Transactions on Energy Systems and Engineering Applications

Article

Automatic differential kinematics of serial manipulator robots through dual numbers

Luis Antonio Orbegoso Moreno[1,*] **and Edgar David Valverde Ramírez**[1]

[1] Department of Mechatronic Engineering, National University of Trujillo, Trujillo, Peru.
[*] Correspondence: lorbegoso@unitru.edu.pe

Received: 21 January 2024; Accepted: 09 October 2024; Published: 18 October 2024

Abstract: Dual Numbers are an extension of real numbers known for its capability of performing exact automatic differentiation of one-valued functions theoretically without error approximation. Also, Differential Kinematics of robots involves the computation of the Jacobian, which is a matrix of partial derivatives of the Forward Kinematic equations with respect to the robot's joints. Thus, to perform the automatic calculation of the Jacobian matrix, this paper presents an extension of dual numbers composed of a scalar real part and a vector dual part, where the real part represents the angular value of the robot joint, and the dual part represents the direction of the corresponding partial derivative for each joint. The presented method was implemented in Matlab through Object Orientes Programming (OOP), and the results for calculating the Jacobian of the KUKA KR 500 robot model for 1000 random postures were subsequently compared in terms of execution time and Mean Squared Error (MSE) with other conventional methods: the geometric method, the symbolic method, and the finite difference method. The results showed a significant improvement in the computing time for calculating the Jacobian of the robotic model compared to the other methods, as well as a minimum MSE having as reference the numerical value of the symbolic calculations.

1. Introduction

In the rapidly developing field of robotics, the use of mathematical tools has become essential for overcoming the complex challenges associated with motion analysis and optimization. As robotics continues to advance, accuracy and productivity remain at the forefront of research and development efforts. The reliance on mathematical tools in this context is reflected by the growing acknowledgement of their indispensability. These tools play a pivotal role in enabling engineers and researchers to effectively analyze and optimize robotic movements. Such movements often involve intricate calculations that require precise measurements and sophisticated algorithms to ensure accuracy and efficiency. Thus, the integration of

How to cite this article: Orbegoso Moreno, Luis Antonio; Valverde Ramírez, Edgar David. Automatic differential kinematics of serial manipulator robots through dual numbers. *Transactions on Energy Systems and Engineering Applications*, 5(2): 625, 2024. DOI:10.32397/tesea.vol5.n2.625

advanced mathematical techniques is crucial in pushing the boundaries of robotics and propelling the field forward [1].

Among these tools of mathematics, dual numbers—first introduced by Clifford in his landmark work [2]—stand out as especially prominent. They are a carefully constructed extension of the real number system, and they provide the field with a multitude of uses that are fundamental to it. These special numerical objects, which are characterized by their two components, which include real and dual parts, are highlighted by providing a clear and complete picture that combines spatial position and orientation. This mixing, frequently combined with other mathematical constructions like quaternions, results in the concept of dual quaternions [3]. Also, as explained further in [4], dual numbers can be employed with rotation matrices, effectively capturing the complex spatial arrangements that are intrinsic to robotic systems. As we continue to refine our understanding of robotic functions, the use of two numbers not only improves our understanding of complex kinematics but also opens up a new avenue for tackling the complex mathematics that underpin the field's ongoing development.

Moreover, the wide-ranging impact of two numbers also resonates in the field of Automatic Differentiation (AD), a crucial method used to calculate gradients precisely, a crucial aspect of negotiating the complex landscapes of optimization with skill and accuracy [5]. As such, the research project described in [6] takes a position that emphasizes the inherent value of AD techniques for scalar functions. This restricted scope leads to an awareness of its intrinsic limits, which become especially apparent when one is faced with the complex problem of calculating Jacobians for rotation elements in the Lie group SO(3), which is of great importance in the field of robotics. As a result, the research adopts a forward-thinking approach by extending the AD theory's boundaries to successfully include differentiable manifolds. This enormous extension is completed with the release of a finely tuned C++ library, which is a realization of this expanded theory, complete with the powerful incorporation of template-based expressions. This novel combination provides scholars and professionals with an easy-to-use interface for quickly performing Jacobian analyses, which is a big step in the direction of increasing the effectiveness and availability of these crucial calculations.

Similar to how the extension of complex numbers results in quaternions, which allow for the sophisticated representation of spatial rotations, the extension of dual numbers also leads to hyper-dual numbers. This is a very promising mathematical construct that was investigated by looking at its algebraic properties in the context of kinematics [7]. It empowers the modeling of complex multibody kinematics and opens the door to the formulation of dynamic equations controlling the motion of rigid bodies when combined with the powerful tools provided by the field of Lie groups [8]. By means of this complex interplay of ideas, the foundations of hyper-dual numbers and their relation to Lie group theory become a powerful tool for addressing the intricacies of physical motion and interaction in a way that harmonizes application and abstraction.

It is necessary to compute second-order position derivatives in the field of mechanics, in addition to first-order differentials. The formulation of Newton's second law depends on this computation. This need serves as a motivation for the research described in [9], where the direct evaluation of motion equation, which includes positional derivatives such as acceleration and velocity, is made possible by the use of hyper-dual numbers. These results are strikingly truncation-free, which distinguishes them from other numerical differentiation approximations. Higher-order derivatives must be analyzed in the context of mechanical modeling because of this crucial second derivative. Then, as mentioned in [10], the third derivative, called jerk, and the fourth derivative, called jounce or snap, have importance in this field. Based on this assumption, the study published in [11] presents an expanded formulation using dual numbers. This expression is a deliberate reaction directed towards the calculation of higher-order derivatives, adding to the larger collection of mathematical instruments accessible for dealing with the complex intricacies of mechanical dynamics.

Turning to yet another important aspect, robotics is centered on the complex process of calculating the Jacobian matrix, which is a basic concept with broad application in research and development [12]. This matrix, which contains the derivatives that relate joint velocities to end-effector velocities, serves as the foundation for a variety of numerical techniques, the most important of which being the kinematic inversion technique [13]. By integrating this inversion mechanism, robotic systems can move backward from desired states of the end-effector to matching joint configurations. This provides useful information about possible paths for obtaining accurate and dynamically adaptive motions [14]. But the Jacobian matrix's impact goes far beyond its use in simple motion analysis; it has a profound impact on more complex techniques like trajectory planning, obstacle avoidance, and developing robust control strategies, which emphasizes the matrix's essential and indispensable status in the field of robotic studies [15]. The thorough understanding and deft manipulation of the Jacobian matrix contribute to our growing understanding of robotic locomotion and set the stage for a plethora of novel applications that have the potential to push the limits of robotic capabilities.

