PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Francisco Jayson Evangelista de Sousa

Cinemática de altitude e trajetória de um veículo para emprego de Unidades de Medida Inercial: modelagem e simulação.

Projeto de Graduação

Projeto de Graduação apresentado ao Departamento de Engenharia Mecânica da PUC-Rio

Orientador: Mauro Speranza Neto

Rio de Janeiro

Dezembro de 2018

DEDICATÓRIA

Ao espirito puro e inocente de buscar aprender e entender o mundo em que vivemos sobre diferentes óticas, tendo a empatia como regra, o respeito como companheiro, o amor como identidade, a fé como guia e que é presente em cada ser humano, mas que deve ser cultivado a cada amanhecer.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por absolutamente tudo, pelas poucas coisas que posso imaginar que ele fez por mim, e principalmente pelas incontáveis as quais não tenho nem noção ainda.

Agradeço a PUC-Rio pela ótima qualidade de ensino e infraestrura, por disponibilizar mestres que se empenham com elevado valor para transimitir o conhecimento que me foi necessário durante todos os períodos aqui e que poderei usar em minha vida profissional.

Agradeço ao meu orientador Mauro Speranza pela dedicação na explicação e orientação.

Agradeço a meus avós Cilene e Viscente por cuidarem de mim e me apoiarem sempre.

Agradeço a minha mãe, Antonia Flaviana, por ser um verdadeiro exemplo em que me espelhei para realizar meus objetivos e que sempre “combatel guerras” para me proporcionar a chance de realizar meus objetivos.

Agradeço ao meu padrasto, Hans Piter, por sua prestação e incentivo.

Agradeço a meus irmãos Anderson e Elisiane por acreditarem em mim e me apoiarem.

Agradeço as minhas tias, Marcia e Raiane, por sempre lutarem e superarem dificuldades para me proporcionar meios para realizar meus objetivos.

Agradeço a Mariana Farias pela sua explendida ajuda, companheirismo, atenção e incentivo em todos os momentos maravilhosos que vivemos.

Agradeço ao Euclides Camacho por ser sempre um amigo e ajudador, colega de curso, e um irmão que, parafraseando Falcão da banda O Rappa, é “sem conta sanguínea”.

Agradeço a Ingrid e Diego por seu louvável companheirismo ao longo de todo esse tempo, mais irmão “sem conta sanguínea”.

Faço um agradecimento especial ao Projeto Contruindo o Saber, que desde 2007 tem me proporcionado experiências com pessoas fantásticas, e que me possibilitou ingressar em instituições que me guiaram para meus objetivos, que foi uma entidade de resistência frente ao abandono do poder publico em relação a parcelas menos abastadas em nossa sociedade.

“Nenhum de nós é tão bom quanto todos nós juntos. ”

Ray kroc

RESUMO

Cinemática de altitude e trajetória de um veiculo para emprego de Unidades de Medida Inercial: modelagem e simulação.

As Unidades de Medida Inercial desempenham atualmente notável valor em nossa sociedade, dado que integram grande parte das tecnologias que realizam processos fundamentais a sociedade moderna, e sua utilização em aeronaves não tripuláveis, os chamados Veiculos Aéreos Não Tripuláveis (VANTs), tem crescido ao decorrer dos anos, de tal maneira que esse mercado atingiu até mesmo consumidores comuns, por meio dos Drones, também são utilizados em motocicletas, e outros dispositivos. Com o crescimento constante desse mercado, e as dificuldades de se analizar e controlar esses mecanismos a fim de se aprimorar e melhorar os sistemas de controle associados a IMU, o presente trabalho se propõe a desenvolver uma abordagem da análise cinemática de um corpo puntiforme que desenvolva trajetórias no espaço tridimensional realizando rotações e translações nos três eixos coordenados utilizando a modelagem através dos Ângulos de Euler, e por conseguintes matrizes de rotação, e Quatérnions. Porém, nesse estudo não são levadas em consideração a análise de influências externas (Dinâmica do corpo) ou de momentos de inércia. O presente trabalho simula, através do software MATLAB (versão R2017a) e utilizando a ferramenta SIMULINK, com base nas Equações envolvendo ângulos de Euler, o comportamento das variáveis cinemáticas atreladas ao movimento, usando o exemplo de um corpo puntiforme percorrendo trajetórias helicoidais nos eixos X, Y e Z. Também é criado um código através da linguagem de programação Visual Basic for Application (VBA) do software Microsoft Excel 2003 para criação de funções e rotinas que possibilitem o uso das principais funções atreladas a Quatérnions para modelagens de rotações.

Palavras Chaves: Visual Basic, Excel, MATLAB, SIMULINK, Quatérnion, Ângulos de Euler, Unidades de Medida Inercial, Rotação, Translação, Movimento.

ABSTRACT

Height kinematics and path of a vehicle for Inertial Measurement Units work. Modeling and simulation.

The Inertial Measurement Unit currently plays a significant role in our society, since it integrates a large part of the technologies that perform fundamental processes in modern society, and its use in unmanned aircraft, the so-called Unmanned Aerial Vehicles (UAV), has grown during the course of of years, in such a way that this market has reached even ordinary consumers, through Drones, also this is used in motobikes and another devices. With the constant growth of this market, and the difficulties of analyzing and controlling these mechanisms in order to improve the control systems associated to IMU, the present work proposes to develop an approach of the kinematic analysis of a body that develops trajectories in the three-dimensional space performing rotations and translations in the three coordinated axes using the Euler Angles modeling, and by means of rotational matrices, and Quaternions. However, in this study, the body sizes has no effect, nor are external influences (body dynamics) or moments of inertia analyzed. This work simulates the behavior of kinematic variables linked to motion through MATLAB software (version R2017a) and using a SIMULINK tool based on the Euler angle equations and the example of a puncture body traversing helical trajectories in the X, Y and Z axes. In this Work, a code is also created through the programming language Visual Basic for Application (VBA) of software Microsoft Excel 2003 to create functions and routines that allow the use of the main functions linked to Quaternions.

Keywords: Visual Basic, MATLAB, SIMULINK, Quaternions, Euler angles, Inertial Unit Mensuring,Rotation, Translation, Motion.

SUMÁRIO

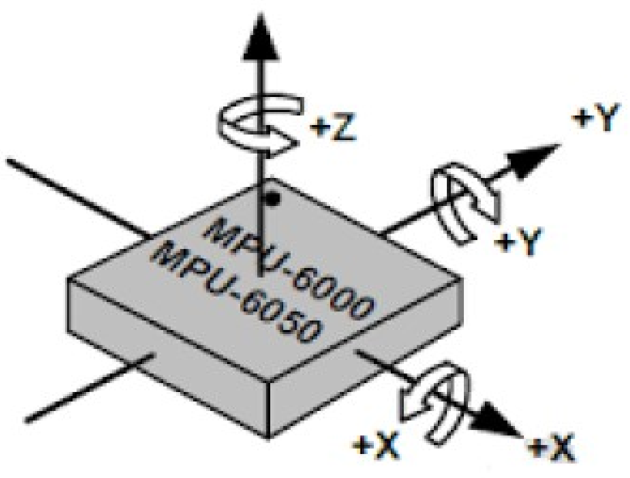
Lista de figuras

1 INTRODUÇÃO

Os drones, aviões, helicópteros, veículos, smartphones e demais corpos que elaboram ou necessitam identificar rotações e translações no espaço tridimensional ganharam mecanismos de controle mais aprimorados tendo em vista a maior gama de utilização e a necessidade de atender requisitos de peformace, segurança, entre outros. Como por exemplo, há estudos para desenvolvimento de drones em ações bélicas ou de contenção de insurgentes, devido a sua capacidade de localizar, confirmar e atacar alvos, e há o uso de drones para reconhecimento e mapeamento de áreas (UBIRATAN 2015).

Para controle do movimento cinemático de corpos pode-se usar uma Unidade de Monitoramento Inercial (do inglês *Inertial Unit Mensuring* – IMU) representada pela Figura 1.1, que consiste em uma combinação entre dispositivos criados a partir da tecnologia dos Sistemas Microeletromecânicos (do inglês *Microelectromechanical System* – MEMS), os acelerômetros, sensores de detecção de aceleração, e os giroscópios, sensores de detecção de velocidade angular, que são capazes de aquisitar entradas cinemáticas e enviar um sinal digital para o sistema embarcado do corpo utilizar (OLIVEIRA;GONÇALVES 2017, ZANONI 2012).

Figura 1.1 – Unidade de Monitoramento Inercial com os graus de liberdade do movimento de translação e rotação.



Fonte: INVENSENSE 2013.

Nesse contexto o presente trabalho tem por objetivo documentar as técnicas de modelagem cinemática das variáveis detectadas pelo IMU, e a partir delas, usar o MATLAB-SIMULINK para simular e calcular a trajetória, as velocidades angulares e lineares, as relações dos ângulos de Euler para o movimento de um corpo puntiforme que percorra uma trajetória livre de influências externas. Desenvolve, posteriormente, através do Visual Basic Application do Microsoft Excel 2013, fórmulas para o cálculo das mesmas variáveis correlatas ao movimento, porém utilizando a metodologia dos Quatérnions, números complexos de 4° dimensão.

Este trabalho se divide em seis capítulos mais esta introdução. O capítulo 2 realiza uma abordagem com respeito a tecnologia MEMS, em que são tratados de maneira resumida os principais aspectos correlatados a essa tecnologia inicialmente. Em seguida são mencionadas as características dos sensores acelerômetros e giroscópios, bem como modelos e seus princípios de funcionamento, e finalmente é abordado os IMUs e suas funcionalidades. Logo após, no capitulo 3, aborda-se a modelagem físico-matemática da cinemática detectada pela IMU para o movimento de um corpo, segundo as premissas já listadas e a metodologia dos Ângulos de Euler com matrizes de rotação. No capitulo 4 há a metodologia da modelagem de rotações utilizando Quatérnions. O capitulo 5 demonstra a metodologia para três simulações do movimento de uma particula em uma trajetória helicoidal ocorrendo nos três eixos cartesianos, utilizando o MATLAB-SIMULINK (versão R2017a). No capitulo 6 há a descrição das funções e rotinas implementadas para o cálculo das operações utilizando quatérnions e para a modelagem de rotações usando tal metodologia, através do Visual Basic Application no Excel 2003, onde as funções são implementadas para uso no ambiente de Planilhas do Excel e também para programadores nessa linguagem que desenvolvam ferramentas que necessitem de tais conceitos. Finalmente, no capitulo 7, são elaboradas as conclusões e considerações finais, que incluem sugestões para trabalhos futuros. Há ainda 2 Anexos contendo os códigos desenvolvidos utilizando o MATLAB e o VBA do Excel.

2 UNIDADE DE MEDIDA INERCIAL

2.1 A tecnologia dos Sistemas Microeletromecânicos.

Atualmente percebe-se que a vida humana se tornou mais prática e confortável do que em épocas mais distantes temporalmente. É compreensível que tal fato ocorre devido às inovações tecnológicas nos mais distintos campos da vida, como por exemplo, ao andar em um carro que possua mais eficiência de rodagem por litro de combustível, decorrente de meticulosos sistemas que possibilitam, entre outras coisas, uma melhor mistura entre ar e combustível para o motor, nos jogos de celulares que detectam o movimento efetuado no aparelho para transmitir para a interface do jogo, entre outros. (MALUF; WILLIAMS, 2004).

Todas essas ferramentas e processos foram possíveis graças a inúmeras pesquisas e estudos na área da Microeletromecânica, que se trata do estudo de dispositivos microscópios, da ordem de micrometros ou até milímetros de comprimento, com partes móveis como membranas, engrenagens, etc. Ou seja, é a mistura entre microcomponentes mecânicos e microcircuitos elétricos de modo a formar um único componente que possibilite execução de tarefas e operações controladas. Assim foi possível desenvolver aparatos mais complexos com essa finalidade, os chamados Sistemas Microeletromecânicos (MEMS - Micro Electro Mechanical Systems), que são dispositivos capazes de desenvolverem tarefas nos mais variáveis e complexos âmbitos de atuação (MALUF; WILLIAMS, 2004, SAFFO, 1997).

Os MEMS, também conhecidos como Microssistemas de Tecnologia (MST – Microsystems Tecnology), são concebidos combinando a tecnologia da microeletrônica baseada em semicondutores, majoritariamente o silício, e através do processo de micro usinagem, técnica que possibilitou a elaboração de chips e placas eletrônicas com os arranjos funcionais mais compactos, ocasionando a miniaturização de circuitos elétricos mais complexos. O propósito do dispositivo é coletar informações através de variáveis físicas ou químicas de vários tipos e faze-las mais sutis ou inteligíveis para os sentidos humanos ou de sistemas de processamento (KORVINK; PAUL, 2004).

Houve ao longo dos anos um grande crescimento da tecnologia dos MEMS, pois cada vez mais os equipamentos para os mais diversos fins, nas mais variadas e complexas áreas tiveram a incorporação dos MEMS em seu funcionamento, ou ocorreu a criação de novos dispositivos baseados nessa tecnologia, devido a sofisticação e desenvolvimento de técnicas de produção da mesma, o que motivou a indústria a incorporação desta nova tecnologia a época após diversos resultados de pesquisas na área.

2.2 A tecnologia das Unidades de Monitoramento Inercial

Uma Unidade de Monitoramento Inercial, IMU, é um dispositivo tipicamente composto por três giroscópios e três acelerômetros para medição das velocidades angulares e acelerações lineares associadas aos três eixos de rotação comumente enunciados como X, Y e Z, ou seja, é o dispositivo capaz de detectar e analisar essas grandes no espaço tridimensional através da detecção de alterações nas grandezas associadas a inercia do sistema, ou seja, velocidades e acelerações lineares e/ou angulares (WETZSTEIN, 2018, P. WON; GOLNARAGHI; W. MELEK, 2009.)

2.2.1. Os acelerômetros:

Os acelerômetros são dispositivos capazes de aferir a aceleração linear de uma massa através de um eixo de referência, ou seja, são dispositivos sensíveis a taxa de variação da velocidade de uma massa ao longo tempo (P. BEEBY, 2004).

A identificação da aceleração pode ser feita de diversas formas por esse sensor, alguns acelerômetros se valem do efeito piezoeléctrico, que é a geração de corrente elétrica devido ao aumento de voltagem decorrido de um aumento de pressão, nesse caso o acelerômetro é feito com micro cristais que quando sofrem uma tensão, alteram a voltagem. Há também os acelerômetros cujo princípio de funcionamento é capacitivo, ou seja, ao se alterar a capacitância do dispositivo sensorial através da variação de distância entre a placas do capacitor ou da área das mesmas, esse emite um sinal elétrico que pode ser convertido em aceleração (P. BEEBY, 2004, MALUF; WILLIAM, 2004), a capacitância C_0 é definida pela formula.

$$C_0 = \epsilon \frac{A}{d}$$

Em que d é a distância entre os eletrodos, A é a área dos eletrodos e ϵ é a permeabilidade elétrica do material dielétrico que fica localizado entre as placas. Há outros MEMS que se valem da variação do coeficiente ϵ, porém não serão tratados aqui.

Todos os acelerômetros apresentam uma estrutura básica semelhante para detecção da aceleração, uma massa inercial suspensa por molas. Porém podem diferir apenas no sensoriamento do deslocamento relativo da massa sensorial em relação a aplicação de uma força externa, tal como mencionado nos exemplos anteriores, onde, por exemplo, no caso capacitivo a massa inercial é posta entre as placas do capacitor para detecção dos deslocamentos, e por conseguinte a aceleração, diferentemente do piezoeléctrico em que a mola pode ser feita de material piezoeléctrico ou possua um filme desse tipo para a detecção de tensões mecânicas na mesma que serão convertidos em sinais elétricos, relacionados ao deslocamento e por conseguinte a aceleração, há ainda a possibilidade de se usar o efeito piezoresitivo, que funcionam analogamente porém a grandeza altera é a resistência elétrica do cristal . Para se detectar acelerações em ambos os eixos coordenados associados ao movimento, normalmente, usa-se três acelerômetros, em cada eixo, para detecção da aceleração.

É importante elaborar a análise do uso do acelerômetro com base em suas especificações primarias em relação as grandezas e parâmetros intervalo completo da escala de detecção, que é comumente dado em termo de acelerações gravitacionais G $(1G \cong 9{,}81\ m/s^2)$, a sensibilidade do sensor (V/G), sua resolução (G), comprimento de Banda (H_z), sensibilidade a eixos cruzados, que é imunidade do sensor a acelerações decorrentes de eixos perpendiculares a direção principal, e, por fim, a resistência a choques mecânicos ou quedas, que é utilizado como fator proteção para o dispositivo durante a operação (G. KORVINK; PAUL, 2006,P. BEEBY, 2004, MALUF; WILLIAM, 2004, LYSHEVSKI, 2002, WETZSTEIN, 2018). Na Figura 2.5 é representado um exemplo de acelerômetro.

2.2.2. Os giroscópios:

Giroscópio é um instrumento de medição desenvolvido para aferir as grandezas angulares, velocidades e deslocamentos, associados a uma massa. Esse dispositivo, que foi criado antes do advento de sistemas de navegação como aqueles baseados em Sistema de Posicionamento Global (Global Position System – GPS), tinha por função manter uma orientação fixa durante as navegações, sendo assim de grande importância (MALUF; WILLIAM, 2004). Tal dispositivo era constituído essencialmente de um volante fixo à anéis concêntricos que giram nos três eixos cartesianos, anéis esses chamados de Gimbal, esse objeto é explicado no capítulo posterior. O grande momento angular do volante contrabalanceava torques aplicados externamente, de tal maneira que a orientação do eixo girante permanecia inalterada (MALUF; WILLIAM, 2004, WETZSTEIN, 2018).

Um convencional giroscópio é formado a partir de volantes concêntricos que o possibilitam realizar rotações tridimensionais. Através do efeito de conservação do momento angular o volante resiste a variações na orientação, esse era o motivo a utilização em embarcações em épocas mais remotas, para aferição das velocidades angulares e deslocamentos angulares, basta medir os ângulos rotacionados pelos gimbals, discos rotacionais, do giroscópio. E há, entretanto, algumas desvantagens associadas a esse tipo de giroscópio, pois por possuir partes moveis mecânicas, estas estão sujeitas a fricções, que podem implicar em erros de aferição das grandezas. (WOODMAN, 2007)

Há, porém giroscópios mais baratos e práticos de se usar, e que podem ser incorporados em muitos outros componentes, são aqueles que utilizam a tecnologia MEMS, estes se valem do efeito Coriolis, que consiste na mudança de percepção da velocidade linear de um corpo em relação a um referencial móvel (GRIMM, 1999). Este tipo de giroscópio elementos vibrantes, como por exemplo massas inerciais fixas por molas, tais molas são compostas normalmente de silício. Porém a velocidade angular é aquisitada através da “medição” da força inercial decorrente do Efeito Coriolis exercida na massa (WOODMAN, 2007).

2.2.3. As IMUs:

As IMUs são os componentes eletrônicos que mesclam as funcionalidades de giroscópios e acelerômetros, e com isso são capazes de analisar, descrever e representar o movimento de um corpo no espaço tridimensional. Para tal usa-se giroscópios e acelerômetros para detecção das grandezas cinemáticas, seja utilizando trios desses em cada eixo, ou, embora menos recorrente, utilizando apenas um de cada um desses dispositivos capazes de serem sensíveis as perturbações nos três eixos (WOODMAN, 2007).

Os IMUs baseados na tecnologia MEMS devem sempre ser testados e calibrados para se garantir a confiabilidade dos valores aferidos e o controle do erro esperado associado a medição. Devido à enorme gama de campos de atuação para esses dispositivos é necessário atentar-se ao grau de precisão e confiabilidade a ser utilizado, de maneira que a escolha represente o adequado dispositivo referente a tarefa que se pretende analisar, já que a IMUs que atuam em áreas com maior umidade, ou seja, as proteções dos circuitos devem ser tal que garanta a integridade do sistema, podem atuar em áreas de elevada temperatura, em que pode ocorrer danos ou fusões de partes de sensores não adequados, podem ser usados em laboratório onde o grau de precisão é maior, e portanto afeta diretamente o custo, podem, também, serem usados com fim recreativo, onde normalmente as especificações são menos rígidas. Portanto é importante adequar a necessidade a escolha do dispositivo de modo a evitar prejuízos de quaisquer naturezas (KEMPE, 2011, WOODMAN 2007, MALUF; WILLIAMS 2004)

3 ÂNGULOS DE EULER

3.1. Historia

Tendo em vista as diferentes formas de representação do movimento de uma partícula ou corpo nos sistemas de coordenadas apresentados, foi necessário o desenvolvimento de técnicas que possibilitassem uma melhor compreensão e representação dos de trajetórias de movimento, para análise das variáveis pertinentes quando se observa esse movimento no espaço euclidiano tridimensional efetuando rotações múltiplas associadas aos eixos tridimensionais, dado que diferentemente dos movimentos de translação e rotações no plano, rotações no espaço tridimensional não são comutativas, ou seja, aplicando-se sequencialmente em cada eixo uma rotação, pode-se chegar a respostas diferentes em um mesmo sistema.

Para evidenciar isso com mais clareza, pode-se usar o exemplo de um avião viajando no sentido Sul que elabore uma rotação de 90° no eixo Leste/Oeste no sentido horário, ou seja, a “cabeça” do avião aponta para o céu, a “barriga” aponta no sentido sul e as asas estão no eixo Leste/Oeste, se o piloto girar o avião de 90° no eixo Norte/Sul no sentido horário, agora o avião estará apontando a “cabeça” para o Leste e mantém a “barriga apontada para o sul” com as asas no eixo Norte/Sul, por fim, se for realizada uma manobra girando 90° no eixo Leste/Oeste no sentido horário, o avião estará viajando para o Leste com a “barriga” apontada para cima e as asas no eixo Norte/Sul. Agora, se forem aplicadas as mesmas rotações, porém em ordem distinta pode-se constatar a não comutatividade da rotação, por exemplo, se avião sofrer as rotações ao longo do eixo Leste/Oeste simultaneamente e por fim, sofrer a rotação no eixo Norte/Sul a posição que o avião se encontra será viajando para o Norte com a “barriga” apontando para o Leste, e asas no eixo vertical. Portanto analisar rotações provem uma necessidade de evidenciar esses efeitos e poder reproduzi-los de maneira fiel, na descrição de um modelo.

Uma das técnicas para parametrização de múltiplas rotações foi descrito e formulado por Leonard Euler (1707-1783), que desenvolveu um método matemático utilizando ângulos que descrevessem as possíveis rotações de um corpo em cada um dos eixos tridimensionais de um plano cartesiano de referência, ou o movimento rotacional completo do corpo como a decomposição em rotações nos eixos, ou seja, qualquer rotação no espaço pode ser representada pela composição de três rotações elementares e sequenciais feitas em três eixos perpendiculares entre si.

Porém a análise de Euler limitava-se a representar os movimentos como rotações combinadas em apenas dois eixos, mais tarde, porém essa abordagem foi reformulada para levar em consideração o terceiro eixo, tal reformulação é comumente chamada de convenção de *Tait-Bryant*, a qual fundamentará o desenvolvimento das equações vistas no presente trabalho.

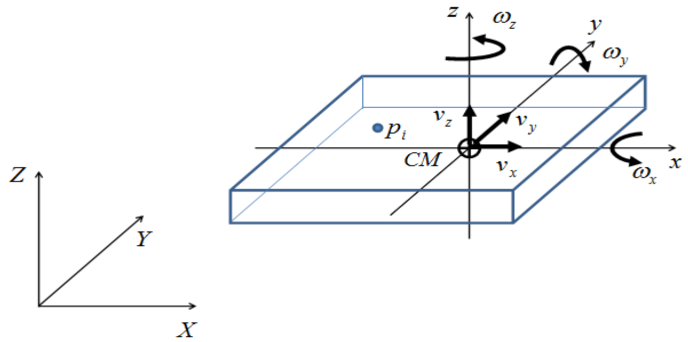
O presente trabalho também se valera da abordagem de *rotações intrínsecas*, ou seja, cada uma das rotações feitas é rotacionado levando em consideração as rotações elaboradas anteriormente de forma sequencial (BIASI; GATTASS, 2002, LAPIN 2008, SYMON 1960)

3.2 Formulação matemática.

Os ângulos usados por *Euler*, segundo a convenção de *Tait-Bryant*, recebem comumente os nomes de Guinada (Yaw), Arfagem (Pitch) e Rolagem (Roll), e são representados pelas letras gregas ψ (rotação no eixo z), θ (rotação no eixo y) e ϕ (rotação no eixo x), respectivamente, esses ângulos representam a magnitude da rotação de um corpo em termos de rotações separadas que produzirão o mesmo efeito. A abordagem proposta por Euler indica o uso de dois grupos de eixos cartesianos, um fixo (coordenadas globais) e outro no centro de massa do corpo (coordenadas locais) para representação do resultado obtido posterior ao conjunto de rotações aplicadas em relação aos eixos iniciais, e para identificação sequencial das rotações a fim de se obter equações sequenciais no modelo matricial.

Para representar as equações do modelo matemático pode-se tomar como exemplo um corpo rígido retangular, como na Figura 3.1:

Figura 3.1 – Grandezas vetoriais associadas ao movimento de rotação e translação tridimensionais, e eixos coordenados no referencial local e global.



Fonte: Notas de Aula – Cinemática Corpo Rígido, PUC-Rio – Mauro Speranza Neto.

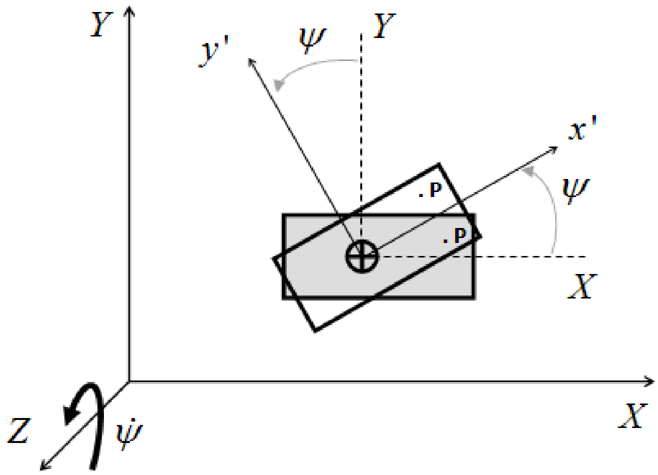
Em que são representadas as velocidades angulares ω e as velocidades transversais v em cada eixo, com Origem no centro de massa do corpo. É explicitado também um ponto p de referência para a modelagem, e por fim, no canto inferior esquerdo há o segundo eixo cartesiano conforme a necessidade proposta por Euler.

Pode-se iniciar o equacionamento do modelo representando apenas os efeitos de rotação do corpo, desconsiderando a translação, ou seja, admitindo que as velocidades são nulas, e, também, serão desconsideradas as dimensões do corpo, ou seja, o mesmo será representado com um objeto puntiforme, evitando assim efeitos de momentos polares e torques. A modelagem se valerá da separação dos efeitos causados por cada rotação, ou seja, será trabalhada cada rotação em cada eixo inicialmente, porém serão tratados os efeitos providos de cada uma sequencialmente, ou seja, cada efeito será adicionado no conjunto de equações a medida que for desenvolvido, para melhorar a análise dos modelos, conforme as proposições já citadas (SYMON 1960, LAGES 2018, JAZAR 2008, SPERANZA NETO 2018)

3.2.1 Guinada.

Iniciando pela guinada, movimento em torno do eixo z, e adotando-se a convenção de valor positivo para rotações no sentido anti-horário conforme a Figura 3.2.

Figura 3.2 – Equacionamento do movimento de rotação pura em Z em relação ao referencial local.



Fonte: Notas de Aula – Cinemática Corpo Rígido, PUC-Rio – Mauro Speranza Neto.

Escrevendo as equações da posição de um ponto genérico *p* após a rotação de um valor ψ em termos dos eixos rotacionados x' e y', e considerando a distância de *p* a origem do eixo de coordenadas locais como fixa, representada por *r* e que o ângulo entre a distância que liga *p* a origem e o eixo das abscissas é α, tem-se:

$$x' = r.\cos(\alpha)$$

$$y' = r.sen(\alpha)$$

Escrevendo as equações para o mesmo ponto rotacionado, agora em relação aos eixos antes da rotação *X* e *Y*, tem-se:

$$X = X_0 + r.\cos(\alpha + \psi)$$

$$Y = Y_0 + r.sen(\alpha + \psi)$$

Onde $X_0\ e\ Y_0$ são as coordenadas iniciais do centro de massa do corpo em relação a origem do referencial global, e usando técnicas de simplificação através das identidades trigonométricas listadas a seguir:

$$sen(a \pm b) = sen(a).\cos(b) \pm sen(b).\cos(a)$$

$$cos(a \pm b) = cos(a).\cos(b) \mp \ sen(a).\text{sen}(\text{b})$$

Portanto pode-se reescrever as equações na forma:

$$X = X_0 + r.\cos(\alpha)\,.\,cos(\psi) - r.\,sen(\alpha).\,sen(\psi)$$

$$Y = Y_0 + r.\,\text{sen}(\alpha)\,.\,cos(\psi) + r.\,cos(\alpha).\,sen(\psi)$$

Portanto utilizando-se os sistemas de equações anteriores, tem-se:

$$X = X_0 + x'.\cos(\psi) - y'.\,sen(\psi)$$

$$Y = Y_0 + x'.\,\text{sen}(\psi) + y'.\,cos(\psi)$$

As equações acima podem ser representadas na forma matricial conforme se segue:

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} + \begin{bmatrix} \cos(\psi) & -sen(\psi) \\ sen(\psi) & cos(\psi) \end{bmatrix} \begin{bmatrix} x' \\ y' \end{bmatrix}$$

Dado que o eixo z não sofre alterações, pois, inicialmente não há rotação no mesmo, e, novamente, Z_0 simboliza a coordenada do centro de massa em relação ao referencial global do eixo z, logo tem-se:

$$Z = Z_0 + z'$$

A representação completa, levando em consideração o eixo z é dada por:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \end{bmatrix} + \begin{bmatrix} \cos(\psi) & -sen(\psi) & 0 \\ sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \qquad (3\ .1)$$

Admitindo que o referencial inicial partiu da origem, ou seja, considerando apenas a rotação:

$$X_0 = Y_0 = Z_0 = 0$$

E através da operação matricial pode-se demonstrar que:

$$A.A^{-1} = A.A^{T} = I \rightarrow A^{-1} = A^{T}$$

$$\begin{bmatrix} \cos(\psi) & -sen(\psi) & 0 \\ sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\psi) & sen(\psi) & 0 \\ -sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Substituindo as formulações anteriores na equação 3.1, tem-se:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\psi) & sen(\psi) & 0 \\ -sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (3.2)$$

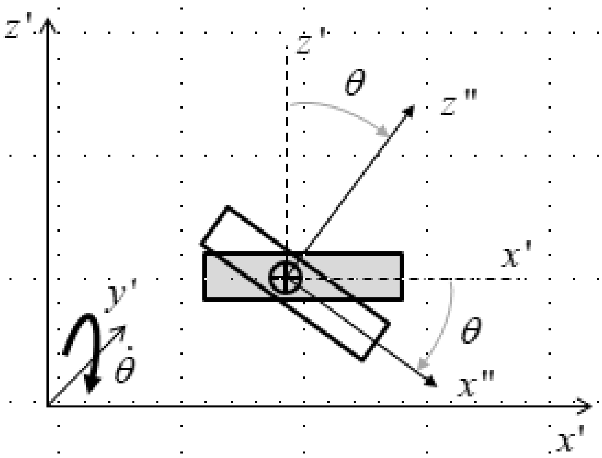
A matriz de ordem maior da equação acima será chamada de matriz de rotação no eixo z e simbolizada por R_z, logo:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = R_z \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

3.2.2 Arfagem.

Efetuando a mesma análise, agora para o movimento em torno do eixo y com base na ilustração que se segue, Figura 3.3.

Figura 3.3 - Equacionamento do movimento de rotação pura em Y’ em relação ao referencial local.



Fonte: Notas de Aula – Cinemática Corpo Rígido, PUC-Rio – Mauro Speranza Neto

Será considerado, conforme a convenção, o sentido de rotação positivo no sentido horário. Com base nos resultados anteriores, obtém-se as seguintes equações em função dos eixos rotacionados z'' e x'' feitos a partir de uma rotação de valor θ:

$$x'' = r.\cos(\alpha)$$

$$z'' = -r.sen(\alpha)$$

Novamente *r* simboliza a distância entre a origem dos eixos e um ponto p qualquer de referência pertencente ao corpo, e α é o ângulo entre essa distância e o eixo das abscissas, e novamente foi considerado apenas os efeitos das rotações. Para a rotação efetuada as equações são:

$$x' = r.\cos(\theta - \alpha)$$

$$z' = -r.\mathrm{sen}(\theta - \alpha)$$

Utilizando a mesma abordagem anterior, pode-se obter o sistema de equações a seguir.

$$x' = x''.\cos(\theta) + z''.sen(\theta)$$

$$z' = -x''.\mathrm{sen}(\theta) + z''.cos(\theta)$$

Considerando sem alterações no eixo y.

$$y' = y''$$

E novamente monta-se o sistema matricial com os três eixos, conforme abaixo:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & sen(\theta) \\ 0 & 1 & 0 \\ -sen(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix}$$

Transformando a equação acima, conforme feito na secção anterior do presente capítulo, obtém-se:

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 & -sen(\theta) \\ 0 & 1 & 0 \\ sen(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad (3.3)$$

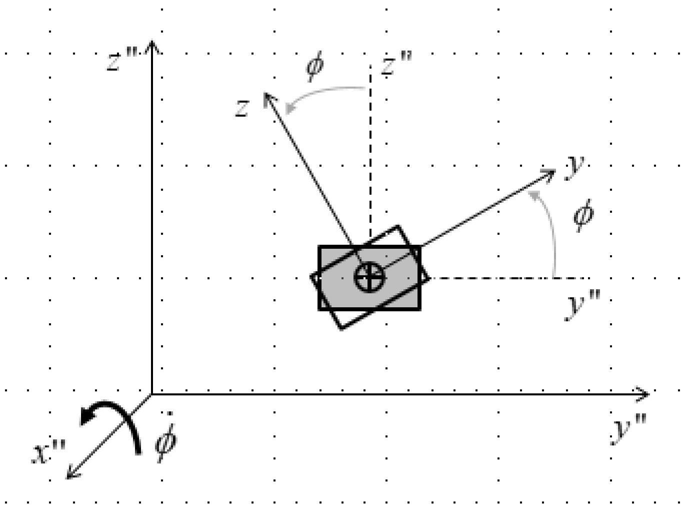
Da mesma maneira pode-se definir a matriz de rotação em torno do eixo y como R_y. Assim pode-se reescrever a equação 3.3 da seguinte forma.

