XXV CONGRESO NACIONAL DE INGENIERÍA MECÁNICA
Santander, 10-13 junio 2025
Universidad de Cantabria
IFToMM

Reducción del coste computacional asociado a la redundancia en la representación de la orientación empleando subespacio nulo en problemas de dinámica de sólidos

Igor Fernández de Bustos[1], Álvaro Noriega[2], Haritz Uriarte[3], Vanessa García-Marina[1]

[1] *Departamento Ingeniería mecánica, Universidad del País Vasco, impfedei@ehu.es, vanessa.garcia@ehu.es*

[2] *Departamento de Construcción e Ingeniería de Fabricación, Universidad de Oviedo, noriegaalvaro@uniovi.es*

[3] *Departamento Expresión gráfica y proyectos, Universidad del País Vasco, haritz.uriarte@ehu.es*

En este trabajo se presenta un estudio sobre la posibilidad de reducir el coste computacional asociado al incremento de variables que se produce cuando se resuelven problemas de dinámica de sólidos empleando sistemas redundantes de parametrización de la matriz de rotación, tal y como sucede cuando se emplean cuaterniones. Para ello se hace uso del subespacio nulo de la matriz Jacobiana de las restricciones asociadas a la parametrización (en el caso de los cuaterniones, la norma unidad). Teniendo en cuenta que estas restricciones afectan exclusivamente a los términos asociados al sólido cuya orientación es representada por el conjunto de parámetros, las dimensiones de este subespacio nulo son muy reducidas. Esto lleva a la posibilidad de aplicar este subespacio nulo a cada submatriz correspondiente a las ecuaciones de equilibrio y restricciones, reduciendo las dimensiones globales del problema. En el caso de problemas de tamaño pequeño en los que se pueden emplear matrices densas la reducción de coste está asociada a la reducción de las dimensiones del sistema a tratar, mientras que en el caso de sistemas de gran tamaño, esta reducción no solo debería afectar a las dimensiones, sino también a la reducción del trabajo organizativo (factorización simbólica y otros) necesario para trabajar de forma eficiente con matrices dispersas.

El cálculo del subespacio nulo habitualmente se considera que tiene un coste que muchas veces hace este tipo de operaciones poco eficaces. Sin embargo, el bajo tamaño de las matrices involucradas así como el empleo de un procedimiento de bajo coste para la obtención del subespacio nulo hace que este método lleve a reducciones notables del coste computacional, habiéndose constatado en las pruebas preliminares llevadas a cabo reducciones del 15% en el caso de cuaterniones y hasta el 80% cuando se emplea la matriz de rotación completa.

En el documento se desarrolla el planteamiento del problema, los algoritmos empleados en el trabajo, varios ejemplos en los que se muestra la eficiencia del algoritmo, y una serie de conclusiones y planteamientos sobre trabajos futuros.

1. Introducción

La representación de la orientación de un sólido en el espacio (o parametrización de la matriz de rotación) es un problema bastante relevante en la cinemática y dinámica de multicuerpos. El uso de ángulos de Euler, parámetros de Euler y parámetros de Rodrigues conduce a problemas de subdeterminación en la proximidad de algunas orientaciones [1]. El uso de otras parametrizaciones del vector de rotación suele conducir a expresiones complejas para la obtención de expresiones relevantes como la matriz de rotación o las velocidades angulares, que, por otro lado, también conducen a problemas de representaciones no únicas de la orientación. Una forma habitual de evitar estos problemas es el uso del cuaternión unitario, que carece de los problemas antes mencionados, o el uso de los elementos de la matriz de rotación como variables para definir la orientación. El inconveniente de estos sistemas es que introducen redundancia, incrementando así la cantidad de variables a tratar. En el caso de la dinámica de multicuerpos, una alternativa es el uso de conceptos del Álgebra de Lie para resolver el problema. El uso del Álgebra de Lie generalmente implica la integración de la velocidad angular, que carece de los problemas que aparecen en el nivel de posición, y luego actualizar incrementalmente la matriz de rotación [2]. Desafortunadamente, estas técnicas son bastante difíciles de aplicar a otros problemas, como el análisis de singularidad, donde es necesario recurrir a otros métodos [3,4].

La aplicación del espacio nulo en la dinámica y optimización de cuerpos múltiples no es un tema nuevo. Se han desarrollado muchos algoritmos como alternativa para resolver varios problemas que surgen en ambas áreas [5–7]. Sin embargo, el uso de estas técnicas se considera demasiado costoso, debido al coste que supone obtener el espacio nulo [8]. En este trabajo recurriremos al algoritmo expuesto en [9] para reducir el costo computacional general siempre que se utilicen sistemas de coordenadas redundantes para resolver problemas multicuerpo.

