# Introduction

This worksheet is split into two sections; the first deals with matrix transformations between coordinate spaces, while the second contains some exercises involving quaternion calculations.

You may find the Symbolab [matrix multiplication calculator](https://www.symbolab.com/solver/matrix-multiply-calculator) useful for finding/checking your results.

# Matrix Transformations

1. An object initially had its axes and origin coincident with the world axes and origin. It was first rotated 30° about the -axis, and then -22° about the world -axis.
   1. What is the matrix that can be used to transform column vectors from object space to world space?  
        
      The “transformation from object space to world space” is the same as “the transform of the object in world space” – so, we first need a matrix to represent a rotation of 30° about the y-axis:  
        
      We also need a rotation of -22° about the x-axis:  
      Next, we need to combine them from right to left in the order they should be applied (since the column vector will be on the far right):
   2. What about the matrix to transform vectors from world space to object space?  
        
      This is the **inverse** of the matrix transform in part (a),  
      To compute this, we can use the fact that the inverse of a rotation matrix is its **transpose**, or we can compute the matrices to rotate through the negatives of the original angle:  
         
      In fact, we could have saved some calculations by noting that the result of combining rotations is still a rotation, so the inverse of the combined transform is just its transpose. This is because a rotation matrix is [orthogonal](https://mathworld.wolfram.com/OrthogonalMatrix.html), i.e. its columns form an orthonormal basis.
   3. Express the object’s -axis using world coordinates.  
      The object’s z-axis in local space is simply , so to express this in world coordinates we can apply the object-to-world transform from part (a):  
      Notice that this the same as the last column of the matrix – because the matrix represents the basis vectors of the transformed space.
2. A robot is at the position (1, 10, 3) and her right, up and forward vectors (expressed in world space) are , and respectively (note that these vectors form an orthonormal basis).  
     
   This is essentially saying that the “right, up and forward” vectors are the axes of the robot’s local space, so that a point in the robot’s space is a linear combination of distances along the local axes, measured from the robot’s position in world space, which defines the origin of the local space.  
     
   This means that the world-space vector from the robot’s position to the point , which is the same point expressed in world coordinates, is:  
   Rearranging this, we see that we can obtain the world-space coordinates from the object-space ones as:  
     
   We can extract equations to convert each local coordinate to a world-space one:  
   a rotation of 30° about the y-axis, followed by a translation to the robot’s origin.
   1. The following points are expressed in object space; calculate their coordinates in world space:
      1. (-1, 2, 0)  
         (0.866 x (-1) + 0.5 x 0 + 1, 2 + 10, -0.5 x (-1) + 0.866 x 0 + 3) = (0.134, 12, 3.5)
      2. (1, 2, 0)  
         (0.866 x 1 + 0.5 x 0 + 1, 2 + 10, -0.5 x 1 + 0.866 x 0 + 3) = (1.866, 12, 2.5)
      3. (0, 0, 0)  
         (0.866 x 0 + 0.5 x 0 + 1, 0 + 10, -0.5 x 0 + 0.866 x 0 + 3) = (1, 10, 3)  
         … The robot’s position is the origin of its local space!
      4. (1, 5, 0.5)  
         (0.866 x 1 + 0.5 x 0.5 + 1, 5 + 10, -0.5 x 1 + 0.866 x 0.5 + 3) = (2.116, 15, 2.933)
      5. (0, 5, 10)  
         (0.866 x 0 + 0.5 x 10 + 1, 5 + 10, -0.5 x 0 + 0.866 x 10 + 3) = (6, 15, 11.66)
   2. The coordinates below are in world space; find their positions relative to the robot:  
        
      We now need to perform the operation in reverse, converting into by finding the local coefficients from the world ones . One way we can do this is to rearrange the functions we were using to do the object-to-world conversion to isolate the world coordinate values, solving them as simultaneous equations:  
      Substituting this into the formula for gives in terms of world coordinates:  
      Now putting this back into the expression for :  
        
      Finally, the y coordinates aren’t affected by the others, so our final functions are:  
      ourse, we could also achieve the same result by finding the inverse matrix, either using numerical methods or by constructing the matrices for the reverse transformations – rotating by -30° about the y-axis and translating by (-1, 10, 3) – and multiplying them, remembering that the translation must be applied first in this case:  
      , but let’s use the formulae again this time:
      1. (1, 10, 3)  
         (0.866 x 1 - 0.5 x 3 + 0.634, 10 - 10, 0.5 x 1 + 0.866 x 3 - 3.098) = (0, 0, 0)  
         … As we would expect, since this is the robot’s position in world space.
      2. (0, 0, 0)  
         (0.866 x 0 - 0.5 x 0 + 0.634, 0 - 10, 0.5 x 0 + 0.866 x 0 - 3.098) = (0.634, -10, -3.098)  
         Rotating the origin has no effect, since rotations are about the origin, and the net result is the translation values.
      3. (2.732, 10, 2)  
         (0.866 x 2.732 - 0.5 x 2 + 0.634, 10 - 10, 0.5 x 2.732 + 0.866 x 2 - 3.098) = (2, 0, 0)
      4. (2, 11, 4)  
         (0.866 x 2 - 0.5 x 4 + 0.634, 11 - 10, 0.5 x 2 + 0.866 x 4 - 3.098) = (0.366, 1, 1.366)
      5. (1, 20, 3)  
         (0.866 x 1 - 0.5 x 3 + 0.634, 20 - 10, 0.5 x 1 + 0.866 x 3 - 3.098) = (0, 10, 0)

# Quaternions

1. A quaternion to rotate through an angle is written as .
   1. Construct a quaternion to rotate 30° about the -axis.
   2. What is the magnitude of this quaternion?  
      1 – because all rotation quaternions have a magnitude of 1! (You can do the maths to check if you like, bearing in mind the identity: ).
   3. What is its conjugate, \*?
   4. Assume the quaternion is used to rotate points from object space to world space. What would the position of the point be under this rotation?  
        
      To apply a quaternion rotation to a point, we first express the point in quaternion form, . We then multiply in a sandwich between and its inverse, noting that:  
      ∙ for rotation quaternions, and  
      This gives us:  
        
      Quaternion multiplication is associative, so it doesn’t matter which order we calculate this in. Breaking it down, we get:  
       with our result above and the inverse of , , in the same way:  
       Giving the transformed point as (0.51, -1.758, 1.643).  
        
      Notice that the coordinate ought to be unchanged; due to rounding errors, it has moved a little, but a higher degree of precision should avoid this! Also notice that the component of the final quaternion has cancelled out to zero, as we would hope for a point.
2. Compute a quaternion that performs twice the rotation of the quaternion .  
     
   First, we extract the half-angle and axis of rotation:  
   Now we form a new quaternion using the new half-angle, :  
   Note that also represents twice the angular displacement of , so we could use the exponentiation formula and the identities for and to give exactly the same result:
3. Consider the quaternions:  
   1. Compute the dot product , given by the formula
   2. Compute the quaternion product , given by the Hamilton product
   3. Compute the difference from to , given by the quaternion (with \*).