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = R_y \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

3.2.3 Rolagem.

Por fim, efetuando os mesmos procedimentos para a rotação no eixo x, rolagem, conforme esquema a seguir, onde é definida orientação positiva para movimentos no sentido anti-horário, e que após a rotação o eixo sofreu uma inclinação de um ângulo ϕ.

Figura 3.4 - Equacionamento do movimento de rotação pura em x'' em relação ao referencial local



Fonte: Notas de Aula – Cinemática Corpo Rígido, PUC-Rio – Mauro Speranza Neto

Os resultados obtidos em termos das grandezas correlatas ao movimento anteriormente definidas são:

$$y''' = r.\cos(\alpha)$$

$$z''' = r.sen(\alpha)$$

Novamente *r* simboliza a distância entre a origem dos eixos e um ponto p qualquer de referência pertencente ao corpo e α é o ângulo entre essa distância e o eixo das ordenadas. Para a rotação efetuada tem-se:

$$y'' = r.\cos(\phi + \alpha)$$

$$z'' = r.\mathrm{sen}(\phi + \alpha)$$

Utilizando a mesma abordagem como feito nas secções anteriores do presente capítulo, pode-se obter o sistema de equações a seguir.

$$y'' = y'''.\cos(\phi) + z'''.sen(\phi)$$
$$z'' = y'''.\mathrm{sen}(\phi) + z'''.cos(\phi)$$

Considerando sem alterações no eixo x para a dada rotação.

$$x'' = x'''$$

E novamente monta-se o sistema matricial com os três eixos, conforme abaixo:

$$\begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -sen(\phi) \\ 0 & sen(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix}$$

E uma vez mais pode-se alterar a equação matricial para a forma como se segue.

$$\begin{bmatrix} x''' \\ y''' \\ z''' \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & sen(\phi) \\ 0 & -sen(\phi) & cos(\phi) \end{bmatrix} \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix} \quad (3.4)$$

Analogamente pode-se definir a matriz de rotação em torno do eixo x como R_x. Assim pode-se reescrever a equação acima da seguinte forma.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_x \begin{bmatrix} x'' \\ y'' \\ z'' \end{bmatrix}$$

3.3. Matriz de rotação direta e inversa.

Agora que foram definidas as matrizes de rotação em cada eixo é possível combina-la para gerar um único transformador que seja função dos ângulos aplicados e resulte um vetor de posição em dado eixo para obter sua forma rotacionada, para tal utiliza-se a combinação das equações 3.2, 3.3 e 3.4, multiplicando as matrizes que descrevem as rotações em cada eixo entre si e na ordem em que ocorreram para se obter a equação matricial, conforme se segue.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R_x R_y R_z \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (3.5)$$

Onde pode-se definir uma matriz rotacional $R(\phi,\theta,\psi)$ que produz a rotação final nos eixos X, Y e Z obtendo-se os eixos rotacionados $x, y\ e\ z,$ conforme se segue:

$$R(\phi,\theta,\psi) = R_x R_y R_z$$

Sendo assim a equação da matriz rotacional pode ser escrita como:

$$R(\phi,\theta,\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\emptyset) & sen(\emptyset) \\ 0 & -sen(\emptyset) & \cos(\emptyset) \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & -sen(\theta) \\ 0 & 1 & 0 \\ sen(\theta) & 0 & cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & sen(\psi) & 0 \\ -sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R(\phi,\theta,\psi) = \begin{bmatrix} \cos(\theta) & 0 & -sen(\theta) \\ sen(\theta).sen(\phi) & \cos(\phi) & sen(\phi).\cos(\theta) \\ cos(\phi).\mathrm{sen}(\theta) & -\mathrm{sen}(\phi) & \cos(\phi).\cos(\theta) \end{bmatrix} \begin{bmatrix} \cos(\psi) & sen(\psi) & 0 \\ -sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta\,.\cos\psi & \mathrm{sen}\psi.\cos\theta & -sen\theta \\ sen\theta.\mathrm{sen}\,\phi\,.\cos\psi - \cos\phi\,.\mathrm{sen}\psi & sen\theta.\mathrm{sen}\,\phi\,.\mathrm{sen}\psi + \cos\phi.\cos\psi & \mathrm{sen}\,\phi\,.\cos\theta \\ \cos\phi\,.sen\theta.\cos\psi + \mathrm{sen}\,\phi\,.\mathrm{sen}\psi & sen\theta.\cos\phi\,.\mathrm{sen}\psi - \mathrm{sen}\,\phi\,.\cos\psi & \cos\phi.\cos\theta \end{bmatrix} \quad (3.6)$$

As equações acima permitem gerar o resultado final de uma rotação com base no instante anterior as rotações e em função dos ângulos aplicados em cada eixo. A equação geral 3.5 pode ser ainda adaptada para se obter o sentido inicial de um corpo antes de rotações elaboradas no mesmo, utilizando a matriz inversa de R^{-1}, também chamada de matriz de rotação inversa, e analogamente pode-se obtê-la, conforme se segue.

$$R = R_x R_y R_z \rightarrow R^{-1} = \left(R_z R_y R_x\right)^{-1} \rightarrow R^{-1} = R_z^{-1} R_y^{-1} R_x^{-1} = R_z^T R_y^T R_x^T = (R_z R_y R_x)^T = R^T$$

$$R^T = \begin{bmatrix} cos\theta cos\psi & sen\phi sen\theta cos\psi - cos\phi sen\psi & cos\phi sen\theta cos\psi + sen\phi sen\psi \\ cos\theta sen\psi & sen\phi sen\theta sen\psi + cos\psi cos\phi & cos\phi sen\theta sen\psi - sen\phi cos\psi \\ -sen\theta & sen\phi cos\theta & cos\phi cos\theta \end{bmatrix} \quad (3.7)$$

3.4 Velocidades angulares.

Agora é possível determinar também as velocidades angulares em relação ao referencial local, centro de massa do corpo, em cada eixo em função das variações impostas pelos ângulos de Euler. O vetor formado pelas velocidades angulares resultantes, ou seja, as taxas resultantes das variações dos ângulos de rotação do corpo decorrentes das rotações dos ângulos de Euler podem ser expressas conforme se segue:

$$\vec{\Omega} = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = R_x\left(\vec{\dot{\phi}} + R_y\left(\vec{\dot{\theta}} + R_z\vec{\dot{\psi}}\right)\right) = R_\phi\left(\vec{\dot{\phi}} + R_\theta\left(\vec{\dot{\theta}} + R_\psi\vec{\dot{\psi}}\right)\right)$$

Onde $\vec{\Omega}$ é o vetor das velocidades angulares resultantes $(w_x, w_y \; e \; w_z)$ e $\vec{\dot{\phi}}, \vec{\dot{\theta}} \; e \; \vec{\dot{\psi}}$ são as taxas de variação angulares dos vetores ângulos de Euler tal que:

$$\vec{\dot{\phi}} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix}, \quad \vec{\dot{\theta}} = \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} \quad e \quad \vec{\dot{\psi}} = \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}$$

Assim $\vec{\Omega}$ representa a obtenção das velocidades angulares resultantes após a rotação em cada eixo, multiplicando-se a taxa de variação do dado ângulo pela matriz de rotação associada sequencialmente, portanto pode-se encontra-lo $\vec{\Omega}$ conforme se segue.

$$\vec{\Omega} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & sen\phi \\ 0 & -sen\phi & \cos\phi \end{bmatrix}\left(\begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix}\left(\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\psi & sen\psi & 0 \\ -sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}\right)\right)$$

$$\vec{\Omega} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & sen\phi \\ 0 & -sen\phi & \cos\phi \end{bmatrix}\left(\begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix}\left(\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}\right)\right)$$

$$\vec{\Omega} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & sen\phi \\ 0 & -sen\phi & \cos\phi \end{bmatrix}\left(\begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} \cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix}\begin{bmatrix} 0 \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}\right)$$

$$\vec{\Omega} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & sen\phi \\ 0 & -sen\phi & \cos\phi \end{bmatrix}\left(\begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -\dot{\psi}sen\theta \\ \dot{\theta} \\ \dot{\psi}cos\theta \end{bmatrix}\right)$$

$$\vec{\Omega} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & sen\phi \\ 0 & -sen\phi & \cos\phi \end{bmatrix}\begin{bmatrix} \dot{\phi} - \dot{\psi}sen\theta \\ \dot{\theta} \\ \dot{\psi}cos\theta \end{bmatrix} = \begin{bmatrix} \dot{\phi} - \dot{\psi}sen\theta \\ \dot{\theta}cos\phi + \dot{\psi}sen\phi cos\theta \\ \dot{\psi}cos\theta cos\phi - \dot{\theta}sen\phi \end{bmatrix}$$

$$\vec{\Omega} = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & \cos\phi & sen\phi cos\theta \\ 0 & -sen\phi & \cos\phi \, cos\theta \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (3.8)$$

A relação expressada na equação 3.8 descreve uma matriz de transformação das taxas de variação dos ângulos de Euler para encontrar as velocidades angulares resultantes do conjunto de rotações em relação ao referencial local, da mesma maneira pode-se encontrar uma solução para o problema contrário, encontrar as taxas de variação dos ângulos de Euler em função das velocidades angulares resultantes utilizando novamente o vetor $\vec{\Omega}$, conforme se segue.

$$\vec{\Omega} = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = R_\phi\left(\vec{\dot{\phi}} + R_\theta\left(\vec{\dot{\theta}} + R_\psi\vec{\dot{\psi}}\right)\right) \rightarrow\ R_\phi^T\vec{\Omega} = \vec{\dot{\phi}} + R_\theta\left(\vec{\dot{\theta}} + R_\psi\vec{\dot{\psi}}\right)$$

$$R_\theta^T R_\phi^T\vec{\Omega} = R_\theta^T\vec{\dot{\phi}} + \vec{\dot{\theta}} + R_\psi\vec{\dot{\psi}}$$

Aproveitando o resultado anteriormente encontrado $R_\psi\vec{\dot{\psi}} = \vec{\dot{\psi}}$, tem-se:

$$R_\theta^T R_\phi^T\vec{\Omega} = R_\theta^T\vec{\dot{\phi}} + \vec{\dot{\theta}} + \vec{\dot{\psi}}$$

$$\begin{bmatrix} \cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -sen\phi \\ 0 & sen\phi & \cos\phi \end{bmatrix}\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -sen\phi \\ 0 & sen\phi & \cos\phi \end{bmatrix}\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} \dot{\phi}\cos\theta \\ 0 \\ -\dot{\phi}sen\theta \end{bmatrix} + \begin{bmatrix} 0 \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{\phi}\cos\theta \\ \dot{\theta} \\ \dot{\psi} - \dot{\phi}sen\theta \end{bmatrix}$$

$$\begin{bmatrix} \cos\theta & sen\phi sen\theta & \cos\phi sen\theta \\ 0 & \cos\phi & -sen\phi \\ -sen\theta & sen\phi cos\theta & \cos\phi\, cos\theta \end{bmatrix}\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & 0 \\ 0 & 1 & 0 \\ -sen\theta & 0 & 1 \end{bmatrix}\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Pode-se obter a inversa da matriz que multiplica o vetor pretendido conforme expressão:

$$\begin{bmatrix} \cos\theta & 0 & 0 \\ 0 & 1 & 0 \\ -sen\theta & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{\cos\theta} & 0 & 0 \\ 0 & 1 & 0 \\ tg\theta & 0 & 1 \end{bmatrix}$$

Substituindo esse valor para isolar o vetor das derivadas dos ângulos de Euler, tem-se:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{1}{\cos\theta} & 0 & 0 \\ 0 & 1 & 0 \\ tg\theta & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta & sen\phi sen\theta & \cos\phi sen\theta \\ 0 & \cos\phi & -sen\phi \\ -sen\theta & sen\phi cos\theta & \cos\phi\, cos\theta \end{bmatrix}\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}\frac{1}{\cos\theta} & 0 & 0\\ 0 & 1 & 0\\ tg\theta & 0 & 1\end{bmatrix}\begin{bmatrix}\cos\theta & sen\phi sen\theta & \cos\phi sen\theta\\ 0 & \cos\phi & -sen\phi\\ -sen\theta & sen\phi cos\theta & \cos\phi\, cos\theta\end{bmatrix}\begin{bmatrix}w_x\\ w_y\\ w_z\end{bmatrix}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & sen\phi tg\theta & \cos\phi tg\theta\\ 0 & \cos\phi & -sen\phi\\ 0 & sen\phi sen\theta tg\theta + sen\phi cos\theta & \cos\phi sen\theta tg\theta + \cos\phi\, cos\theta\end{bmatrix}\begin{bmatrix}w_x\\ w_y\\ w_z\end{bmatrix}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & sen\phi tg\theta & \cos\phi tg\theta\\ 0 & \cos\phi & -sen\phi\\ 0 & \frac{sen\phi sen^2\theta + sen\phi\cos^2\theta}{cos\theta} & \frac{\cos\phi sen^2\theta + \cos\phi\cos^2\theta}{cos\theta}\end{bmatrix}\begin{bmatrix}w_x\\ w_y\\ w_z\end{bmatrix}$$

$$\begin{bmatrix}\dot{\phi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = \begin{bmatrix}1 & sen\phi tg\theta & \cos\phi tg\theta\\ 0 & \cos\phi & -sen\phi\\ 0 & \frac{sen\phi}{cos\theta} & \frac{\cos\phi}{cos\theta}\end{bmatrix}\begin{bmatrix}w_x\\ w_y\\ w_z\end{bmatrix} \qquad (3.8)$$

E como esperado a relação entre as matrizes de transformação das velocidades e taxas de variação angulares será:

$$\begin{bmatrix}1 & sen\phi tg\theta & \cos\phi tg\theta\\ 0 & \cos\phi & -sen\phi\\ 0 & \frac{sen\phi}{cos\theta} & \frac{\cos\phi}{cos\theta}\end{bmatrix} = \begin{bmatrix}1 & 0 & -sen\theta\\ 0 & \cos\phi & sen\phi cos\theta\\ 0 & -sen\phi & \cos\phi\, cos\theta\end{bmatrix}^{-1}$$

$$\begin{bmatrix}1 & sen\phi tg\theta & \cos\phi tg\theta\\ 0 & \cos\phi & -sen\phi\\ 0 & \frac{sen\phi}{cos\theta} & \frac{\cos\phi}{cos\theta}\end{bmatrix}\begin{bmatrix}1 & 0 & -sen\theta\\ 0 & \cos\phi & sen\phi cos\theta\\ 0 & -sen\phi & \cos\phi\, cos\theta\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

3.4.1 Translação.

Após a determinação da correlação das velocidades angulares com os ângulos de Euler, pode-se generalizar o problema admitindo a translação, ou seja, agora o corpo puntiforme além de sofrer os efeitos decorrentes da translação também experimentará aqueles provenientes da translação. Partindo-se da equação 3.5 que descreve a transformação das coordenadas após a aplicação das rotações.

$$\begin{bmatrix}x\\ y\\ z\end{bmatrix} = R_x R_y R_z \begin{bmatrix}X\\ Y\\ Z\end{bmatrix}$$

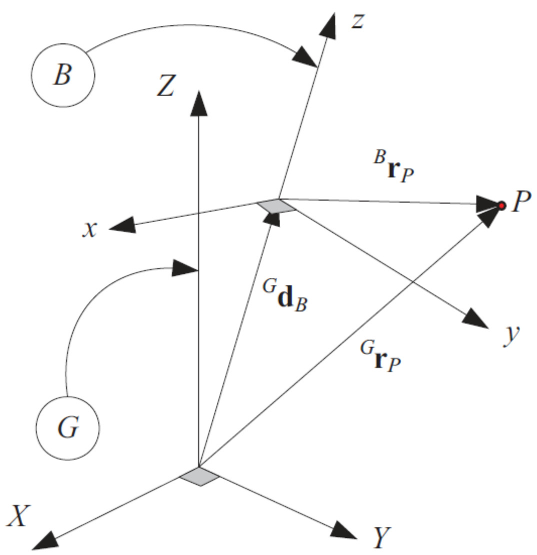
Que será reescrita em função dos valores rotacionados (r_p^G), em relação ao referencial global, e anterior a rotação (r_p^B), em relação ao referencial local (no centro de massa) e vice-versa, da seguinte forma.

$$r_p^B = R_x R_y R_z . r_p^G = R r_p^G = R_B r_p^G \quad (3.9)$$

$$r_p^G = R_z^T R_y^T R_x^T . r_p^B = R^T r_p^B = R_G r_p^B \quad (3.10)$$

Agora analisando o movimento do corpo que se desloca ao longo do espaço conforme a figura 3.5.

Figura 3.5 – Vetores Deslocamento em relação ao referencial local e global no movimento de rotação e translação.



Fonte: Notas de Aula – Cinemática Corpo Rígido, PUC-Rio – Mauro Speranza Neto

Na qual são representados os eixos que descrevem o referencial local, posto no centro de massa do corpo, e o referencial global em um ponto fixo externo, de um corpo tido como puntiforme representado por P. Na figura d_B^G representa o vetor deslocamento devido a translação em relação ao referencial global. Portanto a equação 3.j1 pode ser reescrita para admitir a translação como se segue.

$$r_p^G = R_z^T R_y^T R_x^T . r_p^B + d_p^G = R_G r_p^B + d_B^G \quad (3.10)$$

3.5 Velocidade Linear.

Agora pode-se também determinar o vetor velocidade linear associado ao movimento do corpo no referencial global, isto é, devido a rotação e a translação, para isso basta derivar a equação 3.10 com respeito ao tempo t.

$$\frac{d\,r_p^G}{dt} = \frac{d}{dt}(R_z^T R_y^T R_x^T . r_p^B + d_p^G) \rightarrow \; v_p^G = \dot{r_p^G} = \dot{R_G}. r_p^B + R_G . \dot{r_p^B} + \dot{d_B^G}$$

Como admite-se que o corpo puntiforme é rígido, ou seja, sem deformação a parcela $\dot{r_p^B}$ é nula, tem-se:

$$v_p^G = R_G . r_p^B + \dot{d_p^G}$$

Substituindo a o valor do vetor posição do referencial local r_p^B a partir da equação 3.J3, na equação acima obtém-se:

$$v_p^G = R_G . (R_G^T)^T (r_p^G - d_B^G) + \dot{d_B^G} = \dot{R_G} . R_B (r_p^G - d_B^G) + \dot{d_B^G}$$

Antes de tudo serão determinados os coeficientes da matriz $\dot{R_G}$ tal que:

$$\dot{R_G} = \frac{d}{dt}(R_z^T R_y^T R_x^T) = \dot{R_z^T}(R_y^T R_x^T) + R_z^T \dot{R_y^T} R_x^T + (R_z^T R_y^T)\dot{R_x^T}$$

Sendo assim basta encontrar os coeficientes das matrizes $\dot{R_x^T}, \dot{R_y^T}\; e\; \dot{R_z^T}$. Que serão resolvidos derivando cada coeficiente das matrizes com respeito ao tempo conforme a seguir.

$$\dot{R_z^T} = \frac{d}{dt}\begin{bmatrix} \cos\psi & -sen\psi & 0 \\ sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -\dot{\psi} sen\psi & -\dot{\psi} cos\psi & 0 \\ \dot{\psi} cos\psi & -\dot{\psi} sen\psi & 0 \\ 0 & 0 & 0 \end{bmatrix} = \dot{\psi}\begin{bmatrix} -sen\psi & -cos\psi & 0 \\ cos\psi & -sen\psi & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\dot{R_y^T} = \begin{bmatrix} -\dot{\theta}\mathrm{sen}\theta & 0 & \dot{\theta}\cos\theta \\ 0 & 0 & 0 \\ -\dot{\theta} cos\theta & 0 & -\dot{\theta} sen\theta \end{bmatrix} = \dot{\theta}\begin{bmatrix} -\mathrm{sen}\theta & 0 & \cos\theta \\ 0 & 0 & 0 \\ -cos\theta & 0 & -sen\theta \end{bmatrix}$$

$$\dot{R_x^T} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\dot{\phi}\,\mathrm{sen}\,\phi & -\dot{\phi} cos\phi \\ 0 & \dot{\phi} cos\phi & -\dot{\phi}\,\mathrm{sen}\,\phi \end{bmatrix} = \dot{\phi}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -\mathrm{sen}\,\phi & -cos\phi \\ 0 & cos\phi & -\mathrm{sen}\,\phi \end{bmatrix}$$

Para encontrar a expressão de v_p^G deve-se obter o produto matricial entre $\dot{R_G}\; e\; R_B$, ou seja, entre $\dot{R^T}\; e\; R$.

$$\dot{R^T} R = (\dot{R_z^T}(R_y^T R_x^T) + R_z^T \dot{R_y^T} R_x^T + (R_z^T R_y^T)\dot{R_x^T}) R_x R_y R_z$$

$$\dot{R^T} R = \dot{R_z^T} R_y^T R_x^T R_x R_y R_z + R_z^T \dot{R_y^T} R_x^T R_x R_y R_z + R_z^T R_y^T \dot{R_x^T} R_x R_y R_z$$

$$\dot{R}^T R = \dot{R}_z^T R_y^T I R_y R_z + R_z^T \dot{R}_y^T I R_y R_z + R_z^T R_y^T \dot{R}_x^T R_x R_y R_z$$

$$\dot{R}^T R = \dot{R}_z^T R_z + R_z^T (\dot{R}_y^T R_y) R_z + R_z^T R_y^T (\dot{R}_x^T R_x) R_y R_z$$

Calculando-se cada termo da expressão acima, a começar pelo primeiro obtém-se:

$$\dot{R}_z^T R_z = \dot{\psi} \begin{bmatrix} -sen\psi & -cos\psi & 0 \\ cos\psi & -sen\psi & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\psi & sen\psi & 0 \\ -sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\dot{R}_z^T R_z = \dot{\psi} \begin{bmatrix} -sen\psi\cos\psi + sen\psi\cos\psi & -sen^2\psi - \cos^2\psi & 0 \\ sen^2\psi + \cos^2\psi & sen\psi\cos\psi - sen\psi\cos\psi & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\dot{R}_z^T R_z = \dot{\psi} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Agora para a segunda parcela pode-se encontrar a expressão:

$$R_z^T (\dot{R}_y^T R_y) R_z = R_z^T \left(\dot{\theta} \begin{bmatrix} -\mathrm{sen}\theta & 0 & \cos\theta \\ 0 & 0 & 0 \\ -cos\theta & 0 & -sen\theta \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix} \right) R_z$$

$$R_z^T (\dot{R}_y^T R_y) R_z = R_z^T \left(\dot{\theta} \begin{bmatrix} -\mathrm{sen}\theta\cos\theta + \mathrm{sen}\theta\cos\theta & 0 & \mathrm{sen}^2\theta + \cos^2\theta \\ 0 & 0 & 0 \\ -\mathrm{sen}^2\theta - \cos^2\theta & 0 & \mathrm{sen}\theta\cos\theta - \mathrm{sen}\theta\cos\theta \end{bmatrix} \right) R_z$$

$$R_z^T (\dot{R}_y^T R_y) R_z = R_z^T \left(\dot{\theta} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \right) R_z$$

$$R_z^T (\dot{R}_y^T R_y) R_z = \begin{bmatrix} \cos\psi & -sen\psi & 0 \\ sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \dot{\theta} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\psi & sen\psi & 0 \\ -sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_z^T (\dot{R}_y^T R_y) R_z = \dot{\theta} \begin{bmatrix} \cos\psi & -sen\psi & 0 \\ sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -\cos\psi & -\mathrm{sen}\,\psi & 0 \end{bmatrix}$$

$$R_z^T (\dot{R}_y^T R_y) R_z = \dot{\theta} \begin{bmatrix} 0 & 0 & \cos\psi \\ 0 & 0 & \mathrm{sen}\,\psi \\ -\cos\psi & -\mathrm{sen}\,\psi & 0 \end{bmatrix}$$

Finalmente para a última parcela obtém-se:

$$R_z^T R_y^T (\dot{R}_x^T R_x) R_y R_z = R_z^T R_y^T \left(\dot{\phi} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\mathrm{sen}\,\phi & -cos\phi \\ 0 & cos\phi & -\mathrm{sen}\,\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & sen\phi \\ 0 & -sen\phi & \cos\phi \end{bmatrix} \right) R_y R_z$$

$$= R_z^T R_y^T \left(\dot{\phi} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -\operatorname{sen}\phi cos\phi + \operatorname{sen}\phi cos\phi & -\operatorname{sen}^2\phi - \cos^2\phi \\ 0 & \operatorname{sen}^2\phi + \cos^2\phi & -\operatorname{sen}\phi cos\phi + \operatorname{sen}\phi cos\phi \end{bmatrix} \right) R_y R_z$$

$$= \dot{\phi} R_z^T R_y^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} R_y R_z$$

$$= \dot{\phi} \begin{bmatrix} \cos\psi & -sen\psi & 0 \\ sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & -sen\theta \\ 0 & 1 & 0 \\ sen\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & sen\psi & 0 \\ -sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \dot{\phi} \begin{bmatrix} \cos\psi & -sen\psi & 0 \\ sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} cos\theta & 0 & sen\theta \\ 0 & 1 & 0 \\ -sen\theta & 0 & cos\theta \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ -sen\theta & 0 & -cos\theta \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\psi & sen\psi & 0 \\ -sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \dot{\phi} \begin{bmatrix} \cos\psi & -sen\psi & 0 \\ sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & sen\theta & 0 \\ -sen\theta & 0 & -cos\theta \\ 0 & cos\theta & 0 \end{bmatrix} \begin{bmatrix} \cos\psi & sen\psi & 0 \\ -sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \dot{\phi} \begin{bmatrix} \cos\psi & -sen\psi & 0 \\ sen\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -sen\psi sen\theta & \cos\psi\, sen\theta & 0 \\ -\cos\psi\, sen\theta & -sen\psi sen\theta & -cos\theta \\ -sen\psi cos\theta & \cos\psi\, cos\theta & 0 \end{bmatrix}$$

$$= \dot{\phi} \begin{bmatrix} -\cos\psi\, sen\psi sen\theta + \cos\psi\, sen\psi sen\theta & \cos^2\psi\, sen\theta + \operatorname{sen}^2\psi\, sen\theta & sen\psi cos\theta \\ -\cos^2\psi\, sen\theta - \operatorname{sen}^2\psi\, sen\theta & -\cos\psi\, sen\psi sen\theta + \cos\psi\, sen\psi sen\theta & -\cos\psi\, cos\theta \\ -sen\psi cos\theta & \cos\psi\, cos\theta & 0 \end{bmatrix}$$

$$= \dot{\phi} \begin{bmatrix} 0 & sen\theta & sen\psi cos\theta \\ -sen\theta & 0 & -\cos\psi\, cos\theta \\ -sen\psi cos\theta & \cos\psi\, cos\theta & 0 \end{bmatrix}$$

Com os resultados obtidos anteriormente pode-se solucionar a equação 3.J6 como se segue.

$$\dot{R^T} R = \dot{\psi} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \dot{\theta} \begin{bmatrix} 0 & 0 & \cos\psi \\ 0 & 0 & \operatorname{sen}\psi \\ -\cos\psi & -\operatorname{sen}\psi & 0 \end{bmatrix} +$$

$$+ \dot{\phi} \begin{bmatrix} 0 & sen\theta & sen\psi cos\theta \\ -sen\theta & 0 & -\cos\psi\, cos\theta \\ -sen\psi cos\theta & \cos\psi\, cos\theta & 0 \end{bmatrix}$$

$$\dot{R^T} R = \begin{bmatrix} 0 & -\dot{\psi} + \dot{\phi} sen\theta & \dot{\theta}\cos\psi + \dot{\phi} sen\psi cos\theta \\ \dot{\psi} - \dot{\phi} sen\theta & 0 & \dot{\theta} sen\psi - \dot{\phi} cos\psi cos\theta \\ -\dot{\theta}\cos\psi - \dot{\phi} sen\psi cos\theta & -\dot{\theta} sen\psi + \dot{\phi} cos\psi cos\theta & 0 \end{bmatrix}$$

Com isso pode-se definir a matriz de velocidades angulares de transformação ω_B^G tal como se segue:

$$\omega_B^G = \dot{R^T} R$$

$$\varpi_B^G = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix} \quad (3.11)$$

Na qual os coeficientes ficam correlatados da seguinte forma:

$$w_1 = -\dot{\theta} sen\psi + \dot{\phi} cos\psi cos\theta$$

$$w_2 = \dot{\theta} \cos\psi + \dot{\phi} sen\psi cos\theta$$

$$w_3 = \dot{\psi} - \dot{\phi} sen\theta$$

O conjunto de equações acima pode ser escrito na forma matricial definindo-se a matriz de rotação do referencial global para o local, conforme se segue:

$$w_B^G = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} -\dot{\theta} sen\psi + \dot{\phi} cos\psi cos\theta \\ \dot{\theta} \cos\psi + \dot{\phi} sen\psi cos\theta \\ \dot{\psi} - \dot{\phi} sen\theta \end{bmatrix} = \begin{bmatrix} cos\psi cos\theta & -sen\psi & 0 \\ sen\psi cos\theta & \cos\psi & 0 \\ -sen\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad (3.12)$$

Recordando a equação 3.8 pode-se associa-la o sistema matricial acima.

$$\vec{\Omega} = w_B^B = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & \cos\phi & sen\phi cos\theta \\ 0 & -sen\phi & \cos\phi\, cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

É demonstrada a relação entre os vetores velocidade angular w_B^B e w_B^G conforme se segue.

$$w_B^B = R_B w_B^G \quad (3.13)$$

Pode-se fazer a prova matemática resolvendo o lado direito do sistema matricial a fim de confirmar a veracidade.

$$\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = R_B \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & \cos\phi & sen\phi cos\theta \\ 0 & -sen\phi & \cos\phi\, cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = R_B \begin{bmatrix} cos\psi cos\theta & -sen\psi & 0 \\ sen\psi cos\theta & \cos\psi & 0 \\ -sen\theta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & \cos\phi & sen\phi cos\theta \\ 0 & -sen\phi & \cos\phi\, cos\theta \end{bmatrix} = R_B \begin{bmatrix} cos\psi cos\theta & -sen\psi & 0 \\ sen\psi cos\theta & \cos\psi & 0 \\ -sen\theta & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta).\cos(\psi) & sen(\psi).\cos(\theta) & -sen(\theta) \\ sen(\theta).sen(\phi).\cos(\psi) - \cos(\phi).sen(\psi) & \mathrm{sen}(\theta).\mathrm{sen}(\phi).\mathrm{sen}(\psi) + \cos(\phi).\cos(\psi) & sen(\phi).\cos(\theta) \\ cos(\phi).\mathrm{sen}(\theta).\cos(\psi) + \mathrm{sen}(\phi).\mathrm{sen}(\psi) & \mathrm{sen}(\theta).\cos(\phi).\mathrm{sen}(\psi) - \mathrm{sen}(\phi).\cos(\psi) & \cos(\phi).\cos(\theta) \end{bmatrix} \begin{bmatrix} cos\psi cos\theta & -sen\psi & 0 \\ sen\psi cos\theta & \cos\psi & 0 \\ -sen\theta & 0 & 1 \end{bmatrix}$$

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

Onde os coeficientes da matriz são determinados pelas expressões:

$M_{11} = cos\theta cos\psi cos\psi cos\theta + cos\theta sen\psi sen\psi cos\theta + sen\theta sen\theta$

$M_{11} = \cos^2\theta \cos^2\psi + \cos^2\theta\, sen^2\psi + sen^2\theta = \cos^2\theta\,(\cos^2\psi + sen^2\psi) + sen^2\theta = 1$

$M_{12} = -cos\theta cos\psi sen\psi + cos\theta sen\psi cos\psi = 0$

$M_{13} = -sen\theta$

$M_{21} = (sen\phi sen\theta cos\psi - cos\phi sen\psi)cos\psi cos\theta + (sen\phi sen\theta sen\psi + cos\phi cos\psi)sen\psi cos\theta -$
$sen\phi cos\theta sen\theta$

$M_{21} = sen\phi sen\theta cos\theta(\cos^2\psi + sen^2\psi) - sen\phi cos\theta sen\theta = 0$

$M_{22} = -(sen\phi sen\theta cos\psi - cos\phi sen\psi)sen\psi + (sen\phi sen\theta sen\psi + cos\phi cos\psi)cos\psi$

$M_{22} = -sen\phi sen\theta cos\psi sen\psi + cos\phi sen^2\psi + sen\phi sen\theta sen\psi cos\psi + cos\phi \cos^2\psi$

$M_{22} = cos\phi(sen^2\psi + \cos^2\psi) = cos\phi$

$M_{23} = sen\phi cos\theta$

$M_{31} = (cos\phi sen\theta cos\psi + sen\phi sen\psi)cos\psi cos\theta + \ (cos\phi sen\theta sen\psi -$
$sen\phi cos\psi)sen\psi cos\theta - cos\phi cos\theta sen\theta$

$M_{31} = cos\phi sen\theta cos\theta(\cos^2\psi + sen^2\psi) - cos\phi cos\theta sen\theta = 0$

$M_{32} = -(cos\phi sen\theta cos\psi + sen\phi sen\psi)sen\psi + (cos\phi sen\theta sen\psi - sen\phi cos\psi)cos\psi$

$M_{32} = -cos\phi sen\theta cos\psi sen\psi - sen\phi sen^2\psi + cos\phi sen\theta sen\psi cos\psi - sen\phi \cos^2\psi$

$M_{32} = -sen\phi(sen^2\psi + \cos^2\psi) = -sen\phi$

$M_{33} = cos\phi cos\theta$

Portanto o resultado para a matriz de coeficientes M será:

$$= \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & cos\phi & sen\phi cos\theta \\ 0 & -sen\phi & cos\phi cos\theta \end{bmatrix}$$

A matriz acima é exatamente a matriz w_B^B, o que confirma a validade da expressão 3.13. Agora é possível determinar a forma do vetor velocidade v_p^G, conforme se segue.

$$v_p^G = \dot{R_G}.R_B\left(r_p^G - d_p^G\right) + \dot{d_B^G}$$

$$v_p^G = \varpi_B^G\left(r_p^G - d_p^G\right) + \dot{d_B^G}$$

Onde ϖ_B^G é descrito conforme equação 3.11 conforme abaixo.

$$\varpi_B^G = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix}$$

Ao se multiplicar a matriz ϖ_B^G por um vetor de 3 coordenadas qualquer, o que se obtém é o produto vetorial entre w_B^G e o dado vetor. Está afirmação pode ser provada conforme abaixo para um vetor A qualquer de coordenadas $a_1, a_2\ e\ a_3$.

$$\varpi_B^G.A = \begin{bmatrix} 0 & -w_3 & w_2 \\ w_3 & 0 & -w_1 \\ -w_2 & w_1 & 0 \end{bmatrix}\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_3w_2 - a_2w_3 \\ a_1w_3 - a_3w_1 \\ a_2w_1 - a_1w_2 \end{bmatrix}$$

$$w_B^G x A = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} x \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} a_3w_2 - a_2w_3 \\ a_1w_3 - a_3w_1 \\ a_2w_1 - a_1w_2 \end{bmatrix}$$

Conforme visto acima, pode-se substituir a matriz ϖ_B^G pelo produto vetorial com o vetor w_B^G, sendo assim pode-se reescrever a equação de v_p^G conforme abaixo.

$$v_p^G = w_B^G x\left(r_p^G - d_p^G\right) + \dot{d_B^G} \quad (3.14)$$

De maneira análoga pode-se determinar o vetor de velocidades em relação ao referencial local v_p^B.

$$v_P^B = R_B v_p^G \quad (3.15)$$

$$v_p^B = R_B\left(w_B^G x\left(r_p^G - d_p^G\right) + \dot{d_B^G}\right) = R_B w_B^G x R_B\left(r_p^G - d_p^G\right) + R_B \dot{d_B^G}$$

$$v_p^B = w_B^B x R_B\left(r_p^G - d_p^G\right) + R_B \dot{d_B^G}$$

E utilizando a equação 3.10 a expressão acima se torna

$$r_p^G = R_G r_p^B + d_B^G \rightarrow r_p^B = R_G^T(r_p^G - d_B^G) = R_B(r_p^G - d_B^G)$$

$$v_p^B = w_B^B x r_p^B + R_B \dot{d}_B^G \quad (3.16)$$

Com isso é possível determinar o vetor velocidade do corpo em movimento com relação a ambos os referenciais, local e global.

