Student Article

Revista elektron, Vol. 5, No. 1, pp. 77-82 (2021)
https://doi.org/10.37537/rev.elektron.5.1.125.2021

Control cinemático para un robot cuadrúpedo usando el Método de Newton-Raphson

Kinematic control for a quadruped robot using the Newton-Raphson method

Luis A. Orbegoso, Cristian A. Rodríguez, Edgar D. Valverde

Ingeniería Mecatrónica, Universidad Nacional de Trujillo
Trujillo, La Libertad, Perú
lorbegoso@unitru.edu.pe
carodriguez@unitru.edu.pe
dvalverde@unitru.edu.pe

Recibido: 14/02/21; Aceptado: 26/04/21

***Abstract*— The present work consisted of the implementation of a method for the solution of the inverse kinematic problem of a quadruped robot in four limbs with 3 degrees of freedom each. The proposed solution was supported by the application of an iterative method (Newton-Raphson) and special emphasis was placed on fine-tuning the algorithm parameters, that is, the number of iterations and the threshold distance, in order to guarantee the convergence of the solution in a few iterations. The work proposed to use the same method both for the function of walking by following the trajectory of a closed Bézier curve of five points, as well as for the control of body posture through the development of pitch and roll movements, and yaw from the base. The results of this solution were simulated in the PyBullet module to which the numerical values of the joints of the quadruped robot obtained from the inverse kinematics were exported to validate its behavior and movements in a simulated physical environment.**

Keywords: direct kinematics, inverse kinematics, Newton-Raphson method, quadruped robot.

***Resumen*— El presente trabajo consistió en la implementación de un método para la solución del problema cinemático inverso de un robot cuadrúpedo cuyas cuatro extremidades contaron con 3 grados de libertad cada una. La solución propuesta se apoyó en la aplicación de un método iterativo (Newton-Raphson) y se hizo especial énfasis en afinar los parámetros del algoritmo, es decir la cantidad de iteraciones y la distancia umbral, a fin de garantizar la convergencia de la solución en pocas iteraciones. El trabajo propuso utilizar el mismo método tanto para la función de caminar mediante el seguimiento de la trayectoria de una curva de Bézier cerrada de cinco puntos, así como para la del control de postura del cuerpo a través del desarrollo de los movimientos de pitch, roll y yaw de la base. Los resultados de esta solución fueron simulados en el módulo PyBullet al cual se exportaron los valores numéricos de las articulaciones del robot cuadrúpedo obtenidas a partir de la cinemática inversa para validar su comportamiento y movimientos en un entorno físico simulado.**

Palabras clave: cinemática directa, cinemática inversa, método de Newton-Raphson, robot cuadrúpedo.

I. Introducción

El problema de la cinemática inversa (CI) consiste en encontrar un vector de valores articulares que posicionan y orientan el extremo efector del robot en un lugar deseado o punto objetivo en su dominio de desempeño, lográndose de este modo el control de la postura. El desarrollo de este problema encuentra aplicaciones en diversas áreas tecnológicas como la robótica, los gráficos por ordenador y la ergonomía, entre otras aplicaciones.

Una de las primeras soluciones a este problema consiste en el método geométrico y el método analítico, donde los valores articulares para una posición deseada se hallan a partir de funciones trigonométricas inversas. Pero estos métodos se limitan a estructuras simples generalmente de no más de 3 grados de libertad (GDL) como lo indica Antonio Barrietos et al en [1].

A medida que se aumentan los grados de libertad como también la complejidad geométrica del cuerpo de un robot, se tiene que recurrir a otros métodos que permitan obtener alguna soluciones numéricas que satisfagan la condición de posición deseada. Una de las aproximaciones numéricas más populares consiste en hacer uso de la matriz jacobiana obtenida a partir de la cinemática directa, puesto que esta matriz puede linealizar el problema de la CI para un espacio relativamente reducido al extremo efector. Esta aproximación se mejora al hacer uso del método iterativo de Newton empleando la matriz jacobiana inversa, produciendo transiciones de posturas suaves. Sin embargo, las iteraciones requieren de cierto coste computacional, además de que la jacobiana inversa cuenta con singularidades que otros métodos tales como Jacobian Transpose y Damped Least Square (DLS) buscan solucionar al hacer uso de aproximaciones a la jacobiana inversa.

