INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

RODRIGO SCHNEIATER

ESTUDO COMPARATIVO DE UM CONTROLADOR PID DE ORDEM FRACIONÁRIA COM UM CONTROLADOR PID CONVENCIONAL

Dissertação de mestrado apresentada ao curso de pós-graduação Stricto Sensu de Automação e Controle de Processos do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo como parte dos requisitos para a obtenção do título de Mestre em Automação e Controle.

Professor Orientador: Dr. Alexandre Brincalepe Campo

São Paulo

2014

S385e SCHNEIATER, Rodrigo.
Estudo comparativo de um controlador PID de ordem fracionária com um controlador PID convencional / Rodrigo Schneiater. São Paulo: [s.n.], 2014.
76 f.: il.; 30 cm.

Orientador: Prof. Dr. Alexandre Brincalepe Campo.
Dissertação (Mestrado Profissional em Automação e Controle de Processos) - Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, IFSP, 2014.

1. Cálculo fracionário 2. Controle fracionário 3. PID
4. $PI^{\lambda}D^{\mu}$ 5.LabVIEW 6. CompactRIO
I. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo. II. Título

CDU 681.0

MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO
CAMPUS SÃO PAULO
DIRETORIA GERAL DO *CAMPUS* SÃO PAULO
Coordenadoria de Registros Escolares de Pós-Graduação

ATA DE EXAME DE DEFESA DE DISSERTAÇÃO

Nome do Programa: **Mestrado Profissional em Automação e Controle de Processos**

Nome do(a) Aluno(a) : Rodrigo Schneiater

Nome do Orientador: Prof. Dr. Alexandre Brincalepe Campo

Nome do Co-orientador:

Título do Trabalho: "Estudo comparativo de um controlador PID de ordem fracionária com um controlador PID convencional"

Abaixo o resultado de cada participante da Banca Examinadora

Nome completo dos Participantes Titulares da Banca	**Sigla da Instituição**	**Aprovado / Não Aprovado**
Prof. Dr. Alexandre Brincalepe Campo – Orientador	IFSP – SPO	P/P APROVADO
Prof. Dr. Eduardo Alves da Costa – Membro Interno	IFSP – SPO	APROVADO
Prof. Dr. Diego Colón – Membro Externo	POLI – USP	APROVADO
Nome completo dos Participantes Suplentes da Banca	**Sigla da Instituição**	**Aprovado / Não Aprovado**
Prof. Dr. Ricardo Pires – Membro Interno	IFSP - SPO	
Prof. Dr. Alexandre Simião Caporali – Membro Interno	IFSP – SPO	

Considerando-o: [X] APROVADO
[ ] NÃO APROVADO

Assinaturas

São Paulo, 10 de abril de 2014

P/P Eduardo Costa
Presidente da Banca

Eduardo Costa
Membro Interno

Diego Colón
Membro Externo

Observações:
O Prof. Eduardo A. Costa estava como membro da banca e representou o Prof. Alexandre Campo que participou da banca via video conferência.

Agradecimentos

Agradeço ao apoio e incentivo de meus familiares, em especial meus pais e minha esposa que me apoiaram nos momentos mais difíceis e deram a força para eu continuar em frente. Agradeço também todo o apoio prestado pelos professores do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, em especial o Professor Doutor Alexandre Brincalepe Campo, meu orientador e um grande amigo, que sempre mostrou o caminho certo a seguir para chegar ao fim do trabalho com sucesso, mesmo quando a distância se impôs como um grande obstáculo. Agradeço ao colega e amigo Alexandre Barp que ajudou em vários momentos a respeito do LabVIEW Control Design & Simulation.

Resumo

O cálculo fracionário vem sendo cada vez mais aplicado a sistemas de controle, porém devido ao alto grau de complexidade envolvida e à falta de ferramentas estabelecidas e de métodos de sintonia, poucos são os experimentos práticos realizados. Este trabalho apresenta um estudo comparativo de desempenho entre um controlador PID convencional e um controlador de ordem fracionária $PI^{\lambda}D^{\mu}$, onde são analisados aspectos de tempo de resposta, sobressinal e erro do controlador. Para o desenvolvimento e simulação dos sistemas de controle é utilizado o ambiente LabVIEW e para a implementação do sistema é utilizado o controlador de tempo real, CompactRIO – ambos da National Instruments. Para o experimento, é realizado o controle de posição de um motor CC por meio da planta educacional QET DC Motor da Quanser. A implementação do controlador $PI^{\lambda}D^{\mu}$ no LabVIEW é feita de maneira genérica, de forma que permite o uso deste em outros trabalhos como ferramenta de controle. O controlador tem como entradas o sinal de erro, E(s), e os cinco parâmetros do controlador (K_p, K_i, λ, K_d, e μ) e o sinal de saída, U(s).

Palavras chave: Cálculo Fracionário, Controle Fracionário, PID, $PI^{\lambda}D^{\mu}$, LabVIEW, CompactRIO.

Abstract

Fractional order calculus is more and more applied to control systems, but there are few practical experiments performed considering high complexity involved, lack of established tools and tuning methods. This work presents a comparison study between a conventional PID controller and a fractional order $PI^{\lambda}D^{\mu}$ controller, where the analyzed aspects include controller response time, signal overshooting, error. For the development and simulation of the control systems, it is used LabVIEW development system, and, for the system deployment, the CompactRIO real-time controller – both from National Instruments. For the experiment, it is performed the position control of a DC motor through the educational QET DC Motor plant, from Quanser. The implementation of the $PI^{\lambda}D^{\mu}$ controller in LabVIEW is made generically, as a tool for fractional control. This approach enables other works to use this tool instead of developing all fractional calculus and control systems from the scratch. The controller has error signal, E(s), and the five controller parameters (K_p, K_i, λ, K_d, e μ) as inputs, and the output signal, U(s).

Keywords: Fractional Calculus, Fractional Control, PID, $PI^{\lambda}D^{\mu}$, LabVIEW, CompactRIO.

Lista de Figuras

Sumário

1 INTRODUÇÃO

A teoria de cálculo de ordem fracionária vem sendo elaborada desde a época do desenvolvimento do cálculo diferencial e integral. Trata-se da possibilidade de operar uma derivada ou integral de ordem não inteira.

Porém devido à intrínseca complexidade de operar estas ferramentas de cálculo, suas aplicações foram adiadas, segundo Barbosa e Machado (2006). Atualmente com o avanço da tecnologia e dos métodos computacionais e numéricos é possível realizar as operações relacionadas a esta extensão do cálculo.

Uma das aplicações estudadas para o cálculo de ordem fracionária está em sistemas de controle. As técnicas de modelagem tradicionais, com operadores de ordem inteira, desconsideram alguns fenômenos físicos que modelos de ordem fracionária representam melhor. Da mesma forma, o controlador desenvolvido a partir de uma função de transferência de ordem inteira, por exemplo, um controlador PID, possui como ajustes os seguintes ganhos de atuação:

- Proporcional, que compreende a multiplicação de um valor numérico ao sistema.

- Integral, que compreende a multiplicação de um valor numérico a um operador integral de primeira ordem ao sistema.

- Derivada, que compreende a multiplicação de um valor numérico a um operador diferencial de primeira ordem ao sistema.

Já um controlador $PI^{\lambda}D^{\mu}$ de ordem fracionária possui adicionalmente o ajuste da ordem do operador integral e diferencial. Este controlador tem a seguinte função de transferência:

$$G_c(s) = \frac{U(s)}{E(s)} = K_P + K_I s^{-\lambda} + K_D s^{\mu} \quad (1)$$

$$(\lambda, \mu > 0)$$

No século XIX, Grünwald (1867) e Letnikov (1868) apresentaram equações para realizar o operador de ordem fracionária. Com seus estudos, foi possível observar que o mesmo operador tratava de integração ou derivação dependendo do sinal da ordem. A definição de Grunwald/Letnikov para o operador ${}_a D_t^r$, integro-diferencial, é a seguinte:

$${}_aD_t^r f(t) = \lim_{h \to 0} h^{-r} \sum_{j=0}^{\left[\frac{t-a}{h}\right]} (-1)^j \binom{r}{j} f(t - jh) \quad (2)$$

Onde [.] significa a parte inteira;

r é a ordem da operação. Para $r < 0$, a operação é uma integral e, para $r > 0$, uma derivada;

a e t são os limites da operação;

$$\binom{r}{j} = \frac{\Gamma(r+1)}{\Gamma(j+1)\Gamma(r-j+1)} \quad (3)$$

$\Gamma(.)$ é a função Gama, que é uma extensão da função fatorial com o seu argumento decrementado de um para números reais e complexos.

$\Gamma(t) = \int_0^{\infty} x^{t-1} e^{-x} dx$, fórmula geral para $t \in \mathbb{C}$ (4)

$\Gamma(n) = (n-1)!$, fórumla geral no caso particular de $t = n \in \mathbb{N}$ (5)

Segundo Podlubny, Dorcak e Kostial (1997), a somatória da definição Grunwald/Letnikov gera certa inconveniência durante a manipulação direta das derivadas fracionárias. A partir daí, surgiu a definição Letnikov- Riemann- Liouville:

$$ {}_aD_t^r f(t) = \frac{1}{\Gamma(n-r)} \frac{d^n}{dt^n} \int_a^t \frac{f(\tau)}{(t-\tau)^{r-n+1}} d\tau \quad (6) $$

Para (n-1 < r < n)

Baseando-se na definição de Letnikov-Riemann-Liouville, descreve-se a transformada de Laplace do integrador/derivador fracionário:

$$ \int_0^\infty e^{-st} \; {}_0D_t^r f(t) dt = s^r F(s) - \sum_{k=0}^{n-1} s^k \; {}_0D_t^{r-k-1} f(t) \Big|_{t=0} \quad (7) $$

Para (n-1 < $r \leq n$), onde $s \equiv j\omega$

Mas a definição Letnikov-Riemann-Liouville apresenta uma elevada complexidade em relação às condições iniciais das funções a serem operadas, contendo limites da derivada em t = a. Ainda que estas condições iniciais possam ser resolvidas matematicamente, elas não possuem significado físico, segundo Manabe (1961).

De forma a resolver este impasse das condições iniciais, Podlubny (1997, apud Caputo, 1967) propõe outra definição para o operador integrador/derivador fracionário:

$$ {}_aD_t^r f(t) = \frac{1}{\Gamma(r-n)} \int_a^t \frac{f^{(n)}(\tau)}{(t-\tau)^{r+1-n}} d\tau \quad (8) $$

Para (n-1 < r < n) e $\Gamma(.)$ é a função *Gamma*

Na definição deste autor, as condições iniciais da função são tratadas da mesma forma que em equações diferenciais de ordem inteira (Podlubny, 1997, apud Caputo 1967).

Este trabalho apresenta uma implementação preliminar do controlador de ordem fracionária $PI^{\lambda}D^{\mu}$ e um estudo comparativo da atuação deste controlador contra um controlador PID convencional. É apresentada a implementação do controlador em programação gráfica, LabVIEW da National Instruments, e os métodos utilizados para digitalização das funções.

