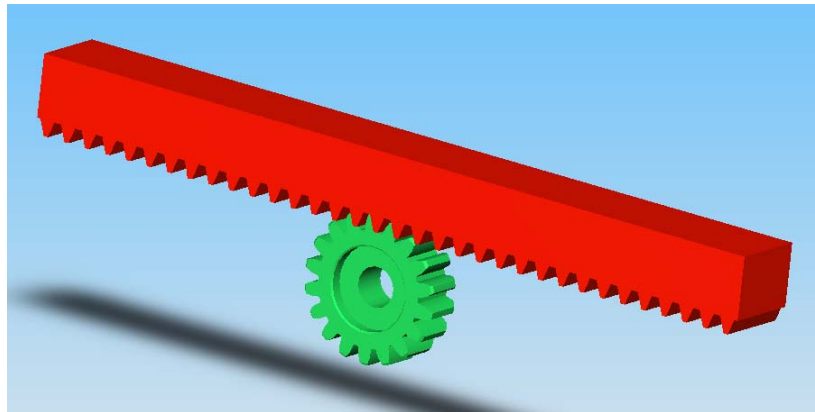


## Application Note #3416

### A Brief Overview of Coordinate Transformation

#### Introduction

The issue of coordinate transformation is unavoidable in the field of motion control. This is because often the force applied to a payload is oriented differently than the desired direction of movement. Coordinate transformation is also important because there are an infinite number of units in which to define position, and often the units interpreted by the control system are different than what the user understands or wishes to specify. For instance, in the rack and pinion system shown below in Figure 1, the applied force is a torque from a motor. This torque causes a rotation of the pinion, which is translated into a 1 dimensional, linear movement of the payload along the rack.



**Figure 1:** Example rack (red linear gear) and pinion (green circular gear) system where the rotational displacement of the pinion gets translated into a linear displacement of the rack.

Also, the distance moved by both parts of the system can be specified in many different units such as the ones shown in Table 1.

<b>Pinion (Green)</b>	<b>Rack (Red)</b>
Degrees ( $^{\circ}$ )	Length (in, m, ft, etc)
Radians ( $\Theta$ )	Linear Encoder Counts (cts)
Rotational Encoder Counts (cts)	Incremental Position
Incremental Position	Absolute Position
Absolute Position	

**Table 1:** List of possible reference units for rack and pinion location/displacements

It is an objective of a motion control system to move the payload with respect to a coordinate and unit system that is easily understandable by the user. To move the payload along the rack in Figure 2, a commanded position must be specified by the controller. This would be in the form of displacement of the motor, often encoder counts. This could be done by the user knowing the desired linear distance, mesh characteristics between the rack and pinion and angle of motor shaft rotation. However, this process would be very complex and difficult to understand but would follow the Equation 1.

$$Dist_r * \left( \frac{T_r}{Dist_r} \right) * \left( \frac{Rev_p}{T_p} \right) * \left( \frac{Cts_m}{Rev_m} \right) = Dis_c$$

Where:

$Dist_r$  = Linear Displacement of Rack (length)

$T_r$  = Number of Teeth on Rack (integer)

$T_p$  = Number of Teeth on pinion (integer)

$Dist_r$  = Distance on Rack (length)

$Rev_p$  = Revolution of Pinion (integer)

$Rev_m$  = Revolution of Motor (integer)

$Cts_m$  = Counts of Motor (integer)

$Dis_c$  = Rotational Displacement of Motor (cts)

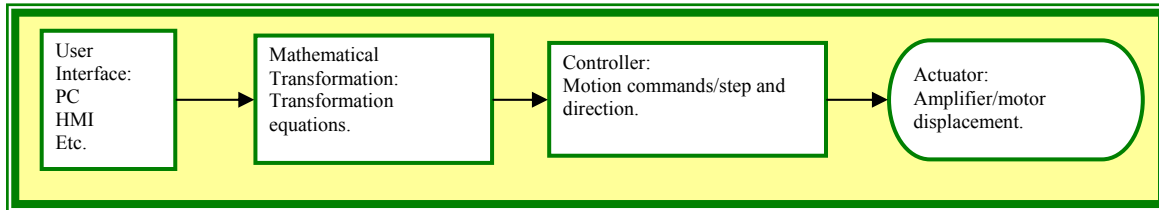
**Equation 1:** Mathematical conversion between linear distance and motor displacement in encoder counts for rack and pinion system shown in Figure 1

There are two options as to how to apply transformation equations to a physical system. One would be outside of the controller, meaning the user must perform the transformations manually then enter commands into the controller in units that it understands. This would be a lengthy and inconvenient method of applying coordinate transformation, especially if multiple axes are involved. Also, the confusion associated with the units and translations involved would make it difficult for the user to truly understand the system. The other option, offered by Galil Motion Control, is to program the controller to accept the desired displacement, in user readable units, and then perform the translation and movement(s) automatically from within the controller. This can be done by implementing custom firmware specific to the application and its specific coordinate transformation, making the user interface much more convenient and efficient.

