of functions that rotate under the $R_\theta^{[1]}$ representation. As before, we will put these functions into vectors and take the product of these vectors:

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} =$$



In diagram notation, putting tensors (in this case vectors) next to each other implies the product introduced at the beginning of Section 6.

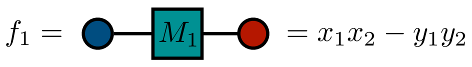
Previously we had found that certain linear combinations of the products of functions transformed under rotations according to the $R_\theta^{[2]}$ or $R_\theta^{[0]}$ representations. These were

$$\begin{aligned} f_1 &= x_1x_2 - y_1y_2 \quad \text{(rotation rate 2)} \\ f_2 &= x_1y_2 + y_1x_2 \quad \text{(rotation rate 2)} \\ f_3 &= x_1x_2 + y_1y_2 \quad \text{(rotation rate 0)} \\ f_4 &= x_1y_2 - y_1x_2 \quad \text{(rotation rate 0)} \end{aligned}$$

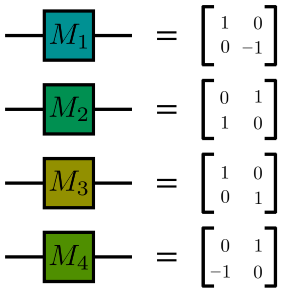
with f_1 and f_2 rotating into each other, enacted by the $R_\theta^{[2]}$ matrix, while f_3 and f_4 rotate back into themselves under any angle. We can think of forming the functions f_1, f_2, f_3, f_4 by making a matrix for each one and contracting it on each side—on the left with $\left[\begin{smallmatrix} x_1 \\ y_1 \end{smallmatrix}\right]$ and on the right with $\left[\begin{smallmatrix} x_2 \\ y_2 \end{smallmatrix}\right]$. To make the function f_1 we can do

$$f_1 = \begin{bmatrix} x_1 & y_1 \end{bmatrix} M_1 \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = x_1x_2 - y_1y_2$$

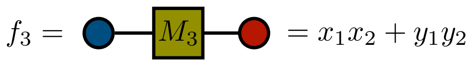
where we have defined a matrix $M_1 = \left[\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right]$. In diagram notation this approach to making f_1 looks like



We can likewise make f_2, f_3, and f_4 by defining corresponding matrices M_2, M_3, and M_4 and contracting them with the original x_1, y_1, x_2, y_2 functions arranged into vectors in a similar way. All the matrices together are

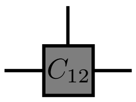

M_1
$= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
M_2
$= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
M_3
$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
M_4
$= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

and we could define f_3, for example, by the following diagram.

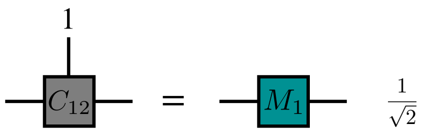


Combiner or Clebsch-Gordan Tensors

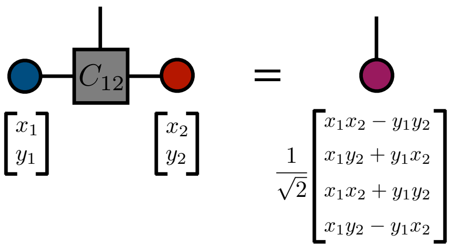
Now we can put our upgraded notation to work for us. Let's define a **tensor** of order three which combines all of these matrices into a single object. Here it is.



You can think of the name C as standing for "combiner" and the 12 subscript as talking about the "spaces" of functions labeled 1 and 2. (The symbol C could also stand for "Clebsch-Gordan" coefficients which is the name of an analogous object used often in the physics and applied math literature.) This tensor—a tensor of order three—lets us organize all of the M matrices we defined above together in the following way. Say we set the top index of C_{12} to the value 1. Then the leftover free indices on the left and right side form a matrix. Choose that matrix to be $M_1/\sqrt{2}$. (The extra factor of $\frac{1}{\sqrt{2}}$ will be helpful for a property we need later.) Diagrammatically this choice looks like:



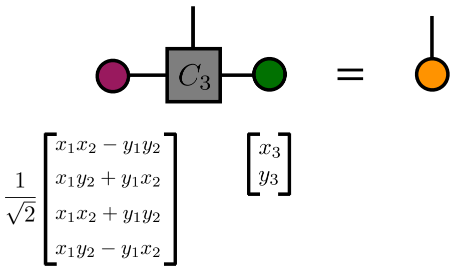
We can similarly choose the remaining entries of C_{12} so that setting the top index to 2, 3, and 4 results in the matrices M_2, M_3, M_4. In this way, we have made a sort of "tensor machine" that will consume simpler functions that originally transformed according to the $R_\theta^{[1]}$ representation (rotating at a rate of 1) and produce new functions that are made from products of these which transform by other well-defined rates. Here is how this whole process looks


C_{12}
=
$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$
$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$
$\frac{1}{\sqrt{2}}\begin{bmatrix} x_1x_2 - y_1y_2 \\ x_1y_2 + y_1x_2 \\ x_1x_2 + y_1y_2 \\ x_1y_2 - y_1x_2 \end{bmatrix}$

The payoff is that now we have a way to visualize and formalize how we would go about combining *any* number of simpler functions into more complex ones, but always such that the more complex functions have well-defined rotation properties.

Networks of Function Combinations

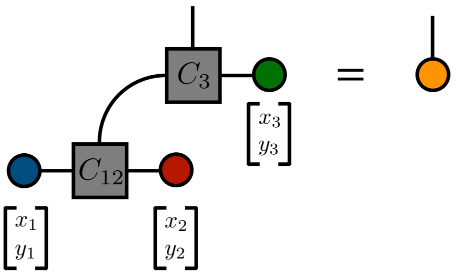
Say we want to take products of *three* sets of variables $[x_1\ y_1]$, $[x_2\ y_2]$, $[x_3\ y_3]$, all which rotate under the $R_\theta^{[1]}$ representation originally, and produce more complicated functions from them. First we can combine the $[x_1\ y_1]$ and $[x_2\ y_2]$ vectors as above using the tensor C_{12}. Then we can make another tensor C_3 that handles the combination of functions transforming under the $R_\theta^{[2]}$ and $R_\theta^{[0]}$ representations, combining them with the $[x_3\ y_3]$ variables, like this:



C_3
=
$\frac{1}{\sqrt{2}}\begin{bmatrix} x_1x_2 - y_1y_2 \\ x_1y_2 + y_1x_2 \\ x_1x_2 + y_1y_2 \\ x_1y_2 - y_1x_2 \end{bmatrix}$
$\begin{bmatrix} x_3 \\ y_3 \end{bmatrix}$

In the figure above, the functions inside the resulting vector are not displayed, but this vector contains eight functions, examples of which are $(x_1x_2 - y_1y_2)x_3 + (x_1y_2 + y_1x_2)y_3$ which has a rotation rate of 1 or $(x_1x_2 - y_1y_2)x_3 - (x_1y_2 + y_1x_2)y_3$ which has rotation rate of 3.

Putting everything together in terms of the original variables, C_{12} and C_3 join forces together as a **network** to process all of the functions from spaces 1, 2, and 3:

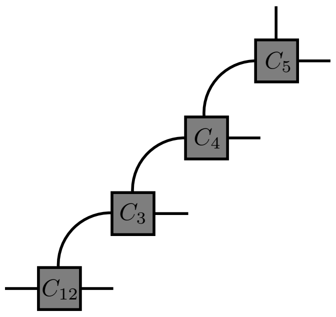


C_3
=
$\begin{bmatrix} x_3 \\ y_3 \end{bmatrix}$
C_{12}
$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$
$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$

Remember that the above diagram is a fully rigorous mathematical equation, not just a drawing or a figure. The left-hand side is a complicated sum, but nothing more than sums and products of functions like x_1, y_2, or x_3, together with some pre-determined numerical coefficients. The right-hand side is a vector containing functions like $(x_1x_2 - y_1y_2)x_3 + (x_1y_2 + y_1x_2)y_3$ which rotate under an angle θ by a specific, well-defined rate.

Larger Networks

What pays off about this way of viewing how functions combine to make well-behaved representations is that it can be shown to work for *any number* of variables. For example, the network for combining five sets of separate functions to make single functions with well-defined rotation properties looks like:

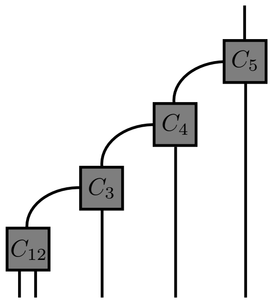


The elements of the C tensors making up such a network can be tabulated once and for all through a straightforward method. For example, tables of these elements for the *three*-dimensional rotation group are published in most quantum physics textbooks since they play a decisive role in that subject.