1.1 Objetivos

A teoria de controle de ordem fracionária vem se popularizando e mas ainda são encontrados poucos estudos com implementação prática como em Jin et al (2009). Também são encontrados muitos estudos que apresentam partes do que seria necessário para realizar a implementação do sistema de controle em ordem fracionária como em Li, Luo, e Chen (2010). Este trabalho tem como principais objetivos:

- Levantar os principais equacionamentos do operador do cálculo fracionário;
- Estudar a implementação do controlador $PI^{\lambda}D^{\mu}$ de ordem fracionária;
- Estudar os métodos de discretização de operador de ordem fracionária;
- Desenvolver um programa em ambiente gráfico para a realização do controlador de ordem fracionária;
- Experimentar o controlador num sistema de controle de motor CC (corrente contínua);

- Comparar os resultados obtidos com o uso do controlador de ordem fracionária com aqueles obtidos com um controlador convencional com base na minimização do erro.

1.2 Justificativa e importância

Este trabalho propõe uma implementação prática de um controlador de ordem fracionária em um ambiente de desenvolvimento de aplicações amplamente usado no setor industrial para o desenvolvimento de controladores em aplicações de alta complexidade. Espera-se que com esta ferramenta seja possível desenvolver controladores mais eficientes e contribuir para resolver problemas de controle que os controladores PID tradicionais não apresentam resultados desejados.

A compilação de todo o processo de desenvolvimento e implementação de um controlador de ordem fracionária num ambiente digital em um único trabalho - desde a aproximação até a síntese do modelo em função de transferência – poderá contribuir como um guia para futuros trabalhos.

1.3 Estrutura do trabalho

Além desta introdução, este trabalho está estruturado em 6 capítulos.

O capítulo 2 apresenta uma visão geral das referências bibliográficas usadas no trabalho. O capítulo é estruturado numa sequência lógica que resume análises de

referências que vão desde artigos antigos onde o poder computacional era inexistente até trabalhos recentes que apresentam os métodos mais modernos para a implementação do controlador em ambiente digital.

No capítulo 3, são apresentados os recursos usados para a implementação prática do controlador no experimento de comparação da eficiência dos controladores. Serão apresentados em detalhes os recursos de programação gráfica utilizados, a plataforma de *Hardware* usada para executar o algoritmo de controle em tempo real e a planta a ser controlada.

O capítulo 4 detalha o trabalho de projeto e implementação do controlador em ambiente de programação gráfica. É apresentado todo o trabalho de aproximação e expansão em função de transferência, implementado num programa com os cálculos realizados de forma genérica possibilitando a parametrização do controlador no programa, sem a necessidade de modelagem em outro ambiente e possibilitando o reuso deste programa em outros trabalhos.

O capitulo 5 apresenta a experimentação realizada com o controlador implementado. São apresentados exercícios teóricos, execução em ambiente simulado, com não-linearidades e experimentos reais. Neste capítulo, também são analisadas as alterações de resposta do sistema impostas por alterações em cada um dos parâmetros do controlador $PI^{\lambda}D^{\mu}$.

O capítulo 6 apresenta as conclusões do trabalho e sugestões de pesquisas futuras associadas a este.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta as referências utilizadas no desenvolvimento deste trabalho, os métodos apresentados e fatores relevantes para este trabalho. A divisão está organizada da seguinte forma: revisão da teoria de cálculo e controle fracionário e referências relacionadas a fatores de implementação do controle fracionário.

2.1 Histórico e teoria do cálculo e controle fracionário

Manabe (1961) já apresentava aplicações relacionadas a integrais e derivadas de ordem não inteira. Ele usa como exemplo o caso da obtenção da corrente, i(t,0), que flui ao longo de um cabo semi-infinito quando uma tensão, v(t,0), é aplicada entre uma das extremidades e o terra. Levando-se em conta a capacitância e resistência do cabo, demonstra que a corrente é função da derivada de ordem 1/2 da tensão aplicada.

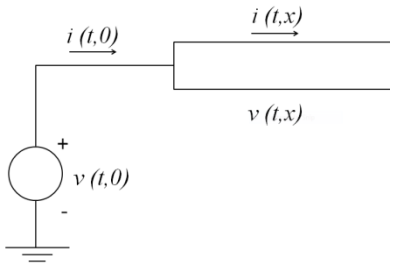

Figura 1 – Sistema de cabo semi-infinito

O estudo do comportamento da integral de ordem fracionária para diversas funções básicas demonstra os diferentes comportamentos em malha aberta, em malha fechada e de resposta em frequência. Todas as demonstrações ocorrem no domínio do tempo contínuo, uma vez que em 1961, a não representação no domínio do tempo discreto se dá devido à indisponibilidade de sistemas digitais para a implementação de controladores.

Podlubny, Dorcak e Kostial (1997), apresentam um estudo amplo sobre as principais definições da integral e derivada de ordem fracionária, discutindo as vantagens de cada modelo. Enfatizam o modelo de Caputo como vantajoso em relação às outras definições, principalmente sobre a abordagem em relação às condições iniciais nas equações diferenciais fracionárias terem a mesma forma que nas equações diferenciais de ordem inteira. A partir daí, os autores exploram a modelagem de sistemas por meio de modelos de ordem fracionária e demonstram que estes representam com maior fidelidade fenômenos físicos que os modelos de ordem inteira desconsideram, mesmo em sistemas consagrados como um forno em aquecimento. Os autores também apresentam o controlador $PI^{\lambda}D^{\mu}$ que é uma generalização do controlador PID e que permite, além dos ajustes dos ganhos proporcional, integral e derivativo, os ajustes das ordens do operador integral e derivativo. Dadas estas características, o controlador possibilita atuações mais eficientes, especialmente em sistemas descritos por modelos de ordem fracionária e tende a ser menos sensível às mudanças nos parâmetros do sistema controlado e do próprio controlador. Concluem que, em 1997, havia grande defasagem entre a teoria de controle fracionário e a implementação destes controladores em razão da escassez de trabalhos relacionados sua aplicação e de métodos numéricos e computacionais de resolução das equações diferenciais fracionárias.

Podlubny (1999) detalha o uso do controlador $PI^{\lambda}D^{\mu}$ comparando um cenário comum no projeto de um controlador com e sem soluções fracionárias. O uso de técnicas convencionais de modelagem podem levar ao desenvolvimento de modelos de ordem inteira que seriam melhor representados por modelos de ordem fracionária. A comparação da atuação de um controlador PD convencional e um de ordem fracionária PD^{μ} a uma "realidade" representada na forma de um sistema de ordem fracionária é feita a partir da representação da "realidade" por um modelo de ordem inteira. Os resultados demonstram claramente que o controlador de ordem fracionária obtém uma resposta melhor que o controlador convencional. Podlubny (1999) menciona alguns circuitos elétricos que podem se comportar como integradores e derivadores de ordem fracionária, mas deixa claro que ainda havia muito a ser desenvolvido para a implementação real dos controladores fracionários.

Xue e Chen (2002) apresentam um estudo comparativo entre quatro tipos de controladores de ordem fracionária. Inicialmente, exploram duas propriedades do operador integrador/derivador fracionário:

a) o operador fracionário comuta com o operador de ordem inteira:

$$\frac{d^n}{dt^n}\left({}_aD_t^{\,r} f(t)\right) = {}_aD_t^{\,r}\left(\frac{d^n f(t)}{dt^n}\right) = {}_aD_t^{\,r+n} f(t) \qquad (9)$$

Onde n é qualquer número inteiro

b) o operador fracionário é linear assim como a derivada de ordem inteira:

$${}_aD_t^r(\lambda f(t) + \mu g(t)) = \lambda\, {}_aD_t^r f(t) + \mu\, {}_aD_t^r g(t) \qquad (10)$$

Estas duas propriedades são importantes pois mostram que o operador fracionário é um operador linear e que este é facilmente associado em cadeia com operadores de ordem inteira.

A partir daí, descrevem os controladores de ordem fracionária: TID (*Tilted Proportional and Integral*), CRONE (*Contrôle Robuste d'Ordre Non Entier*), $PI^{\lambda}D^{\mu}$ e

compensador avanço-atraso fracionário. O estudo apresenta um breve histórico de cada controlador, seu esquema de implementação (no domínio do tempo contínuo) e sugere alguns métodos de sintonia para cada controlador, discutindo as vantagens e desvantagens de cada um:

- O controlador CRONE possui implementações práticas bastante relevantes, especialmente no controle de sistemas de suspensão automotiva, e já possui um Toolbox para MATLAB além de possuir métodos de projeto do controlador baseados em interpretação de diagrama de Bode e de Nichols.

- O compensador de avanço-atraso fracionário também deve ter valor já que o compensador de avanço-atraso é um método de projeto de controlador bastante popular, mas ainda é necessário o desenvolvimento de técnicas de projeto e sintonia deste tipo de controlador.

- O TID pode ser interpretado como um caso particular do $PI^{\lambda}D^{\mu}$ mas já existe um método de ajuste dos parâmetros proposto e testado.

- O controlador $PI^{\lambda}D^{\mu}$ é bastante versátil, uma vez que com a possibilidade dos ajustes de λ e μ, o tornam um filtro linear de infinitas dimensões e pode se tornar bastante popular por ser uma extensão do controlador PID convencional, amplamente adotado no setor industrial, porém seria desejável o desenvolvimento de técnicas para estimativa dos cinco parâmetros do controlador e sintonia automática destes.

2.2 Discretização e implementação prática

Do ponto de vista de aplicação prática do cálculo fracionário a sistemas de controle, um ponto chave é a avaliação numérica ou discretização das equações envolvidas. Existem métodos de discretização direta e indireta, onde os métodos indiretos realizam a discretização em mais de uma etapa. Chen e Moore (2002) apresentam um método de discretização recursiva baseado no operador de Tustin, que pode ser implementado diretamente em sistemas digitais, é estável e apresenta fase mínima, porém possui elevados erros quando a faixa de frequências de interesse é grande. Ainda no mesmo artigo, Chen e Moore (2002) apresentam outro método que combina os operadores de Euler e Tustin para produzirem o operador de Al-Alaoui. Este método apresenta erros de magnitude menores para uma mesma faixa de frequência de interesse.

Em relação à discretização direta e indireta, Al-Alaoui (2009) descreve que não há razões para acreditar que uma abordagem é superior à outra em termos de exatidão, e que o desempenho depende das diferentes aproximações que evoluem ao longo do tempo para ambas as abordagens. Al-Alaoui apresenta um estudo comparativo de desempenho de erro absoluto em magnitude entre quatro técnicas atuais para discretização de sistemas fracionários:

- Discretização indireta aproximando a função de transferência resultante de:

$$s_{[\omega_A,\omega_B]}^{-r} \cong \frac{\omega_B(\omega_A + s)}{(\omega_A)^n(rs^2 + \omega_B s + (1-r)\omega_A\omega_B)}\left(\frac{1+\frac{s}{\omega_B}}{1+\frac{s}{\omega_A}}\right)^r \tag{11}$$

  Onde $[\omega_A, \omega_B]$ é a faixa de frequência na qual a função de transferência deve ser aproximada e r a ordem fracionária do operador s.

Realizando a aproximação da função de transferência resultante por:

- transformada bilinear
- transformada de Al-Alaoui.

- Discretização direta para levar a função do domínio s para z, usando
  - transformadas bilinear (Tustin)
  - trasformada de Al-Alaoui

  e então, usando a técnica de mínimos quadrados (*Least Squares*, LS) pelo método de Pade, Prony ou Shanks para aproximar a função de transferência digital a s^r.