3.6. Aceleração.

Outra etapa importante para determinação das grandezas cinemáticas e o cálculo da expressão que define a aceleração, para isso partisse da derivação da equação 3.14 com respeito ao tempo no referencial global, conforme é visto abaixo.

$$a_p^G = \frac{d}{dt} v_p^G = \dot{v}_p^G = \ddot{r}_p^G = \frac{d}{dt}(w_B^G x(r_p^G - d_B^G) + \dot{d}_B^G)$$

$$a_p^G = \dot{w}_B^G x(r_p^G - d_B^G) + w_B^G x\left(\dot{r}_p^G - \dot{d}_B^G\right) + \ddot{d}_B^G$$

$$a_p^G = \dot{w}_B^G x(r_p^G - d_B^G) + w_B^G x\left(v_p^G - \dot{d}_p^G\right) + \ddot{d}_B^G$$

O termo central da equação acima pode ser substituído utilizando a equação 3.14, obtendo-se:

$$a_p^G = \dot{w}_B^G x(r_p^G - d_B^G) + w_B^G x(w_B^G x(r_p^G - d_p^G)) + \ddot{d}_B^G$$

$$a_p^G = \alpha_B^G x(r_p^G - d_B^G) + w_B^G x(w_B^G x(r_p^G - d_p^G)) + \ddot{d}_B^G \qquad (3.17)$$

A equação 3.17 descreve a aceleração resultante do movimento do corpo no referencial global, onde a primeira parcela da equação $\left(\alpha_B^G x(r_p^G - d_B^G)\right)$ representa a aceleração tangencial a curva de rotação, a segunda parcela $(w_B^G x(w_B^G x(r_p^G - d_p^G)))$ representa a aceleração centrípeta e a última parcela $\left(\ddot{d}_B^G\right)$ representa a aceleração devido a translação do corpo. Pode-se também encontrar a aceleração em relação ao referencial local, ou seja, em relação ao centro de massa do corpo. De maneira análoga ao feito para a velocidade angular e para o vetor deslocamento, basta multiplicar o vetor aceleração no referencial global pela matriz de rotação, conforme se explicita a seguir.

$$a_p^B = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = R(\phi, \theta, \psi) a_p^G = R a_p^G = R_B a_p^G$$

Da mesma maneira pode-se determinar os coeficientes da equação 3.J10 em relação ao referencial local.

$$w_B^B = \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = R_B w_B^G$$

$$\alpha_B^B = \dot{w}_B^B = \frac{d}{dt}\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix} = R_B \alpha_B^G$$

$$v_p^B = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = R_B v_p^G$$

Portanto, substituindo-se esses termos na equação de $a_p^B,$ obtém-se:

$$a_p^B = R_B a_P^G = R_B \left(\alpha_B^G x\left(r_p^G - d_B^G\right) + w_B^G x \left(w_B^G x\left(r_p^G - d_p^G\right)\right) + \ddot{d}_B^G\right)$$

$$a_p^B = R_B \alpha_B^G x R_B\left(r_p^G - d_B^G\right) + R_B w_B^G x \left(R_B w_B^G x R_B\left(r_p^G - d_p^G\right)\right) + R_B \ddot{d}_B^G$$

$$a_p^B = \alpha_B^B x R_B\left(r_p^G - d_B^G\right) + w_B^B x \left(w_B^B x R_B\left(r_p^G - d_p^G\right)\right) + R_B \ddot{d}_B^G$$

Conforme equação 3.10 a equação acima pode ser alterada para.

$$r_p^G = R_G r_p^B + d_B^G \rightarrow r_p^B = R_G^T\left(r_p^G - d_B^G\right) = R_B\left(r_p^G - d_B^G\right)$$

$$a_p^B = \alpha_B^B x r_p^B + w_B^B x\left(w_B^B x r_p^B\right) + R_B \ddot{d}_B^G \quad (3.18)$$

A equação acima descreve a aceleração do corpo com respeito ao referencial local, ou seja, em relação ao centro de massa do corpo para esta análise.

Agora é possível determinar todas as grandezas cinemáticas relacionadas ao movimento do corpo que rotaciona e translada no espaço.

Como a solução proposta, seguindo a parametrização de Euler, representa um sistema matricial “hierárquico”, ou seja, as rotações devem ser aplicadas em cada eixo seguindo uma ordem, a Ordem de Rotação, que nesse caso, convencionou-se usar XYZ, que representa a ordem dos eixos sequencialmente, o que implica que ao se rotacionar o eixo Z todos os eixos são também rotacionados, porém ao rotacionar o eixo Y, apenas ele e X são rotacionados e por fim, ao rotacionar o eixo X, somente ele sofre rotação.

É importante ressaltar que a análise elabora anteriormente será válida para um sistema em que se despreze os efeitos de momentos associados a massa e forças que atuem no corpo, pois o presente trabalho se propõem apenas a avaliar a cinemática associada ao objeto de estudo para tal, este será considerado como um corpo pontual.

Embora a abordagem do movimento usando ângulos de Euler possibilite uma melhor visualização dos fenômenos e utilize apenas funções geométricas para descrição das rotações e possibilite fácil compreensão nos resultados obtidos, quando se usa essa técnica para descrição de pequenas rotações sequenciais, por exemplo a visualização da correção da rota de um avião, ou o sequenciamento de uma rotação em várias variações angulares, de maneira ter um movimento suave e não uma transformação abrupta, podem ocorrer inconsistências nos efeitos esperados, esses erros são representados a seguir (SYMON 1960, LAGES 2018, JAZAR 2008, SPERANZA NETO 2018).

3.7. Gimbal Lock.

Gimbal é um mecanismo formado por três aros concêntricos que podem rotacionar sob os três eixos cartesianos. O efeito conhecido como Gimbal Lock ocorre quando dois dos eixos dos gimbals se encontram paralelos, ou seja, dois círculos gimbals rotacionam sobre o mesmo eixo, isso faz com que nenhum dos eixos possa elaborar rotações no eixo perpendicular a esse plano, com isso há a perda de um grau de liberdade rotacional para o sistema (HOAG 1963, HAND 1971). Tal problema sempre poderá ocorrer ao usar os ângulos de Euler, pois, em muitos casos, para se manter a boa qualidade da representação de movimentos de corpos é necessário que essas sejam suaves, ou seja, uma rotação será feita através de pequenas rotações sequenciais, nesse âmbito o uso dos ângulos de Euler pode gerar inconsistências nos fenômenos não muito intuitivas, porém com muita veracidade.

Esse fenômeno ocorre decorrente da hierarquia usada no sistema de Euler dos eixos de rotação, pois como visto, para uma Ordem de Rotação do tipo XYZ, quando há rotação no eixo Z todos os demais eixos rotacionam juntos variando o movimento, porem o mesmo não ocorre para os eixos Y e X se Z permanecer fixo, ou seja, há uma perda de graus de liberdade do sistema pois os eixos Y e X estarão juntos. Para elucidar de maneira mais clara toma-se o exemplo a seguir.

Supondo que um avião que após realizar uma rotação no eixo Leste/Oeste de 90°, ficando com a “Cabeça” apontada para baixo, com a asa direita apontando para o Leste, a esquerda apontando para o Oeste e a “barriga” para o Sul, tenha então o piloto elaborar uma rotação em torno do eixo vertical no sentido anti-horário. Após uma volta completa, novamente a “barriga” apontando para o Sul, ele tenha decidido rotacionar no eixo Norte/Sul de 90°, é prático pensar que o resultado seria uma rotação suave na vertical, porém usando a matriz de Euler e a hierarquia dos eixos, vimos que na verdade o avião desenvolverá uma rotação horizontal semelhante a anterior, pois perdeu um grau de liberdade de rotação enquanto a rotação de 90° no eixo Leste/Oeste permanecer, com isso haverão orientações que nunca poderão ocorrer, tal efeito é conhecido como Gimbal Lock.

Teoricamente é possível evitar o efeito do Gimbal Lock, por exemplo, registrando todas as sequencias de rotações feitas na ordem em que foram executadas e em seguida alterando os eixos, executar novas rotações pretendidas. Porém em muitos casos, isso se torna inviável, pois é necessário guardar espaço de rotações que vão se acumulando ao longo do tempo e usá-lo a cada nova rotação, o que pode gerar grande esforço computacional e não é uma solução pratica a se fazer.

Outra opção seria representar uma matriz de rotação em relação a posição inicial e a partir disso multiplicar essa matriz por cada nova matriz de rotação aplicada, isso evitar armazenar os dados das rotações anteriores, porém as várias e sucessivas multiplicações podem gerar erros que disformem a realidade física dos dados, ou seja, podem gerar inconsistências nos fenômenos, ainda que se opte por normalizar as essas matrizes, surge outro problema, o custo computacional devido à complexidade em fazê-lo (BIASI; GATTASS 2002, HOAG 1963)

4 QUATÉRNIONS.

Em 1853 foi apresentada ao mundo uma nova ferramenta para análise e solução de problemas envolvendo rotações no espaço tridimensional, tal solução foi proposta pelo matemático Irlandês *William Rowan Hamilton* (1805 -1865) em seu livro intitulado *Lecture on Quaternions*, onde o mesmo apresenta uma generalização dos números complexos simples, segundo a equações abaixo, nessa abordagem é proposto a adição de novos dois termos imaginários *j* e *k.*

$$a + bi$$

$$a + bi + cj + dk$$

É sabido que números complexos podem representar rotações no espaço bidimensional, porém ao se pretender representá-las em três dimensões, Hamilton descobriu a necessidade de acréscimo de mais dois termos complexos para representação do fenômeno, por isso os quatérnios também recebem o nome de números hipercomplexos de dimensão quatro, devido ao fato de se acrescentar mais dois termos imaginários a equação básica dos números complexos, tendo assim 4 termos, incluindo a parte real (VOIG 2018, SYNGE 2004)..

4.1 Números Imaginários, inspirações “imaginárias”.

Por cerca de dez anos Hamilton tentou fazer a modelagem do espaço tridimensional usando números complexos, cuja adição e multiplicação ocorrem no espaço bidimensional, Hamilton desejava encontrar uma generalização dessas equações. Ao que tudo indica a inspiração para propor as equações que definem o tripleto complexo, visto na equação anterior, vieram à mente de Hamilton no dia 16 de outubro de 1843, data a qual Hamilton teria sido convocado para presidir uma reunião na Royal Irish Academy (RIA). Hamilton e sua esposa estavam caminhando ao longo do Royal Canal sobre a Broom Bridge, após sair do observatório de Dunsink em Dublin, quando repentinamente veio em sua mente a resposta para o problema da representação de uma generalização da modelagem do espaço tridimensional usando complexos, Hamilton concluiu que seria necessário a adição de mais um termo complexo para a correta modelagem, e com isso propôs o seu tripleto complexo e se inicia a álgebra dos quatérnios, conforme se segue:

$$i^2 = j^2 = k^2 = ijk = -1 \quad (4.1)$$

4.2 Hipercomplexos de dimensão 4.

Com base na equação 4.1 Hamilton pode determinar outras relações com seu tripleto complexo, conforme se segue.

$$ij = k = -ij$$

$$jk = i = -kj$$

$$ki = j = -ik$$

Como percebido, a multiplicação entre esses termos é não comutativa, tal como rotações no espaço. A matriz a seguir pode representar todas as combinações multiplicativas entre o tripleto complexo de Hamilton.

$$\begin{bmatrix} * & 1 & i & j & k \\ 1 & 1 & i & j & k \\ i & i & -1 & k & -j \\ j & j & -k & -1 & i \\ k & k & j & -i & -1 \end{bmatrix}$$

Onde qualquer multiplicação entre dois imaginários pode ser encontrada identificando o ponto mais inferior que possua o valor de coluna de um dos complexos e o valor de linha do outro, por exemplo, a multiplicação kxk pode ser determinada encontrando o valor que possua coordenada de linha 5, coordenada do *k* mais inferior da primeira coluna da matriz, pelo valor de coluna 5, coordenada do *k* superior da quinta coluna.

Munidos desse conhecimento é possível então utilizar a definição de Hamilton para a generalização dos números complexos em quatro dimensões, os chamados Hipercomplexos de dimensão quatro, também conhecidos como Números de Hamilton ou Quatérnions, sendo essa última a mais utilizada no presente trabalho. A representação geral de um quatérnion q é dada abaixo.

$$q = a + bi + cj + dk$$

A equação acima servirá para representar a forma algébrica de um quatérnion. Pode-se representar ainda o quatérnion como $q = (S, \vec{V})$, onde S representa a parte real e $\vec{V} = (b, c, d)$ representa um vetor com as componentes imaginárias.

As operações de soma e subtração de quatérnions são exatamente iguais as aplicadas em números complexos simples, ou seja, componente a componente. Porém a multiplicação de quatérnions podem ser dadas semelhante a multiplicação de vetores. Tomando-se o exemplo a seguir, em que será ilustrado a multiplicação de dois quatérnions q_1 e q_2.

$$\begin{gathered}
q_1 q_2 = (a_1 + b_1 i + c_1 j + d_1 k)(a_2 + b_2 i + c_2 j + d_2 k) \\
= a_1(a_2 + b_2 i + c_2 j + d_2 k) + b_1(a_2 + b_2 i + c_2 j + d_2 k) + \\
+ c_1(a_2 + b_2 i + c_2 j + d_2 k) + d_1(a_2 + b_2 i + c_2 j + d_2 k) \\
= a_1 a_2 + a_1 b_2 i + a_1 c_2 j + a_1 d_2 k + b_1 a_2 i + b_1 b_2 i^2 + b_1 c_2 ij + b_1 d_2 ik + \\
+ c_1 a_2 j + c_1 b_2 ji + c_1 c_2 j^2 + c_1 d_2 jk + d_1 a_2 k + d_1 b_2 ki + d_1 c_2 kj + d_1 d_2 k^2 \\
= a_1 a_2 - (b_1 b_2 + c_1 c_2 + d_1 d_2) + a_1(b_2 i + c_2 j + d_2 k) + a_2(b_1 i + c_1 j + d_1 k) + \\
+ (c_1 d_2 - d_1 c_2) i + (d_1 b_2 - b_1 d_2) j + (b_1 c_2 - c_1 b_2) k
\end{gathered}$$

Desta forma pode-se representar o produto acima na forma vetorial como se segue.

$$q_1 q_2 = \left(S_1, \overrightarrow{V_1}\right)\left(S_2, \overrightarrow{V_2}\right) = \left(S_1 S_2 - \overrightarrow{V_1}.\overrightarrow{V_2}\, , S_1\overrightarrow{V_2} + S_2\overrightarrow{V_1} + \overrightarrow{V_1} X \overrightarrow{V_2}\right) \qquad (4.2)$$

Através da análise “vetorial” e dos conhecimentos das propriedades de números complexos associados a um quatérnion é possível definir algumas propriedades.

O conjugado de um quatérnion, tal como para números complexos simples, é definido como um valor que possuí o mesmo valor absolutos dos coeficientes, porém com sinal contrário nos termos imaginários. Tomando-se como exemplo, um quatérnion $q = (S, \overrightarrow{V}),$ seu conjugado será $\bar{q} = \left(S, -\vec{V}\right)$.

Outra propriedade que se pode definir analogamente a vetores é o inverso de um número conjugado $q,$ será q^{-1} valendo a relação $q.q^{-1} = 1$.

Assim como no caso matricial, o inverso de um número conjugado só existirá se a sua norma, ou seja, se a magnitude do mesmo for um valor não nulo. E pode-se definir a magnitude de um quatérnion como o produto do mesmo por seu conjugado, logo tem-se (BIASI; GATTAS 2002)

$$|q|^2 = q.\bar{q} = a^2 + b^2 + c^2 + d^2 = S^2 + \left|\vec{V}\right|^2$$

4.3 Modelando Rotações através de quatérnions.

Assim como para conseguir se representar rotações no plano usando números complexos simples é necessário se valer da notação geométrica do complexo, analogamente foi proposto uma notação geométrica para os quatérnions, ou seja, representa-lo a partir de funções de ângulos e sua norma.

4.3.1 Complexos e rotações no Espaço.

A generalização dos números complexos proposta por Hamilton permite a representação de rotações no espaço. Para tal considerando o mesmo princípio adotado para descrever as rotações no plano, através da multiplicação entre números complexos, será usada a multiplicação entre quatérnions para descrição de rotações no espaço.

Assim, um ponto P, cujas coordenadas espaciais são $p_x, p_y\ e\ p_z$, representadas pelo quatérnion $q = \left(0, \vec{P}\right)$, onde se define que a parte real do quatérnion é nula, a fim de representar uma rotação pura. Assim, ao se pretender aplicar uma rotação modelada pelo quatérnion $q_\theta = \left(S, \vec{V}\right)$, onde, tal como no caso plano, será usado esse quatérnion de tal forma que sua norma seja 1, ou seja, um quatérnion unitário para representação da rotação.

Porém, antes de se calcular diretamente a rotação, pode-se analisar a representação de um quatérnion unitário.

$$q_\theta = S + \vec{V} \rightarrow \ |q_\theta| = S^2 + \left|\vec{V}\right|^2 = 1$$

Da equação acima pode-se associar geometricamente os valores $S^2 = \cos(\theta)$ e $\left|\vec{V}\right|^2 = sen(\theta)$. Portanto a representação de q_θ será dada pela seguinte equação.

$$q_\theta = cos\theta + \vec{n}.sen\theta$$

Onde $\vec{n}$ é um vetor unitário, tal que:

$$\vec{n} = \frac{\vec{V}}{\left|\vec{V}\right|^2}.$$

Desta forma é possível calcular o resultado da rotação utilizando um operador rotacional geométrico, aqui denominado $R(q)$, tal que.

$$R(q) = q_\theta q q_\theta^{-1}$$

Como o quatérnion q_θ é unitário pode-se demonstrar que $q_\theta^{-1} = \overline{q_\theta}$, como se segue.

$$q_\theta \overline{q_\theta} = 1 \rightarrow \overline{q_\theta} = \frac{1}{q_\theta}$$

Então pode-se determinar $R(q)$ conforme se segue:

$$R(q) = q_\theta q \overline{q_\theta} = (cos\theta, sen\theta.\vec{n})\big(0, \vec{P}\big)(cos\theta, -sen\theta.\vec{n}) \qquad (3.3.1.2\text{-}5)$$

Para melhor solução do problema, pode-se colocar $\cos^2\theta$ em evidencia e atribuir um vetor $\vec{a}$ que seja igual a $tg\theta.\vec{n}$, portanto a equação acima pode-se ser resolvida conforme se segue.

$$\frac{R(q)}{\cos^2\theta} = (1, \vec{a})\big(0, \vec{P}\big)(0, -\vec{a})$$

$$= (1, \vec{a})\big(0x1 + \vec{P}.\vec{a}\,, -0.\vec{a} + 1.\vec{P} + \vec{P}x\vec{a}\big) = (1, \vec{a})\big(\vec{P}.\vec{a}\,, \vec{P} - \vec{P}x\vec{a}\big)$$

$$= \Big(\vec{P}.\vec{a} - \vec{a}\big(\vec{P} + \vec{P}x\vec{a}\big)\,, \vec{P} - \vec{P}x\vec{a} + \big(\vec{P}.\vec{a}\big)\vec{a} + \vec{a}x\big(\vec{P} - \vec{P}x\vec{a}\big)\Big)$$

$$= \big(\vec{P}.\vec{a} - \vec{a}.\vec{P} - a.\big(\vec{P}x\vec{a}\big)\,, \vec{P} - \vec{P}x\vec{a} + \big(\vec{P}.\vec{a}\big)\vec{a} + \vec{a}x\vec{P} - \vec{a}x\vec{P}x\vec{a}\big)$$

Como $a.\big(\vec{P}x\vec{a}\big)$ é nulo, a equação torna-se:

$$= \big(0\,, \vec{P} - \vec{P}x\vec{a} + \big(\vec{P}.\vec{a}\big)\vec{a} + \vec{a}x\vec{P} - \vec{a}x\vec{P}x\vec{a}\big)$$

$$= \big(0\,, \vec{P} + \vec{a}x\vec{P} + \big(\vec{P}.\vec{a}\big)\vec{a} + \vec{a}x\vec{P} + \vec{a}x\vec{a}x\vec{P}\big)$$

Aplicando a propriedade matricial:

$$\vec{a}x\vec{b}x\vec{c} = (\vec{a}.\vec{c})\vec{b} - (\vec{a}.\vec{b})\vec{c}$$

$$= \big(0\,, \vec{P} + 2.\vec{a}x\vec{P} + \big(\vec{P}.\vec{a}\big)\vec{a} + \big(\vec{a}.\vec{P}\big)\vec{a} - (\vec{a}.\vec{a})\vec{P}\big)$$

$$= \big(0\,, \vec{P} + 2.\vec{a}x\vec{P} + \big(\vec{P}.\vec{a}\big)\vec{a} + \big(\vec{a}.\vec{P}\big)\vec{a} - (\vec{a}.\vec{a})\vec{P}\big)$$

$$= \big(0\,, \vec{P} + 2.\vec{a}x\vec{P} + 2\big(\vec{a}.\vec{P}\big)\vec{a} - (\vec{a}.\vec{a})\vec{P}\big)$$

Elaborando as devidas substituições dos coeficientes tema-se:

$$R(q) = \big(0, \cos^2\theta\,\vec{P} + 2.\cos^2\theta\,.tg\theta.\vec{n}x\vec{P} + 2.\cos^2\theta\,.tg^2\theta.\big(\vec{n}.\vec{P}\big)\vec{n} - \cos^2\theta\,.tg^2\theta(\vec{n}.\vec{n})\vec{P}\big)$$

$$= \big(0, \cos^2\theta\,\vec{P} + 2.sen\,\theta\,.cos\theta.\vec{n}x\vec{P} + 2.sen^2\theta.\big(\vec{n}.\vec{P}\big)\vec{n} - \mathrm{sen}^2\,\theta\,.(\vec{n}.\vec{n})\vec{P}\big)$$

$$= \left(0, (\cos^2 \theta - \text{sen}^2 \theta \,.\, (\vec{n}.\vec{n}))\,\vec{P} + sen\,2\theta \,.\, \vec{n}x\vec{P} + (1 - \cos^2 \theta).\left(\vec{n}.\vec{P}\right)\vec{n}\right)$$

Como o produto escalar do vetor $\vec{n}$ por ele mesmo é igual sua norma, que foi adotada como valor unitário, tem-se:

$$R(q) = \left(0, cos\,2\theta \,.\, \vec{P} + sen\,2\theta \,.\, \vec{n}x\vec{P} + (1 - cos\,2\theta).\left(\vec{n}.\vec{P}\right)\vec{n}\right) \quad (3.21)$$

Essa equação determina as coordenadas do ponto rotacionado após aplicada uma rotação de $\frac{\theta}{2}$.

Portando ao se desejar expressar a rotação sofrida por um ponto, deve-se:

- Escrever as coordenadas do ponto como um quatérnion de parte real nula
- Representar a rotação deseja de um ângulo θ qualquer através do quatérnion $q_\theta = \left(\cos\left(\frac{\theta}{2}\right)\ ,\ sen\left(\frac{\theta}{2}\right).\vec{n}\right)$.
- Achar o valor que satisfaz $R(q) = q_\theta q \overline{q_\theta}$.

Com isso o quatérnion $R(q)$ terá parte real nula, e sua parte imaginária irá representar os efeitos da rotação aplicada ao ponto desejado.

Na aplicação de mais rotações desejadas em diferentes eixos as propriedades aritméticas dos quatérnios possibilitam uma simplificação da análise dos resultados, quando se multiplica, por exemplo por duas rotações associadas aos quatérnions q_1 e q_2, ambos unitários, conforme a equação seguinte.

$$R(q) = q_1.\, q_2.\, q.\, \overline{q_1}.\, \overline{q_2}$$

O valor do produto $q_1.\, q_2$ gera um quatérnion q_3 tal que $\overline{q_3} = \overline{q_1}\,.\,\overline{q_2}$. Portanto a rotação global pode ser sempre ser escrita como o conjunto de rotações feitas em um ponto através de um único quatérnion que compõem os efeito de todas as rotações aplicadas, bastando apenas elaborar a multiplicação entre os sucessivos quatérnions que descrevem cada rotação aplicada e posteriormente utilizar a equação 3.21 para gerar a resposta final do sistema. Para o caso mais genérico a parte real irá fornecer o valor do ângulo rotacionado, através do cosseno desse, e a parte imaginária irá fornecer o vetor que representa a posição após as rotações.

Para ilustrar pode-se utilizar o exemplo de um avião percorrendo em direção ao Norte, após rotacionar sob o eixo Leste/Oeste de 90° irá efetuar uma rotação de mais 90° em torno do eixo vertical. Tais rotações podem ser expressadas pelos quatérnions $q_1 = \left(\cos\frac{\pi}{4}, sen\frac{\pi}{4} . (0,1,0)\right)$ e $q_2 = \left(\cos\frac{\pi}{4}, sen\frac{\pi}{4} . (0,0,1)\right)$, respectivamente, tendo em vista que para um ângulo pretendido de θ a equação utilizará $\frac{\theta}{2}$, tal como já mencionado. Para determinar o resultado final inicialmente deve-se multiplicar os quatérnions q_1 e q_2.

$$q_3 = q_1 . q_2 = \left(\frac{\sqrt{2}}{2}, \left(0, \frac{\sqrt{2}}{2}, 0\right)\right)\left(\frac{\sqrt{2}}{2}, \left(0,0, \frac{\sqrt{2}}{2}\right)\right) = \left(\frac{1}{2}, \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)\right)$$

Que representa uma rotação de $Arccos\left(\frac{1}{2}\right)$, valor angular de 120°, em torno do eixo definido pelo vetor $\left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$. Então basta aplicar a rotação q_3 no avião na posição dada para se determinar o resultado final.

Conforme abordado no presente capítulo, pode-se ver que a parametrização usando quatérnions evita o problema do Gimbal Lock, e, portanto, é possível representar várias rotações sem a perda de graus de liberdade decorrente da sobreposição de eixos, além de possibilitar uma abordagem numericamente mais estável que o uso de ângulos de Euler atrelados com matrizes de rotação (BIASI; GATTAS 2002, SHOEMAKET 1985, KUIPERS 1999, GEOGIEV 2012, JAMBERSI; SILVA 2016)

4.3.2 Interpolações de rotações.

Analisando a interpolação de rotações, ou seja, determinar as rotações intermediarias entre duas rotações limites, por exemplo, para determinar o movimento de um ponto que percorre uma trajetória circular que vai de um ângulo de 0° à 180°, em outras palavras determinar o conjunto de rotações sequenciais que formam a rotação completa de 180°, para tal calcula-se o conjunto de quatérnions intermediários que representaram o quatérnion associado a rotação global.

A representação de um quatérnion no IR^4, espaço vetorial de quarta dimensão, forma uma Hiperesfera, esfera na quarta dimensão, portanto para determinar um quatérnion intermediário que represente uma interpolação adequada entre duas rotações, contrariando o senso intuitivo, não será feita uma interpolação linear, dado que por se tratar de uma Hiperesfera, essa reta seria secante a curva. Para a adequada interpolação é necessário usar a curva na Hiperesfera que liga ambas as rotações, que nesse caso, será um círculo.

O método adequado então é utilizar a chamada Interpolação Linear Esférica, do inglês Spherical Linear Interpolation (SLERP), que consiste em gerar um círculo que contenha todos os quatérnios que representem as rotações intermediarias entre duas rotações principais, onde todos os quatérnions devem ser unitários. A equação que permite então a modelagem de uma rotação intermediaria q_3 qualquer, correspondente a uma angulação intermediária $\theta,$ em função de duas rotações em termos dos quatérnions unitários q_1 e q_2 é dada como:

$$q_3 = \frac{sen(\lambda - \theta)}{sen\lambda}\, q_1 + \frac{sen\theta}{sen\lambda}\, q_2$$

Onde λ representa o descolamento entre q_1 e q_2, que pode ser calculado através da relação.

$$q_2. q_1 = |q_1||q_2|. cos\lambda \rightarrow cos\lambda = 1. q_1. q_2 \rightarrow \lambda = Arccos(q_1. q_2)$$

Sendo assim pode-se determinar o coeficiente adimensional $\alpha = \frac{\theta}{\lambda}$, que representa fração do caminho onde se localiza o quatérnion q_3 em relação a q_1 e q_2. Portanto, pode-se reescrever a expressão da seguinte forma.

$$q_3 = \frac{sen\big((1 - \alpha)\lambda\big)}{sen\lambda}\, q_1 + \frac{sen\alpha. \lambda}{sen\lambda}\, q_2$$

Através dessa equação é possível representar todas as rotações intermediarias variando apenas o valor de $\alpha,$ tendo conhecido os demais fatores, para obter os quatérnios unitários q_3 que representam o conjunto de todas as rotações sequenciais possíveis. É válido ressaltar ainda que a parametrização fornece dois caminhos possíveis a se percorrer dentro do círculo que representa os possíveis quatérnions de rotações intermediárias, porém, segundo a literatura sugere, deve-se optar pelo menor caminho, ou seja, o menor valor do coeficiente $\lambda,$ por exemplo no caso em que for de 60°, há a opção do caminho relacionado a 300°, porém conforme mencionado anteriormente, deve-se optar pelo valor de 60°, pois esse está relacionado ao menor caminho na interpolação SLERP. Há porem o caso em que o ângulo seja de 180°, nesse caso a escolha será arbitrária, pois conduzirá ao mesmo resultado final (BIASI; GATTAS 2002, SHOEMAKET 1985, GEOGIEV 2012, JAMBERSI; SILVA 2016)

5 SIMULAÇÃO USANDO MATLAB.

O presente capitulo tem por objetivo demonstrar o movimento de uma particula através da simulação utilizando apenas as variáveis cinemáticas, ou seja, não se estará levando em consideração as causas ou consequências correlatas ao movimento, apenas simulando o caso ideal de um IMU que determina a trajetória ao longo do tempo em relação a um referencial global, tendo com entrada as acelerações no referencial local.

Foi utilizado o software MATLAB versão R2017a para simular o comportamento do corpo em uma trajetória livre de forças externas ou efeitos dissipativos de qualquer origem, para então simular o IMU. O IMU é posto no corpo e mede os valores de aceleração e velocidade em relação ao referencial local, aqui descrito como o próprio corpo e, para o conjunto de hipóteses desta formulação, também será o centro de massa, já que tanto o corpo quanto o IMU terão as dimensões desprezadas. A figura 5.1 a seguir mostra a correlação das variáveis de entradas e saída para a análise no modelo de Blocos de ligação, o mesmo utilizado pelo MATLAB na modelagem feita através da ferramenta SIMULINK.