Thinking of the C tensors by themselves — the backbone of this network — independently of the functions we would attach to it also has many advantages. For example, we could use the same network to combine vectors instead of functions.

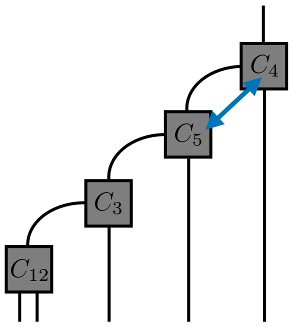
Most interestingly, we can build up a toolbox to answer questions like this: what if we had built up the functions from the individual spaces in a *different order*? In that case, computing the functions mechanically gives results which appear to be different depending on whether we combine $[x_1\ y_1]$ and $[x_2\ y_2]$ first, or combine $[x_3\ y_3]$ and $[x_2\ y_2]$ first. Are the resulting functions really different? It turns out they are not. They are just linear combinations (i.e. sums and differences) of each other. How do we work out these linear combinations? The network of Cs can tell us how.

Say we had combined five sets of functions in the order 1, 2, 3, 4, 5. Then we had done the same combination process but this time put the pair $[x_5\ y_5]$ *before* $[x_4\ y_4]$. We would get different final functions, but can obtain a matrix F which relates one set of functions to the other. This matrix turns out to be equal to a specific tensor network. To obtain this network, first it is helpful to redraw the network of Cs like this:

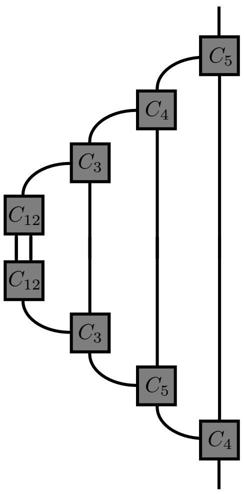


It is the same network as earlier, with precisely the same meaning, just drawn differently. It can be helpful to view is as a huge "matrix" from five two-dimensional spaces (lines entering from the bottom) to a single 2^5 dimensional space (line emanating from the top).

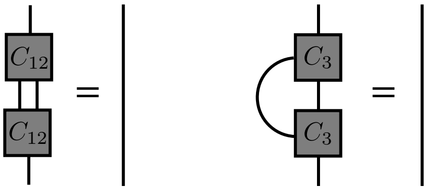
Next, we draw the network that would combine the functions in the other order of 1, 2, 3, 5, 4 (with 4 and 5 permuted).



Then we contract these two networks in the following way

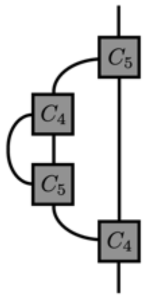


Let's call the above network the **permutation network**. Lastly, we will use a specific property that all of these C tensors have. They are **isometric embeddings** or **isometries** which in this context means that they "cancel" when contracted on their two incoming index lines, like this:

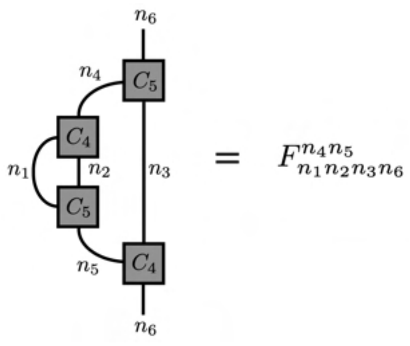


where the plain lines that each pair of tensors "cancels" to can be informally interpreted as saying "connect any lines coming into where this pair of tensors used to be together" or more formally as the pair contracting to make an identity matrix. (By the way, this isometric property is the reason we needed to introduce the extra factor of $1/\sqrt{2}$ when defining C_{12} earlier.)

Using this isometric property to cancel both copies of C_{12} and C_3 from the permutation network, we get its simplified form



which depends only on the tensors C_4 and C_5. If we now explicitly label the rates of rotation n_1, n_2, . . . , n_6 labeling the six kinds of lines in the figure (for technical reasons the outer two must equal, otherwise the value is zero) we get a quantity known as the "*F* **symbol**":



This *F* symbol lets us shortcut making the functions in both of the different orders, and directly tells us how to map from the functions resulting from one ordering to the other. Like the *C* tensors themselves, the *F* symbols can be algorithmically worked out and tabulated for a group like the 2D rotation group and other groups. (As a note to experts: for the case of 3D rotations, the *F* symbols are very closely related to other quantities known alternatively as the "recoupling coefficients" or "Wigner 6–j symbols" [SV12].)