Otro método para hallar la CI es dado por Pechev en [2], donde asume este problema desde el enfoque del control, trasladando el extremo efector del sistema al estado deseado o punto objetivo a partir del punto inicial de dicho efector. Este método al no contar con los problemas de las singularidades de la matriz inversa, es computacionalmente más eficiente que los anteriores mencionados.

Del mismo modo, el método propuesto por Andreas Aristidou en [3] denominado FABRIK soluciona el problema de la CI evitando el uso de ángulos como matrices de rotación y en su lugar encuentra cada posición a partir de la ubicación de puntos de una recta. De este modo se produce una convergencia que requiere de pocas iteraciones y por ende cuenta con un bajo coste computacional, permitiendo el desarrollo de posturas realistas al final del algoritmo.

Uno de los trabajos relacionados al desarrollo de la CI para un robot cuadrúpedo es presentado por Muhammed Arif Sen et al en [4], donde a partir del desarrollo de la cinemática directa para cada una de las extremidades de 3 GDL del robot cuadrúpedo utilizando matrices de transformación homogéneas con los parámetros de Denavit-Hartenberg, encuentra la solución a la CI utilizando el método analítico para posteriormente controlar la orientación de la base en MATLAB.

Otro trabajo relacionado a la CI para un robot cuadrúpedo viene a ser presentado por Alain Segundo Pots et al en [5], donde además de desarrollar la CI para las extremidades del robot haciendo uso nuevamente del método analítico, analiza las singularidades de dichas soluciones para delimitar el área de desempeño de las extremidades relativo a la base.

Cabe resaltar que los robots caminantes, presentan una mayor ventaja, con respecto a los robots con ruedas, para desplazarse en terrenos irregulares. Por el contrario presentan un análisis cinemático más complejo, y un mayor número de parámetros a controlar, por lo tanto un mayor rango de movimientos complejos.

Uno de los trabajos más actualizado y completo para el control de un robot cuadrúpedo es presentado por Lee. J et al en [6] donde se utilizó Aprendizaje Reforzado (AR) en una simulación física del robot cuadrúpedo ANYmal para entrenar el modelo en diferentes escenarios virtuales sin que el robot tenga acceso a la información del entorno externo más que con el conocimiento de sus estados internos a través de sensores propioceptivos. De este modo, se llegó a obtener una Inteligencia Artificial capaz de controlar los movimientos de un robot cuadrúpedo según el entorno y su estado en este. Cuyos movimientos eran seguidos mediante la solución analítica de la CI propia de las extremidades del robot.

Ahora, el objetivo de este trabajo consiste en implementar el método de Newton-Raphson (MNR) explicado por Antonio Nieves y Federico Dominguez en [7], un método iterativo que emplea la inversa de la matriz jacobiana, para realizar el control cinemático de un robot cuadrúpedo con patas de 3 GDL, el cual requiere de la solución en tiempo real de la cinemática inversa para su movimiento y orientación en conjunto en un entorno de simulación física. Para una explicación más ilustrativa y resumida del presente manuscrito, consultar el video en [8].

II. Materiales y métodos

El modelo del robot cuadrúpedo con el cual se trabaja es de autoría propia y fue previamente diseñado en el programa Fusion 360.

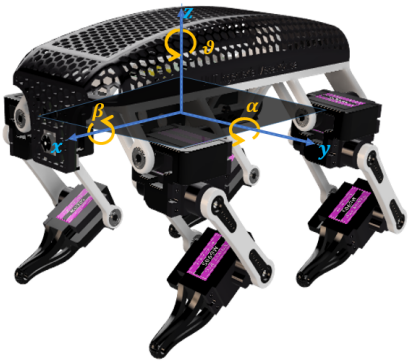

Fig. 1. Modelo CAD del robot cuadrúpedo.

La configuración de las patas de este modelo fueron del tipo mamífero y para su análisis respectivo se consideró el modelo por eslabones que se muestra en la Fig.2.