Our current study represents a significant advancement in this regard, leveraging the powerful combination of dual numbers and the field of robot kinematics. We present a framework that directly computes the Jacobian matrix, a task that typically requires a rigorous geometric approach, by utilizing the special properties of dual numbers. Using dual numbers composed by a scalar real part and a vector dial part to represent the robot's joint variables, we build a paradigm that aesthetically combines joint values and matrices into a cohesive whole. This clever synthesis offers a novel and potent approach to solving persistent problems in robot motion analysis, in addition to promising an increase in computational efficiency. It was demonstrated the effectiveness of this approach with thorough validation on the sturdy of the 6-degree-of-freedom (DoF) KUKA KR 500 robot through the calculation of the Jacobian matrix corresponding to each of 1000 random configuration of the robot and comparing these results with the output of other conventional approaches (the geometric, symbolic, and finite difference methods), where in terms of computation time and MSE, the presented method represented a significant breakthrough, demonstrating the unrealized potential of combining complex mathematical structures with the complexities of robotic systems.

The following is a description of the study's organizational framework: In Section II, the formulation and development of the suggested method for the direct calculation of the Jacobian matrix via the novel use of dual numbers are explained in detail. Moreover, in the same section, a thorough explanation of the kinematic model used to support the assessments relevant to the suggested approach is provided, as well as the presentation of the methods to be compared. Section III, which follows, is devoted to presenting the findings from the conducted experiments and, in doing so, effectively contrasting the efficiency of the suggested strategy with the traditional methods that are employed to compute the Jacobian matrix. The final manifestation of this work's culmination is found in Section IV, where thorough conclusions were drawn.

2. Methods and Materials

2.1. Preliminary Concepts

As mentioned in previous lines, Clifford introduced the dual numbers set, represented by $\mathbb{D}$ in this article, in the latter part of the 19th century, as was previously mentioned [2]. This extension of real numbers are represented by the following format of expressions:

$$\hat{x} = x_r + \varepsilon x_d, \tag{1}$$

where both x_d and x_r belong to real numbers $\mathbb{R}$ and represent the dual and real parts of the dual number $\hat{x}$, respectively. Also, it is worth mentioning that the dual entity ε satisfies the property $\varepsilon^2 = 0$ where $\varepsilon \neq 0$. This numerical group, like other sets such as real and complex sets, meets certain arithmetic properties that are highlighted below.

- Addition

  As can be seen in equation 2, the sum of two dual numbers the result is another dual number, closing property $\hat{x} + \hat{y} \in \mathbb{D}$, whose real part is the sum of real parts and the dual part is the sum of dual parts of the dual numbers to be added. This operation also satisfies the associative property $(\hat{x} + \hat{y}) + \hat{z} = \hat{x} + (\hat{y} + \hat{z})$, and commutative property $\hat{x} + \hat{y} + \hat{z} = \hat{x} + \hat{z} + \hat{y}$ [16].

$$\hat{x} + \hat{y} = (x_r + y_r) + \varepsilon(x_d + y_d). \tag{2}$$

- Additive identity

  The existence of the additive neutral element ϕ_+ indicates that every dual number added with this element, the result is itself $\hat{x} + \phi_+ = \hat{x}$ [16].

$$\phi_+ = 0 + \varepsilon 0. \tag{3}$$

- Additive inverse

  As shown in equation 4, every dual number $\hat{x}$ has its inverse additive $-\hat{x}$; which when added results in the additive identity $\hat{x} + -\hat{x} \ = \phi_+$ [16].

$$-\hat{x} = -x_r - \varepsilon x_d. \tag{4}$$

- Multiplication

  The multiplication of two dual numbers the result is another dual number, closing property $\hat{x} \cdot \hat{y} \in \mathbb{D}$, whose real part is the multiplication of real parts of these numbers and the dual part is the sum of the product between the dual with the real parts of these numbers [16], as is shown in equation 5. It is worth mentioning that given that $x_r, x_d, y_r, y_d, z_r, z_d \in \mathbb{R}$, then is also satisfied the associative property $(\hat{x} \cdot \hat{y}) \cdot \hat{z} = \hat{x} \cdot (\hat{y} \cdot \hat{z})$, and commutative property $\hat{x} \cdot \hat{y} \cdot \hat{z} = \hat{x} \cdot \hat{z} \cdot \hat{y}$; which is not true is that the real and dual part of these numbers belong to other sets whose operations are not commutative nor associative as in the case of quaternions.

$$\hat{x} \cdot \hat{y} = (x_r \cdot y_r) + \varepsilon(x_r \cdot y_d + x_d \cdot y_r). \tag{5}$$

- Multiplicative identity

  Similarly with the addition operation, the multiplicative neutral element, indicated in the equation 6, denotes that every dual number multiplied with this element, the result is itself $\hat{x} \cdot \phi_. = \hat{x}$ [16].

$$\phi_. = 1 + \varepsilon 0 \tag{6}$$

- Inverse multiplicative

  With the exception of dual numbers whose real part is zero $x_r = 0$, all other dual number $\hat{x}$ has its inverse multiplicative $\hat{x}^{-1}$; which when multiplied, the result is the multiplicative identity $\hat{x} \cdot \hat{x}^{-1} \ = \phi_.$ [17].

$$\hat{x}^{-1} = \frac{1}{x_r} - \varepsilon \frac{x_d}{x_r^2}. \tag{7}$$

Now, the most important property that owns the dual numbers is their capability to compute automatic differentiation of one-valued functions ($f : \mathbb{R} \to \mathbb{R}$). This last property can be demonstrated through the Taylor Series of a one-valued function f valued at the dual number $\hat{x}$ around its real part x_r, as is shown in 8; here, the terms whose derivative order is greater than one are canceled due to the dual number property $\varepsilon^2 = 0$ [17].

$$f(\hat{x}) = f(x_r) + \frac{f^{(1)}(x_r)}{1!}(\varepsilon x_d) + \frac{f^{(2)}(x_r)}{2!}\cancelto{0}{(\varepsilon x_d)^2} + \frac{f^{(3)}(x_r)}{3!}\cancelto{0}{(\varepsilon x_d)^3} + \dots \,. \tag{8}$$

Therefore, the result of the expression 8 is the equation 9, where it is proven that a function valued on a dual number, the result is also a dual number, whose real part is the evaluation of the function on x_r, and the dual part is the multiplication of the dual part x_d by the first derivative of the function valued at x_r. Hence, if it is required that in the dual part just be the derivative of the function, then $x_d = 1$ [18].

$$f(\hat{x}) = f(x_r) + \varepsilon x_d f^{(1)}(x_r). \tag{9}$$

A fact that reflects the veracity of the property of automatic differentiation that this numerical set possesses is found in the definition of its multiplicative inverse, which for real numbers would be just simply $\frac{1}{x_r}$ as long as $x_r \neq 0$. However, in the dual part of equation 7 is exactly the derivative of $\frac{1}{x_r}$, which is $\frac{-1}{x_r^2}$, multiplied by the dual value x_d as stated in equation 9.