Starting from multiplying and adding simple functions like x_1 or y_2, then just wanting to extend this construction to functions of three variables while preserving well-defined rotational symmetries, we have actually climbed rather high into some advanced topics. The *F* symbols are the basic ingredient in some cutting-edge thinking about physics in recent decades. One area where they come up are in simplified models of "quantum liquids" which are distinct from previously known phases of matter like solid, liquid, or gas. Unlike such conventional, everyday phases, quantum liquids can be "topologically ordered." Recall from Section 2 that quantum mechanics describes particles like electrons as having wave-like properties. Like sound or light waves, electrons can spread out and "explore" different paths as they move through space. The behavior of an electron going from some point A to point B is computed by summing over all the paths it can take. In special settings, these sums over paths can obey "internal symmetries" quite similar (and in some cases literally identical! to the 2D rotational symmetries flowing through the tensor networks above. The amazing part is that the electrons in these liquids can then "melt" together into new kinds of particles (which are vortex-like disturbances of the liquid) whose properties are accounted for by the kind of F symbol describing the "recoupling" or branching of the paths inside the liquid [LW05]. In some cases, these emergent particles can have surprising properties, such as having fractional electric charge or other even more exotic properties, distinct from the particles such as electrons, neutrinos, or quarks making up the standard model of particle physics. They would be a totally new type of fundamental particle that can only exist inside the symmetric liquid. There is a serious ongoing effort to stabilize such particles in the lab, where they could be used to make inherently fault-tolerant quantum computers [Wol18].

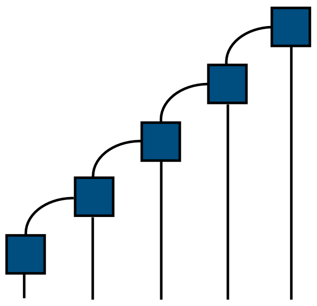
As we end this section, we have only seen a small taste of the subject of group representations acting on functions and their connection to tensor networks, but hopefully some of the main structure has come through. When combining three or more sets of functions that transform in a well-defined way under rotations, the process can be made systematic by viewing it as the contraction of vectors containing the original functions with a "backbone" of tensors making a chain-like network. Computations involving these networks of tensors can be used to systematically work out how combined functions built in different orders relate to one another.

9

MPS or Tensor Train Networks

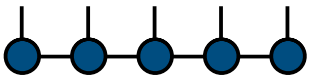
This section concludes with an interesting observation. It turns out that the the network comprised of the "combiner" tensors (that is, the gray, square tensors labeled with Cs) in the previous section is essentially the same as a tensor network that is well-known within quantum physics, called a matrix product state or tensor train. Tensor trains are very efficient at storing information and thus have a wide range of applications, including in modeling fluid flow and in financial modeling. This section gives a brief introduction to tensor trains, their benefits, and some of their applications.

Before concluding the article, we should briefly explore an interesting connection between the group representation theory we have been developing and the topic of tensor networks more broadly. By combining sets of functions to obtain new ones which transform under rotations, we encountered a network of tensors that looked like this:



Compared to the diagram in the previous section, there have been a few alterations (splitting the first tensor into two, removing the extra index from the top) but they are very minor and not important for what we will discuss below.

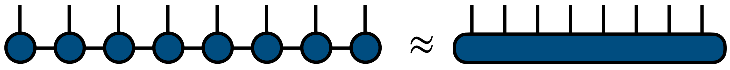
Let's draw the **exact same** structure above but in a slightly different and more standard way:



This network is what is known in the applied math literature as a **tensor train**. Here we have shown a tensor train with five "cores," but they can have any number of cores, including infinitely many. In quantum physics where it is widely used, this structure is known as a **matrix product state** (MPS).

