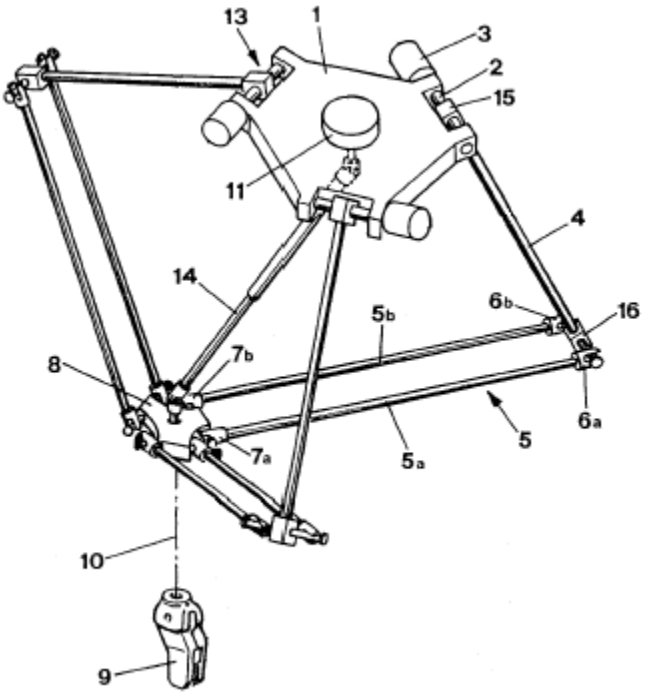
# The Delta Parallel Robot: Kinematics Solutions

**Robert L. Williams II, Ph.D., williar4@ohio.edu**
**Mechanical Engineering, Ohio University, October 2016**

Clavel's Delta Robot[1] is arguably the most successful commercial parallel robot to date. The left image below shows the original design from Clavel's U.S. patent[2], and the right photograph below shows one commercial instantiation of the Delta Robot.




**Delta Robot Design[1]**


**ABB FlexPicker Delta Robot**
www.abb.com

The Delta Robot has 4-degrees-of-freedom (dof), 3-dof for *XYZ* translation, plus a fourth inner leg to control a single rotational freedom at the end-effector platform (about the axis perpendicular to the platform). The remainder of this document will focus only on the 3-dof *XYZ* translation-only Delta Robot since that is being widely applied by 3D printers and Arduino hobbyists.

Presented is a description of the 3-dof Delta Robot, followed by kinematics analysis including analytical solutions for the inverse position kinematics problem and the forward position kinematics problem, and then examples for both, snapshots and trajectories. The velocity equations are also derived This is presented for both revolute-input and prismatic-input Delta Robots.

**For referencing this document, please use:**

R.L. Williams II, "The Delta Parallel Robot: Kinematics Solutions", Internet Publication, www.ohio.edu/people/williar4/html/pdf/DeltaKin.pdf, January 2016.

---

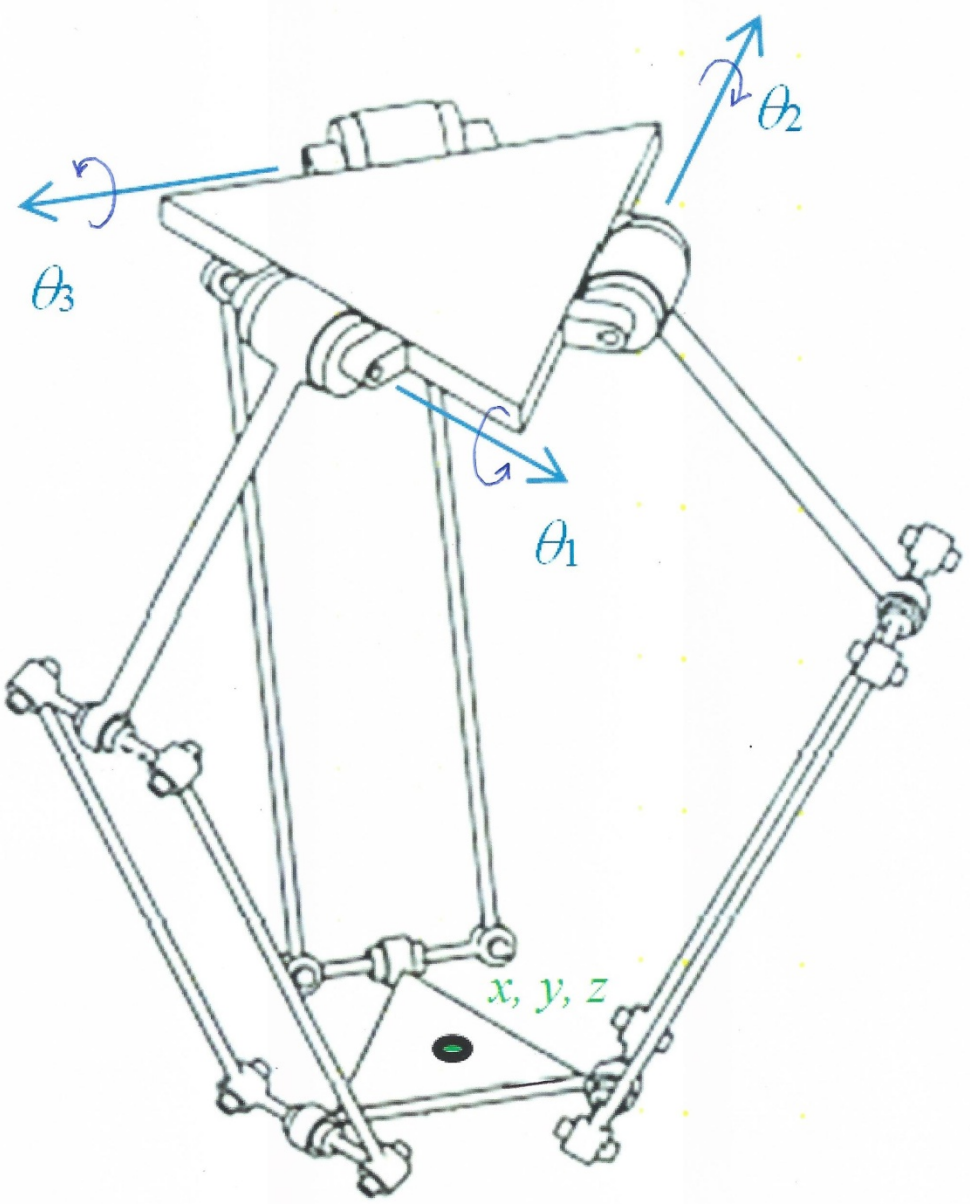
[1] R. Clavel, 1991, "Conception d'un robot parallèle rapide à 4 degrés de liberté", Ph.D. Thesis, EPFL, Lausanne, Switzerland.
[2] R. Clavel, 1990, "Device for the Movement and Positioning of an Element in Space", U.S. Patent No. 4,976,582.

# Table of Contents

# Revolute-Input Delta Robot

## Revolute-Input Delta Robot Description

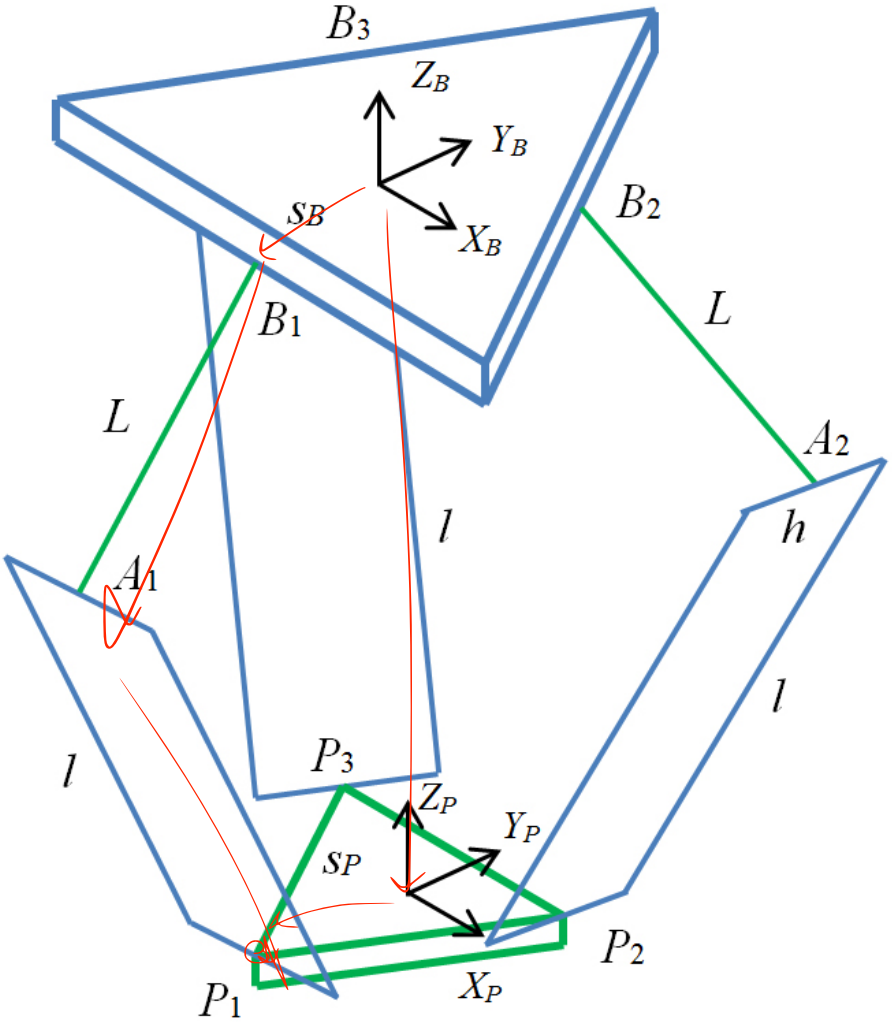
As shown below, the 3-dof Delta Robot is composed of three identical **RUU** legs in parallel between the top fixed base and the bottom moving end-effector platform. The top revolute joints are actuated (indicated by the underbar) via base-fixed rotational actuators. Their control variables are $\theta_i, \quad i=1,2,3$ about the axes shown. In this model $\theta_i$ are measured with the right hand, with zero angle defined as when the actuated link is in the horizontal plane. The parallelogram 4-bar mechanisms of the three lower links ensure the translation-only motion. The universal (**U**) joints are implemented using three non-collocated revolute (**R**) joints (two parallel and one perpendicular, six places) as shown below.




**Delta Parallel Robot Diagram**

adapted from: elmomc.com/capabilities

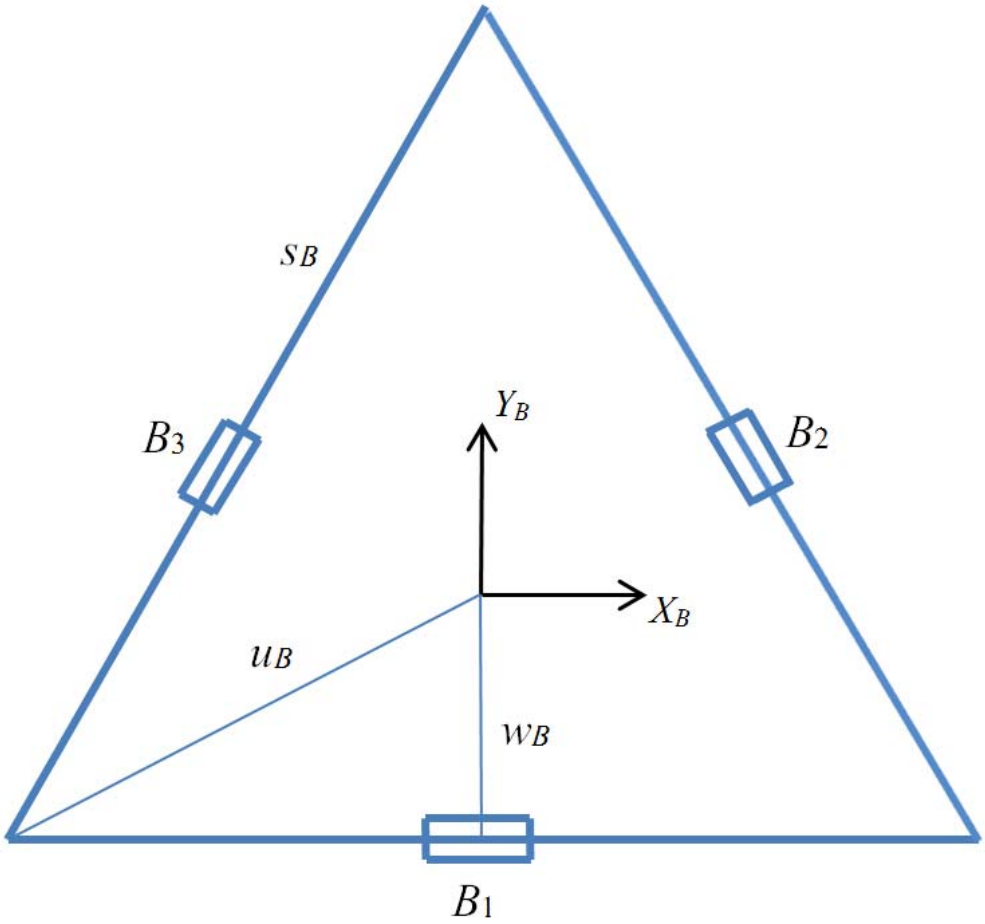
The three-dof Delta Robot is capable of *XYZ* translational control of its moving platform within its workspace. Viewing the three identical **RUU** chains as legs, points $B_i$, $i=1,2,3$ are the hips, points $A_i$, $i=1,2,3$ are the knees, and points $P_i$, $i=1,2,3$ are the ankles. The side length of the base equilateral triangle is $s_B$ and the side length of the moving platform equilateral triangle is $s_P$. The moving platform equilateral triangle is inverted with respect to the base equilateral triangle as shown, in a constant orientation.



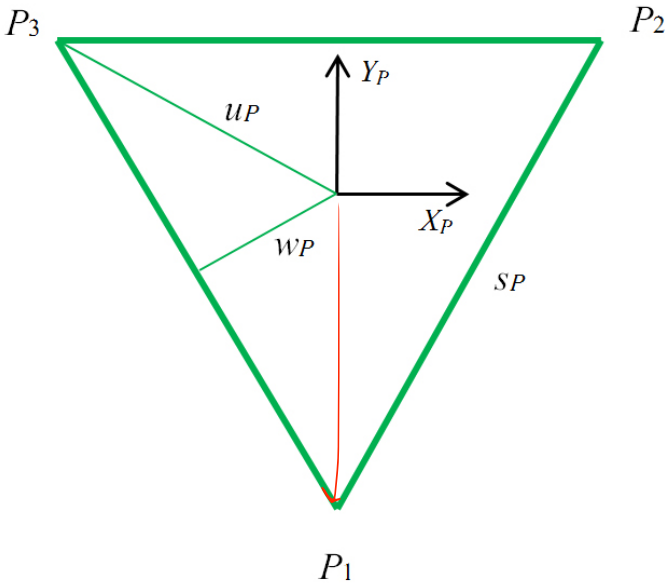

**Delta Robot Kinematic Diagram**

The fixed base Cartesian reference frame is {*B*}, whose origin is located in the center of the base equilateral triangle. The moving platform Cartesian reference frame is {*P*}, whose origin is located in the center of the platform equilateral triangle. The orientation of {*P*} is always identical to the orientation of {*B*} so rotation matrix ${}^B_P\mathbf{R}=\mathbf{I}_3$ is constant. The joint variables are $\mathbf{\Theta}=\{\theta_1 \quad \theta_2 \quad \theta_3\}^T$, and the Cartesian variables are ${}^B\mathbf{P}_P=\{x \quad y \quad z\}^T$. The design shown has high symmetry, with three upper leg lengths *L* and three lower lengths *l* (the parallelogram four-bar mechanisms major lengths).

The Delta Robot fixed base and platform geometric details are shown on the next page.




**Delta Robot Fixed Base Details**




**Delta Robot Moving Platform Details**

The fixed-base revolute joint points $B_i$ are constant in the base frame $\{B\}$ and the platform-fixed U-joint connection points $P_i$ are constant in the base frame $\{P\}$:

$$ {}^B\mathbf{B}_1 = \begin{Bmatrix} 0 \\ -w_B \\ 0 \end{Bmatrix} \qquad {}^B\mathbf{B}_2 = \begin{Bmatrix} \frac{\sqrt{3}}{2} w_B \\ \frac{1}{2} w_B \\ 0 \end{Bmatrix} \qquad {}^B\mathbf{B}_3 = \begin{Bmatrix} -\frac{\sqrt{3}}{2} w_B \\ \frac{1}{2} w_B \\ 0 \end{Bmatrix} $$

$$ {}^P\mathbf{P}_1 = \begin{Bmatrix} 0 \\ -u_P \\ 0 \end{Bmatrix} \qquad {}^P\mathbf{P}_2 = \begin{Bmatrix} \frac{s_P}{2} \\ w_P \\ 0 \end{Bmatrix} \qquad {}^P\mathbf{P}_3 = \begin{Bmatrix} -\frac{s_P}{2} \\ w_P \\ 0 \end{Bmatrix} $$

The vertices of the fixed-based equilateral triangle are:

$$ {}^B\mathbf{b}_1 = \begin{Bmatrix} \frac{s_B}{2} \\ -w_B \\ 0 \end{Bmatrix} \qquad {}^B\mathbf{b}_2 = \begin{Bmatrix} 0 \\ u_B \\ 0 \end{Bmatrix} \qquad {}^B\mathbf{b}_3 = \begin{Bmatrix} -\frac{s_B}{2} \\ -w_B \\ 0 \end{Bmatrix} $$

where:

$$ w_B = \frac{\sqrt{3}}{6} s_B \qquad u_B = \frac{\sqrt{3}}{3} s_B \qquad w_P = \frac{\sqrt{3}}{6} s_P \qquad u_P = \frac{\sqrt{3}}{3} s_P $$

| name | meaning | value (mm) |
|---|---|---|
| $s_B$ | base equilateral triangle side | 567 |
| $s_P$ | platform equilateral triangle side | 76 |
| $L$ | upper legs length | 524 |
| $l$ | lower legs parallelogram length | 1244 |
| $h$ | lower legs parallelogram width | 131 |
| $w_B$ | planar distance from {0} to near base side | 164 |
| $u_B$ | planar distance from {0} to a base vertex | 327 |
| $w_P$ | planar distance from $\{P\}$ to near platform side | 22 |
| $u_P$ | planar distance from $\{P\}$ to a platform vertex | 44 |

The model values above are for a specific commercial delta robot, the ABB FlexPicker IRB 360-1/1600, scaled from a figure (new.abb.com/products). Though Delta Robot symmetry is assumed, the following methods may be adapted to the general case.

