JAIC
Journal of Applied Instrumentation and Control

Análise da Eficiência Computacional para Cinemática Direta e Inversa de Manipuladores Robóticos Utilizando Matrizes de Transformação Homogêneas e Quatérnios Duais

Tames Fernandes Mariano e Eduardo José Lima II

***Resumo*— O presente trabalho propõe um estudo detalhado da implementação de uma ferramenta matemática denominada quatérnios duais e uma análise à fundo sobre sua evolução e potencial aplicação dessa ferramenta na área da robótica. Em paralelo à essa visão é apresentado um comparativo com a forma tradicional de calcular rotação e translação na cinemática de robôs manipuladores. Os quatérnios duais têm sido massivamente empregados na robótica devido ao fato de serem computacionalmente mais eficientes na representação de informações rotacionais do que a representação com matrizes de transformação homogêneas. Assim, este trabalho tem como objetivo fornecer uma explicação detalhada através de um passo-a-passo para o uso da álgebra quaterniônica, de forma simplificada, utilizou-se de exemplos para melhor compreensão da implementação em questão. Embora haja uma grande quantidade de literatura sobre os aspectos teóricos de quatérnios duais, em poucas delas existem exemplos práticos de como realmente funciona o seu uso. Assim, ao dar uma clara noção da introdução à teoria de quatérnios duais, este documento também demonstra sua aplicação a um manipulador robótico do tipo serial com 4 GDL. Nesta dissertação foi possível fazer uma comparação entre a cinemática direta e inversa calculada usando a álgebra de matrizes versus a álgebra quaterniônica, verificou-se que os quatérnios são computacionalmente mais eficientes embora não aloquem menor área da memória de programa para o microcontrolador Atmel Atmega 328/P. Foram realizadas simulações e testes experimentais para comprovar os resultados.**

***Palavras Chaves*— Quatérnios Duais Unitários, Matrizes de Transformações Homogêneas, Cinemática Direta, Cinemática Inversa**

I. INTRODUÇÃO

OS quatérnios simples são uma representação de rotações 3D com diversas vantagens em comparação com matrizes de rotação. Contudo, os objetos rígidos não só giram, como também realizam translação. A rotação quando associada à translação é chamada de transformação rígida, qualquer deslocamento de um objeto rígido no espaço 3D pode ser descrito por uma transformação rígida. Os quatérnios duais são tidos como uma melhor representação de transformações rígidas por tratarem componentes de rotação e translação de forma independente, desta forma ele unifica a translação e a rotação em uma única variável de estado. Esta variável de estado única oferece uma forma robusta, inequívoca e computacionalmente eficiente para representar transformação rígida. Quando os movimentos se tornam mais complexos em termos de memória computacional usada, problemas passam a empregar os sistemas robóticos e os quatérnios duais são os mais indicados para solucionar tais problemas.

Levar em consideração complexidade temporal e espacial de um algoritmo é muito importante em programação. Afinal, é necessário saber se o algoritmo levará microssegundos, horas, dias ou centenas de anos para apresentar uma solução. Também é preciso saber qual é a quantidade de memória que um

Este trabalho contém resultados da Dissertação de Mestrado 2018 da discente Tames Fernandes Mariano e está vinculado ao Programa de Pós-Graduação Engenharia Mecânica (PPGMEC) da Universidade Federal de Minas Gerais (UFMG).

T. Fernandes Mariano, possui graduação e mestrado em Engenharia Mecânica pela Universidade Federal de Minas Gerais. (e-mail: tamesfernandes@gmail.com).

E. J. Lima II, possui graduação em Engenharia Mecânica Ênfase Mecatrônica pela Pontifícia Universidade Católica de Minas Gerais, mestrado em Engenharia Elétrica pela Universidade Federal da Bahia e doutorado em Engenharia Mecânica pela Universidade Federal de Minas Gerais (2006). Atualmente é Professor Adjunto da Universidade Federal de Minas Gerais, atuando nas áreas de mecatrônica, controle, automação e robótica. (e-mail: eduardo@demec.ufmg.br).

algoritmo requer para produzir uma solução. Enfim, a análise de algoritmos serve para classificar algoritmos como adequados ou menos adequados para solução de problemas. Assim, considera-se um algoritmo eficiente quando não se conhece nenhum outro funcionalmente equivalente a ele ou que possua menor complexidade.

Este trabalho se propõe a utilizar os Quatérnios Duais Unitários (QDU) como uma alternativa para cálculo da cinemática direta e inversa, ao mesmo tempo que se pretende comparar os resultados de testes de eficiência de execução de algoritmos obtidas com os resultados devido ao uso das tradicionais Matrizes de Transformação Homogênea (MTH).

Uma transformação 3D completamente rígida pode ser representada por uma componente translacional e outra rotacional através de uma matriz 4x4 homogênea, nota-se que as MTHs possuem a limitação de serem difíceis para realizar a interpolação entre as transformações. Alternativamente, surgem os Quatérnios Duais como a melhor solução para rotações, neles as transformações podem ser gerenciadas com a vantagem de encapsular a translação e a rotação em um único estado que pode ser interpolado sem esforço utilizando oito números escalares ao invés de dezesseis como nas MTHs. Nota-se que os QDUs permitem fácil a incorporação à sistemas existentes, isso porque uma implementação com eles consegue ir de encontro às implementações com MTHs realizando poucas interferências nas montagens existentes.

A importância deste trabalho resume-se na otimização do cálculo de tarefas com manipuladores robóticos, resultando em tarefas operacionais mais ágeis já que o desempenho computacional é o foco do trabalho, bem como a simulação, modelagem e implementação de sistemas para a verificação da maneira mais eficiente de execução de algoritmos de cinemática direta e inversa.

II. Conceituando a Cinemática de Robôs Manipuladores

A classificação de robôs quanto à sua arquitetura e estrutura é fundamental para identificar as diferenças entre eles que influenciarão nos cálculos de cinemática e controle. A arquitetura de um robô se refere à maneira que os respectivos elos e juntas se conectam, já a estrutura de um robô considera o posicionamento dos eixos das juntas para formação da cadeia cinemática do robô. De maneira genérica, um robô serial é aquele cuja cadeia cinemática é aberta parecida com um braço antropomórfico, esse tipo de robô é formado por uma única cadeia cinemática onde apenas um elo está conectado à base, o final da cadeia cinemática é livre para se mover no espaço. Um robô paralelo é formado por no mínimo duas cadeias cinemáticas independentes, as várias cadeias cinemáticas se ligam a um outro elemento fixo que constituirá a base do efetuador(mecanismo de cadeia fechada tem o efetuador ligado à uma base fixa),esses robôs resultam na maioria das vezes em sistemas redundantes com mais atuadores que o número de graus de liberdade controlados pelo efetuador [7][20].

Em [13] os autores explicam que uma modelagem sistemática de robôs requer um método apropriado para a descrição da morfologia deste. Vários métodos e notações foram propostos e o mais utilizado é o de Denavit-Hartenberg (DH), desenvolvido para estruturas abertas simples de robôs seriais, como a mostrada na Fig.1–(a). Este método DH apresenta ambiguidades quando é aplicado a robôs com cadeias cinemáticas fechadas ou robôs estruturados em árvore, Fig. 1 – (b) e (c). Quando o modelo DH convencional não é aplicável, uma alternativa é a notação de Khalil e Kleinfinger (1986) que é um modelo que prevê modificações no modelo DH original e que permite a descrição unificada de robôs complexos e com estruturas mecânicas articuladas, além da tradicional descrição de robôs seriais.