El documento está organizado de la siguiente manera. En primer lugar, se revisan algunas observaciones comunes y conocidas sobre los sistemas de orientación redundantes. Después, se expone brevemente el algoritmo de espacio nulo utilizado en los desarrollos actuales. A continuación, se presentan las ideas generales sobre cómo aplicar este algoritmo para reducir el costo computacional. Después, estas ideas se aplican a un caso particular de un integrador dinámico multicuerpo basado en diferencias centrales [10,11].A continuación se presentan algunos ejemplos numéricos donde se muestran las ventajas del método y finalmente se extraen algunas conclusiones.

2. Un algoritmo eficiente para obtener el subespacio nulo

La base del algoritmo utilizado aquí fue presentada en [9]. El algoritmo se basa en el uso de una factorización LDU, que es bastante común en la bibliografía. La diferencia clave radica en la expresión utilizada para obtener el espacio nulo.

Una factorización LDU general descompone una matriz en la forma:

$$\boldsymbol{A} = \boldsymbol{P}_f^T \boldsymbol{L}\,\boldsymbol{D}\,\boldsymbol{U}\,\boldsymbol{P}_c \tag{1}$$

Siendo:

$\boldsymbol{A}$: una matriz general (no necesariamente cuadrada o regular). Suponemos dimensiones nxm.

$\boldsymbol{P}_f$: una matriz de permutación de filas (ortogonal, dimensiones nxn)

$\boldsymbol{L}$: una matriz trapezoidal inferior, de dimensiones nxr, siendo r el rango de $\boldsymbol{A}$.

$\boldsymbol{D}$: una matriz diagonal (rxr), regular

$\boldsymbol{U}$: una matriz trapezoidal superior, de dimensiones rxm

$\boldsymbol{P}_c$: una matriz de permutación de columnas (ortogonal, dimensiones mxm)

$\boldsymbol{L}$ tiene la estructura:

$$\boldsymbol{L} = \begin{pmatrix} \boldsymbol{L}_{11} \\ \boldsymbol{L}_{21} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ l_{r1} & l_{r2} & l_{r3} & \dots & 1 \\ l_{(r+1)1} & l_{(r+1)2} & l_{(r+1)3} & \dots & l_{(r+1)r} \\ \dots & \dots & \dots & \dots & \dots \\ l_{m1} & l_{m2} & l_{m3} & \dots & l_{mr} \end{pmatrix} \tag{2}$$

De manera que $\boldsymbol{L_{11}}$ es una matriz triangular inferior, mientras que $\boldsymbol{U}$ tiene la estructura:

$$\boldsymbol{U}=\begin{pmatrix}\boldsymbol{U}_{11} & \boldsymbol{U}_{12}\end{pmatrix}=\begin{pmatrix}1 & u_{12} & u_{13} & \dots & u_{1r} & u_{1(r+1)} & \dots & u_{1n}\\ 0 & 1 & u_{23} & \dots & u_{2r} & u_{2(r+1)} & \dots & u_{2n}\\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots\\ 0 & 0 & 0 & \dots & 1 & u_{r(r+1)} & \dots & u_{rn}\end{pmatrix} \tag{3}$$

con lo que $\boldsymbol{U_{11}}$ es una matriz triangular superior. En estas condiciones, $\boldsymbol{D}$ tiene la forma:

$$\boldsymbol{D}=\begin{pmatrix}d_1 & 0 & 0 & \dots & 0\\ 0 & d_2 & 0 & \dots & 0\\ 0 & 0 & d_3 & \dots & 0\\ \dots & \dots & \dots & \dots & \dots\\ 0 & 0 & 0 & \dots & d_r\end{pmatrix} \tag{4}$$

Nótese que en [9] se presenta una interpretación ligeramente diferente del algoritmo, siendo ambas equivalentes. La forma habitual de obtener el espacio nulo de la matriz mediante eliminación de Gauss o cualquier tipo de factorización triangular es:

$$\bar{\boldsymbol{A}}=\boldsymbol{P}_f\boldsymbol{A}\boldsymbol{P}_c^T=\begin{pmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{pmatrix} \tag{5}$$

siendo $\boldsymbol{A_{11}}$ de dimensiones rxr. Es fácil de demostrar que $\boldsymbol{A_{11}}$ es regular. En estas condiciones, si uno calcula $\bar{\boldsymbol{N}}$ según la ecuación (6), $\bar{\boldsymbol{N}}$ es una base del subespacio nulo de $\bar{\boldsymbol{A}}$, y, por lo tanto, $\boldsymbol{N_A}$ obtenido según (7), es una base del subespacio nulo de $\boldsymbol{A}$.

$$\bar{\boldsymbol{N}}=\begin{pmatrix}-\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\\ \boldsymbol{I}\end{pmatrix} \tag{6}$$

$$\boldsymbol{N_A}=\boldsymbol{P}_c\bar{\boldsymbol{N}}=\boldsymbol{P}_c\begin{pmatrix}-\boldsymbol{A}_{11}^{-1}\boldsymbol{A}_{12}\\ \boldsymbol{I}\end{pmatrix} \tag{7}$$