## Revolute-Input Delta Robot Mobility

This section proves that the mobility (the number of degrees-of-freedom) for the Delta robot is indeed 3-dof. Using the spatial Kutzbach mobility equation for the previous Delta Robot figure:

$$M = 6(N-1) - 5J_1 - 4J_2 - 3J_3$$

where:

$M$ is the mobility, or number of degrees-of-freedom
$N$ is the total number of links, including ground
$J_1$ is the number of one-dof joints
$J_2$ is the number of two-dof joints
$J_3$ is the number of three-dof joints

| | |
|---|---|
| $J_1$ – one-dof joints: | revolute and prismatic joints |
| $J_2$ – two-dof joints: | universal joint |
| $J_3$ – three-dof joints: | spherical joint |

For the as-designed Delta Robot, we have:

$N = 17$
$J_1 = 21$
$J_2 = 0$
$J_3 = 0$

$M = 6(17-1) - 5(21) - 4(0) - 3(0)$
$M = -9$ dof

As often happens, the Kutzbach equation fails because the result must obviously be 3-dof. This result predicts the Delta is a severely overconstrained statically indeterminate structure, which is incorrect.

The Kutzbach equation knows nothing about special geometry – in the Delta Robot case, there are three parallel four-bar mechanisms. The overall robot would work kinematically identically to the original Delta Robot if we removed one of the long parallel four-bar mechanism links, along with two revolute joints each. With this equivalent case, the Kutzbach equation yields:

$N = 14$
$J_1 = 15$
$J_2 = 0$
$J_3 = 0$

$M = 6(14-1) - 5(15) - 4(0) - 3(0)$
$M = 3$ dof

which is correct. An alternative approach to calculating the Delta Robot mobility is by ignoring the three parallel four-bar mechanisms, replacing each with a single link instead. In this we must count a universal joint at either end of this virtual link. This approach follows the simplified Delta Robot naming convention **3-<u>R</u>UU**. The Kutzbach equation for this case also succeeds:

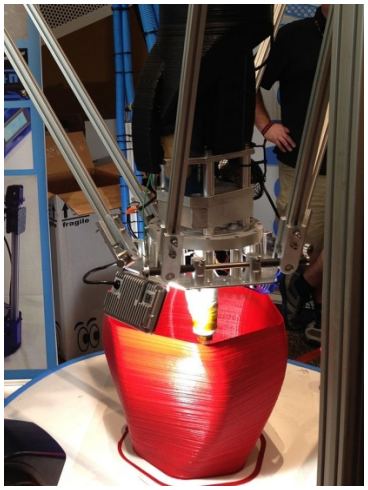
$$N = 8$$
$$J_1 = 3 \qquad M = 6(8-1) - 5(3) - 4(6) - 3(0)$$
$$J_2 = 6 \qquad M = 3 \quad \text{dof}$$
$$J_3 = 0$$

Either of the second two approaches works. The author prefers the former since it is closer to the actual Delta Robot design.

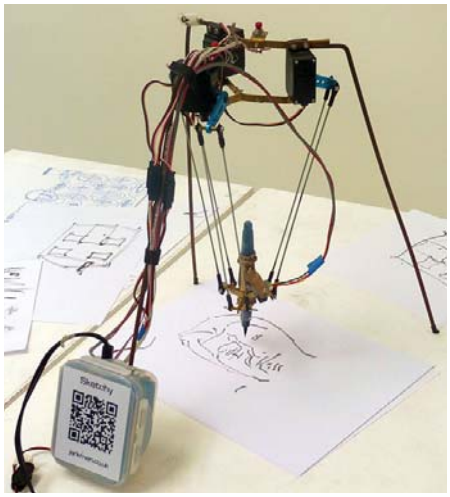
## Practical Revolute-Input Delta Robots

**Delta Chocolate-Handling Robot**
sti.epfl.ch

**Delta Pick-and-Place Robots**
en.wikipedia.org/wiki/Delta_robot



**Delta Robot 3D Printer**
en.wikipedia.org/wiki/Delta_robot



**Sketchy Delta Robot**
en.wikipedia.org/wiki/Delta_robot

**Novint Falcon Haptic Interface**
images.bit-tech.net

**Fanuc Delta Robot**
robot.fanucamerica.com

## Revolute-Input Delta Robot Kinematics Analysis

From the kinematic diagram above, the following three vector-loop closure equations are written for the Delta Robot:

$$\left\{{}^{B}\mathbf{B}_i\right\}+\left\{{}^{B}\mathbf{L}_i\right\}+\left\{{}^{B}\mathbf{l}_i\right\}=\left\{{}^{B}\mathbf{P}_P\right\}+\left[{}^{B}_{P}\mathbf{R}\right]\left\{{}^{P}\mathbf{P}_i\right\}=\left\{{}^{B}\mathbf{P}_P\right\}+\left\{{}^{P}\mathbf{P}_i\right\} \qquad i=1,2,3$$

where $\left[{}^{B}_{P}\mathbf{R}\right]=\left[\mathbf{I}_3\right]$ since no rotations are allowed by the Delta Robot.

The three applicable constraints state that the lower leg lengths must have the correct, constant length $l$ (the virtual length through the center of each parallelogram):

$$l_i=\left\|\left\{{}^{B}\mathbf{l}_i\right\}\right\|=\left\|\left\{{}^{B}\mathbf{P}_P\right\}+\left\{{}^{P}\mathbf{P}_i\right\}-\left\{{}^{B}\mathbf{B}_i\right\}-\left\{{}^{B}\mathbf{L}_i\right\}\right\| \qquad i=1,2,3$$

It will be more convenient to square both sides of the constraint equations above to avoid the square-root in the Euclidean norms:

$$l_i^2=\left\|\left\{{}^{B}\mathbf{l}_i\right\}\right\|^2=l_{ix}^2+l_{iy}^2+l_{iz}^2 \qquad i=1,2,3$$

Again, the Cartesian variables are ${}^{B}\mathbf{P}_P=\left\{x \quad y \quad z\right\}^T$. The constant vector values for points $P_i$ and $B_i$ were given previously. The vectors $\left\{{}^{B}\mathbf{L}_i\right\}$ are dependent on the joint variables $\mathbf{\Theta}=\left\{\theta_1 \quad \theta_2 \quad \theta_3\right\}^T$:

$${}^{B}\mathbf{L}_1=\begin{Bmatrix}0\\-L\cos\theta_1\\-L\sin\theta_1\end{Bmatrix} \qquad {}^{B}\mathbf{L}_2=\begin{Bmatrix}\frac{\sqrt{3}}{2}L\cos\theta_2\\\frac{1}{2}L\cos\theta_2\\-L\sin\theta_2\end{Bmatrix} \qquad {}^{B}\mathbf{L}_3=\begin{Bmatrix}-\frac{\sqrt{3}}{2}L\cos\theta_3\\\frac{1}{2}L\cos\theta_3\\-L\sin\theta_3\end{Bmatrix}$$

Substituting all above values into the vector-loop closure equations yields:

$$\left\{{}^{B}\mathbf{l}_1\right\}=\begin{Bmatrix}x\\y+L\cos\theta_1+a\\z+L\sin\theta_1\end{Bmatrix} \qquad \left\{{}^{B}\mathbf{l}_2\right\}=\begin{Bmatrix}x-\frac{\sqrt{3}}{2}L\cos\theta_2+b\\y-\frac{1}{2}L\cos\theta_2+c\\z+L\sin\theta_2\end{Bmatrix} \qquad \left\{{}^{B}\mathbf{l}_3\right\}=\begin{Bmatrix}x+\frac{\sqrt{3}}{2}L\cos\theta_3-b\\y-\frac{1}{2}L\cos\theta_3+c\\z+L\sin\theta_3\end{Bmatrix}$$

$$a = w_B - u_P$$

where: $$b = \frac{s_P}{2} - \frac{\sqrt{3}}{2} w_B$$

$$c = w_P - \frac{1}{2} w_B$$

And the three constraint equations yield the kinematics equations for the Delta Robot:

$$\begin{cases} 2L(y+a)\cos\theta_1 + 2zL\sin\theta_1 + x^2 + y^2 + z^2 + a^2 + L^2 + 2ya - l^2 = 0 \\ -L(\sqrt{3}(x+b) + y + c)\cos\theta_2 + 2zL\sin\theta_2 + x^2 + y^2 + z^2 + b^2 + c^2 + L^2 + 2xb + 2yc - l^2 = 0 \\ L(\sqrt{3}(x-b) - y - c)\cos\theta_3 + 2zL\sin\theta_3 + x^2 + y^2 + z^2 + b^2 + c^2 + L^2 - 2xb + 2yc - l^2 = 0 \end{cases}$$

The three absolute vector knee points are found using ${}^B\mathbf{A}_i = {}^B\mathbf{B}_i + {}^B\mathbf{L}_i$, $i = 1,2,3$:

$${}^B\mathbf{A}_1 = \begin{Bmatrix} 0 \\ -w_B - L\cos\theta_1 \\ -L\sin\theta_1 \end{Bmatrix} \qquad {}^B\mathbf{A}_2 = \begin{Bmatrix} \frac{\sqrt{3}}{2}(w_B + L\cos\theta_2) \\ \frac{1}{2}(w_B + L\cos\theta_2) \\ -L\sin\theta_2 \end{Bmatrix} \qquad {}^B\mathbf{A}_3 = \begin{Bmatrix} -\frac{\sqrt{3}}{2}(w_B + L\cos\theta_3) \\ \frac{1}{2}(w_B + L\cos\theta_3) \\ -L\sin\theta_3 \end{Bmatrix}$$

## Inverse Position Kinematics (IPK) Solution

The 3-dof Delta Robot inverse position kinematics (IPK) problem is stated: Given the Cartesian position of the moving platform control point (the origin of {*P*}), ${}^B\mathbf{P}_P = \{x \quad y \quad z\}^T$, calculate the three required actuated revolute joint angles $\boldsymbol{\Theta} = \{\theta_1 \quad \theta_2 \quad \theta_3\}^T$. The IPK solution for parallel robots is often straightforward; the IPK solution for the Delta Robot is not trivial, but can be found analytically. Referring to the Delta Robot kinematic diagram above, the IPK problem can be solved independently for each of the three **RUU** legs. Geometrically, each leg IPK solution is the intersection between a known circle (radius *L*, centered on the base triangle **R** joint point ${}^B\mathbf{B}_i$) and a known sphere (radius *l*, centered on the moving platform vertex ${}^P\mathbf{P}_i$).

This solution may be done geometrically/trigonometrically. However, we will now accomplish this IPK solution analytically, using the three constraint equations applied to the vector loop-closure equations (derived previously). The three independent scalar IPK equations are of the form:

$$E_i\cos\theta_i + F_i\sin\theta_i + G_i = 0 \qquad i = 1,2,3$$

where:

$$E_1 = 2L(y+a)$$
$$F_1 = 2zL$$
$$G_1 = x^2 + y^2 + z^2 + a^2 + L^2 + 2ya - l^2$$

$$E_2 = -L(\sqrt{3}(x+b) + y + c)$$
$$F_2 = 2zL$$
$$G_2 = x^2 + y^2 + z^2 + b^2 + c^2 + L^2 + 2(xb + yc) - l^2$$

$$E_3 = L(\sqrt{3}(x-b) - y - c)$$
$$F_3 = 2zL$$
$$G_3 = x^2 + y^2 + z^2 + b^2 + c^2 + L^2 + 2(-xb + yc) - l^2$$

The equation $E_i \cos\theta_i + F_i \sin\theta_i + G_i = 0$ appears a lot in robot and mechanism kinematics and is readily solved using the **Tangent Half-Angle Substitution**.

If we define $t_i = \tan\frac{\theta_i}{2}$ then $\cos\theta_i = \frac{1-t_i^2}{1+t_i^2}$ and $\sin\theta_i = \frac{2t_i}{1+t_i^2}$

Substitute the **Tangent Half-Angle Substitution** into the *EFG* equation:

$$E_i\left(\frac{1-t_i^2}{1+t_i^2}\right) + F_i\left(\frac{2t_i}{1+t_i^2}\right) + G_i = 0 \qquad E_i(1-t_i^2) + F_i(2t_i) + G_i(1+t_i^2) = 0$$

$$(G_i - E_i)t_i^2 + (2F_i)t_i + (G_i + E_i) = 0 \qquad \text{quadratic formula: } t_{i_{1,2}} = \frac{-F_i \pm \sqrt{E_i^2 + F_i^2 - G_i^2}}{G_i - E_i}$$

Solve for $\theta_i$ by inverting the original Tangent Half-Angle Substitution definition:

$$\theta_i = 2\tan^{-1}(t_i)$$

Two $\theta_i$ solutions result from the $\pm$ in the quadratic formula. Both are correct since there are two valid solutions – knee left and knee right. This yields two IPK branch solutions for each leg of the Delta Robot, for a total of 8 possible valid solutions. Generally the one solution with all knees kinked out instead of in will be chosen.

## Forward Position Kinematics (FPK) Solution

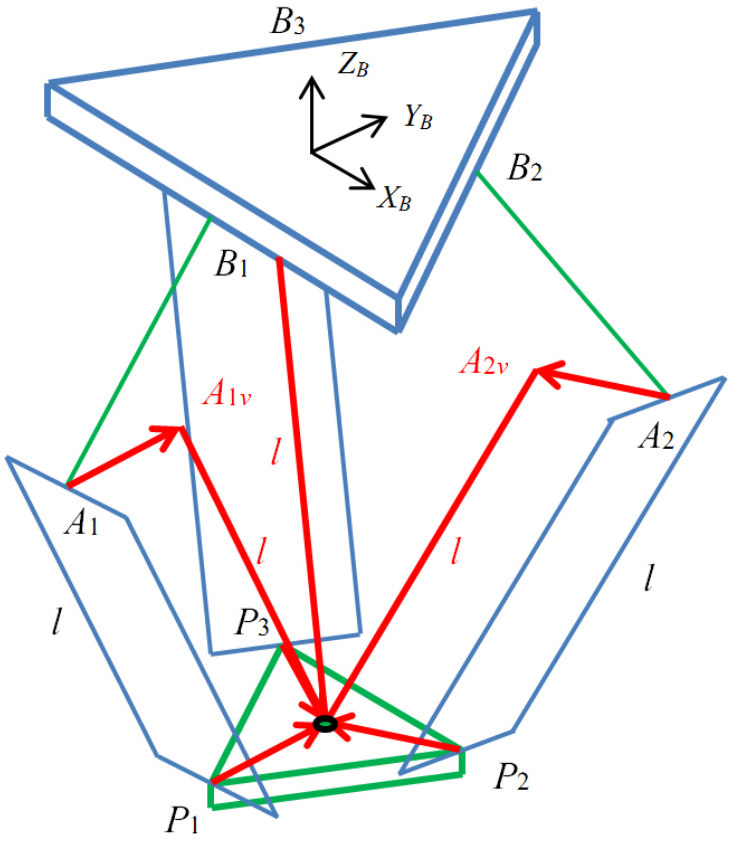
The 3-dof Delta Robot forward position kinematics (FPK) problem is stated: Given the three actuated joint angles $\boldsymbol{\Theta} = \{\theta_1 \quad \theta_2 \quad \theta_3\}^T$, calculate the resulting Cartesian position of the moving platform control point (the origin of $\{P\}$), ${}^B\mathbf{P}_P = \{x \quad y \quad z\}^T$. The FPK solution for parallel robots is generally very difficult. It requires the solution of multiple coupled nonlinear algebraic equations, from the three constraint equations applied to the vector loop-closure equations (derived previously). Multiple valid solutions generally result.

