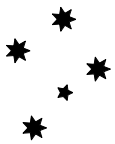

PRIME OPTICS

Coordinate System Alignment

for Optical Surface Modelling

2021 09 29, 2024 02 13

Telephone :	+61-7-5442 8813	*Local Time :*	UT + 10
SMS :	+61-429-02 8831	*EMail :*	damien@primeoptics.com.au
Contact :	**Damien Jones**	*Web :*	www.primeoptics.com.au

1 Introduction

The impetus for this note is the need to understand coordinate transformations and how these relate to the OpticStudio® "Coordinate Break" model.

2 Coordinate Transformation via Intrinsic Rotations of Coordinate Axes

So called intrinsic rotations follow the moving or tilting coordinate system.

Initially the global and local coordinate systems are one and the same.

The local coordinate system is translated first to a pre-determined origin (this can be a decentration, for example); and then the axes are rotated in succession.

The first rotation is around a convenient or appropriate local axis; followed by two more around the newly tilted local axes to reach a target local coordinate system.

New point coordinates are relative to this local system.

In this example, the local y- and z-axes are rotated around the local x-axis by Θ_x ; then the new z-axis and original x-axis are rotated around the new y-axis by Θ_y; and lastly, the new x- and y-axes are rotated around the new z-axis by Θ_z .

This is the most common transformation sequence required in optical systems modelling because rotationally symmetric component tilts are by convention around their local x- and y-axes. Less common components, such as gratings and cylindrical components, often require a z-rotation, the z-axis being the optical axis by convention.

The transformation in matrix form is:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = M_z(\theta_z) \cdot M_y(\theta_y) \cdot M_x(\theta_x) \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix}$$

where:

$$M_x(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & \sin\theta_x \\ 0 & -\sin\theta_x & \cos\theta_x \end{bmatrix}$$

$$M_y(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y \\ 0 & 1 & 0 \\ \sin\theta_y & 0 & \cos\theta_y \end{bmatrix}$$

$$M_z(\theta_z) = \begin{bmatrix} \cos\theta_z & \sin\theta_z & 0 \\ -\sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and:

$$(x_0, y_0, z_0) = new\ local\ origin$$

The operation is not commutative, so must be used with care.

The operation can be reversed or “unwound” back to the global system:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = M_x(-\theta_x) \cdot M_y\left(-\theta_y\right) \cdot M_z(-\theta_z) \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

The initial transformation and its reverse can be described as X-Y’-Z” and Z–Y’–X” rotations respectively.

3 Some Commentary on the OpticStudio® Notation

The reverse coordinate transformation, with one exception, literally follows the ZEMAX/OpticStudio® Coordinate Break algorithm described in the older manuals (and the current Knowledge Base, see below) with the order flag set to zero. Thus, in the Lens Data Editor, x- and y-offsets are applied first, then a tilt is nominated around the local x-axis, then around the new local y-axis, then around the new local z-axis, thereby arriving at the nominated local coordinate system.

The exception here is that the z-offset is also applied along with the x- and y-offsets rather than being applied as a “thickness” after the transformation. It is therefore a complete translation of the local coordinate system origin before rotations of the local coordinate system’s axes are applied.

If the Coordinate Break order flag is set to 1 then the z-rotation is done first, offsets last.

The derivation so far is for transforming the coordinates of fixed points in a global coordinate system into a rotated and displaced local coordinate system; and then “unwinding” it back to global.

By contrast, ZEMAX/OpticStudio® uses a series of local coordinate systems for each component. Consequently, a Coordinate Break tilt will apply to the local coordinate system along with the function that defines the surface.

The so-called “Rotation Matrix” is the transformation required to convert the local surface and function coordinates back into the global system and is exactly the same as the “unwinding” operation described above.

So using the ZEMAX notation, the “rotation matrices” are just:

$$R_x(\theta_x) = M_x(-\theta_x)$$
$$R_y\left(\theta_y\right) = M_y\left(-\theta_y\right)$$
$$R_z(\theta_z) = M_z(-\theta_z)$$

And so:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_x(\theta_x) \cdot R_y\left(\theta_y\right) \cdot R_z(\theta_z) \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

which on face value is consistent with the descriptions in the ZEMAX® manual and in the Knowledge Base.

Somewhat conveniently, the respective columns of the resultant “rotation matrix” (the resultant matrix product) now become the components of the 3 local coordinate axes, which is very handy for display in the “System Prescription” in the “Global Vertex” section.