No final do artigo, baseado na comparação dos erros absolutos em magnitude de cada um dos métodos, Al-Alaoui conclui que o método de discretização direta pela transformada de Al-Alaoui aplicando a aproximação de mínimos quadrados de Prony é o que implica em menores erros na faixa de frequência de interesse.

Barbosa e Machado (2006) apresentam um roteiro detalhado para implementar controladores de ordem fracionária no tempo discreto baseando-se na aproximação dos mínimos quadrados. Em seu artigo, aborda com detalhes os três passos para a implementação prática de uma função s^r com $r \in \mathbb{R}$: a discretização do operador s^r usando uma função geradora $H^r(z^{-1})$; a obtenção da resposta ao impulso $h^r(k)$ da equivalente discreta fracionária por meio da expansão por série de potências (PSE) de $H^r(z^{-1})$; e finalmente a aplicação das técnicas de modelamento de sinais de Padé, Prony ou Shanks a $h^r(k)$ para obter a aproximação desejada na forma de um filtro IIR. Ainda, os autores apresentam um conjunto de exemplos para ilustrar como determinar bons parâmetros para truncamento de funções nos métodos de aproximação. Com isso, é possível usar estes métodos no desenvolvimento de um programa que implemente o controlador de ordem fracionária.

JIN et al (2009) descrevem que a aplicação de controladores de ordem fracionária tem crescido nos últimos tempos e vem sendo um tema recorrente de pesquisa, porém são predominantemente estudos teóricos ou ilustrações simuladas. Apresentam, então, estudo pioneiro experimental de controladores de ordem fracionária usando o ambiente LabVIEW, da National Instruments. A versatilidade da linguagem, as ferramentas de controle disponíveis e a integração com *hardware* são os principais fatores para justificar o uso da ferramenta. Os autores comparam o desempenho da implementação do controlador de ordem fracionária PD^{μ} contra um controlador PID de ordem inteira num sistema de controle de posição de um motor CC. Para a implementação do controlador de ordem inteira, Jin et al (2009) usam as ferramentas do módulo Control Design & Simulation do LabVIEW. Para a implementação do controlador de ordem fracionária, uma função de transferência no domínio do tempo discreto (z) aproximada é calculada e só então implementada no LabVIEW, já que não existem, ainda, funções que implementem operadores de ordem fracionária neste programa. Os resultados comparativos confirmam que o controlador de ordem fracionária possui uma resposta mais rápida, sem sobressinal e mais robusta a alterações na planta.

Li, Luo e Chen (2010) descrevem que o volume de trabalhos relacionados a aplicação de controladores de ordem fracionária é crescente, e cada vez mais seus benefícios são comprovados, apesar de ainda não existirem métodos para sintonizar os parâmetros destes controladores. Uma forma de simplificar isto é o desenvolvimento de métodos de sintonia dedicados a certos tipos de plantas. Li, Luo e Chen (2010) propõem um método para sintonia de um controlador PD^{μ} de ordem fracionária em aplicações de controle de plantas de segunda ordem. O controlador PD^{μ} de ordem fracionária tem a seguinte forma:

$$C(s) = K_p(1 + K_d s^{\mu}) \quad (12)$$

Onde C(s) é a função de transferência do controlador, K_p é o ganho proporcional K_d é o ganho derivativo e μ é a ordem do operador derivativo.

Dessa forma, os autores apresentam um experimento prático aplicando este método na sintonia do controlador numa aplicação de controle de motor CC numa célula dinamométrica comprovando a eficiência do controlador e do método de sintonia.

3 MATERIAIS E MÉTODOS

Neste capítulo são apresentadas as ferramentas que serão usadas no experimento prático de comparação de desempenho entre o controlador PID convencional e o controlador $PI^{\lambda}D^{\mu}$. Para isto, será usada uma planta experimental a ser controlada, um controlador de tempo real e o ambiente de programação gráfica LabVIEW da National Instruments.

3.1 Ambiente de programação gráfica LabVIEW

Por conta de vários fatores como a possibilidade de uso do mesmo programa em diferentes plataformas de execução, desde simulação no PC à execução em tempo real em plataforma dedicada e algoritmos de cálculo já disponíveis foi escolhido como ambiente de desenvolvimento o LabVIEW da National Instruments.

O LabVIEW é um ambiente gráfico de desenvolvimento de aplicações. A linguagem gráfica também é conhecida como G. Foi lançada em 1986 e é amplamente usada em sistemas de teste automatizado, aquisição de dados e sistemas embarcados.

O programa desenvolvido em LabVIEW é chamados VI, que é a extensão do nome de arquivo e significa instrumento virtual (na sigla em inglês). A programação

em LabVIEW é composta por três elementos: painel frontal, diagrama de blocos e ícone e painel de conexões.

O painel frontal é a interface de operação do VI. Nele, o programador distribui controles e indicadores gráficos que serão usados pelo usuário para operar o programa. Um exemplo é apresentado na figura 2.

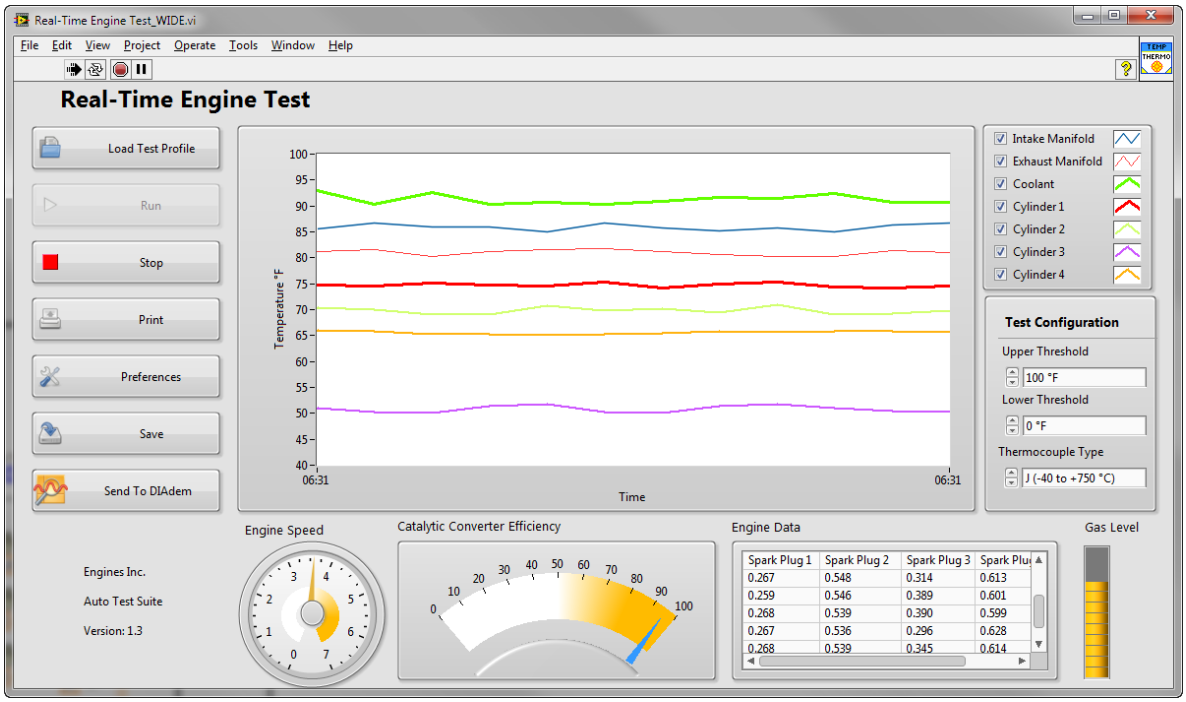

Figura 2 – Painel Frontal do LabVIEW

O diagrama de blocos é a janela onde a lógica de programação é estruturada. O LabVIEW possui uma biblioteca de funções com centenas de funções prontas para dar produtividade ao programador. A figura 3 apresenta um exemplo de diagrama de blocos.

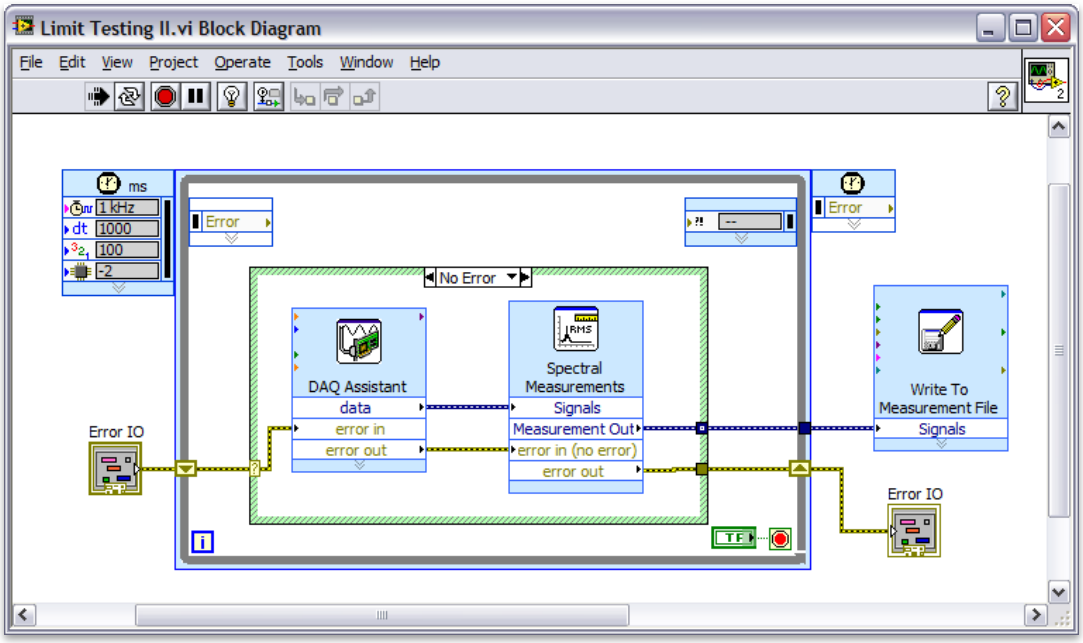

Figura 3 – Diagrama de blocos do LabVIEW

O ícone e painel de conexões são usados quando o VI será usado como sub-rotina de um VI maior, chamado no LabVIEW de SubVI. O ícone é configurado de forma a representar graficamente o SubVI no VI principal e o painel de conexões é a interface entre o subVI e o VI principal. É por meio dele que o VI principal passa parâmetros para o subVI e recebe os resultados após a execução deste. A figura 4 apresenta um exemplo do ícone e painel de conexões.

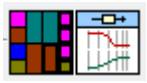

Figura 4 – Exemplo de ícone (direita) e painel de conexões (esquerda)

3.2 Controlador de tempo real

Com o objetivo de minimizar a possibilidade de ter resultados distorcidos, que poderiam ser causados por atrasos ou por jitter no controlador, é utilizada uma plataforma de tempo real para realizar o controlador.

A plataforma escolhida para implementar o controlador foi o CompactRIO da National Instruments, figura 5.