Thanks to the translation-only motion of the 3-dof Delta Robot, there is a straightforward analytical solution for which the correct solution set is easily chosen. Since $\boldsymbol{\Theta}=\{\theta_1 \quad \theta_2 \quad \theta_3\}^T$ are given, we calculate the three absolute vector knee points using ${}^B\mathbf{A}_i = {}^B\mathbf{B}_i + {}^B\mathbf{L}_i$, $i=1,2,3$. Referring to the Delta Robot FPK diagram below, since we know that the moving platform orientation is constant, always horizontal with $\left[{}^B_P\mathbf{R}\right]=[\mathbf{I}_3]$, we define three virtual sphere centers ${}^B\mathbf{A}_{iv} = {}^B\mathbf{A}_i - {}^P\mathbf{P}_i$, $i=1,2,3$:

$$ {}^B\mathbf{A}_{1v} = \begin{Bmatrix} 0 \\ -w_B - L\cos\theta_1 + u_P \\ -L\sin\theta_1 \end{Bmatrix} \quad {}^B\mathbf{A}_{2v} = \begin{Bmatrix} \frac{\sqrt{3}}{2}(w_B + L\cos\theta_2) - \frac{s_P}{2} \\ \frac{1}{2}(w_B + L\cos\theta_2) - w_P \\ -L\sin\theta_2 \end{Bmatrix} \quad {}^B\mathbf{A}_{3v} = \begin{Bmatrix} -\frac{\sqrt{3}}{2}(w_B + L\cos\theta_3) + \frac{s_P}{2} \\ \frac{1}{2}(w_B + L\cos\theta_3) - w_P \\ -L\sin\theta_3 \end{Bmatrix} $$

and then the Delta Robot FPK solution is the intersection point of three known spheres. Let a sphere be referred as a vector center point {**c**} and scalar radius *r*, ({**c**}, *r*). Therefore, the FPK unknown point $\left\{{}^B\mathbf{P}_P\right\}$ is the intersection of the three known spheres:

$$(\left\{{}^B\mathbf{A}_{1v}\right\}, l) \qquad (\left\{{}^B\mathbf{A}_{2v}\right\}, l) \qquad (\left\{{}^B\mathbf{A}_{3v}\right\}, l).$$




**Delta Robot FPK Diagram**

Appendix A presents an analytical solution for the intersection point of the three given spheres, from Williams et al.[3] This solution also requires the solving of coupled transcendental equations. The appendix presents the equations and analytical solution methods, and then discusses imaginary solutions, singularities, and multiple solutions that can plague the algorithm, but all turn out to be no problem in this design.

In particular, with this existing three-spheres-intersection algorithm, if all three given sphere centers $\left\{ {}^{B}\mathbf{A}_{iv} \right\}$ have the same *Z* height (a common case for the Delta Robot), there will be an algorithmic singularity preventing a successful solution (dividing by zero). One way to fix this problem is to simple rotate coordinates so all $\left\{ {}^{B}\mathbf{A}_{iv} \right\}$ *Z* values are no longer the same, taking care to reverse this coordinate transformation after the solution is accomplished. However, we present another solution (Appendix B) for the intersection of three spheres assuming that all three sphere Z heights are identical, to be used in place of the primary solution when necessary.

Another applicable problem to be addressed is that the intersection of three spheres yields two solutions in general (only one solution if the spheres meet tangentially, and zero solutions if the center distance is too great for the given sphere radii *l* – in this latter case the solution is imaginary and the input data is not consistent with Delta Robot assembly). The spheres-intersection algorithm calculates both solution sets and it is possible to automatically make the computer choose the correct solution by ensuring it is below the base triangle rather than above it.

This three-spheres-intersection approach to the FPK for the Delta Robot yields results identical to solving the three kinematics equations for ${}^{B}\mathbf{P}_{P} = \left\{ x \quad y \quad z \right\}^{T}$ given $\mathbf{\Theta} = \left\{ \theta_1 \quad \theta_2 \quad \theta_3 \right\}^{T}$.

[3] R.L. Williams II, J.S. Albus, and R.V. Bostelman, 2004, "3D Cable-Based Cartesian Metrology System", Journal of Robotic Systems, 21(5): 237-257.

## Revolute-Input Delta Robot Velocity Kinematics Equations

The revolute-input Delta Robot velocity kinematics equations come from the first time derivative of the three position constraint equations presented earlier:

$$2L\dot{y}\cos\theta_1 - 2L(y+a)\dot{\theta}_1\sin\theta_1 + 2L\dot{z}\sin\theta_1 + 2Lz\dot{\theta}_1\cos\theta_1 + 2x\dot{x} + 2(y+a)\dot{y} + 2z\dot{z} = 0$$
$$-L(\sqrt{3}\dot{x}+\dot{y})\cos\theta_2 + L(\sqrt{3}(x+b)+y+c)\dot{\theta}_2\sin\theta_2 + 2L\dot{z}\sin\theta_2 + 2Lz\dot{\theta}_2\cos\theta_2 + 2(x+b)\dot{x} + 2(y+c)\dot{y} + 2z\dot{z} = 0$$
$$L(\sqrt{3}\dot{x}-\dot{y})\cos\theta_3 - L(\sqrt{3}(x-b)-y-c)\dot{\theta}_3\sin\theta_3 + 2L\dot{z}\sin\theta_3 + 2Lz\dot{\theta}_3\cos\theta_3 + 2(x-b)\dot{x} + 2(y+c)\dot{y} + 2z\dot{z} = 0$$

Re-written:

$$x\dot{x} + (y+a)\dot{y} + L\dot{y}\cos\theta_1 + z\dot{z} + L\dot{z}\sin\theta_1 = L(y+a)\dot{\theta}_1\sin\theta_1 - Lz\dot{\theta}_1\cos\theta_1$$
$$2(x+b)\dot{x} + 2(y+c)\dot{y} - L(\sqrt{3}\dot{x}+\dot{y})\cos\theta_2 + 2z\dot{z} + 2L\dot{z}\sin\theta_2 = -L(\sqrt{3}(x+b)+y+c)\dot{\theta}_2\sin\theta_2 - 2Lz\dot{\theta}_2\cos\theta_2$$
$$2(x-b)\dot{x} + 2(y+c)\dot{y} + L(\sqrt{3}\dot{x}-\dot{y})\cos\theta_3 + 2z\dot{z} + 2L\dot{z}\sin\theta_3 = L(\sqrt{3}(x-b)-y-c)\dot{\theta}_3\sin\theta_3 - 2Lz\dot{\theta}_3\cos\theta_3$$

Written in matrix-vector form:

$$[\mathbf{A}]\{\dot{\mathbf{X}}\} = [\mathbf{B}]\{\dot{\mathbf{\Theta}}\}$$

$$\begin{bmatrix} x & y+a+L\cos\theta_1 & z+L\sin\theta_1 \\ 2(x+b)-\sqrt{3}L\cos\theta_2 & 2(y+c)-L\cos\theta_2 & 2(z+L\sin\theta_2) \\ 2(x-b)+\sqrt{3}L\cos\theta_3 & 2(y+c)-L\cos\theta_3 & 2(z+L\sin\theta_3) \end{bmatrix} \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{bmatrix} b_{11} & 0 & 0 \\ 0 & b_{22} & 0 \\ 0 & 0 & b_{33} \end{bmatrix} \begin{Bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix}$$

where:

$$b_{11} = L\left[(y+a)\sin\theta_1 - z\cos\theta_1\right]$$
$$b_{22} = -L\left[(\sqrt{3}(x+b)+y+c)\sin\theta_2 + 2z\cos\theta_2\right]$$
$$b_{33} = L\left[(\sqrt{3}(x-b)-y-c)\sin\theta_3 - 2z\cos\theta_3\right]$$

## Revolute-Input Delta Robot Position Kinematics Examples

For these examples, the Delta Robot dimensions are from the table given earlier for the ABB FlexPicker IRB 360-1/1600, $s_B = 0.567$, $s_P = 0.076$, $L = 0.524$, $l = 1.244$, and $h = 0.131$ (m).

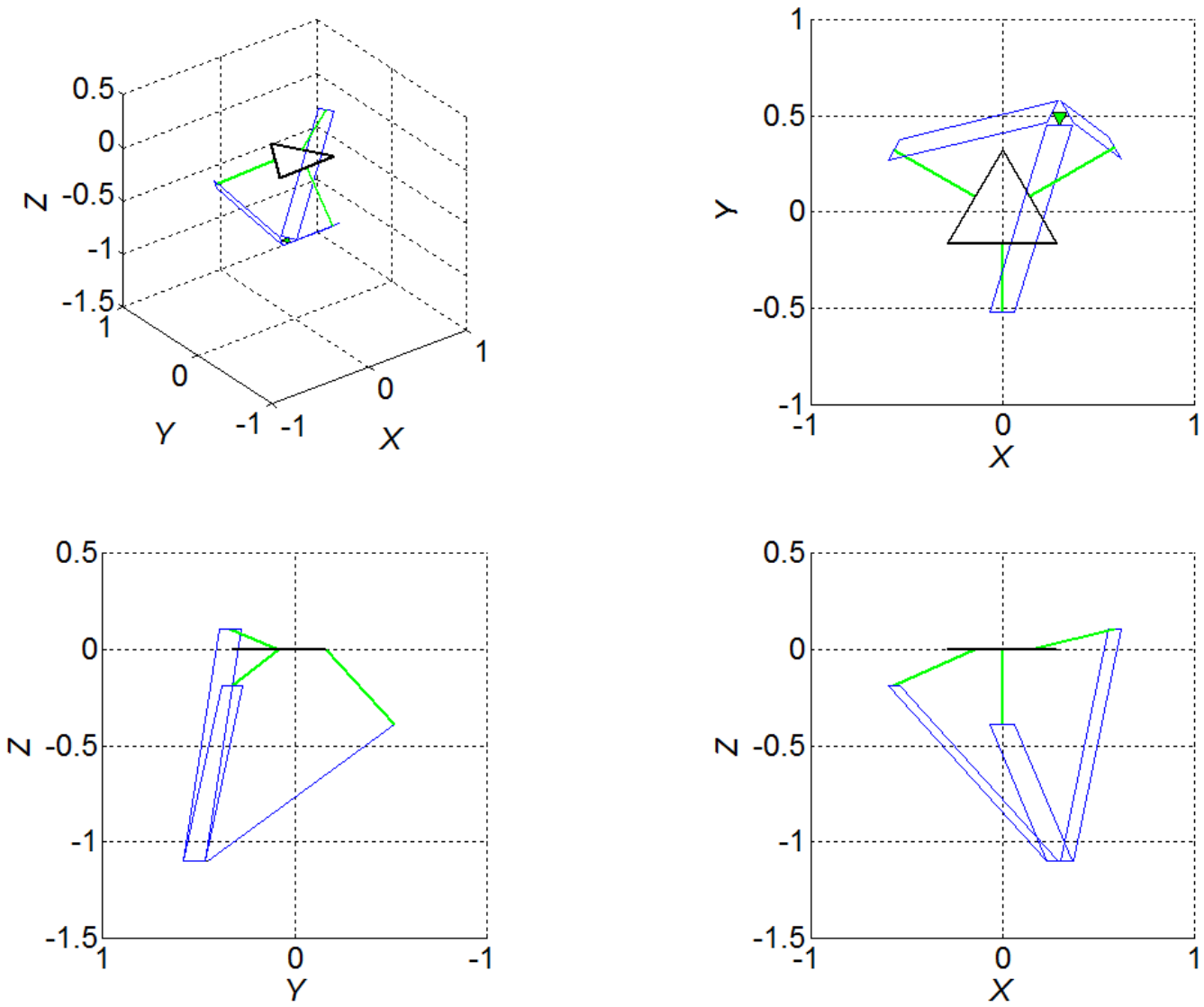
### Inverse Position Kinematics Examples

**Snapshot Examples** **Nominal Position.** Given $\left\{ {}^{B}\mathbf{P}_{P} \right\} = \left\{ 0 \quad 0 \quad -0.9 \right\}^{T}$ m, the calculated IPK results are (the preferred kinked out solution):

$$\boldsymbol{\Theta} = \left\{ -20.5^{\circ} \quad -20.5^{\circ} \quad -20.5^{\circ} \right\}^{T}$$

**General Position.** Given $\left\{ {}^{B}\mathbf{P}_{P} \right\} = \left\{ 0.3 \quad 0.5 \quad -1.1 \right\}^{T}$ m, the calculated IPK results are (the preferred kinked out solution):

$$\boldsymbol{\Theta} = \left\{ 47.5^{\circ} \quad -11.6^{\circ} \quad 21.4^{\circ} \right\}^{T}$$

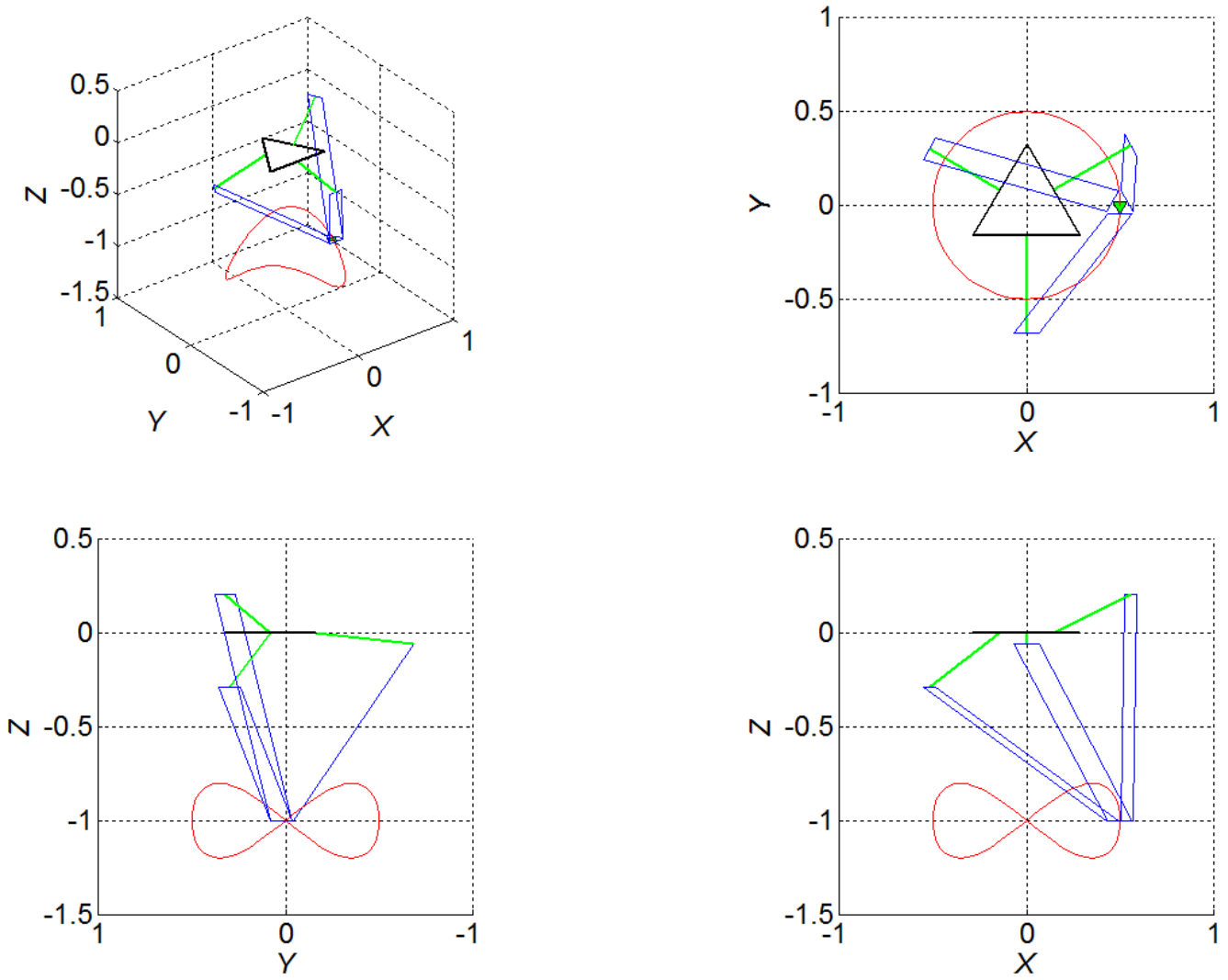



**General Inverse Position Snapshot Example**

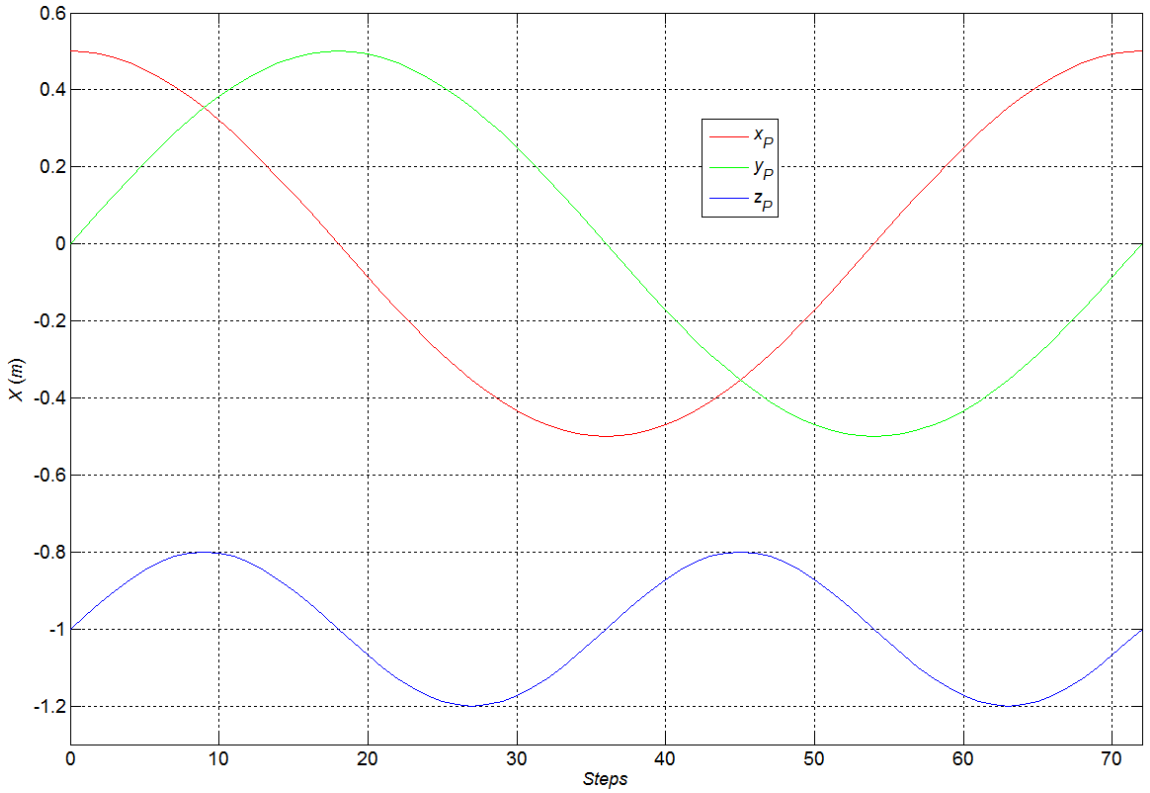
**IPK Trajectory Example**

The moving platform control point $\{P\}$ traces an $XY$ circle of center $\{0 \quad 0 \quad -1\}^T$ m and radius 0.5 m. At the same time, the $Z$ displacement goes through 2 complete sine wave motions centered on $Z = -1$ m with a 0.2 m amplitude.

