INTERFACE PARA SIMULAÇÕES DA DINÂMICA DE CABOS COM ANIMAÇÃO TRIDIMENSIONAL

Prof. Me. Roberto Aguiar Junior (Escola Técnica João XXIII) E-mail: raguiar.jr@gmail.com
Prof. Dr. Carlos Rodrigues Rocha (IFRS) E-mail: carlos.rocha@riogrande.ifrs.edu.br
Prof. Dr. Sebastião C. P. Gomes (FURG) E-mail: sebastiaogomes@furg.br

Resumo: O artigo utiliza o formalismo discreto para a modelagem dinâmica de cabos umbilicais, o qual aproxima o cabo por diversos elos rígidos conectados entre si por juntas esféricas, cada uma permitindo movimentos rotacionais em três graus de liberdade: elevação, azimute e torção. O formalismo permitiu analisar e criar algoritmos que geram modelos dinâmicos de forma automática, em função do número de elos usados para aproximar de forma discreta a flexibilidade contínua. O ambiente simulado consiste em um cabo fixo em uma de suas extremidades, como uma embarcação na superfície, que a conecta a um ROV na outra extremidade. Este é considerado a carga terminal do cabo. Uma vez testada a consistência dos resultados, desenvolveu-se um software que gera uma visualização em três dimensões dos resultados. Para o desenvolvimento do software foi utilizada a linguagem de programação Python e a biblioteca gráfica OpenGL (Open Graphics Library). A interface do usuário foi baseada no framework Qt, através da biblioteca PyQt. Como resultado desse estudo tem-se um simulador de animações tridimensionais que foi validado de forma qualitativa e constatou-se que as simulações apresentaram-se de acordo com o esperado fisicamente.

Palavras-chave: Cabos, modelagem dinâmica, algoritmos, animações tridimensionais, Python, OpenGL.

INTERFACE FOR THREE-DIMENSIONAL ANIMATION OF CABLE DYNAMICS

Abstract: The article uses the discrete formalism for the dynamic modeling of umbilical cables, which approaches the cable by several rigid links connected by spherical joints, each allowing rotational movements in three degrees of freedom: elevation, azimuth and torsion. The formalism allowed to analyze and to create algorithms that generate dynamic models of automatic form, as a function of the number of links used to approach in a discreet way the continuous flexibility. The simulated environment consists of a cable attached at one end, such as a surface vessel, which connects it to a ROV at the other end. This is considered the terminal load of the cable. Once the consistency of the results was tested, software was developed that generates a three-dimensional visualization of the results. For the development of the software, the Python programming language and the OpenGL graphic library were used. The user interface was based on the Qt framework, through the PyQt library. As a result of this study we have a simulator of three-dimensional animations that was validated in a qualitative way and it was verified that the simulations showed physically expected results.

Keywords: Cables, dynamic modeling, algorithms, three-dimensional animations, Python, OpenGL.

1. INTRODUÇÃO

Pesquisas relacionadas ao ambiente subaquático, tais como a extração de petróleo e gás offshore têm atraído a atenção da comunidade cientifica, especialmente no caso de estruturas flexíveis como cabos e risers. Os cabos umbilicais estão presentes em plataformas petrolíferas offshore, plataformas e unidades flutuantes de produção e unidades móveis marítimas de perfuração. Neste meio, um exemplo de aplicação consiste em cabos conectados entre estruturas flutuantes e veículos subaquáticos não tripulados do tipo ROV (Remotely Operated Vehicle).

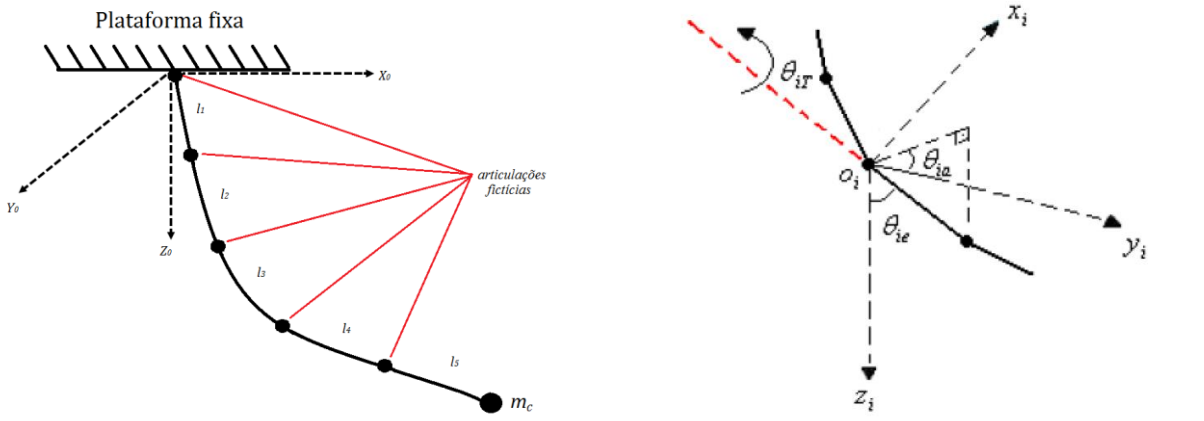
Atualmente, têm-se desenvolvido trabalhos sobre a modelagem dinâmica de estruturas flexíveis do tipo cabo utilizando-se formalismo de Euler-Lagrange. O formalismo discreto foi utilizado por Pereira [1] para criar um método que obtém o modelo dinâmico de estruturas flexíveis do tipo cabo. Este método supõe um cabo formado por pequenos elos rígidos

conectados por articulações fictícias esféricas elásticas sendo que cada articulação permite três movimentos livres: elevação, azimute e torção. Zanela [2] criou algoritmos que, independentes do número de elos que o cabo é dividido, permitem gerar automaticamente as equações dos modelos dinâmicos para as estruturas flexíveis.

No presente trabalho considerou-se a seguinte situação: um cabo com a extremidade superior articulada a uma plataforma fixa e a extremidade inferior conectada a um ROV. Com base nessa situação desenvolveu-se o software CabelSim, que permite gerar animações tridimensionais de simulações das dinâmicas de estruturas flexíveis, a partir dos algoritmos de Zanela [2]. Essa implementação deu-se conforme o número de elos em que o cabo é divido, utilizando-se de aplicativos com licença gratuita e sem vínculo com outros softwares, apresentando assim, uma nova forma de visualização e análise das simulações.