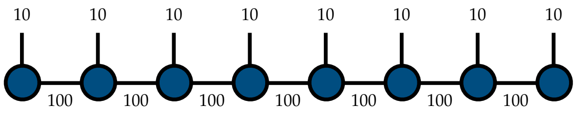
This humble looking network is a powerful tool for an increasingly diverse portfolio of applications, including modeling fluid flow and other processes described by differential equations [GLD$^+$22], simulating (and sometimes beating!) quantum computers [SVS$^+$22], high-performance matrix computations [RO22], and financial modeling [KP22]. If you are interested in using tensor trains or matrix product states, a good way to get started is to use the **ITensor software** [FWS22] which offers tools for instantiating and performing operations on this kind of data structure. As one of the authors of ITensor, the inspiration for this article was actually based on our plans to support a larger set of rotation group symmetries within the ITensor software in the near future. Currently ITensor can handle two-dimensional rotations like the type described in this article, but not yet three-dimensional rotations (or other so-called "non-Abelian" groups) which have a much more complicated structure.

To get a feel for why a tensor train is so powerful, let's do some counting. First of all, a key thing to understand is that collectively the tensors making up the network are implicitly representing a single larger (higher-order) tensor:



The reason for the approximately equals sign (≈) above is that the tensor train on the left can't represent *all* eighth-order tensors on the right, but only a special subset. By putting different numbers into the tensors on the left, one can (implicitly) make an important subset of tensors of the type on the right, but technically there are vastly more eighth-order tensors than eighth-order tensor trains.

But that's ok! The reason is that a tensor train is a **compression format**, similar in spirit to a JPEG image or an MP3 audio file (and even similar in some of the details [ME22]). To get an idea of just how much compression it achieves, let's say the tensor train above has the following dimensions for its indices:



A rough count of the number of parameters needed to store it on a computer would be $100 \times 100 \times 10 \times 8 = 800,000$ parameters (which is not so bad—less than a megabyte). In contrast, storing an entire eighth-order tensor would require $10^8 = 100,000,000$ or a hundred million parameters, which is 125 times as much. If we went to tensor trains with ten indices but the same internal dimensions of 100, this compression ratio would grow to 10,000 (meaning we are using 10,000 less memory to store the same object) and would only continue to improve as we considered ever larger tensors.

In practice, it is not at all an exaggeration to say that tensor train or MPS techniques routinely allow calculations to be performed in minutes or hours that otherwise would take *thousands of years* to perform if the same calculation was done in a naive way.

Conclusions, Connections, and Further Reading

In this article we started with basic notions of how vectors and functions such as x or y change under two-dimensional rotations. Then by asking how more complicated functions—formed out of sums and products of simpler functions—change or transform under rotations we were led to the idea of higher representations and ultimately to tensor networks which are a natural way to organize the way functions combine into other functions with well-defined rotation behavior.

There are many more topics we could have touched on, but which would take another article or two to do. One topic could be how the **complex numbers** are more or less the 2D rotation group in disguise (if we identify $1 = R_0$ and $i = R_{\frac{\pi}{2}}$). This shift in perspective helps one to see that complex numbers are not so much "imaginary" as much as the idea that treating rotations like an alternate kind of number system (recalling Section 4 on group theory) can be helpful for mathematics such as factoring polynomials.

Another topic would be generalizing the structure we discussed to 3D rotations. We would see that, in contrast to 2D rotations, 3D rotations do not *commute*, meaning that when 3D rotations are composed, the final position you reach depends on the order in which you compose them. One implication of this fact is that the matrices representing 3D rotations are no longer always 2×2 in size, but can grow to arbitrary sizes. Nevertheless, the tensor train networks we considered above are powerful enough to handle the representation theory of 3D rotations as well. Last but not least, the rotation groups are examples of **Lie groups** which are not only groups but also smooth manifolds, forging connections between the mathematics of groups and the field of geometry.

The study of 2D rotations has provided us a small window into some of the beautiful structure underlying mathematics and the natural world. Patterns occurring throughout nature at every level share a "fearful symmetry" that leaves us with a strong intuition that the natural world is "fearfully and wonderfully made."

To learn more about some of the mathematical topics touched on in this article, some helpful resources are:

- "Lecture Notes on Group Theory in Physics", by Daniel Arovas [Aro16]
- Article on topological phases of matter in Quanta Magazine [Wol18]
- Collection of review articles and learning resources about tensor networks on

tensornetwork.org [ten]

- Blog post on the tensor product operation in mathematics [Bra18]
- The excellent book "An Introduction to Tensors and Group Theory for Physicists" by Jeevanjee [Jee11]

Appendix A: Brief Review of Matrix Multiplication

Matrices are usually motivated for writing systems of equations in a more abstract form. For example, the equations

$$\begin{aligned} -3x + 2y &= 7 \\ x + y &= 4 \end{aligned}$$

can equivalently be written in matrix form as

$$\begin{bmatrix} -3 & 2 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \end{bmatrix}$$

which has the advantage of letting us study the matrix of equation coefficients separately from the variables they multiply and the result vector $\left[\begin{smallmatrix} 7 \\ 4 \end{smallmatrix}\right]$ on the right, which turns out to be a very powerful concept. It is also the motivation for the **rules of matrix multiplication**, which we will briefly review now.