## Discussion

There are two forms of motion control systems, open and closed loop. Because there is only a feedforward signal in an open loop system, only one transformation is required between the user input and motion commands. For example, with the rack and pinion system described earlier, if the motor that is providing the torque on the pinion is a stepper motor with no feedback, the linear distance can be translated into stepper counts

and executed. This would follow the following steps seen in Figure 3 that are represented in Equation 2.



**Figure 3:** Flow chart of open loop control system

$$Dist_r * \left( \frac{T_r}{Dist_r} \right) * \left( \frac{Rev_p}{T_p} \right) * \left( \frac{Steps_m}{Rev_m} \right) = Dis_s$$

Where:

$Dist_r$  = Linear Displacement of Rack (length)

$T_r$  = Number of Teeth on Rack (integer)

$T_p$  = Number of Teeth on pinion (integer)

$Dist_r$  = Distance on Rack (length)

$Rev_p$  = Revolution of Pinion (integer)

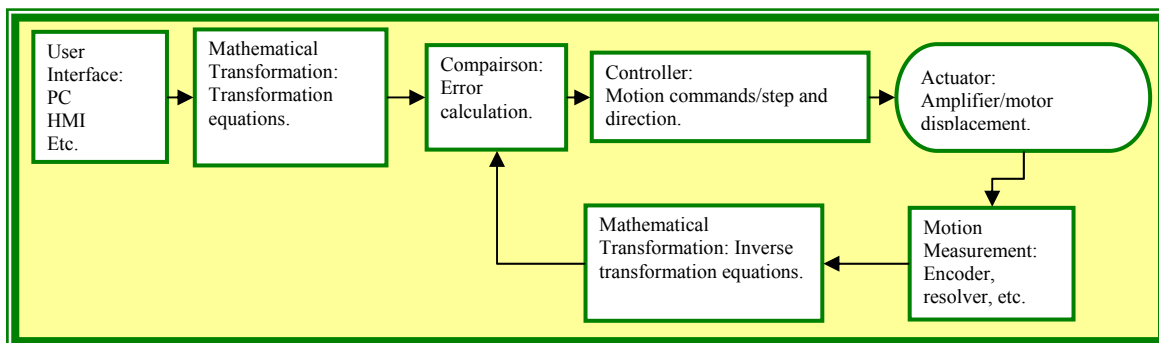
$Rev_m$  = Revolution of Motor (integer)

$Steps_m$  = Steps of Motor (integer)

$Dis_s$  = Rotational Displacement of Motor (steps)

**Equation 2:** Mathematical conversion between linear distance and motor displacement in steps for rack and pinion system shown in Figure 1 actuated by a stepper motor.

If the system were a closed loop servo system, the motion command would not only need to be transformed on the feedforward path but the payload position would also need to be transformed on the feedback path as well. This would be necessary to compare the actual displacement to commanded displacement and close the loop. For the simple rack and pinion system above these feedback coordinate transformation equations, or inverse transformation equations, can be seen in Equation 3



**Figure 4:** Flowchart of closed loop control system

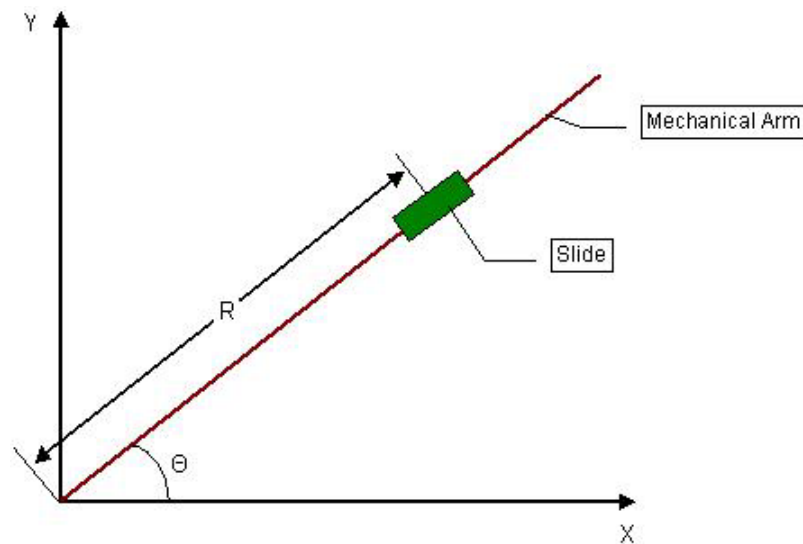
$$Dis_s * \left( \frac{Rev_m}{Cts_m} \right) * \left( \frac{T_p}{Rev_p} \right) * \left( \frac{Dist_r}{T_r} \right) = Dist_r$$