2. MODELAGEM DINÂMICA DE CABOS

Considera-se um cabo de flexibilidade contínua divido em n partes rígidas de comprimentos $l_1, l_2, l_3, \dots, l_n$, chamados de elos, conectados por juntas esféricas conforme mostra a Fig. 1 (a). Os elos possuem massas centradas nos seus centros de massas, ou seja, (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), ..., (x_n, y_n, z_n), com massas $m_1, m_2, m_3, \dots, m_n$, respectivamente, e (x_c, y_c, z_c) são as coordenadas do centro de massa da carga de massa m_c que, nesse caso, é a massa do ROV. Para cada junta i do cabo consideram-se os ângulos de azimute, elevação e torção, representados por $\theta_{ia}, \theta_{ie}, \theta_{iT}$, respectivamente, conforme a representação mostrada na Fig. 1 (b). No referencial x_i, y_i, z_i tem-se: o eixo $o_i z_i$ é paralelo ao eixo oz_0 do referencial inicial (sempre na direção vertical), o eixo $o_i y_i$ é paralelo à projeção da parte rígida anterior à i-*ésima* articulação (projeção no plano horizontal) e o eixo $o_i x_i$, é ortogonal ao eixo $o_i y_i$. Em cada articulação são consideradas três constantes elásticas, ou seja, na i-*ésima* articulação são consideradas as constantes elásticas k_{ia}, k_{ie}, k_{iT}, devidas aos ângulos de azimute, elevação e torção, respectivamente.



(a) Representação Discreta

(b) Ângulos de azimute, elevação e torção na i-*ésima* articulação

Figura 1: Estrutura flexível

A energia cinética é definida por:

$$E_C = E_{C_R} + E_{C_T} \tag{1}$$

onde E_{C_R} é a energia cinética devido ao movimento de rotação do cabo e E_{C_T} é a energia cinética devido ao movimento de translação do cabo.

A energia cinética devido ao movimento de rotação do cabo é definida por:

$$E_{C_R} = \frac{1}{2} I_{R_{1e}} \dot{\theta}_{1e}^2 + \frac{1}{2} I_{R_{2e}} \dot{\theta}_{2e}^2 + \frac{1}{2} I_{R_{3e}} \dot{\theta}_{3e}^2 + \cdots + \frac{1}{2} I_{R_{ne}} \dot{\theta}_{ne}^2 + \frac{1}{2} I_{R_{1a}} \dot{\theta}_{1a}^2 + \frac{1}{2} I_{R_{2a}} \dot{\theta}_{2a}^2 + \\ + \frac{1}{2} I_{R_{3a}} \dot{\theta}_{3a}^2 + \cdots + \frac{1}{2} I_{R_{na}} \dot{\theta}_{na}^2 + \frac{1}{2} I_{T_1} \dot{\theta}_{1T}^2 + \frac{1}{2} I_{T_2} \dot{\theta}_{2T}^2 + \frac{1}{2} I_{T_3} \dot{\theta}_{3T}^2 + \cdots + \frac{1}{2} \left(I_{T_n} + I_{T_c}\right) \dot{\theta}_{nT}^2 \quad (2)$$

onde $\theta_{1e}, \theta_{2e}, \theta_{3e}, \ldots, \theta_{ne}$ são os ângulos de elevação, $\theta_{1a}, \theta_{2a}, \theta_{3a}, \ldots, \theta_{na}$ são os ângulos de azimute, $\theta_{1T}, \theta_{2T}, \theta_{3T}, \ldots, \theta_{nT}$ são os ângulos de torção, $I_{R_{1e}}, I_{R_{2e}}, I_{R_{3e}}, \ldots, I_{R_{ne}}$ são os momentos de inércia relativos ao movimento de elevação, $I_{R_{1a}}, I_{R_{2a}}, I_{R_{3a}}, \ldots, I_{R_{na}}$ são os momentos de inércia relativos ao movimento de azimute e $I_{T_1}, I_{T_2}, I_{T_3}, \ldots, I_{T_n}, I_{T_c}$ são os momentos de inércia relativos aos movimentos de torção.

A energia cinética devido ao movimento de translação do cabo é definida por:

$$E_{C_T} = \frac{1}{2} m_1 (\dot{x}_1^2 + \dot{y}_1^2 + \dot{z}_1^2) + \frac{1}{2} m_2 (\dot{x}_2^2 + \dot{y}_2^2 + \dot{z}_2^2) + \frac{1}{2} m_3 (\dot{x}_3^2 + \dot{y}_3^2 + \dot{z}_3^2) + \cdots + \frac{1}{2} m_n (\dot{x}_n^2 + \dot{y}_n^2 + \dot{z}_n^2) \\ + \frac{1}{2} m_c (\dot{x}_c^2 + \dot{y}_c^2 + \dot{z}_c^2) \quad (3)$$

onde $\frac{1}{2} m_1 (\dot{x}_1^2 + \dot{y}_1^2 + \dot{z}_1^2)$, $\frac{1}{2} m_2 (\dot{x}_2^2 + \dot{y}_2^2 + \dot{z}_2^2)$, $\frac{1}{2} m_3 (\dot{x}_3^2 + \dot{y}_3^2 + \dot{z}_3^2)$, ..., $\frac{1}{2} m_n (\dot{x}_n^2 + \dot{y}_n^2 + \dot{z}_n^2)$ e $\frac{1}{2} m_c (\dot{x}_c^2 + \dot{y}_c^2 + \dot{z}_c^2)$ são as energias cinéticas relativas ao movimentos das massas m_1, m_2, m_3, ..., m_n dos elos e da massa m_c da carga terminal, respectivamente.