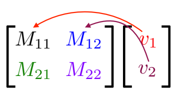
Say we want to perform the following matrix times vector multiplication.

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

The result will be the following vector.

$$\begin{bmatrix} (M_{11}v_1 + M_{12}v_2) \\ (M_{21}v_1 + M_{22}v_2) \end{bmatrix}$$

We can visualize the multiplication process as follows. First we "distribute" and multiply the elements of the vector across the first row of the matrix



and add the products together. This gives the first entry of the result vector.

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} M_{11}v_1 + M_{12}v_2 \\ \end{bmatrix}$$

We repeat the process for the second row of the matrix

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} M_{11}v_1 + M_{12}v_2 \\ \end{bmatrix}$$

giving the next, and in this case final, entry of the result vector.

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} M_{11}v_1 + M_{12}v_2 \\ M_{21}v_1 + M_{22}v_2 \end{bmatrix}$$

An important corollary of the matrix multiplication rule is that multiplying by the vector $\hat{x} = \left[\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right]$ gives the first column of the matrix and multiplying by $\hat{y} = \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$ gives the second column. Concretely,

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} M_{11} \\ M_{21} \end{bmatrix}$$

and

$$\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} M_{12} \\ M_{22} \end{bmatrix}$$

which was an important trick we used in Section 3 to deduce the entries of the rotation matrix R_θ.

Appendix B: Rotation of Functions

First, it's helpful to review how functions transform under coordinate transformations. The simplest case is a function of one variable $f(x)$ and a transformation which is a translation to the right by a. The translated function $T_a f(x)$ is related to the original one as

$$T_a f(x) = f(x - a)\ . \tag{8}$$

Why is the new argument $x - a$ and not $x + a$ if the translation is to the right (positive direction)? We can understand the minus sign by reasoning that the value of the function $T_a f(a)$ should be the same as $f(0)$ since the point at 0 shifted to the right by a equals a. Then $T_a f(a) = f(a-a) = f(0)$ by the definition in Equation (8), agreeing with our reasoning.

Similarly, if we translate a function of an angle $f(\phi)$ by an amount θ, the transformed function will be $\mathcal{R}_\theta f(\phi) = f(\phi - \theta)$. We can use this formula to work out how a two-dimensional function $f(x,y)$ transforms under rotations.

Given a function $f(x,y)$ defined over the two-dimensional plane, what is the result of rotating f by an angle θ? In other words, we want to know how to write $\mathcal{R}_\theta f(x,y)$ in terms of $f(x,y)$. To work this out, change to polar coordinates:

$$x = r\cos\phi \tag{9}$$

$$y = r\sin\phi \tag{10}$$

defining $\tilde{f}(r,\phi) = f(r\cos\phi, r\sin\phi)$ where a tilde over f indicates it takes polar arguments. Then the result of rotating $\tilde{f}$ by θ is by our earlier reasoning

$$\mathcal{R}_\theta \tilde{f}(r,\phi) = \tilde{f}(r,\phi-\theta) = f(r\cos(\phi-\theta), r\sin(\phi-\theta)) \tag{11}$$

where in the last equation the two slots refer to values plugged into the function where x and y normally go. Finally, using the trigonometry identities

$$\cos(\phi-\theta) = \cos\phi\cos\theta + \sin\phi\sin\theta$$
$$\sin(\phi-\theta) = \sin\phi\cos\theta - \cos\phi\sin\theta$$

we find that using these identities in Equation (11) above gives

$$\begin{aligned}\mathcal{R}_\theta f(x,y) =& \mathcal{R}_\theta \tilde{f}(r,\phi)\\ &= f(r\cos\phi\cos\theta + r\sin\phi\sin\theta, r\sin\phi\cos\theta - r\cos\phi\sin\theta)\\ &= f(\cos\theta\, x + \sin\theta\, y, -\sin\theta\, x + \cos\theta\, y)\end{aligned}$$

where we used Equation (11) between the second and third lines and the formulas (9) and (10) to change variables from r and ϕ back to x and y.