Figura 5 – Controlador CompactRIO da National Instruments

O CompactRIO da National Instruments é um controlador de tempo real que possui arquitetura de aplicação com entrada e saída de sinais ponto a ponto, o que favorece muito a aplicação em sistemas de controle em malha fechada. Além disso toda a programação implementada no LabVIEW para projeto de controlador e simulação de sistemas dinâmicos pode ser embarcada no CompactRIO de forma direta, tornando a plataforma de implementação transparente, ou seja, não há necessidade de compatibilizar a programação para execução em um PC Desktop executando sistema operacional Windows ou para execução em um controlador

CompactRIO executando sistema operacional de tempo real. O mesmo VI pode ser executado em ambos os “targets” de implementação.

A arquitetura de *hardware* do controlador é apresentada na figura 6. O controlador possui uma CPU que executa sistema operacional de tempo real. Dependendo do modelo do CompactRIO o sistema operacional é WindRiver VXWorks, LinuxRT ou ETS Pharlap (o modelo utilizado neste trabalho, cRIO-9022 executa WindRiver VXWorks). O módulo LabVIEW Real-Time permite que o sistema operacional utilizado não influencie no desenvolvimento da aplicação de controle, o LabVIEW torna transparente esta camada da aplicação. Esta CPU é ligada a um FPGA configurável, por meio do LabVIEW FPGA, que acessa cada um dos canais de entrada ou saída de sinais. Ainda, tais canais são disponibilizados por meio de módulos, da chamada Série C, que possuem diferentes condicionamentos de sinais para as mais diversas aplicações e conectividade direta com sensores ou atuadores. Existem mais de 50 opções de módulos para diversos sinais.

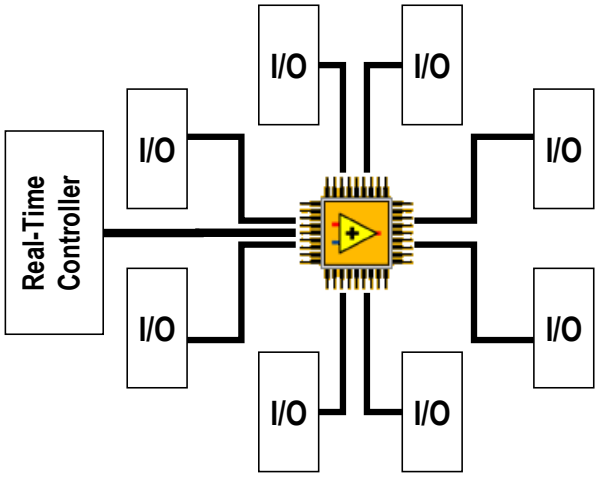

Figura 6 – Arquitetura de *hardware* do CompactRIO

Neste projeto, os algoritmos de controle serão executados na CPU de tempo real. O FPGA será usado apenas para condicionamento dos sinais e cálculo de escala para conversão de unidades de medição dos sensores. É um recurso com

alta capacidade de processamento de sinais que atualmente permite inclusive execução de matemática em ponto flutuante, altíssima confiabilidade e determinismo, mas ainda não estão disponíveis os solvers usados pelo módulo de projeto de controladores e simulação de sistemas.

3.3 Planta experimental a ser controlada

Dado o objetivo deste trabalho, foi escolhida uma planta de baixa complexidade para o desenvolvimento de um sistema de controle de posição de um motor CC. Especificamente, será utilizada a planta QET DC Motor Control da Quanser, que é uma planta didática voltada ao ensino de teoria de controle (Figura 7).

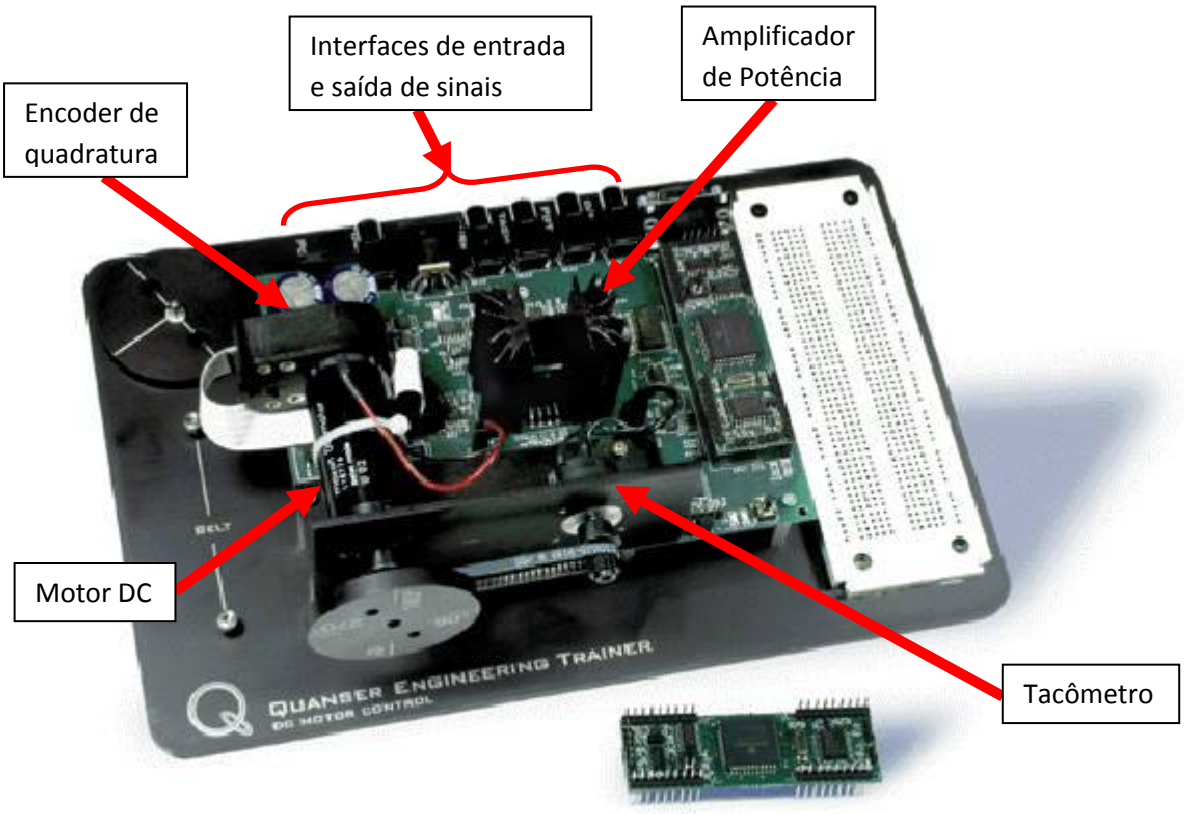

Figura 7 – Quanser QET DC Motor Control

Tal planta é um sistema composto por um motor CC com tacômetro analógico e encoder de quadratura acoplados e amplificador de potência linear para atuação no motor.

Como os dados de especificações do motor são disponibilizados pelo fabricante da planta, a modelagem do motor é possível além da possibilidade de levantamento do modelo por algoritmo de identificação de sistemas.

4 PROJETO DO CONTROLADOR

Este capítulo detalha o desenvolvimento do controlador fracionário $PI^{\lambda}D^{\mu}$ por meio do método de discretização indireta no ambiente LabVIEW da National Instruments de forma genérica, implementando a função de transferência aproximada que realiza s^r, estruturando o controlador fracionário a partir desta função de transferência e usando o módulo Control Design and Simulation do LabVIEW para a solução numérica e execução em ambiente de simulação e em tempo real, possibilitando a aplicação do controlador de forma ampla.

Neste ponto, o trabalho de desenvolvimento do controlador pode ser dividido em duas etapas, o programa que realiza o operador de ordem fracionária s^r e a integração deste num programa que implemente a função de transferência do controlador $PI^{\lambda}D^{\mu}$.

4.1 Desenvolvimento do programa que resolve s^r

Para a implementação deste programa, foram adotados os métodos propostos por Barbosa e Machado (2006), Chen e Moore (2002) e Al-Alaoui (2009). Conforme apresentado nestes trabalhos, um dos métodos mais eficientes para a

implementação do operador s^r para r não inteiro é a discretização direta de s^r, usando-se a transformada de Al-Alaoui.

Segundo Al-Alaoui (2009), a discretização direta substitui s^r com uma transformada direta do domínio s para o domínio z (neste caso, a transformação de Al-Alaoui). Isto resulta numa expressão não-racional em z e esta expressão é, então, aproximada para uma expressão racional em z por meio de mínimos quadrados pelo método de Prony para, finalmente, obter a função de transferência racional no domínio z que se aproxime de s^r. Este método não convergiu para um resultado satisfatório e os esforços de desenvolvimento são apresentados no anexo 1.

De maneira alternativa, o programa foi desenvolvido com base no método de discretização indireta. Segundo Al-Alaoui (2009), a discretização indireta de sistemas fracionários segue o seguinte procedimento: é feita uma aproximação de s^r com uma expressão racional de transformada em s e então é aplicada uma transformada do domínio s para o domínio z para, assim, obter uma expressão racional em z que se aproxime da expressão original, s^r. Como o LabVIEW possui ambiente de simulação de sistemas e consegue operar com modelos no domínio do tempo contínuo, não será necessário realizar a etapa de transformação do domínio s para o domínio z.

O processo para realização da aproximação analógica a s^r é o seguinte:

- determinar a faixa de frequência de interesse na qual o modelo aproximado vai trabalhar;

- realizar uma aproximação analógica de ordem inteira da função de transferência de ordem fracionária;

- discretizar a aproximação analógica.

Al-Alaoui (2009) apresenta a aproximação a partir da equação 11, descrita anteriomente. Ela é expandida por expansão de frações contínua (*continued fraction expansion*, CFE), expansão de Taylor ou o método recursivo de pólos e zeros para obter uma aproximação analógica. Este último é usado neste trabalho e descrito a seguir:

$$\left(\frac{1+\frac{s}{\frac{d}{b}\omega_A}}{1+\frac{s}{\frac{b}{d}\omega_B}}\right)^r = \lim_{N\to\infty}\prod_{k=-N}^{k=N}\frac{1+\frac{s}{\omega'_k}}{1+\frac{s}{\omega_k}} \tag{13}$$

$$\omega'_k = \left(\frac{d}{b}\omega_A\right)^{\frac{r-2k}{2N+1}} \tag{14}$$

$$\omega_k = \left(\frac{b}{d}\omega_B\right)^{\frac{r+2k}{2N+1}} \tag{15}$$

$$s^r \approx K\left(\frac{ds^2+bs\omega_B}{d(1-r)s^2+bs\omega_B+dr}\right)\prod_{k=-N}^{k=N}\frac{s+\omega'_k}{s+\omega_k} \tag{16}$$

$$K = \left(\frac{d}{b}\omega_A\right)^r\prod_{k=-N}^{k=N}\frac{\omega_k}{\omega'_k} \tag{17}$$

Onde b, d e N são parâmetros da aproximação.

Em resumo, a sequência da aproximação é a seguinte:

- Determina-se a faixa de frequência $[\omega_A, \omega_B]$ e N;

- Baseando-se na ordem fracionária r, calcula-se ω'_k e ω_k de acordo com as equações 14 e 15;

- Calcula-se K a partir da equação 17;

- Obtem-se a função de transferência racional aproximada por meio da equação 16 para representar s^r.

