

## Indexing for Basic Robotics Commands (HP PRIME)

1. **X-Rotation:** This command works to make the rotation matrix on the X-axis. It works with radians and variables.

**Indexation:** `rx(variable or radians)`

**Examples:**

$$rx\left(\frac{\pi}{2}\right) \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad rx(q1) \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(q1) & -\sin(q1) & 0 \\ 0 & \sin(q1) & \cos(q1) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2. **Y-Rotation:** This command works to make the rotation matrix on the Y-axis. It works with radians and variables.

**Indexation:** `ry(variable or radians)`

**Examples:**

$$ry(\pi) \quad \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ry(q2) \quad \begin{bmatrix} \cos(q2) & 0 & \sin(q2) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(q2) & 0 & \cos(q2) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. **Z-Rotation:** This command works to make the rotation matrix on the Y-axis. It works with radians and variables.

**Indexation:** `rz(variable or radians)`

**Examples:**

$$rz\left(\frac{\pi}{3}\right) \quad \begin{bmatrix} \frac{1}{2} & -\frac{1}{2}\sqrt{3} & 0 & 0 \\ \frac{1}{2}\sqrt{3} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad rz(q5) \quad \begin{bmatrix} \cos(q5) & -\sin(q5) & 0 & 0 \\ \sin(q5) & \cos(q5) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. **XYZ-Translation:** Produces a homogeneous transformation matrix for translation in 3D space.

**Indexation:** `tr(value in x, value in y, value in z)`

**Examples:**

$$tr(1,2,3) \quad \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad tr(l1,l2,l3) \quad \begin{bmatrix} 1 & 0 & 0 & l1 \\ 0 & 1 & 0 & l2 \\ 0 & 0 & 1 & l3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5. **Denavit–Hartenberg Transformation:** Applies the Denavit–Hartenberg (DH) parameters to construct a transformation matrix. Only works with radians and variables!.

**Indexation:** `dh(theta, d, a, alpha)`

**Examples:**

$$\text{dh}\left(\frac{\pi}{2}, 3, 0, \pi\right) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{dh}(q_1, l_1, l_2, q_2) = \begin{bmatrix} \cos(q_1) & -l_2 \cos(q_1) \\ \sin(q_1) & l_2 \sin(q_1) \\ 0 & l_1 \\ 0 & 1 \end{bmatrix}$$

6. **Inverse Homogeneous:** Computes the inverse of a homogeneous transformation matrix.

**Indexation:** `ih(Homogeneous Matrix)`

**Examples:**

$$\text{ih}\left(\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{ih}\left(\begin{bmatrix} \cos(q_1) & -\sin(q_1) & 0 & 0 \\ \sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}\right) = \begin{bmatrix} \cos(q_1) & \sin(q_1) & 0 & 0 \\ -\sin(q_1) & \cos(q_1) & 0 & 0 \\ 0 & 0 & 1 & -l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

7. **Inverse Denavit-Hartenberg:** Calculates the inverse of a matrix using Denavit-Hartenberg parameters. Joins the homogeneous inverse and denavit-Hartenberg commands.

**Indexation:** `idh(theta, d, a, alpha)`

**Examples:**

$$\text{idh}\left(\frac{\pi}{2}, 0, 1, 0\right) = \begin{bmatrix} 0 & 1 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{idh}(q_3, l_1, 0, 0) = \begin{bmatrix} \cos(q_3) & \sin(q_3) & 0 & 0 \\ -\sin(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & -l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

8. **Matrix Multiplication (Inverse Kinematics):** Calculate the result of multiplying 2 matrices, left matrix and base matrix of homogeneous transformation, to be able to match expressions in a system of vector equations. It works by entering the homogeneous matrix on the left and delivering the coordinate [i,j] to which it is to be equalized. Below is an example of use.

$$\begin{bmatrix} C_1 & S_1 & 0 & 0 \\ 0 & 0 & 1 & -l_1 \\ S_1 & -C_1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_2 & 0 & -S_2 & -S_2 q_3 \\ S_2 & 0 & C_2 & C_2 q_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad ({}^0\mathbf{A}_1)^{-1} {}^0\mathbf{T}_3 = {}^1\mathbf{A}_2 {}^2\mathbf{A}_3$$

**Indexation:** `mm(Left Matrix, i, j)`

**Example:**

$$\text{mm}\left(\begin{bmatrix} \cos(q_1) & \sin(q_1) & 0 & 0 \\ 0 & 0 & 1 & -l_1 \\ \sin(q_1) & -\cos(q_1) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, 3, 4\right) = \begin{bmatrix} p_x \sin(q_1) - p_y \cos(q_1) \\ \dots \end{bmatrix}$$

From this example, we get the following:

$$p_x \cdot \sin(q_1) - p_y \cdot \cos(q_1) = 0$$

9. **Matrix Multiplication Rotation (Inverse Kinematics):** It does the same as the previous command (Matrix Multiplication), but only for rotation matrices. Homogeneous transformation matrices can be used but will only deliver the results for the 3x3 rotation matrix.

**Indexation:** `rr(Left Matrix, i, j)`

**Example:**

$$rr \left( \begin{bmatrix} \sin(q_1) & 0 & 0 \\ 0 & 1 & -l_1 \\ -\cos(q_1) & 0 & 0 \end{bmatrix}, 3, 3 \right) \quad -ax * \cos(q_1)$$

10. **Rotation Matrix Extraction (4x4 to 3x3):** Extracts the rotation matrix from a 4x4 matrix.