The result above shows we can rotate a function $f(x,y)$ by an angle θ by making the following substitutions:

$$\begin{aligned} x &\to \ \cos\theta\, x + \sin\theta\, y \\ y &\to \ -\sin\theta\, x + \cos\theta\, y \end{aligned}$$

everywhere that x and y appear in the definition of that function. Equations (6) and (7) are special cases of these transformations.

Appendix C: Rotation Matrix for the $n = 2$ Representation

To work out the general entries of the matrix $R_\theta^{[2]}$, or the rate 2 representation of the rotation group, consider the action of rotations of the functions $x^2 - y^2$ and $2xy$ under a *general angle* of rotation. First the function $x^2 - y^2$ transforms as

$$\begin{aligned} \mathcal{R}_\theta(x^2 - y^2) &= (\mathcal{R}_\theta x)(\mathcal{R}_\theta x) - (\mathcal{R}_\theta y)(\mathcal{R}_\theta y) \\ &= (\cos\theta\ x + \sin\theta\ y)(\cos\theta\ x + \sin\theta\ y) - (-\sin\theta\ x + \cos\theta\ y)(-\sin\theta\ x + \cos\theta\ y) \\ &= (\cos^2\theta - \sin^2\theta)\ (x^2 - y^2) + 4(\cos\theta\sin\theta)\ xy \\ &= \cos(2\theta)\ (x^2 - y^2) + \sin(2\theta)\ (2xy) \end{aligned} \tag{12}$$

where in the last line we have used the trigonometry identities

$$\cos^2\theta - \sin^2\theta = \cos(2\theta)$$
$$2\cos\theta\sin\theta = \sin(2\theta)$$

Next the function $2xy$ transforms as

$$\begin{aligned} \mathcal{R}_\theta(2xy) &= 2(\mathcal{R}_\theta x)(\mathcal{R}_\theta y) \\ &= 2(\cos\theta\ x + \sin\theta\ y)(-\sin\theta\ x + \cos\theta\ y) \\ &= -(2\cos\theta\sin\theta)(x^2 - y^2) + (\cos^2\theta - \sin^2\theta)(2xy) \\ &= -\sin(2\theta)(x^2 - y^2) + \cos(2\theta)(2xy). \end{aligned} \tag{13}$$

Putting lines (12) and (13) together from above and recalling that each of these expressions was the result of acting the rotation operator $\mathcal{R}_\theta$ on each of the functions, we can write the equations together in matrix form as

$$\mathcal{R}_\theta \begin{bmatrix} x^2 - y^2 \\ 2xy \end{bmatrix} = \begin{bmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \sin(2\theta) \end{bmatrix} \begin{bmatrix} x^2 - y^2 \\ 2xy \end{bmatrix}$$

which tells us that for these functions the rotation operator acts as the matrix

$$R_\theta^{[2]} = \begin{bmatrix} \cos(2\theta) & -\sin(2\theta) \\ \sin(2\theta) & \sin(2\theta) \end{bmatrix}.$$

Bibliography

[AAB+19] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019. https://doi.org/10.1038/s41586-019-1666-5.

[Aro16] Daniel Arovas. Lecture notes on group theory in physics, 2016. Available online: https://courses.physics.ucsd.edu/2016/Spring/physics220/LECTURES/GROUP_THEORY.pdf. Accessed 22 February 2023.

[BAO+20] Alexander Bogatskiy, Brandon Anderson, Jan Offermann, Marwah Roussi, David Miller, and Risi Kondor. Lorentz group equivariant neural network for particle physics. In Hal Daumé III and Aarti Singh, editors, *International Conference on Machine Learning*, volume 119, pages 992–1002. PMLR, 2020.

[BC17] Jacob C. Bridgeman and Christopher T. Chubb. Hand-waving and interpretive dance: an introductory course on tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 50(22):223001, 2017. https://doi.org/10.1088/1751-8121/aa6dc3.

[Bra18] Tai-Danae Bradley. The tensor product, demystified. Math3ma, 2018. Available online: https://www.math3ma.com/blog/the-tensor-product-demystified. Accessed 22 February 2023.

