DINCON 2017
CONFERÊNCIA BRASILEIRA DE DINÂMICA, CONTROLE E APLICAÇÕES
30 de outubro a 01 de novembro de 2017 – São José do Rio Preto/SP

Plataforma de Testes para Pilotos Automáticos aplicados a VANT's

Diogo de Oliveira Costa[1]

Departamento de Eletrônica Aplicada, ITA, São José dos Campos, SP

Paulo Rogerio Ferreira[2]

Departamento de Eletrônica Aplicada, ITA, São José dos Campos, SP

Neusa Maria Franco de Oliveira[3]

Departamento de Eletrônica Aplicada, ITA, São José dos Campos, SP

Resumo. Este artigo tem por objetivo o desenvolvimento de uma plataforma de testes em *Software-In-the-Loop* (SIL) para executar testes de funcionamento de um sistema de Piloto Automático (PA). O desenvolvimento do PA partiu de um controlador em tempo contínuo que posteriormente foi discretizado. Foi programado em Python um ambiente para realização do controle discreto assim como os blocos de navegação e guiagem para operar em SIL com o Matlab/Simulink. A malha fechada do sistema, composta pelo modelo da aeronave e o ambiente de simulação dos controles, permite obter as respostas da aeronave dado comandos de referência no sistema de piloto automático.

Palavras-chave. *Software-In-the-Loop*, SIL, Piloto Automático, Guiagem, Navegação

1 Introdução

A indústria aeroespacial está entre as primeiras a desenvolver simulações com o objetivo de desenvolver formas viáveis e seguras de realizar testes sobre sistemas de controle de voo, muito embora, desde o início da utilização destes testes, as aplicações passaram a ser bastante diversificadas, como indicado por [5, 6].

Dentre as técnicas possíveis de serem utilizadas para simulações estão o *Software-In-the-Loop* (SIL) e o *Hardware-In-the-Loop* (HIL). Ambas as técnicas são amplamente utilizadas por engenheiros e projetistas para avaliar e validar, de forma segura e com baixo custo, os componentes de um sistema em desenvolvimento [4].

No caso da simulação SIL, apenas o *software* normalmente executado no sistema embarcado é testado. Todo o *hardware* envolvido é substituído por uma camada de *software*

[1]eng.costadiogo@gmail.com
[2]paulorogerio01@gmail.com
[3]neusa@ita.br

de emulação especial que lê ou insere dados no nível de interface de *software* mais baixo possível. Muitos dispositivos reais são substituídos por modelos matemáticos, principalmente quando não se tem as características de funcionamento bem definidas, o modelo é muito complexo ou na verificação de determinados requisitos de projeto.

A aplicação do SIL permite realizar diferentes testes em diferentes cenários, em um ambiente com situações próximas do real. Além disso, o SIL permite que todos os erros no software sejam solucionados. O HIL lida com a reprodução do ambiente real onde o sistema em desenvolvimento será embarcado e assim traz à tona os erros que ocorrem devido à interação do *software* com o *hardware* [1].

O objetivo do presente trabalho é contribuir para o desenvolvimento tecnológico por meio do estudo e implementação em SIL de um sistema de piloto automático (PA) de uma aeronave de testes. O PA é constituído por um bloco de guiagem, um bloco de navegação e um bloco de controle longitudinal e latero-direcional, sendo este último desenvolvido por [2].

2 Propósito

Este trabalho tem como objetivo apresentar uma plataforma de testes para PA de aeronaves, especificamente de aeronaves de asas fixas.

A aeronave que o PA controlará na plataforma de teste desenvolvida e apresentada neste trabalho é o modelo matemático do *Remotely Piloted Aircraft System* (RPAS) Prometheus [7]. Esta aeronave possui uma envergadura de 3,2m e um peso de decolagem máxima de 31kg. Além disso, opera com um único motor na configuração de impulsor, isto é, com a hélice na parte de trás. Isto permite uma variedade de possibilidades para fixação de sensores, por exemplo, câmeras na frente da aeronave [7].

O controlador para o RPAS, projetado por [2] tem foco no controle de velocidade, ângulo de inclinação e ângulo de azimute do trajeto de voo, para operar em uma determinada condição de voo. Este controlador será simulado em um ambiente desenvolvido em linguagem Python, juntamente com os blocos de navegação e guiagem, com intuito de operar em SIL através da UDP com o Matlab/Simulink, onde estará o modelo da aeronave. Assim, as respostas da aeronave serão simuladas no Matlab® uma vez recebidos os comandos de referência gerados no sistema de piloto automático desenvolvido no ambiente Python.