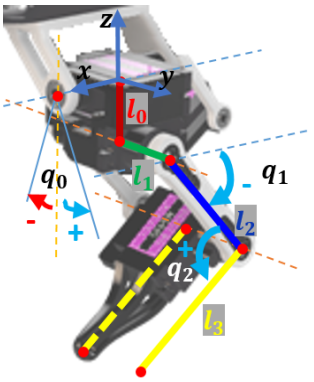

Fig. 2. Modelo simplificado de una pata del robot cuadrúpedo.

Los datos de las dimensiones de la base y las longitudes de los eslabones que conforman las extremidades del robot así como las variables a considerar para el análisis cinemático del robot cuadrúpedo presentado se muestran en la Tabla I.

TABLA I
TABLA DE DIMENSIONES Y VARIABLES DEL ROBOT CUADRÚPEDO

Dimensionamiento Físico	Largo del robot	15 cm
	Ancho del robot	13.5 cm
	Longitud l_0	1.2532 cm
	Longitud l_1	0.585 cm
	Longitud l_2	4.5 cm
	Longitud l_3	5 cm
Variables	Ángulo de inclinación (Pitch)	α
	Ángulo de balanceo (Roll)	β
	Ángulo de guiñada (Yaw)	ϑ
	Ángulo de articulación de giro lateral	q_0
	Ángulo de articulación l_2	q_1
	Ángulo entre l_2 y l_3	q_2

Los valores numéricos que comprenden a cada articulación fueron de $-\frac{\pi}{2} < q_0 < \frac{\pi}{2}$, $-\pi < q_1 < 0$ y $0 < q_2 < \pi$. Los ángulos se expresan en términos de conjunto como el vector **[q]**=[q_0, q_1, q_2,], cuyo valor inicial para todas las patas del robot cuadrúpedo es $\mathbf{[q_0]} = [0, -\frac{\pi}{6}, \frac{\pi}{3}]$. Del mismo modo, el dominio de valores que comprenden los ángulos referente a la orientación de la base fueron de $-\frac{\pi}{3} \leq \alpha \leq \frac{\pi}{3}$, $-\frac{\pi}{6} \leq \beta \leq \frac{\pi}{6}$ y $-\frac{\pi}{6} \leq \vartheta \leq \frac{\pi}{6}$.

La solución a la cinemática directa se obtuvo mediante la aplicación de cuaternios en lugar de a la matriz de transformación homogénea. Se partió del origen de coordenadas relativo a la base de cada extremidad y con un sistema de vectores base ortonormal que conformaban en un inicio a la matriz identidad. Luego, por cada rotación en las articulaciones se rotó a cada vector base haciendo uso del cuaternio asociado con dicha rotación, de este modo se obtenía un nuevo sistema base ortonormal de vectores obtenido a partir de la rotación del sistema base anterior respecto al eje de la articulación rotativa, mientras que por cada traslación se sumaba a la coordenada actual de la articulación la longitud del eslabón siguiente multiplicado por el vector unitario de la base correspondiente a la dirección de dicho eslabón.

Posteriormente, la solución numérica de la cinemática inversa para las patas del robot cuadrúpedo se desarrolló mediante el uso del MNR, tomando como función objetivo a la función de $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ obtenida como desarrollo de la cinemática directa para una pata del robot.

Así mismo, se utilizaron librerías Python tales como Numpy para trabajar con matrices y SciPy para manejar las rotaciones en el espacio. Las funciones resultantes de la cinemática directa y su jacobiana inversa fueron tratadas y calculadas usando el módulo SymPy para trabajar con expresiones simbólicas y posterior sustitución a valor numérico. El módulo de Tkinter se empleó para el desarrollo de la interfaz gráfica la cual a través de widgets permitió interactivamente manipular la trayectoria así como el movimiento de cada extremidad y la dirección de la base del robot, en esa misma interfaz, Matplotlib se empleó para graficar los resultados del MNR. Finalmente, el módulo de PyBullet sirvió para simular en un entorno físico dichos resultados del robot cuadrúpedo en Unified Robot Description Format (URDF).