Esta foi a sequência implementada na programação em LabVIEW. As figuras 8 e 9 apresentam respectivamente os diagramas de blocos dos programas que calculam ω'_k e ω_k.

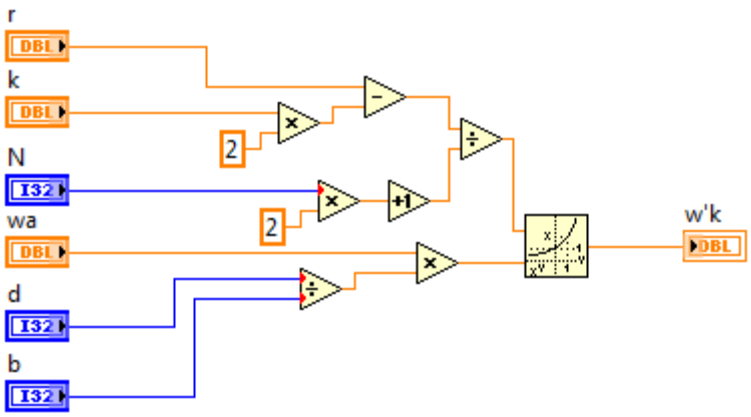

Figura 8 – Diagrama de blocos do programa que calcula ω'_k

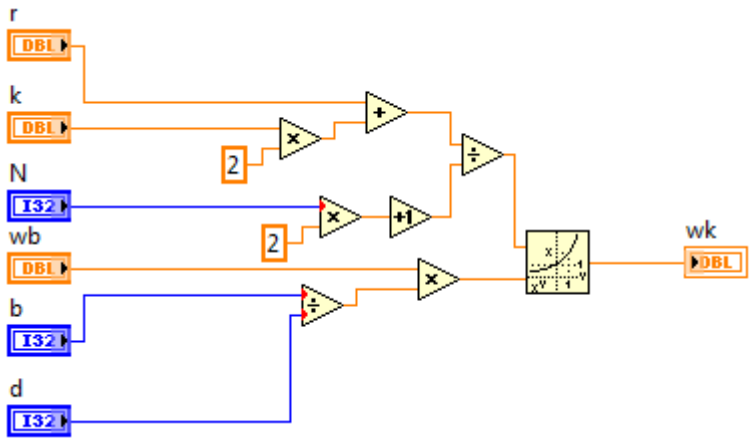

Figura 9 – Diagrama de blocos do programa que calcula ω_k

Utilizando-se estes dois programas como SubVIs, foi elaborado o programa que calcula K (figura 10).

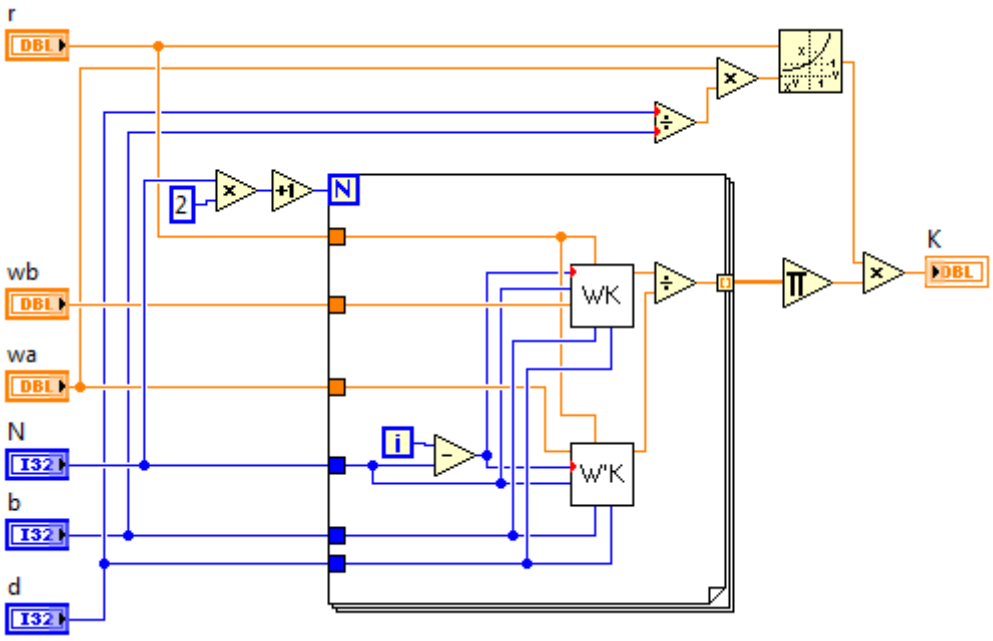

Figura 10 – Diagrama de blocos do programa que calcula K

Estes três programas são usados como subVIs no programa que implementa a equação 16 e calcula s^r (Figura 11).

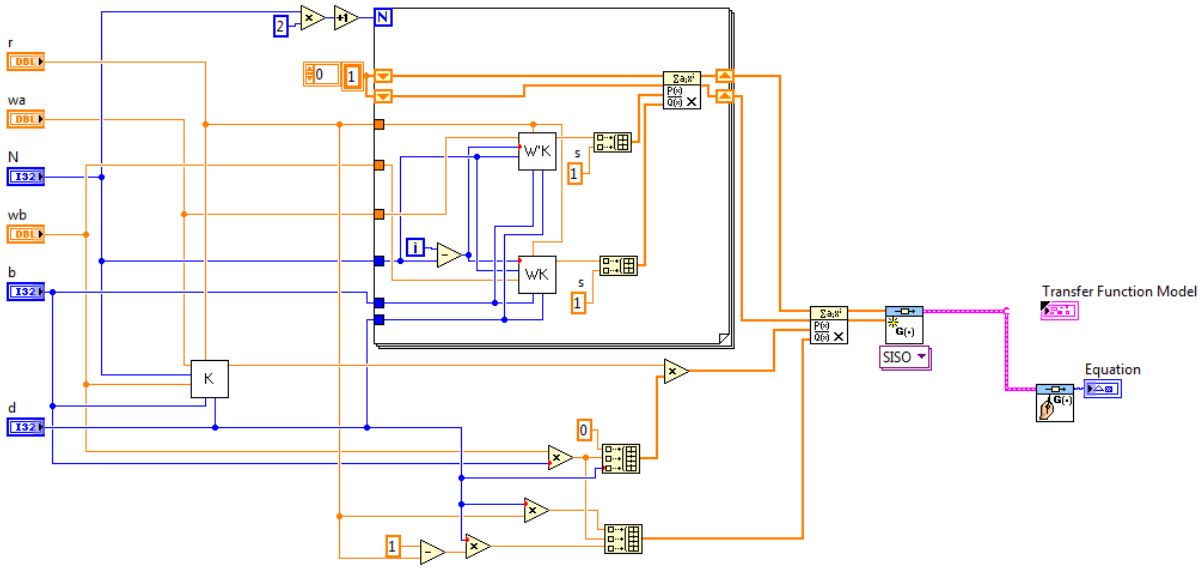

Figura 11 – Diagrama de blocos do programa que calcula s^r

Este programa foi executado com algumas configurações de seus parâmetros para avaliar seu desempenho. Nesta análise, foram observados o ganho e a fase do

sistema ao longo da faixa de frequência selecionada em ω_A e ω_B. Inicialmente, foram usados os mesmos parâmetros que Al-Alaoui (2009):

b = 10

d = 9

N = 3

ω_A = 0,01Hz

ω_B = 100Hz

Com esta configuração, é possível observar que na faixa de frequência de interesse, os resultados foram muito interessantes para valores pequenos de |r| e se afasta dos resultados teóricos para valores maiores de |r|. A figura 12 mostra a visualização da resposta em frequência de s^r para valores inteiros de -3 ≤ r ≤ 3.

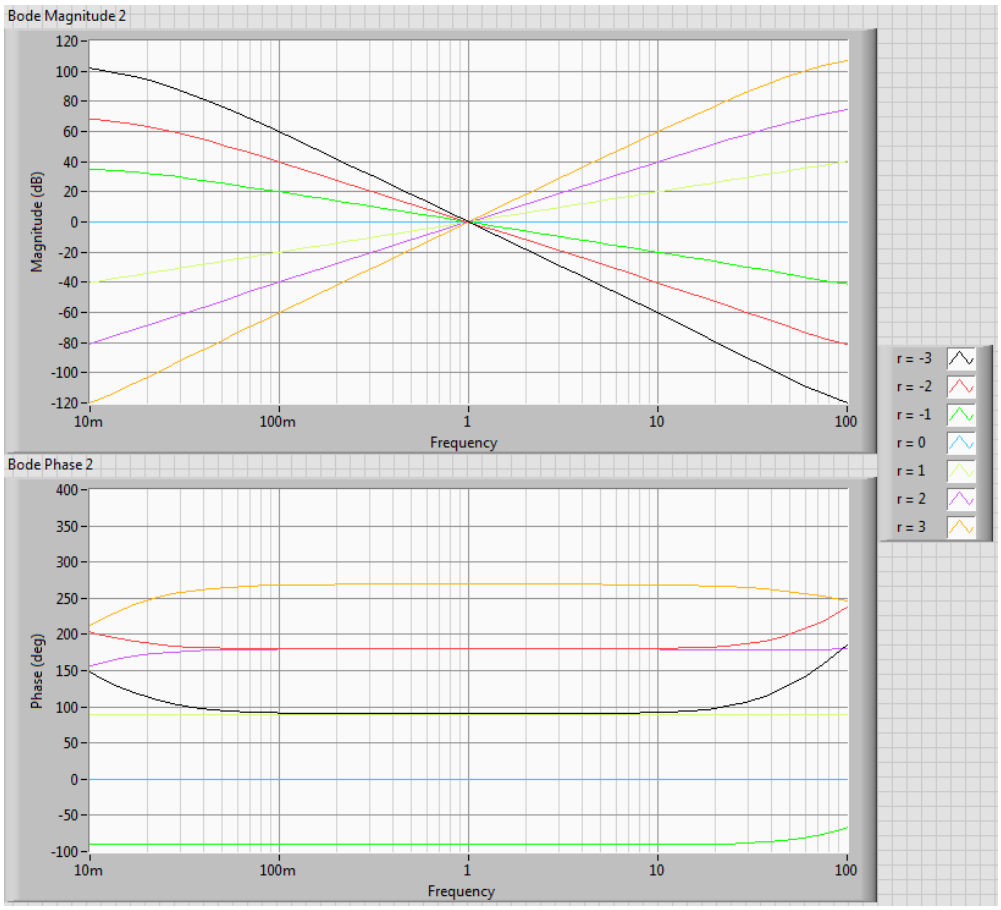

Figura 12 – Resposta em frequência de s^r para -3 ≤ r ≤ 3

Por conta desta observação, foi definido o uso deste programa apenas na faixa de -1 < r < 1. Dessa forma este programa será usado sempre nas condições em que a aproximação retorna os melhores resultados. Para aplicações com |r| > 1, foi usada a propriedade da multiplicação de potências de mesma base e usar o programa desenvolvido neste trabalho para implementar a s^r para a parte fracionária de r e os programas disponíveis na biblioteca do LabVIEW para implementar s^r para a parte inteira de r. Os modelos resultantes destes dois programas são associados em série (multiplicados) para que tenhamos o modelo de s^r para qualquer valor de r real.