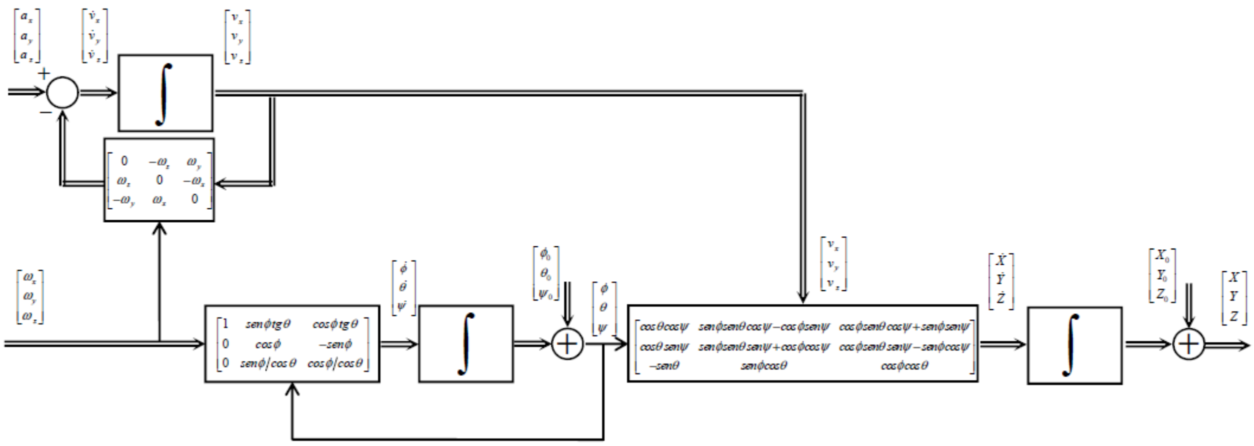


Figura 5.1 Representação em blocos do equacionamento das variáveis de movimento translacional e rotacional.

O diagrama acima será utilizado no SIMULINK para determinação da trajetória do corpo em relação ao referencial global externo ao mesmo.

Esse diagrama representa as equações de Euler-Newton para o movimento do de uma particula. No SIMULINK-MATLAB é possível representar os conjuntos de equações matriciais e suas dependências dentro de campos encapsulados (Blocos de modelagem). A seguir será descrito as funções encapsuladas, bem como sua representação via SIMULINK. O código do MATLAB, bem como o SIMULINK estarão disponíveis no Apêndice B.

5.1 Objetos da simulação no SIMULINK.

Como mencionado anteriormente é possível "encapsular" o conjunto de funções, bem como sua representação visual, dentro de blocos no SIMULINK, cada bloco possui uma especificação para o modelo como representado na figura 5.2 a seguir.

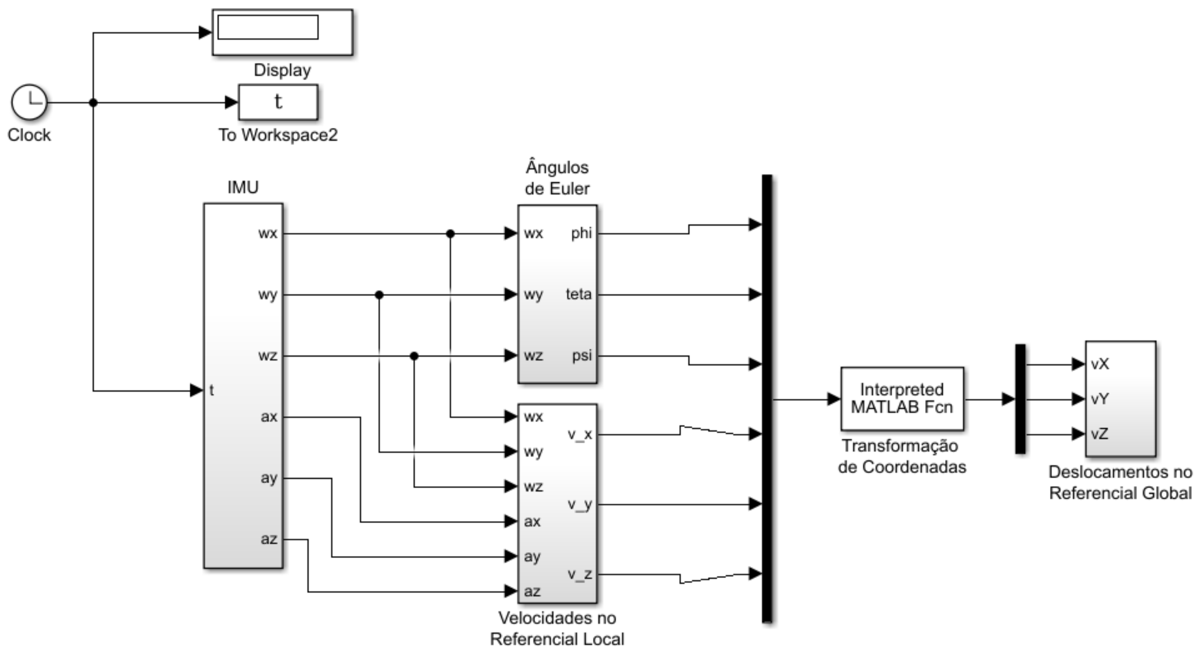


Figura 5.2 Representação em blocos do equacionamento gerada via SIMULINK.

Adiante serão vistos o conteúdo e a descrição de cada função em cada trecho do conjunto acima.

5.1.1. Cronometro.

O conjunto de objetos abaixo, figura 5.3, é responsável pela contagem do tempo de simulação do programa.

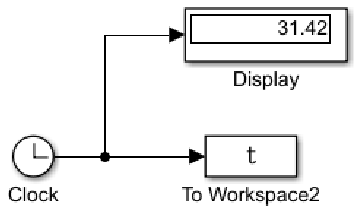


Figura 5.3 Objeto cronometro da simulação via SIMULINK.

O display informa o valor em segundos para o tempo de execução do programa, e um variável vetorial contendo informações do tempo é exportada para o programa.

5.1.2. Blocos de entradas do sistema.

O conjunto de objetos abaixo, figura 5.4, representa a IMU responsável pela detecção das variáveis de entrada, que nesse caso, é formada por um giroscópio e por um acelerômetro que irão detectar velocidades angulares e acelerações respectivamente, também é mostrado o interior do IMU que é modelado pelo SIMULINK, a partir desses valores o programa fornecerá os valores de saída esperados e que serão computados pelo programa.

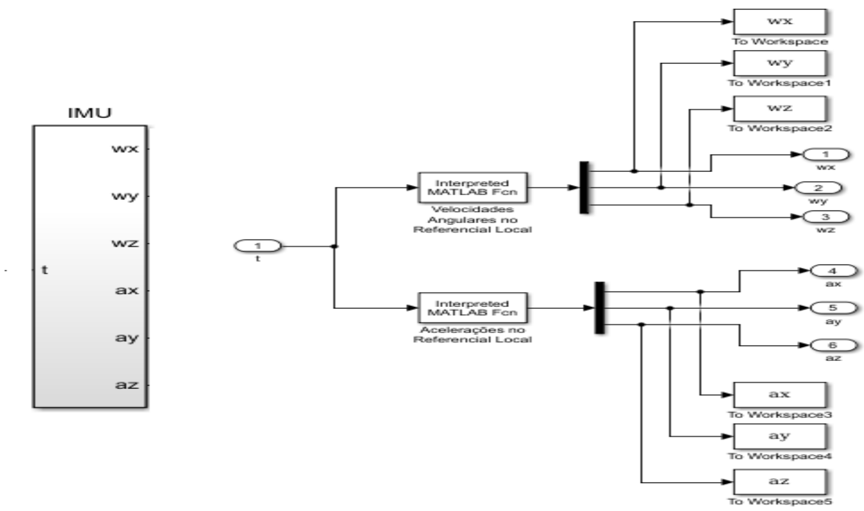


Figura 5.4 Representação em blocos das entradas detectadas pelo giroscópio e pelo acelerômetro (IMU) e seu conteúdo modelado pelo SIMULINK.

5.1.3. Ângulos de Euler.

O objeto a seguir, figura 5.5, é responsável pelos cálculos para obtenção dos ângulos de Euler a partir das velocidades angulares, também e exposto o mecanismo interno para o cálculo.

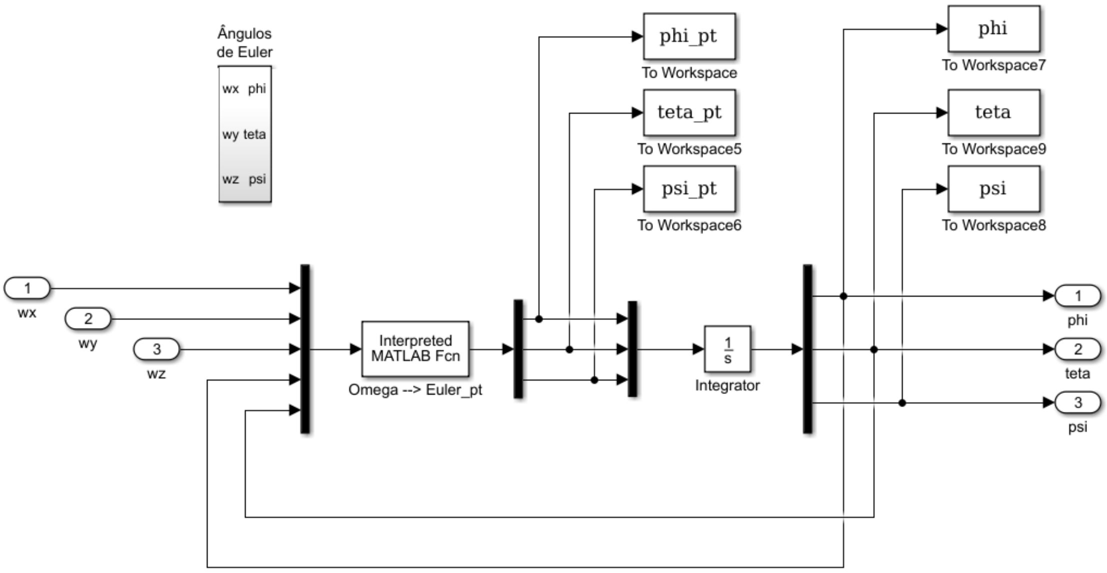


Figura 5.5 Representação em blocos das entradas em velocidade angular, e do interior do bloco, programa de cálculo através do SIMULINK.

Como pode-se observar a IMU recebe as velocidades angulares detectadas através do giroscópio e usa uma função de para converte-las para os valores das taxas de variação temporais dos ângulos de Euler, através da função Euler_pt (olhar apêndice B). Como visto anteriormente no capitulo 3, os valores do vetor velocidade angular no referencial local Ω podem ser utilizado para determinação das derivadas dos ângulos de Euler segundo a equação abaixo.

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & sen\phi tg\theta & \cos\phi tg\theta \\ 0 & \cos\phi & -sen\phi \\ 0 & \frac{sen\phi}{cos\theta} & \frac{\cos\phi}{cos\theta} \end{bmatrix} \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix}$$

Através da equação acima e conhecendo os valores das velocidades angulares nos três eixos, a partir da matriz de transformação encontra-se as taxas de variação. Porém como se pode perceber, a equação matricial necessita do prévio conhecimento dos ângulos de Euler relacionado a rotação no eixo y, θ, e no eixo x, ϕ, na qual o SIMULINK consegue resolver através do circuito de realimentação mostrado no canto inferior da figura anterior, ou seja, os valores gerados como resposta são reinseridos para se alcançar o valor através de um processo iterativo.

Com isso o bloco de função gera as taxas de variação dos ângulos de Euler, representadas por *phi_pt, psi_pt* e *theta_pt* que são exportadas para uso posterior e integradas para se obter os valores dos ângulos de Euler.

$$a_p^B = \alpha_B^B x r_p^B + w_B^B x\left(w_B^B x r_p^B\right) + R_B \ddot{d}_B^G$$

5.1.4. Velocidades no referencial local.

O seguinte bloco tem por função receber os valores das velocidades angulares, detectadas pelo giroscópio, e das acelerações, detectadas pelo acelerômetro, e com isso determinar os valores das velocidades no referencial local. Também é representado o conteúdo do bloco figura 5.6.

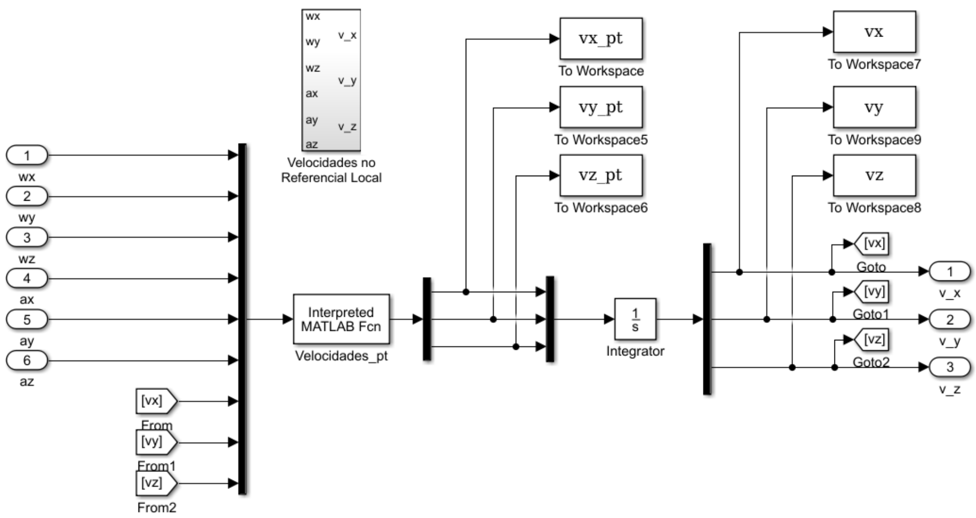


Figura 5.6 Representação em blocos das entradas em aceleração linear, e do interior do bloco, programa de cálculo.

Através do bloco de função o programa implementa a função *Velocidades_pt* que calcula as derivadas das velocidades locais, para isso, o bloco recebe os valores das velocidades angulares, acelerações e velocidades locais, sendo esse último fornecido através do circuito de realimentação, que novamente o MATLAB computa através de iteração. A função *Velocidades_pt* se vale da equação 3.18 do capitulo 3. Que pode ser reescrita da seguinte forma para o caso em que $\ddot{d}_B^G$ é nulo, ou seja, não há aceleração devida a translação.

$$\dot{v}_{xyz} = a_{xyz} - w_{xyz} x v_{xyz}$$

Com essa expressão e através do circuito de realimentação gera-se a derivada da velocidade local com respeito ao tempo, que então é exportada para uso posterior do programa e integrada para se obter as velocidades com respeito ao referencial local nos três eixos coordenados.

5.1.5. Vetor posição no referencial global.

Com isso a parte esquerda do circuito explicitado na figura 5.2 gera como resposta as variáveis de velocidades e ângulos de Euler para serem aquisitados pela função *Loc2Glob* que efetua uma transformação de coordenadas locais para globais. O programa se vale da equação abaixo.

$$v_p^G = R^T v_p^B$$

Que como visto no capitulo 3, representa a obtenção do vetor de velocidades lineares com respeito ao referencial global obtido ao multiplicar o vetor de velocidades lineares no referencial local, no objeto, pela matriz de rotação dos ângulos de Euler. Esses valores são entradas para o bloco final de solução, segue sua representação de seu conteúdo interno, Figura 5.7.

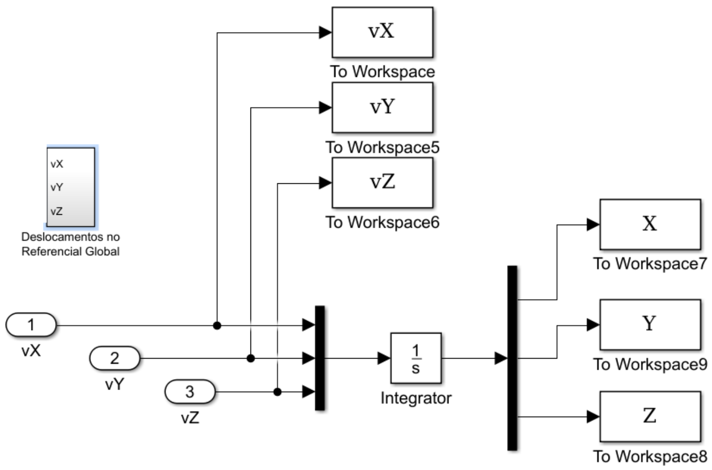


Figura 5.7 Representação em blocos das velocidades lineares no referencial local, e do interior do bloco, programa de cálculo.

Como se pode perceber o bloco recebe os valores das velocidades lineares no referencial local e as integra para obter o vetor posição da partícula em relação ao referencial global.

5.2. Teste de Simulação.

Agora será testado o programa simulador feito utilizando o software MATLAB e a ferramenta SIMULINK, para tal se definirá uma trajetória que se espera que o objeto descreve quando submetido a determinados valores de aceleração e velocidades angulares aquisitados pelo IMU, as trajetórias serão elaboradas em ambos os eixos ordenados, tomados como eventos individuais não correlatados, conforme se verá.

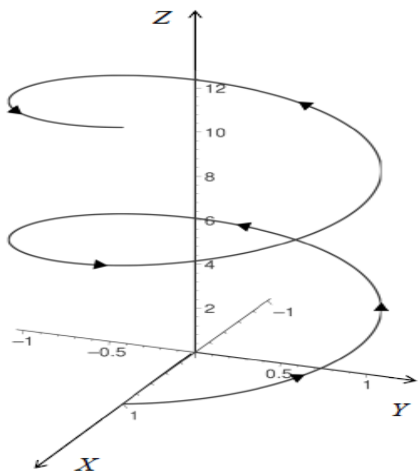
5.2.1. Trajetórias.

Como exemplo para simulação, optou-se por uma trajetória helicoidal elaborada pela particula feita separadamente em cada eixo, o programa então deve retornar esses percursos quando as condições de movimento no referencial local aquisitados pelo IMU, velocidade angulares e acelerações lineares, forem tais que descrevam esses comportamentos segundo algumas premissas previas.

5.2.2. Helicoide.

A helicoide é uma curva que descreve uma rotação em um plano acrescida de um deslocamento constante no eixo perpendicular a esse plano, de maneira que as velocidades angulares nos eixos que interceptam o plano são nulas, e velocidade angular perpendicular ao plano é constante (SCIENTIFIC AMERICAN 2005), a representação de uma curva helicoidal é expressa na figura 5.8.

Figura 5.8 Helicoide com base em XY.



Fonte: Notas de aula, Mauro Speranza Neto.

Tomando-se uma helicoide tal qual a figura anterior, isto é, com velocidades angulares em x e y nulas, e deslocamento translacional em z, tem-se que o vetor de velocidades angulares será:

$$\vec{\Omega} = (0,0,w_z)$$

A parametrização da curva permite determinar as velocidades conforme representação na figura 5.9, onde é mostrada a helicoide vista de cima para determinação das velocidades no plano XY.

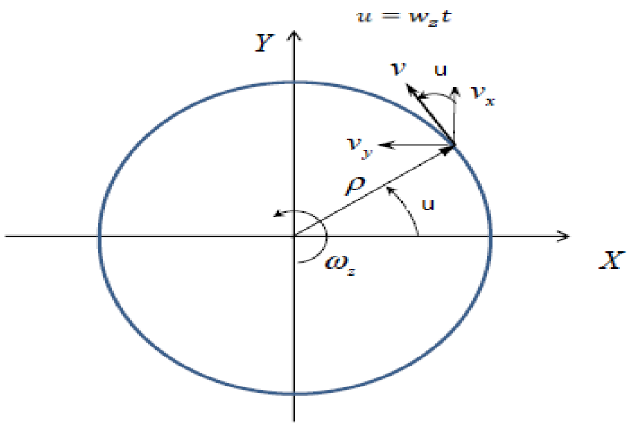


Figura 5.9 Representação dos vetores velocidade associados a base de uma helicoide circular.

Através da figura 5.9 pode-se determinar as velocidades em x, y e z no referencial local para a helicoide em z:

$$v_x = v\ cosu = v\cos(w_z t) = w_z \rho \cos(w_z t)$$

Fonte: Notas de Aula Mauro Speranza Neto.

$$v_y = v\ senu = v\,\mathrm{sen}(w_z t) = w_z \rho\,\mathrm{sen}(w_z t) \qquad (5.1)$$

$$v_z = Pulso(t = 0\ até\ t = fim)$$

De maneira análoga pode-se determinar as equações de velocidade para um movimento helicoidal semelhante nos eixos x e y, respectivamente, conforme se segue.

$$v_y = v\ cosu = v\cos(w_x t) = w_x \rho \cos(w_x t)$$

$$v_z = v\ senu = v\,\mathrm{sen}(w_x t) = w_x \rho\,\mathrm{sen}(w_x t) \qquad (5.2)$$

$$v_x = Pulso(t = 0\ até\ t = fim)$$

$$v_x = v\ cosu = v\cos(w_y t) = w_y \rho \cos(w_y t)$$

$$v_z = v\ senu = v\,\mathrm{sen}(w_y t) = w_y \rho\,\mathrm{sen}(w_y t) \qquad (5.3)$$

$$v_y = Pulso(t = 0\ até\ t = fim)$$

Como visto no capitulo 3, a equação matricial abaixo possibilita a determinação do vetor aceleração para a helicoidal no eixo Z.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} + \begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

Que resultará na equação 5.2 representada como se segue.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \end{bmatrix} + \begin{bmatrix} -v_y w_z \\ v_x w_z \\ 0 \end{bmatrix} = \begin{bmatrix} -w_z^2 \rho sen(w_z t) - w_z^2 \rho sen(w_z t) \\ w_z^2 \rho sen(w_z t) + w_z^2 \rho sen(w_z t) \\ 0 \end{bmatrix} = \begin{bmatrix} -2w_z^2 \rho sen(w_z t) \\ 2w_z^2 \rho sen(w_z t) \\ 0 \end{bmatrix} \quad (5.4)$$

Semelhantemente, pode-se determinar as acelerações que serão captadas pelo giroscópio relativas aos movimentos helicoidais em x e y, respectivamente.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} 0 \\ 2w_x^2 \rho sen(w_x t) \\ -2w_x^2 \rho sen(w_x t) \end{bmatrix} \quad (5.5)$$

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = \begin{bmatrix} 2w_x^2 \rho sen(w_x t) \\ 0 \\ -2w_x^2 \rho sen(w_x t) \end{bmatrix} \quad (5.6)$$

Com esse conjunto de equações pode-se determinar as variáveis de entrada necessárias para cada simulação relativa ao movimento em cada eixo para simular os dados captados pelo IMU.

5.4. Determinação das variáveis de entrada.

No SIMULINK pode-se representar as variáveis de entradas que gerarão a curva pretendida. Tal como visto na figura 5.2 o bloco que representa as velocidades angulares detectadas pelo giroscópio, e as acelerações, detectadas pelo acelerômetro, terão como valores a seguinte representação demonstrada na figura 5.10, onde é evidenciado os valores que são aquisitados pelo IMU.

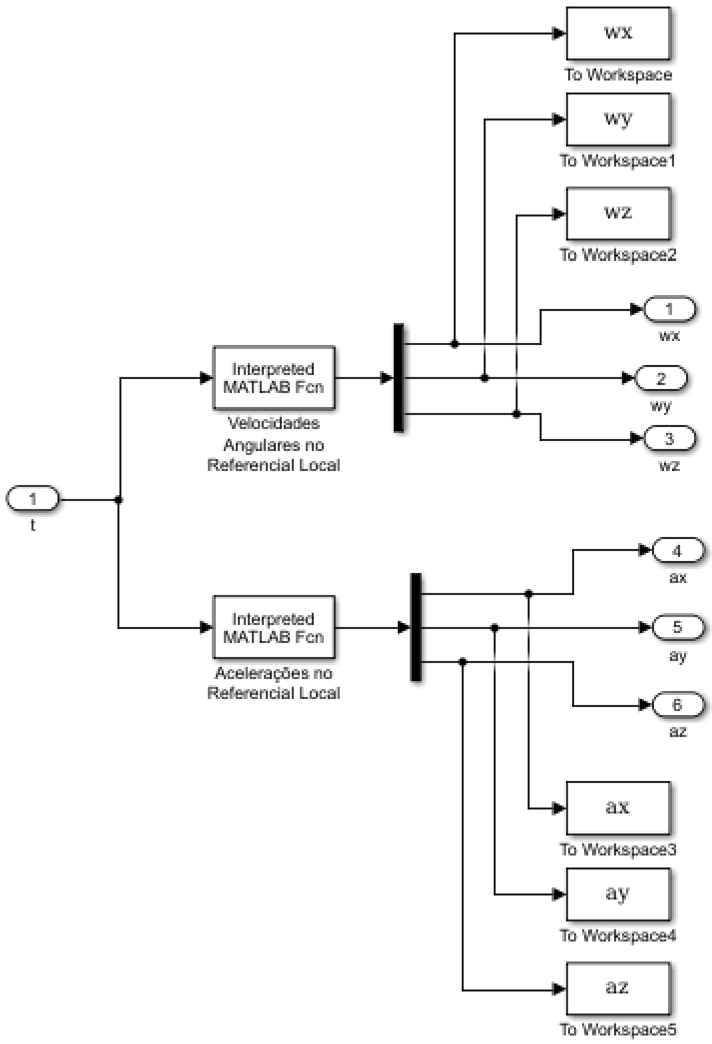


Figura 5.10 – Representação das variáveis aquisitadas pelo IMU através do SIMULINK.

Os valores destas variáveis serão gerados para a simulação através das funções definidas no programa (vide Apêndice B), tais funções são *wx_wy_wz()* e *ax_ay_az()* que geram os valores para as velocidades angulares e acelerações lineares em cada simulação, conforme as características em cada uma, para assim ser um programa mais genérico.

A aceleração em cada simulação terão uma componente nula, para a helicoide em Z, a velocidade nesse eixo será nula em todo o tempo de movimento, porém terá dois picos intermediários, degraus instantâneos, que representam, respectivamente, a passagem da velocidade em Z de nula para o valor constante e posteriormente a desaceleração instantânea, retornando fazendo a velocidade retornar para um valor nulo, representando o início e o fim do movimento em Z, e da mesma maneira espera-se que ocorra com as simulações de movimentos helicoidais em X e Y, conforme representado na figura 5.12.

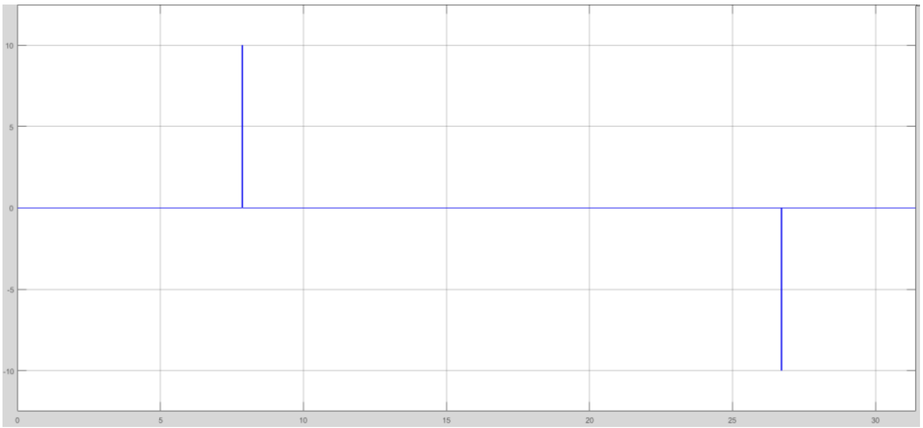


Figura 5.12 Curva da aceleração linear em X, Y e Z com respeito ao tempo.

Com os resultados das entradas já determinadas o IMU deverá ser capaz de, a partir desses valores, gerar a posição da partícula ao longo do tempo em relação ao referencial global tal como visto na secção 4.1 do presente capítulo, e através do MATLAB o programa irá plotar gráficos das variáveis cinemáticas para análise e confirmação daquilo que se espera para cada trajetória em cada uma das simulações, segundo as condições iniciais especificadas e condições de contorno que possam estar atreladas ao movimento, como será visto na próxima seção.

5.5. Analise dos resultados.

Através dos resultados gráficos gerados pelo SIMULINK-MATLAB será possível analisar a conformidade dos dados e determinar se o programa gera resultados que representem fidedignamente a realidade física esperada dentro das hipóteses adotadas. O MATLAB plota os gráficos relativos aos valores temporais obtidos via SIMULINK, no programa há um modulo especifico para cada simulação que recebe os dados e gera os gráficos das grandezas físicas associadas durante cada simulação, porém devem ser feitas separadamente (vide Apêndice B).

5.6 Trajetória de movimento.

No MATLAB serão plotados os gráficos que descrevem a trajetória da partícula em cada caso, helicoide em X, Y e Z conforme será visto adiante.

5.6.1 Trajetória de movimento em Z.

A seguinte figura, figura 5.13, é plotada pelo MATLAB-SIMULINK e fornece a trajetória da partícula em relação ao referencial global, para a helicoide em Z.

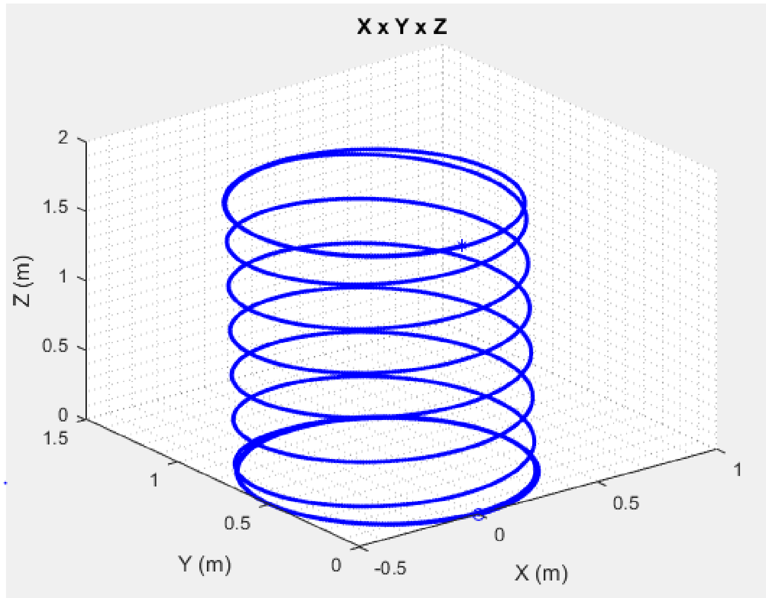


Figura 5.13 Curva de deslocamento espacial da partícula gerada via simulação pelo MATLAB-SIMULINK, para a helicoide em Z.

Na figura 5.13, tal como se esperava, é gerada uma trajetória helicoidal em relação a um referencial global no espaço tridimensional. É possível verificar também que as velocidades angulares, cujos vetores são paralelos ao plano XY são nulas, para que haja rotação apenas em relação ao eixo Z. Na figura 5.14 a seguir é demostrada as vistas das curvas em relação aos planos XY, XZ e YZ, respectivamente.

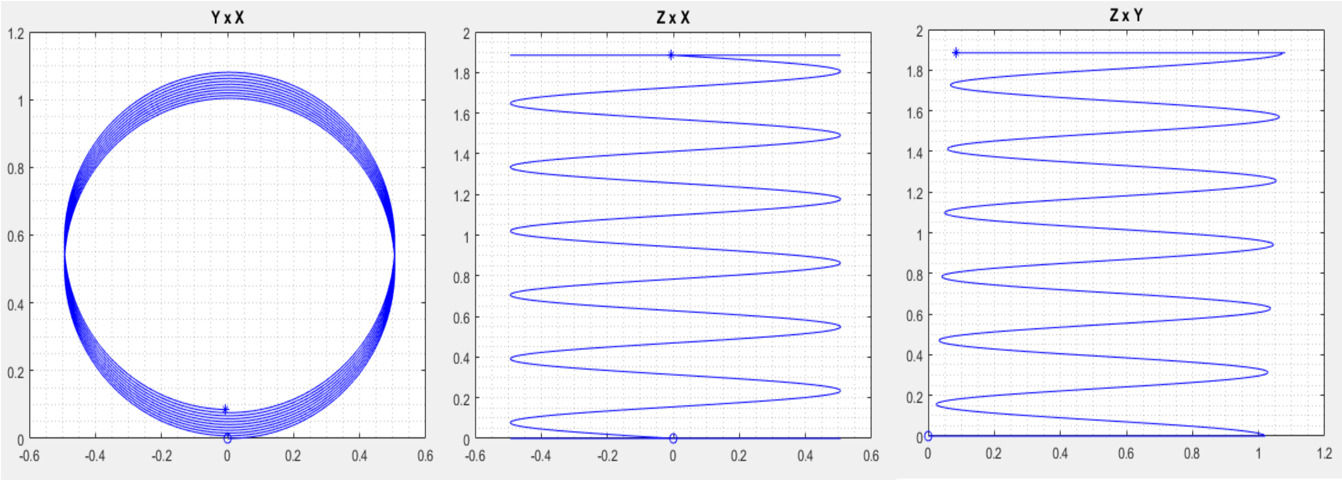


Figura 5.14 Vistas em cada plano, relativo a cada eixo, da trajetória helicoidal do movimento da partícula em Z, geradas através do SIMULINK.

A figura 5.14 demonstra a validade dos dados e o comportamento esperado para o deslocamento do corpo ao longo do tempo em relação ao referencial global, bem como para as velocidades angulares, segundo a formulação proposta. A figura 5.15 a seguir representa os valores das posições associadas a cada um dos três eixos em relação ao tempo.

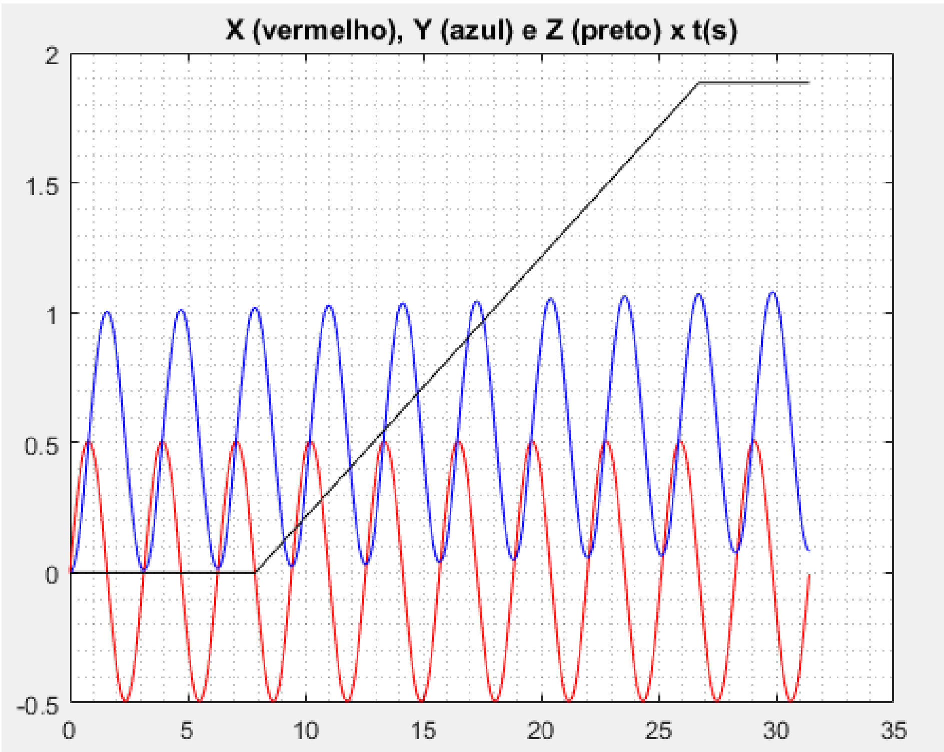


Figura 5.15 Posição da partícula em relação a cada eixo no referencial global ao longo do tempo para a helicoide em Z, geradas através do SIMULINK.