2.2. Extending to Dual Numbers for Partial Derivatives of Multivalued Functions

Until now, it was presented the capability of ordinary dual numbers to compute automatic differentiations for one-valued functions. Nevertheless, for multivalued functions ($f : \mathbb{R}^n \to \mathbb{R}$) the first derivative is found in an array of the first partial derivative of the function's output with respect to each of its inputs, known as the gradient of the function f. Thus, to extend the capacity of dual numbers to perform the automatic computation of multivalued function's gradients, we present a variety of dual numbers composed by a scalar value as the real part, and a vector as the dual part, as shown in equation 10.

$$\hat{x} = x + \varepsilon \vec{v}. \tag{10}$$

Here, $x \in \mathbb{R}$ and $\vec{v} \in \mathbb{R}^{1\times n}$; for this reason, this set of extended dual numbers is denoted as $\mathbb{D}_{\mathbb{R}\oplus\mathbb{R}^n}$, where $\mathbb{D}$ represents the set of dual numbers, $\mathbb{R}$ denotes the real numbers (scalars), $\mathbb{R}^n$ represents a vector space of dimension n over the real numbers, and $\oplus$ indicates the direct sum of these sets. And like any numerical set, this one has arithmetic properties of internal composition, as shown in the following items.

- Addition

  Like ordinary dual numbers, this operation satisfies the properties of closure, commutativity and associativity. But this operation can occur as long as the dimension of the vector of the dual part of $\hat{x}_1$ is equal to the dimension of the vector of the dual part of $\hat{x}_2$.

$$\hat{x}_1 + \hat{x}_2 = (x_1 + x_2) + \varepsilon(\vec{v}_1 + \vec{v}_2). \tag{11}$$

- Additive identity

  In this case, the additive neutral element is one whose scalar part is 0 and its dual part is a vector with all components equal to 0.

$$\phi_{+} = 0 + \varepsilon \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \end{bmatrix}_{[1\times n]}. \tag{12}$$

- Additive inverse
  The additive inverse is only the negation of the scalar term, as well as the negation of all components of the vector of the dual part.

$$-\hat{x} = -x - \varepsilon\vec{v}. \tag{13}$$

- Multiplication
  This operation, as happened with ordinary dual numbers, complies with the property of closure, commutativity and associativity. It is important to note that the property $\varepsilon^2 = 0$ is what allows only scalar values to exist in the real part, while only vector values exist in the dual part. And as in the case of addition, this operation can only be carried out if the dimension of the vectors of the vector parts of the numbers $\hat{x}_1$ and $\hat{x}_2$ are equal.

$$\hat{x}_1 \cdot \hat{x}_2 = (x_1 \cdot x_2) + \varepsilon(x_1 \cdot \vec{v}_2 + x_2 \cdot \vec{v}_1). \tag{14}$$

- Multiplicative identity
  The multiplicative neutral element has only unity as a scalar value, therefore, the components of the vector of the dual part are all 0.

$$\phi_{\cdot} = 1 + \varepsilon \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \end{bmatrix}_{[1\times n]}. \tag{15}$$

- Inverse multiplicative
  Due to the fact that the inverse of a dual number only depends on its real value being different from 0, the formula is not different from the property of the multiplicative inverse of ordinary dual numbers. In this case, the vector is negated and divided by the square of the scalar value of the extended dual number.

$$\hat{x}^{-1} = \frac{1}{x} - \varepsilon\frac{\vec{v}}{x^2}. \tag{16}$$

Regarding the result of operating on the extended dual numbers of the set $\mathbb{D}_{\mathbb{R}\oplus\mathbb{R}^n}$ with scalar values, this is discussed in the following items.

- Multiplication by a scalar
  In this case, the scalar value is distributed to both the real part and the dual part of the element of $\mathbb{D}_{\mathbb{R}\oplus\mathbb{R}^n}$.

$$\lambda \cdot \hat{x} = \lambda \cdot x + \varepsilon\lambda \cdot \vec{v}. \tag{17}$$

- Addition with a scalar
  Here the scalar number can be considered as an element of the set $\mathbb{D}_{\mathbb{R}\oplus\mathbb{R}^n}$ whose vector part is of the same dimension as that of the element $\hat{x}$, but with all the components of the vector equal to 0.

$$\lambda + \hat{x} = (\lambda + x) + \varepsilon\vec{v}. \tag{18}$$

Now, to compute the gradient of a multivalued function, if the function f has n inputs, then n extended dual numbers will be created whose dual part has a vector of n components. Thus, for the first input it will be indicated that the first component of its vector is 1 while the others will be 0, this will indicate that the derivative with respect to the first input of the function will be found in the first component of the vector of the dual part; with this same logic, we proceed with the second input of the function, where this time it is the second component of its vector that will have the value of 1 and the other components will be equal to 0, which will indicate that the partial derivative with respect to the second input will be found in the second component of the vector of the dual part of the number resulting from the evaluation of f; and so on consecutively for the other numbers as is indicated in 19.

$$\begin{array}{ccccl} \hat{x}_1 & = & x_1 & + & \varepsilon[1 \;\; 0 \;\; \dots \;\; 0]_{[1\times n]} \\ \hat{x}_2 & = & x_2 & + & \varepsilon[0 \;\; 1 \;\; \dots \;\; 0]_{[1\times n]} \\ & & & \vdots & \\ \hat{x}_n & = & x_n & + & \varepsilon[0 \;\; 0 \;\; \dots \;\; 1]_{[1\times n]} \end{array}. \tag{19}$$

The result of evaluating the multivalued function f with these extended dual numbers $\hat{x}_1, \hat{x}_2, ..., \hat{x}_n$ will be another element of $\mathbb{D}_{\mathbb{R}\oplus\mathbb{R}^n}$ whose scalar part will be the function f evaluated in the scalar parts $x_1, x_2, ..., x_n$, while the part dual will be the gradient vector of the function f, as shown below as,

$$f(\hat{x}_1, \hat{x}_2, ..., \hat{x}_n) = f(x_1, x_2, ..., x_n) + \varepsilon\nabla f. \tag{20}$$

The extended dual numbers that form the system of equations 19 can be summarized and compacted into a single matrix expression, as shown in equation 21. Where the dual column vector $\hat{X}$ is equal to the column vector of scalar values $\hat{X}$ plus the dual part, which is a square matrix M composed by the vertical stack of the vectors of the dual parts of the numbers $\hat{x}_1, \hat{x}_2, ..., \hat{x}_n$; vectors which form a canonical or standard basis for an n-dimensional space, which generates that the matrix M be the identity matrix. The expression 21 also allows generalizing the basis for an n-dimensional space, then the matrix M can be any matrix of rank and dimension n.

$$\hat{X}_{[n\times 1]} = X_{[n\times 1]} + \varepsilon M_{[n\times n]}. \tag{21}$$