Esta es considerada comúnmente como una de las formas más rápidas de calcular el espacio nulo de una matriz general, pero la alternativa presentada en [9] tiene algunas ventajas. Primero veamos el algoritmo. En lugar de la expresión (6), se propone (8), de modo que (9) es también base del espacio nulo de $\boldsymbol{A}$:

$$\bar{\boldsymbol{N}}=\begin{pmatrix}-\boldsymbol{U}_{11}^{-1}\boldsymbol{U}_{12}\\ \boldsymbol{I}\end{pmatrix} \tag{8}$$

$$\boldsymbol{N_A}=\boldsymbol{P}_c\bar{\boldsymbol{N}}=\boldsymbol{P}_c\begin{pmatrix}-\boldsymbol{U}_{11}^{-1}\boldsymbol{U}_{12}\\ \boldsymbol{I}\end{pmatrix} \tag{9}$$

En [9] se presentó un enfoque similar para calcular el espacio nulo izquierdo de la matriz utilizando $\boldsymbol{L}$. Las ventajas de este enfoque son varias. En primer lugar, el espacio nulo obtenido carece de escala. Esto es: si hubiera un problema relevante con la escala de las filas/columnas en $\boldsymbol{A}$, se elimina tan pronto como se haya utilizado un esquema de pivotaje adecuado. Esto se debe al hecho de que, siempre que el pivote haya sido elegido como el elemento más grande (en valor absoluto) en la fila, todos los elementos en $\boldsymbol{U_{11}}$ y $\boldsymbol{U_{12}}$ son menores o iguales a 1, y el elemento de la diagonal principal de $\boldsymbol{U_{11}}$ es uno. En segundo lugar, el costo computacional del cálculo de $\boldsymbol{U}_{11}^{-1}\boldsymbol{U}_{12}$ es bastante pequeño, debido al hecho de que ya calculamos $\boldsymbol{U_{11}}$ en la factorización y es una matriz triangular.

3. Utilizando el subespacio nulo para reducir el coste de la resolución de la cinemática y la dinámica en parametrizaciones redundantes de la matriz de rotación.

El uso de una parametrización redundante de la matriz de rotación aumenta la cantidad de variables a resolver y, por lo tanto, también aumenta el coste computacional. También requiere que se introduzca un conjunto de ecuaciones derivadas de la redundancia. Por ejemplo, si se utiliza el cuaternión unitario, es necesario introducir, para un sólido i dado, la restricción:

$$o_{i0}^2+o_{i1}^2+o_{i2}^2+o_{i3}^2=1 \tag{10}$$

siendo $\boldsymbol{o}_i=\begin{pmatrix} o_{i0} & o_{i1} & o_{i2} & o_{03} \end{pmatrix}^T$ el cuaternión unitario que define la orientación del sólido i. Esta ecuación debe verificarse para cada uno de los sólidos, por lo que se tienen tantas ecuaciones agregadas en el sistema como variables redundantes. Definamos todas esas ecuaciones en la forma:

$$\boldsymbol{\phi}_r(\boldsymbol{q})=\boldsymbol{0} \tag{11}$$

siendo $\boldsymbol{q}$ el conjunto de coordenadas que definen tanto la posición como la orientación de todos los sólidos. Así, sin pérdida de generalidad, se puede escribir (12), siendo ns la cantidad de sólidos en el sistema, y siendo cada uno de los $\boldsymbol{o}_i$ un subconjunto de $\boldsymbol{q}$.

$$\boldsymbol{q}=\begin{pmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \dots \\ \boldsymbol{q}_{ns} \end{pmatrix} \tag{12}$$

Así, por ejemplo, si se quiere resolver un problema de dinámica de sólidos, se debería resolver el sistema compuesto por las ecuaciones (13), (14) y (11)

$$\boldsymbol{M}\ddot{\boldsymbol{q}}=\boldsymbol{f}(\boldsymbol{q},\dot{\boldsymbol{q}},t)+\boldsymbol{G}^T\boldsymbol{\lambda} \tag{13}$$

$$\boldsymbol{\phi}(\boldsymbol{q})=\boldsymbol{0} \tag{14}$$

Siendo (14) el sistema de ecuaciones de restricción impuestas por los pares del sistema.