A energia potencial é definida por:

$$E_P = \frac{1}{2} k_{1e} \theta_{1e}^2 + \frac{1}{2} k_{2e} (\theta_{2e} - \theta_{1e})^2 + \frac{1}{2} k_{3e} (\theta_{3e} - \theta_{2e})^2 + \cdots + \frac{1}{2} k_{ne} \left(\theta_{ne} - \theta_{(n-1)e}\right)^2 + \frac{1}{2} k_{1a} \theta_{1a}^2 + \\ + \frac{1}{2} k_{2a} (\theta_{2a} - \theta_{1a})^2 + \frac{1}{2} k_{3a} (\theta_{3a} - \theta_{2a})^2 + \cdots + \frac{1}{2} k_{na} \left(\theta_{na} - \theta_{(n-1)a}\right)^2 + \frac{1}{2} k_{1T} \theta_{1T}^2 + \frac{1}{2} k_{2T} (\theta_{2T} - \theta_{1T})^2 \quad (4) \\ + \frac{1}{2} k_{3T} (\theta_{3T} - \theta_{2T})^2 + \cdots + \frac{1}{2} k_{nT} \left(\theta_{nT} - \theta_{(n-1)T}\right)^2 + m_1 g h_1 + m_2 g h_2 + m_3 g h_3 + \cdots + m_n g h_n$$

onde k_{1e}, k_{2e}, k_{3e}, ..., k_{ne} são as constantes elásticas nas articulações referentes aos ângulos de elevação, k_{1a}, k_{2a}, k_{3a}, ..., k_{na} são constantes elásticas nas articulações referentes aos ângulos de azimute, k_{1T}, k_{2T}, k_{3T}, ..., k_{nT} são constantes elásticas nas articulações referentes aos ângulos de torção e $m_1 g h_1 + m_2 g h_2 + m_3 g h_3 + \cdots + m_n g h_n$ é a energia potencial gravitacional, sendo $h_1, h_2, h_3, \ldots, h_n$ as alturas, que são definidas por:

$$h_1 = \frac{l_1}{2} (1 - \cos\theta_{1e}), h_2 = l_1 (1 - \cos\theta_{1e}) + \frac{l_2}{2} (1 - \cos\theta_{2e}), \ldots, \\ h_n = \sum_{i=1}^{n-1} l_i (1 - \cos\theta_{ie}) + \frac{l_n}{2} (1 - \cos\theta_{ne}) \quad (5)$$

Tendo definido as energias cinética e potencial, obtêm-se o Lagrangeano do sistema:

$$L = \frac{1}{2} I_{R_{1e}} \dot{\theta}_{1e}^2 + \frac{1}{2} I_{R_{2e}} \dot{\theta}_{2e}^2 + \frac{1}{2} I_{R_{3e}} \dot{\theta}_{3e}^2 + \cdots + \frac{1}{2} I_{R_{ne}} \dot{\theta}_{ne}^2 + \frac{1}{2} I_{R_{1a}} \dot{\theta}_{1a}^2 + \frac{1}{2} I_{R_{2a}} \dot{\theta}_{2a}^2 \\ + \frac{1}{2} I_{R_{3a}} \dot{\theta}_{3a}^2 + \cdots + \frac{1}{2} I_{R_{na}} \dot{\theta}_{na}^2 + \quad (6)$$

$$+\frac{1}{2}I_{T_1}\dot{\theta}_{1T}^2+\frac{1}{2}I_{T_2}\dot{\theta}_{2T}^2+\frac{1}{2}I_{T_3}\dot{\theta}_{3T}^2+\cdots+\frac{1}{2}\left(I_{T_n}+I_{T_c}\right)\dot{\theta}_{nT}^2+\frac{1}{2}m_1(\dot{x}_1^2+\dot{y}_1^2+\dot{z}_1^2)$$
$$+\frac{1}{2}m_2(\dot{x}_2^2+\dot{y}_2^2+\dot{z}_2^2)+$$
$$+\frac{1}{2}m_3(\dot{x}_3^2+\dot{y}_3^2+\dot{z}_3^2)+\cdots+\frac{1}{2}m_n(\dot{x}_n^2+\dot{y}_n^2+\dot{z}_n^2)+\frac{1}{2}m_c(\dot{x}_c^2+\dot{y}_c^2+\dot{z}_c^2)-\frac{1}{2}k_{1e}\theta_{1e}^2$$
$$+$$
$$-\frac{1}{2}k_{2e}(\theta_{2e}-\theta_{1e})^2-\frac{1}{2}k_{3e}(\theta_{3e}-\theta_{2e})^2-\cdots-\frac{1}{2}k_{ne}\left(\theta_{ne}-\theta_{(n-1)e}\right)^2-\frac{1}{2}k_{1a}\theta_{1a}^2$$
$$-\frac{1}{2}k_{2a}(\theta_{2a}-\theta_{1a})^2+$$
$$-\frac{1}{2}k_{3a}(\theta_{3a}-\theta_{2a})^2-\cdots-\frac{1}{2}k_{na}\left(\theta_{na}-\theta_{(n-1)a}\right)^2-\frac{1}{2}k_{1T}\theta_{1T}^2-\frac{1}{2}k_{2T}(\theta_{2T}-\theta_{1T})^2$$
$$-\frac{1}{2}k_{3T}(\theta_{3T}-\theta_{2T})^2+$$
$$-\cdots-\frac{1}{2}k_{nT}\left(\theta_{nT}-\theta_{(n-1)T}\right)^2-m_1gh_1-m_2gh_2-m_3gh_3-\cdots-m_ngh_n$$

De acordo com Pereira *et. al*. [3] o Lagrangeano do sistema pode ser posto em função de coordenadas angulares, conforme segue:

$$L=\sum_{b=1}^{n}\left\{\frac{1}{2}I_{R_{be}}\dot{\theta}_{be}^2+\frac{1}{2}I_{R_{ba}}\dot{\theta}_{ba}^2+\frac{1}{2}I_{T_b}\dot{\theta}_{bT}^2-\frac{1}{2}k_{be}\left(\theta_{be}-\theta_{(b-1)e}\right)^2+\right.$$
$$\left.-\frac{1}{2}k_{ba}\left(\theta_{ba}-\theta_{(b-1)a}\right)^2-\frac{1}{2}k_{bT}\left(\theta_{bT}-\theta_{(b-1)T}\right)^2-m_bg\left[\sum_{i=1}^{b-1}a_i+\frac{l_b}{2}(1-\cos\theta_{be})\right]\right\}+$$
$$+\sum_{b=1}^{n}\frac{1}{2}m_b\left\{\frac{l_b}{2}\left[\operatorname{sen}\theta_{be}\cos\left(\sum_{i=1}^{b}\theta_{ia}\right)\sum_{i=1}^{b}\dot{\theta}_{ia}+\cos\theta_{be}\operatorname{sen}\left(\sum_{i=1}^{b}\theta_{ia}\right)\dot{\theta}_{be}\right]+\right.$$
$$\left.+\sum_{j=1}^{b-1}l_j\left[\operatorname{sen}\theta_{je}\cos\left(\sum_{i=1}^{j}\theta_{ia}\right)\sum_{i=1}^{j}\dot{\theta}_{ia}+\cos\theta_{je}\operatorname{sen}\left(\sum_{i=1}^{j}\theta_{ia}\right)\dot{\theta}_{je}\right]\right\}^2+$$
$$+\sum_{b=1}^{n}\frac{1}{2}m_b\left\{\frac{l_b}{2}\left[-\operatorname{sen}\theta_{be}\operatorname{sen}\left(\sum_{i=1}^{b}\theta_{ia}\right)\sum_{i=1}^{b}\dot{\theta}_{ia}+\cos\theta_{be}\cos\left(\sum_{i=1}^{b}\theta_{ia}\right)\dot{\theta}_{be}\right]+\right.$$
$$\left.+\sum_{j=1}^{b-1}l_j\left[-\operatorname{sen}\theta_{je}\operatorname{sen}\left(\sum_{i=1}^{j}\theta_{ia}\right)\sum_{i=1}^{j}\dot{\theta}_{ia}+\cos\theta_{je}\cos\left(\sum_{i=1}^{j}\theta_{ia}\right)\dot{\theta}_{je}\right]\right\}^2+ \tag{7}$$
$$+\sum_{b=1}^{n}\frac{1}{2}m_b\left[-\frac{l_b}{2}\operatorname{sen}\theta_{be}\dot{\theta}_{be}-\sum_{j=1}^{b-1}l_j\operatorname{sen}\theta_{je}\dot{\theta}_{je}\right]^2+\frac{1}{2}I_{T_c}\dot{\theta}_{nT}^2+$$
$$+\frac{1}{2}m_c\left\{\sum_{b=1}^{n}l_b\left[\operatorname{sen}\theta_{be}\cos\left(\sum_{i=1}^{b}\theta_{ia}\right)\sum_{i=1}^{b}\dot{\theta}_{ia}+\cos\theta_{be}\operatorname{sen}\left(\sum_{i=1}^{b}\theta_{ia}\right)\dot{\theta}_{be}\right]\right\}^2+$$
$$+\frac{1}{2}m_c\left\{\sum_{b=1}^{n}l_b\left[-\operatorname{sen}\theta_{be}\operatorname{sen}\left(\sum_{i=1}^{b}\theta_{ia}\right)\sum_{i=1}^{b}\dot{\theta}_{ia}+\cos\theta_{be}\cos\left(\sum_{i=1}^{b}\theta_{ia}\right)\dot{\theta}_{be}\right]\right\}^2+$$

$$+\frac{1}{2}m_c\left[-\sum_{b=1}^{n} l_b \operatorname{sen}\theta_{be}\dot{\theta}_{be}\right]^2$$

Aplicando-se as equações de Euler-Lagrange, obtêm-se o modelo dinâmico do sistema na seguinte forma:

$$I(\vec{\theta})\ddot{\vec{\theta}} + C\dot{\vec{\theta}} + K\vec{\theta} + \vec{f}\left(\vec{\theta},\dot{\vec{\theta}}\right) + \vec{G}\left(\vec{\theta},\dot{\vec{\theta}}\right) = \vec{T}_m \quad (8)$$

onde $I(\vec{\theta})$ é a matriz de inércia, C é a matriz de coeficientes de atrito, K é a matriz de constantes elásticas, $\vec{f}\left(\vec{\theta},\dot{\vec{\theta}}\right)$ é o vetor de esforços Coriolis-centrifugos, $\vec{G}$é o vetor gravitacional e $\vec{T}_m$é o vetor dos torques externos resultantes dos ângulos de azimute, elevação e torção atuantes em cada articulação.

Em Gomes *et. al*. [4] foram propostos algoritmos que geram de forma automática o modelo dinâmico do cabo, considerando-se um número qualquer de elos para representar de forma discreta a flexibilidade contínua. Em Oliveira [5] foi desenvolvido um formalismo para se considerar cargas com dinâmica própria, como um ROV, gerando forças e torques que interagem com a dinâmica do cabo. O software desenvolvido no presente trabalho considera todas essas recentes proposições e permite a visualização em três dimensões de animações, relativas à complexa dinâmica do cabo e sua carga terminal com dinâmica própria.

3. METODOLOGIA E TECNOLOGIA DE DESENVOLVIMENTO

3.1. Metodologia de desenvolvimento

Definir uma metodologia para o desenvolvimento do *software* é uma fase inicial importante para promover qualidade do produto de *software* final. Portanto, a seguir apresentam-se as quatro etapas realizadas para a produção do aplicativo CableSim.

Como primeira etapa, optou-se pelo desenvolvimento incremental, que aborda intercaladamente, as atividades de especificação, desenvolvimento e validação. Nesse modelo são adicionadas funcionalidades a cada versão do sistema.

A segunda etapa adotada é a especificação dos requisitos do *software*, ou seja, determinar quais são as tarefas que o programa necessita realizar, os serviços que ele deve oferecer e as restrições referentes ao seu funcionamento (Sommerville [6]). Os requisitos elencados, considerando-se que foram descritos em nível de usuário, são listados conforme segue: possibilitar a importação dos dados gerados pelos algoritmos genéricos implementados no MATLAB; ser capaz de efetuar animações de acordo com os dados informados pelo usuário juntamente com as informações importadas (requisito é de suma importância para o projeto, visto que se refere ao objetivo geral do mesmo); possuir uma interface em que o usuário informe os dados adicionais da animação, tais como: tamanho do cabo, quantidade de elos e tempo de animação; permitir a aproximação entre o tempo da animação e o tempo real decorrido; possibilitar a adição de coordenadas cartesianas em cada elo para uma melhor visualização do posicionamento do elo e o cabo como um todo.

Na terceira etapa utilizou-se o diagrama de casos de uso para descrever as funcionalidades do sistema e captar o comportamento pretendido com o mesmo. O diagrama faz parte da Linguagem de Modelagem Unificada (UML, do inglês *Unified Modelling Language*) que, segundo Balzert [7], é uma linguagem de notação gráfica usada para criar

modelos de projetos de *software*. Um caso de uso abrange a interação entre atores e sistemas e um ator representa um conjunto de papeis que o usuário do sistema de *software* desempenha na vida real (Booch *et. al.* [8]). Os casos de usos foram elaborados com o intuito de demonstrar que têm dois atores envolvidos no projeto: o ator Usuário que é o responsável pela utilização do aplicativo e o MATLAB que é o ator externo (fornece os dados da animação).