In this way, in a more general way expression 20 can be rewritten as shown in equation 22, where the role of M as a transformation matrix for the gradient vector of the function is highlighted, being the trivial case when the row vectors that make up this matrix are the canonical basis for an n-dimension space, which makes the matrix M the identity matrix and therefore the gradient vector is not altered.

$$f(\hat{X}_{[n\times 1]}) = f(X_{[n\times 1]}) + \varepsilon\nabla f M_{[n\times n]}. \tag{22}$$

Finally, for the general case of a multivalued vectorial function $(f : \mathbb{R}^n \to \mathbb{R}^m)$, in this case what will multiply the matrix M will no longer be the gradient vector, but rather the Jacobian matrix of the function f as is expressed in equation 23, and in this way the automatic calculation of the Jacobian matrix is carried out. It is important to highlight the similarity that equation 23 has with equation 9 of ordinary dual numbers, where in both cases automatic differentiation is carried out, but in one it is carried out for multivalued vector functions and in the other for unvalued functions.

$$f(\hat{X}_{[n\times 1]})_{[m\times 1]} = f(X_{[n\times 1]})_{[m\times 1]} + \varepsilon J_{[m\times n]} M_{[n\times n]}. \tag{23}$$

For the present work, since it is about the calculation of the Jacobian matrix, then we will work with the canonical bases for an n-dimensional space, which means that the matrix M will be the identity matrix.

2.3. Calculation of the Numerical Jacobian for Forward Kinematics Using Dual Numbers

To apply the theory presented in the previous lines, to calculate the value of the Jacobian matrix automatically from the forward kinematics equation, which is a multivalued vector equation. For each i joint of the n-degree of freedom robot, a dual number belonging to the set $\mathbb{D}_{\mathbb{R}\oplus\mathbb{R}^n}$ denoted by $\hat{q}_i$ will be created, according to the logic indicated in the equation 19. In this way, to represent the set formed by all these n dual numbers, the notation $\hat{Q}_{[n\times 1]}$ will be adopted, according to the notation presented in 21.

To calculate the forward kinematics (FK), homogeneous transformation matrices (HTM) were used, which are matrices of 4×4 elements, which represent transformations of the reference frames between the frames of the robot links until reaching the end effector through matrix multiplications as is shown in equation 24, where in general the matrix $T_i^{i-1}(\hat{q}_i)$ indicates that is the transformation of the joint frame $i-1$ to the joint frame i, and is also in function of the joint $\hat{q}_i$ [19].

$$FK(\hat{Q}_{[n\times 1]}) = T_1^0(\hat{q}_1)T_2^1(\hat{q}_2)\ldots T_i^{i-1}(\hat{q}_i)\ldots T_n^{n-1}(\hat{q}_n). \tag{24}$$

Once computed the FK, the result is a 4×4 HTM mainly composed by a 3×3 rotation matrix $R(\hat{Q}_{[n\times 1]})$, and a column position vector $P(\hat{Q}_{[n\times 1]})$ as is indicated in the following equation.

$$FK(\hat{Q}_{[n\times 1]})_{[4\times 4]} = \begin{bmatrix} R(\hat{Q}_{[n\times 1]})_{[3\times 3]} & P(\hat{Q}_{[n\times 1]})_{[3\times 1]} \\ [0]_{[1\times 3]} & 1 \end{bmatrix}. \tag{25}$$

With the position vector P found in the equation 25, the calculation of the linear velocity Jacobian J_v is straightforward, because this matrix is found directly in the dual part of the resulted number as is proved in the equation 26.

$$P(\hat{Q}_{[n\times 1]})_{[3\times 1]} = P(Q_{[n\times 1]})_{[3\times 1]} + \varepsilon J_{v[3\times n]}. \tag{26}$$

Unlike the angular velocity Jacobian J_ω, which can not be directly found in the dual part of the equation 27. This is because the derivative of a rotation matrix is a tensor of $3 \times 3 \times n$, which doesn't belong to the rank of a matrix.

$$R(\hat{Q}_{[n\times 1]})_{[3\times 1]} = R(Q_{[n\times 1]})_{[3\times 3]} + \varepsilon R'_{[3\times 3\times n]}. \tag{27}$$

Nonetheless, the angular velocity Jacobian can be calculated indirectly through the multiplication of the derivative of the rotation matrix R' with its transpose R^T, which in deed can be found directly as terms of the equation 27. The result is this multiplication is an antisymmetric matrix in whose components that do not belong to its diagonal are the gradients $(\omega_x, \omega_y, \omega_z)$, as is indicated in the following equation.

$$R'R^T = \begin{bmatrix} 0 & -\omega_{z[1\times n]} & \omega_{y[1\times n]} \\ \omega_{z[1\times n]} & 0 & -\omega_{x[1\times n]} \\ -\omega_{y[1\times n]} & \omega_{x[1\times n]} & 0 \end{bmatrix}. \tag{28}$$

Therefore, with the result can be formed the angular velocity Jacobian through stacking the previously calculated gradient vectors as indicated in equation 29.

$$J_{\omega[3\times n]} = \begin{bmatrix} \omega_{x[1\times n]} \\ \omega_{y[1\times n]} \\ \omega_{z[1\times n]} \end{bmatrix}. \tag{29}$$

And finally, the total Jacobian matrix of FK becomes the vertical stacking of the linear velocity Jacobian matrix together with the angular velocity Jacobian matrix.

$$J_{[6\times n]} = \begin{bmatrix} J_{v[3\times n]} \\ J_{\omega[3\times n]} \end{bmatrix}. \tag{30}$$

It is important to indicate that equation 28 may not be necessary and directly calculate the Jacobian matrix of angular velocity or directly the entire Jacobian matrix if other tools are used to calculate the transformations of the FK reference frames, such as dual quaternions.

Below is a pseudocode as a summary of the steps to follow to perform the automatic calculation of the Jacobian matrix through dual numbers. It is worth mentioning that the elements of set $\mathbb{D}_{\mathbb{R}\oplus\mathbb{R}^n}$ are implemented at the code level using OOP in MATLAB.

Algorithm 1: Algorithm to Compute the Jacobian Matrix through Dual Numbers

```
Data: $[\hat{q}_1, \hat{q}_2, \ldots, \hat{q}_n], HTM$
/* HTM is a function that calculates the homogeneous transformation matrix from a joint and its index $i$ */
Result: $y = J$
$N \leftarrow n$;
$i \leftarrow 1$;
$FK \leftarrow I$;   /* Start with a $4 \times 4$ identity matrix */
while $i \leq N$ do
    $FK \leftarrow FK \cdot HTM(\hat{q}_i, i)$;
    $i \leftarrow i + 1$;
end
$P \leftarrow FK(1:3, 4)$;
$J_v \leftarrow P.dual$;
$R \leftarrow FK(1:3, 1:3)$;
$RR \leftarrow R.dual \cdot R.real.T$;
$\omega_x \leftarrow RR(3,2)$;
$\omega_y \leftarrow RR(1,3)$;
$\omega_z \leftarrow RR(2,1)$;
$J_\omega \leftarrow [\omega_x; \omega_y; \omega_z]$;
$J \leftarrow [J_v; J_\omega]$;
```

2.4. Comparison Methods

For this work, 3 methods were used to compare the execution time for calculating the Jacobian matrix, these methods were: Geometric method, symbolic method, and the finite difference method. Because the implementation of a class causes a delay in calling its methods and attributes, to make a fair comparison, a class for real numbers was also implemented in MATLAB, which were used in the comparison methods.