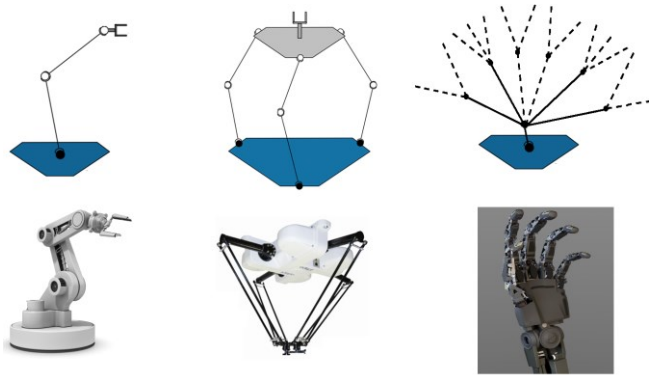

Fig.1 Estruturas de Robôs (a) Braço robótico; (b)Robô Paralelo; (c) Robô modelo Dexterous [9] [25] [39]

Em análise à robôs seriais de estruturas abertas simples, [32] explica a convenção de Denavit Hartenberg. Esta convenção precisou existir considerando que os sistemas de coordenadas de cada junta do manipulador não poderiam ser definidos arbitrariamente, por uma questão de consistência e eficiência computacional o modelo adotado permitiu a localização destes sistemas sem gerar ambiguidades.

O autor [34] ao explicar o modelo Denavit-Hartenberg (DH), afirma que não é possível representar qualquer matriz de transformação homogênea usando somente quatro parâmetros (θ, d, a, α), desta forma para garantir somente as transformações homogêneas passíveis de serem expressas, duas outras proposições devem ser atendidas no modelo DH juntamente com os quatro parâmetros de modo que permita especificar qualquer transformação homogênea resultante no manipulador robótico : (1) o eixo x da junta posterior deve ser perpendicular ao eixo z da junta anterior, de modo a garantir que o produto escalar destes dois eixos seja nulo e portanto os ângulos θ e α existam; (2) o eixo x da junta posterior deve interceptar o eixo z da junta anterior, garantindo que o deslocamento de um elo para o outro resulte em uma equação que possa ser expressa pela combinação linear dos dois eixos.

O Sistema nsa definido por convenção para o efetuador está ilustrado na Figura 2, onde o eixo z representa a direção de ataque ou direção para a qual o efetuador está apontando (“approach”) , o eixo y é a direção de escorregamento para abertura e fechamento da garra/ferramenta (“sliding”) e o eixo x é direção normal que é ortogonal ao plano formado pelas direções z e y (approach e sliding) saindo pela regra da mão

direita. Esta convenção é também conhecida como movimentos de rolagem, arfagem e guinada (roll, pitch e yaw – z, y e x) ou ainda pode ser chamada X-Y-Z ângulos fixos. [11][28][30].

Um robô cuja configuração é conhecida, ou seja, o comprimento de todos os elos e ângulos articulares do robô foram medidos e determinados. O cálculo da posição e da orientação do efetuador do robô é chamado de análise cinemática direta. Em outras palavras, se todas as variáveis articulares do robô são dadas, então, com o uso das equações de cinemática direta é possível calcular onde terminal final do robô está em qualquer instante de tempo [23].

As coordenadas finais podem ser representadas pelo vetor de posição d, onde n,s,a são vetores ortonormais que descrevem a orientação do efetuador em relação a base [29]. Estes vetores formam a matriz de rotação R. [35].

Para realizar o comando de um manipulador é necessário o modelo inverso, nele é dado a posição e orientação desejadas para o efetuador, deseja-se saber os valores de θi em cada junta. A cinemática inversa é muito parecida com a equação cinemática direta, porém com as entradas e saídas invertidas. O problema da cinemática inversa de posição pode ser bastante difícil de se resolver pois resulta em doze equações não lineares e n incógnitas [23] [35].

A. Matrizes de Transformação Homogêneas (MTH)

Por definição, uma transformação é a realização de um movimento no espaço. Uma transformação é uma mudança no estado de um referencial (localização e orientação) e, portanto, um vetor, um objeto, ou um referencial móvel se movendo no espaço em relação a um sistema de referência fixo poderá ser representado como um referencial. A transformação pode se apresentar como uma das seguintes formas: uma translação pura, uma rotação pura em torno de um eixo ou ainda uma combinação de translações e/ ou rotações. A movimentação de um corpo rígido no espaço pode ser modelada como uma transformação linear representada por matrizes de transformação. A matriz afim consiste na combinação de um, ou de mais tipos de transformações lineares. Busca-se então uma representação para a translação e rotação de um objeto já que esses são os tipos de movimentação possíveis em um robô[23]:

$$\boldsymbol{H} = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Na representação padrão cada uma das matrizes pode ser obtidas por meio de quatro transformações homogêneas básicas, de acordo com a estrutura rígida do robô manipulador resultando em [35]:

$$\boldsymbol{H}_{DH_i}^{i-1} = \boldsymbol{R}_{z,\theta} * \boldsymbol{T}_{z,d} * \boldsymbol{T}_{x,\mathrm{a}} * \boldsymbol{R}_{x,\alpha}$$
$$= \begin{bmatrix} C_{\theta_i} & -S_{\theta_i} & 0 & 0 \\ S_{\theta_i} & C_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} * \tag{2}$$
$$* \begin{bmatrix} 1 & 0 & 0 & \alpha_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_{\alpha_i} & -S_{\alpha_i} & 0 \\ 0 & S_{\alpha_i} & C_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C_{\theta_i} = \cos(\theta_i) \qquad S_{\theta_i} = \mathrm{sen}(\theta_i)$$
$$C_{\alpha_i} = \cos(\alpha_i) \qquad S_{\alpha_i} = \mathrm{sen}(\alpha_i)$$

A matriz global de rotação nsa, também chamada de matriz global *roll, pitch* e *yaw* é dada:

$$R_{ZYX} = R_{Z_\omega} * R_{Y_\emptyset} * R_{X_\gamma}$$

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$
$$= \begin{bmatrix} (C_\omega\, C_\emptyset) & (C_\omega\, S_\emptyset\, S_\gamma - S_\omega C_\gamma) & (C_\omega\, S_\emptyset\, C_\gamma + S_\omega S_\gamma) \\ (S_\omega\, C_\emptyset) & (S_\omega\, S_\emptyset\, S_\gamma + C_\omega C_\gamma) & (S_\omega\, S_\emptyset\, C_\gamma - C_\omega S_\gamma) \\ -S_\emptyset & (\, C_\emptyset\, S_\gamma) & (\, C_\emptyset\, C_\gamma) \end{bmatrix} \tag{3}$$

Os ângulos extraídos da matriz de rotação nsa são dados por:

$$\emptyset_{\mathrm{rot.Y}} = \mathrm{atan2}\left(\frac{-r_{31}}{\sqrt{{r_{11}}^2 + {r_{21}}^2}}\right) \tag{4}$$

$$\omega_{\mathrm{rot.Z}} = \mathrm{atan2}\left(\frac{r_{21}/C_\emptyset}{r_{11}/C_\emptyset}\right) = \mathrm{atan2}\left(\frac{r_{21}}{r_{11}}\right) \tag{5}$$

$$\gamma_{\mathrm{rot.X}} = \mathrm{atan2}\left(\frac{r_{32}/C_\emptyset}{r_{33}/C_\emptyset}\right) = \mathrm{atan2}\left(\frac{r_{32}}{r_{33}}\right) \tag{6}$$