Em seguida, foram realizadas análises para valores não inteiros de r. Neste caso, variando o valor de r de 0 a 1 e de 0 a -1, foi possível observar que os valores intermediários retornam uma resposta em frequência também intermediária entre as respostas dos valores inteiros. A figuras 13 e 14 mostram esta observação.

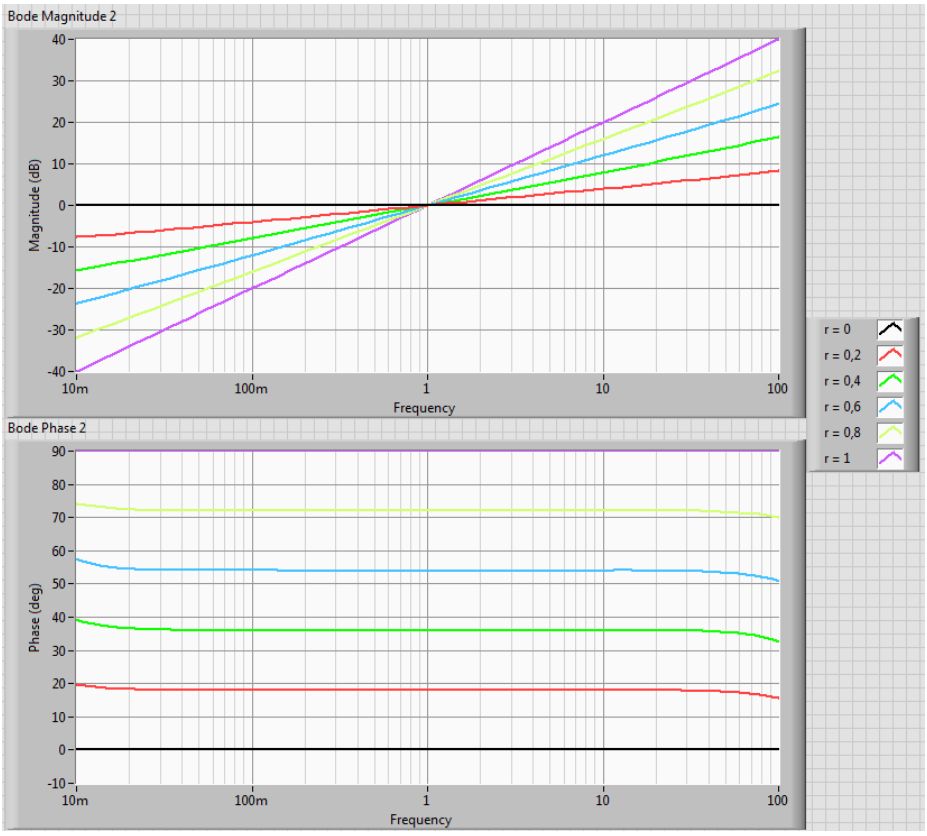

Figura 13 – Resposta em frequência de s^r para $0 \leq r \leq 1$

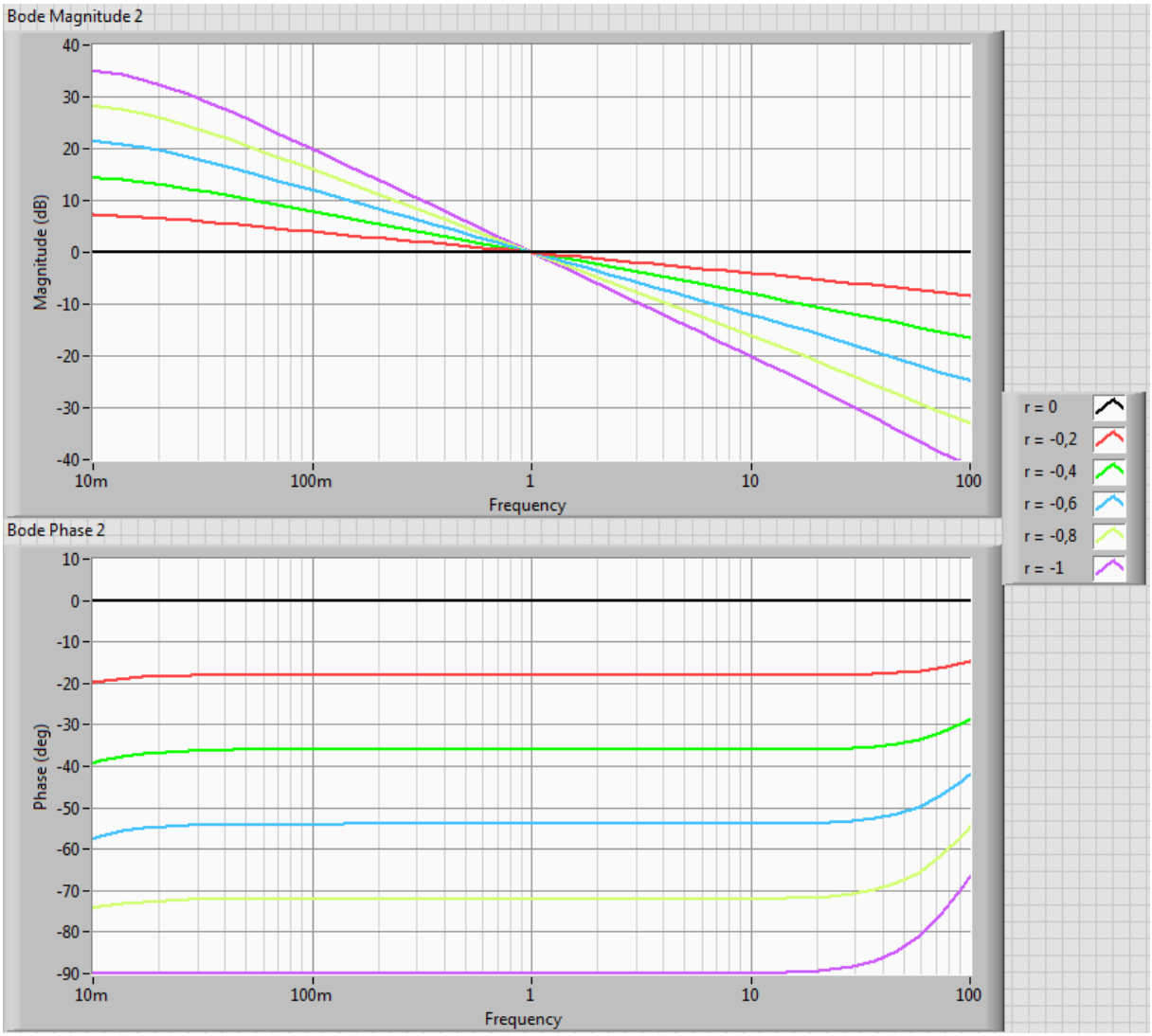

Figura 14 – Resposta em frequência de s^r para -1 ≤ r ≤ 0

As observações seguintes foram feitas alterando-se os parâmetros da técnica de aproximação: primeiro a faixa de frequência (ω_A e ω_B) e em seguida os parâmetros N, b e d.

Aplicando-se valores diferentes de ω_A e ω_B, foi possível observar que o método não se comporta adequadamente para outros valores de frequência e que estes valores de ω_A = 0,01Hz e ω_B = 100Hz, não podem ser alterados arbitrariamente ou então o resultado do operador será comprometido. Durante o desenvolvimento deste trabalho foi observado, de forma empírica que há algumas configurações dos

valores de frequência que retornam resultados satisfatórios. Estas configurações são as que obedecem a seguinte condição:

$$\omega_A = \frac{1}{\omega_B} \tag{18}$$

Para quaisquer outros valores de ω_A e ω_B o resultado da aproximação não retornará um valor razoável para s^r. É possível observar uma deformação na resposta em frequência para valores que não atendam à condição acima. A figura 15 mostra a mesma observação feita anteriormente, para -1 ≤ r ≤ 0 alterando ω_A para 0,1Hz e mantendo ω_B = 100Hz.

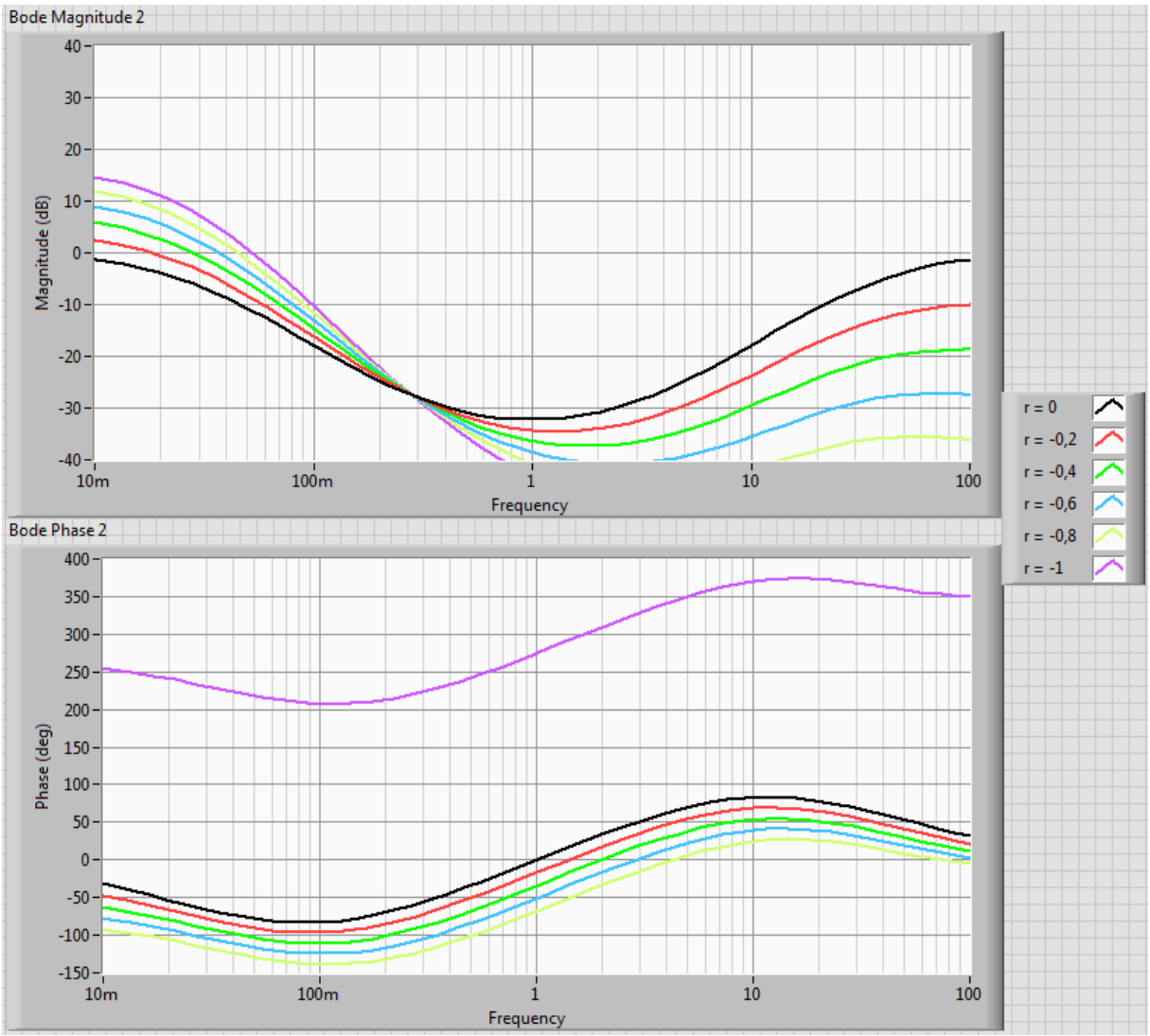

Figura 15 – Deformação na resposta em frequência de s^r por $\omega_A \neq \frac{1}{\omega_B}$

Em seguida, foram analisados os comportamentos inerentes a mudanças dos parâmetros b, d e N.