On the other hand, to compare the error of the Jacobian matrix obtained by these methods, including the proposed method, the numerical values of the result of the symbolic Jacobian were considered as reference values. The calculation of the Mean Square Error (MSE) was carried out with the following formula,

$$MSE = \frac{\sum_{i=1}^{6} \sum_{j=1}^{n} (J_s(i,j) - \hat{J}(i,j))^2}{6 \cdot n}, \tag{31}$$

where J_s is the numerical value of the symbolic Jacobian once its terms are replaced by the values corresponding to a configuration of the robot, while $\hat{J}$ is the Jacobian value obtained by another method other than the symbolic method.

2.4.1. Geometric Method

This method stands out in its ability to efficiently compute the Jacobian matrix for a kinematic chain derived from the Denavit-Hartenberg algorithm. The crux of its effectiveness lies in the systematic propagation of velocity through the entire kinematic chain, adeptly capturing the nuanced interactions at each joint and culminating in a comprehensive representation at the end effector [20]. This distinctive methodology not only attains remarkable computational speed but also ensures a high degree of accuracy, making it an invaluable tool in the realm of robotic kinematics.

This method operates in two stages. In the first stage, the forward kinematics are computed using the homogeneous transformation matrices, denoted as $FK = T_1^0 T_2^1 \cdots T_n^{n-1}$. Subsequently, the second stage involves determining the contribution of each joint to the velocity of the end-effector. This is accomplished by computing the forward kinematics of each joint, denoted as $FK_i = T_0^0 T_1^0 \cdots T_i^{i-1}$, where T_0^0 represents the identity matrix of dimensions 4×4. Depending on the joint type indexed by i, the subsequent steps vary. In the case of a revolute joint, it becomes necessary to compute the vector $p_n^i = FK(1:3,4) - FK_i(1:3,4)$, signifying the distance between the end-effector and the position of the corresponding joint i. Equation 32 illustrates the computation of a column in the Jacobian matrix according to this method.

$$J_{i[6\times1]} = \begin{cases} \begin{bmatrix} z_{i-1}^0 \times p_n^{i-1} \\ z_{i-1}^0 \end{bmatrix} & \text{if } j \text{ is a revolute joint} \\ \\ \begin{bmatrix} z_{i-1}^0 \\ [0]_{[3\times1]} \end{bmatrix} & \text{if } j \text{ is a prismatic joint} \end{cases} . \tag{32}$$

Subsequently, upon the completion of the computation for each column of the Jacobian matrix, the final step involves the horizontal concatenation of these vectors, as exemplified in equation 33. This process results in the formation of the complete Jacobian matrix, denoted as J, representing the comprehensive mapping of the joint velocities to the end-effector's linear and angular velocities.

$$J_{[6\times n]} = \begin{bmatrix} J_1 & J_2 & \dots & J_n \end{bmatrix}. \tag{33}$$

For a more comprehensive understanding, the pseudocode for implementing this method is presented below. This pseudocode meticulously encapsulates the sequential steps elucidated earlier.

2.4.2. Symbolic Method

This method uses the power of symbolic calculus to first symbolically calculate the direct kinematics, and then computes the Jacobian matrix using symbolic differentiation of the resulting FK equation. Although it

Algorithm 2: Algorithm to Compute the Jacobian Using the Geometric Method

```
Data: $[q_1, q_2, \ldots, q_n], HTM$
/* HTM is a function that calculates the homogeneous transformation matrix from a joint and its index $i$ */
Result: $y = J$
$N \leftarrow n$;
$i \leftarrow 1$;
$FK \leftarrow I$;   /* Start with a $4 \times 4$ identity matrix */
while $i \leq N$ do
    $FK \leftarrow FK \cdot HTM(q_i, i)$;
    $i \leftarrow i + 1$;
end
$i \leftarrow 1$;
$FKi \leftarrow I$;   /* Start with a $4 \times 4$ identity matrix */
$J \leftarrow []$;
while $i \leq N$ do
    $z \leftarrow FKi(1:3, 3)$;
    if $i$ is a revolute joint then
        $p \leftarrow FK(1:3, 4) - FKi(1:3, 4)$;
        $Ji \leftarrow [z \times p; z]$;
    else
        $Ji \leftarrow [z; 0; 0; 0]$;
    end
    $J(:, i) \leftarrow Ji$;
    $FKi \leftarrow FKi \cdot HTM(q_i, i)$;
    $i \leftarrow i + 1$;
end
```

is a general method for any formulation of equations, in itself, the symbolic calculation is very slow, so only the time it takes to carry out the substitution and evaluation of the numerical variables in the equation of the Jacobian matrix will be taken into account [21].

2.4.3. Finite Difference Method

This is a general method that allows approximating Jacobian matrices by performing finite differences based on the definition of the derivative [22]. For a multivalued vector function ($f : \mathbb{R}^n \rightarrow \mathbb{R}^m$), the ith column of its respective Jacobian matrix can be approximated using the equation 34 and therefore, the complete Jacobian matrix is calculated by horizontally stacking these columns as in equation 33. It should be noted that for this work we used a $\epsilon = 10^{-5}$.

$$J_i^T \approx \frac{f(x_1, x_2, \ldots, x_i + \epsilon, \ldots, x_n) - f(x_1, x_2, \ldots, x_i, \ldots, x_n)}{\epsilon}. \tag{34}$$

Now, for the direct kinematics function, since HTMs are used, then this function is of type $(f : \mathbb{R}^n \rightarrow \mathbb{R}^{4\times4})$. So to apply equation 34, what is done is to turn the FK a vector composed of the horizontal stacking of the transpose of its columns, as shown in equation 35.

$$f_{[1\times 12]} = [FK(1:3,1)^T, FK(1:3,2)^T, FK(1:3,3)^T, FK(1:3,4)^T]. \tag{35}$$

However, the Jacobian matrix obtained with equations 34 and 33, called $J_{a[12\times n]}$, is not the matrix that interests us. But it is known that the last 3 rows of this are the rows of the linear velocity Jacobian $j_v = J_a(10:12, 1:n)$. Then what is done is to calculate a vector matrix that becomes the derivative of the rotation matrix as indicated in equation 36, and this matrix is multiplied by the transpose of the rotation matrix as indicated in 28 and finally these results are concatenated using equation 30.

$$R' = \begin{bmatrix} J_a(1,1:n) & J_a(4,1:n) & J_a(7,1:n) \\ J_a(2,1:n) & J_a(5,1:n) & J_a(8,1:n) \\ J_a(3,1:n) & J_a(6,1:n) & J_a(9,1:n) \end{bmatrix}. \tag{36}$$

The steps of this method are illustrated below with the following pseudocode.