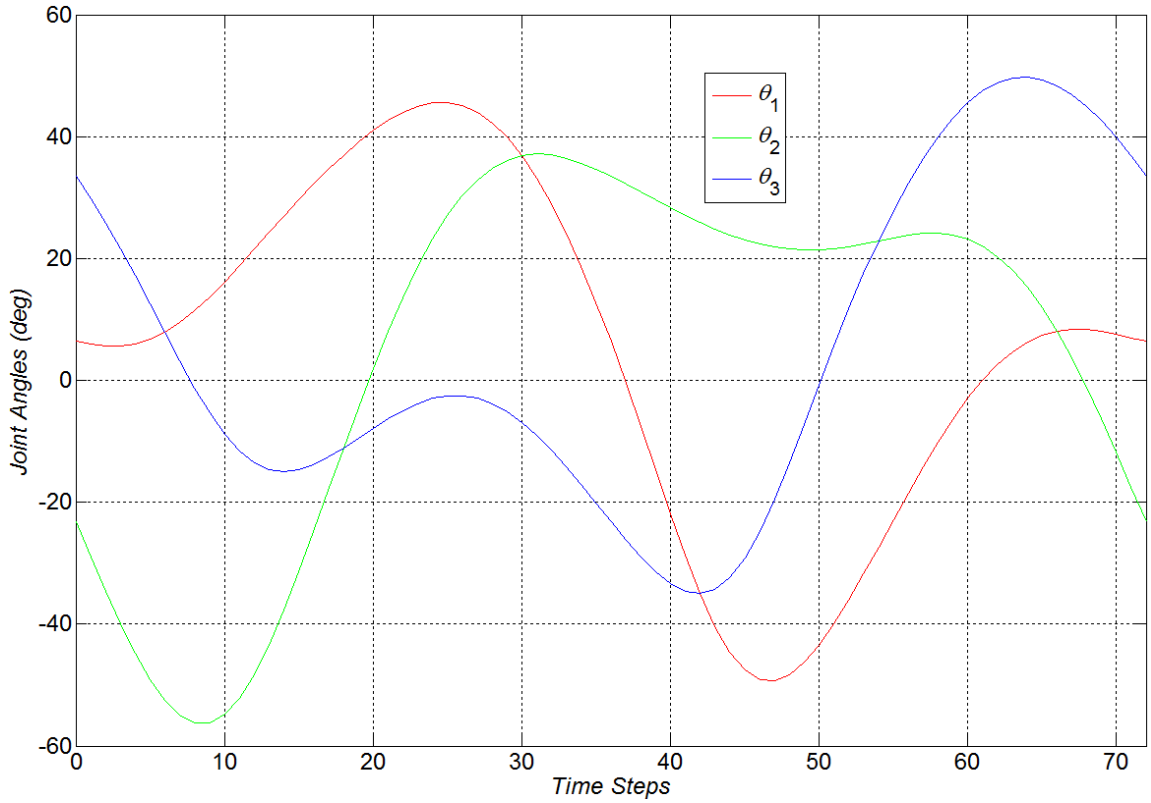
This IPK trajectory, at the end of motion, is pictured along with the simulated Delta Parallel Robot, in the MATLAB graphics below.




***XY* Circular Trajectory with *Z* sine wave**




**Commanded IPK Cartesian Positions**




**Calculated IPK Joint Angles**

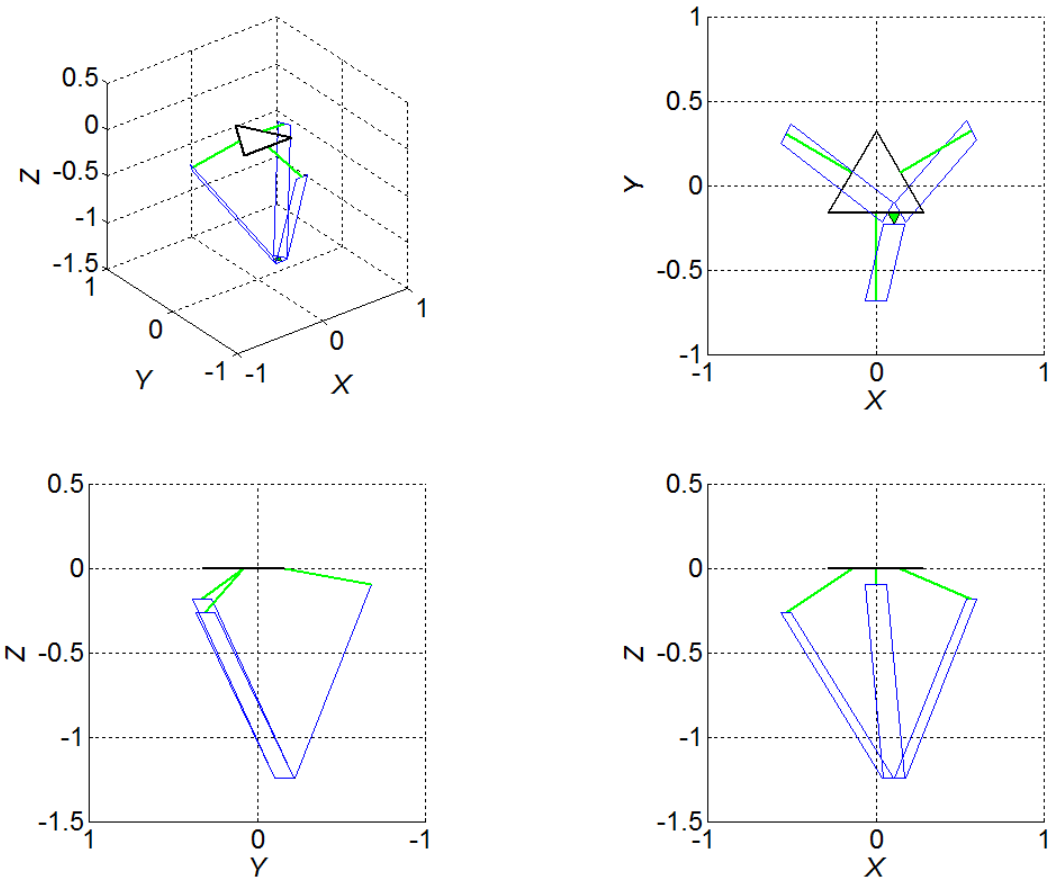
## Forward Position Kinematics Examples

**Snapshot Examples** **Nominal Position.** Given $\boldsymbol{\Theta} = \{0 \quad 0 \quad 0\}^T$, the calculated FPK results are (the admissible solution below the base, using the equal-*Z*-heights spheres intersection algorithm):

$$\left\{ {}^B\mathbf{P}_P \right\} = \{0 \quad 0 \quad -1.065\}^T \text{ m}$$

**General Position.** Given $\boldsymbol{\Theta} = \{10^\circ \quad 20^\circ \quad 30^\circ\}^T$ m, the calculated FPK results are (the admissible solution below the base, using the non-equal-*Z*-heights spheres intersection algorithm):

$$\left\{ {}^B\mathbf{P}_P \right\} = \{0.108 \quad -0.180 \quad -1.244\}^T \text{ m}$$




**General Forward Position Snapshot Example**

**Circular Check Examples**

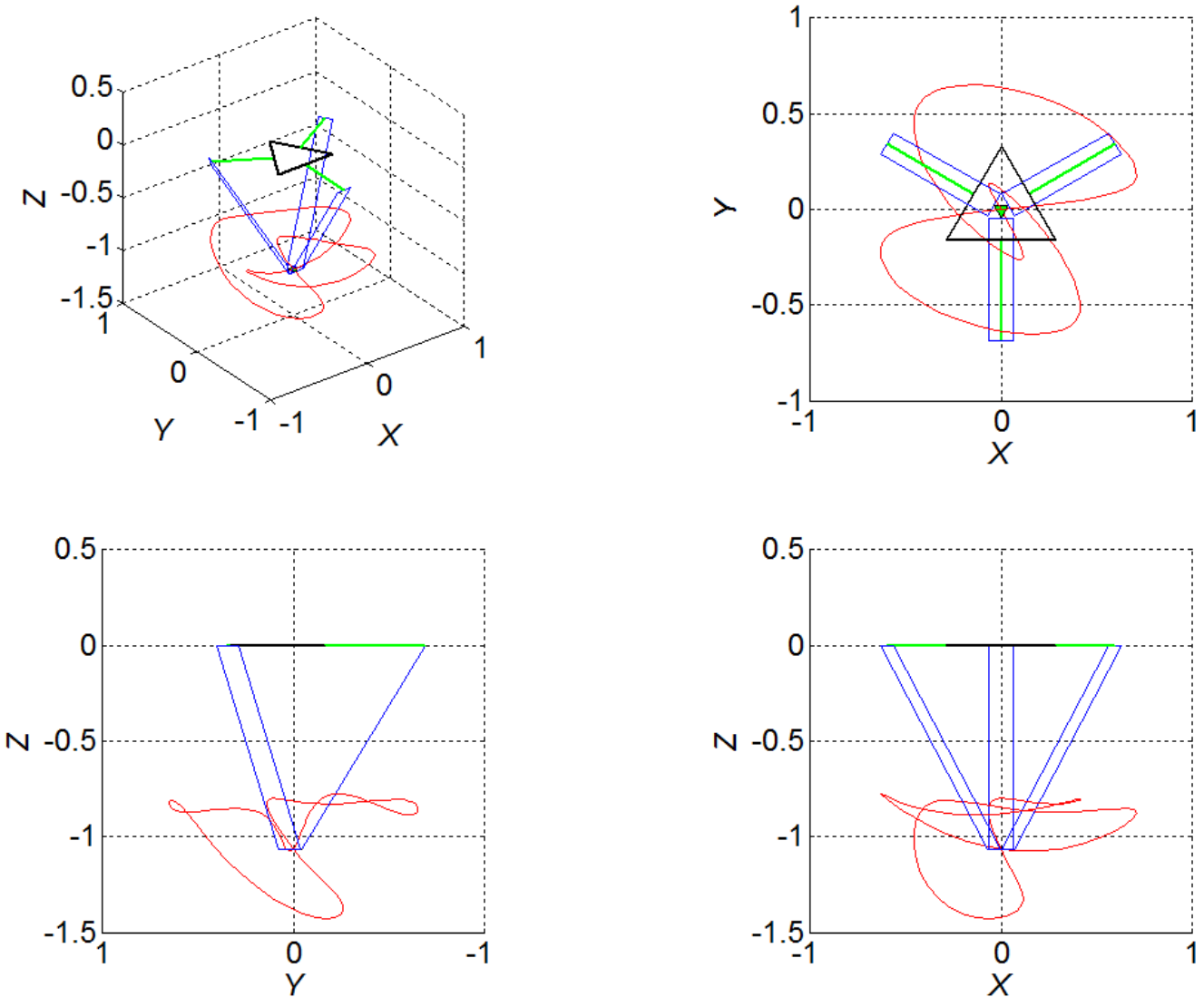
All snapshot examples were reversed to show that the circular check validation works; i.e.: When given $\left\{ {}^B\mathbf{P}_P \right\} = \{0 \quad 0 \quad -1.065\}^T$, the **IPK** solution calculated $\boldsymbol{\Theta} = \{0 \quad 0 \quad 0\}^T$ and when given $\left\{ {}^B\mathbf{P}_P \right\} = \{0.108 \quad -0.180 \quad -1.244\}^T$, the **IPK** solution calculated $\boldsymbol{\Theta} = \{10^\circ \quad 20^\circ \quad 30^\circ\}^T$. When given $\boldsymbol{\Theta} = \{-20.5^\circ \quad -20.5^\circ \quad -20.5^\circ\}^T$, the **FPK** solution calculated $\left\{ {}^B\mathbf{P}_P \right\} = \{0 \quad 0 \quad -0.9\}^T$ and when given $\boldsymbol{\Theta} = \{47.5^\circ \quad -11.6^\circ \quad 21.4^\circ\}^T$, the **FPK** solution calculated $\left\{ {}^B\mathbf{P}_P \right\} = \{0.3 \quad 0.5 \quad -1.1\}^T$.

**FPK Trajectory Example**

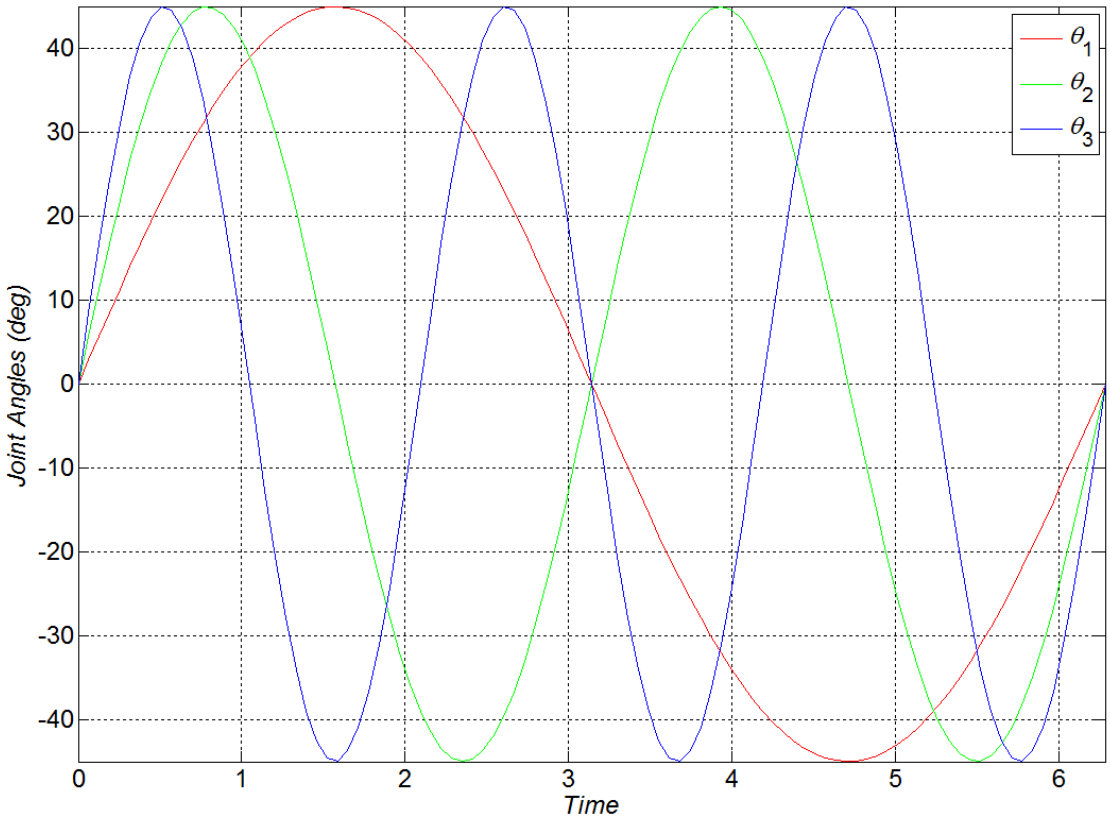
The IPK solution is more useful for Delta Robot control, to specify where the tool should be in *XYZ*. This FPK trajectory example is just for demonstration purposes, not yielding any useful robot motion. Simultaneously the three revolute joint angles are commanded as follows:

$$\theta_1(t) = \theta_{\max} \sin(t)$$
$$\theta_2(t) = \theta_{\max} \sin(2t)$$
$$\theta_3(t) = \theta_{\max} \sin(3t)$$