A última etapa consiste no desenvolvimento do sistema utilizando-se o paradigma da orientação a objetos, visto que esse paradigma oferece diversas vantagens, como por exemplo, simplificar o desenvolvimento, facilitar a reutilização do código e agilizar a manutenção do mesmo. O aplicativo resultante da utilização da orientação a objetos é formado por unidades menores e independentes que reduz o nível de complexidade do sistema tornando fácil e rápido o seu desenvolvimento e a sua manutenção (Schach [9]).

3.2. Ferramentas de desenvolvimento

O desenvolvimento do aplicativo foi realizado na plataforma de programação Python, por ser uma linguagem com as seguintes características: multiplataform, permitir ser desenvolvida usando o paradigma da orientação a objetos, possuir licença gratuita, facilidade de modificação e ter um crescente uso no meio científico (Millman *et. al.* [10]). Python é uma plataforma de desenvolvimento que possui bibliotecas matemáticas eficientes para a computação científica, como as que foram utilizadas no desenvolvimento do CableSim: NumPy e Matplotlib. Numpy permite a realização de operações matemáticas com matrizes utilizando o recurso de vetorização que resulta em uma maior velocidade dos cálculos, visto que as operações são realizadas por bibliotecas binárias maduras e otimizadas para essa finalidade. Matplotlib é um pacote gráfico 2D usado para o desenvolvimento de aplicativos em Python, permitindo a geração de imagens com qualidade de publicação através de sua interface gráfica básica de usuário, que possibilita exportar os gráficos nos principais formatos de imagens (JPG, JPEG, PNG), em Portable Document Format (PDF) e Scalable Vector Graphics (SVG). Python também apresenta vantagens quando comparado com linguagens legadas, como C/C++ e Fortran, como é o que descreve o artigo de Rashed *et. al*. [11] destacando o seu uso na ciência computacional. Como Python é uma linguagem de uso geral, os programas desenvolvidos nesta linguagem apresentam execução um pouco mais lenta que programas criados na linguagem Java, mas podem ser desenvolvidos 3 a 5 vezes mais rápidos do que os aplicativos em Java.

Para realizar a modelagem do mundo virtual do CableSim optou-se pelo OpenGL moderno, que pertence a um consórcio de empresas voltadas para o estudo de hardwares e softwares gráficos, além de ser uma biblioteca multiplataforma com suporte aos novos recursos de hardware. A partir da versão 3.1 a biblioteca passou por diversas mudanças para se adequar e suportar os novos recursos de hardware, introduzindo o *pipeline* com *shaders* programáveis, ou seja, o *pipeline* programável (Angel *et. al*. [12]). Diante deste novo recurso do OpenGL, chamado atualmente de OpenGL moderno, todas as aplicações requerem *shaders* programáveis, que são programas desenvolvidos na linguagem GLSL (*OpenGL Shading Language*). Os *shaders* são executados diretamente no processador de renderização de gráficos em tempo real, chamado de GPU (*Graphics Processing Unit*). O GPU está cada vez mais presente em aplicações de propósito geral por possuir desempenho significativo no processamento paralelo de vértices (*Vertex Shader*) e fragmentos, chamado também de *pixels*, (*Fragment Shader*). O *vertex shader* é executado uma vez por vértice e sua função é efetuar operações matemáticas de transformações (rotação, translação e escala) dos vértices no

espaço tridimensional. O *fragment shader* é executado uma vez por pixel e realiza cálculos de cor, iluminação e textura.

Para desenvolver a interface do usuário, foi utilizada a biblioteca PyQt, a qual utiliza a linguagem de programação Python e o *framework* Qt. Qt é um *framework* de desenvolvimento que possui ferramentas apropriadas para facilitar a criação de aplicativos e interfaces de área de trabalho mais sofisticadas. Com PyQt é possível desenvolver aplicações para diversas plataformas, como Windows, Linux, Mac OS X e sistemas baseados em Unix (Summerfield [13]). Além de ser multiplataforma, Qt possui bindings para outras linguagens permitindo assim a portabilidade no desenvolvimento do software, está em constante atualização e têm-se grande base de conhecimento disponível. Essas características foram importantes para a escolha do Qt como interface gráfica do CableSim.

4. ANIMAÇÃO

A animação do cabo ocorre identificando-se a posição de cada elo em cada instante de tempo da simulação, sendo que esta posição é determinada através de três movimentos livres: elevação, azimute e torção. Informações sobre o tempo da animação também são importantes para que se tenha uma aproximação entre o tempo da simulação e o tempo real.

Com base no disposto, criou-se um padrão para que o CableSim importe de forma correta os dados essenciais para realizar a simulação do arquivo no formato TXT. O padrão é constituído de colunas que estão diretamente relacionadas com a quantidade de elos e aos movimentos angulares, ou seja, as colunas do arquivo são formadas pela quantidade de elos multiplicada por três (número de ângulos dos movimentos livres, elevação, azimute e torção, respectivamente). O mesmo deve possuir também uma última coluna que se refere ao tempo em cada instante da simulação (Fig. 2).

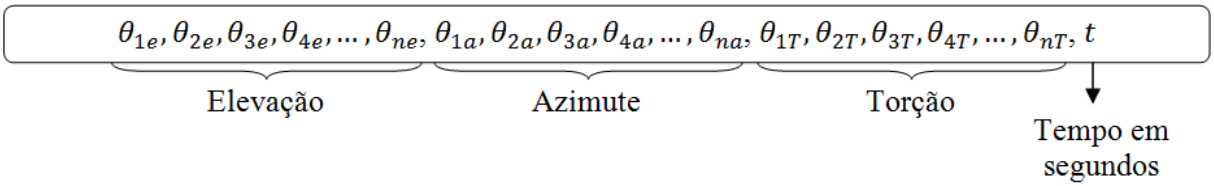


Figura 2: Representação de uma linha do arquivo TXT.