3 Métodos

O projeto de um sistema de piloto automático para veículos autônomos possui várias etapas como o modelamento matemático, categorização da aeronave e testes do sistema. Visto isso, os objetivos deste trabalho são desenvolver uma plataforma de testes capaz de simular o ambiente onde o PA será embarcado, com suas conexões entre os softwares e a etapa de testes do PA.

O uso de uma plataforma de testes em bancada diminui a demanda de vários fatores que dificultam ensaios em campo como por exemplo o grande espaço físico necessário para

testes em voo. Além da economia de recursos, o sistema diminui o tempo gasto em testes preliminares que são necessários para a busca da estabilidade do sistema [3, 8].

O sistema consiste em um *software* que simula a aeronave enquanto o PA recebe os estados desta, realiza o seu processamento e realimenta as informações fechando assim a malha de controle. Esse tipo de abordagem possibilita um rápido retorno das informações dos testes [7]. O PA foi escrito em Python que é uma linguagem livre e multiplataforma.

3.1 Aeronave

Uma abordagem bastante empregada na representação dos modelos de aeronave a ser utilizada no projeto de controladores é a linearização do modelo em torno de um ponto de equilíbrio. Após a linearização, aplica-se as ferramentas da teoria de controle para o projeto dos controladores.

O modelo linearizado do RPAS utilizado neste trabalho foi obtido com base no seguinte ponto de ajuste do envelope de voo: velocidade $V_{\mathrm{k}} = 35m/s$ e altitude $h = 1500m$ [2].

Segundo [2], a aeronave foi linearizada e posteriormente teve seus movimentos longitudinal e latero-direcional desacoplados em espaços de estados no modelo $\dot{x} = Ax + Bu$ e saída $y = Cx + Du$.

3.2 Modelo Longitudinal Linearizado

As variáveis de estados desacopladas para o movimento longitudinal são:

$$x = \begin{bmatrix} u & w & q & \theta & z \end{bmatrix}^{\mathrm{T}} \tag{1}$$

$$u = \begin{bmatrix} \delta_{\mathrm{th}} & \delta_{\mathrm{el}} \end{bmatrix}^{\mathrm{T}} \tag{2}$$

$$y = \begin{bmatrix} u & w & q & \theta & z\gamma & V_{\mathrm{k}} \end{bmatrix}^{\mathrm{T}} \tag{3}$$

Onde:

u	componente x da velocidade no sistema corpo
w	componente z da velocidade no sistema corpo
q	taxa de variação do ângulo de arfagem
θ	ângulo de arfagem
z	posição vertical no NED
γ	ângulo de trajetória
V_{k}	velocidade da aeronave
δ_{th}	potência aplicada ao motor
δ_{el}	comando da superfície de profundor

3.3 Modelo Latero-Direcional Linearizado

As variáveis de estados desacopladas para o movimento latero-direcional são:

$$x = \begin{bmatrix} v & p & r & \phi & \psi \end{bmatrix}^{\mathrm{T}} \quad (4)$$

$$u = \begin{bmatrix} \delta_{\mathrm{a}} & \delta_{\mathrm{r}} \end{bmatrix}^{\mathrm{T}} \quad (5)$$

$$y = \begin{bmatrix} v & p & r & \phi & \psi & \gamma & a_{\mathrm{y}} \end{bmatrix}^{\mathrm{T}} \quad (6)$$

Onde:

v	componente y da velocidade no sistema corpo
p	taxa de variação do ângulo de rolagem
r	taxa de variação do ângulo de guinada
ϕ	ângulo de rolagem
ψ	ângulo de guinada
γ	ângulo de trajetória
a_{y}	aceleração da aeronave
δ_{a}	comando da superfície de aileron
δ_{r}	comando da superfície de leme

3.4 Controle e discretização do sistema

Os controladores em tempo contínuo, utilizados neste trabalho foram obtidos por meio do pacote SISOTOOL do Matlab a partir das equações características obtidas dos modelos longitudinal e latero-direcional em espaço de estados [2].

Existem diversos métodos de discretização e estes consistem essencialmente em converter uma função G(s) em G(z), observando alguns critérios. Neste trabalho, para discretização do sistema foi observado o funcionamento do sistema em malha fechada, e as funções de transferência da malha. Desta forma ao empregar o método de *Tustin*, o sistema dinâmico de tempo contínuo foi discretizado dado as entradas e um tempo de amostragem de 0,05s.