Algorithm 3: Numeric Jacobian Calculation using Finite Differences

```
Data: $FK, Q, \epsilon$
/* FK is a predefined function that calculates forward kinematics, and Q is an array of joint values */
Result: $J$
$n \leftarrow$ length$(Q)$; /* Dimension of the input $Q$ */
$FK0 \leftarrow FK(Q)$; /* Calculate $f_0$ when no perturbation happens */
$f0 \leftarrow [FK0(1:3,1)^T, FK0(1:3,2)^T, FK0(1:3,3)^T, FK0(1:3,4)^T]$;
$Ja \leftarrow$ zeros(length$(f0), nx)$;
for $i \leftarrow 1$ to $nx$ do
    $Q_{\text{plus}} \leftarrow Q$;
    $Q_{\text{plus}}(i) \leftarrow Q(i) + \epsilon$;
    $FK1 \leftarrow FK(Q_{\text{plus}})$;
    $f1 \leftarrow [FK1(1:3,1)^T, FK1(1:3,2)^T, FK1(1:3,3)^T, FK1(1:3,4)^T]$;
    $Ja(:,i) \leftarrow (f1 - f0)/\epsilon$;
end
$j_v \leftarrow J_a(10:12, 1:n)$;
$dR \leftarrow [J_a(1:3,1:n), J_a(4:6,1:n), J_a(7:9,1:n)]$;
$RR \leftarrow dR \cdot FK0(1:3,1:3).T$;
$\omega_x \leftarrow RR(3,2)$;
$\omega_y \leftarrow RR(1,3)$;
$\omega_z \leftarrow RR(2,1)$;
$J_\omega \leftarrow [\omega_x; \omega_y; \omega_z]$;
$J \leftarrow [J_v; J_\omega]$;
```

2.5. The Robotic Model

The KUKA KR 500 robotic model, is the model that was used in this work to obtain the Jacobian matrix. With six degrees of freedom (DoF), this advanced robotic arm is used in industrial settings. It is purposefully made to handle challenging tasks and adapt to the changing needs of dynamic industrial environments [23].

Within the framework of this model, the DH parameters were obtained by means of the examination of the research report [23]. The kinematic framework of the robotic system was established based on these essential parameters, which are listed in Table 1.

Table 1. Denavit-Hartenberg parameters of the robotic model.

Joint **N**°	Denavit-Hartenberg Parameters			
	$\boldsymbol{\theta}_i$(rad)	$\boldsymbol{d}_i$(m)	$\boldsymbol{a}_i$(m)	$\boldsymbol{\alpha}_i$(rad)
1	q_1	-1.045	0.500	$\frac{\pi}{2}$
2	q_2	0	1.300	0
3	$q_3 + \frac{\pi}{2}$	0	0.055	$-\frac{\pi}{2}$
4	q_4	-1.025	0	$\frac{\pi}{2}$
5	q_5	0	0	$-\frac{\pi}{2}$
6	q_6	-0.290	0	π

Table 2 presents the limits for each articulation in this robotic model. These limits will be taken into account when creating 1000 random points, all of which must be within the robot's task space in order to calculate the Jacobian matrices for each joint configuration.

Table 2. Robot joint limits.

Joint **N**°	$\boldsymbol{\theta}_{\min}$(deg)	$\boldsymbol{\theta}_{\max}$(deg)
1	$-185°$	$185°$
2	$-40°$	$110°$
3	$-184°$	$60°$
4	$-350°$	$350°$
5	$-118°$	$118°$
6	$-350°$	$350°$

3. Tests and Results

Figure 1 presents the point cloud resulting from 1000 random configurations for each of the 6 axes of the case study robotic model. These configurations were generated within the joint limits specified in Table 2. This diverse set of joint positions spans a wide range of possible robot states, providing a comprehensive view of its workspace.

Regarding the comparison of the calculation time of the Jacobian matrix between the 4 intervening methods: method with dual numbers, geometric method, symbolic method, and method with finite differences. Figure 2 shows a violin plot where the distribution of the data of the time spent for each of the evaluated methods can be seen. At first glance, it may seem that the dual number method is the method that has the least time invested to calculate differential kinematics.

But to better appreciate these differences, Table 3 shows the minimum, maximum, median, mean, and standard deviation values for the time data of each method, where the minimum values in each column are highlighted in bold. In this way, it is evident that in all these categories, the method that uses dual numbers is the one with the least investment of time.

Now, regarding the error in the calculation of the Jacobian matrix, Figure 3 shows the violin diagram of the logarithm with base 10 of the MSE, according to the equation 31, for 3 methods with the exception of the symbolic method, because as had been clarified previously, the numerical values of this method