Un enfoque directo sería aplicar un integrador paso a paso, y en cada paso linealizar el conjunto de ecuaciones obtenido y resolverlas (probablemente de forma iterativa). Esto lleva a la resolución de un sistema de ecuaciones lineales de tamaño $ns \cdot nc$ siendo nc igual al número de coordenadas de cada sólido. Por ejemplo, utilizando ecuaciones cartesianas y cuaterniones, $nc=7$. En problemas pequeños, donde es interesante utilizar matrices densas, se podría recurrir a resolver el sistema utilizando factorización triangular, con un coste $O(n^3)=O(ns^3nc^3)$. Esto significa que el coste global se incrementa en un factor $(7/6)^3=1.59$ al usar cuaterniones en lugar de ángulos de Euler. Esto llega hasta el caso extremo cuando se utiliza la propia matriz de rotación, que eleva la penalización a un factor 2^3=8. Esto muestra la importancia que puede implicar reducir esta penalización. En el caso de utilizar matrices dispersas, es bastante común el uso de solvers dispersos generales. En este caso, la gran dispersión del conjunto de ecuaciones (14) puede reducir la diferencia, pero hay que tener en cuenta que esos solvers no siempre aprovecharán al máximo las características particulares de las ecuaciones de restricción. Una forma posible de reducir el impacto de estos problemas es el uso del método del espacio nulo explicado anteriormente.

Para un sólido i en particular, se puede afirmar:

$$\phi_{ri}(\boldsymbol{o}_i)=0 \tag{15}$$

Si se hace una linealización:

$$\phi_{ri}(\boldsymbol{o}_i)\simeq\phi_{ri}(\boldsymbol{o}_{i0})+\boldsymbol{H}_{ri}(\boldsymbol{o}_{i0})(\boldsymbol{o}_i-\boldsymbol{o}_{i0})=0\rightarrow\boldsymbol{H}_{ri}(\boldsymbol{o}_{i0})\boldsymbol{o}_i=-\phi_{ri}(\boldsymbol{o}_{i0})+\boldsymbol{H}_{ri}(\boldsymbol{o}_{i0})\boldsymbol{o}_{i0} \tag{16}$$

Siendo:

$$\boldsymbol{H}_{ri}(\boldsymbol{o}_{i0})=\left|\frac{\partial\phi_{ri}(\boldsymbol{o}_i)}{\partial\boldsymbol{o}_i}\right|_{\boldsymbol{o}_{i0}} \tag{17}$$

Se puede escribir:

$$\boldsymbol{o}_i=\boldsymbol{o}_{ip}+\boldsymbol{N}_{ri}\bar{\boldsymbol{o}}_i \tag{18}$$

Siendo $\boldsymbol{o}_{ip}$ una solución particular de la ecuación (16), $\boldsymbol{N}_{ri}$ el subespacio nulo de $\boldsymbol{H}_{ri}(\boldsymbol{o}_{i0})$ y $\bar{\boldsymbol{o}}_i$ cualquier vector de tamaño 3. Tanto $\boldsymbol{o}_{ip}$ como $\boldsymbol{N}_{ri}$ pueden calcularse en la misma factorización que se expone en [9].

Aunque cualquier elección para una solución particular es adecuada, suele ser interesante recurrir a la solución de norma mínima, por un pequeño coste adicional. La ecuación (18) permite reducir la cantidad de variables de cualquier sistema de ecuaciones cuya variable sea $\boldsymbol{q}$. Es fácil generalizar esta idea al caso de ecuaciones expresadas en términos de $\dot{\boldsymbol{q}}$ o $\ddot{\boldsymbol{q}}$. Tomando la derivada con respecto al tiempo de (15):

$$\left|\frac{\partial \phi_{ri}(\boldsymbol{o}_i)}{\partial \boldsymbol{o}_i}\right|_{\boldsymbol{o}_{i0}} \frac{\partial \boldsymbol{o}_i}{\partial t}=0 \rightarrow \boldsymbol{H}_{ri}(\boldsymbol{o}_{i0})\dot{\boldsymbol{o}}_i=0 \tag{19}$$

Que permite escribir:

$$\dot{\boldsymbol{o}}_i=\dot{\boldsymbol{o}}_{ip}+\boldsymbol{N}_{ri}\dot{\bar{\boldsymbol{o}}}_i \tag{20}$$

Siendo $\dot{\boldsymbol{o}}_{ip}$ una solución particular de la ecuación (19), $\boldsymbol{N}_{ri}$ el subespacio nulo de $\boldsymbol{H}_{ri}(\boldsymbol{o}_{i0})$ y $\dot{\bar{\boldsymbol{o}}}_i$ cualquier vector de tamaño 3. Nótese que $\dot{\boldsymbol{o}}_{ip}$ se puede obtener con la misma factorización realizada para obtener $\boldsymbol{N}_{ri}$ y $\boldsymbol{o}_{ip}$, en el caso de que se necesiten la variable y su derivada. También es fácil averiguar que, en este caso, una solución particular $\dot{\boldsymbol{o}}_i$ de es $\dot{\boldsymbol{o}}_i=\boldsymbol{0}$. En estas condiciones:

$$\dot{\boldsymbol{o}}_i=\boldsymbol{N}_{ri}\dot{\bar{\boldsymbol{o}}}_i \tag{21}$$