Os valores usados inicialmente nestes parâmetros foram b = 10, d = 9 e N = 3, como Al-Alaoui (2009).

Foi observado que os ajustes de b e d podem ser usados como ajustes finos da faixa de frequência. Foi observado ainda que para valores de b > d, a faixa de frequência onde a aproximação é válida vai aumentando conforme a diferença entre b e d aumenta e é obsevado comportamento inverso, reduzindo-se a faixa de frequência onde a aproximação é válida para valores de b < d.

O parâmetro N determina a ordem da função de transferência resultante, que se aproxima de s^r. Foi observado que para faixas de frequência maiores que 4 décadas (ω_A < 0,01Hz e ω_B > 100Hz) o valor de N = 3 retorna uma função de transferência cujo polinômio possui uma ordem muito baixa para realizar a aproximação na faixa de frequência desejada e a resposta em frequência que deveria ser plana, apresenta variação. Dessa forma, caso haja interesse em resposta do sistema numa faixa de frequência maior, N pode ser aumentado para garantir a resposta adequada do sistema. Cabe ressaltar que aumentando o valor de N, o grau do denominador e do numerador da função de transferência crescem, adicionando complexidade ao sistema, o que pode comprometer desempenho e inserir maior erro na discretização para aplicações reais. Para N = 0, o grau do denominador e numerador da função de transferência é 3 e para cada incremento de N o grau aumenta 2 ordens, por exemplo, para N = 1, grau 5, para N = 2, grau 7, para N = 3, grau 9.

Tendo desenvolvido o operador fracionário é possível prosseguir na implementação do controlador $PI^{\lambda}D^{\mu}$.

4.2 Desenvolvimento do Controlador PI$^{\lambda}$D$^{\mu}$

Conforme apresentado por Podlubny, Dorkak e Kostial (1997), o controlador PI$^{\lambda}$D$^{\mu}$ possui a forma apresentada na equação (1):

$$G_c(s) = \frac{U(s)}{E(s)} = K_P + K_I s^{-\lambda} + K_D s^{\mu} \quad (1)$$

$$(\lambda, \mu > 0)$$

Para implementar um programa que gere o controlador, usaremos o programa da Figura 11 como função. Para usá-lo na condição descrita acima na equação (18) e não ter a limitação de ordem menor que |1|, faremos a seguinte adaptação à expressão do controlador PI$^{\lambda}$D$^{\mu}$:

$$G_c(s) = \frac{U(s)}{E(s)} = K_P + K_I s^{-\lambda_a} s^{-\lambda_b} + K_D s^{\mu_a} s^{\mu_b} \quad (19)$$

Onde:

λ_a é a parte inteira de λ

λ_b é a parte não inteira de λ

μ_a é a parte inteira de μ

μ_b é a parte não inteira de μ

Os operadores de ordem inteira poderão ser implementados usando as funções do módulo Control Design & Simulation e apenas os operadores fracionários serão implementados por meio do algoritmo proposto e os operadores de ordem fracionária serão calculados por meio do programa da figura 11 garantindo que $-1 < r < 1$.

O diagrama de blocos do controlador PI$^{\lambda}$D$^{\mu}$ é apresentado na figura 16. O programa que o executará é preparado para receber os 5 parâmetros.

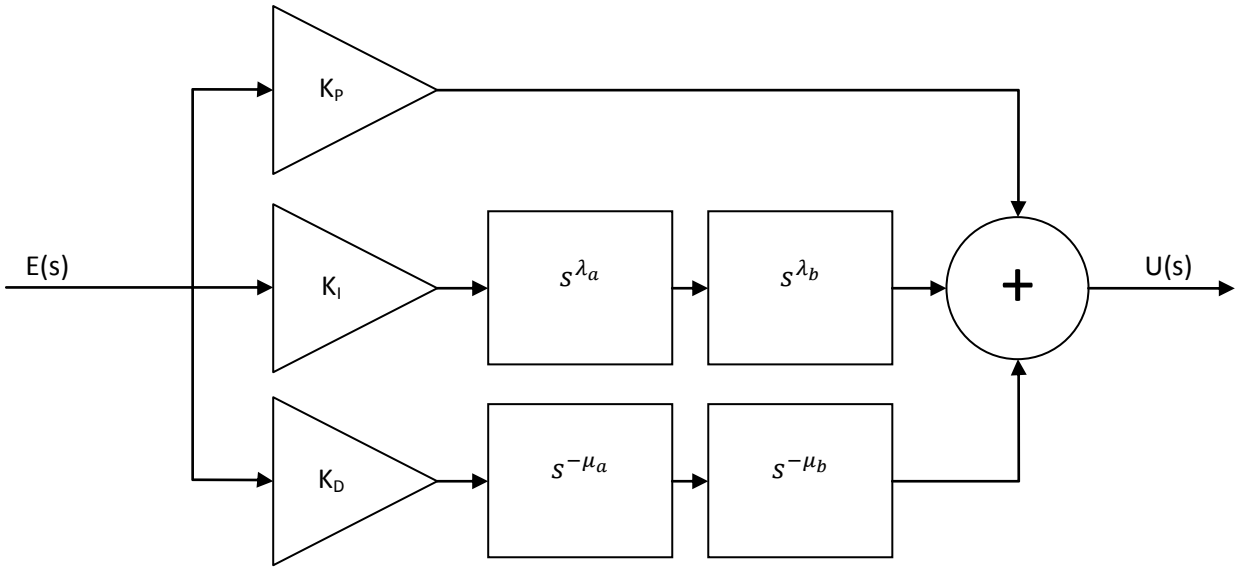

Figura 16 – Diagrama de blocos do controlador $PI^{\lambda}D^{\mu}$

Dessa forma, obtém-se o controlador PID de ordem fracionária que é usado na comparação com o controlador PID convencional.

5 EXPERIMENTOS E RESULTADOS

Este capítulo apresenta os experimentos realizados com o controlador $PI^{\lambda}D^{\mu}$ e os resultados observados. Foram executados experimentos de sistema de controle de posição de motor CC com modelos teóricos em *software*, acrescentando-se não-linearidades do mundo real em ambiente de simulação em *software* e experimentos reais com a planta real e o controlador de tempo real.

5.1 Experimento teórico em *software*

O objetivo destes experimentos foi observar se o controlador $PI^{\lambda}D^{\mu}$ apresentava resultados coerentes, validando a implementação, e observar o comportamento da resposta do sistema conforme os parâmetros do controlador eram alterados. Neste ponto, como trata-se de experimento inicial, foi importante isolar ao máximo quaisquer variáveis que pudessem influenciar os resultados. A condição de contorno para este experimento foi: implementar o experimento da forma mais próxima possível de um ambiente ideal, sem limitações e não-linearidades observadas em experimentos reais.

Para atender a esta condição de contorno, o experimento foi realizado com o modelo teórico da planta, levantado com base nos dados fornecidos pelo fabricante

e a malha de controle foi implementada com as funções de projeto de sistemas de controle do LabVIEW, a biblioteca *Control Design*.

O modelo da planta para posição calculado a partir dos dados técnicos fornecidos pelo fabricante é o seguinte:

$$\frac{\theta(s)}{V(s)} = \frac{19{,}1}{0{,}103s^2 + s} \tag{20}$$

Onde θ é a posição angular do eixo do motor em radianos e V é a tensão aplicada

Usando-se a biblioteca *Control Desgin*, é possível implementar a malha de controle e levantar-se as respostas do sistema.

Inicialmente, foram implementadas duas malhas de controle, uma com um controlador PID convencional, usando-se o bloco *Create PID* da biblioteca do LabVIEW *Control Design* conectado em série com o modelo da planta e com realimentação unitária e outra malha usando-se o controlador $PI^{\lambda}D^{\mu}$. As respostas dos dois sistemas foram plotadas num único gráfico para facilitar a comparação.

O primeiro objetivo foi validar a implementação do controlador. Dessa forma, foi adotado o valor de 0,999 para os parâmetros λ e μ para observar quão próximo os resultados são aos de um PID convencional (não foi adotado o valor 1 pois neste caso, o bloco s^r não é usado pelo programa). Para parâmetros K_p, K_i e K_d foram aplicados os mesmos valores aos dois controladores. Na figura 17 é possível observar a resposta dos dois sistemas de controle com $K_p = 1$, $K_i = 0{,}1$ e $K_d = 0{,}08$.

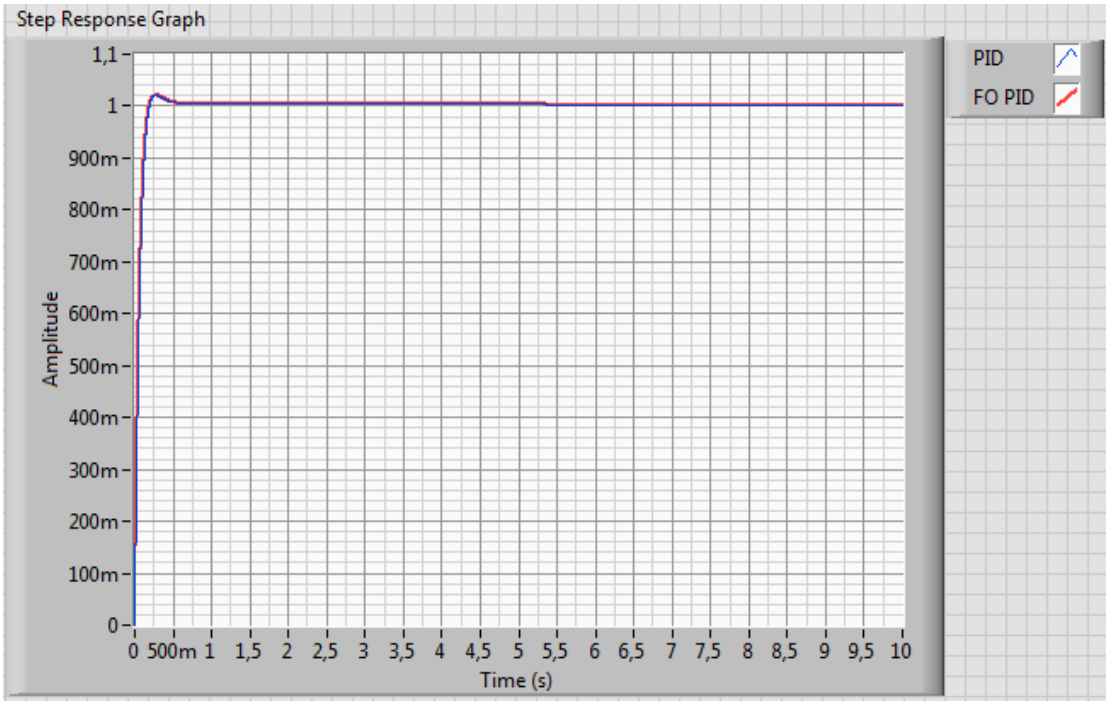

Figura 17 – Resposta ao degrau ideal dos sistemas com PID e $PI^{\lambda}D^{\mu}$

Os resultados foram muito próximos, não sendo possível observar diferença nas respostas. Para melhorar esta análise, foi calculada a diferença ponto a ponto entre as duas respostas. A Figura 18 mostra o gráfico desta diferença.