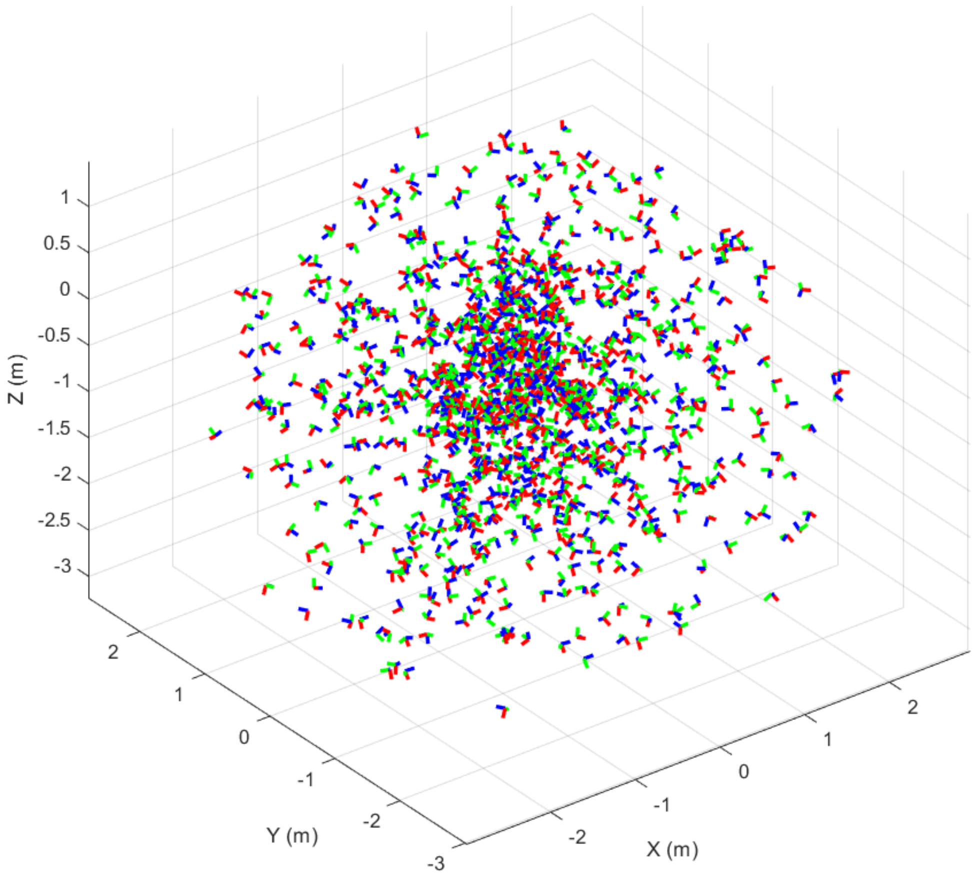

Figure 1. Robot posture cloud for 1000 random configurations.

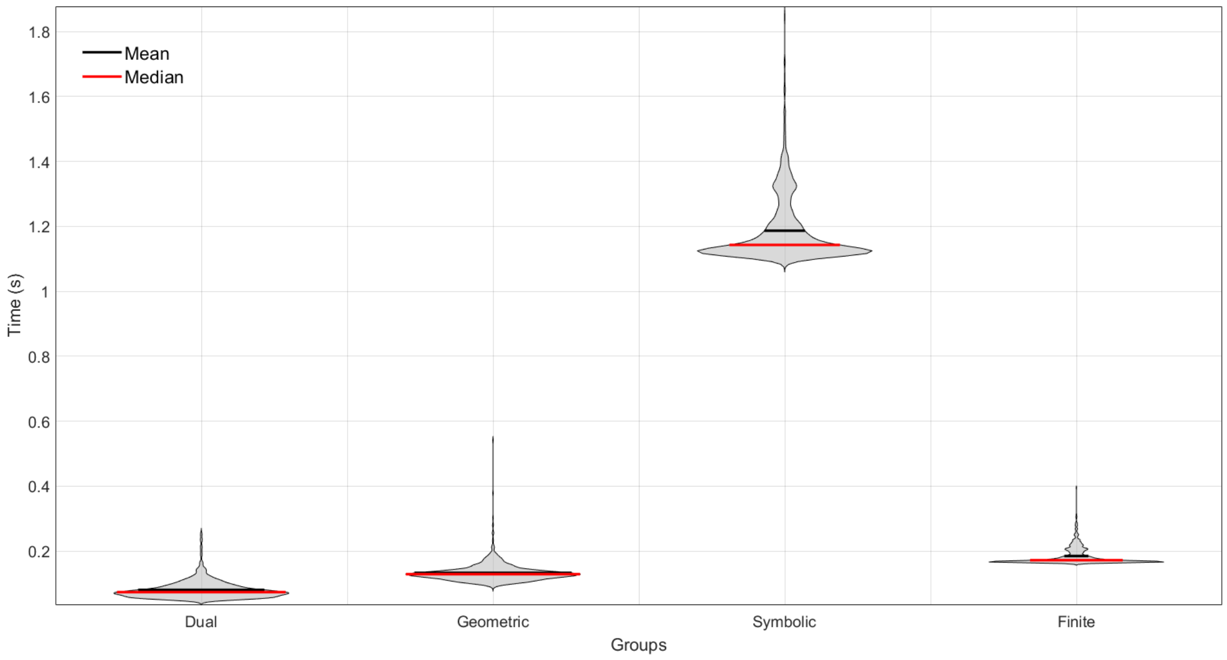

Figure 2. Violin plot of Jacobian matrix computation time for each compared method.

Table 3. Computation times for different methods.

Method	Computation Time (s)				
	Min	Max	Median	Mean	Standard deviation
Dual Numbers	**0.0487**	**0.2556**	**0.0736**	**0.0806**	0.0279
Geometric	0.0889	0.5407	0.1289	0.1328	0.0281
Symbolic	1.0920	1.8438	1.1428	1.1865	0.0983
Finite differences	0.1635	0.3944	0.1718	0.1846	**0.0270**

were used for the basis of this comparison. So, what can be seen in this figure is that both the method that uses dual numbers and the geometric method are those that have approximately an exponential degree of error of -16 with respect to the symbolic values. Unlike the finite difference method, whose error has an exponential degree of -6 approximately.

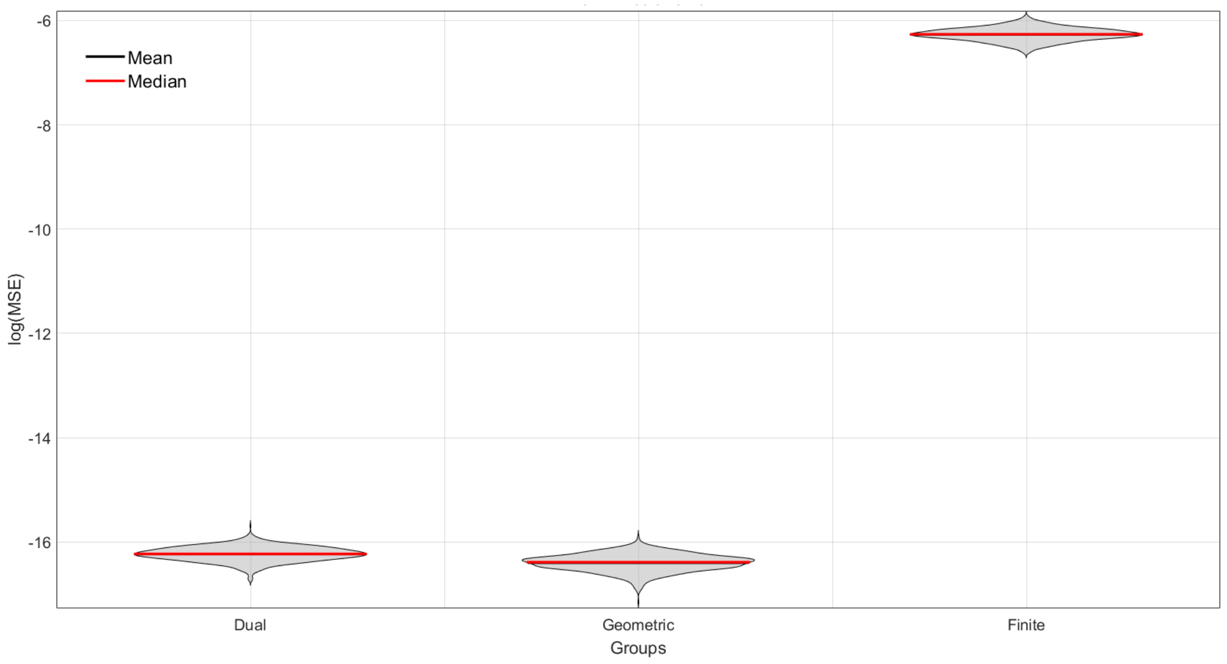

Figure 3. Violin plot of the logarithm of the MSE error relative to the results of the symbolic calculation.