Conforme descrito no código (vide Apendice B) a partícula só irá se deslocar no eixo vertical após duas voltas e meia no plano e ao chegar ao ápice do deslocamento vertical, este irá gerar mais uma revolução e meia para todas as simulações, como visto no gráfico, também se pode perceber que os deslocamentos em X e Y são funções senoidais, ou seja, formam uma curva circular no plano XY, conforme esperado para a helicoide em Z, onde esses movimentos descrevem revoluções nesse plano. A rampa que descreve o deslocamento em Z está dentro do esperado, pois demonstra a variação linear do deslocamento ao longo do tempo, dado que, como foi proposto, este sobe sem efeitos resultantes de aceleração, ou seja, há uma aceleração tal que somada com a gravidade o resultado final será nula ao longo de todo o movimento. Pode-se perceber através dos valores de máximo e mínimo das curvas senoidais, que o círculo não é centrado na origem dos eixos, que seu centro, na verdade, está localizado ao longo da reta (0, 0.5, Z) conforme já mostrado no plote das zonas vista em cada plano.

5.6.2 Trajetória de movimento em Y.

A Figura 5.16 é plotada pelo MATLAB-SIMULINK e descreve a Helicoide em Y do movimento da partícula para a segunda simulação.

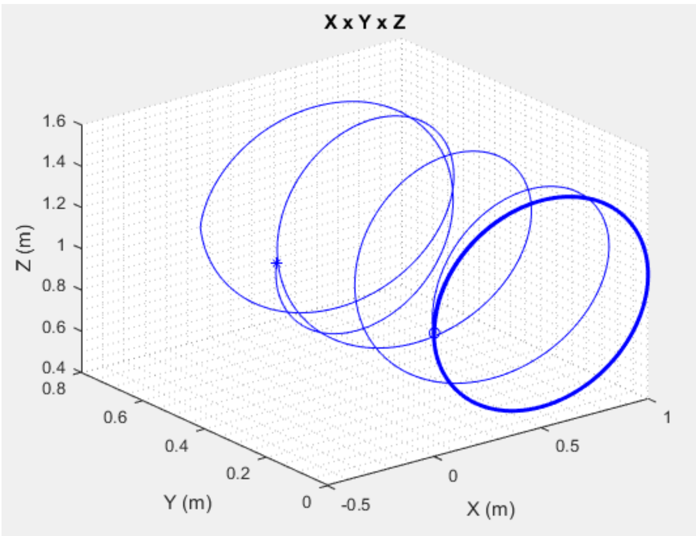


Figura 5.16 Curva de deslocamento espacial da partícula gerada via simulação pelo MATLAB-SIMULINK, para a helicoide em Y.

Novamente há a comprovação do resultado esperado para o programa na descrição da curva de movimento, como visto na figura 5.16. A figura 5.17 demonstra as vistas das curvas em relação aos planos XY, YZ e XZ, respectivamente.

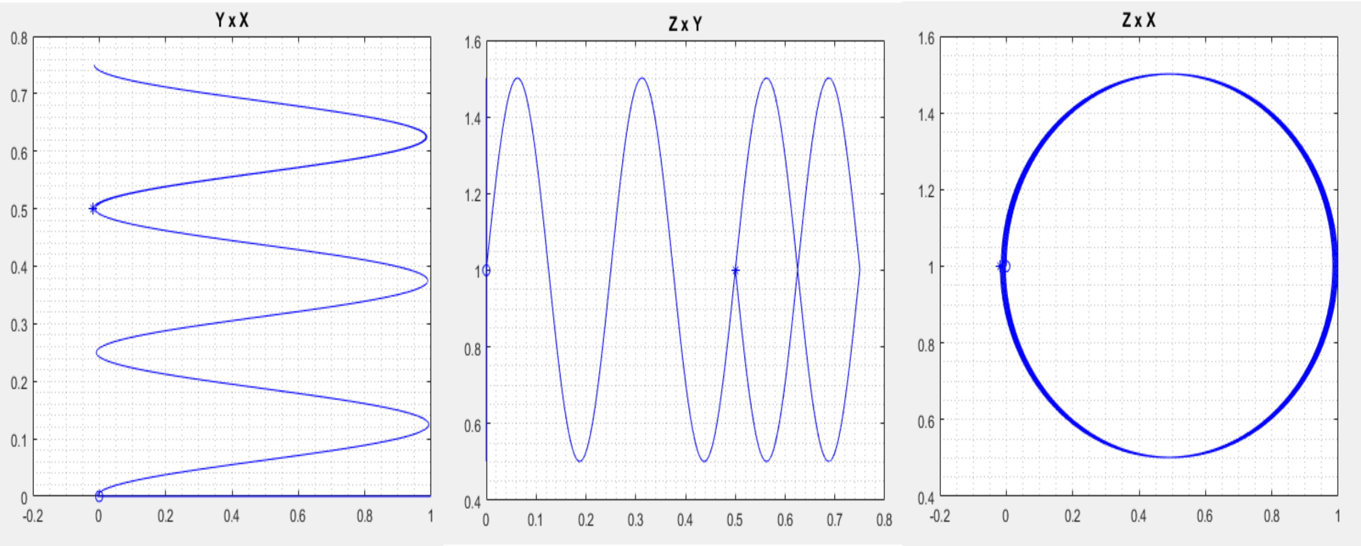


Figura 5.17 Vistas em cada plano, relativo a cada eixo, da trajetória helicoidal do movimento da partícula em Y, gerados através do SIMULINK.

A figura 5.17 demonstra a validade dos dados e o comportamento esperado para o deslocamento do corpo ao longo do tempo em relação ao referencial global, bem como para as velocidades angulares, segundo a formulação proposta. A figura 4.18 a seguir representa os valores das posições associadas a cada um dos três eixos em relação ao tempo para a segunda simulação da helicoide em Y.

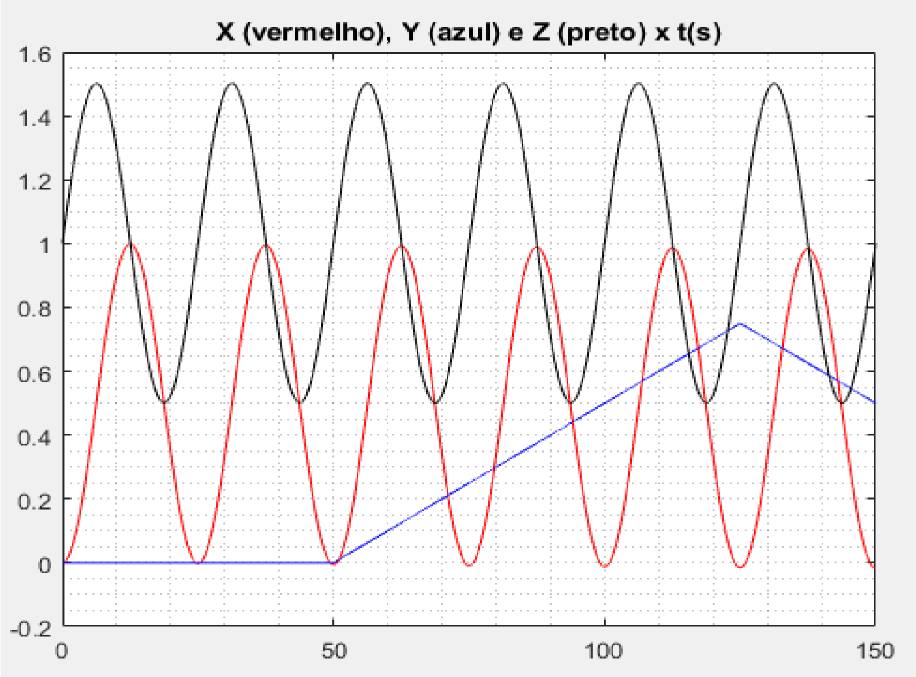


Figura 5.18 - Posição da partícula em relação a cada eixo no referencial global ao longo do tempo para helicoide em Y, gerada através do SIMULINK.

Conforme descrito no código (vide Apendice B) a partícula só irá se deslocar no eixo vertical após uma volta no plano e ao chegar em determinada altura o mesmo começa a descer suavemente. Novamente se demonstra, conforme esperado, os deslocamentos atrelados a X e Z serão senoides, conforme parametrização da helicoide em Y. E a trajetória em Y, segundo o especificado, deverá ser fixa durante a primeira volta, reta horizontal, e em seguida subir constantemente e posteriormente declinar constantemente, região simbolizada por um trapezoide no gráfico. E pode-se também determinar as coordenadas do centro da helicoide através dos valores médios de X e Z, que resultará na equação da reta (0.5, Y, 0.75).

5.6.3 Trajetória de movimento em X.

A Figura 5.19 é plotada pelo MATLAB-SIMULINK e descreve a Helicoide em X do movimento da partícula para a segunda simulação.

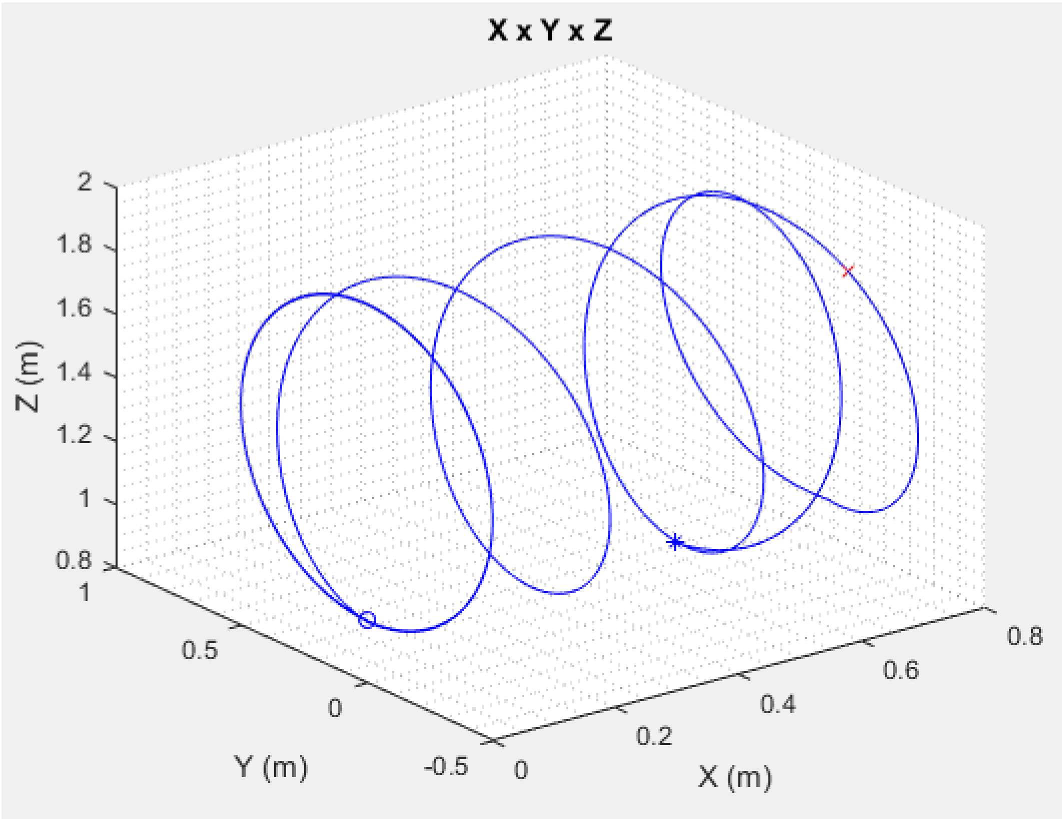


Figura 5.19 Curva de deslocamento espacial da partícula gerada via simulação pelo MATLAB-SIMULINK, para a helicoide em X.

Novamente há a comprovação do resultado esperado para o programa na descrição da curva de movimento, como visto na figura 4.19. A figura 4.20 demonstra as vistas das curvas em relação aos planos XY, ZX e YZ, respectivamente.

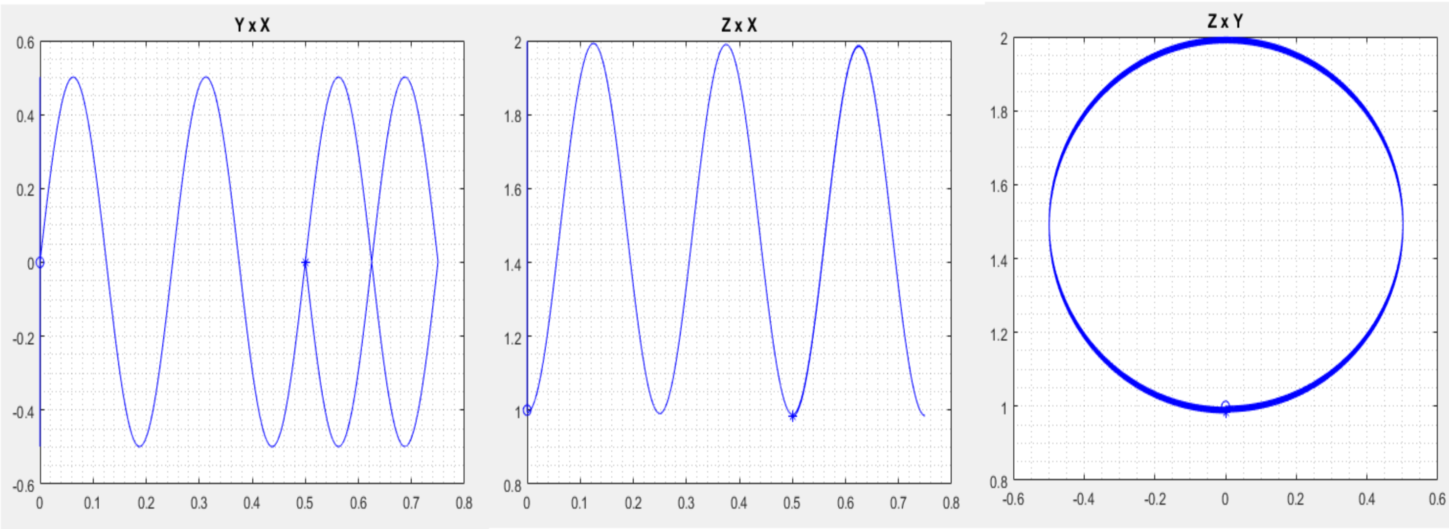


Figura 5.20 Vistas em cada plano, relativo a cada eixo, da trajetória helicoidal do movimento da partícula em X, gerados através do SIMULINK.

A figura 5.20 novamente demonstra a validade dos dados e o comportamento esperado para o deslocamento do corpo ao longo do tempo em relação ao referencial global, porém, para a helicoide em X. A figura 5.21 a seguir representa os valores das posições associadas a cada um dos três eixos em relação ao tempo para a segunda simulação da helicoide em X.

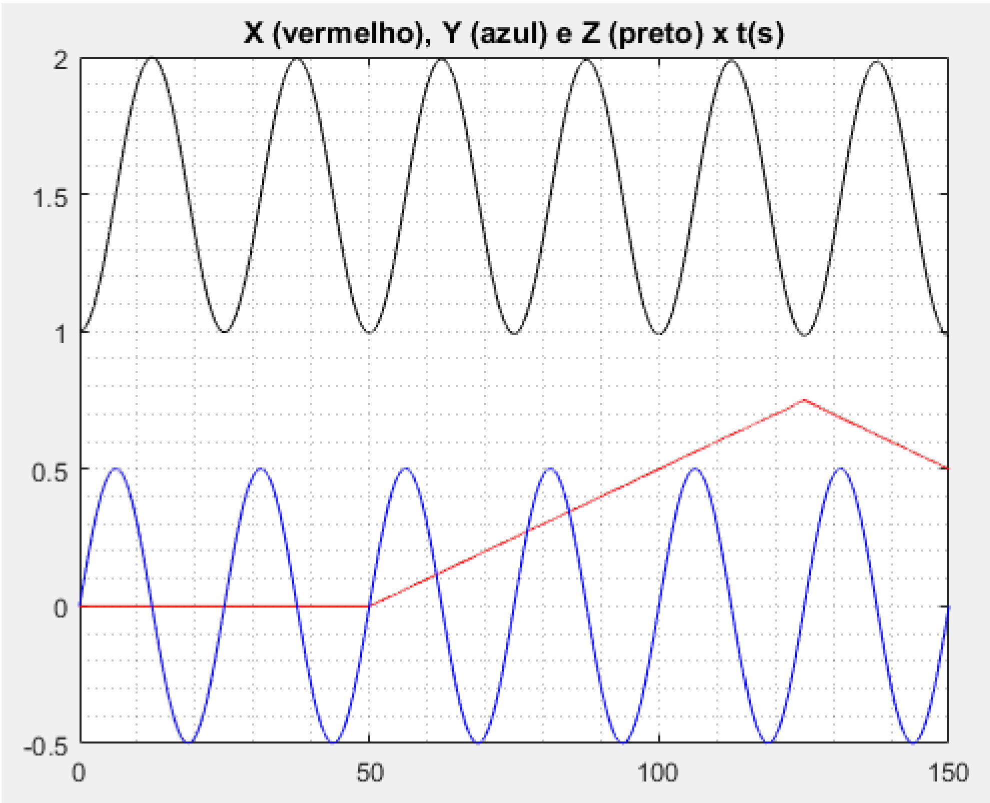


Figura 5.21 - Posição da partícula em relação a cada eixo no referencial global ao longo do tempo para helicoide em X, gerada através do SIMULINK.

Conforme descrito no código (vide Apendice B) a partícula só irá se deslocar no eixo vertical após duas voltas no plano YZ e ao chegar em determinada altura o mesmo começa a descer suavemente. Novamente se demonstra, conforme esperado, os deslocamentos atrelados a base, Y e Z, como senoides, conforme parametrização da helicoide em X. E a trajetória em X, segundo o especificado, deverá ser fixa durante a primeira volta, reta horizontal, e em seguida subir constantemente e posteriormente declinar constantemente, região simbolizada por um trapezoide no gráfico, assim como foi na trajetória helicoidal em Y. Analogamente pode-se determinar a equação da reta que intercepta o centro da helicoide através dos valores médios de Y e Z, que será (X, 0, 1.5).

5.7 Velocidades da partícula no espaço.

As velocidades lineares no referencial local para o movimento da partícula devem obedecer ao comportamento descrito pelas equações 5.1, 5.2 e 5.3 para cada simulação, ou seja, devem ser velocidades senoidais com amplitudes iguais ao produto do raio da base da trajetória helicoidal pela velocidade angular nos relativos eixos ortogonais às bases para cada simulação e devem ser constantes ao longo desse eixo, levando em consideração os comportamentos adotados na programação.

5.7.1 Gráficos das Velocidades da partícula para a simulação em Z.

O gráfico que é representado pela figura 5.22, descreve a curva das velocidades em Z gerada através da simulação, no gráfico também é plotado o deslocamento nesse eixo.

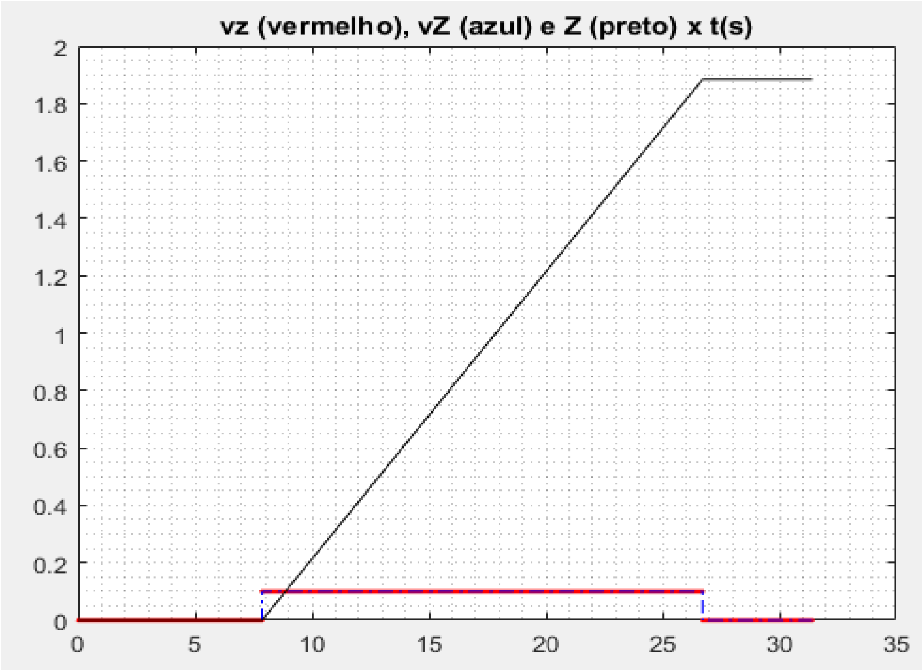


Figura 5.22 Curva de deslocamento para movimento helicoidal em Z, e velocidades no referencial global e local relativo ao eixo Z, gerada pela simulação no SIMULINK.

Como se pode observar pelo gráfico a velocidade da particulaserá constante ao longo do movimento no eixo Z, com a devida ressalva mencionada anteriormente. No gráfico são plotadas as velocidades relativas ao referencial do corpo, local, e ao referencial global, e como esperado não há alteração nesse valor, pois, como o corpo está livre de rotações nos demais eixos, não há alteração no sentido do eixo z, relativo ao referencial local, portanto ambas as velocidades terão o mesmo valor, representado no gráfico como curvas sobrepostas.

A figura 5.23 descreve a velocidade no eixo Y no referencial global, em azul, e local, em vermelho, que como já discutido, deverá ter um comportamento senoidal para essa simulação.

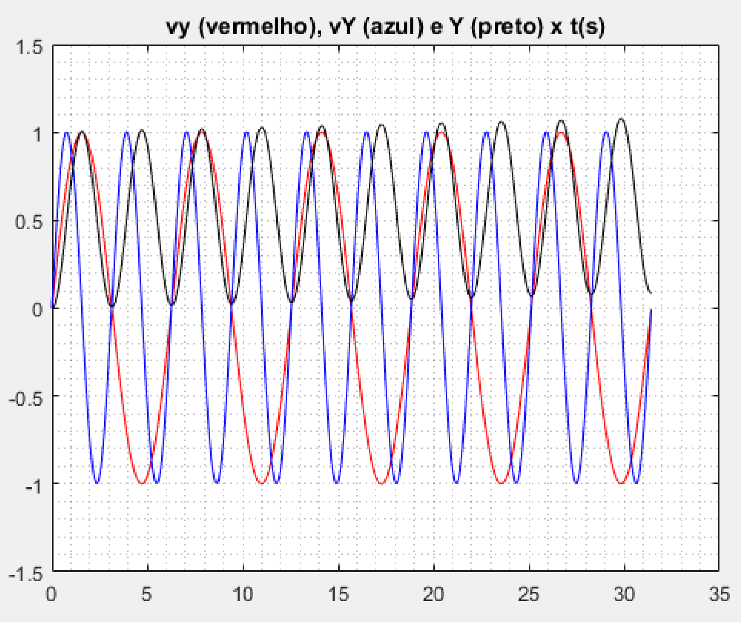


Figura 5.23 Curva de deslocamento, e velocidades no referencial global e local relativo ao eixo Y, para a simulação do movimento helicoidal em Z, gerado através do SIMULINK.

O gráfico acima descreve exatamente o comportamento esperado para a velocidade no eixo Y ao longo do tempo, ou seja, uma senoide em ambos os referenciais, tanto local quanto global. Também e posto a variação da posição Y. No próximo gráfico, figura 5.24, é descrito a curva da velocidade linear de X, em relação ao referencial local e global.

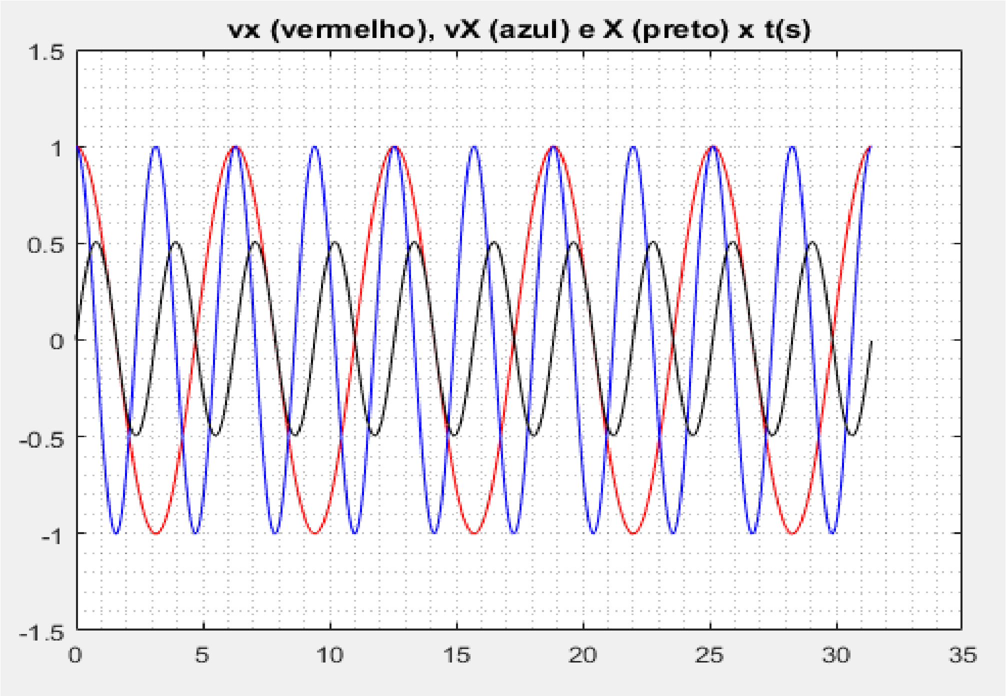


Figura 5.24 Curva de deslocamento, e velocidades no referencial global e local relativo ao eixo X, para a simulação do movimento helicoidal em Z, gerado através do SIMULINK.

E tal como esperado se obtém curvas senoidais para as velocidades. No gráfico também plotado o deslocamento em x, em função do tempo.

Como pode-se perceber nos gráficos acima há uma separação das curvas que descrevem a velocidade no referencial local e global, tal separação ocorre devido a uma diferença de frequência entre as senoide, que como visto, a curva que descreve a velocidade linear global, possuí uma frequência maior. Isso pode ser facilmente explicado a partir da matriz de rotação que é usada para a transformação de coordenadas tal como relacionado na equação demonstrada a seguir.

$$v_p^B = R_B v_p^G \rightarrow v_p^G = R_B^T v_p^B$$

Para a trajetória helicoidal, livre de rotações nos eixos X e Y, a matriz de R_B será a própria matriz de rotação no eixo Z.

$$R_B = \begin{bmatrix} cos\psi & sen\psi & 0 \\ -sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow R_B^T = \begin{bmatrix} cos\psi & -sen\psi & 0 \\ sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Fazendo a transformação de coordenadas como se segue e utilizando as equações 3.H1, tem-se:

$$v_p^G = \begin{bmatrix} V_X \\ V_Y \\ V_Z \end{bmatrix} = \begin{bmatrix} cos\psi & -sen\psi & 0 \\ sen\psi & cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$$

$$V_X = v_x cos\psi - v_y sen\psi = w_z\rho(\cos(w_z t)\, cos\psi - sen(w_z t) sen\psi) = w_z\rho(\cos^2\psi - sen^2\psi)$$

$$V_X = w_z\rho\cos(2\psi)$$

$$V_Y = v_x sen\psi + v_y cos\psi = w_z\rho(\cos(w_z t)\, sen\psi + sen(w_z t) cos\psi) = w_z\rho(2 sen\psi cos\psi)$$

$$V_X = w_z\rho\,\mathrm{sen}(2\psi)$$

$$V_Z = v_z$$

Como provado acima a velocidade no referencial global deverá ter uma frequência igual ao dobro daquela em comparação ao referencial local, tal efeito é decorrente do produto do vetor de velocidades no referencial local pela matriz de rotação no eixo Z. Também é demonstrado que para o eixo Z, a velocidade não varia, independente do referencial, conforme as hipóteses feitas.

5.7.2 Gráficos das Velocidades da partícula para a simulação em Y.

O gráfico que é representado pela figura 5.25, descreve a curva das velocidades em Z gerada através da simulação, no gráfico também é plotado o deslocamento nesse eixo, porém agora para segunda simulação, da helicoidal em Y.

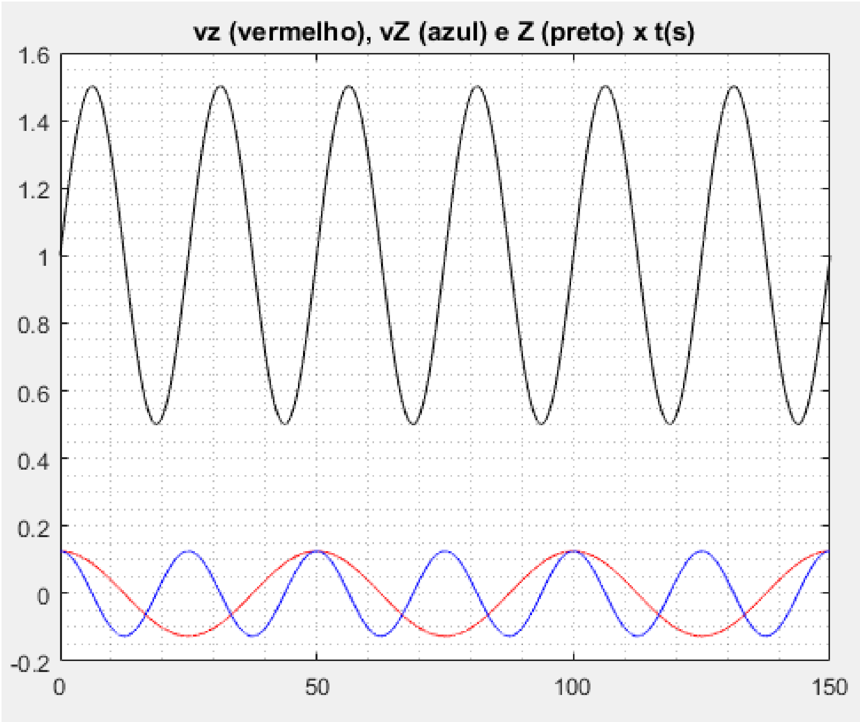


Figura 5.25 Curva de deslocamento para movimento helicoidal em Y, e velocidades no referencial global e local relativo ao eixo Z, gerada pela simulação no SIMULINK.

Como se pode observar pelo gráfico a velocidade da partícula será senoidal ao longo do movimento no eixo Z, dado que para essa simulação a base rotacional da Helicoide encontra-se no plano XZ. Novamente é plotada a variação da posição em Z, que analogamente a simulação anterior, essa também deve ser senoidal. E novamente percebe-se a diferença entre as frequências de oscilação da velocidade em relação aos referenciais Global e Local, devido a transformação dada pela matriz de rotação.

A figura 5.26 descreve a velocidade no eixo Y no referencial global, em azul, e local, em vermelho, que como já discutido, deverá ter um comportamento linear, porém variável, ao longo do movimento para essa simulação.

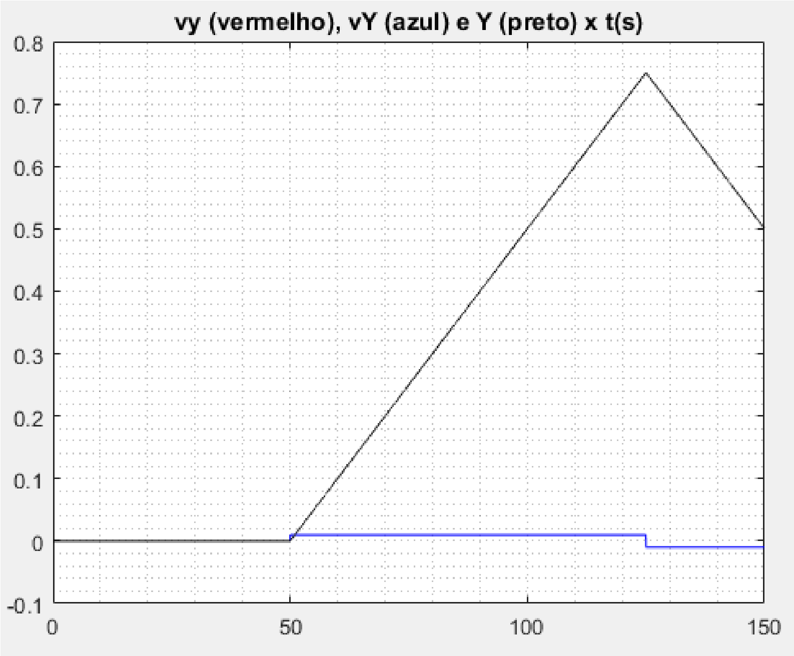


Figura 5.26 Curva de deslocamento, e velocidades no referencial global e local relativo ao eixo Y, para a simulação do movimento helicoidal em Y, gerado através do SIMULINK.