III. Propuesta

Partiendo de la ecuación de la cinemática directa ***P**=F([q])* que relaciona la posición del extremo efector **P** en función de una serie de valores articulares **[q]**. Pero si el valor del extremo efector **P** ya está dado y se quiere conocer los valores angulares **[q]**, tales valores se pueden interpretar como las raíces que anulan a una función de la forma *G=F([q])-**P***; cuya solución aproximada planteada por el MNR se obtiene directamente a partir de *n* iteraciones partiendo desde un valor articular inicial y cercano a la solución: $\mathbf{[q_0]}$.

$$\mathbf{[q_{n+1}]} = \mathbf{[q_n]} - \left(\frac{dF([q])}{d([q])}\right)^{-1} \cdot (F(\mathbf{[q_n]}) - \mathbf{P}) \qquad (1)$$

Lo cual se cumple para el caso de la cinemática directa para una extremidad del robot cuadrúpedo, considerándola como una función de $\mathbb{R}^3 \rightarrow \mathbb{R}^3$ $F(q_0,q_1,q_2)=[X,Y,Z]$, donde sus componentes son expresadas por las relaciones articulares:

$$X=-l_2\cos(q_1)+l_3\cos(q_1+q_2)$$
$$Y=l_0\sin(q_0)-l_1\cos(q_0)-l_2\sin(q_0)\sin(q_1)+l_3\sin(q_0)\sin(q_1+q_2)$$
$$Z=-l_0\cos(q_0)-l_1\sin(q_0)+l_2\sin(q_1)\cos(q_0)-l_3\sin(q_1+q_2)\cos(q_0)$$

Ahora, la derivada de la solución a la cinemática directa se conoce como Matriz Jacobiana y su inversa viene a ser:

$$J^{-1}(\mathbf{[q]}) = \left(\frac{dF(q_0,q_1,q_2)}{d(q_0,q_1,q_2)}\right)^{-1}$$

Por lo que, sustituyendo a la jacobiana inversa en la Ecuación (1) y para una posición de la extremidad deseada ***P**=[x,y,z]*, se tiene la siguiente expresión iterativa:

$$\mathbf{[q_{n+1}]} = \mathbf{[q_n]} - J^{-1}(\mathbf{[q_n]}) \cdot (F(\mathbf{[q_n]}) - [x, y, z]) \qquad (2)$$

De este modo, se puede calcular para cada iteración, el error de la posición para el valor de los tres ángulos obtenidos $\mathbf{[q_{n+1}]}$ definido de la siguiente forma:

$$\text{Error} = |\ [x, y, z] - F(\mathbf{[q_{n+1}]})\ | \qquad (3)$$

En la implementación de la Ecuación (2) para resolver la cinemática inversa del robot es necesario trabajar con un número de iteraciones *n* y una distancia permisible definida como:

$$d_p \geq |\ [x, y, z] - F(\mathbf{[q_0]})\ | \qquad (4)$$

De modo que se garantice la convergencia de la solución numérica de la cinemática inversa con un error despreciable. Para el presente trabajo se eligieron los valores de 5 iteraciones y una distancia permisible de 0.6 cm.

Por otro lado, para la trayectoria de cada pata a considerar, se utilizó como camino cerrado una curva de Bézier (CB) cerrada con 5 puntos de control como se muestra en el lado izquierdo de la Fig 3, puesto que estos puntos permiten modificaciones en la trayectoria de forma dinámica y en tiempo real. En seguida, se tomaron 100 puntos de muestreo en esta curva para que sean utilizados como puntos objetivos durante el algoritmo y garanticen una distancia entre dichos puntos menor a 0.6 cm.

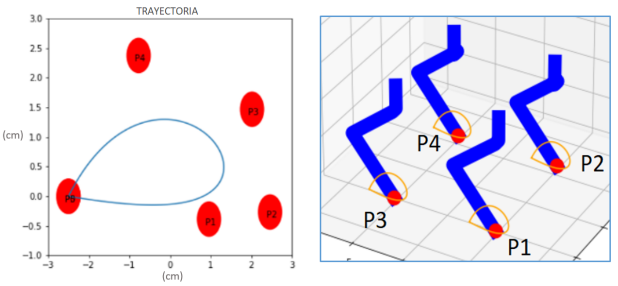

Fig. 3. Curva de Bézier y numeración de las patas del robot cuadrúpedo.