En caso de que se necesiten las aceleraciones, se puede volver a derivar respecto del tiempo:

$$\boldsymbol{H}_{ri}(\boldsymbol{o}_{i0})\ddot{\boldsymbol{o}}_i=0 \tag{22}$$

Y, de nuevo:

$$\ddot{\boldsymbol{o}}_i=\ddot{\boldsymbol{o}}_{ip}+\boldsymbol{N}_{ri}\ddot{\bar{\boldsymbol{o}}}_i \tag{23}$$

Siendo $\ddot{\boldsymbol{o}}_{ip}$ una solución particular de la ecuación (23), $\boldsymbol{N}_{ri}$ el espacio nulo de $\boldsymbol{H}_{ri}(\boldsymbol{o}_{i0})$ y $\ddot{\bar{\boldsymbol{o}}}_i$ cualquier vector de tamaño 3. Además, una solución particular de $\ddot{\boldsymbol{o}}_i$ es $\ddot{\boldsymbol{o}}_i=\boldsymbol{0}$. En estas condiciones:

$$\ddot{\boldsymbol{o}}_i=\boldsymbol{N}_{ri}\ddot{\bar{\boldsymbol{o}}}_i \tag{24}$$

Las ecuaciones (18), (21) y (24) permiten reducir cualquier sistema de ecuaciones lineales cuyas variables incluyan $\boldsymbol{o}_i$, $\dot{\boldsymbol{o}}_i$ y $\ddot{\boldsymbol{o}}_i$.

4. Un caso particular: aplicación a la dinámica de sólidos utilizando diferencias centrales.

Aquí se aplicarán estas ideas para reducir el costo del solver presentado en [10,11] para integrar problemas de dinámica de cuerpos múltiples. En dicho algoritmo, para cada sólido, la ecuación de equilibrio linealizada se puede escribir de la forma:

$$\begin{pmatrix} \boldsymbol{M}_{di} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}^{\boldsymbol{T}}(\boldsymbol{o}_i)\left|\frac{d\dot{\boldsymbol{\omega}}_i}{d\dot{\bar{\boldsymbol{o}}}_i}\right|_{\boldsymbol{o}_i,\dot{\boldsymbol{o}}_i^{(0)}} \end{pmatrix}\begin{pmatrix} \ddot{\boldsymbol{x}}_i \\ \ddot{\bar{\boldsymbol{o}}}_i \end{pmatrix}+$$

$$+\begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}(\boldsymbol{o}_i)^T\left|\frac{d\dot{\boldsymbol{\omega}}_i}{d\dot{\boldsymbol{o}}_i}\right|_{\boldsymbol{o}_i,\dot{\boldsymbol{o}}_i^{(0)},\ddot{\boldsymbol{o}}_i^{(0)}}+\boldsymbol{\omega}_i^{(0)}x\boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}(\boldsymbol{o}_i)^T\left|\frac{d\boldsymbol{\omega}_i}{d\dot{\boldsymbol{o}}_i}\right|_{\boldsymbol{o}_i}+\left|\frac{d\left(\boldsymbol{\omega}_i x\boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}(\boldsymbol{o}_i)^T\boldsymbol{\omega}_i^{(0)}\right)}{d\boldsymbol{\omega}_i}\right|_{\boldsymbol{o}_i^{(0)}}\left|\frac{d\boldsymbol{\omega}_i}{d\dot{\boldsymbol{o}}_i}\right|_{\boldsymbol{o}_i} \end{pmatrix}\begin{pmatrix} \dot{\boldsymbol{x}}_i \\ \dot{\boldsymbol{o}}_i \end{pmatrix}= \tag{25}$$

$$=\begin{pmatrix} \boldsymbol{f}_{exti} \\ \boldsymbol{t}_{exti}+\boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}(\boldsymbol{o}_i)^T\left|\frac{d\dot{\boldsymbol{\omega}}_i}{d\dot{\boldsymbol{o}}_i}\right|_{\boldsymbol{o}_i,\dot{\boldsymbol{o}}_i^{(0)},\ddot{\boldsymbol{o}}_i^{(0)}}\dot{\boldsymbol{o}}_i^{(0)}+\left|\frac{d\left(\boldsymbol{\omega}_i x\boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}(\boldsymbol{o}_i)^T\boldsymbol{\omega}_i^{(0)}\right)}{d\boldsymbol{\omega}_i}\right|\left|\frac{d\boldsymbol{\omega}_i}{d\dot{\boldsymbol{o}}_i}\right|_{\boldsymbol{o}_i^{(0)}}\dot{\boldsymbol{o}}_i \end{pmatrix}$$

Siendo:

$\boldsymbol{M}_{di}$ la matriz de masa traslacional del sólido i. Es una matriz diagonal de 3x3 con la matriz del sólido en todos los elementos de la diagonal.