We can see this if we ignore the translation part of the transformation and rewrite the matrix product after invoking vector notation (bold characters represent vector quantities) and using some shorthand:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = [\boldsymbol{i}' \quad \boldsymbol{j}' \quad \boldsymbol{k}'] \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

$$\boldsymbol{i}', \boldsymbol{j}' and\ \boldsymbol{k}' are\ the\ local\ axes\ relative\ to\ global\ space.$$
$$Expanding\ the\ columns:$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} i_1 & j_1 & k_1 \\ i_2 & j_2 & k_2 \\ i_3 & j_3 & k_3 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

where the **rows** can be seen to be the components of the 3 local coordinate axes in the global x, y, and z directions respectively. And of course, the **columns** are the components of each respective axis in global space, or in other words, the normal 3D Cartesian basis vectors of the local space.

The form of the individual rotation matrices is consistent with a series of so-called “extrinsic rotations” around the fixed global system axes.

However, it should be clear by now that the distinction between intrinsic and extrinsic rotations depends very much on one’s point of view.

In this context the transformations are simply “global-to-local” and vice versa.

4 Some Commentary on the OpticStudio® Knowledge Base Article

There is a Knowledge Base article (titled “Rotation Matrix and Tilt About X/Y/Z in OpticStudio”) on the OpticStudio® website which contains some ambiguity. On the plus side, the analysis of the given matrices is sound and obviously agrees with what the program calculates.

However, under the heading “Intrinsic and Extrinsic Rotations”, it is stated that the reverse Z-Y’-X” rotation described above is an “extrinsic rotation”. In light of the above discussion, this is clearly ambiguous. On the spreadsheet, it looks like another series of intrinsic rotations with the angles of opposite sign, and in reverse order. However, as demonstrated above, the matrices are also consistent with a series of extrinsic rotations with angles of the same sign. Go figure!

The two dot points which follow are also ambiguous, by the same reasoning.

In closing, it is often much easier to visualise a series of intrinsic rotations in a convenient order unless the extrinsic rotation angles are clear multiples of 90 degrees. This can often be the case with non-sequential objects.

5 Extracting Rotation Angles

The rotation matrix product can be used to extract rotation angles, as per the knowledge base article.

Thus, for order flag "0":

$$R_x(\theta_x)\cdot R_y(\theta_y)\cdot R_z(\theta_z) = \begin{bmatrix} \cos\theta_y\cos\theta_z & -\cos\theta_y\sin\theta_z & \sin\theta_y \\ \cos\theta_x\sin\theta_z+\sin\theta_x\sin\theta_y\cos\theta_z & \cos\theta_x\cos\theta_z-\sin\theta_x\sin\theta_y\sin\theta_z & -\sin\theta_x\cos\theta_y \\ \sin\theta_x\sin\theta_z-\cos\theta_x\sin\theta_y\cos\theta_z & \sin\theta_x\cos\theta_z+\cos\theta_x\sin\theta_y\sin\theta_z & \cos\theta_x\cos\theta_y \end{bmatrix}$$

We can declare a function:

$$ArcTan2(Y,X) = \tan^{-1}\left(\frac{Y}{X}\right) = \theta$$
$$with\ quadrant\ information, such\ that\ -\pi < \theta \le \pi$$

Thus:

$$\theta_x = ArcTan2(-R23, R33)$$
$$\theta_y = \sin^{-1}(R13), \qquad -\frac{\pi}{2} \le \theta_y \le \frac{\pi}{2}$$
$$\theta_z = ArcTan2(-R12, R11)$$

following the OpticStudio® notation.

The rotation matrix product can also be used to extract the [different] rotation angles that would be required if the order flag is non-zero. Of course, the order of the matrices is also reversed.

Thus:

$$R_z(\theta_z)\cdot R_y(\theta_y)\cdot R_x(\theta_x) = \begin{bmatrix} \cos\theta_y\cos\theta_z & -\cos\theta_x\sin\theta_z+\sin\theta_x\sin\theta_y\cos\theta_z & \sin\theta_x\sin\theta_z+\cos\theta_x\sin\theta_y\cos\theta_z \\ \cos\theta_y\sin\theta_z & \cos\theta_x\cos\theta_z+\sin\theta_x\sin\theta_y\sin\theta_z & -\sin\theta_x\cos\theta_z+\cos\theta_x\sin\theta_y\sin\theta_z \\ -\sin\theta_y & \sin\theta_x\cos\theta_y & \cos\theta_x\cos\theta_y \end{bmatrix}$$

and

$$\theta_x = ArcTan2(R32, R33)$$
$$\theta_y = \sin^{-1}(-R31)$$
$$\theta_z = ArcTan2(R21, R11)$$

The symmetry between the 2 matrix products is clear – indeed, they are the transpose of each other, with angles of opposite sign. Of course, the extracted angles will not be the same.

6 Disclaimer

Unpacking all this has been very helpful for me. Any errors spotted are all mine. The reader should verify that the above methods work for his or her particular situation.

Please email any suggestions and errors directly to me.

Thanks for reading and I hope it has been helpful.