Where:

All variables are referenced above.

**Equation 3:** Inverse transformation equations of system described in Figure 4. Feedforward equations can be seen as Equation 1.

Examining a system such as the arm slide system seen in Figure 5 illustrates the fact that the transformation equations can get substantially more complex with only slightly more complex systems.



**Figure 5:** Arm slide mechanism.

In the arm slide system above the position of the slide can be specified in many different ways. As is with any system the number of equations needed to specify the location of a point or object is the same as its degrees of freedom or “ways to move.” The slide in Figure 4 has 2 degrees of freedom; the rotation of the arm around the origin and the distance of the slide along the arm. This indicates that there needs to be two equations of motion to specify the slide’s position. Seen in Figure 4, this can be done by specifying the angle of the arm off of the X axis ( $\Theta$ ) and the distance of the slide along the arm ( $R$ ).

It may not be convenient for the user to specify the desired location of the slide in  $\Theta$ ,  $R$  coordinates. Rather specifying the slide’s location in  $X$ ,  $Y$  coordinates may be more efficient. To do this a coordinate transformation must be performed. This can be done by applying the following equations seen in Equation(s) 4.

$$X = R * \cos(\Theta)$$

$$Y = R * \sin(\Theta)$$

**Equation(s) 4:** Feedforward (transformation) equations for system in figure 5 to transform between  $\Theta$  R and X Y coordinates.

Even after applying these equations to the system, further transformation between unit systems may still be necessary.

To close the servo loop within this system it would be necessary to compare the actual vs commanded position, therefore inverse transformation equations are necessary to transform the feedback back into  $\Theta$ , R coordinates. The inverse transformation equations may not be obvious based upon examination of the feedforward equations but can be seen in Equation(s) 5.

$$\Theta = \text{Tan}^{-1}\left(\frac{Y}{X}\right)$$

$$R = \sqrt{X^2 + Y^2}$$

**Equation(s) 5:** Feedback (inverse transformation) equations for system in Figure 5 to transform between X Y and  $\Theta$  R coordinates.

## Application

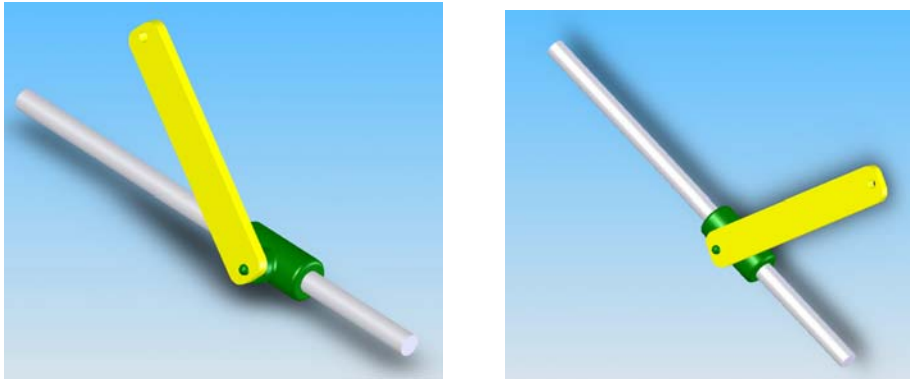
As mentioned above the application of transformation equations can be done within a Galil motion controller by implementing custom firmware. With this, the controller can automatically and seamlessly convert the user input into the necessary signals on both the feedforward and feedback path. The implementation of the transformation equations can be easily handled by the increased processing capabilities of the Galil DMC-40x0 Accelera series motion controller. The DMC-40x0 controller calculates squaring in approximately 10 $\mu$ s and approximately 15 $\mu$ s for inverse trigonometric ratios such as arcsine, arctangent and arccosine.