3.5 Plataforma de teste

O sistema projetado foi executado em um computador pessoal com processador Intel® Core™ i3 com CPU de 2GHz, 4GB de memória RAM, com sistema operacional Windows 10 com o auxílio do Matlab® 7.12.0(R2011a) e um interpretador de Python 2.7. O sistema foi subdividido em duas porções, sendo uma no Matlab® e outra em porção em Python onde é simulado o PA.

A figura 1 exibe os blocos do PA e a maneira que ele foi subdividido. O primeiro bloco é responsável pelo recebimento das informações através da UDP, em seguida o componente do sistema referente à navegação, que envia esse resultado para o bloco de guiagem. O bloco de guiagem calcula as correções na atitude da aeronave e envia para os controladores

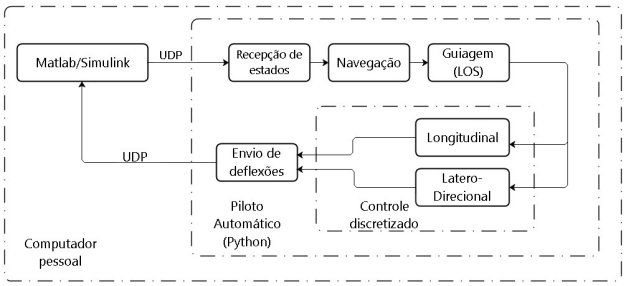

Figura 1: Diagrama de blocos do SIL.

latero-direcional e longitudinal que envia através da UDP de volta ao Matlab/Simulink para fechar o laço de controle.

O bloco de navegação consiste em rotacionar as velocidades da aeronave do sistema corpo para o sistema NED e de posse dessas informações é feita a integração dessas velocidades para se obter as posições atuais da aeronave.

O bloco de guiagem calcula as correções na atitude da aeronave para ser alcançado o *waypoint* (WP). Foi usado o método *Line Of Sight* (LOS) para determinar a guinada da aeronave, este determina as referências de um controle PID que fornece o sinal para o bloco de controle latero-direcional. O sinal de referência do controle longitudinal é fornecido por outro controle PID que tem como referência a diferenças entre a altitude atual e a esperada.

O bloco de controle de posse das novas referências, faz o cálculo das deflexões das superfícies de controle da aeronave e as envia para o Matlab® através da UDP.

4 Resultados

Foi construído um controlador para ser testado na plataforma SIL usando o modelo linearizado da aeronave. Os controladores foram discretizados usando a ferramenta C2D do Matlab® com o método de *Tustin* e passo de tempo 0,05s, esse controlador foi programado em Python que é uma linguagem que pode ser embarcada em dispositivos eletrônicos.

Foram escolhidos quatro pontos de passagem onde houvesse mudanças tanto na altitude quanto na direção do voo. O controle de guiagem responsável pela seleção dos WP troca a sua referência ao alcançar um raio de 10m do ponto de destino. Como referência para a trajetória foi plotado o percurso de voo sendo feito totalmente em Matlab® com o mesmo sistema de controle porém em tempo contínuo, sendo os mesmos WP.

O PA implementado em linguagem Python foi utilizado para realizar uma missão, controlando a aeronave usando seus modelos matemáticos longitudinal e latero-direcional simulados no Matlab®. A missão é visualizada em coordenadas NED (North-East-Down). A Figura 2 mostra o comportamento da aeronave quanto a altitude, foi verificado que o controle apresentou um sobressalto ao alcançar o ponto de passagem, porém com essa exceção o controlador foi capaz de manter o valor esperado em regime permanente.

A Figura 3 mostra o comportamento da trajetória em NE mostrando os pontos de

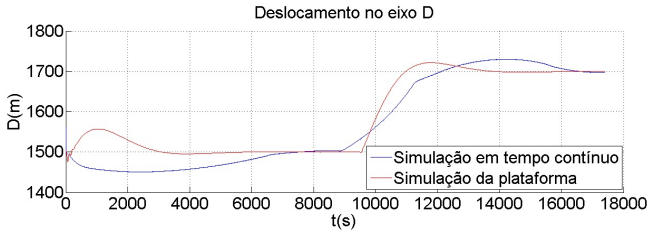

Figura 2: Deslocamento no eixo D comparando a resposta da plataforma desenvolvida (sistema discretizado) com a simulação de um sistema contínuo.

passagem, onde se pode ver que a plataforma foi capaz de seguir os WP especificados. Foi verificada uma leve oscilação na guinada da aeronave, a qual pode ter acontecido devido aos ganhos escolhidos para o controlador de guiagem. Nas Figuras 2 e 3 pode-se notar a passagem nos WP em todos as direções verificando um bom resultado no desacoplamento dos dois modos da aeronave.