$\boldsymbol{R}(\boldsymbol{o}_i)$ la matriz de rotacion obtenida a partir de la orientación del sólido.

$\boldsymbol{I}_{gi}$ la matriz de inercia rotacional en el centro de gravedad del sólido expresada en el sistema local de coordenadas.

$\boldsymbol{\omega}_i$ la velocidad angular del sólido

$\dot{\boldsymbol{o}}_i^{(0)}$ la aproximación actual de la primera derivada de los parámetros que definen la orientación del sólido.

$\boldsymbol{x}$ las coordenadas de traslación del sólido

$\boldsymbol{f}_{exti}$ el vector de fuerzas externas aplicadas al sólido

$\boldsymbol{t}_{exti}$ los pares aplicados al sólido

Es importante indicar que todas las magnitudes en esta expresión se plantean en t, dado que en diferencias centrales la expresión de equilibrio se formula al inicio del intervalo. Los términos que dependen de las velocidades se resuelven de forma iterativa, dado que en diferencias centrales se conoce la variable en el instante inicial, pero la primera derivada se resuelve en ese paso de integración. A esta ecuación así linealizada se le aplica el método de integración, y el número de variables es 3 más el número de variables en el sistema de orientación empleado.

Se puede reformular la ecuación (25):

$$\begin{pmatrix} \boldsymbol{M}_{di} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_{ri} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{x}}_i \\ \ddot{\boldsymbol{o}}_i \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{ri} \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{x}}_i \\ \dot{\boldsymbol{o}}_i \end{pmatrix} = \\ = \begin{pmatrix} \boldsymbol{f}_{exti} \\ \boldsymbol{t}_{exti} + \boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}(\boldsymbol{o}_i)^T \left| \dfrac{d\dot{\boldsymbol{\omega}}_i}{d\dot{\boldsymbol{o}}_i} \right|_{\boldsymbol{o}_i, \dot{\boldsymbol{o}}_i^{(0)}, \ddot{\boldsymbol{o}}_i^{(0)}} \dot{\boldsymbol{o}}_i^{(0)} + \left| \dfrac{d\left(\boldsymbol{\omega}_i x \boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}(\boldsymbol{o}_i)^T \boldsymbol{\omega}_i^{(0)}\right)}{d\boldsymbol{\omega}_i} \right| \left| \dfrac{d\boldsymbol{\omega}_i}{d\dot{\boldsymbol{o}}_i} \right|_{\boldsymbol{o}_i^{(0)}} \dot{\boldsymbol{o}}_i \end{pmatrix} \tag{26}$$

Y, en lugar de emplear esta ecuación en el integrador, introducimos (21) y (24), llegando a:

$$\begin{pmatrix} \boldsymbol{M}_{di} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_{ri}\boldsymbol{N}_{ri} \end{pmatrix} \begin{pmatrix} \ddot{\boldsymbol{x}}_i \\ \dot{\tilde{\boldsymbol{o}}}_i \end{pmatrix} + \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{C}_{ri}\boldsymbol{N}_{ri} \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{x}}_i \\ \tilde{\boldsymbol{o}}_i \end{pmatrix} = \\ = \begin{pmatrix} \boldsymbol{f}_{exti} \\ \boldsymbol{t}_{exti} + \boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}(\boldsymbol{o}_i)^T \left| \dfrac{d\dot{\boldsymbol{\omega}}_i}{d\dot{\boldsymbol{o}}_i} \right|_{\boldsymbol{o}_i, \dot{\boldsymbol{o}}_i^{(0)}, \ddot{\boldsymbol{o}}_i^{(0)}} \dot{\boldsymbol{o}}_i^{(0)} + \left| \dfrac{d\left(\boldsymbol{\omega}_i x \boldsymbol{R}(\boldsymbol{o}_i)\boldsymbol{I}_{gi}\boldsymbol{R}(\boldsymbol{o}_i)^T \boldsymbol{\omega}_i^{(0)}\right)}{d\boldsymbol{\omega}_i} \right| \left| \dfrac{d\boldsymbol{\omega}_i}{d\dot{\boldsymbol{o}}_i} \right|_{\boldsymbol{o}_i^{(0)}} \dot{\boldsymbol{o}}_i \end{pmatrix} \tag{27}$$

Con esto se consigue reducir la cantidad de variables a 6. Un tratamiento similar se puede aplicar a las ecuaciones de restricción o las de elementos discretos, como muelles o amortiguadores. Una vez integrado el sistema, basta con aplicar las ecuaciones (18), (20) y/o (23) para obtener las variables en el sistema de orientación completo.

5. Ejemplos

Para demostrar la eficiencia del método, se han elegido tres ejemplos. Todos ellos se han extraído del IFToMM Multibody Benchmark [12]. Se han probado en un ordenador con una CPU Intel Core i7-7700HQ a 2,8 GHz. La velocidad de reloj se ha fijado para reducir la dispersión en los resultados debido a la limitación de la CPU. Los resultados que se muestran son el promedio de 5 ejecuciones.