A variável n é o número de elos em que o cabo é dividido. As primeiras colunas referem-se aos movimentos angulares de elevação, as colunas seguintes constam os movimentos angulares de azimute, em seguida estão dispostos os movimentos angulares de torção, e a última coluna consta o tempo total da animação.

Além destas informações, percebeu-se que outras seriam necessárias para executar uma simulação, tais como: número de pontos do arquivo, passo, quantidade de elos e o tamanho do elo. Estas informações são definidas nesse artigo como parâmetros da animação. O parâmetro número de pontos do arquivo refere-se à quantidade de linhas que o arquivo de dados da simulação possui. Em cada linha do arquivo de dados encontram-se os ângulos tridimensionais referentes à posição de cada elo em um determinado instante de tempo. No software CableSim, as linhas são utilizadas para manipular os *frames* da animação.

Uma animação é o processo de mostrar uma sequencia de imagens em alta velocidade, suavemente diferentes, de forma que o observador tenha a sensação de movimento contínuo desconsiderando as imagens individuais. A velocidade em que as imagens são exibidas é

chamada de taxa de exibição e a unidade de medida é qps (quadros por segundo) ou fps (*frames* por segundo).

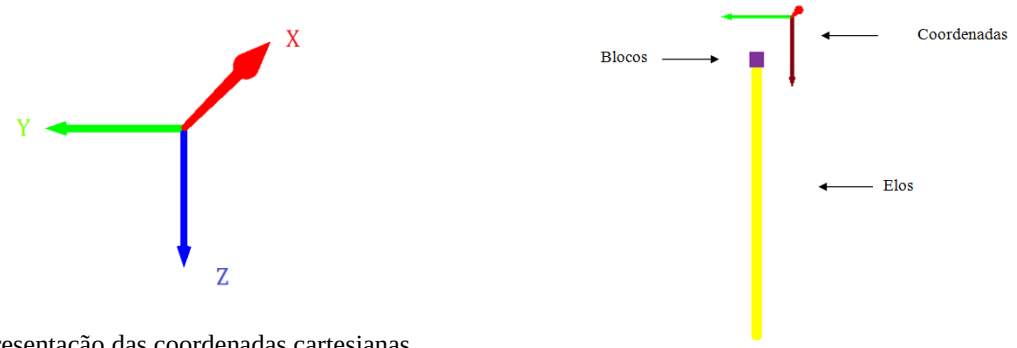
O passo determina a quantidade de linhas que será apresentada na animação. O passo é calculado utilizando-se o conceito de Progressão Aritmética gerando assim uma sequência numérica a partir do segundo elemento, pois o primeiro termo é sempre o número 1, ou seja, a primeira linha do arquivo TXT sempre é executada na animação. Portanto, o passo determina o número que é utilizado para somar ao seu antecessor formando a sequência de execução da animação. Por exemplo, se o passo for 1, todas as linhas do arquivo de dados serão animadas, caso o passo seja 2, serão animadas apenas as linhas ímpares do arquivo, tais como: 1, 3, 5, 7, 9, ... Caso o passo seja igual a 3, a animação ocorre considerando-se um pulo de execução de 3 em 3 linhas.

O parâmetro quantidade de elos refere-se à quantidade de elos em que o cabo umbilical é dividido. O parâmetro tamanho do elo especifica o tamanho de cada elo na unidade de medida em metros. Com essas duas informações o sistema calcula o tamanho do cabo.

Estes quatro parâmetros também são essenciais para executar uma animação. Mesmo que alguns parâmetros se consigam obter do arquivo de dados, optou-se por armazená-los separadamente com o objetivo de organizar o processo de animação, facilitar a interface com o usuário e também minimizar o acesso a esse arquivo agilizando assim a animação. O armazenamento desses parâmetros dá-se em um arquivo no formato XML implementado utilizando-se a biblioteca ElementTree. ElementTree é uma biblioteca utilizada para criar arquivos no formato XML, possui fácil integração com a linguagem de programação Python e ainda, têm licença livre.

Tendo-se definido os arquivos XML e TXT que contêm os parâmetros e os dados da simulação, respectivamente, têm-se as informações imprescindíveis para se efetuar uma animação no ambiente tridimensional.

Antes de efetivamente executar uma simulação, o software cria o mundo virtual para então, realizar uma animação, ganhando assim desempenho computacional. O mundo virtual é formado por um ambiente subaquático considerando-se um cabo fixo na extremidade superior e conectado a um ROV na parte inferior e por um sistema de referencia inercial definido horizontalmente no plano XY e verticalmente com eixo Z para baixo utilizando a regra da mão direita para representar o sentido de giro. Neste *software* o cabo umbilical é formado por um conjunto de cilindros sendo que cada cilindro representa um elo. O ROV é desenhado com dois paralelepípedos chamados, neste trabalho, de blocos. As coordenadas cartesianas são formadas pela junção das seguintes figuras geométricas: cone, cilindro e esfera. O sistema de coordenadas da Fig. 3 (a) é uma representação do sistema inercial, com Z para baixo e X na direção do observador. A visualização é feita por uma câmera sintética posicionada de frente para o plano YZ. Uma câmera sintética é um artificio utilizado para projetar no plano uma parte do mundo virtual como se fosse visto por um observador para dar realismo à cena, como se uma pessoa de fato estivesse observando, usualmente a projeção é realizada com o efeito de perspectiva. A Figura 3 (b) mostra o resultado da criação dos objetos do mundo virtual contendo as coordenadas cartesianas, quatro elos e os blocos que representam a carga terminal.



(a) Representação das coordenadas cartesianas

(b) Exemplo que ilustra a criação do mundo virtual com 4 elos

Figura 3: Elementos do mundo virtual

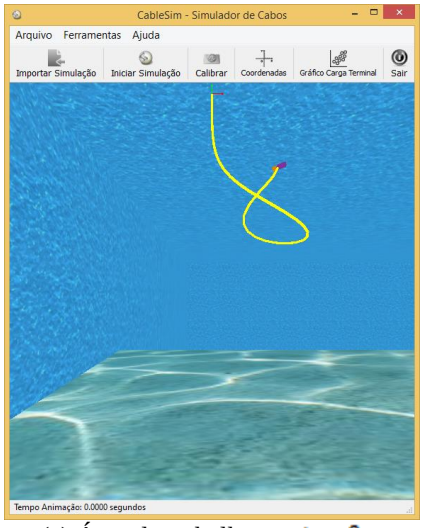
A animação é gerenciada pelo *framework* Qt implementando um temporizador, de acordo com o parâmetro passo, que neste projeto é o principal responsável por executar os *frames* da animação. As coordenadas (x, y, z) de cada elo são calculadas em um instante de tempo t, sendo que esse instante varia entre t_0 e t_{np}, onde np é o número de pontos da simulação. Ao mesmo tempo em que são calculadas as coordenadas de cada elo aplicam-se as transformações homogêneas de rotação e translação de acordo com os ângulos de elevação e azimute e as coordenadas calculadas de cada elo no instante t. Ainda no mesmo instante t calcula-se a transformação homogênea de rotação no eixo Z. No instante de tempo $t_0 = 0$, o aplicativo desenha o cabo de acordo com os parâmetros tamanho do elo e quantidade de elos e também os ângulos de elevação, azimute e torção.