O gráfico acima descreve exatamente o comportamento esperado para a velocidade no eixo Y, que é análogo ao da variável Z na simulação anterior. Porém, conforme previamente determinado, a partícula só se deslocará em Y após realizada uma revolução, por isso há um trecho com velocidade nula, e posteriormente este irá descer, ou seja, se moverá contrário ao referencial, trecho de velocidade negativa. Também e posto a variação da posição Y. No próximo gráfico, figura 5.27, é descrito a curva da velocidade linear de X, em relação ao referencial local e global.

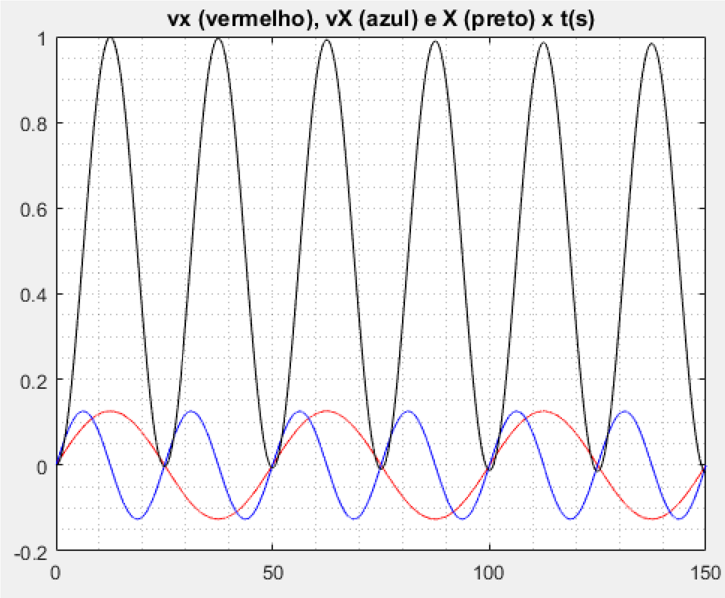


Figura 5.27 Curva de deslocamento, e velocidades no referencial global e local relativo ao eixo X, para a simulação do movimento helicoidal em Y, gerado através do SIMULINK.

E tal como esperado se obtém curvas senoidais para as velocidades. No gráfico também plotado o deslocamento em x, em função do tempo.

5.7.3 Gráficos das Velocidades da partícula para a simulação em X.

O gráfico que é representado pela figura 5.28, descreve a curva das velocidades em Z gerada através da simulação, no gráfico também é plotado o deslocamento nesse eixo, porém agora para terceira simulação, da helicoidal em X.

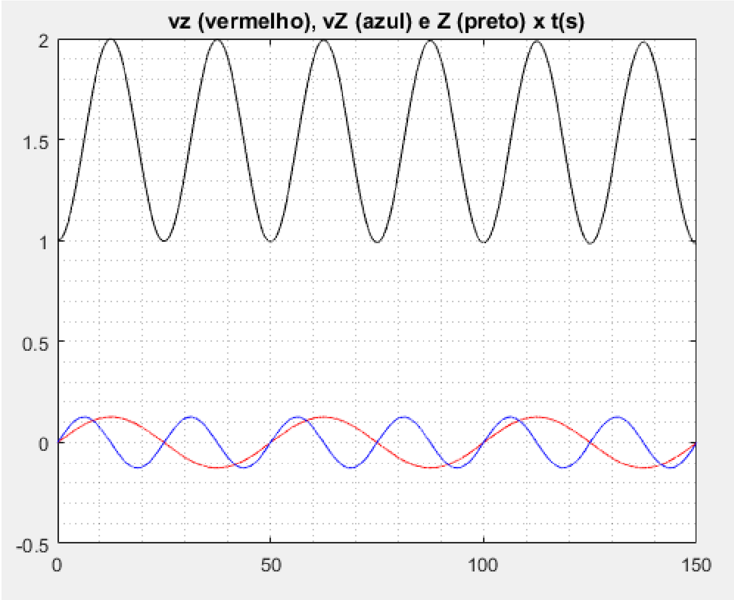


Figura 5.28 Curva de deslocamento para movimento helicoidal em X, e velocidades no referencial global e local relativo ao eixo Z, gerada pela simulação no SIMULINK.

Tal como no caso para a simulação anterior a velocidade da partícula será senoidal ao longo do movimento no eixo Z, dado que para essa simulação a base rotacional da Helicoide encontra-se no plano YZ. Novamente é plotada a variação da posição em Z.

A figura 5.29 descreve a velocidade no eixo Y no referencial global, em azul, e local, em vermelho, que como já discutido, deverá ter um comportamento linear, porém variável, ao longo do movimento para essa simulação.

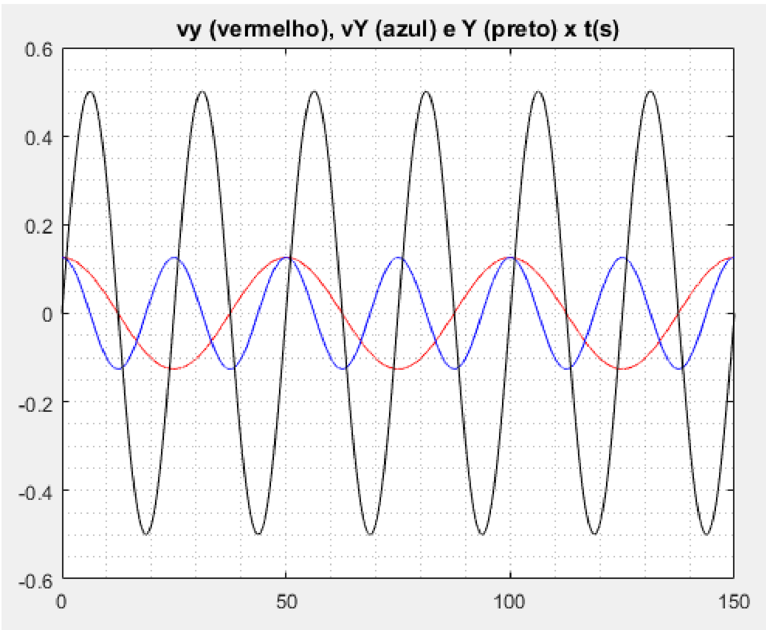


Figura 5.29 Curva de deslocamento, e velocidades no referencial global e local relativo ao eixo Y, para a simulação do movimento helicoidal em X, gerado através do SIMULINK.

O gráfico acima descreve exatamente o comportamento esperado para a velocidade no eixo Y, que agora será senoidal, dado que a helicoide ocorre ao longo do eixo X, insto é, ela rotaciona no plano YZ. No próximo gráfico, figura 5.30, é descrito a curva da velocidade linear de X, em relação ao referencial local e global.

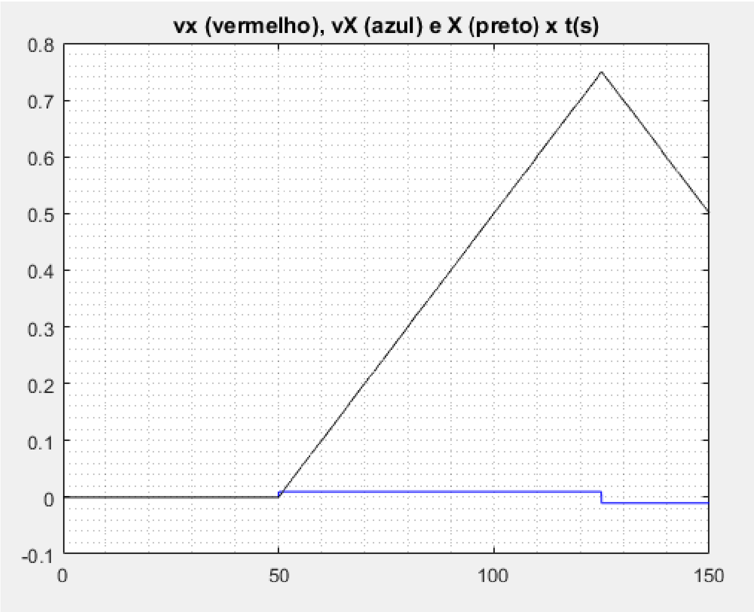


Figura 5.30 Curva de deslocamento, e velocidades no referencial global e local relativo ao eixo X, para a simulação do movimento helicoidal em X, gerado através do SIMULINK.

E tal como esperado obtém-se uma trajetória linear, e como imposto previamente na programação desta simulação, a partícula só se deslocará em X após duas revoluções, e posteriormente ele realizará um movimento retrogrado.

5.8 Ângulos de Euler.

Os ângulos de Euler esperados para uma partícula que descreve uma trajetória helicoidal deverão ser nulos em relação aos eixos X e Y, ou seja, ϕ e θ, respectivamente.

5.8.1 Gráficos dos Ângulos de Euler para Simulação em Z.

A curva gerada pela simulação é representada pela Figura 5.31, onde são plotados os valores dos ângulos em gruas em relação ao tempo.

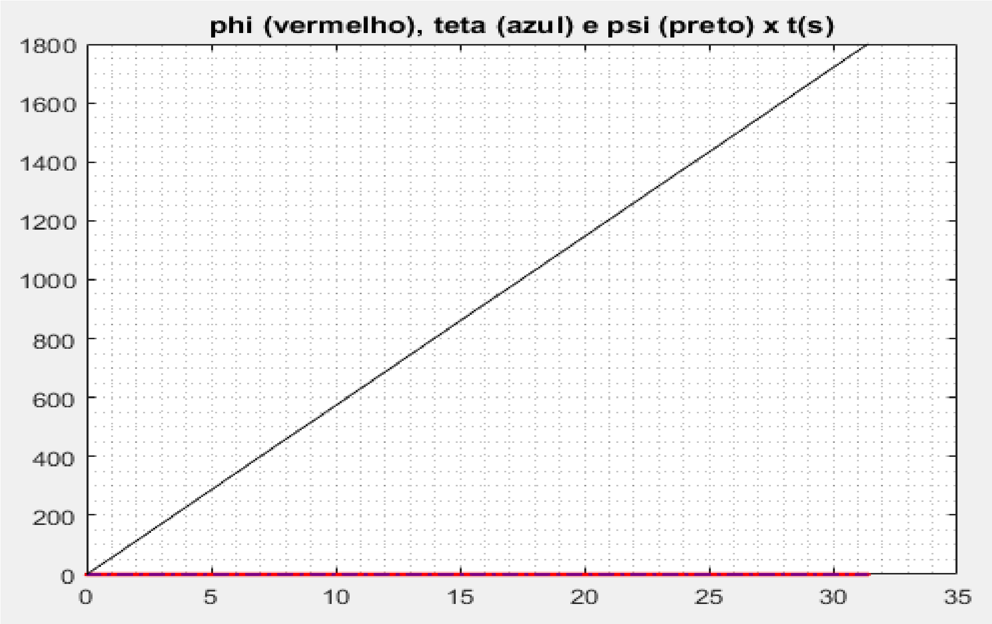


Figura 5.31 Ângulos de Euler ao longo do tempo para as rotações aplicadas, para uma helicoide em Z.

Tal como esperado os valores dos ângulos de Euler relativos aos eixos X e Y são nulos, porém, também conforme se esperava, o valor de ψ será uma reta que parte da origem, conforme a equação 3.8 do capitulo 3 que será alterada para representar apenas rotações em Z.

$$\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & -sen\theta \\ 0 & cos\phi & sen\phi cos\theta \\ 0 & -sen\phi & cos\phi cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$$\begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Que demonstra que a derivada dos ângulos de Euler será igual as velocidades angulares nos eixos, quando só há rotação em Z, portanto.

$$w_x = \frac{d\phi}{dt} = 0 \qquad w_y = \frac{d\theta}{dt} = 0 \qquad w_z = \frac{d\psi}{dt} = cte$$

$$\psi = w_z t$$

Esse resultado para ψ foi utilizado na seção anterior sem demonstração, agora, porém, é comprovado que o gráfico de ψ ao longo do tempo deverá ser uma reta que parte da origem, seguindo as formulações e hipóteses feitas para o movimento da partícula, é valido lembrar ainda que foi admitido como entrada um valor unitário para a velocidade angular no eixo Z, como já mencionado anteriormente.

5.8.2 Gráficos dos Ângulos de Euler para Simulação em Y.

A curva gerada pela simulação é representada pela Figura 5.32, onde são plotados os valores dos ângulos em gruas em relação ao tempo.

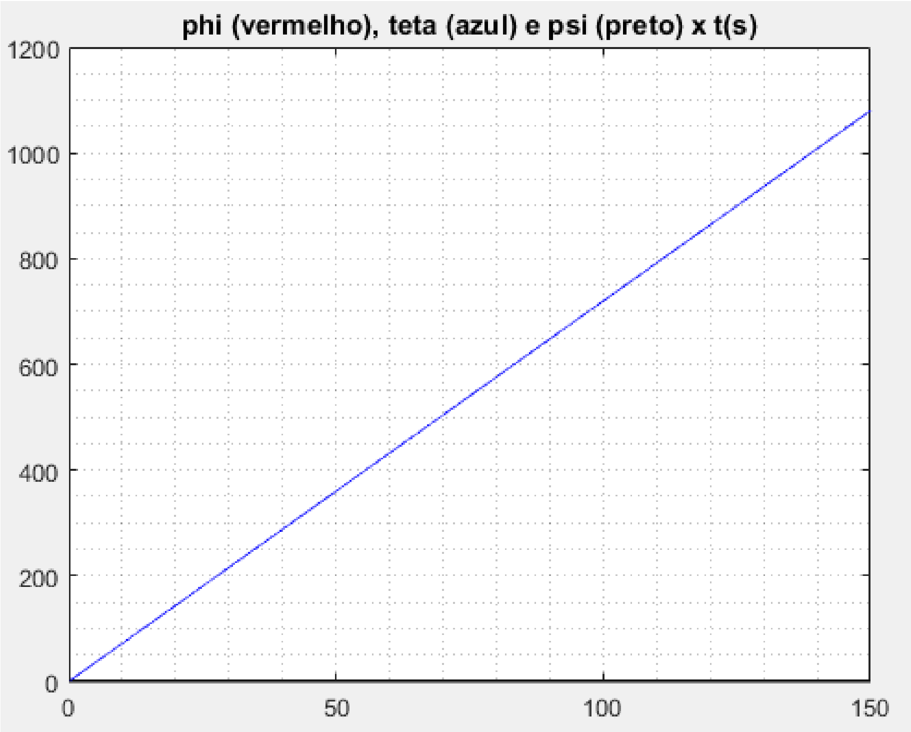


Figura 5.32 Ângulos de Euler ao longo do tempo para as rotações aplicadas.

Tal como esperado os valores dos ângulos de Euler relativos aos eixos X e Z são nulos, retas horizontais sobrepostas e o valor de θ será uma reta que parte da origem, conforme a equação 3.8 demonstrada analogamente na simulação anterior.

5.8.3 Gráficos dos Ângulos de Euler para Simulação em X.

A curva gerada pela simulação é representada pela Figura 5.33, onde são plotados os valores dos ângulos em gruas em relação ao tempo.

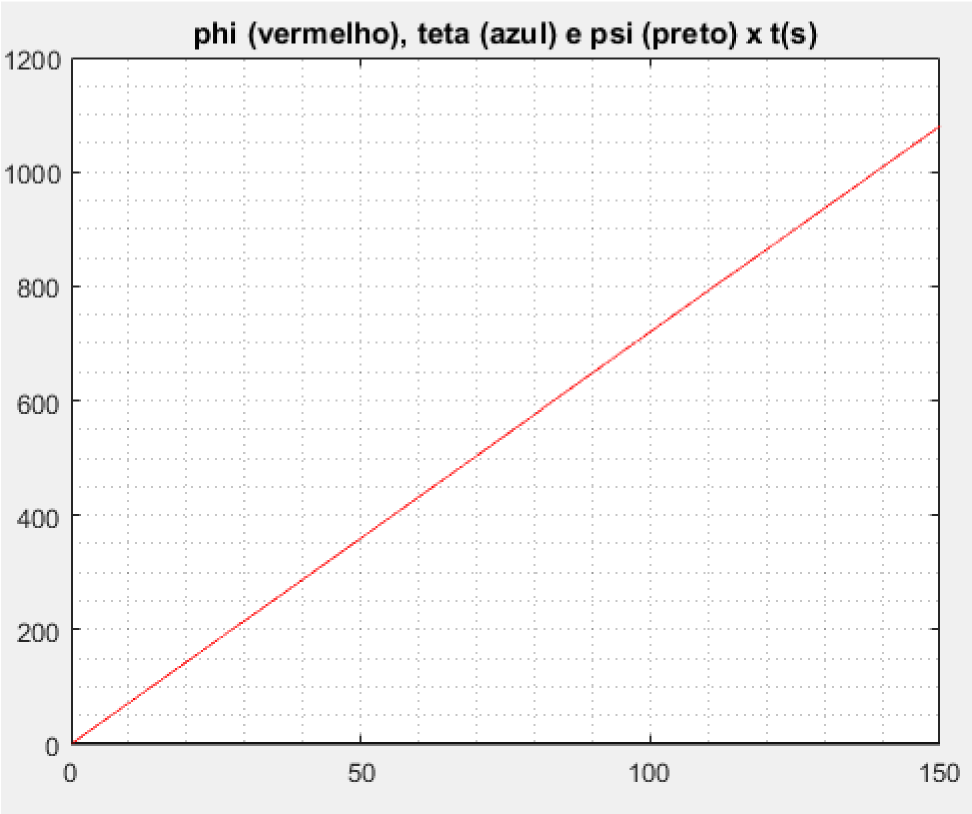


Figura 5.33 - Ângulos de Euler ao longo do tempo para as rotações aplicadas, para a helicoide em X.

Tal como esperado e analogamente as simulações anteriores, os valores dos ângulos de Euler relativos aos eixos Y e Z, para essa simulação, são nulos, dado que não há rotação nesses eixos, retas horizontais sobrepostas. E o valor de ϕ será uma reta que parte da origem, conforme a equação 3.8 demonstrada analogamente na simulação anterior.

5.9 Derivadas temporais dos ângulos de Euler.

5.9.1 Gráficos das derivadas temporais dos ângulos de Euler para helicoide realizada no eixo Z.

As derivadas são representadas pela curva do gráfico a seguir, figura 5.34, em que os valores estão em graus por segundo ao quadrado ($°/s^2$).

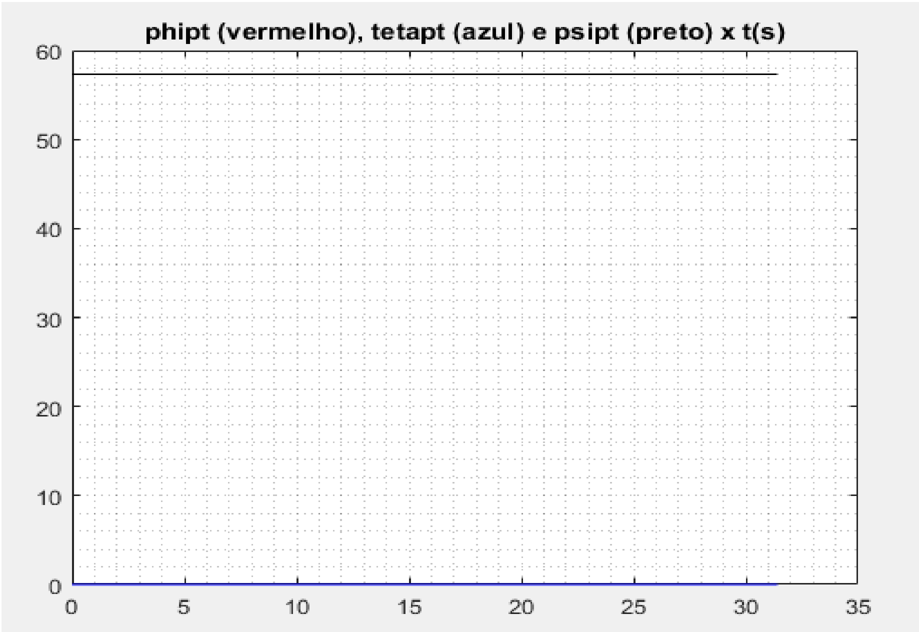


Figura 5.34 Taxas de variação dos ângulos de Euler para helicoide em Z.

Confirmando a predição, os valores das derivadas angulares em X e Y, representadas pelas curvas horizontais sobrepostas na origem, são nulas durante todo o movimento, já que não há rotação relativa a esses eixos, e também é visto a curva superior que representa a taxa de variação temporal de $\psi,$ que como visto na seção anterior, deverá ser igual a velocidade angular em Z, isto é, $1\ rad/s$ ou cerca de $57{,}3\ °/s,$ para essa simulação. A confirmação da validade das hipóteses acima também pode ser mostrada ao se plotar o gráfico com as velocidades angulares de entrada aquisitadas pelo giroscópio, como visto na Figura 5.35, onde igualmente os valores são plotados em graus por segundo no eixo vertical.

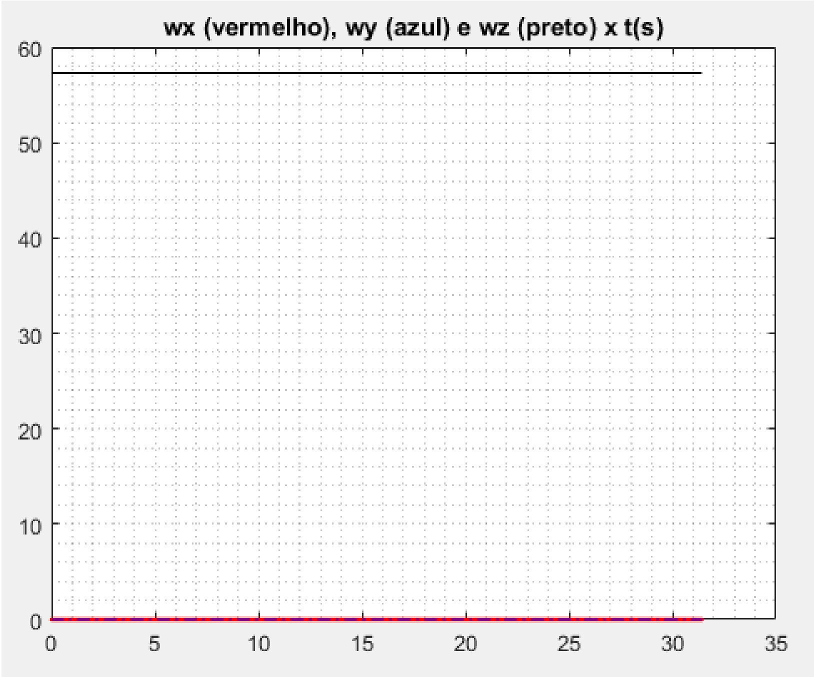


Figura 5.35 Velocidades angulares relativas aos eixos coordenados.

Novamente confirmando o esperado, dado que as velocidades angulares em X e Y também deverão ser nulas e a velocidade em Z foi determinada como $1\ rad/s$.

Como visto ao longo dessa seção, para o exemplo de uma partícula percorrendo uma trajetória helicoidal, onde as velocidades angulares em X e Y são nulas e em Z é um valor constante predeterminado, nesse exemplo igual a $1\ rad/s,$ as entradas detectadas pelo giroscópio, e as acelerações lineares nos eixos são todas nulas, entradas detectadas pelo acelerômetro.

5.9.2 Gráficos das derivadas temporais dos ângulos de Euler para helicoide realizada no eixo Y.

As derivadas são representadas pela curva do gráfico a seguir, figura 5.36, em que os valores estão em graus por segundo ao quadrado $(°/s^2)$.

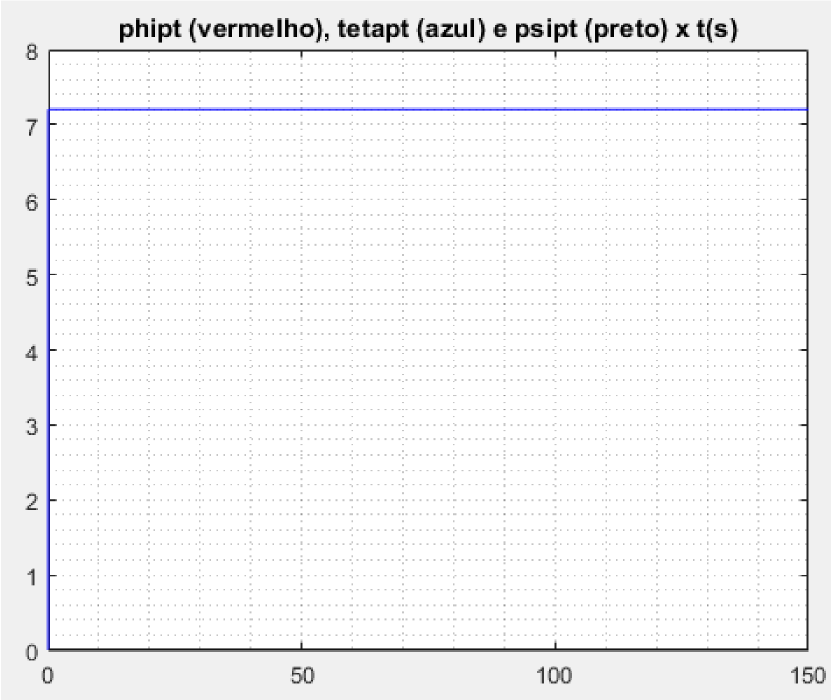


Figura 5.36 Taxas de variação dos ângulos de Euler
para helicoide em Y.

Analogamente a simulação anterior, por não existir rotação, desta vez, em X e Z, os ângulos de Euler associados a esses eixos apresentaram taxa de variação temporal nula, retas sobrepostas no eixo horizontal, e para aquele relativo ao eixo Y, eixo de rotação da helicoide, apresentará valor constante, dado que a velocidade angular nesse eixo é constante. Assim como no caso anterior a confirmação da validade das hipóteses acima também pode ser mostrada ao se plotar o gráfico com

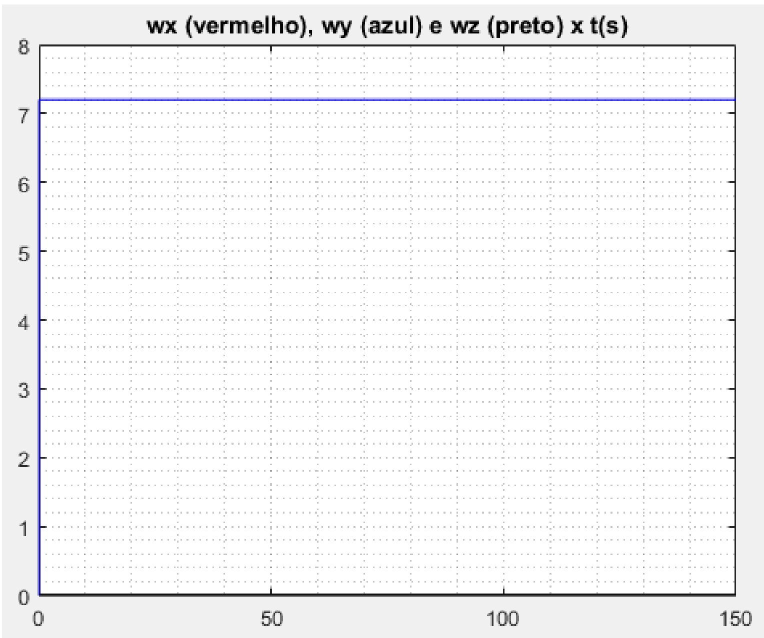


Figura 5.37 Velocidades angulares relativas aos eixos coordenados, para a
simulação em Y.

as velocidades angulares de entrada aquisitadas pelo giroscópio, como visto na Figura 5.37, onde igualmente os valores são plotados em graus por segundo no eixo vertical.

Novamente confirmando o esperado, dado que as velocidades angulares em X e Z, analogamente a simulação anterior, deverão ser nulas e a velocidade em Y uma constante de valor igual a $\dot{\theta}$.

5.9.3 Gráficos das derivadas temporais dos ângulos de Euler para helicoide realizada no eixo X.

As derivadas são representadas pela curva do gráfico a seguir, figura 5.38, em que os valores estão em graus por segundo ao quadrado $(°/s^2)$.

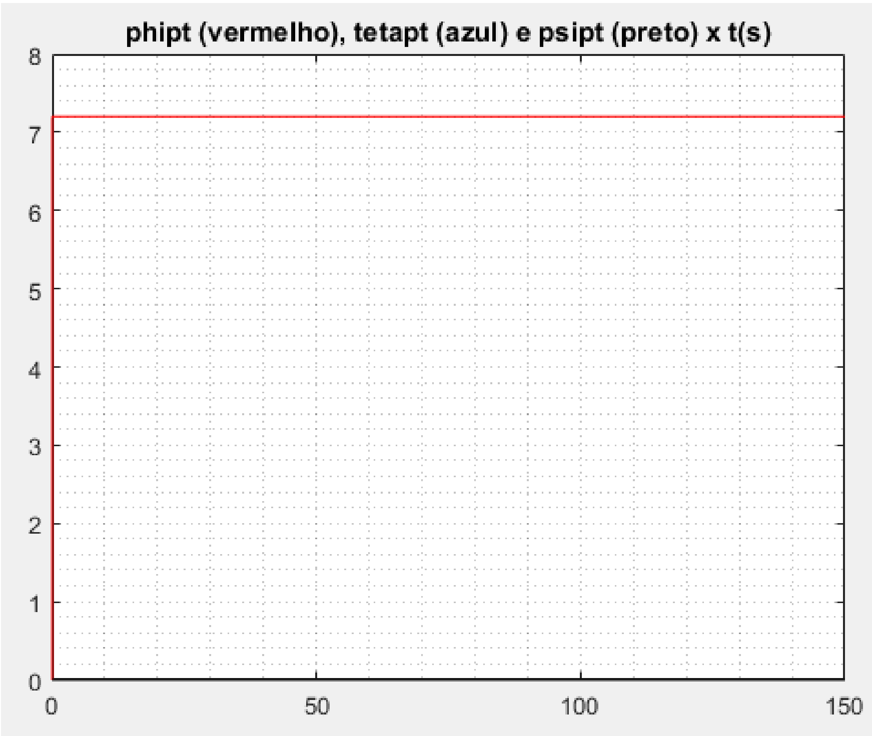


Figura 5.38 Taxas de variação dos ângulos de Euler para helicoide em X.

Analogamente as simulações anteriores, por não existir rotação, desta vez, em Y e Z, os ângulos de Euler associados a esses eixos apresentaram taxa de variação temporal nula, retas sobrepostas no eixo horizontal, e para aquele relativo ao eixo X, eixo de rotação da helicoide, apresentará valor constante, dado que a velocidade angular nesse eixo é constante. Assim como já mencionado anteriormente a validade das hipóteses acima também pode ser mostrada ao se plotar o gráfico com as velocidades angulares de entrada aquisitadas pelo giroscópio, como visto na Figura 5.39, onde igualmente os valores são plotados em graus por segundo no eixo vertical.

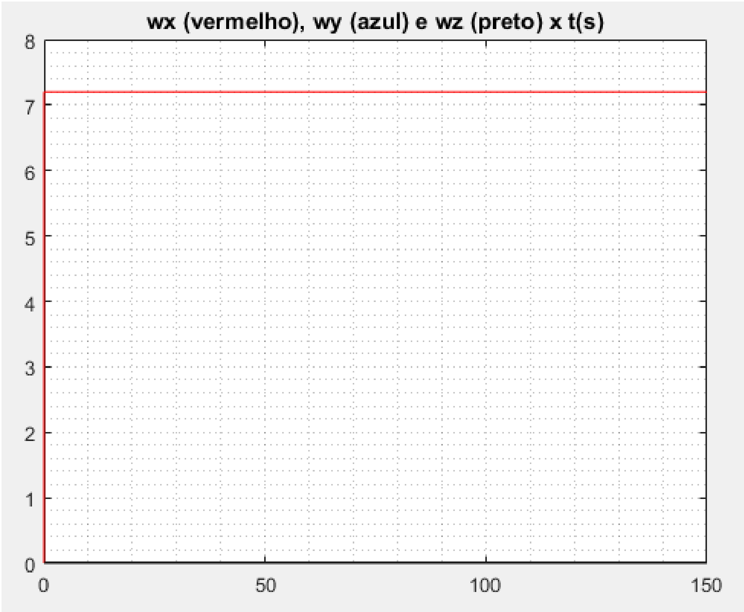


Figura 5.39 Velocidades angulares relativas aos eixos coordenados, para a simulação em X.

Tal como esperado, uma vez mais, as velocidades angulares em Y e Z são zero decorrente do fato de não haver rotação nesse eixo, e uma vez mais, o eixo de rotação da helicoide, nessa simulação, X, apresentará valor constante e igual a $\phi,$ para essa simulação.

Para todas as simulações apresentadas no presente capítulo, foi admitindo comportamento ideal do sistema, sem efeitos de forças externas, momentos (já que o corpo é tido como rígido e puntiforme), ou fatores dissipativos. E também estás se admitindo baixa inercia de detecção dos sensores, ou seja, o tempo de resposta ao estimulo tende a zero (instantâneo), não há ruídos na detecção e que demais efeitos que possam influenciar acréscimo de erros nas medições são desprezíveis, tais como, intemperes, erros de aproximação de valores, entre outros. Segundo essa formulação o programa respondeu adequadamente, gerando resultados satisfatórios e comprovaram a realidade física do movimento do corpo.

6 IMPLEMENTAÇÃO DAS FUNÇÕES DE QUATÉRNIONS UTILIZANDO PROGRAMAÇÃO VISUAL BASIC FOR APPLICATION (VBA) DO EXCEL.

6.1 Descrição.

No presente capítulo é explicada as rotinas implementadas nos módulos, são implementados funções e macros que determinam as principais operações envolvendo quatérnions, como demonstrado no capitulo 3, tal inspiração veio ao ler o trabalho de desenvolvimento em linguagem de programação C dos professores Sergio Coutinho de Biasi e Marcelo Gattass (BIASSI;GATTAS 2002).

Há um conjunto de módulos destinados a aplicações para usuários programadores, são eles o modulo “QUAT_MODULO_1” e o modulo de classe “clsQuaternion”, ambos os módulos devem ser utilizados juntos. O segundo modulo, “QUAT_MODULO_2” converter as funções presentes no modulo “QUAT_MODULO_1” para ambiente de planilha.