Mientras que para la coordinación de las patas al momento de andar, se consideró la enumeración que se

muestra en el lado derecho de la Fig.3, de modo que las patas P_1 y P_4 estén coordinadas debido a que siempre coinciden con el mismo el punto ubicado en sus respectivas curvas de Bézier al momento de caminar, lo mismo para las patas P_2 y P_3. Sin embargo, estas parejas de patas se diferencian unas de las otras en que al momento de andar se encuentran desfasadas 50 puntos en la curva de Bézier con respecto a la otra pareja, lo que permite que mientras una pareja de patas está alzada, la otra pareja esté en tierra y de este modo se desplace el robot.

Para desarrollar los movimientos de pitch, roll y yaw que controlan la orientación de la base del robot, se requiere que el extremo de las extremidades se mantengan fijas, por lo que se tomó en cuenta el análisis que muestra la Fig.4. para aplicar el MNR para este caso.

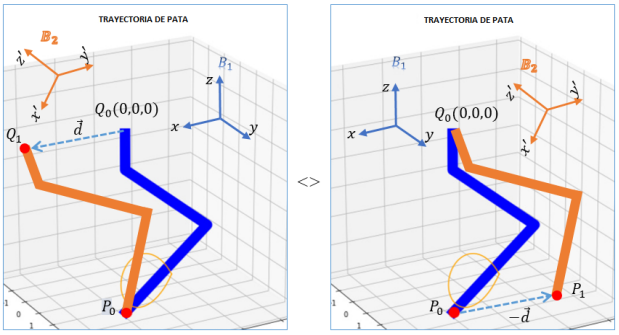

Fig. 4. Movimiento absoluto y relativo respecto a la base Q_0 de una pata ante una traslación e inclinación de la base del robot.

El desplazamiento e inclinación de la base del robot producen un desplazamiento **d** del punto base $\mathbf{Q_0}$ de la pata hacia el punto $\mathbf{Q_1}$ y una rotación alrededor del extremo de la pata en $\mathbf{P_0}$. Esto mismo pero desde la perspectiva de la base de la pata es equivalente a un desplazamiento en sentido contrario del extremo de la pata generando el punto objetivo $\mathbf{P_1}$ seguido de una rotación expresado por la nueva base $\boldsymbol{\beta_2}$, dicha base se forma a partir de los ángulos α, β y ϑ para los movimientos de pitch, roll y yaw respectivamente presentados en la Tabla I, de modo que el MNR se aplicará expresando las posiciones relativas al punto base Q_0 en la nueva base $\boldsymbol{\beta_2}$.

$$\mathbf{P_1} = \boldsymbol{\beta_2}^{-1} \cdot (\mathbf{P_0} - \mathbf{d}) \tag{5}$$

IV. Pruebas y resultados

En lo que corresponde a los resultados obtenidos en las simulaciones; por un lado, la simulación en el módulo de Matplotlib solo muestra el comportamiento ideal de los movimientos de las patas del robot cuadrúpedo resultante de la cinemática inversa con el MNR sin la interacción del cuerpo del robot con el entorno y la gravedad. Por otro lado, la simulación en el módulo de PyBullet añade estos factores faltantes haciendo uso del formato URDF del modelo robótico, por lo que tiene un comportamiento más realista a una implementación física del robot.

Para una mejor visualización de los resultados obtenidos en la simulación en PyBullet, se puede revisar el video en [9], donde se aprecia el control de orientación de la base del robot y el desplazamiento del mismo.

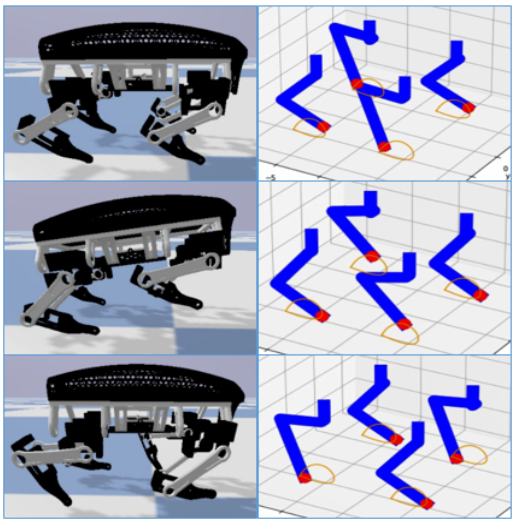

Fig. 5. Desplazamiento del robot cuadrúpedo.