5.1. Péndulo simple

Este es el peor escenario posible. El péndulo está construido con dos cuerpos. Uno de ellos es el suelo y el otro es el péndulo. El problema con este ejemplo es que la pequeña cantidad de cuerpos da lugar a matrices de tamaño pequeño y la construcción de ecuaciones es de un orden similar al de la solución. Además, en el código se tiene en cuenta el hecho de que el primer cuerpo es fijo, reduciendo así aún más el coste del conjunto lineal de ecuaciones.

El mecanismo está compuesto por una masa puntual m = 1 kg y una varilla sin masa de longitud L = 1 m. La varilla está conectada al suelo a través de una articulación giratoria que restringe el movimiento del sistema al plano x-y. El sistema se mueve bajo los efectos de la gravedad desde una posición horizontal y el tiempo total de simulación es de 10 s.

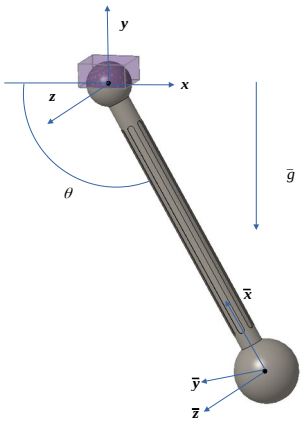

Figura 1: Representación del péndulo simple

En una primera prueba, la comparación se realiza utilizando cuaterniones como parametrización de la matriz de rotación. Los tiempos medios totales para resolver el problema sin y con reducción se muestran en la tabla 1. Como se puede ver, el uso de la reducción en este caso es contraproducente. Los resultados obtenidos con ambos métodos son iguales a una precisión cercana a la de la máquina. Esto se debe al pequeño tamaño de las matrices involucradas, lo que lleva a una falta de relevancia del término O(n^3) en el coste computacional. Dicho en otras palabras, el coste de construcción de las ecuaciones es, en este caso, mayor que el de la resolución.

Tabla 1: Resultados en el caso del péndulo simple empleando cuaterniones

Método	Tiempo
Normal	0.1160
Reducido	0.1344

Si se utiliza la matriz de rotación completa como conjunto de parámetros que definen la orientación, se obtienen los resultados de la tabla 2.

Tabla 2: Resultados en el caso del péndulo simple empleando la matriz de rotación

Método	Tiempo
Normal	0.4019
Reducido	0.3701

Como se puede ver, el coste de emplear la matriz de rotación completa es tan grande que incluso en este caso el algoritmo reducido compensa. De nuevo, los resultados obtenidos con el método normal y el reducido son similares a nivel de precisión de máquina.

5.2. Mecanismo de Bricard

El segundo ejemplo es el mecanismo de Bricard rectangular. Es importante porque está restringido de manera redundante en todo el rango de su movimiento, lo que significa que es bastante sensible a cuestiones numéricas. Por lo tanto, es una buena opción para verificar si el algoritmo conduce a alguna degradación numérica de la solución.

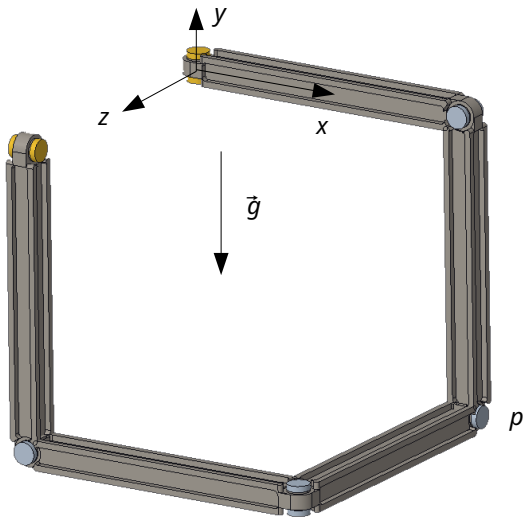

Figura 2: Mecanismo de Bricard

Nuevamente, los resultados son comparables a nivel de la precisión de la máquina en ambos algoritmos, el original y el que utiliza la reducción del espacio nulo de los parámetros de orientación, ambos utilizando cuaterniones y la matriz de rotación. Esto demuestra que la degradación numérica introducida por la reducción es inexistente. Cuando se utilizan cuaterniones como parametrización de la matriz de rotación, los resultados se muestran en la tabla 3.

Tabla 3: Resultados en el caso del mecanismo de Bricard empleando cuaterniones

Método	Tiempo
Normal	0.4667
Reducido	0.3947

Los resultados muestran una mejora de alrededor del 15%. Si se utiliza la matriz de rotación completa, los resultados se muestran en la tabla 4.