A inversa de uma Matriz de Transformação Homogênea (MTH) é dada:

$$\boldsymbol{H^{-1}} = \begin{bmatrix} n_x & n_y & n_z & (-n_x p_x - n_y p_y - n_z p_z) \\ s_x & s_y & s_z & (\, -s_x p_x - s_y p_y - s_z p_z) \\ a_x & a_y & a_z & (-a_x p_x - a_y p_y - a_z p_z) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

B. Quatérnios Duais Unitários (QDU)

Uma transformação 3D completamente rígida é composta de uma componente translacional e outra rotacional, que é tradicionalmente calculada através de uma matriz 4x4 homogênea. No entanto, as matrizes homogêneas possuem a limitação de ser difícil realizar a interpolação entre as transformações. Alternativamente, surgem os quatérnios duais como a melhor solução para rotações, neles as transformações podem ser gerenciadas com a vantagem de encapsular a translação e a rotação em um único estado que pode ser interpolado sem esforço [31].

A comunidade de roboticistas tem dado cada vez mais atenção especial aos QDUs tanto para modelagem cinemática para robôs com n graus de liberdade quanto para estratégias de controle cinemático, isso porque a economia de memória e a eficiência computacional dos QDUs foram destacadas como superiores em relação às MTHs [38] [19].

Os QDUs já foram empregados de forma efetiva na

computação gráfica, em visões computadorizadas, na navegação e etc. As operações com QDs destacam-se também pela maior robustez à erros numéricos [24]. A falta de robustez nos cálculos é ruim por resultar em problemas de precisão numérica (ou estabilidade numérica) que surgem devido à aritmética inexata do computador, pois os números adotados nos cálculos são normalmente os números de ponto flutuante de precisão fixa ou até mesmo inteiros, após realizar vários cálculos usando o computador nota-se que ao invés dos números reais exatos esperados tem-se em teoria precisões arbitrárias, levando à respostas incorretas[22]

A comunidade de roboticistas tem dado cada vez mais atenção especial aos QDUs tanto para modelagem cinemática quanto para propostas de controle, isso porque a economia de memória e a eficiência computacional dos QDUs apresentam superioridade em relação às MTHs [19][38].

Dentre as vantagens dos quatérnios duais [17][21][24][30]:

- Combinam rotação e translação em uma única variável de estado;
- São uma representação compacta (8 números escalares);
- Eles são facilmente convertidos em outras formas (por exemplo, forma matricial);
- Podem ser facilmente interpolados sem ambiguidade;
- Computacionalmente mais eficientes (quando comparado a matrizes de transformação homogênea);
- Podem ser incorporados em um sistema já implementado sem fazer muita interferência;
- Apresentam uma forma única de representação para coordenadas de transformação rígida;
- Ausência de singularidades ao serem representados no espaço Euclidiano;
- Melhor robustez devido à erros numéricos;
- Melhor performance na modelagem cinemática de braços robóticos com n graus de liberdade e, também, no controle proporcional.

Os quatérnios simples, segundo [15], tem sido uma ferramenta popular na computação gráfica 3D há mais de 20 anos, são quatro termos contendo números reais (q_r, q_x, q_y, q_z), dos quais os três termos (q_x, q_y, q_z) são componentes de um vetor. Os quatérnios podem ser representados :

$$Q = q_r + q_x\,\vec{\boldsymbol{\imath}} + q_y\,\vec{\boldsymbol{\jmath}} + q_z\,\vec{\boldsymbol{k}} \tag{8}$$

ou

$$Q = q_r + \vec{\boldsymbol{q}}$$

Onde $\vec{\boldsymbol{\imath}}$, $\vec{\boldsymbol{\jmath}}$ e $\vec{\boldsymbol{k}}$ são vetores unitários associados com os eixos do sistema de coordenadas cartesianas do vetor $\vec{\boldsymbol{q}}$ e q_r representa a parte real do quatérnio.

Os quatérnios clássicos são restritos para a representação somente de rotações, no entanto, em aplicações gráficas normalmente trabalha-se com rotação composta com a translação, isto é, transformações rígidas [15].

Um quatérnio dual pode ser utilizado para definir um corpo rígido rotacionando um ângulo φ em torno de um eixo $\vec{\boldsymbol{u}}$ que passa pela origem conforme:

$$Q = \cos\left(\frac{\varphi}{2}\right) + \vec{\boldsymbol{u}}\,\sin\left(\frac{\varphi}{2}\right) \tag{9}$$

$$q_1 = \cos\left(\frac{\varphi}{2}\right) \tag{10}$$

$$q_2 = \boldsymbol{i}\,\sin\left(\frac{\varphi}{2}\right) \tag{11}$$

$$q_3 = \boldsymbol{j}\,\sin\left(\frac{\varphi}{2}\right) \tag{12}$$

$$q_4 = \boldsymbol{k}\,\sin\left(\frac{\varphi}{2}\right) \tag{13}$$

Os quatérnios duais são entidades matemáticas cujas componentes são números duplos, e eles podem ser expressos por:

$$\underline{Q} = Q + \boldsymbol{\varepsilon}\,Q_0 \tag{14}$$

Onde, tanto Q quanto Q_0 são quatérnios simples e $\boldsymbol{\varepsilon}$ é a unidade dual. Um quatérnio dual pode formular um problema de forma mais concisa, resolvê-lo mais rapidamente e em menos etapas, apresentar o resultado mais claramente para os outros, ser posto em prática com menos linhas de código e depurado sem esforço.

Outra representação dos quatérnios duais é dada por:

$$\underline{Q} = \begin{bmatrix} Q_{P_0} \\ Q_{P_i} \\ Q_{P_j} \\ Q_{P_k} \\ Q_{D_0} \\ Q_{D_i} \\ Q_{D_j} \\ Q_{D_k} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} \tag{15}$$

Os quatérnios duais representam transformações rígidas da mesma maneira como quatérnios clássicos representam rotações. Os quatérnios duais associados a algoritmos puderam ser aplicados em combinações de movimento, análises de movimento, enquadramento espacial chave, visão por computador e hardware gráfico [15].

Em [2] é mostrada a cinemática em manipuladores robóticos utilizando quatérnios duais para proceder uma sequência de multiplicações que podem ser empregadas em conjunto com os parâmetros de Denavit-Hartenberg de modo a encontrar o modelo cinemático direto. Fazendo uso dos quatérnios duais, onde o símbolo '■' representa a multiplicação de dois quatérnios duais, a equação resultante é dada por:

$$\underline{\boldsymbol{Q}}_{DH_i}^{\,i-1} = \underline{\boldsymbol{R}}_{z,\theta} \blacksquare \underline{\boldsymbol{T}}_{z,d} \blacksquare \underline{\boldsymbol{T}}_{x,\mathrm{a}} \blacksquare \underline{\boldsymbol{R}}_{x,\alpha} \tag{16}$$

$$\underline{\boldsymbol{Q}}_{DH_i}^{\,i-1} = \begin{bmatrix} \cos(\theta_i/2) \\ 0 \\ 0 \\ \mathrm{sen}(\theta_i/2) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \blacksquare \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ d_i/2 \end{bmatrix} \blacksquare \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ \mathbf{a}_i/2 \\ 0 \\ 0 \end{bmatrix} \blacksquare \begin{bmatrix} \cos(\alpha_i/2) \\ \mathrm{sen}(\alpha_i/2) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Os quatérnios duais tiram vantagem em relação às matrizes homogêneas quando se trata da memória ocupada por estes, uma vez que as matrizes homogêneas exigem doze números para representar seis graus de liberdade enquanto que os quatérnios duais exigem apenas oito [26].