En la Fig.5. se muestran los resultados de la simulación del desplazamiento para el robot cuadrúpedo. Se puede apreciar que el MNR permite hacer un seguimiento cíclico para cada extremidad de los 100 puntos de muestreo que se hicieron en la trayectoria planteada de la CB, permitiendo de este modo el desplazamiento del robot en el entorno de simulación física.

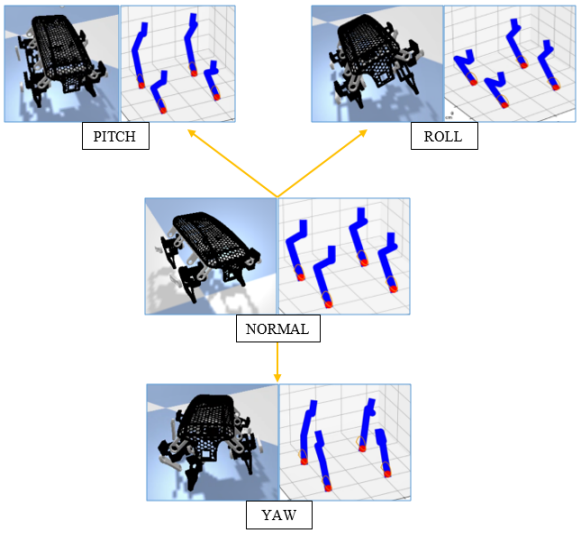

Fig. 6. Desarrollo de los movimientos de pitch, roll y yaw para el robot cuadrúpedo.

Por otro lado, como se observa en la Fig.6, respecto al control de la orientación de la base del robot cuadrúpedo, dado que los extremos de las extremidades se mantienen fijos, se partió de la Ecuación (5) para que este problema sea admisible para el MNR. De esta forma, los movimientos del pitch, roll y yaw son realizables como se observa el cuerpo del cuadrúpedo simulado en el entorno de PyBullet a la izquierda de cada toma, que al igual que con los resultados el modelo matemático graficados en la derecha de la toma, se ve que solo la base del robot es la que se mueve y orienta manteniéndose los extremos de las patas fijas al suelo.

Ahora, la gráfica presentada en la Fig.7 muestra la evaluación del error en el MNR dado *n* iteraciones respecto a la distancia inicial, la cual no supera los 5 cm, entre el extremo de la pata y un punto objetivo aleatorio. Se utilizó

la Ecuación (3), siendo el punto objetivo un vector aleatorio para una distancia dada. Para la elaboración de esta gráfica, en todo momento inicial la extremidad se encontraba con sus valores articulares iniciales de $\mathbf{[q_0]} = [0, -\frac{\pi}{6}, \frac{\pi}{3}]$, estos valores hacen que la distancia desde la base de la extremidad hasta su extremo efector sea por defecto de 6 cm, y siendo aproximadamente 11 cm (longitud de extremidad estirada) la distancia máxima que el extremo efecto puede alcanzar respecto a su base, de modo que los valores usados para la separación inicial a una distancia de 0.5 cm a 5 cm garantizan que no se supere la distancia máxima anteriormente mencionada.

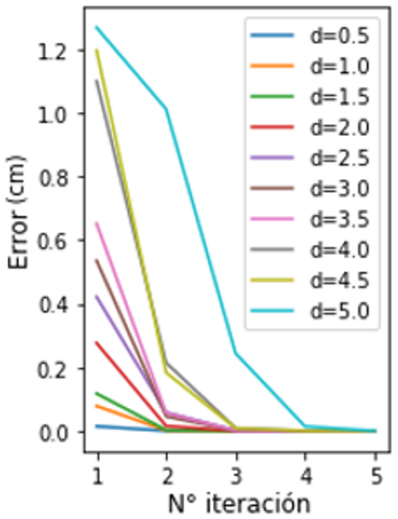

Fig. 7. Error de la posición en la n-ésima iteración para distintas distancias iniciales de partida.