## Summary

It is necessary to understand the concept of coordinate transformation in the field of motion control because of the plethora of coordinate units, systems and planes that are available to define the system. The process of developing coordinate transformation equations with respect to these systems is not the same for any two applications. It often depends on convenience, need and interpretation by the user. To help provide a better understanding and demonstrate the points made above, various examples are provided in the appendix.

## Appendix

### Example 1: Cylinder mounted slide arm system.



Figure(s) 1-2: 3-D representation of the cylinder mounted arm slide system in 2 positions.

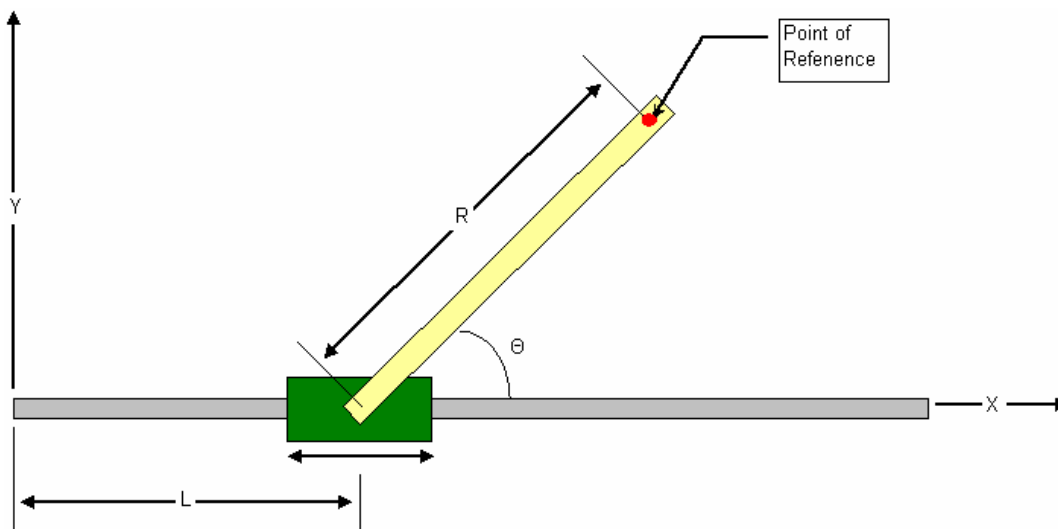


Figure 3: Simplified mathematical model of the system in Figure(s) 1-2. In this system  $L$  and  $\Theta$  are variable and  $R$  is fixed and known.

#### Forward Transformation Equations:

$$X = L + R * \cos(\Theta)$$

$$Y = R * \sin(\Theta)$$

#### Inverse Transformation Equations:

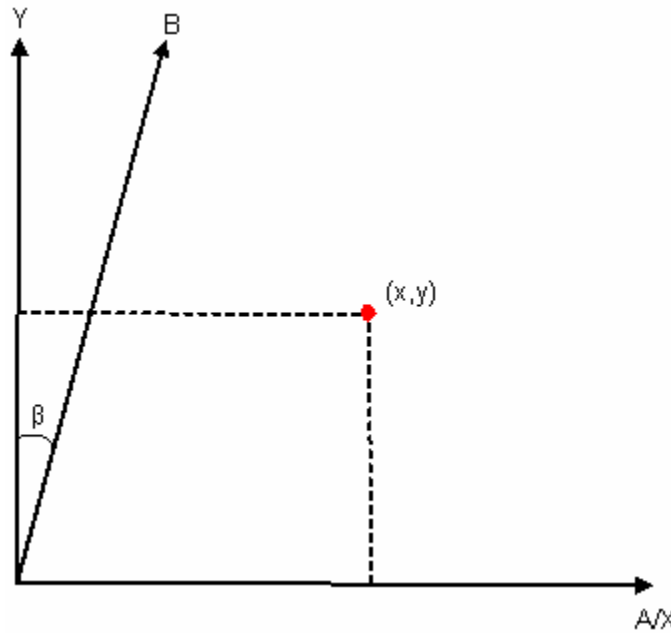
$$\Theta = \sin^{-1}\left(\frac{Y}{R}\right)$$

$$L = X - R * \cos(\Theta)$$

**Note:** The inverse equation that defines  $L$  requires the input  $\Theta$ , which is defined by the other inverse transformation equation. Even though it is mathematically incorrect to define one transformation equation with the other, it is an illustration as to how

convenience and user intuition is important with the development of these equations. In this case X and Y are related more seamlessly to  $\Theta$  and L using  $\Theta$  in the definition of L. Defining these equations explicitly with a strict mathematical relationship would be very cumbersome. This makes the equations easier to handle because in a physical system  $\Theta$  could be easily measured off of an encoder mounted to the motor that actuates the arm, making the measurement accurate and easy.