O programa foi elaborado na versão de 2003 do Excel e não foram utilizadas nenhuma referência do VBAProject ou interação com Interface de Programação de Aplicativos (Application Programming Interface – API) do Sistema operacional.

6.2 Modulo de Classe.

A classe “clsQuaternion” é criada uma classe que representa um quatérnion, onde são recebidos ou lidos os quatro coeficientes que o determina, há ainda nesse modulo uma propriedade que mostra em massagem de tela para o usuário a representação do quatérnion.

6.3 Modulo para programadores.

O Modulo “QUAT_MODULO_1” possui funções de operações entre quatérnions representados através da classe “clsQuaternion”. Esse modulo é destinado para programadores VBA, já que as funções e a classe podem ser utilizadas em outros módulos no ambiente VBA, em outras classes criadas pelo usuário, em formulários ou para criação de outras funções no ambiente de planilha.

Para utilizar os módulos, basta o usuário acessar o ambiente VBA através da aba Desenvolvedor no EXCEL, e clicar no ícone do Visual Basic, ou utilizar o atalho “ALT + F11”. Uma vez no ambiente VBA, basta adicionar um novo modulo e um modulo de classe e colar os códigos disponibilizados no apêndice A, porém deve-se atentar que o modulo de classe utilizado tenha o nome “clsQuaternion”, pois o tipo é declarado usando essa nomenclatura no modulo “QUAT_MODULO_1”. Abaixo são disponibilizadas as descrições das funções e rotinas desenvolvidas no modulo.

QuatCreate retorna um quatérnion representado através da classe CLS_QUATERNION, para isso é necessário fornecer os valores dos coeficientes do quatérnion.

QuatAdd retorna a soma entre dois quatérnions.

QuatSub retorna a subtração entre dois quatérnions.

QuatMultScale retorna a multiplicação de um quatérnion por um número real.

QuatProdScaler retorna o produto escalar entre dois quatérnions.

QuatMult retorna a multiplicação entre dois quatérnions.

QuatConj retorna o conjugado de um quatérnion.

QuatNorm retorna a norma de um quatérnion.

QuatNormalize retorna um quatérnion normalizado.

QuatInverse retorna o inverso de um quatérnion.

QuatDiv retorna a divisão entre dois quatérnions.

QuatCompare compara dois quatérnions, retornando TRUE em caso de igualdade e FALSE caso contrário.

QuatRotationXYZ gera uma rotação simbolizada na forma de um quatérnion, ao se passar o ângulo e o eixo X, Y ou Z.

QuatRotationGeral gera uma rotação simbolizada na forma de um quatérnion, ao se passar o ângulo e as coordenadas do vetor unitário desse eixo.

QuatGeraRotation Gera a rotação de um ponto em relação ao quatérnion que descreve uma rotação.

QuatIntermediaryRot gera um quatérnion que representa uma rotação intermediária através da interpolação entre duas rotações.

6.4. Modulo para ambiente planilha.

O Modulo "QUAT_MODULO_2" implementa no ambiente de planilha funções para cálculo das operações com quatérnions. Nesse modulo os quatérnions são representados utilizando o tipo String do Excel, ou seja, no formato de texto. Porém é necessário que a forma do quatérnion seja uma string como no exemplo que se segue.

$$"\boldsymbol{a + bi + cj + dk}"$$

Uma vez que as funções identificam o quatérnion pela posição em relação aos coeficientes imaginários e os sinais, sendo assim se o usuário desejar utilizar alguma função em um quatérnion, por exemplo, de valor real nulo, parte imaginaria *i* igual a $-2{,}14,$ parte imaginaria *j* igual a $67500,$ e parte imaginária *k* igual a $0{,}23,$ deve-se escreve-lo na forma "0-2,14i+67500j+0,23k".

Para acessar o conjunto de funções deve-se criar um modulo e colar o código disponibilizado no Apêndice A, de maneira semelhante como no módulo anterior. A chamada das funções é feita no ambiente de planilha da mesma maneira que funções padrões do Excel, isto é, usando o símbolo de igual (=) e posteriormente o nome da função. Abaixo são disponibilizadas as descrições das funções implementadas no módulo QUAT_MODULO_2, há porem funções encapsuladas para uso somente dentro do módulo e apenas pelo programa, sendo disponibilizadas apenas no Apêndice A.

QUAT_QUATERNION gera um quatérnion na forma de texto ao passar os valores dos coeficientes.

QUAT_IMPORTA_QUATERNION converte uma classe CLS_QUATERNION para string.

QUAT_SOMA retorna a soma de dois quatérnions.

QUAT_SUBTRACAO retorna a subtração entre dois quatérnions

QUAT_MULTIPLICACAO retorna a multiplicação entre dois quatérnions

QUAT_MULTIPLICACAO_MULTIPLAS retorna a multiplicação entre vários quatérnions.

QUAT_PRODUTO_ESCALAR retorna o produto escalar entre dois quatérnions.

QUAT_DIVISAO retorna a divisão entre dois quatérnions

QUAT_MULT_ESCALAR retorna a multiplicação de um escalar por um quatérnion

QUAT_CONJUGADO retorna o conjugado de um quatérnion

QUAT_NORMA retorna a norma de um quatérnion

QUAT_INVERSO retorna o inverso de um quatérnion

QUAT_COMPARA retorna TRUE caso dois quatérnions sejam iguais e FALSE caso contrário.

QUAT_ROTACAO_XYZ gera o quatérnion que descreve uma dada rotação passando-se o ângulo de rotação e o eixo X, Y ou Z.

QUAT_ROTACAO_GERAL gera o quatérnion que descreve uma dada rotação passando-se o ângulo e as coordenadas do vetor unitário de um eixo qualquer.

QUAT_ROTACAO_QUATERNION rotaciona dado ponto, passado na forma de quatérnion, em torno de um eixo qualquer, passando-se as coordenadas do vetor unitário nesse eixo, e o ângulo de rotação.

QUAT_ROTACAO_QUATERNION_XYZ rotaciona dado ponto, passado na forma de quatérnion, em torno de um eixo X, Y ou Z e o ângulo de rotação.

QUAT_ROTACAO_QUAT_QUAT rotaciona dado ponto, passado na forma de quatérnion, em torno de um eixo qualquer, passando o quatérnion que descreve a rotação e o dado eixo.

QUAT_ROTACAO_INTERMEDIARIA gera a rotação intermediaria entre duas rotações passadas na forma de quatérnion, dada a fração da Hiperesfera.

QUAT_ROTACOES_MULTIPLAS gera o quatérnion que descreve sucessivas rotações de um dado ponto, todas passadas na forma de quatérnion.

7 CONCLUSÃO

A tecnologia do MEMS tem avançado constantemente, e com isso mais e mais dispositivo são desenvolvidos ou aprimorados ao se inserir esses componentes, o exemplo dos VANTS evidencia bastante essa temática, as IMUs desempenham papel fundamental, nos dias atuais, para a determinação e controle de tais objetos. Sendo assim um programa computacional capaz de analisar e gerar valores quantitativos e qualitativos do movimento de um corpo a partir dos dados aquisitados pelos sensores do IMU desempenhará singular papel para o desenvolvimento de modelos mais eficientes, uma vez que a técnica possa ser usada de maneira simples.

Portanto o presente trabalho proporciona uma solução para o problema da analise dos dados do IMU, diante das premissas elencadas, pois servirá de base para o desenvolvimento de programas que determinem a realidade física do movimento levando em consideração a modelagem real do objeto.

O presente trabalho cumpre a tarefa de documentar as principais metodologias para modelagem de rotações, e ainda proporciona ferramentas para utilização de quatérnions para a modelagem de rotações, tendo em vista que com a essa modelagem utilizando tal metodologia evita-se o problema já mencionado no capitulo 3, o Gimbal Lock, ou seja, a perda de um grau de liberdade de rotação, e ainda, tal metodologia possibilita a reprodução mais simples e adequada de rotações intermediarias. O presente trabalho também fundamenta a simulação utilizando ângulos de Euler no MATLAB-SIMLINK, e conforme esperado, os parâmetros demonstraram comportamento físico conforme as previsões, ou seja, o simulador gerou as trajetórias corretas quando as entradas foram aquisitadas pelo IMU, os dados podem ser posteriormente instrumentados e enviados para Arduinos ou Softwares Embarcados em corpos que utilizem uma IMU.

Pretende-se dar continuidade, em trabalhos futuros, a modelagem utilizando ângulos de Euler implementada através do Excel VBA (tendo em vista a fácil disponibilidade desse software para todos os tipos de usuários) e também simulações para quatérnions e ângulos de Euler utilizando tal linguagem. Também pretende-se desenvolver simulações no MATLAB-SIMULINK para modelagem utilizando quatérnions, e ainda, deseja-se em futuros trabalhos, abordar a questão de ruídos nas leituras para determinar e, se possível, especificar filtros para e dispositivos para o tratamento do sinal.

REFERÊNCIAS BIBLIOGRAFICAS

ABINEE, Associação Brasileira da Industria Elétrica e Eletronica. Importações janeiro a outubro de 2018. Disponivel em: < http://www.abinee.org.br/abinee/decon/decon10.htm> Acesso em 20 de novembro de 2018

BIASI, Sergio C. de; GATTASS, Marcelo. Notas de Aulas - Utilização de Quatérnions para Representação de Rotações em 3D. Rio de Janeiro, Pontifícia Universidade Católica, 2002.

G. KORVINK, Jan; PAUL, Oliver. MEMS, A Pratical Guide to Desing, Analysis and Applocations. New York: William Andrew Publishing INC., 2006.

GEOGIEV, S. New Aspects on Elementary Functions in the Context of Quaternionic Analysis. University of Sofia, Bulgaria, 2012. Disponivel em: < https://scielo.conicyt.cl/pdf/cubo/v14n1/art08.pdf > Acesso em 22 de outubro de 2018.

GRIMM, Alice Marlene. Notas de Aula – Meteorologia Básica, Força de Coriolis Capitulo 7. Paraná, Universidade Federal do Paraná – Departamento de física,1999. Disponível em < http://fisica.ufpr.br/grimm/aposmeteo/cap7/cap7-3.html> Acessado em 03 de Outubro de 2018.

HAND, James A. Apollo: Guidance Navigation and Control. Massachusetts, Massachusetts Institute of Tecnology, 1971.

VOIG, John. Quaternions Algebras. Hanover, Dartmouth College, 2018

HOAG, David. Considerations of Apollo IMU Gimbal Lock. Massachusetts, Massachusetts Institute of Tecnology, 1963.

INVENSENSE. MPU-6000/6050 product specification. Revision 3.4. Sunnyvale (CA), 2013. Disponível em: <https://www.cdiweb.com/datasheets/invensense/MPU-6050_DataSheet_V3%204.pdf>. Acesso em: 05 de dezembro de 2018.

JAMBERSI, Andreyson Bicudo; SILVA, Samuel da. A Sutileza dos Quatérnions no Movimento de Rotação de Corpos Rígidos. Universidade Estadual Paulista – UNESP, São Paulo, 2016. Disponivel em: < http://www.scielo.br/pdf/rbef/v38n2/1806-1117-rbef-38-02-e2313.pdf > Acesso em 22 de outubro de 2018.

JAZAR, Reza N. Vehicle Dynamics: Theory and Applications. New York, Springer Science & LCC, 2008.

KEMPE, Volker. Inertial MEMS: Principles and Pratice. Cambridge, Cambridge Univerity Press, 2011.

KUIPERS, J. B. Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality. Princeton University Press, Princeton, 1999.

LAGES, Walter Fetter. Notas de Aulas – Descrições e Transformações Espaciais. Rio Grande do Sul, Universidade Federal do Rio Grande do Sul, 2018. Disponível em: <http://www.ece.ufrgs.br/~fetter/eng10026> Acesso em 15 de outubro de 2018.

LAPIN, Sergey. Notas de Aulas – Leornard Paul Euler: His Life and His Works. Washington State University, 2008.

LYSHEVSKI, Sergey Edward. MEMS and NEMS: Systems, Devices, and Structures. New York, CRC Press L.L.C., 2002.

MACLEHOSE, James. Sir Isaac Newton’s Principia. London, University of Glasgow, 1871.

MALUF, Nadim; WILLIAMS, Kirt. An Introduction to Microelectromechanical Systems Engineering. Boston: Artech House, 2004.

NETO SPERANZA, Mauro. Notas de Aulas – Cinemática Corpo Rígido. Rio de Janeiro, Pontifícia Universidade Católica, 2018.

OLIVEIRA, Waldri dos Santos; GONÇALVES, Eduardo Nunes. Implementação em C: Filtro de Kalman, Fusão de Sensores para determinação de Ângulos. Minas Gerais, For Science: Revista Cientifica do IFMG, 2017. Disponivel em: <http://www.forscience.ifmg.edu.br/forscience/index.php/forscience/article/view/287> Acesso em: 05 de dezembro de 2018.

P. BEEBY, Steve; et al. MEMS, Mechanical Sensors. London, Artech House INC, 2004.

P. WON, Seong-Hoon; GOLNARAGHI, Farid; W. MELEK, Wael. A Fastening Tool Tracking System Using an IMU and a Position Sensor With Kalman Filters and a Fuzzy Expert System. Vol 56, 2009. Disponível em: <https://ieeexplore.ieee.org/document/4689397> Acesso em 30 de setembro.

SAFFO, Paulo. Sensor: The Next Wave of Infotech Innovation, 1997. Disponível em: <http://www.saffo.com/essays/sensors-the-next-wave-of-infotech-innovation/> Acesso em: 25 de setembro. 2018.

SCIENTIFIC AMERICAN, BRASIL. Coleção Gênios da Ciência: Arquimedes, pioneiro da matemática. Nº 7. Edição Especial (2005).

SHOEMAKET, Ken. Animating Rotation With Quaternion Curves.Colorado State University, São Francisco,1985. Disponivel em: <https://www.engr.colostate.edu/ECE481A2/Readings/Rotation_Animation.pdf> Acesso em 22 de outubro de 2018.

SYMON, Keyth R. Mecânica. Massachusetts, Addison-Wesley Publishing Company INC, 1960.

SYNGE, J. L.; et al. The Mathematical Papers of Sir William Rowan Hamilton. Royal Irish Academy, 2004. Disponivel em: < https://www.maths.tcd.ie/pub/HistMath/People/Hamilton/Papers.html> Acesso em 20 de outubro de 2018.

SZE, S. M. Semiconductor Sensors. John Wiley & Sons, INC., 1994.

UBIRATAN, Edmundo. Especial Drones: Eles estão entre nós. Revista AERO Magazine 248, janeiro 2015.

WETZSTEIN, Gordon. Notas de Aula: Inertial Measurement Units I – Lecture 9. Stanford University, 2018. Disponível em: < https://stanford.edu/class/ee267/lectures/> Acesso em 30 de setembro.

WOODMAN, Oliver J. An Introduction to Inertial Navigation. United Kingdom, University of Cambridge, 2007. Disponível em: <https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf > Acesso em 03 de outubro de 2018.

ZANONI, Fábio DORO. Modelagem e implementação do sistema de navegação para um AUV. 2012. 245 f. Dissertação (Mestrado em Engenharia Elétrica) - Escola Politécnica, Universidade de São Paulo, São Paulo, 2012. Disponível em: <http://www.teses.usp.br/teses/disponiveis/3/3152/tde-23032012-114741/pt-br.php>. Acesso em: 05 de dezembro de 2018.

ANEXO A - PROGRAMA EM VBA DE EQUAÇÕES DE QUATÉRNIONS.

CLASSE clsQuaternion:

```
*Módulo de Classe: Criação da classe Quaternion
'* Arquivo gerado: clsQuaternion
'* Projeto: Trabalho de conclusão de curso - Implementação da classe que representa um quatérnion
'* Gestor: PUC-Rio
'* Autores: FJES Francisco Jayson Evangelista de Sousa
'*
'* Histórico de evolução:
'Versão Autor Data Observações
'1 FJES 09/10/2018 início desenvolvimento
'*
'*
'*****************************************Option Explicit
' Parte real do quaternion
Private pPitch As Double
' Parte imaginaria i do quaternion
Private PYaw As Double
' Parte imaginaria j do quaternion
Private pRoll As Double
'/****************************************
' Atribuição: PARTE REAL
' Descrição:
'Atribuição do valor para a parte real do quatérnion.
'*****************************************
Property Get Guinada() As Double
   PARTE_REAL = pReal
End Property
Property Let PARTE_REAL(VALOR As Double)
   pReal = VALOR
End Property
'/****************************************
'Atribuição: i
' Descrição:
'*Atribuição do valor para a parte imaginaria i do quatérnion.
'*****************************************
Property Get i() As Double
   i = PI
End Property
Property Let i(VALOR As Double)
   PI = VALOR
End Property
'/****************************************
'* Atribuição: j
'*Descrição:
'Atribuição do valor para a parte imaginaria j do quatérnion.
'*****************************************
Property Get j() As Double
   j = pj
End Property
Property Let j(VALOR As Double)
   pj = VALOR
End Property
'/****************************************
'AT Atribuição: k
' Descrição:
'*Atribuição do valor para a parte imaginaria k do quatérnion.
'*****************************************
Property Get k() As Double
   k = pk
End Property
Property Let k(VALOR As Double)
   pk = VALOR
End Property
```

```
'/*****************************************'*SB Rotina: Show
'*Descrição:
'*Mostra uma menssagem com a representação do quatérnion.
'******************************************
Public Sub Show()
   Dim Amsg As String
   Amsg=CStr(Round(PARTE_REAL, 5))

   If i >= 0 Then
      Amsg = Amsg & " + " & CStr(Round(i, 5)) & "i"
   Else
      Amsg = Amsg & CStr(Round(i, 5)) & "i"
   End If
   If j >= 0 Then
      Amsg = Amsg & " + " & CStr(Round(j, 5)) & "j"
   Else
      Amsg = Amsg & CStr(Round(j, 5)) & "j"
   End If
   If k >= 0 Then
      Amsg = Amsg & " + " & CStr(Round(k, 5)) & "k"
   Else
      Amsg = Amsg & CStr(Round(k, 5)) & "k"
   End If
   MsgBox Amsg

End Sub
```

Modulo QUAT_MODULO_1:

```
'*Módulo de implementação: Funções do modulo
'*Arquivo gerado: QUAT_MODULO 1
'*Letras identificadoras: Quat
'*Projeto: Trabalho de conclusão de curso em Engenharia Mecânica - Implementação de funções manipuladoras de hipercomplexos de 4° dimensão (Quatérnions) para uso no ambiente VBA
'*Gestor:  PUC-Rio
'*Autores: (FJES) Francisco Jayson Evangelista de Sousa
'*Histórico de evolução:
'* Versão  Autor    Data     Observações
'*    1      FJES    09/10/2018  início desenvolvimento
'*************************************************
' Função: Quat  Create
'*Descrição da função:
'* Cria um quatérnion do tipo q=a+bi+cj+dk, usando a classe clsQuaternion.
'* Parâmetros:
'* PARTE_REAL - parte real do quaternion
'* i - parte imaginária i do quaternion
'*j - parte imaginária j do quaternion
'*k - parte imaginária k do quaternion
'* Valor retornado
'* num    - objeto da classe clsQuaternion
'*************************************************
Public Function QuatCreate(ByVal PARTE_REAL As Double, ByVal i As Double, ByVal j As Double, ByVal k As Double) As clsQuaternion
    Dim num As clsQuaternion
    Set num = New clsQuaternion
    num.PARTE_REAL = PARTE_REAL
    num.i = i
    num.j = j
    num.k = k
    Set QuatCreate = num
    Set num = Nothing
End Function
'/************************************************
'*Função: Quat  Add
' Descrição da função:
'*Soma dois quaternions usando a classe clsQuaternion
'* Parâmetros:
'*x - Quaternion a ser somado
'*y - Quaternion a ser somado
'*Valor retornado
'*Resp     - objeto da classe clsQuaternion que representa o resultado
'*************************************************
Public Function QuatAdd(ByVal x As clsQuaternion, ByVal y As clsQuaternion) As clsQuaternion
    Dim Resp As clsQuaternion
    Set Resp = New clsQuaternion
    Resp.i = x.i + y.i
    Resp.j = x.j + y.j
    Resp.k = x.k + y.k
    Resp.PARTE_REAL = x.PARTE_REAL + y.PARTE_REAL
    Set QuatAdd = Resp
    Set Resp = Nothing
End Function
'/************************************************
'* Função: Quat  Sub
' Descrição da função:
'*Subtrai dois quaternions usando a classe clsQuaternion
'*Parâmetros:
'*x - Quaternion a ser subtraido
'*y - Quaternion de subtração
'*Valor retornado
```

```
'*Resp      - objeto da classe clsQuaternion que representa o resultado
'**************************************************
Public Function QuatSub(ByVal x As clsQuaternion, ByVal y As clsQuaternion) As clsQuaternion
    Dim Resp As clsQuaternion
    Set Resp = New clsQuaternion
    Resp.i = x.i - y.i
    Resp.j = x.j - y.j
    Resp.k = x.k - y.k
    Resp.PARTE_REAL = x.PARTE_REAL - y.PARTE_REAL
    Set QuatSub = Resp
    Set Resp = Nothing
End Function
'/**************************************************
**********************
'*Função: Quat  Mult Scalse
'* Descrição da função:
'*Multiplica um quaternion por um escalar
'*Parâmetros:
'*x - Quaternion a ser multiplicado
'*y - fator escalar multiplicativo
'* Valor retornado
'*Resp      - objeto da classe clsQuaternion que representa o resultado
'**************************************************
*********************/
Public Function QuatMultScale(ByVal x As clsQuaternion, ByVal y As Double) As clsQuaternion
    Dim Resp As clsQuaternion
    Set Resp = New clsQuaternion
    Resp.i = x.i * y
    Resp.j = x.j * y
    Resp.k = x.k * y
    Resp.PARTE_REAL = x.PARTE_REAL * y
    Set QuatMultScale = Resp
    Set Resp = Nothing
End Function
'/**************************************************
'*Função: Quat  Prod Scalse
'*Descrição da função:
'*       Realiza o produto escalar entre dois quatérnions
'* Parâmetros:
'*     x - Quaternion a ser multiplicado
'*     y - Quaternion a ser multiplicado
'*
'*  $FV Valor retornado
'*     Resp    - valor do produto escalar
'*
'**************************************************
*********************/
Public Function QuatProdScale(ByVal x As clsQuaternion, ByVal y As clsQuaternion) As Double
    Dim Resp As Double
    Resp = x.i * y.i + x.j * y.j + x.k * y.k + x.PARTE_REAL * y.PARTE_REAL
    QuatProdScale = Resp
End Function
'/**************************************************
**********************
'*Função: Quat  Mult
'* Descrição da função:
'*     Multiplicação simples entre dois quatérnions
'* Parâmetros:
'*x - Quaternion a ser multiplicado
'* y - Quaternion a ser multiplicado
'* Valor retornado
'*Resp      - objeto da classe clsQuaternion que representa o resultado
'**************************************************
*********************/
```

```
Public Function QuatMult(ByVal x As clsQuaternion, ByVal y As clsQuaternion) As clsQuaternion
    Dim Resp As clsQuaternion
    Set Resp = New clsQuaternion
    Resp.PARTE_REAL = x.PARTE_REAL * y.PARTE_REAL - x.i * y.i - x.j * y.j - x.k * y.k
    Resp.i = y.PARTE_REAL * x.i + x.PARTE_REAL * y.i + x.j * y.k - x.k * y.j
    Resp.j = x.PARTE_REAL * y.j + x.j * y.PARTE_REAL + x.k * y.i - x.i * y.k
    Resp.k = x.PARTE_REAL * y.k + x.i * y.j + x.k * y.PARTE_REAL - x.j * y.i
    Set QuatMult = Resp
    Set Resp = Nothing
End Function
'/************************************************
**********************
'* Função: Quat  Conj
'* Descrição da função:
'* Fornece o conjugado de um quatérnion
'* Parâmetros:
'* x - Quaternion de referencia
'* Valor retornado
'* Resp    - objeto da classe clsQuaternion que representa o conjugado
'************************************************
*********************/
Public Function QuatConj(ByVal x As clsQuaternion) As clsQuaternion
    Dim Resp As clsQuaternion
    Set Resp = New clsQuaternion
    Resp.i = -x.i
    Resp.j = -x.j
    Resp.k = -x.k
    Resp.PARTE_REAL = x.PARTE_REAL
    Set QuatConj = Resp
    Set Resp = Nothing
End Function
```

```
'/************************************************
**********************
'* Função: Quat  Norm
'* Descrição da função:
'* Gera a norma do quaternion
'* Parâmetros:
'*x - Quaternion de referencia
'* Valor retornado:
'* valor da norma
'************************************************
*********************/
Public Function QuatNorm(ByVal x As clsQuaternion) As Double
    QuatNorm = ((x.PARTE_REAL) ^ 2 + (x.i) ^ 2 + (x.j) ^ 2 + (x.k) ^ 2) ^ 0.5
End Function
'/************************************************
**********************
'* Função: Quat  Normalize
'* Descrição da função:
'*Normaliza um quaternion
'* Parâmetros:
'*q - Quaternion de referencia
'* Valor retornado
'' objeto da classe clsQuaternion que representa o quaternion normalizado
'*caso a norma do quaternion seja nula retorna o proprio quaternion
'************************************************
*********************/
Public Function QuatNormalize(ByVal q As clsQuaternion) As clsQuaternion
    Dim n As Double
    n = QuatNorm(q)
    If n = 0 Then
        MsgBox "Erro! valor não pode ser divido por nulo", vbCritical + vbOKOnly
        Set QuatNormalize = q
    Else
```

```
        Set QuatNormalize = QuatMultScale(q, 1 / n)
    End If
End Function
'/*************************************************
**********************
'* Função: Quat  Inverse
'* Descrição da função:
'*Inverte um quatérnion
'* Parâmetros:
'* q - Quaternion de referência
'* Valor retornado
'objeto da classe clsQuaternion que representa o quaternion invertido
'caso a norma do quaternion seja nula retorna o proprio quaternion
'*************************************************
*********************/
Public Function QuatInverse(ByVal q As clsQuaternion) As clsQuaternion
    Dim n As Double
    n = QuatNorm(q)
    If n = 0 Then
    MsgBox "Erro! valor não pode ser divido por nulo", vbCritical + vbOKOnly
        Set QuatInverse = q
    Else
        Set QuatInverse = QuatMultScale(q, 1 / (n * n))
    End If
End Function
'/*************************************************
**********************
'* Função: Quat  Div
'* Descrição da função:
'* Divide dois quatérnions
'* Parâmetros:
'* Q1 - Numerador da divisão
'* Q2 - Denominador da divisão
'* Valor retornado
'*Resp - objeto da classe clsQuaternion que representa resultado
'*************************************************
*********************/
Public Function QuatDiv(ByVal Q1 As clsQuaternion, ByVal Q2 As clsQuaternion) As clsQuaternion
    Dim Resp As clsQuaternion
    Dim n As Double
    Set Resp = New clsQuaternion
    n = QuatNorm(Q2)
    If n = 0 Then
        MsgBox "Erro! valor não pode ser divido por nulo", vbCritical + vbOKOnly
        Set QuatDiv = Q1
    Else
        Set QuatDiv = QuatMult(Q1, QuatInverse(Q2))
    End If
    Set Resp = Nothing
End Function
'/*************************************************
**********************
'* Função: Quat  Compare
'* Descrição da função:
'*Compara dois quatérnions
'* Parâmetros:
'* Q1 - Quatérnion de referência
'*     Q2 - Quatérnion de referência
'* Valor retornado
'*TRUE - Caso os quatérnions sejam iguais
'* FALSE - Caso os quatérnions sejam diferentes
'*************************************************
*********************/
Public Function QuatCompare(ByVal Q1 As clsQuaternion, ByVal Q2 As clsQuaternion) As Boolean
```

```
    If Q1.PARTE_REAL = Q2.PARTE_REAL And Q1.i = Q2.i And Q1.j = Q2.j And Q1.k = Q2.k Then
        QuatCompare = True
    Else
        QuatCompare = False
    End If
End Function
'/*********************************************************************
'* Função: Quat  Rotation XYZ
'* Descrição da função:
'*Gera um quatérnion que represente a rotação em um dos eixos principais X, Y ou Z
''*Parâmetros:
'*     EixoRotacao - string que representa o eixo, podendo ser: "x","y" ou "z" (O valor do parametro pode estar em caixa alta ou baixa)
'* AnguloRotacao - Angulo em que o objeto foi rotacionado no eixo em graus
'* Valor retornado
'* q - objeto da classe clsQuaternion que representa o quaternion que descreve a rotação
'*Caso haja atribuição de um eixo indevido, ou erro de digitação, a função será encerrada gerando apenas msg de erro.
''*********************************************************************/
Public Function QuatRotationXYZ(ByVal EixoRotacao As String, ByVal AnguloRotacao As Double) As clsQuaternion
    Dim q As clsQuaternion
    Dim NUM_PI, f As Double
    NUM_PI = Application.WorksheetFunction.PI
    f = NUM_PI / 180
    Set q = New clsQuaternion
    q.PARTE_REAL = Cos(AnguloRotacao * 0.5 * f)
    Select Case UCase(EixoRotacao)
        Case "X"
            q.i = Sin(AnguloRotacao * 0.5 * f)
            q.j = 0
            q.k = 0
        Case "Y"
            q.j = Sin(AnguloRotacao * 0.5 * f)
            q.i = 0
            q.k = 0
        Case "Z"
            q.k = Sin(AnguloRotacao * 0.5 * f)
            q.i = 0
            q.j = 0
        Case Else
            MsgBox "Eixo não existente"
            Exit Function
    End Select
    Set QuatRotationXYZ = q
End Function
'/*********************************************************************
'* Função: Quat  Rotation Geral
'* Descrição da função:
'* Gera o quatérnion que representa uma rotação em um dado eixo, a função ja gera a normalização do vetor eixo
'* Parâmetros:
'*CoordenadaVetor1 - Coordenada x do vetor que representa o eixo de rotação
'*CoordenadaVetor2 - Coordenada y do vetor que representa o eixo de rotação
'*CoordenadaVetor3 - Coordenada z do vetor que representa o eixo de rotação
'* AnguloRotacao - valor do angulo de rotação em graus
'*Valor retornado
```

```
'* q - objeto da classe clsQuaternion que representa a rotação no dado eixo
'************************************************
Public Function QuatRotationGeral(ByVal CoordenadaVetor1 As Double, ByVal CoordenadaVetor2 As Double,ByVal CoordenadaVetor3 As Double, ByVal AnguloRotacao As Double) As clsQuaternion
    Dim q As clsQuaternion
    Dim norma As Double
    Set q = New clsQuaternion
    Dim NUM_PI, f As Double
    NUM_PI = Application.WorksheetFunction.PI
    f = NUM_PI / 180
    q.PARTE_REAL = Cos(AnguloRotacao * 0.5 * f)
norma = (CoordenadaVetor1 ^ 2 + CoordenadaVetor2 ^ 2 + CoordenadaVetor3 ^ 2) ^ 0.5
    q.i = Sin(AnguloRotacao * 0.5 * f) * CoordenadaVetor1 / norma
    q.j = Sin(AnguloRotacao * 0.5 * f) * CoordenadaVetor2 / norma
    q.k = Sin(AnguloRotacao * 0.5 * f) * CoordenadaVetor3 / norma
    Set QuatRotationGeral = q
End Function
'/**********************************************
'* Função: Quat  Gera Rotation
'* Descrição da função:
'*    Gera a rotação de um ponto em relação ao quatérnion que representa sua rotação.
'* Parâmetros:
'*q_teta - Quatérnion que representa a rotação
'*q_ponto - Quatérnion que representa o ponto que se pretende rotacionar
'* Valor retornado
'*objeto da classe clsQuaternion que representa o resultado após a rotação.
'************************************************
Public Function QuatGeraRotation(ByVal q_teta As clsQuaternion, ByVal q_ponto As clsQuaternion) As clsQuaternion
    Set QuatGeraRotation = QuatMult(q_teta, QuatMult(q_ponto, QuatConj(q_teta)))
End Function
'/**********************************************
'* Função: Quat  Intermediary Rot
'* Descrição da função:
'* Gera um quatérnion que representa um rotação intermediária através da interpolação entre duas rotações
'* Parâmetros:
'* Q1 - Quatérnion que representa um rotação
'* Q2 - Quatérnion que representa um rotação
'* PathFraction - Fração do caminho entre as duas rotações
'* Valor retornado
'*objeto da classe clsQuaternion que representa a interpolação entre as rotações dadas.
'************************************************
Public Function QuatIntermediaryRot(ByVal Q1 As clsQuaternion, ByVal Q2 As clsQuaternion, ByVal PathFraction As Double) As clsQuaternion
Dim f1, f2, omega As Double
On Error GoTo ERRO
omega = QuatProdScale(Q1, Q2)
omega = Application.WorksheetFunction.Acos(omega)
omega = Application.WorksheetFunction.Radians(omega)
```

```
f1 = Sin((1 - PathFraction) * omega) / Sin(omega)
f2 = Sin(PathFraction * omega) / Sin(omega)
GoTo FIM
ERRO:
MsgBox "Verifique se ambos os quaternions são unitários", vbCritical + vbOKOnly, "Atenção"
FIM:
Set QuatIntermediaryRot = QuatAdd(QuatMultScale(Q1, f1), QuatMultScale(Q2, f2))
End Function
```