[Bra19] Tai-Danae Bradley. Matrices as tensor network diagrams. Math3ma, 2019. Available online: https://www.math3ma.com/blog/matrices-as-tensor-network-diagrams. Accessed on 22 February 2023.

[DWL+21] Nima Dehmamy, Robin Walters, Yanchen Liu, Dashun Wang, and Rose Yu. Automatic symmetry discovery with lie algebra convolutional network. *Advances in Neural Information Processing Systems*, 34:2503–2515, 2021.

[FWS22] Matthew Fishman, Steven R. White, and E. Miles Stoudenmire. The ITensor Software Library for Tensor Network Calculations. *SciPost Phys. Codebases*, page 4, 2022. https://doi.org/10.21468/SciPostPhysCodeb.4.

[GLD+22] Nikita Gourianov, Michael Lubasch, Sergey Dolgov, Quincy Y van den Berg, Hessam Babaee, Peyman Givi, Martin Kiffner, and Dieter Jaksch. A quantum-inspired approach to exploit turbulence structures. *Nature Computational Science*, 2(1):30–37, 2022. https://doi.org/10.1038/s43588-021-00181-1.

[Jee11] Nadir Jeevanjee. *An introduction to tensors and group theory for physicists*. Birkhäuser Science, 1st edition, 2011. https://doi.org/10.1007/978-3-319-14794-9.

[KLT18] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch–Gordan nets: a fully Fourier space spherical convolutional neural network. *Advances in Neural Information Processing Systems*, 31, 2018.

[KP22] Michael Kastoryano and Nicola Pancotti. A highly efficient tensor network algorithm for multi-asset fourier options pricing. *arXiv preprint arXiv:2203.02804*, 2022.

[KYG+07] Jens Koch, Terri M. Yu, Jay Gambetta, A. A. Houck, D. I. Schuster, J. Majer, Alexandre Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf. Charge-insensitive qubit design derived from the Cooper pair box. *Phys. Rev. A*, 76:042319, 2007. https://doi.org/10.1103/PhysRevA.76.042319.

[LW05] Michael A. Levin and Xiao-Gang Wen. String-net condensation: A physical mechanism for topological phases. *Phys. Rev. B*, 71:045110, Jan 2005. https://doi.org/10.1103/PhysRevB.71.045110.

[ME22] James C McCord and Glen Evenbly. Improved wavelets for image compression from unitary circuits. *arXiv preprint arXiv:2203.02556*, 2022.

[Orú14] Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of physics*, 349:117–158, 2014. https://doi.org/10.1016/j.aop.2014.06.013.

[Pen71] Roger Penrose. Applications of negative dimensional tensors. *Combinatorial mathematics and its applications*, 1:221–244, 1971.

[Per04] Alejandro Perez. Introduction to loop quantum gravity and spin foams. *arXiv preprint gr-qc/0409061*, 2004.

[RO22] Gleb Ryzhakov and Ivan Oseledets. Constructive TT-representation of the tensors given as index interaction functions with applications. *arXiv preprint arXiv:2206.03832*, 2022.

[RWH13] Kristen Rohlfs, Thomas L. Wilson, and Susanne Hüttemeister. *Tools of radio astronomy*. Springer Berlin, Heidelberg, 6th edition, 2013. https://doi.org/10.1007/978-3-642-39950-3.

[SV12] Sukhwinder Singh and Guifre Vidal. Tensor network states and algorithms in the presence of a global SU(2) symmetry. *Phys. Rev. B*, 86:195114, Nov 2012. https://doi.org/10.1103/PhysRevB.86.195114.

[SVS$^+$22] Rishi Sreedhar, Pontus Vikstål, Marika Svensson, Andreas Ask, Göran Johansson, and Laura García-Álvarez. The quantum approximate optimization algorithm performance with low entanglement and high circuit depth. *arXiv preprint arXiv:2207.03404*, 2022.

[ten] The Tensor Network. Available online: https://tensornetwork.org. Accessed 22 February 2023.

[Wig90] Eugene P. Wigner. The unreasonable effectiveness of mathematics in the natural sciences. In *Mathematics and Science*, pages 291–306. World Scientific, 1990. https://doi.org/10.1002/cpa.3160130102.

[Wol18] Natalie Wolchover. Physicists aim to classify all possible phases of matter. Quanta Magazine, 2018. Available online:

https://www.quantamagazine.org/
physicists-aim-to-classify-all-possible-phases-of-matter-20180103.
Accessed 22 February 2023.