A representação para a translação de um quatérnio dual pode ser dada por:

$$transl\left(\underline{Q}\right) = 2 * \underline{Q}_D \blacksquare \underline{Q}^* \tag{17}$$

$$transl\left(\underline{Q}\right) = 2 * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} \blacksquare \begin{bmatrix} q_1 \\ -q_2 \\ -q_3 \\ -q_4 \\ q_5 \\ -q_6 \\ -q_7 \\ -q_8 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ x_E \\ y_E \\ z_E \end{bmatrix}$$

$$p = \begin{bmatrix} x_E \\ y_E \\ z_E \end{bmatrix} = 2 \begin{bmatrix} -q_2 & +q_1 & -q_4 & +q_3 \\ -q_3 & +q_4 & +q_1 & -q_2 \\ -q_4 & -q_3 & +q_2 & +q_1 \end{bmatrix} \begin{bmatrix} q_5 \\ q_6 \\ q_7 \\ q_8 \end{bmatrix} \tag{18}$$

Em notação quaterniônica, o sistema nsa ou sistema roll, pitch e yaw possui equivalência em relação às MTHs. Os parâmetros de Euler conhecidos para os quatérnios unitários e de onde podem ser extraídas as rotações em torno dos eixos Y, Z, X possui matriz de rotação equivalente dada por:

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \tag{19}$$

$$\begin{bmatrix} (1-2{q_3}^2-2{q_4}^2) & 2(q_2q_3-q_1q_4) & 2(q_2q_4+q_1q_3) \\ 2(q_2q_3+q_1q_4) & (1-2{q_2}^2-2{q_4}^2) & 2(q_3q_4-q_1q_2) \\ 2(q_2q_4-q_1q_3) & 2(q_3q_4+q_1q_2) & (1-2{q_2}^2-2{q_3}^2) \end{bmatrix}$$

A inversa de um Quatérnio Dual (QD) é dada:

$$\underline{\boldsymbol{Q}}^{-1} = \frac{\underline{\boldsymbol{Q}}^*}{\left\|\underline{\boldsymbol{Q}}\right\|^2} \tag{20}$$

$$\underline{Q}^{-1} = \begin{bmatrix} q_1 \\ -q_2 \\ -q_3 \\ -q_4 \\ q_5 \\ -q_6 \\ -q_7 \\ -q_8 \end{bmatrix}$$

III. ANÁLISE DE DESEMPENHO COMPUTACIONAL

Dados dois algoritmos diferentes para resolver o mesmo problema, deseja-se escolher qual desses algoritmos é o melhor, para isso deve-se pensar em termos de eficiência (ou custo computacional), ou seja, aquele algoritmo que consumir menos recursos para realizar uma mesma tarefa é o mais eficiente. Em geral, a análise do custo computacional é feita de duas formas: em termos de tempo e de espaço ocupado. Quando se trata da eficiência espacial ou eficiência da memória, deseja-se a informação de quanta memória está sendo utilizada pelo algoritmo para armazenar os dados, sejam matrizes, vetores ou escalares e, também, informação em relação à hierarquia das memórias acessadas (registros, cache, memória principal, armazenamento em disco). Quando se trata de eficiência temporal, a intenção é saber quanto tempo um algoritmo leva para realizar determinada tarefa. É razoável de se pensar que o tempo vai ser proporcional ao número de operações de ponto flutuante ("flop") feito pelo algoritmo, observa-se que o tempo total não depende apenas disso, mas também de outros fatores como memória, taxas de transferências de dados da memória para o CPU, redes, etc. A contagem do número de operações ("flop") para realizar determinada tarefa é uma maneira de verificar a eficiência computacional. É importante salientar que a eficiência temporal e a eficiência espacial não são independentes, uma leva à outra já que a habilidade de um algoritmo acessar uma memória influencia diretamente no tempo de execução do algoritmo [36].

A. Multiplicações, Adições e Funções trigonométricas

Em [2] afirma-se que o custo de cálculo de transformações rígidas usando matrizes de transformação homogênea é maior que o custo de utilizar quatérnios duais, dos resultados das análises deste autor decorrem-se o custo de cálculo da multiplicação de m MTHs levando em conta o custo individual do cálculo de cada MTH e também multiplicações entre elas, dado por:

$$custo_{TOTAL}\left(\boldsymbol{H}_{DH}{}_1^0 * \boldsymbol{H}_{DH}{}_2^1 \cdots \boldsymbol{H}_{DH}{}_m^{m-1}\right) = \{func.trig., mult., adições\}$$

$$custo_{TOTAL}\left(\boldsymbol{H}_{DH}{}_m^0\right) = \{4m, 70m-64, 48m-48\} \tag{21}$$

Em comparação às MTHs, [2] também revela que o custo de cálculo da multiplicação de m QDUs que leva em conta o custo individual do cálculo de cada QDU e também a multiplicações entre eles:

$$custo_{TOTAL}\left(\underline{\boldsymbol{Q}}_{DH}{}_1^0 \blacksquare \underline{\boldsymbol{Q}}_{DH}{}_2^1 \cdots \underline{\boldsymbol{Q}}_{DH}{}_m^{m-1}\right) = \{func.trig., mult., adições\}$$

$$custo_{TOTAL}\left(\underline{\boldsymbol{Q}}_{DH}{}_m^0\right) = \{4m, 60m-48, 44m-40\} \tag{22}$$

A comparação entre as equações (21) e (22) foi mostrada no Gráfico 1, nele é evidenciado que à medida que o número de graus de liberdade do robô cresce (m cresce) os Quatérnios Duais executarão menos operações (multiplicação e somas ou subtrações) do que as Matrizes de Transformação Homogêneas:

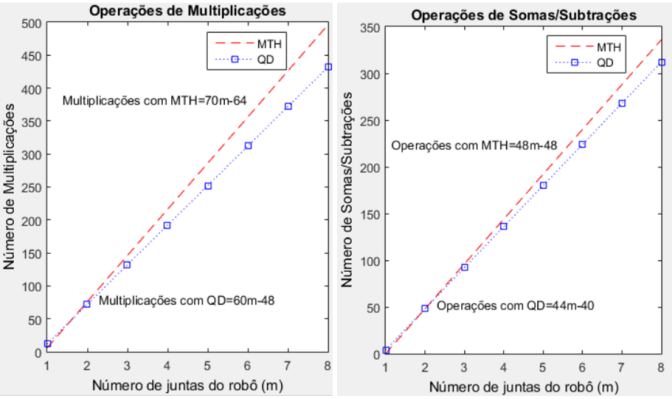

Gráfico 1: Comparação entre QDU e MTH operações de multiplicação e somas ou subtrações

B. *Número de Flop*

O custo computacional de resolver um sistema linear é, muitas vezes, determinado pelo tempo que um computador leva para executar os cálculos. Em geral, o tempo de computação depende de dois fatores: a velocidade do processador e o número de operações exigidas pelo algoritmo [3][40].