**Example 2:** Offset reference axis



**Figure 4:** AB axes with XY point and fixed, known angle  $\beta$  defining the offset from Y to B

**Forward Transformation Equations:**

$$X = A + B * \sin(\beta)$$

$$Y = B * \cos(\beta)$$

**Inverse Transformation Equations:**

$$B = \frac{Y}{\cos(\beta)}$$

$$A = X - B * \sin(\beta)$$

**Note:** In this case, due to the angle  $\beta$  being constant (for example purposes the value of  $\beta$  will be  $1^\circ$ ), the trigonometric relationships can be simplified to fixed numbers. This will make the equations less cumbersome and will allow for faster computing time due to the elimination of the trigonometric calculation for simple integer math. This can be seen done in the following set of equations.

**Forward Transformation Equations:**

$$X = A + 0.0174 * B$$

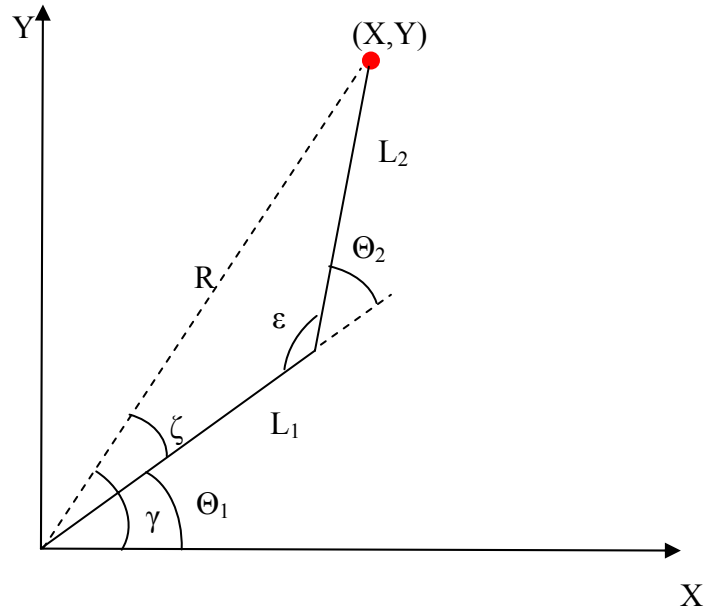
$$Y = B * .9998$$

**Inverse Transformation Equations:**

$$B = \frac{Y}{0.9998}$$

$$A = X - B * 0.0174$$

**Example 3:** Double jointed arm (based on a SCARA robotic arm design)



**Figure 5:** Double jointed arm with angle off of X axis  $\Theta_1$  and angle of joint  $\Theta_2$ . In this system  $\Theta_1$  and  $\Theta_2$  are variable and  $L_1$  and  $L_2$  are fixed and known.

**Forward Transformation Equations:**

$$X = L_1 * \cos(\Theta_1) + L_2 * \cos(\Theta_1 + \Theta_2)$$

$$Y = L_1 * \sin(\Theta_1) + L_2 * \sin(\Theta_1 + \Theta_2)$$

**Inverse Transformation Equations:**

Definition of angle  $\gamma$  and distance R:

$$\gamma = \tan^{-1}(Y/X)$$

$$R^2 = X^2 + Y^2$$

Derivation of  $\Theta_2$  from the law of cosines:

$$R^2 = L_1^2 + L_2^2 - (2 * L_1 * L_2) * \cos(\varepsilon)$$

Therefore:

$$\varepsilon = \cos^{-1} \left\{ \frac{X^2 + Y^2 - L_1^2 - L_2^2}{(-2 * L_1 * L_2)} \right\}$$

Resulting in:

$$\Theta_2 = 180 - \varepsilon$$

Or:

$$\Theta_2 = 180 - \cos^{-1} \left\{ \frac{X^2 + Y^2 - L_1^2 - L_2^2}{(-2 * L_1 * L_2)} \right\}$$