**Indexation:** `er(Homogeneous Matrix)`

**Example:**

$$er \left( \begin{bmatrix} \cos(q_3) & \sin(q_3) & 0 & 0 \\ -\sin(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & -l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \quad \begin{bmatrix} \cos(q_3) & \sin(q_3) & 0 \\ -\sin(q_3) & \cos(q_3) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

11. **Position Vector Extraction (4x4 to 3x1):** Extracts the position vector from a 4x4 transformation matrix.

**Indexation:** `er(Homogeneous Matrix)`

**Example:**

$$ep \left( \begin{bmatrix} \cos(q_3) & \sin(q_3) & 0 & 0 \\ -\sin(q_3) & \cos(q_3) & 0 & 0 \\ 0 & 0 & 1 & -l_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \quad \begin{bmatrix} 0 \\ 0 \\ -l_1 \end{bmatrix}$$

12. **Analytical Jacobian Calculation:** Computes the analytical Jacobian matrix for a given set of variables (You need 6 functions to deliver a result. In general, it delivers more columns than necessary, so you must be able to differentiate according to the true variables how many columns to use from the result).

**Indexation:** `ja(px, py, pz, phi, theta, psi)`

**Example:**

$$ja(l_3 \cos(q_1), \sin(q_1 + q_2), l_1 - l_4 + q_3, q_1 + q_2 + q_4, \pi, \pi) \quad \begin{bmatrix} \text{diff}(l_3 \cos(q_1), q_1) & 0 & 0 & 0 \\ \cos(q_1 + q_2) & \cos(q_1 + q_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

13. **Analytical Jacobian Calculation (with Homogeneous Matrix):** It works the same as the previous command (12), but instead of entering the position functions, you just have to enter the homogeneous transformation matrix, and the Euler angles functions.

**Indexation:** `jat(Homogeneous Matrix, phi, theta, psi)`

**Example:**

$$jat \left( \begin{bmatrix} \cos(q_1) * \sin(q_3) + \cos(q_2) * \cos(q_3) * \sin(q_1) \\ \cos(q_3) * \sin(q_2) \\ 0 \\ l_4 * \sin(q_1) * \sin(q_2) - l_4 * \cos(q_1) * \cos(q_2) \\ -l_4 * \cos(q_1) * \sin(q_2) - l_4 * \cos(q_2) * \sin(q_1) \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \phi, \theta, \psi \right) \quad \begin{bmatrix} \dots \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- 14. Geometric Jacobian (Linear Velocity):** Calculates the linear velocity portion of the geometric Jacobian matrix. It works only up to 7 variables and you must take into account how many you are using to know how many columns to write.

**Indexation:** `jv(Homogeneous Matrix)`

**Example:**

$$jv \begin{bmatrix} (\cos(q_1) * \cos(q_2) - \sin(q_1) * \sin(q_2)) * \cos(q_4) + \\ (\cos(q_1) * \cos(q_2) - \sin(q_1) * \sin(q_2)) * \sin(q_4) + \\ -l_2 * \sin(q_1) - l_3 * \cos(q_1) * \sin(q_2) - l_3 * \cos(q_2) * \sin(q_1) \\ l_2 * \cos(q_1) + l_3 * \cos(q_1) * \cos(q_2) - l_3 * \sin(q_1) * \sin(q_2) \\ 0 \end{bmatrix}$$

- 15. Geometric Jacobian (Angular Velocity):** Computes the angular velocity portion of the geometric Jacobian matrix. It works the same as the previous command but the result must be interpreted according to the following:

$$\Omega = \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}$$

And remember that the program indicates that "Q" in capital letters is equivalent to the derivative.

$$Q_n = \dot{q}_n$$

**Indexation:** `jw(Homogeneous Matrix)`

**Example:**

$$jw \begin{bmatrix} (\cos(q_1) * \cos(q_2) - \sin(q_1) * \sin(q_2)) * \cos(q_4) + \\ (\cos(q_1) * \cos(q_2) - \sin(q_1) * \sin(q_2)) * \sin(q_4) + \\ 0 & -Q_1 - Q_2 - Q_4 & 0 \\ Q_1 + Q_2 + Q_4 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

From this example, we get the following:

$$w_x = 0$$

$$w_y = 0$$

$$w_z = \dot{q}_1 + \dot{q}_2 + \dot{q}_4$$

- 16. Euler Rotation to Rotation Matrix:** Allows you to obtain the rotation matrix for the different configurations of Euler rotations (variables only). Possible configurations are: vww, wuw, xyz.

**Indexation:** `etr(Euler Config)`

**Example:**

$$etr(vww) \begin{bmatrix} -\sin(\phi) * \sin(\psi) + \cos(\theta) * \cos(\phi) * \cos(\psi) & \dots & \cos(\phi) * \sin(\psi) + \cos(\theta) * \cos(\psi) * \sin(\phi) \\ \dots & \dots & \dots \\ -\cos(\psi) * \sin(\theta) & \dots & \dots \end{bmatrix}$$

- 17. Atan2:** This is a function commonly used in robotics, it works by delivering the values of X and Y, and follows the following relationships:

**Indexation:** `atg2(x, y)`

**Example:**

<code>atg2(5,-3)</code>	$-\text{ATAN}\left(\frac{3}{5}\right)$
<code>atg2(-2,1)</code>	$-\text{ATAN}\left(\frac{1}{2}\right) + \pi$

$$\text{atan2}(y, x) = \begin{cases} \arctan\left(\frac{y}{x}\right) & \text{if } x > 0 \\ \arctan\left(\frac{y}{x}\right) + \pi & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan\left(\frac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0 \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0 \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0 \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0 \end{cases}$$