

We have seen that in \mathbb{R}^2 the matrix corresponding to a counterclockwise (ccw) rotation thru an angle θ is given by $\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$. We seek the matrix corresponding to a counterclockwise rotation thru an angle θ about an given axis. We will regard the given axis as a new z -axis. We thus need new x and y axes, and we want them to form a right-handed coordinate system.

Our new coordinates will be found relative to our new unit vectors, which we'll call \mathbf{i}_1 , \mathbf{j}_1 and \mathbf{k}_1 . These vectors will be mutually orthogonal, and will each have length one. Their mutual orthogonality assures us that they will form a linearly independent set (Why?). Since there will be three of them, all in \mathbb{R}^3 , they will form a basis. Given a vector \mathbf{v} , we will easily find the coordinates of \mathbf{v} by taking dot products with these new basis vectors, to get the "Fourier series" representation $\mathbf{v} = (\mathbf{v} \cdot \mathbf{i}_1)\mathbf{i}_1 + (\mathbf{v} \cdot \mathbf{j}_1)\mathbf{j}_1 + (\mathbf{v} \cdot \mathbf{k}_1)\mathbf{k}_1$.

We have seen that the dot product can be written as a matrix product. We can thus write $\mathbf{v} \cdot \mathbf{i}_1 = \mathbf{i}_1^T \mathbf{v}$, and so on, and so get $\mathbf{v} = (\mathbf{i}_1^T \mathbf{v})\mathbf{i}_1 + (\mathbf{j}_1^T \mathbf{v})\mathbf{j}_1 + (\mathbf{k}_1^T \mathbf{v})\mathbf{k}_1$. I have put in the parentheses because without them we'd have, for example, in $\mathbf{i}_1^T \mathbf{v} \mathbf{i}_1$ a product of 1×3 , 3×1 and 3×3 matrices, which don't match. We allow ourselves to do this (as humans) because we think of 1×1 matrices as scalars. But what if we wrote the Fourier representation so a computer would not be bothered – by putting the scalar *after* the vector:

$$\mathbf{v} = \mathbf{i}_1 \mathbf{i}_1^T \mathbf{v} + \mathbf{j}_1 \mathbf{j}_1^T \mathbf{v} + \mathbf{k}_1 \mathbf{k}_1^T \mathbf{v} = (\mathbf{i}_1 \mathbf{i}_1^T + \mathbf{j}_1 \mathbf{j}_1^T + \mathbf{k}_1 \mathbf{k}_1^T) \mathbf{v}$$

each summand is a product: $(3 \times 1) \times (1 \times 3) \times (3 \times 1) = 3 \times 1$, and the dimensions "line up." Just for fun, compute the matrix $\mathbf{i}_1 \mathbf{i}_1^T + \mathbf{j}_1 \mathbf{j}_1^T + \mathbf{k}_1 \mathbf{k}_1^T$ when $\mathbf{i}_1 = (\frac{6}{7}, \frac{2}{7}, \frac{-3}{7})$, $\mathbf{j}_1 = (\frac{2}{7}, \frac{3}{7}, \frac{6}{7})$ and $\mathbf{k}_1 = (\frac{3}{7}, \frac{-6}{7}, \frac{2}{7})$. What did you expect?

If we had started with $\mathbf{i}_1 = \mathbf{i}$, $\mathbf{j}_1 = \mathbf{j}$ and $\mathbf{k}_1 = \mathbf{k}$, the rotation would have been easy: $\mathbf{v} = a\mathbf{i}_1 + b\mathbf{j}_1 + c\mathbf{k}_1$ would be sent to $M_\theta \mathbf{v} = (a \cos \theta - b \sin \theta)\mathbf{i}_1 + (a \sin \theta + b \cos \theta)\mathbf{j}_1 + c\mathbf{k}_1$, because we'd just be adding in the z -axis, and ccw means "ccw as seen from above." Let's make things messy for the moment by putting in what a , b and c are:

$$\mathbf{v} = a\mathbf{i}_1 + b\mathbf{j}_1 + c\mathbf{k}_1 = (\mathbf{v} \cdot \mathbf{i}_1)\mathbf{i}_1 + (\mathbf{v} \cdot \mathbf{j}_1)\mathbf{j}_1 + (\mathbf{v} \cdot \mathbf{k}_1)\mathbf{k}_1,$$