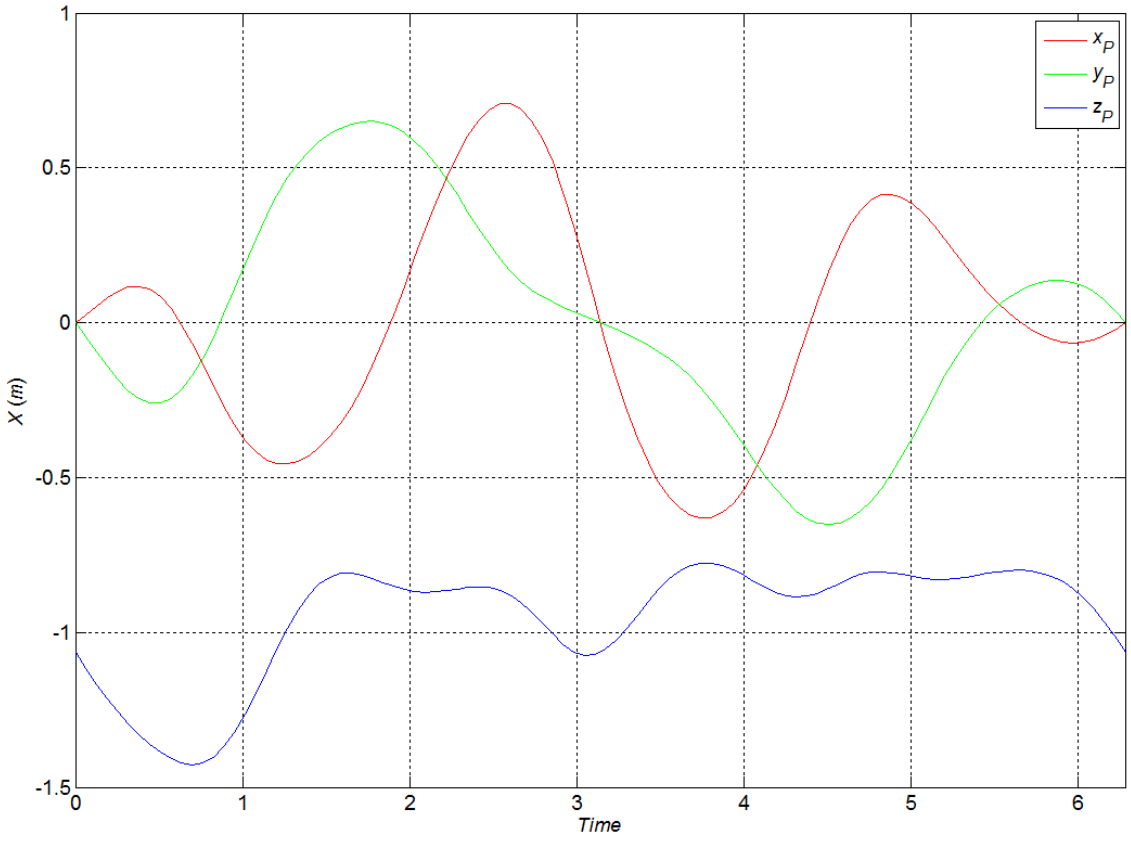
where $\theta_{\max} = 45^\circ$ and $t$ proceeds from 0 to $2\pi$ in 100 steps. This closed FPK trajectory, at the end of motion, is pictured along with the simulated Delta Parallel Robot, in the MATLAB graphics below




**FPK Trajectory**




**Commanded FPK Joint Angles**



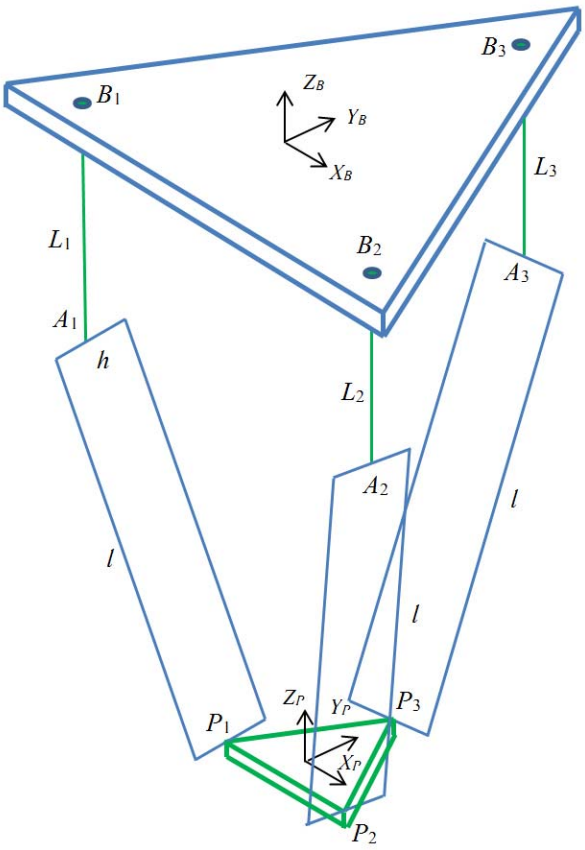

**Calculated FPK Cartesian Positions**

# Prismatic-Input Delta Robot

## Prismatic-Input Delta Robot Description

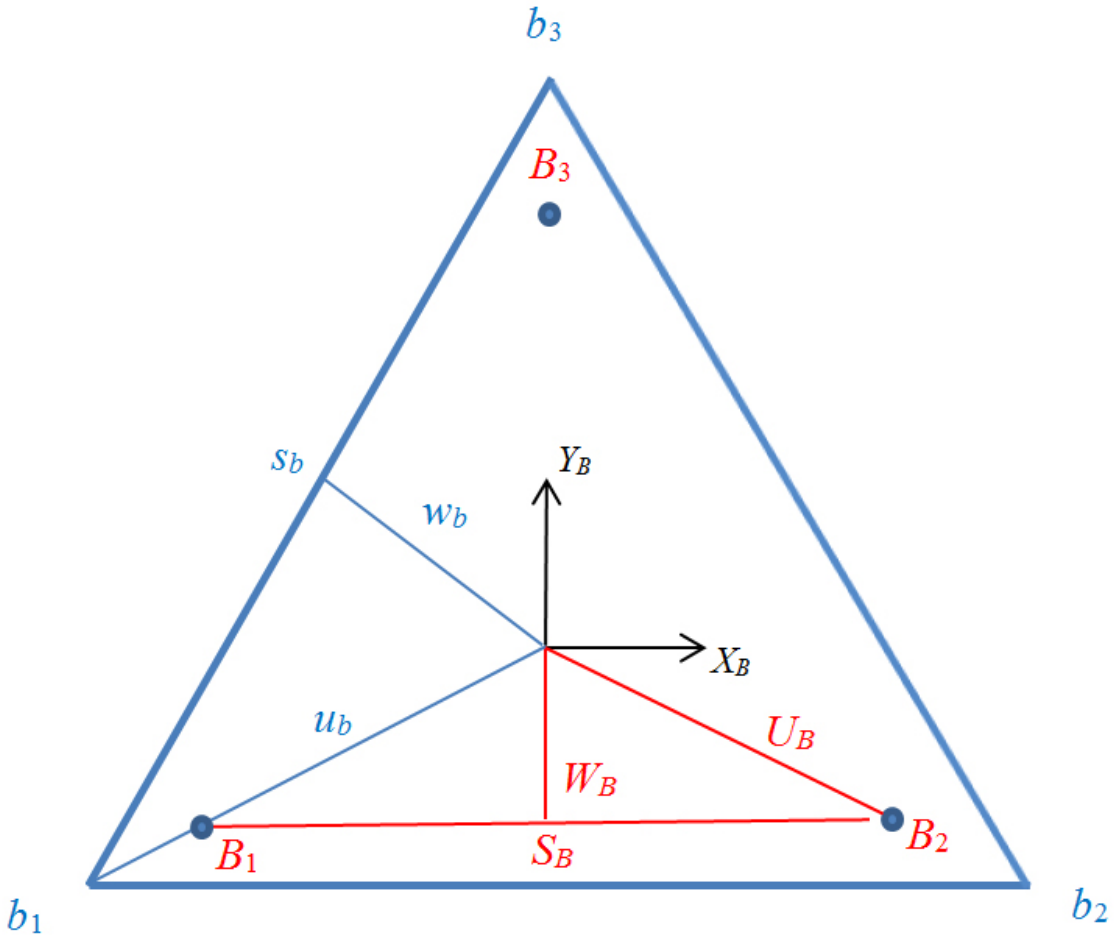
The Prismatic-Input Delta Robot is fundamentally similar to the original Revolute-Input Delta Robot. The major difference is that the three inputs now are driven by three linear-sliding prismatic joints instead of three revolute joints. This design change simplifies the kinematics equations and the IPK and FPK equations and solutions significantly, because the three prismatic inputs are aligned with the {*B*} frame *Z* axis, and there are no sines nor cosines required as in the revolute-input case.

As shown below, the 3-dof prismatic-input Delta Robot is composed of three identical **PUU** legs in parallel between the top fixed base and the bottom moving end-effector platform. The control variables are $L_i$, $i = 1,2,3$. In this model a positive change in $L_i$ is downward, in the $-Z_B$ direction. The three-dof Delta Robot is again capable of *XYZ* translational control of its moving platform within its workspace. The joint variables are $\mathbf{L} = \{L_1 \quad L_2 \quad L_3\}^T$, and the Cartesian variables are ${}^B\mathbf{P}_P = \{x \quad y \quad z\}^T$.

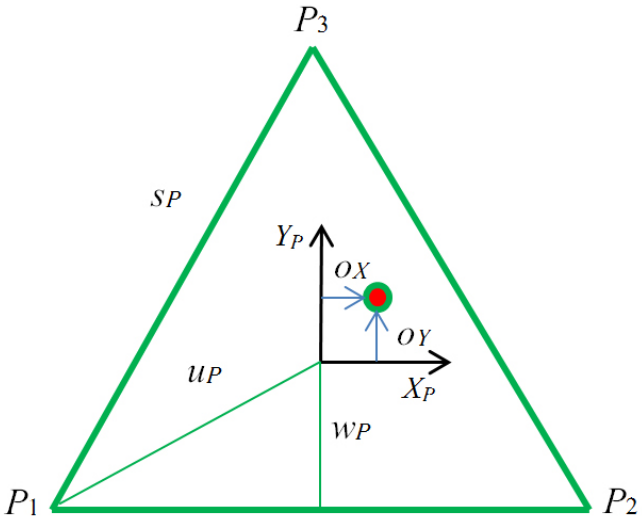



**Delta Robot Kinematic Diagram – Prismatic Inputs**

This robot is also known as the Linear Delta Robot or the Linear-Rail Delta Robot. The constant geometry (base points, platform vertices, etc.) presented earlier for the revolute-input Delta robot still apply to the prismatic-input Delta Robot. Also, the Mobility (dof) calculations for the prismatic-input case are identical to that presented for the revolute-input case, yielding *M* = 3. The Prismatic-Input Delta Robot fixed base and platform geometric details are shown on the next page.




**Prismatic-Input Delta Robot Fixed Base Details**




**Prismatic-Input Delta Robot Moving Platform Details**

The fixed-base prismatic joint points $B_i$ are constant in the base frame $\{B\}$ and the platform-fixed U-joint connection points $P_i$ are constant in the base frame $\{P\}$. Both lie on the vertices of equilateral triangles, of sides $S_B$ and $s_P$, respectively.

$$ {}^{P}\mathbf{B}_1 = \begin{Bmatrix} -\frac{S_B}{2} \\ -W_B \\ 0 \end{Bmatrix} \qquad {}^{P}\mathbf{B}_2 = \begin{Bmatrix} \frac{S_B}{2} \\ -W_B \\ 0 \end{Bmatrix} \qquad {}^{P}\mathbf{B}_3 = \begin{Bmatrix} 0 \\ U_B \\ 0 \end{Bmatrix} $$

$$ {}^{P}\mathbf{P}_1 = \begin{Bmatrix} -\frac{s_P}{2} \\ -w_P \\ 0 \end{Bmatrix} \qquad {}^{P}\mathbf{P}_2 = \begin{Bmatrix} \frac{s_P}{2} \\ -w_P \\ 0 \end{Bmatrix} \qquad {}^{P}\mathbf{P}_3 = \begin{Bmatrix} 0 \\ u_P \\ 0 \end{Bmatrix} $$

The vertices of the fixed-based equilateral triangle are not directly used in the kinematics equations; they are used in MATLAB graphics for the support frame:

$$ {}^{B}\mathbf{b}_1 = \begin{Bmatrix} -\frac{s_b}{2} \\ -w_b \\ 0 \end{Bmatrix} \qquad {}^{B}\mathbf{b}_2 = \begin{Bmatrix} \frac{s_b}{2} \\ -w_b \\ 0 \end{Bmatrix} \qquad {}^{B}\mathbf{b}_3 = \begin{Bmatrix} 0 \\ u_b \\ 0 \end{Bmatrix} $$

where:

$$ W_B = \frac{\sqrt{3}}{6} S_B \qquad U_B = \frac{\sqrt{3}}{3} S_B \qquad w_b = \frac{\sqrt{3}}{6} s_b \qquad u_b = \frac{\sqrt{3}}{3} s_b $$

$$ w_P = \frac{\sqrt{3}}{6} s_P \qquad u_P = \frac{\sqrt{3}}{3} s_P $$

## Prismatic-Input Delta Robot Parameters

| name | meaning | value (mm) |
|---|---|---|
| $s_b$ | base equilateral triangle side | 432 |
| $w_b$ | planar distance from {0} to near base side | 124.7 |
| $u_b$ | planar distance from {0} to a base vertex | 249.4 |
| $H$ | frame height | 686 |
| $S_B$ | P joints ($B_i$) equilateral triangle side | 246 |
| $W_B$ | same as $w_b$, for P joints equilateral triangle | 71 |
| $U_B$ | same as $u_b$, for P joints equilateral triangle | 142 |
| $s_P$ | platform equilateral triangle side | 127 |
| $w_P$ | planar distance from {$P$} to near platform side | 37 |
| $u_P$ | planar distance from {$P$} to a platform vertex | 73 |
| $o_X$ | nozzle $X$ offset | 10 |
| $o_Y$ | nozzle $Y$ offset | 30 |
| $L_{min}$ | $i$ = 1,2,3 minimum prismatic joints lengths | 67 |
| $L_{max}$ | $i$ = 1,2,3 maximum prismatic joints lengths | 479 |
| $l$ | lower legs parallelogram length | 264 |
| $h$ | lower legs parallelogram width | 44 |

The model values above are for a specific commercial prismatic-input delta robot 3D printer, the Delta Maker (deltamaker.com). Though Delta Robot symmetry is assumed, the following methods may be adapted to the general case for the prismatic-input delta robot.

Note that each individual prismatic length limits are $67 \le L_i \le 479$ mm; however, for all three prismatic lengths equal, $L_i = 358$ mm is the maximum extent to avoid motion through the 3D printing surface.

## Practical Prismatic-Input Delta Robots

forums.reprap.org




deltamaker.com

## Prismatic-Input Delta Robot Kinematics Analysis

From the kinematic diagram above, the following three vector-loop closure equations are written for the Delta Robot:

$$\left\{ {}^{B}\mathbf{B}_i \right\} + \left\{ {}^{B}\mathbf{L}_i \right\} + \left\{ {}^{B}\mathbf{l}_i \right\} = \left\{ {}^{B}\mathbf{P}_P \right\} + \left[ {}^{B}_{P}\mathbf{R} \right] \left\{ {}^{P}\mathbf{P}_i \right\} = \left\{ {}^{B}\mathbf{P}_P \right\} + \left\{ {}^{P}\mathbf{P}_i \right\} \qquad i = 1,2,3$$

The three applicable constraints state that the lower leg lengths must have the correct, constant length $l$ (the virtual length through the center of each parallelogram):

$$l_i = \left\| \left\{ {}^{B}\mathbf{l}_i \right\} \right\| = \left\| \left\{ {}^{B}\mathbf{P}_P \right\} + \left\{ {}^{P}\mathbf{P}_i \right\} - \left\{ {}^{B}\mathbf{B}_i \right\} - \left\{ {}^{B}\mathbf{L}_i \right\} \right\| \qquad i = 1,2,3$$

It will be more convenient to square both sides of the constraint equations above to avoid the square-root in the Euclidean norms:

$$l_i^2 = \left\| \left\{ {}^{B}\mathbf{l}_i \right\} \right\|^2 = l_{ix}^2 + l_{iy}^2 + l_{iz}^2 \qquad i = 1,2,3$$

Again, the Cartesian variables are ${}^{B}\mathbf{P}_P = \left\{ x \quad y \quad z \right\}^T$. The constant vector values for points $P_i$ and $B_i$ were given previously. The vectors $\left\{ {}^{B}\mathbf{L}_i \right\}$ are dependent on the joint variables $\mathbf{L} = \left\{ L_1 \quad L_2 \quad L_3 \right\}^T$; these formulas are much simpler than those for the rotational-input case presented earlier:

$${}^{B}\mathbf{L}_i = \begin{Bmatrix} 0 \\ 0 \\ -L_i \end{Bmatrix} \qquad i = 1,2,3$$

Substituting all above values into the vector-loop closure equations yields:

$$\left\{ {}^{B}\mathbf{l}_1 \right\} = \begin{Bmatrix} x + a \\ y + b \\ z + L_1 \end{Bmatrix} \qquad \left\{ {}^{B}\mathbf{l}_2 \right\} = \begin{Bmatrix} x - a \\ y + b \\ z + L_2 \end{Bmatrix} \qquad \left\{ {}^{B}\mathbf{l}_3 \right\} = \begin{Bmatrix} x \\ y + c \\ z + L_3 \end{Bmatrix}$$

where:
$$a = \frac{S_B}{2} - \frac{s_P}{2}$$
$$b = W_B - w_P$$
$$c = u_P - U_B$$

And the three constraint equations yield the kinematics equations for the prismatic-input Delta Robot:

$$x^2 + y^2 + z^2 + a^2 + b^2 + 2ax + 2by + 2zL_1 + L_1^2 - l^2 = 0$$
$$x^2 + y^2 + z^2 + a^2 + b^2 - 2ax + 2by + 2zL_2 + L_2^2 - l^2 = 0$$
$$x^2 + y^2 + z^2 + c^2 + 2cy + 2zL_3 + L_3^2 - l^2 = 0$$

The three absolute vector knee points are found using ${}^B\mathbf{A}_i = {}^B\mathbf{B}_i + {}^B\mathbf{L}_i$, $i = 1,2,3$:

$${}^B\mathbf{A}_1 = \begin{Bmatrix} -\frac{S_B}{2} \\ -W_B \\ -L_1 \end{Bmatrix} \qquad {}^B\mathbf{A}_2 = \begin{Bmatrix} \frac{S_B}{2} \\ -W_B \\ -L_2 \end{Bmatrix} \qquad {}^B\mathbf{A}_3 = \begin{Bmatrix} 0 \\ U_B \\ -L_3 \end{Bmatrix}$$

## Inverse Position Kinematics (IPK) Solution

The 3-dof prismatic-input Delta Robot inverse position kinematics (IPK) problem is stated: Given the Cartesian position of the moving platform control point (the origin of {*P*}), ${}^B\mathbf{P}_P = \{x \quad y \quad z\}^T$, calculate the three required actuated prismatic joint angles $\mathbf{L} = \{L_1 \quad L_2 \quad L_3\}^T$. The IPK solution for the prismatic-input Delta Robot is much simpler than that for the revolute-input Delta robot presented earlier and is easily found analytically. Referring to the prismatic-input Delta Robot kinematic diagram above, the IPK problem can be solved independently for each of the three **PUU** legs. Geometrically, each leg IPK solution is the intersection between a vertical line of unknown length $L_i$ (passing through base point ${}^B\mathbf{B}_i$) and a known sphere (radius *l*, centered on the moving platform vertex ${}^P\mathbf{P}_i$).