O mundo virtual é composto por uma lista de objetos, no caso do atual projeto, são figuras geométricas. Para minimizar os recursos de hardware ao executar uma animação, o programa gerencia a lista de objetos do mundo virtual, ou seja, em cada instante t, altera-se a posição de cada objeto da lista, portanto os objetos são desenhados na cena uma única vez por animação evitando a utilização de memória sem necessidade.

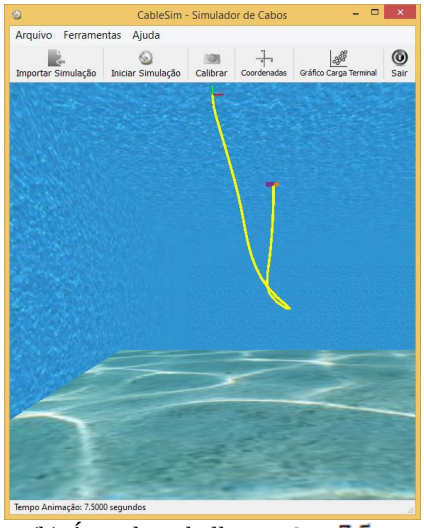
5. RESULTADOS

O aplicativo desenvolvido gera animações de simulações permitindo a análise do comportamento do cabo no espaço tridimensional conforme o número de elos considerados em cada simulação. O cabo foi considerado submerso em água e, portanto, está sujeito ao arrasto hidrodinâmico, às correntes subaquáticas e às forças de empuxo. Considera-se um cabo com comprimento de 1.200 metros, diâmetro de 1 cm e 32 elos, sendo 600 kg a massa da exterminada inferior. O arquivo utilizado para essa animação possui 12.001 linhas com pulo de execução igual a 2 e passo de integração de 9 ms, ou seja, a velocidade da animação é de 107,15 fps.

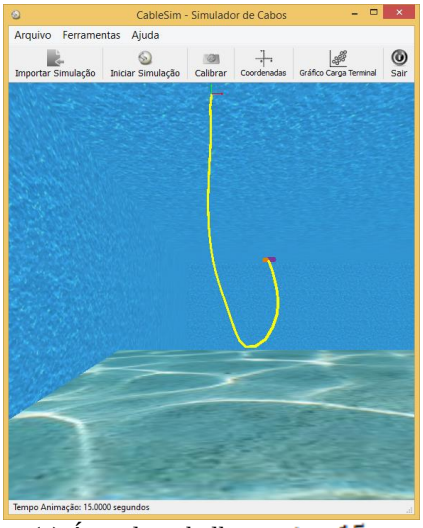
As animações são apresentadas a partir de captura de *frames* com o objetivo de considerar o movimento do cabo durante sua evolução temporal. A Fig. 4 apresenta uma simulação com tempo de 60 segundos e o movimento do cabo a uma velocidade de 7,5 fps. Na Figura 4 (a), o cabo é solto no tempo $t = 0$ e nas imagens entre a Fig. 4 (b) e a Fig. 4 (i) tem-se uma sensação de que o cabo está caindo em queda livre, sem atuação de forças, até ficar totalmente esticado em $t \cong 28{,}8$. Em seguida ele sobe devido ao impacto causado quando esticado.



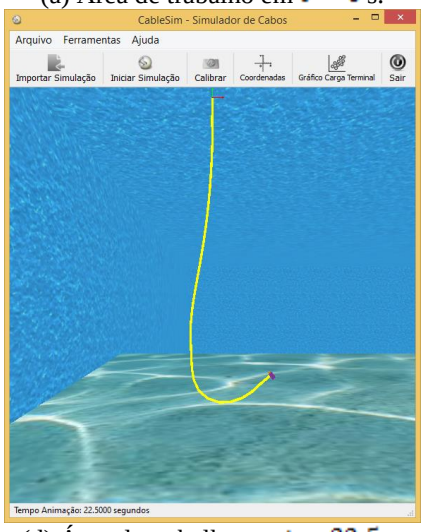
(a) Área de trabalho em $t = 0$ s.



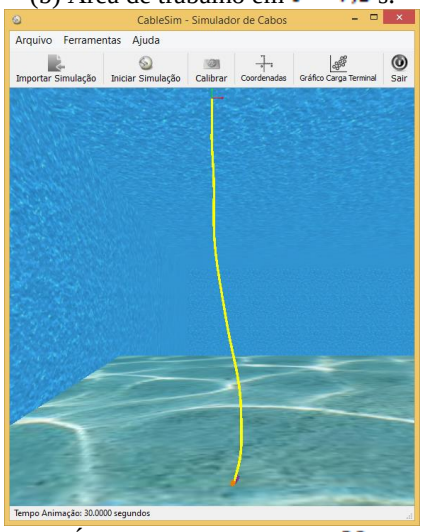
(b) Área de trabalho em $t = 7{,}5$ s.



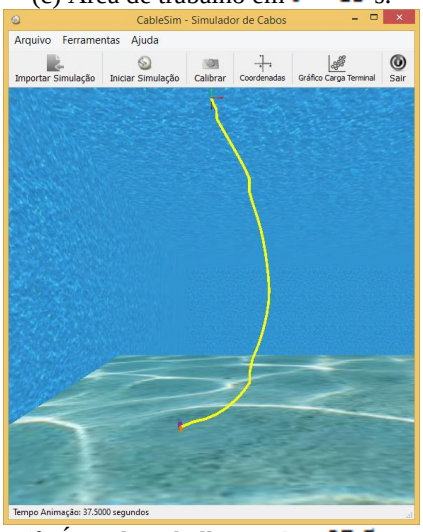
(c) Área de trabalho em $t = 15$ s.