the Fourier representation. Therefore, using our "put-them-in-order" approach,

$$\begin{aligned} M_\theta \mathbf{v} &= (a \cos \theta - b \sin \theta)\mathbf{i}_1 + (a \sin \theta + b \cos \theta)\mathbf{j}_1 + c\mathbf{k}_1 \\ &= ((\mathbf{v} \cdot \mathbf{i}_1) \cos \theta - (\mathbf{v} \cdot \mathbf{j}_1) \sin \theta)\mathbf{i}_1 + ((\mathbf{v} \cdot \mathbf{i}_1) \sin \theta + (\mathbf{v} \cdot \mathbf{j}_1) \cos \theta)\mathbf{j}_1 + (\mathbf{v} \cdot \mathbf{k}_1)\mathbf{k}_1 \\ &= \cos \theta \mathbf{i}_1 \mathbf{i}_1^T \mathbf{v} - \sin \theta \mathbf{i}_1 \mathbf{j}_1^T \mathbf{v} + \sin \theta \mathbf{j}_1 \mathbf{i}_1^T \mathbf{v} + \cos \theta \mathbf{j}_1 \mathbf{j}_1^T \mathbf{v} + \mathbf{k}_1 \mathbf{k}_1^T \mathbf{v} \\ &= \cos \theta (\mathbf{i}_1 \mathbf{i}_1^T + \mathbf{j}_1 \mathbf{j}_1^T) \mathbf{v} - \sin \theta (\mathbf{i}_1 \mathbf{j}_1^T - \mathbf{j}_1 \mathbf{i}_1^T) \mathbf{v} + \mathbf{k}_1 \mathbf{k}_1^T \mathbf{v} \\ &= \left[\cos \theta (\mathbf{i}_1 \mathbf{i}_1^T + \mathbf{j}_1 \mathbf{j}_1^T) - \sin \theta (\mathbf{i}_1 \mathbf{j}_1^T - \mathbf{j}_1 \mathbf{i}_1^T) + \mathbf{k}_1 \mathbf{k}_1^T \right] \mathbf{v}. \end{aligned}$$

This gives us the matrix: $M_\theta = \cos \theta (\mathbf{i}_1 \mathbf{i}_1^T + \mathbf{j}_1 \mathbf{j}_1^T) - \sin \theta (\mathbf{i}_1 \mathbf{j}_1^T - \mathbf{j}_1 \mathbf{i}_1^T) + \mathbf{k}_1 \mathbf{k}_1^T$, where we use the "usual" coordinates for vectors in \mathbb{R}^3 .

Our problem now is, given an axis $\vec{p} = (a, b, c)$, which need not be a unit vector, how do we, treating \vec{p} as having the *direction* we want our z -axis to point in, find vectors \mathbf{i}_1 , \mathbf{j}_1 and \mathbf{k}_1 ? Here is *one* way to proceed: we pick one of the vectors we know how to make, orthogonal to \vec{p} ,

$$\begin{pmatrix} -b \\ a \\ 0 \end{pmatrix}, \begin{pmatrix} -c \\ 0 \\ a \end{pmatrix}, \begin{pmatrix} 0 \\ -c \\ b \end{pmatrix} \text{ as our } \mathbf{i}_1\text{-direction, and call it } \mathbf{x}_1 \text{ for now,}$$

making sure it's non-zero. Our \mathbf{x}_1 will have one of the letters a , b , c missing. Call that letter's number d_1 for now, and let \mathbf{u}_1 be the one of \mathbf{i} , \mathbf{j} and \mathbf{k} that corresponds to the missing letter. Thus, if b is the missing letter, \mathbf{u}_1 will be \mathbf{j} . We then construct the vector

$$\mathbf{y}_1 := -d_1 \vec{p} + |\vec{p}|^2 \mathbf{u}_1.$$

Finally, we let

$$\mathbf{i}_1 := \mathbf{x}_1 / |\mathbf{x}_1|, \mathbf{j}_1 := \mathbf{y}_1 / |\mathbf{y}_1|, \mathbf{k}_1 := \vec{p} / |\vec{p}|.$$

As an easy example, suppose $\vec{p} = (0, 0, -1)$. We pick, say $\mathbf{x}_1 = (-c, 0, a) = (-1, 0, 0)$, so b is missing, $d_1 = b = 0$, and $\mathbf{u}_1 = \mathbf{j}$. This gives

$$\mathbf{i}_1 := \mathbf{x}_1 / |\mathbf{x}_1| = -\mathbf{i}, \mathbf{j}_1 := \mathbf{y}_1 / |\mathbf{y}_1| = \mathbf{j}, \mathbf{k}_1 := \vec{p} / |\vec{p}| = -\mathbf{k}.$$

This "recipe" gives a right-handed coordinate system, but *why* it always is needs to wait a little while.