Em [1] realizou-se uma comparação (termo em inglês 'benchmark') para mensurar a distribuição de memória em algoritmos, a forma fundamental para avaliar a performace foi o tempo para completar uma tarefa específica. Em sua definição de tempo, o autor afirma ter usado em seus experimentos uma medição de tempo externa (do inglês 'external clock') para contabilizar o tempo decorrido em cada operação, mas outros meios de contabilizar o tempo, segundo o autor, poderiam ser cuidadosamente analisados se fosse possível, claro, visualizar o tempo dedicado para uma tarefa específica. Segundo o autor, o difícil é a mediçao do tempo de CPU, pois, quando usado para medir o tempo de operação poderia variar dependendo do sistema operacional levando a uma comparação injusta dependendo do processador de cada computador. A quantidade aritmética para avaliar a performace é dada em FLOP ou MFLOP (10^6 FLOP), essa é uma unidade de contagem de tarefas executadas. É importante não confundir com 'Flop/s' ou 'MFlop/s' que é usado para medir a taxa de operações por segundo. De maneira simplificada, os custos de execução das operações mais comuns são dados por:

Tabela I - Comparação das operações mais comuns

Operação	Peso da Operação Ponto Flutuante
Adição, Subtração e Multiplicação	1 flop
Divisão e Raiz Quadrada	4 flop
Função exponencial e Funções trigonométricas	8 flop

C. *Comandos de Interrupção em Algoritmos*

Uma interrupção, de acordo com [8] , é um evento assíncrono, imprevisível e que pode ocorrer múltiplas vezes antes, durante ou após um ciclo de instrução de um programa, requerendo portanto imediata atenção. Na entrada e saída de uma interrupção, um dispositivo externo ou uma condição interna pode forçar a CPU a interromper a execução do programa principal temporariamente para que possa executar um outro programa conhecido como rotina de interrupção do serviço (*ISR- Interrupt Service Routine*), o programador escreve o comando da rotina de serviço de interrupção em outro endereço. Essa rotina satisfaz as necessidades do dispositivo externo ou da condição interna. Uma vez que a interrupção é reconhecida, o microcontrolador salva internamente o endereço de retorno e o conteúdo de alguns outros registros e ramifica-se automaticamente para um endereço pré-definido pelo fabricante. O microcontrolador geralmente deixa incompleta a instrução atual e com o endereço de retorno e conteúdo de certos registros internos, depois de completar a rotina de interrupção, retorna à instrução anterior para executar o controle no ponto de parada do programa principal. A interrupção é muito semelhante às sub-rotinas ou funções encontradas linguagem C [27].

D. *Análise de Complexidade de Algoritmos – Modelo RAM*

O modelo de Máquina Acesso aleatório (sigla RAM do inglês "Random Access Machine") caracteriza o tempo de execução como uma função do tamanho da entrada (n) no algoritmo. Muito cuidado para não confundir este modelo com outro termo da computação relacionado à memória RAM (sigla RAM também usada para se referir ao acesso aleatório à memória do inglês "Random Access Memory"). O modelo RAM em questão possui uma abordagem teórica que analisa uma descrição de alto nível do algoritmo, trata-se de um modelo de computação genérico que contém instruções encontradas em algoritmos reais cujos tempos são considerados constantes na maioria dos computadores, tais como instruções aritméticas (soma, subtração, resto, piso, etc), de movimentação de dados (carregar, armazenar, copiar), de controle (desvio condicional, chamada e retorno de funções). O modelo mede a eficiência de execução contando o número de passos de acordo com o tamanho da entrada no programa. O modelo parte do princípio de que independente da máquina usada (uso de um computador hipotético) é possível contar os passos ('step') de acesso aleatório ou RAM de acordo com a as instruções em cada linha do algoritmo, obedecendo as duas descrições à seguir [10] [33]:

- Cada acesso à memória leva exatamente um passo, ou seja, cada operação simples no algoritmo (+, *, -, =, if, call) é contabilizada uma única vez.
- Loops (for, while) e sub-rotinas não são considerados como operações simples. Em vez disso, eles são contabilizados como uma composição de muitas operações. Não faria sentido classificá-los como uma operação de etapa única, já que executar um loop 1.000.000 vezes certamente demoraria muito mais do que executá-lo 10 vezes. O tempo que leva para percorrer um loop ou executar um subprograma depende do número (n) de iterações do loop ou da natureza específica da sub-rotina.

IV. Microcontrolador, Memória e Compilação de Dados

Um microprocessador é um circuito integrado responsável pelo processamento de dados, é a unidade lógica que executa

operações aritméticas com diversos registradores especiais, para este componente funcionar existe a necessidade de outros dispositivos externos que contenham memória de leitura e escrita para então haver o armazenamento de dados e programas, no microprocessador os dados são apenas processados, necessitando de dispositivos periféricos que comuniquem com estes dados para o armazenamento permanente, a conversão, a interface, etc. Por outro lado, uma unidade de comando controlado (MCU) ou microcontrolador é considerado mais do que um microprocessador. Além de todas as funções de um microprocessador, o microcontrolador possui memória RAM, memória ROM, temporizadores, contadores, porta serial, conversores e portas de I/O em um só circuito integrado, ou seja, um microcomputador em um único chip [6][12][18].

Em [14], [16] e [37] explica-se os três tipos de memórias, suas características e funções dentro do microcontrolador principal (Atmel Atmega328/P [4]) da placa Arduino UNO [5] empregada no projeto como:

- Memória EEPROM é uma memória não volátil o que significa que as informações contidas nela continuarão gravadas mesmo quando o sistema for desligado. Os dados nela só podem ser lidos byte-a-byte, e por isso pode ser um pouco complicado usá-la. Essa memória pode sofrer modificações durante a execução do algoritmo, pois pode ser usada para leitura e também gravação durante a execução do programa. Essa memória é mais lenta que a memória SRAM.
- Memória Flash é aquela que possui a mesma tecnologia empregada em pen drives e cartões SD, é chamada pelo fabricante Atmel de memória de programa (*sketch*) e é usada para armazenar o algoritmo em si e quaisquer dos dados (variáveis) que serão utilizados no programa são gravados nesta memória, esta é uma memória não volátil e mesmo que haja uma queda de energia suas informações continuam inalteradas. O código do programa é executado a partir da memória flash e ela não sofre modificações durante a execução do algoritmo, a modificação desta memória requer alterações no código fonte e nova compilação do programa.
- Memória SRAM (do inglês "Static Random Access Memory") é a memória que pode ser modificada durante a execução do algoritmo, assim como a EEPROM é usada para leitura e escrita de dados a partir da execução do programa, a diferença é que se trata de uma memória volátil e os dados dessa memória são perdidos caso haja queda de energia. É uma memória de grande importância, pois é nela onde se passa todas as ações tais como: alocação estática e dinâmica de variáveis, ponteiros para chamadas de funções, etc. É nela que desenvolvedores concentram esforços de otimização, pois é nela onde acontecem a maioria dos problemas como a reinicialização indesejada (inglês "auto-reset") do microcontrolador e/ou dados corrompidos que exibem resultados imprecisos ou com erros grosseiros retornados pelo MCU.