To better clarify what is visualized in the previous figure, Table 4 shows the statistics of the error data. Where it is evident that the geometric method is the one that has the lowest values in all the metrics in the table, with the exception of the standard deviation. However, it is also possible to show that the values of the logarithm of the MSE for the method that uses dual numbers are very close to those of the geometric method, so both have close performances.

Table 4. Logarithm of the Error for Different Methods.

Method	Logarithm of the MSE				
	Min	Max	Median	Mean	Standard deviation
Dual Numbers	-16.727	-15.690	-16.228	-16.229	0.139
Geometric	**-17.150**	**-15.893**	**-16.387**	**-16.397**	0.164
Finite differences	-6.630	-5.910	-6.266	-6.269	**0.124**

4. Conclusions

The findings of this study highlight the efficacy of the proposed method, employing an extension of ordinary dual numbers featuring a scalar real part and a vector dual part. This novel approach facilitates the automatic computation of the Jacobian matrix, streamlining the process of differential kinematics computation in the context of robotic kinematics.

A comprehensive comparison was conducted against well-established methods, namely the geometric, symbolic, and finite difference methods. Notably, the proposed method exhibited the most favorable statistics in terms of minimum, maximum, median, and mean computation times, underscoring its efficiency and the simplified nature of Jacobian matrix calculations.

In evaluating the mean square error (MSE) of the Jacobian matrices, the proposed method, leveraging extended dual numbers, demonstrated competitive performance when compared to the geometric method. Although the latter slightly surpassed the dual numbers approach, the essence lies in their comparable effectiveness.

It is imperative to acknowledge that, despite the advantages offered by the extended dual numbers method for automated Jacobian matrix calculations, certain limitations were observed in its current implementation. The methodology was executed within a high-level programming environment like MATLAB, utilizing Object-Oriented Programming (OOP). It is crucial to recognize that the full potential of this method may not be fully realized until implemented in a lower-level programming language, thereby warranting further exploration and optimization.

Funding: This research received no external funding.

Author contributions: Writing & Editing, L. Orbegoso; Validation, L. Orbegoso; Methodology, L. Orbegoso; Software, L. Orbegoso; Investigation, E. Valverde; Conceptualization, E. Valverde; Analysis, E. Valverde; Review, E. Valverde.

Disclosure statement: The authors declare no conflict of interest.

References

[1] V. Brodsky and M. Shoham. Dual numbers representation of rigid body dynamics. *Mechanism and Machine Theory*, 34(5):693–718, 1999.

[2] Clifford. Preliminary sketch of biquaternions. *Proceedings of The London Mathematical Society*, pages 381–395, 1871.

[3] Neil T Dantam. Robust and efficient forward, differential, and inverse kinematics using dual quaternions. *The International Journal of Robotics Research*, 40(10-11):1087–1105, 2021.

[4] You-Liang Gu and J. Luh. Dual-number transformation and its applications to robotics. *IEEE Journal on Robotics and Automation*, 3(6):615–623, 1987.

[5] Dan Piponi. Automatic differentiation, c++ templates, and photogrammetry. *Journal of Graphics Tools*, 9, 01 2004.

[6] Hannes Sommer, Cédric Pradalier, and Paul Timothy Furgale. Automatic differentiation on differentiable manifolds as a tool for robotics. In *International Symposium of Robotics Research*, 2013.

[7] Avraham Cohen and Moshe Shoham. Application of hyper-dual numbers to multi-body kinematics. *Journal of Mechanisms and Robotics*, 8, 05 2015.

[8] Avraham Cohen and Moshe Shoham. Application of hyper-dual numbers to rigid bodies equations of motion. *Mechanism and Machine Theory*, 111:76–84, 2017.

[9] David González Sánchez. Dual numbers and automatic differentiation to efficiently compute velocities and accelerations. *Acta Applicandae Mathematicae*, July 2020.

[10] David Eager, Ann-Marie Pendrill, and Nina Reistad. Beyond velocity and acceleration: jerk, snap and higher derivatives. *European Journal of Physics*, 37(6):065008, oct 2016.

[11] A. Espinosa-Romero R. Peón-Escalante and F. Peñuñuri. Higher order kinematic formulas and its numerical computation employing dual numbers. *Mechanics Based Design of Structures and Machines*, 0(0):1–16, 2023.

[12] Jan Brinker, Michael Lorenz, Sami Charaf Eddine, and Burkhard Corves. Analytical derivation and application of the jacobian matrix of parallel kinematic manipulators. 11 2015.

[13] Jessica Villalobos, Irma Y. Sanchez, and Fernando Martell. Singularity analysis and complete methods to compute the inverse kinematics for a 6-dof ur/tm-type robot. *Robotics*, 11(6), 2022.

[14] Jesse Haviland and Peter Corke. A systematic approach to computing the manipulator jacobian and hessian using the elementary transform sequence. *ArXiv*, abs/2010.08696, 2020.

[15] Avantsa V.S.S. Somasundar and G. Yedukondalu. Robotic path planning and simulation by jacobian inverse for industrial applications. *Procedia Computer Science*, 133:338–347, 2018. International Conference on Robotics and Smart Manufacturing (RoSMa2018).

[16] W. Kandasamy and Florentin Smarandache. *Dual Numbers*. 01 2014.

[17] Nicolas Behr, Giuseppe Dattoli, Ambra Lattanzi, and Silvia Licciardi. Dual Numbers and Operational Umbral Methods. *Axioms*, 8(3):77, 7 2019.

[18] Philipp Rehner and Gernot Bauer. Application of Generalized (Hyper-) Dual Numbers in Equation of State Modeling. *Frontiers in chemical engineering*, 3, 10 2021.

[19] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st edition, 2008.

[20] Balaguer C. Barrientos A., Peñín F. and Aracil R. *Fundamentos de Robótica*. McGraw Hill, 2007.

[21] Soeren Laue. On the Equivalence of Automatic and Symbolic Differentiation. *arXiv (Cornell University)*, 1 2019.

[22] Vasily E. Tarasov. Exact Finite-Difference Calculus: Beyond Set of Entire Functions. *Mathematics*, 12(7):972, 3 2024.

[23] Třešňák Adam. *Forces Acting on the Robot during Grinding*. České vysoké učení technické v Praze, 2017.