This solution may be done geometrically/trigonometrically. However, we will now accomplish this IPK solution analytically, using the three constraint equations independently (derived previously). The three independent scalar IPK equations are quadratic equations of the form:

$$L_i^2 + 2zL_i + C_i = 0 \qquad i = 1,2,3$$

where:

$$C_1 = x^2 + y^2 + z^2 + a^2 + b^2 + 2ax + 2by - l^2$$
$$C_2 = x^2 + y^2 + z^2 + a^2 + b^2 - 2ax + 2by - l^2$$
$$C_3 = x^2 + y^2 + z^2 + c^2 + 2cy - l^2$$

So we simply have three independent quadratic equations to solve for the prismatic-length inputs $L_i$, for each leg independently, where $A_i = 1$, $B_i = 2z$, and the $C_i$ are given above. The IPK solution simplifies quite nicely:

$$L_i = -z \pm \sqrt{z^2 - C_i} \qquad i = 1,2,3$$

Two $L_i$ solutions result from the $\pm$ in the quadratic formula. These solutions can be referred to as knee up and knee down for each leg. This yields two IPK branch solutions for each leg of the prismatic-input Delta Robot, for a total of 8 possible solutions. Generally the one overall solution with all knees up will be chosen.

When $z^2 < C_i$, the solution for $L_i$ is imaginary. This case should never occur in theory since the prismatic joint can extend as far as needed to maintain a real solution for each leg. However, in practice, there are of course prismatic joint limits. When $z^2 = C_i$, the two solution branches (knee up and knee down) have become the same solution.

The IPK input *xyz* is for the moving platform geometric center. When the desired control point is offset from the center (as in the case of many 3D printers), an initial transformation is required prior to implementing the IPK solution:

$$\left[{}^{B}_{P}T\right] = \left[{}^{B}_{N}T\right]\left[{}^{P}_{N}T^{-1}\right]$$

where $\left[{}^{P}_{N}T\right] = \begin{bmatrix} 1 & 0 & 0 & O_X \\ 0 & 1 & 0 & O_Y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and $\left[{}^{P}_{N}T^{-1}\right] = \begin{bmatrix} 1 & 0 & 0 & -O_X \\ 0 & 1 & 0 & -O_Y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

This is a very simple transformation since the prismatic-input Delta Robot allows only translational motion, with $\left[{}^{B}_{P}R\right] = \left[{}^{B}_{N}R\right] = \left[I_3\right]$. To save a lot of computations with 1s and 0s, simply subtract $O_X$ and $O_Y$ from the *x* and *y* components, respectively, of the given $(xyz)_N$ to obtain the IPK moving-platform-center input *xyz*. *N* stands for **nozzle** in a 3D printer. The *z* component is unchanged in this transformation.

## Forward Position Kinematics (FPK) Solution

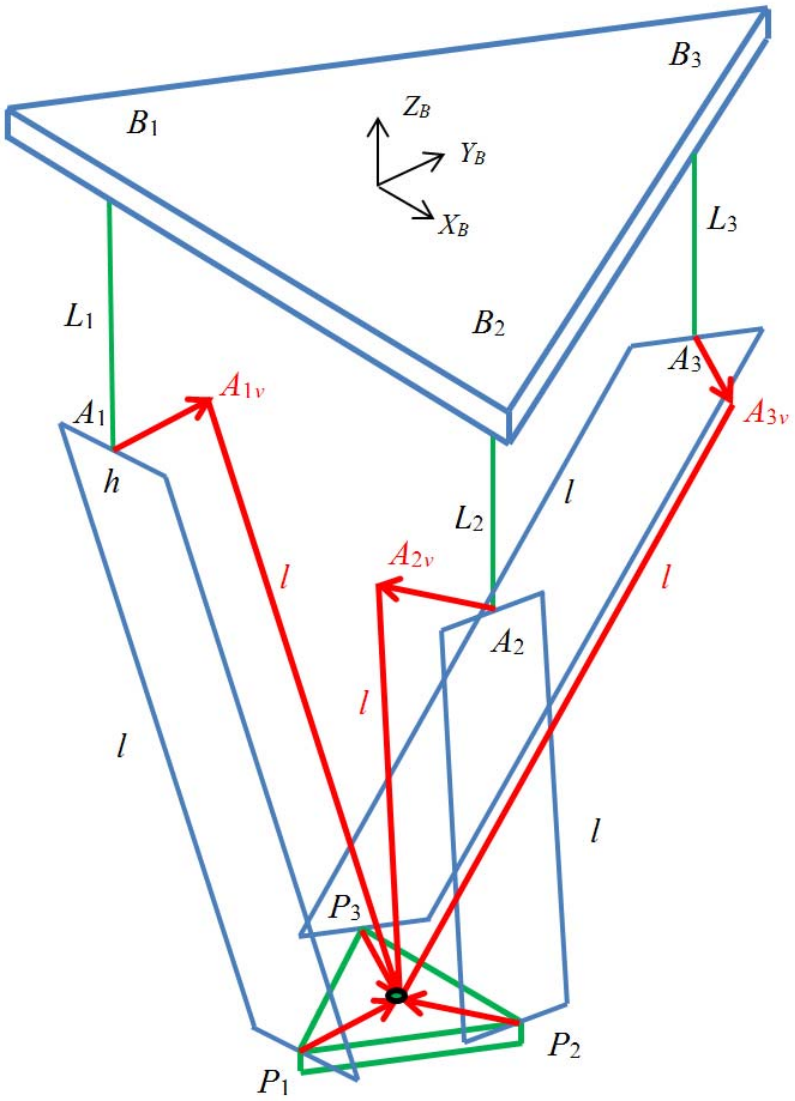
The 3-dof prismatic-input Delta Robot forward position kinematics (FPK) problem is stated: Given the three actuated joint angles $\mathbf{L}=\begin{Bmatrix} L_1 & L_2 & L_3 \end{Bmatrix}^T$, calculate the resulting Cartesian position of the moving platform control point (the origin of {*P*}), ${}^B\mathbf{P}_P=\begin{Bmatrix} x & y & z \end{Bmatrix}^T$. The FPK solution for parallel robots is generally very difficult. It requires the solution of multiple coupled nonlinear algebraic equations, from the three constraint equations applied to the vector loop-closure equations (derived previously). Multiple valid solutions generally result.

Thanks to the translation-only motion of the 3-dof Delta Robot, there is a straightforward analytical solution for which the correct solution set is easily chosen. Since $\mathbf{L}=\begin{Bmatrix} L_1 & L_2 & L_3 \end{Bmatrix}^T$ are given, we calculate the three absolute vector knee points using ${}^B\mathbf{A}_i = {}^B\mathbf{B}_i + {}^B\mathbf{L}_i$, $i=1,2,3$. Referring to the prismatic-input Delta Robot FPK diagram below, since we know that the moving platform orientation is constant, always horizontal with $\left[{}^B_P\mathbf{R}\right]=\left[\mathbf{I}_3\right]$, we define three virtual sphere centers ${}^B\mathbf{A}_{iv} = {}^B\mathbf{A}_i - {}^P\mathbf{P}_i$, $i=1,2,3$:

$${}^B\mathbf{A}_{1v}=\begin{Bmatrix} -\frac{S_B}{2}+\frac{s_P}{2} \\ -W_B+w_P \\ -L_1 \end{Bmatrix} \qquad {}^B\mathbf{A}_{2v}=\begin{Bmatrix} \frac{S_B}{2}-\frac{s_P}{2} \\ -W_B+w_P \\ -L_2 \end{Bmatrix} \qquad {}^B\mathbf{A}_{3v}=\begin{Bmatrix} 0 \\ U_B-u_P \\ -L_3 \end{Bmatrix}$$

and then the prismatic-input Delta Robot FPK solution is the intersection point of three known spheres. Let a sphere be referred as a vector center point {**c**} and scalar radius *r*, ({**c**}, *r*). Therefore, the FPK unknown point $\left\{{}^B\mathbf{P}_P\right\}$ is the intersection of the three known spheres:

$$(\left\{{}^B\mathbf{A}_{1v}\right\}, l) \qquad (\left\{{}^B\mathbf{A}_{2v}\right\}, l) \qquad (\left\{{}^B\mathbf{A}_{3v}\right\}, l).$$




**Delta Robot FPK Diagram – Prismatic Inputs**

Appendix A presents an analytical solution for the intersection point of the three given spheres, from Williams et al.[4] This solution also requires the solving of coupled transcendental equations. The appendix presents the equations and analytical solution methods, and then discusses imaginary solutions, singularities, and multiple solutions that can plague the algorithm, but all turn out to be no problem in this design.

In particular, with this existing three-spheres-intersection algorithm, if all three given sphere centers $\left\{ {}^{B}\mathbf{A}_{iv} \right\}$ have the same $Z$ height (a common case for the Delta Robot), there will be an algorithmic singularity preventing a successful solution (dividing by zero). One way to fix this problem is to simple rotate coordinates so all $\left\{ {}^{B}\mathbf{A}_{iv} \right\}$ $Z$ values are no longer the same, taking care to reverse this coordinate transformation after the solution is accomplished. However, we present another solution

[4] R.L. Williams II, J.S. Albus, and R.V. Bostelman, 2004, "3D Cable-Based Cartesian Metrology System", Journal of Robotic Systems, 21(5): 237-257.

(Appendix B) for the intersection of three spheres assuming that all three sphere Z heights are identical, to be used in place of the primary solution when necessary.

Another applicable problem to be addressed is that the intersection of three spheres yields two solutions in general (only one solution if the spheres meet tangentially, and zero solutions if the center distance is too great for the given sphere radii $l$ – in this latter case the solution is imaginary and the input data is not consistent with prismatic-input Delta Robot assembly). The spheres-intersection algorithm calculates both solution sets and it is possible to automatically make the computer choose the correct solution by ensuring it is below the base triangle rather than above it.

The FPK solution *xyz* is for the moving platform geometric center. When the desired control point is offset from the center (as in the case of many 3D printers), a further transformation is required after to implementing the FPK solution:

$$\left[{}^{B}_{N}T\right]=\left[{}^{B}_{P}T\right]\left[{}^{P}_{N}T\right]$$

where $\left[{}^{P}_{N}T\right]$ was given previously with the IPK solution. This is a very simple transformation since the prismatic-input Delta Robot allows only translational motion, with $\left[{}^{B}_{P}R\right]=\left[{}^{B}_{N}R\right]=\left[I_3\right]$. To save a lot of computations with 1s and 0s, simply add $O_X$ and $O_Y$ to the $x$ and $y$ components, respectively, of the FPK moving-platform-center solution *xyz*, to obtain the desired FPK solution $(xyz)_N$. The $z$ component is unchanged in this transformation.

## Alternate FPK Solution

This three-spheres-intersection approach to the FPK for the prismatic-input Delta Robot yields results identical to solving the three kinematics equations for ${}^{B}\mathbf{P}_{P} = \{x \quad y \quad z\}^{T}$ given $\mathbf{L} = \{L_1 \quad L_2 \quad L_3\}^{T}$. Since the three constraint equations for the prismatic-input Delta Robot are much simpler than those for the revolute-input Delta Robot, we now present an alternative FPK analytical solution. The three constraint equations are repeated below:

$$x^2 + y^2 + z^2 + a^2 + b^2 + 2ax + 2by + 2zL_1 + L_1^2 - l^2 = 0$$
$$x^2 + y^2 + z^2 + a^2 + b^2 - 2ax + 2by + 2zL_2 + L_2^2 - l^2 = 0$$
$$x^2 + y^2 + z^2 + c^2 + 2cy + 2zL_3 + L_3^2 - l^2 = 0$$

Subtracting the second equation from the first equation yields a linear equation, expressing *x* as a function of *z* only:

$$x = f(z) = dz + e$$

where:

$$d = \frac{L_2 - L_1}{2a} \qquad e = \frac{L_2^2 - L_1^2}{4a}$$

Further, subtracting the third equation from the first equation and substituting $x = f(z)$ from above yields another linear equation, expressing *y* as a function of *z* only:

$$y = g(z) = Dz + E$$

where:

$$D = \frac{L_3 - L_1 - ad}{b - c} \qquad E = \frac{c^2 - a^2 - b^2 - 2ae + L_3^2 - L_1^2}{2(b - c)}$$

Substituting both $x = f(z)$ and $y = g(z)$ into the third equation yields a single equation in one unknown *z*, a quadratic polynomial:

$$Az^2 + Bz + C = 0$$

where:

$$A = d^2 + D^2 + 1$$
$$B = 2(de + DE + cD + L_3)$$
$$C = e^2 + E^2 + c^2 + 2cE + L_3^2 - l^2$$

And so the alternate analytical FPK solution is:

$$z_{1,2} = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

$$x_{1,2} = f(z_{1,2}) = dz_{1,2} + e$$

$$y_{1,2} = g(z_{1,2}) = Dz_{1,2} + E$$

There are two possible solution sets $(x_1, y_1, z_1)$ and $(x_2, y_2, z_2)$, due to the $\pm$ in the quadratic formula. Generally only one solution will be used, the one where *xyz* is below the top-mounted base triangle.

## Prismatic-Input Delta Robot Velocity Kinematics Equations

The prismatic-input Delta Robot velocity kinematics equations come from the first time derivative of the three position constraint equations presented earlier:

$$\begin{aligned}(x+a)\dot{x}+(y+b)\dot{y}+(z+L_1)\dot{z} &= -(z+L_1)\dot{L}_1\\ (x-a)\dot{x}+(y+b)\dot{y}+(z+L_2)\dot{z} &= -(z+L_2)\dot{L}_2\\ x\dot{x}+(y+c)\dot{y}+(z+L_3)\dot{z} &= -(z+L_3)\dot{L}_3\end{aligned}$$

Written in matrix-vector form:

$$[\mathbf{A}]\{\dot{\mathbf{X}}\} = [\mathbf{B}]\{\dot{\mathbf{L}}\}$$

$$\begin{bmatrix} x+a & y+b & z+L_1\\ x-a & y+b & z+L_2\\ x & y+c & z+L_3\end{bmatrix}\begin{Bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{Bmatrix} = -\begin{bmatrix} z+L_1 & 0 & 0\\ 0 & z+L_2 & 0\\ 0 & 0 & z+L_3\end{bmatrix}\begin{Bmatrix}\dot{L}_1\\ \dot{L}_2\\ \dot{L}_3\end{Bmatrix}$$

The forward velocity solution is:

$$\{\dot{\mathbf{X}}\} = [\mathbf{A}]^{-1}[\mathbf{B}]\{\dot{\mathbf{L}}\}$$

And the inverse velocity solution is:

$$\{\dot{\mathbf{L}}\} = [\mathbf{B}]^{-1}[\mathbf{A}]\{\dot{\mathbf{X}}\}$$

$$\begin{Bmatrix}\dot{L}_1\\ \dot{L}_2\\ \dot{L}_3\end{Bmatrix} = -\begin{bmatrix} \dfrac{x+a}{z+L_1} & \dfrac{y+b}{z+L_1} & 1\\ \dfrac{x-a}{z+L_2} & \dfrac{y+b}{z+L_2} & 1\\ \dfrac{x}{z+L_3} & \dfrac{y+c}{z+L_3} & 1\end{bmatrix}\begin{Bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{Bmatrix}$$

## Prismatic-Input Delta Robot Position Kinematics Examples

For these examples, the prismatic-input Delta Robot dimensions are from the table given earlier for the Delta Maker 3D Printer, $s_b$ = 0.432, $S_B$ = 0.246, $s_P$ = 0.127, $l$ = 0.264, and $h$ = 0.044 (m).