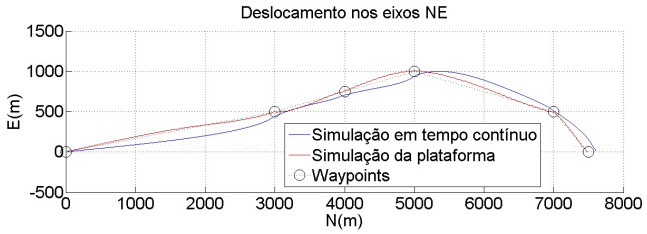

Figura 3: Deslocamento da aeronave em NE mostrando os WP e comparando resposta da plataforma desenvolvida (sistema discretizado) com a simulação de um sistema contínuo.

5 Discussões

Durante o processo de construção dos controladores verificou-se as dificuldades apontadas pelos autores [3, 4, 8]. Os autores tiveram problemas na implementação do dispositivo em sistema embarcado. Foram feitas tentativas de implementação da comunicação através da serial, porém a velocidade da conexão ficou abaixo da necessária para o algoritmo. Por isso, optou-se pela utilização do protocolo UDP que é mais veloz por não possuir a confirmação de recebimento da informação.

O raio de abrangência dos WP foi fixada em 10m por tentativa e erro. Com um raio menor o controlador diminui o nível de acertos do ponto, fazendo com que a aeronave fique circulando o WP até conseguir executar uma curva que passe pelo WP.

As curvas da aeronave durante trajetórias retas foi um comportamento inesperado, porém isso talvez possa ser resolvido mudando o tipo do controlador que envia a referência ao controle latero-direcional. Ocorreram oscilações na altitude porém, ao mudar os ganhos do controlador, foram alteradas. Um ajuste no ganhos dos controladores podem levar a uma diminuição nas oscilações na altitude.

6 Conclusões

Este trabalho expõe os resultados obtidos na implementação de um controle longitudinal e latero-direcional para um sistema de piloto automático. Para o projeto deste sistema utilizou-se a técnica de discretização do modelo por meio da função C2D do Matlab® e posteriormente a configuração e programação do SIL em linguagem Python.

As incertezas verificadas nos resultados do controle implementado em SIL podem ser explicadas tanto pelo método de discretização, escolhido para obtenção dos controles embarcados, como pelos protocolos de comunicação adotados para transmissão de informações na malha.

A utilização da ferramenta Matlab® mostrou-se de grande importância para a conclusão deste trabalho, uma vez que a mesma proporcionou uma análise/comparação adequada das respostas da aeronave mediante as entradas de referências aplicadas ao sistema.

Referências

[1] E. Atkins, A. Ollero, A. Tsourdos, Unmanned Aircraft Systems, Encyclopedia of aerospace engineering, John Wiley & Sons, (2017).

[2] E. S. Bianchetti, A Flight Control System for UAV Prometheus, São José dos Campos, Trabalho de Conclusão de Curso (Graduação), Instituto Tecnológico de Aeronáutica, (2012).

[3] A. Bittar, H. V. Figuereido, P. A. Guimaraes and A. C. Mendes, Guidance Software-In-the-Loop simulation using X-Plane and Simulink for UAVs, Unmanned Aircraft Systems (ICUAS), 2014 International Conference on, Orlando, FL, pp. 993-1002, (2014).

[4] P. Chudy, P. Dittrich and P. Rzucidlo, Hil simulation of a light aircraft flight control system, 2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC), Williamsburg, VA, pp. 6D1-1-6D1-13, (2012).

[5] MACLAY, David. Simulation gets into the loop. IEE Review, v. 43, n. 3, p. 109-112, (1997).

[6] S. Nabi, M. Balike, J. Allen and K. Rzemien, An Overview of Hardware-In-the-Loop Testing Systems at Visteon, SAE Technical Paper 2004-01-1240, (2004), DOI:10.4271/2004-01-1240.

[7] Dlr.de. (2017). DLR - DLR Portal. [online] Disponível em: http://www.dlr.de [Acessado 10 Jun. 2017].

[8] P. Lu and Q. Geng, Real-time simulation system for UAV based on Matlab/Simulink, Computing, Control and Industrial Engineering (CCIE), 2011 IEEE 2nd International Conference on, Wuhan, pp. 399-404, (2011).