Supporting definition of  $\gamma$  using the sine theorem:

$$\frac{L_2}{\sin(\xi)} = \frac{R}{\sin(180 - \Theta_2)}$$

This simplifies to:

$$\frac{L_2}{\sin(\xi)} = \frac{R}{\sin(\Theta_2)}$$

Then:

$$\xi = \sin^{-1} \left[ \frac{L_2 * \sin(\Theta_2)}{R} \right]$$

Which yields:

$$\Theta_1 = \gamma - \xi$$

Or:

$$\Theta_1 = \sin^{-1} \left[ \frac{L_2 * \sin(\Theta_2)}{R} \right] - \tan^{-1}(Y/X)$$

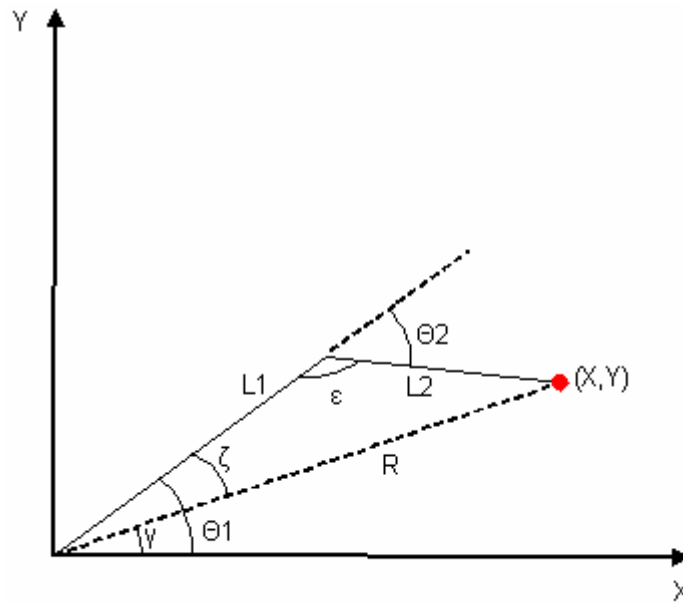
**Note:** The above analysis defines the variables  $R^2$ ,  $\Theta_1$ ,  $\Theta_2$ ,  $\zeta$  and  $\gamma$  given the point X, Y and lengths  $L_1$  and  $L_2$ . With coordinate transformation it is important to tailor the transformation equations to the system. For instance with the SCARA robot in Figure 5 it is mathematically possible to define the position of X and Y with radius R and angle  $\gamma$  using the equations below:

Definition of R and  $\gamma$  given X and Y:

$$R^2 = X^2 + Y^2$$
$$\gamma = \tan^{-1}(Y/X)$$

However using these 2 equations ignores the angle  $\varepsilon$ . This angle is important to define because it would dictate the position of a motor placed at that joint, which would be necessary in an actual physical system to control the arm.

Another consideration when developing transformation equations are the ranges in which the coordinate transformation equations remain valid. For instance in the case of the SCARA robot in Figure 5, the angle  $\Theta_2$  can swing to a negative angle. This results in the robot transforming from a “right hand” configuration to a “left hand” configuration. Changing the configuration of this joint from right to left handed will invalidate the given transformation equations. Specifically, this is because the angle  $\varepsilon$  was derived with the law of cosines. The law of cosines will only provide interior angles, like  $\varepsilon$  given  $L_1$ ,  $L_2$ ,  $X$  and  $Y$  in Figure 5. If the angle  $\Theta_2$  were to be negative than the reference for angle  $\varepsilon$  would switch sides of the arm and make the associated transformation equations invalid. To remedy this, a condition must be placed on the set of transformation equations changing the definition of  $\varepsilon$  when  $\Theta_2$  is negative, such as in Figure 6:



$$\varepsilon = 360 - \cos^{-1} \left\{ \frac{X^2 + Y^2 - L_1^2 - L_2^2}{(-2 * L_1 * L_2)} \right\}$$

**Figure 6:** Figure of the SCARA robotic arm with  $\Theta_2$  negative and the conditional transformation equation to define the angle  $\varepsilon$ .

Applying this conditional set of circumstances would be possible, and will provide for valid solutions for all given angles, lengths and positions, however in actual systems, this may not be necessary due to physical restraints. Also it may be more convenient to simply restrict the range of motion of the robot for all practical purposes and apply a single set of transformation equations.

There is another situation that may occur with coordinate transformation, where in which the coordinate transformation equations provide 2 different solutions given a single definition set of parameters, like angles and lengths. This phenomenon of dual solutions is mathematically correct, however only one solution physically exists. To avoid this, certain constraints must be placed on the system, either mathematically and/or physically when defining the equations.