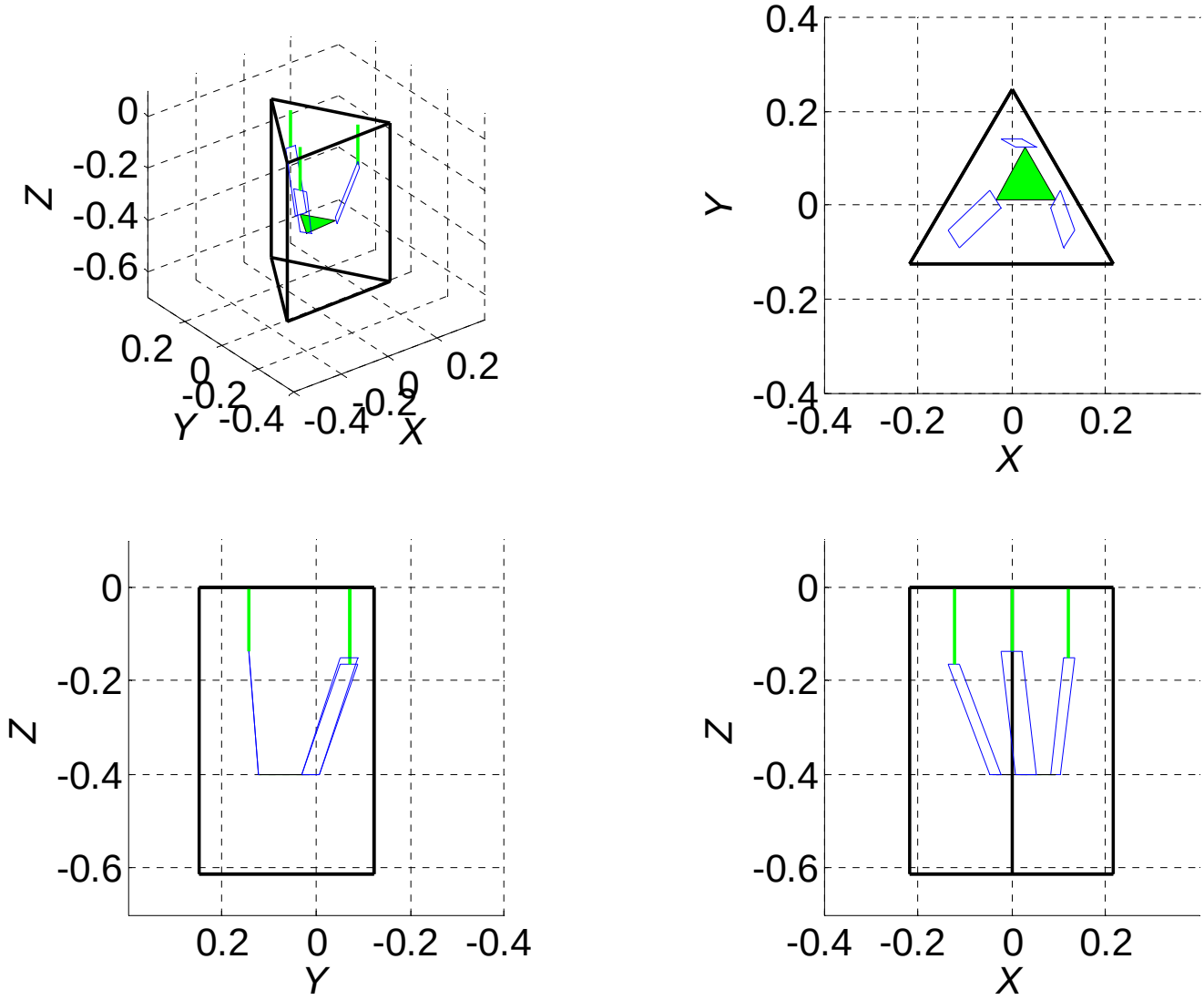
### Inverse Position Kinematics Examples

**Snapshot Examples** **Nominal Position.** Given $\left\{{}^B\mathbf{P}_P\right\} = \left\{0 \quad 0 \quad -0.5\right\}^T$ m, the calculated IPK results are (the preferred lower solution):

$$\mathbf{L} = \left\{0.2451 \quad 0.2451 \quad 0.2451\right\}^T \text{ m}$$

**General Position.** Given $\left\{{}^B\mathbf{P}_P\right\} = \left\{0.03 \quad 0.05 \quad -0.40\right\}^T$ m, the calculated IPK results are (the preferred kinked out solution):

$$\mathbf{L} = \left\{0.1664 \quad 0.1516 \quad 0.1384\right\}^T \text{ m}$$

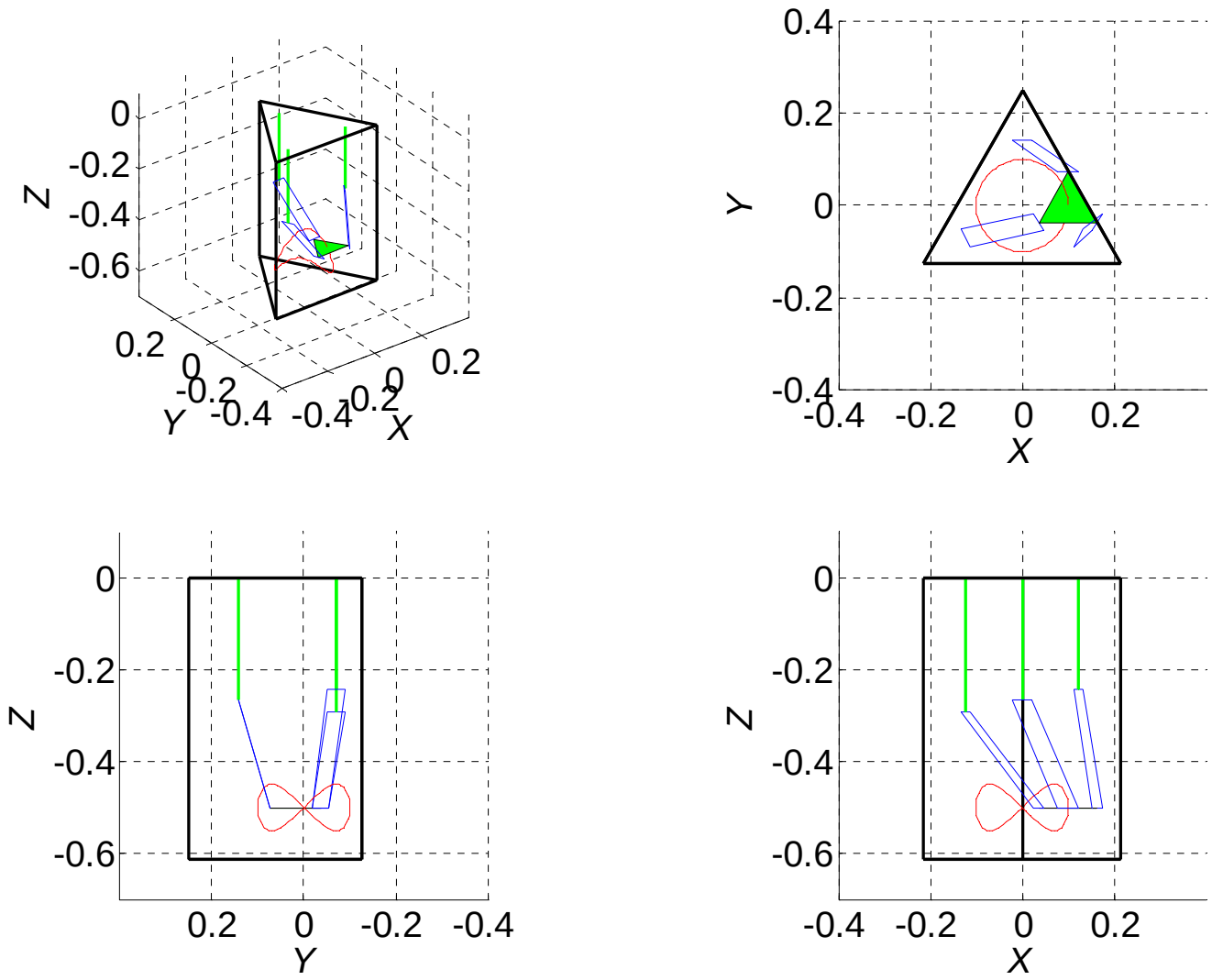



**General Inverse Position Snapshot Example**

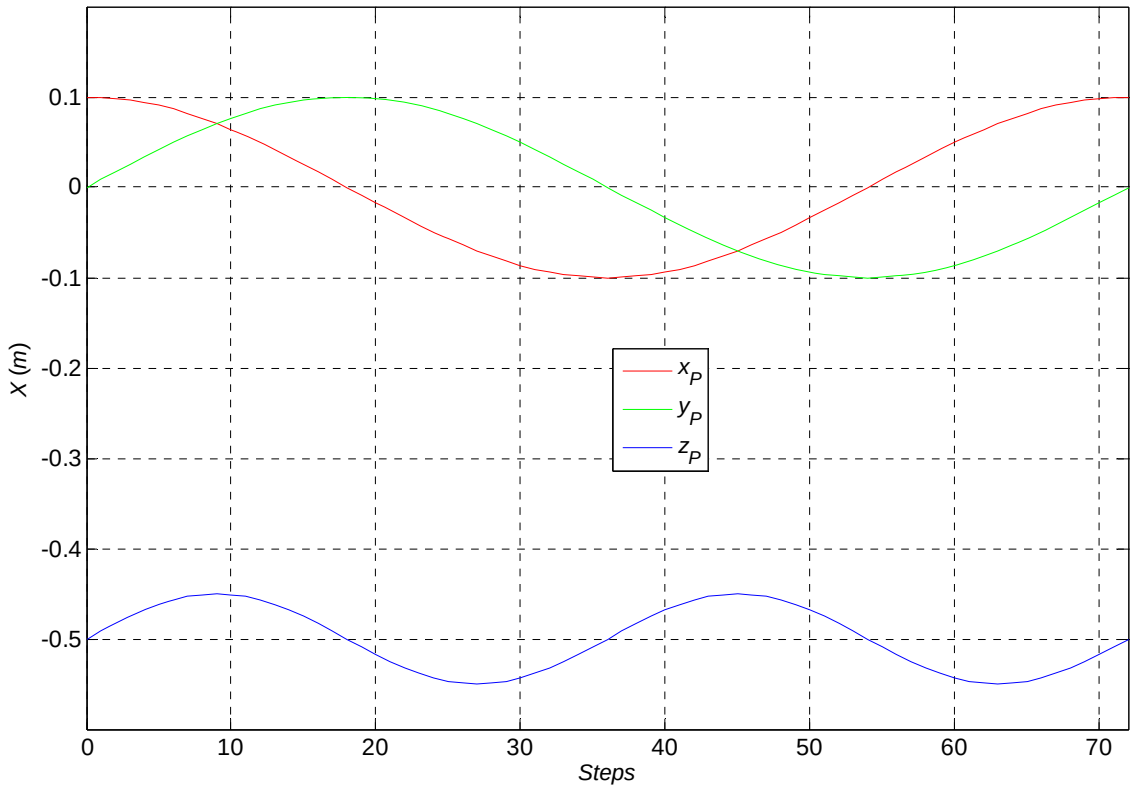
**IPK Trajectory Example**

The moving platform control point $\{P\}$ traces an $XY$ circle of center $\{0 \quad 0 \quad -0.5\}^T$ m and radius 0.1 m. At the same time, the $Z$ displacement goes through 2 complete sine wave motions centered on $Z = -1$ m with a 0.05 m amplitude.

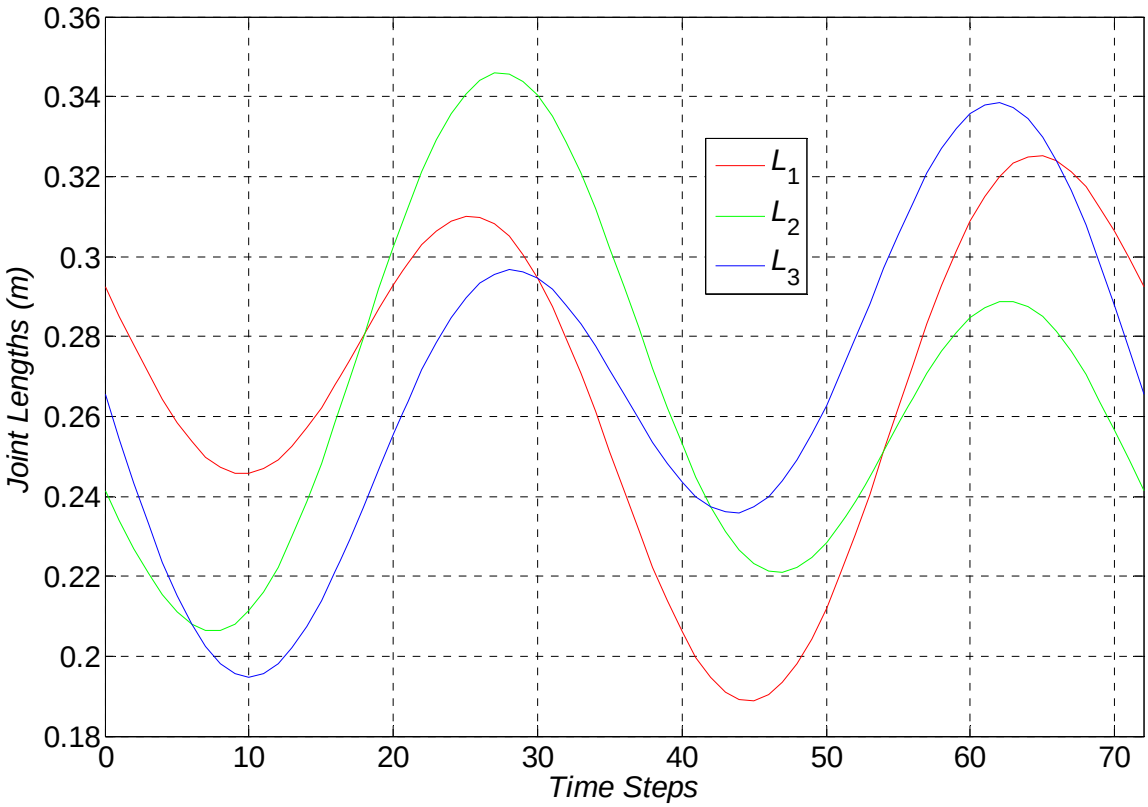
This IPK trajectory, at the end of motion, is pictured along with the simulated Delta Parallel Robot, in the MATLAB graphics below.




***XY* Circular Trajectory with *Z* sine wave**




**Commanded IPK Cartesian Positions**




**Calculated IPK Joint Angles**

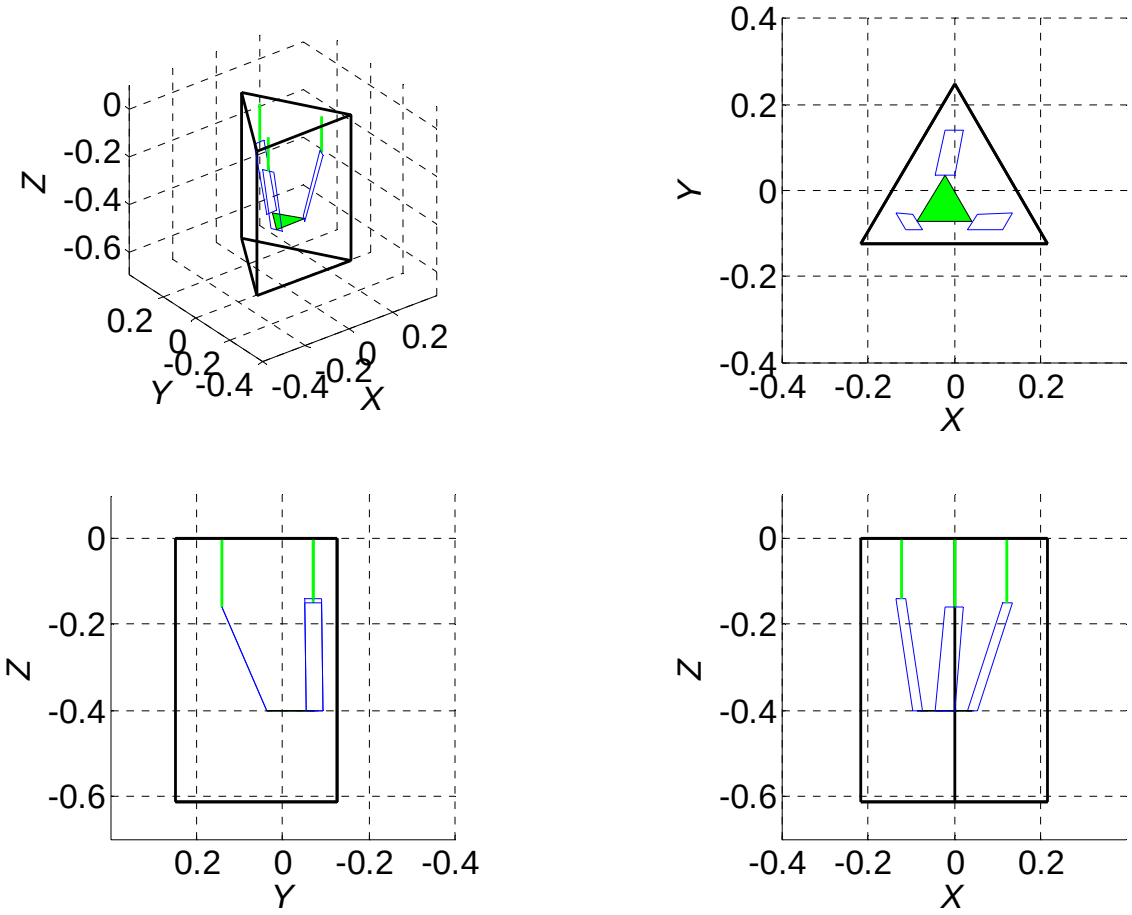
## Forward Position Kinematics Examples

**Snapshot Examples** **Nominal Position.** Given $\mathbf{L}=\{0.2 \quad 0.2 \quad 0.2\}^T$ m, the calculated FPK results are (the admissible lower solution, using the equal-*Z*-heights spheres intersection algorithm):

$$\left\{{}^B\mathbf{P}_P\right\}=\{0 \quad 0 \quad -0.4549\}^T \text{ m}$$

**General Position.** Given $\mathbf{L}=\{0.14 \quad 0.15 \quad 0.16\}^T$ m, the calculated FPK results are (the admissible lower solution, using the non-equal-*Z*-heights spheres intersection algorithm):

$$\left\{{}^B\mathbf{P}_P\right\}=\{-0.0215 \quad -0.0363 \quad -0.4012\}^T \text{ m}$$




**General Forward Position Snapshot Example**

**Circular Check Examples**

All snapshot examples were reversed to show that the circular check validation works; i.e.: When given $\left\{{}^B\mathbf{P}_P\right\}=\{0 \quad 0 \quad -0.4549\}^T$, the **IPK** solution calculated $\mathbf{L}=\{0.2 \quad 0.2 \quad 0.2\}^T$ and when given $\left\{{}^B\mathbf{P}_P\right\}=\{-0.0215 \quad -0.0363 \quad -0.4012\}^T$, the **IPK** solution calculated $\mathbf{L}=\{0.14 \quad 0.15 \quad 0.16\}^T$. When given $\mathbf{L}=\{0.2451 \quad 0.2451 \quad 0.2451\}^T$, the **FPK** solution calculated $\left\{{}^B\mathbf{P}_P\right\}=\{0 \quad 0 \quad -0.5\}^T$ and when given $\mathbf{L}=\{0.1664 \quad 0.1516 \quad 0.1384\}^T$, the **FPK** solution calculated $\left\{{}^B\mathbf{P}_P\right\}=\{0.03 \quad 0.05 \quad -0.40\}^T$.