Tabla 4: Resultados en el caso del mecanismo de Bricard empleando la matriz de rotación

Método	Tiempo
Normal	21.9593
Reducido	4.2322

Obviamente, en este caso la reducción de coste es mucho más importante, llegando al 80%. Si por alguna razón fuese necesario emplear la matriz de rotación en un problema particular, este algoritmo representaría una mejora muy importante.

6. Conclusiones y trabajos futuros

El uso del subespacio nulo es una buena opción para reducir el coste computacional asociado a la resolución de problemas multicuerpo, tanto dinámicos como cinemáticos. El algoritmo presentado aquí permite mitigar en gran medida los costes computacionales derivados del uso de sistemas de coordenadas redundantes, lo que permite considerar incluso el uso de sistemas altamente redundantes, como la propia matriz de rotación.

El método es de aplicación general, y se han establecido las bases generales para su aplicación. El método se ha integrado con éxito en una dinámica multicuerpo de diferencias centrales, lo que permite reducir considerablemente el coste computacional del algoritmo. Se han resuelto algunos ejemplos con este algoritmo. Los resultados numéricos muestran que el algoritmo de espacio nulo utilizado es de una eficiencia considerable.

Otros desarrollos incluyen la aplicación de la técnica a otros algoritmos, como métodos implícitos para resolver dinámicas multicuerpo y también a otros problemas, como problemas de posición inicial o síntesis de mecanismos.

7. Agradecimientos

Los autores quieren agradecer al Gobierno Vasco por la financiación al grupo de investigación IT1542-22. También agradecen especialmente la ayuda PID2021-124677NB-I00 otorgada por el ministerio de ciencia e innovación MCIN/AEI/10.13039/501100011033 y por “ERDF A way of making Europe”

8. Referencias

[1] O.A. Bauchau, L. Trainelli, The Vectorial Parameterization of Rotation, Nonlinear Dynamics 2003 32:1 32 (2003) 71–92. https://doi.org/10.1023/A:1024265401576.

[2] A. Kissel, J. Taves, D. Negrut, Constrained Multibody Kinematics and Dynamics in Absolute Coordinates: A Discussion of Three Approaches to Representing Rigid Body Rotation, Journal of Computational and Nonlinear Dynamics 17 (2022). https://doi.org/10.1115/1.4055140.

[3] A. Müller, Computational local mobility analysis of mechanisms with higher kinematic pairs, in: Proceedings of the ASME Design Engineering Technical Conference, 2016. https://doi.org/10.1115/DETC2016-60267.

[4] I. Fernández de Bustos, J. Agirrebeitia, R. Avilés, G. Ajuria, Second order analysis of the mobility of kinematic loops via acceleration compatibility analysis, Mechanism and Machine Theory 44 (2009). https://doi.org/10.1016/j.mechmachtheory.2009.04.007.

[5] T.F. Coleman, A. Pothen, The Null Space Problem II. Algorithms, SIAM. J. on Algebraic and Discrete Methods 8 (1987) 544–563. https://doi.org/10.1137/0608045.

[6] I. Fernández De Bustos, V. Garcíamarina, G. Urkullu, Solving the minimum distance problem for the synthesis of mechanisms, 2017. https://doi.org/10.1007/978-3-319-44156-6_40.

[7] I. Fernández de Bustos, H. Uriarte, G. Urkullu, V. García-Marina, A non-damped stabilization algorithm for multibody dynamics, Meccanica 57 (2022) 371–399. https://doi.org/10.1007/s11012-021-01433-0.

[8] J. García de Jalon, E. Bayo, Kinematic and Dynamic Simulation of Multibody Systemas, Springer-Verlag New York, 1994. https://doi.org/10.1007/978-1-4612-2600-0.

[9] I. Fernández de Bustos, V. García-Marina, G. Urkullu, M. Abasolo, An efficient LDU algorithm for the minimal least squares solution of linear systems, Journal of Computational and Applied Mathematics 344 (2018) 346–355. https://doi.org/10.1016/j.cam.2018.05.037.

[10] G. Urkullu, I.F. de Bustos, V. García-Marina, H. Uriarte, Direct integration of the equations of multibody dynamics using central differences and linearization, Mechanism and Machine Theory 133 (2019) 432–458. https://doi.org/10.1016/j.mechmachtheory.2018.11.024.

[11] G. Urkullu, I. Fernández-de-Bustos, A. Olabarrieta, R. Ansola, Estudio de la eficiencia del método de integración directa mediante diferencias centrales (DIMCD)., DYNA-Ingeniería e Industria 96 (2021).

[12] M. González, D. Dopico, U. Lugrís, J. Cuadrado, A benchmarking system for MBS simulation software: Problem standardization and performance measurement, Multibody System Dynamics 16 (2006) 179–190. https://doi.org/10.1007/s11044-006-9020-8.