Como se observa en la Fig.7, el módulo de error en cada caso de las distancias consideradas tiene una convergencia en la 5ta iteración, donde el valor para cualquiera de los casos tiende a 0. Sin embargo, para distancias relativamente pequeñas tales como dentro del rango comprendido entre 0.5 cm y 3.5 cm, el error llega a ser casi nulo a partir de la 3ra iteración; pero la 5ta iteración resulta ser más general en asegurar la convergencia para el intervalo de distancias de 0.5 cm a 5 cm.

Una forma de corroborar lo anteriormente mencionado, es mediante la evaluación de la probabilidad para 100 casos en que el error final resultante del algoritmo después de 5 iteraciones es menor a un error del orden de 10^{-b} cm, tomando *b* valores enteros de 10 a 16, respecto al valor para una distancia permisible dada en la Ecuación (4); lo cual se muestra en la Tabla II.

TABLA II
PROBABILIDAD DE ERROR MENOR AL UMBRAL DADO EN RELACIÓN A LA DISTANCIA PERMISIBLE EN LA 5° ITERACIÓN

DISTANCIA PERMISIBLE (CM)	ERROR (CM)						
	E-10	*E-11*	*E-12*	*E-13*	*E-14*	*E-15*	*E-16*
0.6	1	1	1	1	1	0.76	0.05
1.7	1	1	1	1	1	066	0.01
2.8	0.99	0.93	0.89	0.9	0.81	0.42	0.01
3.9	0.68	0.43	0.31	0.16	0.1	0.01	0
5	0.13	0.04	0.02	0	0	0	0

De este modo, se llega a que trabajando el MNR con 5 iteraciones y una distancia umbral permisible de 0.6 cm según los resultados de la Tabla II garantiza una probabilidad de 0.76 y 0.05 en tener un error menor a 10^{-15} cm y 10^{-16} cm respectivamente. Por lo que, las distancias que superen a este valor serían divididas en valores menores o iguales a este.

Es lógico suponer que cuanto más pequeña sea la distancia permisible una mayor posibilidad de convergencia en la 5ta iteración para obtener errores menores a 10^{-15} cm y 10^{-16} cm. Sin embargo para la construcción de la Tabla II se decidió partir con el valor de una distancia permisible de 0.6 cm para evaluar si a partir de aquel valor hasta la distancia de 5 cm las probabilidades de convergencia para errores relativamente bajos en otras distancias llegaba a ser tan altas como las probabilidades obtenidas de la distancia permisible de partida. Sin embargo, como se puede visualizar en la tabla para las distancias permisibles de 0.6 cm y 1.7 cm, las probabilidades de convergencia coinciden en ser 100% hasta para obtener un error menor que 10^{-14} cm; a partir de ahí la distancia permisible de partida mantiene probabilidades superiores a las de 1.7 cm para errores mucho más bajos.

Ahora, se hizo una comparación del tiempo de ejecución, error final y número de iteraciones que conlleva usar el MNR dividiendo una distancia superior a 0.6 cm en comparación contra el umbral de la distancia permisible seleccionada y otra sin dicha división. Tales resultados se muestran en la Tabla III.

TABLA III
COMPARACIÓN DEL TIEMPO DE EJECUCIÓN, ERROR FINAL Y NÚMERO DE ITERACIONES EMPLEADAS PARA ALCANZAR PUNTOS OBJETIVOS LEJANOS CON Y SIN DIVISIÓN DE DISTANCIAS

DISTANCIA (CM)	PARÁMETROS	SIN DIVISIÓN	CON DIVISIÓN
3	TIEMPO (S)	0.006979942	0.035902023
	ERROR (CM)	2.23152E-15	1.84444E-15
	ITERACIONES	5	25
3.89	TIEMPO (S)	0.004986048	0.033908844
	ERROR (CM)	1.09143E-10	1.9984E-15
	ITERACIONES	5	32
4.33	TIEMPO (S)	0.004985809	0.038897514
	ERROR (CM)	9.86773E-09	4.57757E-16
	ITERACIONES	5	36
5	TIEMPO (S)	0.005982876	0.055850029
	ERROR (CM)	0.026794166	6.28037E-16
	ITERACIONES	5	41

Como se observa en la Tabla III, en un principio la comparación del error obtenido al operar el MNR con una distancia dividida y otra sin división no es muy significativo, ocasionando un coste computacional

innecesario para distancias inferiores o iguales a los 3 cm. Sin embargo, a medida que la distancia se incrementa se puede apreciar que el error final obtenido en una distancia con división previa es significativamente menor al error obtenido en una sin previa división. Pues como se verifica en la Tabla II, para una distancia de 5 cm es imposible que se alcance un error menor a 10^{-15} cm, hecho que se supera al aplicar el MNR para cada segmento de 0.6 cm en una distancia de 5 cm hasta llegar al punto objetivo, como se muestra en la Tabla III.