## Acknowledgements

This work was initiated during the last week of the author's sabbatical from Ohio University at the University of Puerto Rico (UPR), Mayaguez, during Fall Semester 2014. The author gratefully acknowledges financial support from Ricky Valentin, UPR ME chair, and the Russ College of Engineering & Technology at Ohio University.

# Appendices

## Appendix A. Three-Spheres Intersection Algorithm

We now derive the equations and solution for the intersection point of three given spheres. This solution is required in the forward pose kinematics solution for many cable-suspended robots and other parallel robots. Let us assume that the three given spheres are ($\mathbf{c}_1$,$r_1$), ($\mathbf{c}_2$,$r_2$), and ($\mathbf{c}_3$,$r_3$). That is, center vectors $\mathbf{c}_1 = \{x_1 \quad y_1 \quad z_1\}^T$, $\mathbf{c}_2 = \{x_2 \quad y_2 \quad z_2\}^T$, $\mathbf{c}_3 = \{x_3 \quad y_3 \quad z_3\}^T$, and radii $r_1$, $r_2$, and $r_3$ are known (The three sphere center vectors must be expressed in the same frame, {0} in this appendix; the answer will be in the same coordinate frame). The equations of the three spheres are:

$$\begin{aligned} (x-x_1)^2 + (y-y_1)^2 + (z-z_1)^2 &= r_1^2 \\ (x-x_2)^2 + (y-y_2)^2 + (z-z_2)^2 &= r_2^2 \\ (x-x_3)^2 + (y-y_3)^2 + (z-z_3)^2 &= r_3^2 \end{aligned} \tag{A.1}$$

Equations (A.1) are three coupled nonlinear equations in the three unknowns *x*, *y*, and *z*. The solution will yield the intersection point $\mathbf{P} = \{x \quad y \quad z\}^T$. The solution approach is to expand equations (A.1) and combine them in ways so that we obtain $x = f(y)$ and $z = f(y)$; we then substitute these functions into one of the original sphere equations and obtain one quadratic equation in *y* only. This can be readily solved, yielding two *y* solutions. Then we again use $x = f(y)$ and $z = f(y)$ to determine the remaining unknowns *x* and *z*, one for each *y* solution. Let us now derive this solution.

First, expand equations (A.1) by squaring all left side terms. Then subtract the third from the first and the third from the second equations, yielding (notice this eliminates the squares of the unknowns):

$$a_{11}x + a_{12}y + a_{13}z = b_1 \tag{A.2}$$

$$a_{21}x + a_{22}y + a_{23}z = b_2 \tag{A.3}$$

where:

$$\begin{aligned} a_{11} &= 2(x_3 - x_1) & a_{21} &= 2(x_3 - x_2) \\ a_{12} &= 2(y_3 - y_1) & a_{22} &= 2(y_3 - y_2) \\ a_{13} &= 2(z_3 - z_1) & a_{23} &= 2(z_3 - z_2) \end{aligned}$$

$$\begin{aligned} b_1 &= r_1^2 - r_3^2 - x_1^2 - y_1^2 - z_1^2 + x_3^2 + y_3^2 + z_3^2 \\ b_2 &= r_2^2 - r_3^2 - x_2^2 - y_2^2 - z_2^2 + x_3^2 + y_3^2 + z_3^2 \end{aligned}$$

Solve for *z* in (A.2) and (A.3):

$$z = \frac{b_1}{a_{13}} - \frac{a_{11}}{a_{13}}x - \frac{a_{12}}{a_{13}}y \tag{A.4}$$

$$z = \frac{b_2}{a_{23}} - \frac{a_{21}}{a_{23}}x - \frac{a_{22}}{a_{23}}y \tag{A.5}$$

Subtract (A.4) from (A.5) to eliminate *z* and obtain $x = f(y)$:

$$x = f(y) = a_4 y + a_5 \tag{A.6}$$

where:

$$a_4 = -\frac{a_2}{a_1} \qquad a_5 = -\frac{a_3}{a_1} \qquad a_1 = \frac{a_{11}}{a_{13}} - \frac{a_{21}}{a_{23}} \qquad a_2 = \frac{a_{12}}{a_{13}} - \frac{a_{22}}{a_{23}} \qquad a_3 = \frac{b_2}{a_{23}} - \frac{b_1}{a_{13}}$$

Substitute (A.6) into (A.5) to eliminate *x* and obtain $z = f(y)$:

$$z = f(y) = a_6 y + a_7 \tag{A.7}$$

where:

$$a_6 = \frac{-a_{21}a_4 - a_{22}}{a_{23}} \qquad a_7 = \frac{b_2 - a_{21}a_5}{a_{23}}$$

Now substitute (A.6) and (A.7) into the first equation in (A.1) to eliminate *x* and *z* and obtain a single quadratic in *y* only:

$$ay^2 + by + c = 0 \tag{A.8}$$

where:

$$\begin{aligned} a &= a_4^2 + 1 + a_6^2 \\ b &= 2a_4(a_5 - x_1) - 2y_1 + 2a_6(a_7 - z_1) \\ c &= a_5(a_5 - 2x_1) + a_7(a_7 - 2z_1) + x_1^2 + y_1^2 + z_1^2 - r_1^2 \end{aligned}$$

There are two solutions for *y*:

$$y_\pm = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{A.9}$$

To complete the intersection of three spheres solution, substitute both *y* values $y_+$ and $y_-$ from (A.9) into (A.6) and (A.7):

$$x_\pm = a_4 y_\pm + a_5 \tag{A.10}$$

$$z_\pm = a_6 y_\pm + a_7 \tag{A.11}$$

In general there are two solutions, one corresponding to the positive and the second to the negative in (A.9). Obviously, the + and – solutions cannot be switched:

$$\{x_+ \quad y_+ \quad z_+\}^T \qquad \{x_- \quad y_- \quad z_-\}^T \tag{A.12}$$

## Example

Let us now present an example to demonstrate the solutions in the intersection of three spheres algorithm. Given three spheres (**c**,*r*):

$$\left(\begin{Bmatrix}0 & 0 & 0\end{Bmatrix}^T, \sqrt{2}\right) \qquad \left(\begin{Bmatrix}3 & 0 & 0\end{Bmatrix}^T, \sqrt{5}\right) \qquad \left(\begin{Bmatrix}1 & -3 & 1\end{Bmatrix}^T, 3\right) \tag{A.13}$$

The intersection of three spheres algorithm yields the following two valid solutions:

$$\begin{Bmatrix}x_+ & y_+ & z_+\end{Bmatrix}^T = \begin{Bmatrix}1 & 0 & 1\end{Bmatrix}^T \qquad \begin{Bmatrix}x_- & y_- & z_-\end{Bmatrix}^T = \begin{Bmatrix}1 & -0.6 & -0.8\end{Bmatrix}^T \tag{A.14}$$

These two solutions may be verified by a 3D sketch. This completes the intersection of three spheres algorithm. In the next subsections we present several important topics related to this three-spheres intersection algorithm: imaginary solutions, singularities, and multiple solutions.

## Imaginary Solutions

The three spheres intersection algorithm can yield imaginary solutions. This occurs when the radicand $b^2 - 4ac$ in (A.9) is less than zero; this yields imaginary solutions for $y_\pm$, which physically means not all three spheres intersect. If this occurs in the hardware, there is either a joint angle sensing error or a modeling error, since the hardware should assemble properly.

A special case occurs when the radicand $b^2 - 4ac$ in (A.9) is equal to zero. In this case, both solutions have degenerated to a single solution, i.e. two spheres meet tangentially in a single point, and the third sphere also passes through this point.

## Singularities

The three spheres intersection algorithm and hence the overall forward pose kinematics solution is subject to singularities. These are all algorithmic singularities, i.e. there is division by zero in the mathematics, but no problem exists in the hardware (no loss or gain in degrees of freedom). This subsection derives and analyzes the algorithmic singularities for the three spheres intersection algorithm presented above. Different possible three spheres intersection algorithms exist, by combining different equations starting with (A.1) and eliminating and solving for different variables first. Each has a different set of algorithmic singularities. We only analyze the algorithm presented above.

Inspecting the algorithm, represented in equations (A.1) – (A.12), we see there are four singularity conditions, all involving division by zero.

**Singularity Conditions**

$$\begin{aligned} a_{13} &= 0 \\ a_{23} &= 0 \\ a_1 &= 0 \\ a &= 0 \end{aligned} \tag{A.15}$$

The first two singularity conditions:

$$a_{13} = 2\left(z_3 - z_1\right) = 0 \tag{A.16}$$

$$a_{23} = 2\left(z_3 - z_2\right) = 0 \tag{A.17}$$

are satisfied when the centers of spheres 1 and 3 or spheres 2 and 3 have the same *z* coordinate, i.e. $z_1 = z_3$ or $z_2 = z_3$. Therefore, in the nominal case where all four virtual sphere centers have the same *z* height, this three-spheres intersection algorithm is always singular. An alternate solution is presented in Appendix B to overcome this problem.

The third singularity condition,

$$a_1 = \frac{a_{11}}{a_{13}} - \frac{a_{21}}{a_{23}} = 0 \tag{A.18}$$

Simplifies to:

$$\frac{x_3 - x_1}{z_3 - z_1} = \frac{x_3 - x_2}{z_3 - z_2} \tag{A.19}$$

For this condition to be satisfied, the centers of spheres 1, 2, and 3 must be collinear in the *XZ* plane. In general, singularity condition 3 lies along the edge of the useful workspace and thus presents no problem in hardware implementation if the system is properly designed regarding workspace limitations.

The fourth singularity condition,

$$a = a_4^2 + 1 + a_6^2 = 0 \tag{A.20}$$

Is satisfied when:

$$a_4^2 + a_6^2 = -1 \tag{A.21}$$

It is impossible to satisfy this condition as long as $a_4$ and $a_6$ from (A.6) and (A.7) are real numbers, as is the case in hardware implementations. Thus, the fourth singularity condition is never a problem.

## Multiple Solutions

In general the three spheres intersection algorithm yields two distinct, correct solutions ($\pm$ in (A.9 – A.11)). Generally only one of these is the correct valid solution, determined by the admissible Delta Robot assembly configurations.

## Appendix B. Simplified Three-Spheres Intersection Algorithm

We now derive the equations and solution for the intersection point of three given spheres, assuming all three spheres have identical vertical center heights. Assume that the three given spheres are $(\mathbf{c}_1, r_1)$, $(\mathbf{c}_2, r_2)$, and $(\mathbf{c}_3, r_3)$. That is, center vectors $\mathbf{c}_1 = \{x_1 \quad y_1 \quad z_1\}^T$, $\mathbf{c}_2 = \{x_2 \quad y_2 \quad z_2\}^T$, $\mathbf{c}_3 = \{x_3 \quad y_3 \quad z_3\}^T$, and radii $r_1$, $r_2$, and $r_3$ are known. The three sphere center vectors must be expressed in the same frame, {*B*} here, and the answer will be in the same coordinate frame. The equations of the three spheres to intersect are (choosing the first three spheres):

$$(x-x_1)^2 + (y-y_1)^2 + (z-z_n)^2 = r_1^2 \tag{B.1}$$
$$(x-x_2)^2 + (y-y_2)^2 + (z-z_n)^2 = r_2^2 \tag{B.2}$$
$$(x-x_3)^2 + (y-y_3)^2 + (z-z_n)^2 = r_3^2 \tag{B.3}$$

Since all *Z* sphere-center heights are the same, we have $z_1 = z_2 = z_3 = z_n$. The unknown three-spheres intersection point is $\mathbf{P} = \{x \quad y \quad z\}^T$. Expanding (1-3) yields:

$$x^2 - 2x_1x + x_1^2 + y^2 - 2y_1y + y_1^2 + z^2 - 2z_nz + z_n^2 = r_1^2 \tag{B.4}$$
$$x^2 - 2x_2x + x_2^2 + y^2 - 2y_2y + y_2^2 + z^2 - 2z_nz + z_n^2 = r_2^2 \tag{B.5}$$
$$x^2 - 2x_3x + x_3^2 + y^2 - 2y_3y + y_3^2 + z^2 - 2z_nz + z_n^2 = r_3^2 \tag{B.6}$$

Subtracting (6) from (4) and (6) from (5) yields:

$$2(x_3-x_1)x + 2(y_3-y_1)y + x_1^2 + y_1^2 - x_3^2 - y_3^2 = r_1^2 - r_3^2 \tag{B.7}$$
$$2(x_3-x_2)x + 2(y_3-y_2)y + x_2^2 + y_2^2 - x_3^2 - y_3^2 = r_2^2 - r_3^2 \tag{B.8}$$

All non-linear terms of the unknowns *x*, *y* cancelled out in the subtractions above. Also, all *z*-related terms cancelled out in the above subtractions since all sphere-center *z* heights are identical. Equations (7-8) are two linear equations in the two unknowns *x*, *y*, of the following form.

$$\begin{bmatrix} a & b \\ d & e \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{Bmatrix} c \\ f \end{Bmatrix} \tag{B.9}$$

where:

$$a = 2(x_3 - x_1)$$
$$b = 2(y_3 - y_1)$$
$$c = r_1^2 - r_3^2 - x_1^2 - y_1^2 + x_3^2 + y_3^2$$
$$d = 2(x_3 - x_2)$$
$$e = 2(y_3 - y_2)$$
$$f = r_2^2 - r_3^2 - x_2^2 - y_2^2 + x_3^2 + y_3^2$$

The unique solution for two of the unknowns *x*, *y* is:

$$x = \frac{ce - bf}{ae - bd}$$
$$y = \frac{af - cd}{ae - bd} \tag{B.10}$$

Returning to (1) to solve for the remaining unknown *z*:

$$Az^2 + Bz + C = 0 \tag{B.11}$$

where:

$$A = 1$$
$$B = -2z_n$$
$$C = z_n^2 - r_1^2 + (x - x_1)^2 + (y - y_1)^2$$

Knowing the unique values *x*, *y*, the two possible solutions for the unknown *z* are found from the quadratic formula:

$$z_{p,m} = \frac{-B \pm \sqrt{B^2 - 4C}}{2} \tag{B.12}$$

For the Delta Robot, ALWAYS choose the *z* height solution that is below the base triangle, i.e. negative *z*, since that is the only physically-admissible solution.

This simplified three-spheres intersection algorithm solution for *x*, *y*, *z* fails in two cases:

i) When the determinant of the coefficient matrix in the *x*, *y*, linear solution (B.10) is zero.

$$ae - bd = 2(x_3 - x_1)2(y_3 - y_2) - 2(y_3 - y_1)2(x_3 - x_2) = 0 \tag{B.13}$$

This is an algorithmic singularity whose condition can be simplified as follows. (B.13) becomes:

$$(x_3 - x_1)(y_3 - y_2) = (y_3 - y_1)(x_3 - x_2) \tag{B.14}$$

If (B.14) is satisfied there will be an algorithmic singularity. Note that the algorithmic singularity condition (B.14) is only a function of constant terms. Therefore, this singularity can be avoided by design, i.e. proper placement of the robot base locations in the *XY* plane. For a symmetric Delta Robot, this particular algorithmic singularity is avoided by design.

ii) When the radicand in (B.12) is negative, the solution for *z* will be imaginary. The condition $B^2 - 4C < 0$ yields:

$$(x - x_1)^2 + (y - y_1)^2 > r_1^2 \tag{B.15}$$

When this inequality is satisfied, the solution for *z* will be imaginary, which means that the robot will not assemble for that configuration. Note that (B.15) is an inequality for a circle. This singularity will NEVER occur if valid inputs are given for the FPK problem, i.e. the Delta Robot assembles.