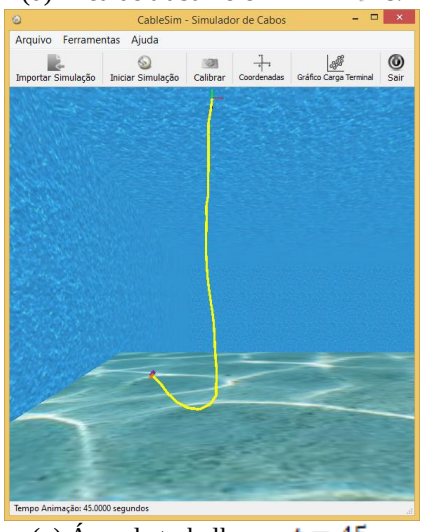
(d) Área de trabalho em $t = 22{,}5$ s.



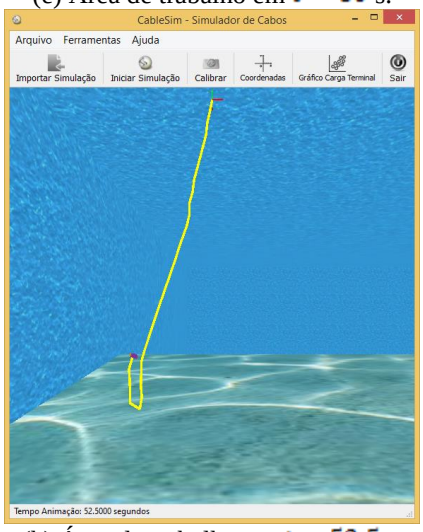
(e) Área de trabalho em $t = 30$ s.



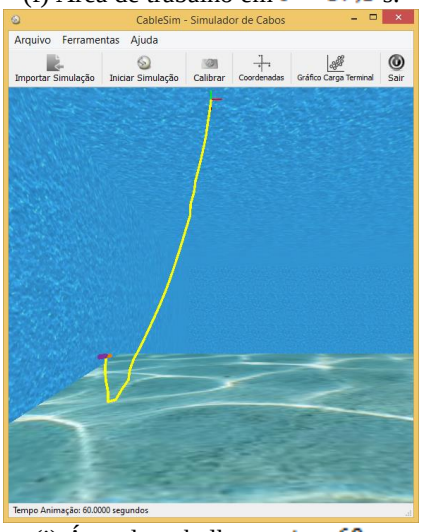
(f) Área de trabalho em $t = 37{,}5$ s.



(g) Área de trabalho em $t = 45$ s.



(h) Área de trabalho em $t = 52{,}5$ s.



(i) Área de trabalho em $t = 60$ s.

Figura 4: Animação de 7,5 fps: cabo com 32 metros de comprimento divido em 32 elos de tamanho 1 metro.

6. CONCLUSÕES

Dinâmicas com muitos graus de liberdade, como é o caso de estruturas flexíveis do tipo cabo, necessitam que suas simulações sejam analisadas a partir de animações em três dimensões. Neste sentido, o presente artigo mostra em detalhes o desenvolvimento de um software que permite realizar animações em cabos com veículos do tipo ROV como carga terminal. Portanto, como produto principal resultante deste artigo tem-se o aplicativo CableSim, que implementa as funcionalidades essenciais para a realização de animações tridimensionais de estruturas flexíveis do tipo cabo, permitindo assim uma melhor análise dos resultados de uma simulações. O software desenvolvido permitiu concluir que a modelagem dinâmica proposta possibilita uma grande sensação de realidade física, validando assim a modelagem de forma qualitativa.

REFERÊNCIAS

[1] **Pereira, A.E.L.,** 2010, *"O Método da Decomposição de Adomian Aplicado à Interação Fluido-estrutura de um Cabo,"* Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, 178 p.

[2] **Zanela, E.B.,** 2013, *"Modelagem Analítica de Estruturas do Tipo Cabo para Aplicações Subaquáticas,"* Universidade Federal do Rio Grande, Rio Grande, Brasil, 137 p.

[3] **Pereira, A.E.L., Gomes, S.C.P. and Bertoli, A.**, 2013, *A new formal-ism for the dynamic modeling of cables*. Mathematical and Computer Modelling of Dynamical Systems: Methods, Tools and Applications in Engineering and Related Sciences. vol. 19, no. 3, pp. 263-276.

[4] **Gomes, S. C. P., Zanela, E. B., Pereira, A. E. L.**, 2016. *Automatic generation of dynamic models of cables*. Ocean Engineering, 121, pg 559–571.

[5] **Oliveira, V. S.**, 2015. *Algoritmos genéricos para a geração de modelos dinâmicos de cabos umbilicais e veículos subaquáticos*. Dissertação de Mestrado, Modelagem Computacional, FURG.

[6] **Sommerville, I.**, 2011, *Engenharia de Software*. 9. ed. São Paulo: Pearson Prentice Hall.

[7] **Balzert, H.**, 2008, *UML 2: compacto*. Rio de Janeiro: Elsevier.

[8] **Booch, G.; Rumbaugh, J.; Jacobson, I.**, 2005, *UML: guia do usuário*. 2. ed. Rio de Janeiro: Elsevier.

[9] **Schach, S. R.**, 2010, *Engenharia de software: os paradigmas clássico e orientado a objetos*. 7. ed. São Paulo: McGraw-Hill.

[10] **Millman, K. J.; Aivazis, M.**, 2011, *Python for Scientists and Engineers*. Computing in Science & Engineering, vol.13, no. 2, pp. 9-12.

[11] **Rashed, M. G.; Ahsan, R.**, 2012, *Python in Computational Science: Applications and Possibilities*. International Journal of Computer Applications, vol. 46, no. 20, pp. 26-30.

[12] **Angel, E.; Shreiner, D.**, 2011, *Teaching a Shader-Based Introduction to Computer Graphics*. IEEE Computer Graphics and Applications, vol. 31, no. 2, pp. 9-13.

[13] **Summerfield, M.**, 2007, *Rapid GUI programming with Python and Qt: The Definitive Guide to PyQt Programming*. United States of America: Pearson Education.