Sin embargo, vale resaltar que a pesar de que el número de iteraciones y la distancia permisible seleccionadas permite obtener resultados de los valores angulares con errores del orden de 10^{-15} cm y 10^{-16} cm, en la práctica estos valores de mucha precisión serían inútiles debido a la propia resolución que los actuadores reales pueden tener. Abriéndose la posibilidad de elegir otra distancia permisible o número de iteraciones que generen valores angulares solo con las cifras significativas correctas admisibles a la resolución de los actuadores.

V. Conclusion

Después de analizar y discutir los resultados obtenidos en el presente trabajo se llegaron a las siguientes conclusiones:

- El empleo del MNR permite resolver el problema cinemático inverso para el modelo del robot cuadrúpedo en tiempo real. Por ende, se admite para el desarrollo del control cinemático tanto para el desplazamiento como para la orientación de la base del robot cuadrúpedo.
- A partir de la quinta iteración en el MNR se generan errores de posición los cuales se pueden considerar despreciables.
- Utilizar como parámetro una distancia máxima de 0.6 cm a partir del cual dividir distancias más largas entre el extremo de la pata y el punto objetivo, disminuye significativamente el error de posición final, pero aumenta las iteraciones y el tiempo de ejecución.

Finalmente, el MNR permite resolver satisfactoriamente la cinemática inversa para el modelo del robot cuadrúpedo planteado, permitiendo su incorporación para realizar funciones como el desplazamiento y la orientación de la base. El método puede llegar a presentar errores del orden de 10^{-15} cm y 10^{-16} cm, pero que en la implementación se tendría ajustar el error según la resolución de los actuadores para evitar iteraciones innecesarias en el MNR.

Referencias

[1] A. Barrientos, F. Peñín, C. Balaguer y R. Aracil, “Fundamentos de robótica”, McGraw-Hill, 2da edición, pp. 134–143, 1997.

[2] Alexandre N. Pechev, Inverse kinematics without matrix inversion, in: Proc. of the 2008 IEEE International Conf. on Robotics and Automation, Pasadena, CA, USA, May 19–23 2008, pp. 2005–2012

[3] Andreas Aristidou, Joan Lasenby. (15 mayo 2011). Forward And Backward Reaching Inverse Kinematics (FABRIK). ELSEVIER, pp. 1-2.

[4] Şen, Muhammed Arif & Bakırcıoğlu, Veli & Kalyoncu, Mete. (2017). Inverse Kinematic Analysis Of A Quadruped Robot. International Journal of Scientific & Technology Research. 6.

[5] Potts, Alain & Jaime, José. (2011). A Kinematical and Dynamical Analysis of a Quadruped Robot. 10.5772/25500.

[6] Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., & Hutter, M. (2020). Learning quadrupedal locomotion over challenging terrain. Science Robotics, 5(47), eabc5986. https://doi.org/10.1126/scirobotics.abc5986

[7] Antonio Nieves Hurtado, Federico C. Dominguez, “Métodos numéricos aplicados a la ingeniería”, 1ra edicion,pp. 302 - 311, 2014.

[8] Orbegoso, L. [LUIS ANTONIO ORBEGOSO MORENO]. (2020, noviembre 24). Control cinemático para un robot cuadrúpedo usando el Método de Newton-Raphson [Archivo de video]. Recuperado de https://youtu.be/PqOzn9ZfPPQ

[9] Orbegoso, L. [LUIS ANTONIO ORBEGOSO MORENO]. (2020, noviembre 24). Simulación en PyBullet de un robot cuadrúpedo [Archivo de video]. Recuperado de https://youtu.be/_OSwQZyZjEo