Modulo QUAT_MODULO_2:

```
'*  Módulo de implementação na planilha: Funções do modulo
'*  Arquivo gerado:        QUAT_MODULO 2
'*  Letras identificadoras:      QUAT
'*  Projeto: Trabalho de conclusão de curso - Implementação de funções manipuladoras Quatérnions para usuários no ambiente de uma planilha
'*  Gestor:  PUC-Rio
'*  Autores: FJES Francisco Jayson Evangelista de Sousa
'*  $HA Histórico de evolução:
'*    Versão  Autor   Data    Observações
'*      1     FJES   15/10/2018  início desenvolvimento
'*  OBS: Para facilitar a inserção dos dados, ao se inserir as funções usar o comando Ctrl + Shift + A
'**************************************************
'*  $FC Função: QUAT  QUATERNION
'*  $ED Descrição da função:
'*Cria um quatérnion do tipo q=a+bi+cj+dk, usando uma string.
'*  $EP Parâmetros:
'*    PARTE_REAL - parte real do quaternion
'*    IMAGINARIO_I - parte imaginária i do quaternion
'*    IMAGINARIO_J - parte imaginária j do quaternion
'*    IMAGINARIO_K - parte imaginária k do quaternion
'* Valor retornado
'* aux    - String que representa o quatérnion
'**************************************************
Public Function QUAT_QUATERNION(PARTE_REAL As Double, IMAGINARIO_I As Double, IMAGINARIO_J As Double, IMAGINARIO_K As Double) As String

Dim aux As String
aux = CStr(Round(PARTE_REAL, 4))
If IMAGINARIO_I >= 0 Then
   aux = aux & "+" & CStr(Round(IMAGINARIO_I, 4)) & "i"
Else
   aux = aux & CStr(Round(IMAGINARIO_I, 4)) & "i"
End If
If IMAGINARIO_J >= 0 Then
   aux = aux & "+" & CStr(Round(IMAGINARIO_J, 4)) & "j"
Else
   aux = aux & CStr(Round(IMAGINARIO_J, 4)) & "j"
End If
If IMAGINARIO_K >= 0 Then
   aux = aux & "+" & CStr(Round(IMAGINARIO_K, 4)) & "k"
Else
   aux = aux & CStr(Round(IMAGINARIO_K, 4)) & "k"
End If
QUAT_QUATERNION = aux
End Function
'/************************************************
' Função: Vetorizar Palavra
'* Descrição da função:
'*Gera um vetor de string a partir de uma string
'* Parâmetros:
'*    palavra - string a ser "vetorizada"
'* Valor retornado
'*     aux    - Vetor de string em que cada elemeto é um campo da string
'**************************************************
```

```
Private Function VetorizarPalavra(ByVal palavra As String) As String()
Dim tam, i As Long
tam = Len(palavra)
ReDim aux(1 To tam) As String
i = 1
While i <= tam
aux(i) = Mid(palavra, i, 1)
i = i + 1
Wend
VetorizarPalavra = aux()
End Function
'/*************************************************
' Função: QUAT IMPORTA QUATERNION
'* Descrição da função:
'*converte uma string que representa um quaternion para a classe clsQuaternion
'* Parâmetros:
'* Quaternion_Texto - string que representa um quatérnino na forma a+bi+cj+dk
'* Valor retornado
'*    q  - Objeto da classe clsQuaternion que representa o quaternion convertido
'**************************************************
Function QUAT_IMPORTA_QUATERNION(ByVal Quaternion_Texto As String) As clsQuaternion
Dim i, tam As Long
Dim t, ver, sinal As Integer
Dim Resp As String
Dim q As clsQuaternion
tam = Len(Quaternion_Texto)
ReDim vet(1 To tam) As String
Set q = New clsQuaternion
vet = VetorizarPalavra(Quaternion_Texto)
Resp = ""
i = 1
ver = 1
sinal = 0

If vet(1) = "-" Then
i = 2
t = 1
End If
While i <= tam
Resp = Resp & vet(i)
If vet(i) = "+" Or vet(i) = "-" Then
   Select Case ver
      Case 1
            If t = 1 Then
               q.PARTE_REAL = CDbl(Left(Resp, Len(Resp) - 1)) * (-1)
            Else
               q.PARTE_REAL = CDbl(Left(Resp, Len(Resp) - 1))
            End If
      Case 2
            q.i = CDbl(Left(Resp, Len(Resp) - 2))
      Case 3
            q.j = CDbl(Left(Resp, Len(Resp) - 2))
   End Select
   Resp = CStr(vet(i))
   ver = ver + 1
End If
i = i + 1
Wend
q.k = CDbl(Left(Resp, Len(Resp) - 1))
Set QUAT_IMPORTA_QUATERNION = q
End Function
'/*************************************************
'*Função: QUAT GERA QUATERNION
'* Descrição da função:
'*      converte uma quaternio representado pela classe clsQuaternion em uma string
'* Parâmetros:
'* Q1 - objeto da classe clsQuaternion que representa um quatérnino.
```

```
'* Valor retornado
'string que representa o quaternion convertido
'************************************************
Function QUAT_GERA_QUATERNION(ByVal Q1 As clsQuaternion) As String
   QUAT_GERA_QUATERNION = QUAT_QUATERNION(Q1.PARTE_REAL, Q1.i, Q1.j, Q1.k)
End Function
'/***********************************************
'*Função: QUAT SOMA
'* Descrição da função:
'* Soma dois quaternios
'* Parâmetros:
'*      QUATERNION_1 - quatérnino a ser somado.
'*      QUATERNION_2 - quatérnino a ser somado.
'* Valor retornado
'*     - string que representa o resultado
'************************************************
Function QUAT_SOMA(QUATERNION_1 As String, QUATERNION_2 As String) As String
Dim q As clsQuaternion
Set q = QuatAdd(QUAT_IMPORTA_QUATERNION(QUATERNION_1), QUAT_IMPORTA_QUATERNION(QUATERNION_2))
QUAT_SOMA = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
'/***********************************************
'* Função: QUAT SUBTRACAO
'* Descrição da função:
'* Subtrai dois quaternios
'* Parâmetros:
'*QUATERNION_1 - quatérnino a ser subtraido.
'*QUATERNION_2 - quatérnino de subtração.
'* Valor retornado
'* string que representa o resultado
'************************************************
Function QUAT_SUBTRACAO(QUATERNION_1 As String, QUATERNION_2 As String) As String
Dim q As clsQuaternion
Set q = QuatSub(QUAT_IMPORTA_QUATERNION(QUATERNION_1), QUAT_IMPORTA_QUATERNION(QUATERNION_2))
QUAT_SUBTRACAO = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
'/***********************************************
'* Função: QUAT MULTIPLICACAO
'* Descrição da função:
'* Multiplica dois quaternios
'* Parâmetros:
'*      QUATERNION_1 - quatérnino a ser multiplicado.
'*      QUATERNION_2 - quatérnino a ser multiplicado.
'* Valor retornado
'*     - string que representa o resultado
'************************************************
Function QUAT_MULTIPLICACAO(QUATERNION_1 As String, QUATERNION_2 As String) As String
Dim q As clsQuaternion
```

```
Set q = QuatMult(QUAT_IMPORTA_QUATERNION(QUATERNION_1), QUAT_IMPORTA_QUATERNION(QUATERNION_2))
QUAT_MULTIPLICACAO = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
'/***********************************************
'*Função: QUAT MULTIPLICACAO MULTIPLAS
'* Descrição da função:
'* Multiplica dois quaternios
'* Parâmetros:
'*QUATERNIONS - conjunto de quatérninos (objeto range) a serem multiplicados entre si.
'* Valor retornado
'* q - string que representa o resultado
'************************************************
Function QUAT_MULTIPLICACAO_MULTIPLAS(ParamArray QUATERNIONS() As Variant) As String
    Dim total As Double
    Dim rng As Variant
    Dim s, q As String
    q = "1+0i+0j+0k"
    s = QUATERNION_PONTO
    ' Ciclo nos diferentes parâmetros indicados
    For Each rng In QUATERNIONS
        ' Verifica se foi indicado um Range
        If TypeOf rng Is Range Then
            Dim r As Range
            Set r = rng
            ' Novo ciclo nas células do Range
            Dim cell As Range
            For Each cell In r
                q = QUAT_MULTIPLICACAO(q, cell.Value)
            Next
        End If
    Next
    ' Atribui o resultado à função
    QUAT_MULTIPLICACAO_MULTIPLAS = q
End Function
'/***********************************************
'* Função: QUAT PRODUTO ESCALAR
'* Descrição da função:
'*gera o produto escalar entre dois quaternios
'* Parâmetros:
'*QUATERNION_1 - quatérnino a ser multiplicado.
'*QUATERNION_2 - quatérnino a ser multiplicado.
'* Valor retornado
'*DOUBLE que representa o produto escalar entre dois quaternions
'************************************************
Function QUAT_PRODUTO_ESCALAR(QUATERNION_1 As String, QUATERNION_2 As String) As Double
QUAT_PRODUTO_ESCALAR = QuatProdScale(QUAT_IMPORTA_QUATERNION(QUATERNION_1), QUAT_IMPORTA_QUATERNION(QUATERNION_2))
End Function
'/***********************************************
'* Função: QUAT DIVISAO
'* Descrição da função:
'*gera a divisão entre dois quaternios
'* Parâmetros:
```

```
'*        QUATERNION_1 - quatérnino que representa o numerador da divisão.
'*        QUATERNION_2 - quatérnino que representa o denominador da divisão.
'* Valor retornado
'*string que representa o resultado da divisão
'*valor de erro caso norma do denominador seja nula
'*************************************************
Function QUAT_DIVISAO(QUATERNION_1 As String, QUATERNION_2 As String) As String
Dim q As clsQuaternion
Set q = QuatDiv(QUAT_IMPORTA_QUATERNION(QUATERNION_1), QUAT_IMPORTA_QUATERNION(QUATERNION_2))
QUAT_DIVISAO = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
'/***********************************************
'* Função: QUAT MULT ESCALAR
'* Descrição da função:
'*      gera a multiplicação de um quaternion por um escalar.
'* Parâmetros:
'*        QUATERNION_1 - quatérnino a ser multiplicado.
'*     VALOR - fator multiplicativo.
'* Valor retornado
'*      - string que representa o resultado
'*************************************************
Function QUAT_MULT_ESCALAR(QUATERNION_1 As String, VALOR As String) As String
Dim q As clsQuaternion
Set q = QuatMultScale(QUAT_IMPORTA_QUATERNION(QUATERNION_1), CDbl(VALOR))
QUAT_MULT_ESCALAR = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
'/***********************************************
'*Função: QUAT CONJUGADO
'* Descrição da função:
'*     gera o conjugado de um quaternio
'* Parâmetros:
'*          QUATERNION - quatérnino de referencia.
'* Valor retornado
' string que representa o conjugado do quaternion
'*************************************************
Function QUAT_CONJUGADO(QUATERNION As String) As String
Dim q As clsQuaternion
Set q = QuatConj(QUAT_IMPORTA_QUATERNION(QUATERNION))
QUAT_CONJUGADO = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
'/***********************************************
'* Função: QUAT NORMA
'* Descrição da função:
'*         gera o produto escalar entre dois quaternios
'* Parâmetros:
'*          QUATERNION - quatérnino de referencia.
'* Valor retornado
'*      - Valor da norma
'*************************************************
```

```
Function QUAT_NORMA(QUATERNION As String) As Double
QUAT_NORMA = QuatNorm(QUAT_IMPORTA_QUATERNION(QUATERNION))
End Function
'/***********************************************
'* Função: QUAT INVERSO
'*Descrição da função:
'*      gera o produto escalar entre dois quaternios
'* Parâmetros:
'*      QUATERNION - quaternino de referencia
'* Valor retornado
'*      - string que representa o resultado do inverso do quatérnion
'*      - string que representa o proprio quaternion caso sua norma seja nula
'*************************************************
Function QUAT_INVERSO(QUATERNION As String) As String
Dim q As clsQuaternion
Set q = QuatInverse(QUAT_IMPORTA_QUATERNION(QUATERNION))
QUAT_INVERSO = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
'/***********************************************
'* Função: QUAT COMPARA
'* Descrição da função:
'* gera o produto escalar entre dois quaternios
'* Parâmetros:
'*QUATERNION_1 - quatérnino de referencia
'*QUATERNION_2 - quatérnino de referencia
'* Valor retornado
'*      TRUE - caso sejam iguais
'*      TRUE - caso sejam distintos
'*************************************************
Function QUAT_COMPARA(QUATERNION_1 As String, QUATERNION_2 As String) As Boolean
QUAT_COMPARA = QuatCompare(QUAT_IMPORTA_QUATERNION(QUATERNION_1), QUAT_IMPORTA_QUATERNION(QUATERNION_2))
End Function
'/***********************************************
'* Função: QUAT ROTACAO XYZ
'* Descrição da função:
'*      gera o quaternion que descreve o resultado da rotação de um ponto
'* Parâmetros:
'*      COORD_PONTO_X - coordenada x de um ponto que se pretende rotacionar
'*      COORD_PONTO_Y - coordenada y de um ponto que se pretende rotacionar
'*      COORD_PONTO_Z - coordenada z de um ponto que se pretende rotacionar
'*      ANGULO - valor do angulo de rotação em graus.
'*      EIXO  - string que representa o eixo, "x","y" ou "z" (podendo ser em caixa alta).
'* Valor retornado
'*      - string que representa o quaternion resultado da rotação do ponto especificado no dado eixo
'*************************************************
Function QUAT_ROTACAO_XYZ(ByVal COORD_PONTO_X As Double, ByVal COORD_PONTO_Y As Double, _
```

ByVal COORD_PONTO_Z As Double, ByVal ANGULO As Double, EIXO As String) As String
Dim q As clsQuaternion
Set q = QuatGeraRotation(QuatRotationXYZ(EIXO, ANGULO), QuatCreate(0, COORD_PONTO_X, COORD_PONTO_Y, COORD_PONTO_Z))
QUAT_ROTACAO_XYZ = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
*'/***
' Função: QUAT ROTACAO GERAL*
' Descrição da função:*
' gera o quaternion que descreve o resultado da rotação de um ponto*
' Parâmetros:*
' COORD_PONTO_X - coordenada x de um ponto que se pretende rotacionar*
' COORD_PONTO_Y - coordenada y de um ponto que se pretende rotacionar*
' COORD_PONTO_Z - coordenada z de um ponto que se pretende rotacionar*
' ANGULO - valor do angulo de rotação em graus.*
' COORD_EIXO_X - coordenada x do vetor que descreve o eixo de rotação, não precisa ser normalizada*
' COORD_EIXO_Y - coordenada y do vetor que descreve o eixo de rotação, não precisa ser normalizada*
' COORD_EIXO_Z - coordenada z do vetor que descreve o eixo de rotação, não precisa ser normalizada*
' Valor retornado*
*'*string que representa o quaternion resultado da rotação do ponto especificado no dado eixo*
*'***
Function QUAT_ROTACAO_GERAL(ByVal COORD_PONTO_X As Double, ByVal COORD_PONTO_Y As Double, ByVal COORD_PONTO_Z As Double, _
ByVal ANGULO As Double, COORD_EIXO_X As Double, COORD_EIXO_Y As Double, COORD_EIXO_Z As Double) As String
Dim q As clsQuaternion
Set q = QuatGeraRotation(QuatRotationGeral(COORD_EIXO_X, COORD_EIXO_Y, COORD_EIXO_Z, ANGULO), QuatCreate(0, COORD_PONTO_X, COORD_PONTO_Y, COORD_PONTO_Z))
QUAT_ROTACAO_GERAL = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
*'/***
' Função: QUAT ROTACAO QUATERNION*
' Descrição da função:*
' gera a rotação de um ponto descrito através de um quaternion em relação a um dado eixo*
' Parâmetros:*
' QUATERNION - quaternion que descreve o ponto a ser rotacionado*
' ANGULO - valor do angulo de rotação em graus.*
' COORD_EIXO_X - coordenada x do vetor que descreve o eixo de rotação, não precisa ser normalizada*
' COORD_EIXO_Y - coordenada y do vetor que descreve o eixo de rotação, não precisa ser normalizada*
' COORD_EIXO_Z - coordenada z do vetor que descreve o eixo de rotação, não precisa ser normalizada*

```
'* Valor retornado
'*        - string que representa o quaternion resultado da rotação do ponto especificado no dado eixo
'*************************************************
Function QUAT_ROTACAO_QUATERNION(ByVal QUATERNION_PONTO As String, ByVal ANGULO As Double, ByVal COORD_EIXO_X As Double, _
                    ByVal COORD_EIXO_Y As Double, ByVal COORD_EIXO_Z As Double) As String
Dim q, w As clsQuaternion
Set w = QUAT_IMPORTA_QUATERNION(QUATERNION_PONTO)
Set q = QuatGeraRotation(QuatRotationGeral(w.i, w.j, w.k, ANGULO), QuatCreate(0, COORD_PONTO_X, COORD_PONTO_Y, COORD_PONTO_Z))
QUAT_ROTACAO_QUATERNION = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
'/***********************************************
'* Função: QUAT ROTACAO QUATERNION XYZ
'* Descrição da função:
'*        gera o produto escalar entre dois quaternios
'* Parâmetros:
'*          QUATERNION - quaternion que descreve o ponto a ser rotacionado
'*      ANGULO - valor do angulo de rotação em graus.
'*       EIXO - string que representa o eixo podendo ser "x","y" ou "z" (os mesmos podem estar em caixa alta)
'* Valor retornado
'*        - string que representa o quaternion resultado da rotação do ponto especificado no dado eixo
'*************************************************
Function QUAT_ROTACAO_QUATERNION_XYZ(ByVal QUATERNION As String, ByVal ANGULO As Double, EIXO As String) As String
Dim q As clsQuaternion
Set q = QuatGeraRotation(QuatRotationXYZ(EIXO, ANGULO), QUAT_IMPORTA_QUATERNION(QUATERNION))
QUAT_ROTACAO_QUATERNION_XYZ = QUAT_QUATERNION(q.PARTE_REAL, q.i, q.j, q.k)
End Function
'/***********************************************
'*Função: QUAT ROTACAO QUAT QUAT
'* Descrição da função:
'*        gera o produto escalar entre dois quaternios
'* Parâmetros:
'*       QUATERNION_PONTO - quatérnino que representa o ponto a ser rotacionado
'*        QUATERNION_2 - quatérnino que descreve a rotação
'* Valor retornado
'*        - string que representa o quaternion resultado da rotação
'*************************************************
Function QUAT_ROTACAO_QUAT_QUAT(ByVal QUATERNION_PONTO As String, ByVal QUATERNION_ROT As String) As String
    QUAT_ROTACAO_QUAT_QUAT = QUAT_MULTIPLICACAO(QUATERNION_
```

```
ROT,
QUAT_MULTIPLICACAO(QUATERNION_PONTO,
QUAT_CONJUGADO(QUATERNION_ROT)))
End Function
'/***********************************************
'*Função: QUAT ROTACAO INTERMEDIARIA
'* Descrição da função:
'*     gera o quaternios que descreve uma interpolação entre rotações, ou seja, uma rotação intermediaria através do método SLERP
'* Parâmetros:
'*       QUATERNION_1 - quatérnino que descreve uma dada rotação.
'*       QUATERNION_2 - quatérnino que descreve uma dada rotação.
'*       FRACAO_ANGULAR - valor que represenat a fração do caminho entre as dadas rotações, admensional.
'* Valor retornado
'*    - string que representa o quaternion que descreve a interpolação
'************************************************
Function QUAT_ROTACAO_INTERMEDIARIA(ByVal QUATERNION_1 As String, ByVal QUATERNION_2 As String, FRACAO_ANGULAR As Double) As String
QUAT_ROTACAO_INTERMEDIARIA = QUAT_GERA_QUATERNION(QuatIntermediaryRot(QUAT_IMPORTA_QUATERNION(QUATERNION_1), QUAT_IMPORTA_QUATERNION(QUATERNION_2), FRACAO_ANGULAR))
End Function
'/***********************************************
'* Função: QUAT ROTACOES MULTIPLAS
'* Descrição da função:
'* gera o  quaternio que descreve o resultado de um ponto após multiplas rotações
'* Parâmetros:
'*       QUATERNION_PONTO - quatérnino que representa o ponto a ser rotacionado.
'*       QUATERNIONS_ROT - conjuto de quatérninons (objeto range) que descrevem as rotações.
'* Valor retornado
'*       - string que representa o quaternion resultado da rotação
'************************************************
Function QUAT_ROTACOES_MULTIPLAS(ByVal QUATERNION_PONTO As String, ParamArray QUATERNIONS_ROT() As Variant) As String
   Dim total As Double
   Dim rng As Variant
   Dim s, q As String
   q = "1+0i+0j+0k"
   s = QUATERNION_PONTO
   ' Ciclo nos diferentes parâmetros indicados
   For Each rng In QUATERNIONS_ROT
       ' Verifica se foi indicado um Range
      If TypeOf rng Is Range Then
         Dim r As Range
         Set r = rng
         ' Novo ciclo nas células do Range
         Dim cell As Range
         For Each cell In r
            q = QUAT_MULTIPLICACAO(q, cell.Value)
         Next
      End If
```

```
Next
    ' Atribui o resultado à função
  QUAT_ROTACOES_MULTIPLAS =
QUAT_MULTIPLICACAO(q,
QUAT_MULTIPLICACAO(QUATERNION_
PONTO, QUAT_CONJUGADO(q)))
End Function
```

ANEXO B - FUNÇÕES MATLAB UTILIZADAS NA SIMULAÇÃO DO CAPITULO 4.

Velocidades_pt.

```
%
% Taxa Variação Velocidades Locais
%
function[v_pt]=Velocidades_pt(wx,wy,wz,ax,ay,
az,vx,vy,vz)
%  wx, wy, wz  --> velocidades ângulares no
referencial local
%  ax, ay, az  --> acelerações lineares no
referencial local
%  vx, vy, vz  --> velocidades lineares no
referencial local
% v_pt <-- taxas de variação das velocidades
no referencial local
v_pt=[ax; ay; az]-cross([wx; wy; wz],[vx; vy; vz]);
end
```

wx_wy_wz

```
%
% Velocidades Angulares interpoladas
%
function[w]=wx_wy_wz(t)
% t --> instante de tempo (t)
% w <-- vetor velocidade angular (rad/s)
global t_w w_x w_y w_z
if t <= t_w(1)
     w1 = 0;
     w2 = 0;
     w3 = 0;
else
    if t > t_w(length(t_w))
       w1 = 0;
       w2 = 0;
       w3 = 0;
    else
       w1 = interp1(t_w,w_x,t);
       w2 = interp1(t_w,w_y,t);
       w3 = interp1(t_w,w_z,t);
    end;
end;
w=[w1;w2;w3];
```

ax_ay_az

```
%
% Acelerações Lineares interpoladas
%
function[a]=ax_ay_az(t)
% t --> instante de tempo (t)
% a <-- vetor aceleracao (m/s2)
global t_a a_x a_y a_z
if t <= t_a(1)
     a1 = 0;
     a2 = 0;
     a3 = 0;
else
    if t > t_a(length(t_a))
       a1 = 0;
       a2 = 0;
       a3 = 0;
    else
       a1 = interp1(t_a,a_x,t);
       a2 = interp1(t_a,a_y,t);
       a3 = interp1(t_a,a_z,t);
    end;
end;
a=[a1;a2;a3];
```

Euler_pt

```
%
% Taxa Variação Ângulos de Euler
%
function[e_pt]=Euler_pt(wx,wy,wz,phi,teta)
```

```
%   wx, wy, wz   --> velocidades ângulares no referencial local
%  phi, teta   --> ângulos de Euler
%   e_pt <-- taxas de variação dos ângulos de Euler
e_pt=[ 1, sin(phi)*tan(teta), cos(phi)*tan(teta);
      0,          cos(phi),          -sin(phi);
      0,                              sin(phi)/cos(teta), cos(phi)/cos(teta)]*[wx; wy; wz];
end
```

Loc2Glob

```
%
% Transformação de Coordenadas Local --> Global
%
function[v_glob]=Loc2Glob(phi,teta,psi,vx,vy,vz)
%  vx, vy, vz  --> variáveis no referencial local
%  phi, teta, psi   --> ângulos de Euler
%  v_glob <-- variáveis no referencial global
v_glob=[                              cos(psi)*cos(teta), cos(psi)*sin(phi)*sin(teta)  -  cos(phi)*sin(psi), sin(phi)*sin(psi) + cos(phi)*cos(psi)*sin(teta);
        cos(teta)*sin(psi),     cos(phi)*cos(psi)     + sin(phi)*sin(psi)*sin(teta), cos(phi)*sin(psi)*sin(teta) - cos(psi)*sin(phi);
                -sin(teta), cos(teta)*sin(phi), cos(phi)*cos(teta)]*[vx; vy; vz];
end
```

plotar_cinematica

```
clc
close all
figure(1)
plot3(X,Y,Z,'b')
title('X x Y x Z')
xlabel('X (m)')
ylabel('Y (m)')
zlabel('Z (m)')
grid minor
hold
plot3(X(1),Y(1),Z(1),'bo')
plot3(X(length(X)),Y(length(Y)),Z(length(Z)),'b*')
% ANIMAÇÃO
i=0;
for ta=0:dt:tf
   i=i+1;
   ht = plot3(X(i),Y(i),Z(i),'xr');
   pause(dt/100)
   delete(ht)
end
figure(2)
plot(t,ax/9.81)
title('ax (g) x t(s)')
grid minor
figure(3)
plot(t,ay/9.81)
title('ay (g) x t(s)')
grid minor
figure(4)
plot(t,az/9.81)
title('az (g) x t(s)')
grid minor
figure(5)
plot(t,wx*180/pi)
title('wx (graus/s) x t(s)')
grid minor
figure(6)
plot(t,wy*180/pi)
title('wy (graus/s) x t(s)')
grid minor
figure(7)
plot(t,wz*180/pi)
```

```
title('wz (graus/s) x t(s)')
grid minor
figure(8)
plot(t,wx*180/pi,'r',t,wy*180/pi,'b',t,wz*180/pi,'k')
title('wx (vermelho), wy (azul) e wz (preto) x t(s)')
grid minor
figure(9)
plot(t,phi_pt*180/pi)
title('phipt (graus/s) x t(s)')
grid minor
figure(10)
plot(t,teta_pt*180/pi)
title('tetapt (graus/s) x t(s)')
grid minor
figure(11)
plot(t,psi_pt*180/pi)
title('psipt (graus/s) x t(s)')
grid minor
figure(12)
plot(t,phi_pt*180/pi,'r',t,teta_pt*180/pi,'b',t,psi_pt*180/pi,'k')
title('phipt (vermelho), tetapt (azul) e psipt (preto) x t(s)')
grid minor
figure(13)
plot(t,phi*180/pi)
title('phi (graus) x t(s)')
grid minor
figure(14)
plot(t,teta*180/pi)
title('teta (graus) x t(s)')
grid minor
figure(15)
plot(t,psi*180/pi)
title('psi (graus) x t(s)')
grid minor
figure(16)
plot(t,phi*180/pi,'r',t,teta*180/pi,'b',t,psi*180/pi,'k')
title('phi (vermelho), teta (azul) e psi (preto) x t(s)')
grid minor
figure(17)
plot(t,vx)
title('vx (m/s) x t(s)')
grid minor
figure(18)
plot(t,vy)
title('vy (m/s) x t(s)')
grid minor
figure(19)
plot(t,vz)
title('vz (m/s) x t(s)')
grid minor
figure(20)
plot(t,vX)
title('vX (m/s) x t(s)')
grid minor
figure(21)
plot(t,vY)
title('vY (m/s) x t(s)')
grid minor
figure(22)
plot(t,vZ)
title('vZ (m/s) x t(s)')
grid minor
figure(23)
plot(t,X)
title('X (m) x t(s)')
grid minor
figure(24)
plot(t,Y)
title('Y (m) x t(s)')
grid minor
figure(25)
plot(t,Z)
```

```
title('Z (m) x t(s)')
grid minor
figure(26)
plot(t,vx,'r',t,vX,'b',t,X,'k')
title('vx (vermelho), vX (azul) e X (preto) x t(s)')
grid minor
figure(27)
plot(t,vy,'r',t,vY,'b',t,Y,'k')
title('vy (vermelho), vY (azul) e Y (preto) x t(s)')
grid minor
figure(28)
plot(t,vz,'r',t,vZ,'b',t,Z,'k')
title('vz (vermelho), vZ (azul) e Z (preto) x t(s)')
grid minor
figure(29)
plot(t,X,'r',t,Y,'b',t,Z,'k')
title('X (vermelho), Y (azul) e Z (preto) x t(s)')
grid minor
figure(30)
plot(X,Y,'b')
title('Y x X')
hold
plot(X(1),Y(1),'bo')
plot(X(length(X)),Y(length(Y)),'b*')
grid minor
figure(31)
plot(X,Z,'b')
title('Z x X')
hold
plot(X(1),Z(1),'bo')
plot(X(length(X)),Z(length(Z)),'b*')
grid minor
figure(32)
plot(Y,Z,'b')
title('Z x Y')
hold
plot(Y(1),Z(1),'bo')
plot(Y(length(Y)),Z(length(Z)),'b*')
grid minor
```

simula_cinematica_eixo_x

```
clc
clear all
close all
global t_a a_x a_y a_z
global t_w w_x w_y w_z
% HELICOIDE HORIZONTAL
dt=0.01;
tetao=2*pi;
tetap=3*pi;
tetaf=6*pi;
rpm=1.2;
w=rpm*2*pi/60 % rad/s = w*180/pi graus/s = w*60/2/pi rpm
 to=tetao/w;
tp=tetap/w;
tf=tetaf/w
ro=1;
v=0.01;
v0=[0;w*ro;0];
X0=0;Y0=0;Z0=1;
r0=[X0;Y0;Z0];
i=0;
for t=0:dt:tf+dt
  i=i+1;
  t_(i)=t;
  teta(i)=w*t;
  wx(i) = w;
  wy(i) = 0;
  wz(i) = 0;
  ax(i) = 0;
  ay(i) = -2*wx(i)*wx(i)*ro*sin(wx(i)*t);
  az(i) = 2*wx(i)*wx(i)*ro*sin((wx(i)*t)+pi/2);
  if t >= to & t <= to+dt
    ax(i) = v/dt;
  end
  if t >= to+tp & t <= to+tp+dt
    ax(i) = -v/dt;
```

```
    end
end
figure(1)
plot(t_,wx,t_,wy,t_,wz)
title('Componentes do Vetor Velocidade Angular')
xlabel('t(s)')
ylabel('wx, wy, wz (rad/s)')
legend('wx','wy','wz')
grid minor
figure(2)
plot(t_,ax,t_,ay,t_,az)
title('Componentes do Vetor Aceleração Linear')
xlabel('t(s)')
ylabel('ax, ay, az (m/s2)')
legend('ax','ay','az')
grid minor
t_a=t_; a_x=ax; a_y=ay; a_z=az; t_w=t_; w_x=wx; w_y=wy; w_z=wz;
clear wx wy wz ax ay az t_
sim('cinematica_corpo_rigido_IMU')
plotar_cinematica
```

simula_cinematica_eixo_y

```
clc
clear all
close all
global t_a a_x a_y a_z
global t_w w_x w_y w_z
% HELICOIDE LATERAL
dt=0.01;
tetao=2*pi;
tetap=3*pi;
tetaf=6*pi;
rpm=1.2;
w=rpm*2*pi/60 % rad/s = w*180/pi graus/s = w*60/2/pi rpm
to=tetao/w;
tp=tetap/w;
tf=tetaf/w
ro=1;
v=0.01;
v0=[0;0;w*ro];
X0=0;Y0=0;Z0=1;
r0=[X0;Y0;Z0];
i=0;
for t=0:dt:tf+dt
    i=i+1;
    t_(i)=t;
    teta(i)=w*t;
    wx(i) = 0;
    wy(i) = w;
    wz(i) = 0;
    ax(i) = 2*wy(i)*wy(i)*ro*sin((wy(i)*t)+pi/2);
    ay(i) = 0;
    az(i) = -2*wy(i)*wy(i)*ro*sin(wy(i)*t);0;
    if t >= to & t <= to+dt
        ay(i) = v/dt;
    end
    if t >= to+tp & t <= to+tp+dt
        ay(i) = -v/dt;
    end
end
figure(1)
plot(t_,wx,t_,wy,t_,wz)
title('Componentes do Vetor Velocidade Angular')
xlabel('t(s)')
ylabel('wx, wy, wz (rad/s)')
legend('wx','wy','wz')
grid minor
figure(2)
plot(t_,ax,t_,ay,t_,az)
title('Componentes do Vetor Aceleração Linear')
xlabel('t(s)')
```

```
ylabel('ax, ay, az (m/s2)')
legend('ax','ay','az')
grid minor
t_a=t_;   a_x=ax;   a_y=ay;   a_z=az;   t_w=t_;
w_x=wx; w_y=wy; w_z=wz;
clear wx wy wz ax ay az t_
sim('cinematica_corpo_rigido_IMU')
plotar_cinematica
```

simula_cinematica_eixo_z

```
clc
clear all
close all
global t_a a_x a_y a_z
global t_w w_x w_y w_z
% HELICOIDE VERTICAL
dt=0.01;
tetao=2*pi;
tetap=3*pi;
tetaf=4*pi;
rpm=1.2;
w=rpm*2*pi/60  % rad/s = w*180/pi graus/s = w*60/2/pi rpm
to=tetao/w;
tp=tetap/w;
tf=tetaf/w
ro=1;
v=0.01;
v0=[w*ro;0;0];
X0=0;Y0=0;Z0=1;
r0=[X0;Y0;Z0];
i=0;
for t=0:dt:tf+dt
    i=i+1;
    t_(i)=t;
    teta(i)=w*t;
    wx(i) = 0;
    wy(i) = 0;
    wz(i) = w;
     ax(i) = -2*wz(i)*wz(i)*ro*sin(wz(i)*t);
    ay(i) = 2*wz(i)*wz(i)*ro*sin((wz(i)*t)+pi/2);
    az(i) = 0;
    if t >= to & t <= to+dt
        az(i) = v/dt;
    end
    if t >= to+tp & t <= to+tp+dt
        az(i) = -v/dt;
    end
end
figure(1)
plot(t_,wx,t_,wy,t_,wz)
title('Componentes do Vetor Velocidade Angular')
xlabel('t(s)')
ylabel('wx, wy, wz (rad/s)')
legend('wx','wy','wz')
grid minor
figure(2)
plot(t_,ax,t_,ay,t_,az)
title('Componentes do Vetor Aceleração Linear')
xlabel('t(s)')
ylabel('ax, ay, az (m/s2)')
legend('ax','ay','az')
grid minor
t_a=t_;   a_x=ax;   a_y=ay;   a_z=az;   t_w=t_;
w_x=wx; w_y=wy; w_z=wz;
clear wx wy wz ax ay az t_
sim('cinematica_corpo_rigido_IMU')
plotar_cinematica
```