V. METODOLOGIA

Foi construído um protótipo com 4GDL (Graus de Liberdade) e um desenho foi feito em escala real com três dimensões no software SolidWork para melhor acompanhar o movimento e espaço de trabalho do manipulador:

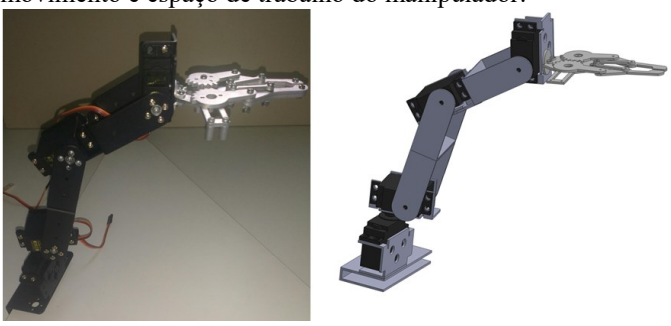

Fig. 2 Comparativo Robô 4GDL e Desenho em SolidWorks

Os cálculos das cinemáticas direta e inversa foram realizados para o manipulador de três maneiras distintas: por geometria, por matrizes de transformação homogênea e por meio dos quatérnios duais. Nos três casos, as equações da cinemática direta chegaram às formas compactas:

$$\boldsymbol{x_E} = \mathrm{a}_4 C_{234} C_1 + \mathrm{a}_3 C_1 C_{23} + \mathrm{a}_2 C_1 C_2 + d C_1 \quad (23)$$

$$\boldsymbol{y_E} = \mathrm{a}_4 C_{234} S_1 + \mathrm{a}_3 S_1 C_{23} + \mathrm{a}_2 S_1 C_2 + d S_1 \quad (24)$$

$$\boldsymbol{z_E} = \mathrm{a}_4 S_{234} + \mathrm{a}_3 S_{23} + \mathrm{a}_2 S_2 + d_1 \quad (25)$$

$$\emptyset_{\mathrm{rot.Y}} = -(\theta_2 + \theta_3 + \theta_4) \quad (26)$$

$$\omega_{\mathrm{rot.Z}} = \theta_1 \quad (27)$$

$$\gamma_{\mathrm{rot.X}} = 90° \quad (28)$$

A cinemática inversa, para os três casos, chegou às equações compactas:

$$\boldsymbol{\theta_1} = \arctan2\left(\frac{y_E}{x_E}\right) \quad (29)$$

$$\boldsymbol{\theta_3} = \arctan2\left(\frac{\pm\sqrt{1-\boldsymbol{\beta}^2}}{\boldsymbol{\beta}}\right) \quad (30)$$

$$\boldsymbol{\beta} = \frac{\left(z_E - d_1 + \mathrm{a}_4 S_\emptyset\right)^2 + \left(\sqrt{{x_E}^2 + {y_E}^2} - \mathrm{a}_4 C_\emptyset - d\right)^2 - {a_2}^2 - {a_3}^2}{2 * a_2 * a_3}$$

$$\boldsymbol{\theta_2} = \arctan2\left(\frac{z_E - d_1 + \mathrm{a}_4 \sin(\emptyset)}{\sqrt{{x_E}^2 + {y_E}^2} - \mathrm{a}_4 \cos(\emptyset) - d}\right) - \arctan2\left(\frac{a_3 \sin\theta_3}{a_2 + a_3 \cos\theta_3}\right) \quad (31)$$

$$\boldsymbol{\theta_4} = -\emptyset - \theta_2 - \theta_3 \quad (32)$$

Simplificações no número de operações foram executadas tais como substituições trigonométricas de ângulos conhecidos aliados às equações de identidades trigonométricas fundamentais. Os gráficos exibem a comparação entre MTHs e QDUs. O Gráfico 2 mostra a comparação entre cada uma das

transformações da cinemática direta do robô com 4GDL ($m = 4$), já o Gráfico 3 mostra as transformações da cinemática direta que envolvem multiplicação de matrizes e multiplicação quatérnios duais.

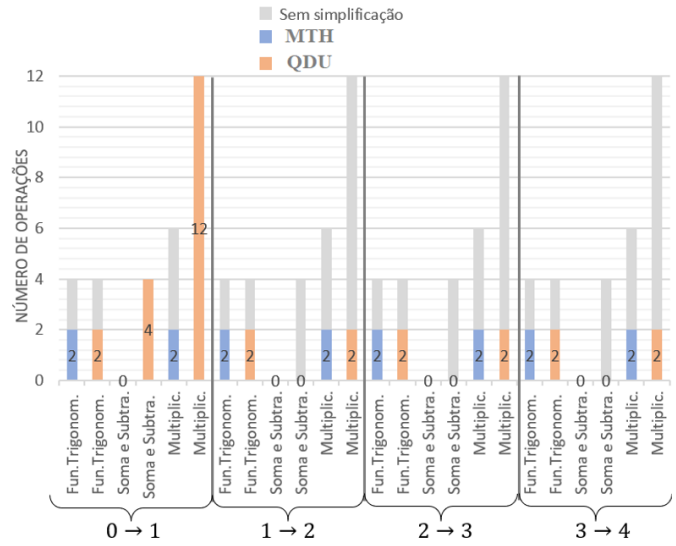

Gráfico 2. Comparação entre cada uma das transformações da cinemática direta do robô com 4GDL

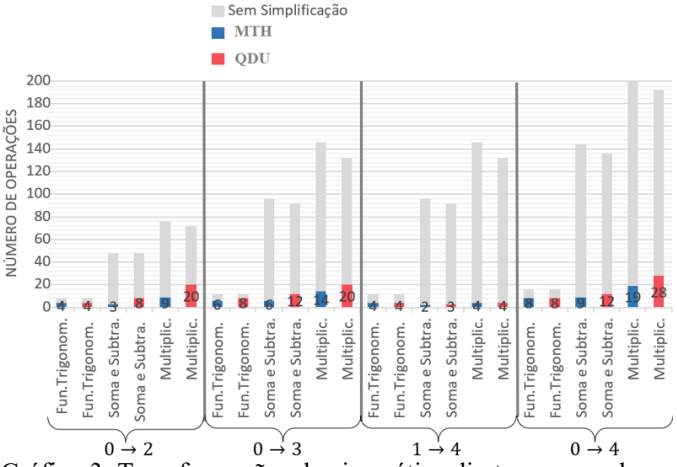

Gráfico 3. Transformações da cinemática direta que envolvem multiplicação de matrizes e multiplicação quatérnios duais

Um comparativo entre as operações foi realizado para o Microcontrolador Atmega 328/P (MCU), atribuiu-se a operação de menor custo (soma) como base de comparação com as demais operações. Os valores como foram comparados com a variável do tipo 'float' são por definição os números de flop de cada operação, os resultados foram apresentado em:

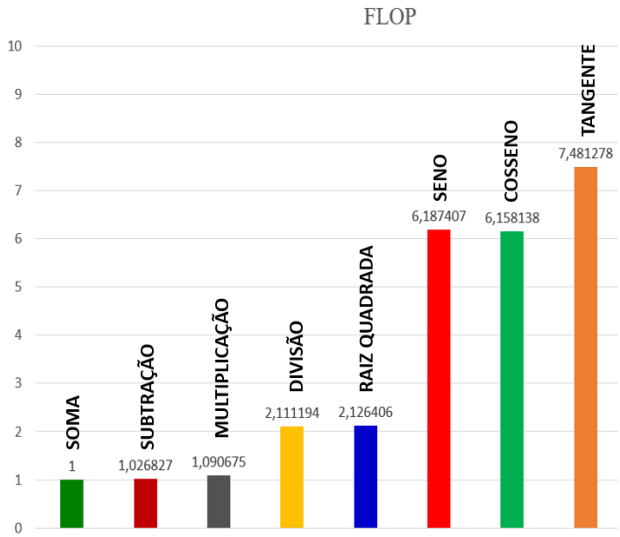

Gráfico 4. Número de Flop para cada operação no MCU

Foi possível contabilizar o número total de flop que operações utilizando as MTHs e os QDUs utilizariam ao serem implementados no MCU, os resultados foram exibidos:

Tabela II - Número total de flop para Cinem. Direta com MTH e QDU

Cinemática Direta com Simplif.	Operações			FLOP total
	Funções Trigon.	Soma e Subtrações	Multip.	
MTH	8	9	19	92
QDU	12	21	52	169

Tabela III - Número total de flop para Cinem. Inversa com MTH e QDU

Cinemática Inversa com Simplif.	Operações			FLOP total
	Funções Trigon.	Soma e Subtrações	Multip.	
MTH	4	14	22	68
QDU	4	39	100	171

Tabela IV - Interrupção aplicada aos algoritmos de cinemática direta MTH e QDU

	Valor do contador	Tempo em segundos
MTH	233.176	$0{,}233s$
QDU	152.770	$0{,}152s$

A comparação entre as funções utilizadas nos algoritmos MTH e QDU para contagem do número de passos pelo modelo RAM foram apresentadas em:

Tabela V - Comparação entre o número de passos modelo RAM para os algoritmos MTH e QDU

Função	Passos do algoritmo MTH	Passos do algoritmo QDU
Declaração Variáveis início do algoritmo	212	100
Quatro Funções de transformação MTH/QDU	232	212
Três Funções de multiplicação mult_matriz / mult_qua	624	369
Função print_array	94	34
Função para obter a translação final translacao_mat/translação_quat	38	46
TOTAL de passos do algoritmo	**1.200**	**761**

Os resultados para algoritmos aplicados à cinemática direta apontam que os QDUs ocupam 1,48 vezes mais espaço de armazenamento para o programa (*sketch*) do que as MTHs, em contrapartida o QDU utiliza quase 2 vezes (1,93) menos a memória dinâmica do MCU, conforme mostrado em:

Tabela VI - Algoritmos de cinemática direta utilizando MTH e QDU

	Espaço de Armazenamento para o programa (SKETCH)	Variáveis globais e locais, uso de memória dinâmica
Total Atmega 328/P	32256 bytes	2048 bytes
MTH	4780 bytes (14% do total)	958 bytes (46% do total)
QDU	7088 bytes (21% do total)	496 bytes (24% do total)

VI. CONCLUSÃO

O cálculo da cinemática direta e inversa fazendo uso do software MAPLE utilizando as Matrizes de Transformação Homogênea e os Quatérnios Duais Unitários chegaram aos mesmos resultados conforme esperado para as duas aplicações.

A cinemática direta utilizando microcontrolador Atmel Atmega328/P para comparação entre Matrizes de Transformação Homogênea e Quatérnios Duais Unitários demonstrou que os QDUs foram superiores nos testes de eficiência temporal utilizando interrupção sendo 1,53 vezes mais rápidos e no teste modelo RAM para análise da eficiência de memória dinâmica ocupada onde os QDUs executaram 1,57 passos a menos do que as MTHs.

O teste para contagem de operações e número de Flop mostrou que os QDUs possuem mais operações do que as MTHs devido à dificuldade em realizar simplificações aritméticas e identidades trigonométricas nos resultados com os QDU.

A análise utilizando o próprio software Arduino IDE verificou que as MTHs ocupavam menor memória de armazenamento do programa, o que era esperado já que estas executaram menos operações aritméticas conforme demostrado em primeiro teste com contagem de operações e número de Flop, por outro lado, os QDUs ocuparam menor memória dinâmica do microcontrolador explicando o motivo de os QDUs terem tido um melhor desempenho nos testes de interrupção e modelo RAM.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] ADDISON,Cliff;ALLWRIGHT,James;BINSTED,Norma; BISHOP,Nigel;CARPENTER,Bryan;DALLOZ,Peter;GEE,David;GETOV,Vladimir;HEY,Tony; HOCKNEY,Roger; LEMKE,Max; MERLIN,John; PINCHES,Mark; SCOTT,Chris; WOLTON, Ivan; *"The Genesis Distributed Memory Benchmarks - I Methodology and General Relativity benchmark with results for the SUPRENUM Computer",* John Wiley & Sons, Ltd, Inglaterra, 1993.

[2] ADORNO, B. V. Two-arm Manipulation: From Manipulators to Enhanced Human-Robot Collaboration. PhD thesis, Université Montpellier 2, Montpellier. França, 2011.

[3] Álgebra linear com aplicações; Howard Anton, Chris Rorres; tradução técnica: Claus Ivo Doering.10 ed. Porto Alegre : Bookman, 2012.

[4] ATMEL, Manual de Instruções Microcontrolador Atmel 328/P; "Atmel-42735B-328/P Datasheet Summary-11/2016", 2016

[5] ATMEL, Manual de Instruções Microcontrolador Atmel 16U2; "Microcontroller with 8/16/32K Bytes of ISP Flash and USB Controller - ATmega16U2 Datasheet Summary",2009. Disponível em: <https://pdf1.alldatasheet.com/datasheet-pdf/view/313554/ ATMEL/ ATmega16U2.html> Acesso em 01 ago.2018

[6] BATES, Martin P.; Programming 8-Bit PIC Microcontrollers in C with interactive Hardware Simulation. 2 ed. Reino Unido: Newnes, julho de 2008

[7] BRANDSTÖTTER, Mathias; Adaptable Serial Manipulators in Modular Design; Instituto de Engenharia de Automação e Controle; Tese de doutorado, Hall in Tirol, 2016

[8] CADY, Fredrick M. Microcontrollers and Microcomputers Principles of software and Hardware Engineering, 2 ed., New York, Oxford University Press, Department of Electrical and Computer Engineering, 2010.

[9] CHEN, Jinbao; HAN,Dong; PENG, Zhuang ; Active Grasping Control of Virtual-Dexterous-Robot Hand with Open Inventor,Hindawi Publishing Corporation Mathematical Problems in Engineering; 2014

[10] CORMEN, Thomas H.; LEISERSON, Charles E. ;RIVEST, Ronald L., CLIFFORD, Stein; Introduction to Algorithms, 3 ed., Massachusetts Institute of Technology, Inglaterra, 2009

[11] CRAIG,John J.;Introduction to Robotics: Mechanics and Control; 3.ed; editora Pearson Education, Estados Unidos, 2005

[12] CRISP, John. Introduction to Microprocessors and Microcontrollers, 2 ed., Estados Unidos:Newnes, 2004

[13] DOMBRE, Etienne; KHALIL, Wisama; Control Systems, Robotics and Manufacturing Series, Robot Manipulators: Modeling, Performance, Analysis and Control; Wiley-ISTE, 2007

[14] EARL, Bill; Memories of an Arduino; Adafruit Learning System, 2018. Disponível em: <learn.adafruit.com/memories-of-an-arduino> Acesso em: 07 ago.2018

[15] FENG, Xiang; WAN, Wanggen. Dual Quaternion Blending Algorithm and Its Application in Character Animation. TELKOMNIKA, Indonesia., Vol. 11, n.10, p. 5553-5562, October 2013

[16] FIORE, James M.; "Embedded Controllers Using C and Arduino / 2E"; Versão 2.0.9, 2018

[17] GE, A. Varshney, J. P. Menon, and C. F. Chang, *"Double quaternions for motion interpolation"* in Proceedings of the ASME Design Engineering Technical Conference, 1998.

[18] GONNET, G.H. e BAEZA-YATES, R.; Handbook of Algorithms and Data Structures. Addison-Wesley, Reading, Mass., 2 ed. Graham, 1991

[19] GOUASMI, Mahmoud; OUALI, Mohammed; BRAHIM, Fernini ; Robot Kinematics Using Dual Quaternions; International Journal of Robotics and Automation (IJRA), Vol. 1, No. 1, pp. 13-30; Blida, Argélia; Março, 2012

[20] KHALIL Wisama; KLEINFINGER,J.F; "A new geometric notation for open and closed-loop robots",Laboratorio de Automação de Nantes, NANTES CEDEX - França, Conference Paper, Maio, 1986

[21] KENWRIGHT,*"A Beginners Guide to Dual-Quaternions: What They Are,How They Work, and How to Use Them for 3D Character*

Hierarchies", The 20th International Conference on Computer Graphics, Visualization and Computer Vision, no. June 26–28, pp. 1–10, 2012
[22] MEI Gang; TIPPER John C.; XU Nengxiong; Numerical Robustness in Geometric Computation: An Expository Summary;No. 6, p.2717-2727; International Journal Applied Mathematics & Information Sciences;2014
[23] NIKU, Saeed Benjamin; *"Introdução à robótica: análise, controle, aplicações"*; tradução e revisão técnica de Sérgio Gilberto Taboada, Rio de Janeiro : LTC, 2ed, 2015
[24] OZGURA,Erol; MEZOUAR, Youcef; *Kinematic modeling and control of a robot arm using unit dual quaternions*. Jornal: Robotics and Autonomous Systems; p.66–73, França, 2016
[25] PARK, Frank C.; Parallel Robots; Robotics Laboratory, Seoul National University, Seoul, Korea;J. Baillieul, T. Samad (eds.), Encyclopedia of Systems and Control; Springer-Verlag London, 2015
[26] RADAVELLI, Luiz; SIMONIA, Roberto; PIERI, Edson Roberto de; MARTINS, Daniel. A Comparative Study of the Kinematics of Robots Manipulators by Denavit-Hartenberg and Dual Quaternion. Asociación Argentina Mechanical Computational Vol XXXI, p.2833-2848, Salta - Argentina, 13-16 November 2012.
[27] RAFIQUZZAMAN, M., Fundamentals of Digital Logic and Microcontrollers, 6 ed., Nova Jersey e Canadá:John Wiley & Sons, 2014
[28] SCHILLING, Robert J.; Fundamentals of Robotics:Analysis and Control;Prentice-Hall of India, Universidade Clarkson, New Delhi, 5ed. 2003
[29] ROMANO, Vitor Ferreira; "Robótica Industrial:Aplicação na Indústria de Manufatura e de Processos"; Editora Edgard Blucher, 1ed., 2002
[30] SCHILLING, "Universally manipulable body models— dual quaternion representations in layered and dynamic MMCs", Autonomous Robots, vol. 30, no. 4, pp. 399–425, 2011
[31] SHOEMAKE, "Animating rotation with quaternion curves" in Proceedings of the 12th annual conference on Computer graphics and interactive techniques. ACM Press, pp. 245–254, 1985.
[32] SICILIANO, Bruno; SCIAVICCO, Lorenzo; VILLANI, Luigi; ORIOLO, Giuseppe; "Robotics:Modelling, Planning and Control"; Springer Handbook of Robotics. Napoli and Stanford, 2009
[33] SKIENA, Steven S. ; The Algorithm Design Manual, 2 ed.; Springer; Londres, 2008
[34] SPONG, M. W., Hutchinson, S., & Vidyasagar, M.. *Robot Dynamics and Control*,. 2 edition. January 28, 2004
[35] SPONG, M. W., Hutchinson, S., & Vidyasagar, M. (2005). *Robot Modeling and Control*. 1 edition. van Zutven, P. W. M., Modeling stability and identification of humanoid robots. Master's thesis, Eindhoven University of Technology, Eindhoven., 2005.
[36] UEBERHUBER, Christoph W.; "Numerical Computation 1: Methods, Software, and Analysis"; Springer, Berlin Heidelberg, 1995
[37] WAGNER, Gerd; DIACONESCU, Mircea; *"Optimize Arduino Memory Usage"*;Alemanha,2015. Disponível em: <https://www.codeproject.com/Articles/1013667/Optimize-Arduino-Memory-Usage> Acesso em: 07 ago.2018
[38] WANG, X.; YU, C. ; Unit-Dual-Quaternion-Based PID Control Scheme for Rigid-Body Transformation; 18º World Congress The International Federation of Automatic Control, Milano, Itália; 2011
[39] YAVUZ, Sırma Ç; Kinematic Analysis For Robot Arm; Yildiz Technical University, Grubo Coşkun Yetim; Istanbul,2009
[40] ZIVIANI, Nivio; Projeto de algoritmos com implementações em Pascal e C, 4 ed.,Pioneira Informática, São Paulo, 1999

Received: August 15, 2018
Accepted: February 05, 2019
Published: February 27, 2019

Analysis of Computational Efficiency applied to Forward and Inverse Kinematics of Robotic Manipulators using Homogeneous Transformation Matrixes and Dual Quaternions

Abstract— The present work proposes a detailed study of the implementation of a mathematical tool called dual quaternions and a depth analysis of its evolution and potential application on the robotics field. In parallel to this view is presented a comparison between the traditional way of calculating rotation and translation in the kinematics of manipulator robots. Dual quaternions have been massively used in robotics because they are computationally more efficient in representing rotational information than the representation with homogeneous transformation matrices. Thus, this work aims to provide a detailed explanation through a step-by-step for the use of quaternion algebra, in a simplified way, were used examples to better understand the implementation. Although there is a lot of literature about the theoretical aspects of dual quaternions, in a few of them there are practical examples of how their use really works. This work gives a clear notion of the introduction to the dual quaternion theory, in addition to it, this document also demonstrates its application to a 4-DOF serial robotic manipulator. In this dissertation it was possible to make a comparison between the direct and inverse kinematics calculated using the matrix algebra versus the quaternionic algebra, it was verified that the quaternions are computationally more efficient although they do not allocate smaller area of the program memory for the Atmel microcontroller Atmega 328/P. Simulations and experimental tests were performed to prove the results.

Index Terms— Unit Dual Quaternion, Matrices of Homogeneous Transformations, Direct Kinematics, Inverse Kinematics.