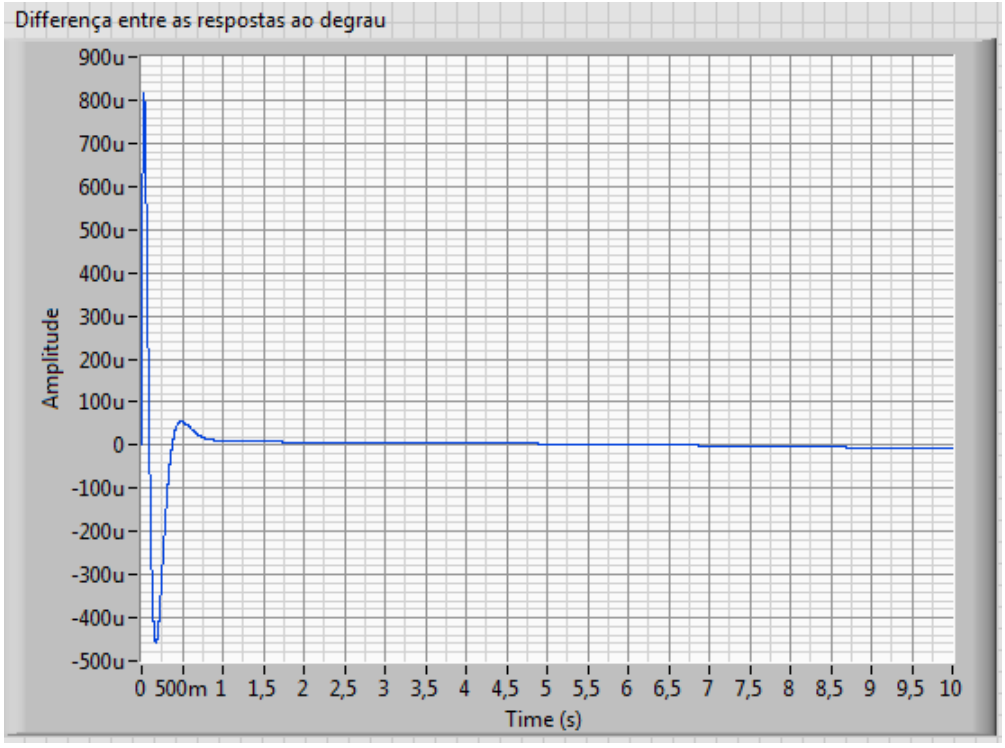

Figura 18 – Diferença entre a reposta ideal do sistema com PID e com $PI^{\lambda}D^{\mu}$

Com este gráfico, é possível mensurar a amplitude da diferença entre as duas respostas. É possível observar que a maior diferença aparece durante o transitório do sistema, ou seja, nos primeiros 500ms. Ainda assim, a diferença não ultrapassa 0,001 para a resposta ao degrau unitário. Outro ponto importante é que é esperado que as respostas sejam próximas mas não necessariamente iguais, já que as ordens de integração e derivação são levemente diferentes ($1 \neq 0{,}999$).

Estes resultados reforçam a validade do algoritmo e da implementação em soma às observações descritas no capítulo 4.

A partir daí, o próximo passo foi variar os parâmetros λ e μ e observar a resposta a fim de analisar as alterações e traçar o comportamento imposto por estes parâmetros.

Para verificar esta situação, foi construído um gráfico com as respostas do sistema com PID para diferentes valores de K_d (Figura 19) e outro com as respostas do sistema com $PI^{\lambda}D^{\mu}$ para diferentes valores de μ (Figura 20).

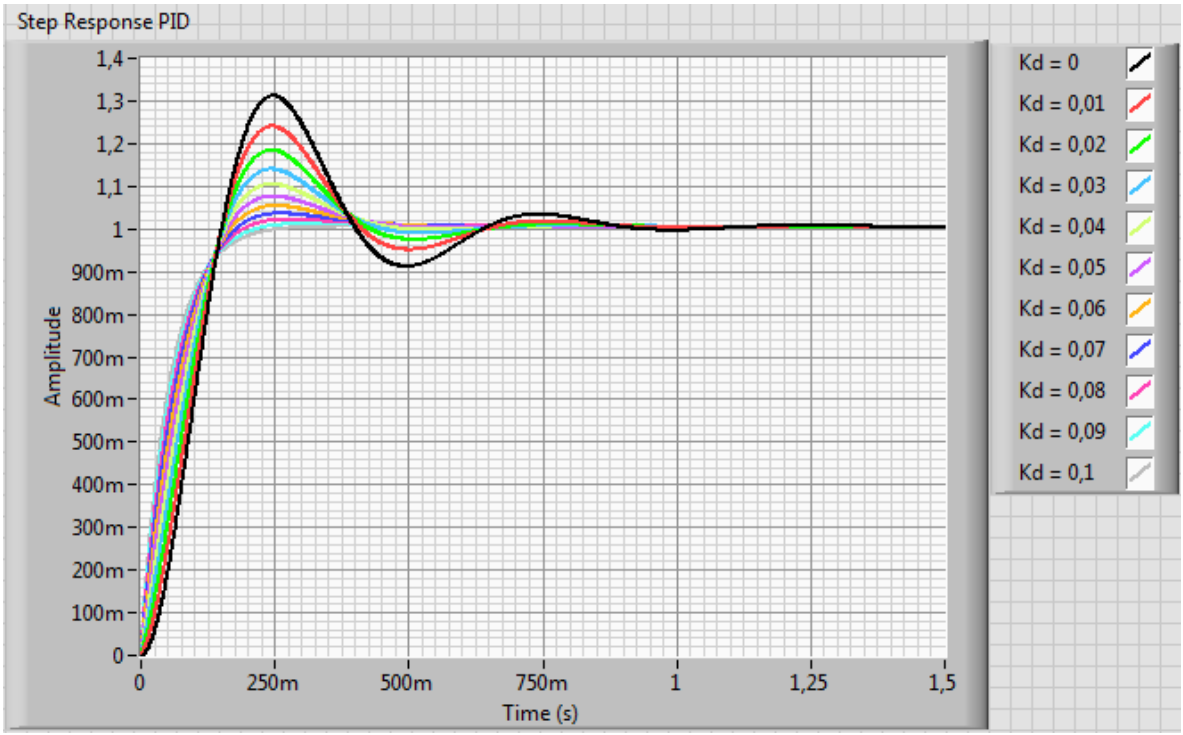

Figura 19 – Respostas ao degrau do sistema com PID para diferentes valores de K_d

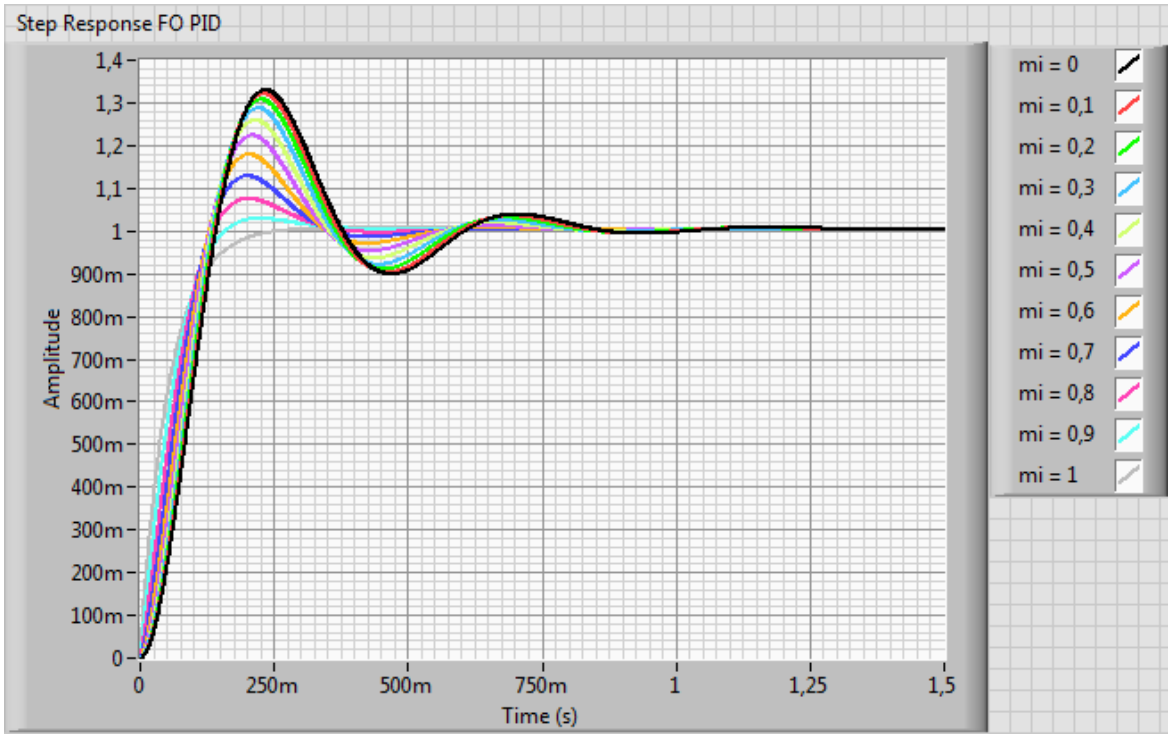

Figura 20 – Respostas ao degrau do sistema com $PI^{\lambda}D^{\mu}$ para diferentes valores de μ

Este experimento foi realizado da seguinte forma:

Para o primeiro gráfico, foi usado o PID convencional em malha fechada com o modelo da planta. Foram mantidos $K_p = 1$ e $K_i = 0,1$ enquanto K_d foi alterado de 0 a 0,1 em passos de 0,01.

Para o segundo gráfico, foi usado o $PI^{\lambda}D^{\mu}$ em malha fechada com o modelo da planta. Foram mantidos $K_p = 1$, $K_i = 0,1$, $\lambda = 0,999$ e $K_d = 0,1$ enquanto μ foi alterado de 0 a 1 em passos de 0,1.

Numa primeira análise, é possível observar que ambos aumentam o amortecimento da resposta para valores maiores de K_d ou μ. Isto leva a uma observação preliminar que K_d e μ podem ser usados juntos para obter-se um ajuste mais fino da atuação derivativa. Mas observando com mais detalhe as respostas, aplicando-se uma ampliação no eixo de tempo, é possível observar certa diferença na variação das respostas (Figuras 21 e 22).

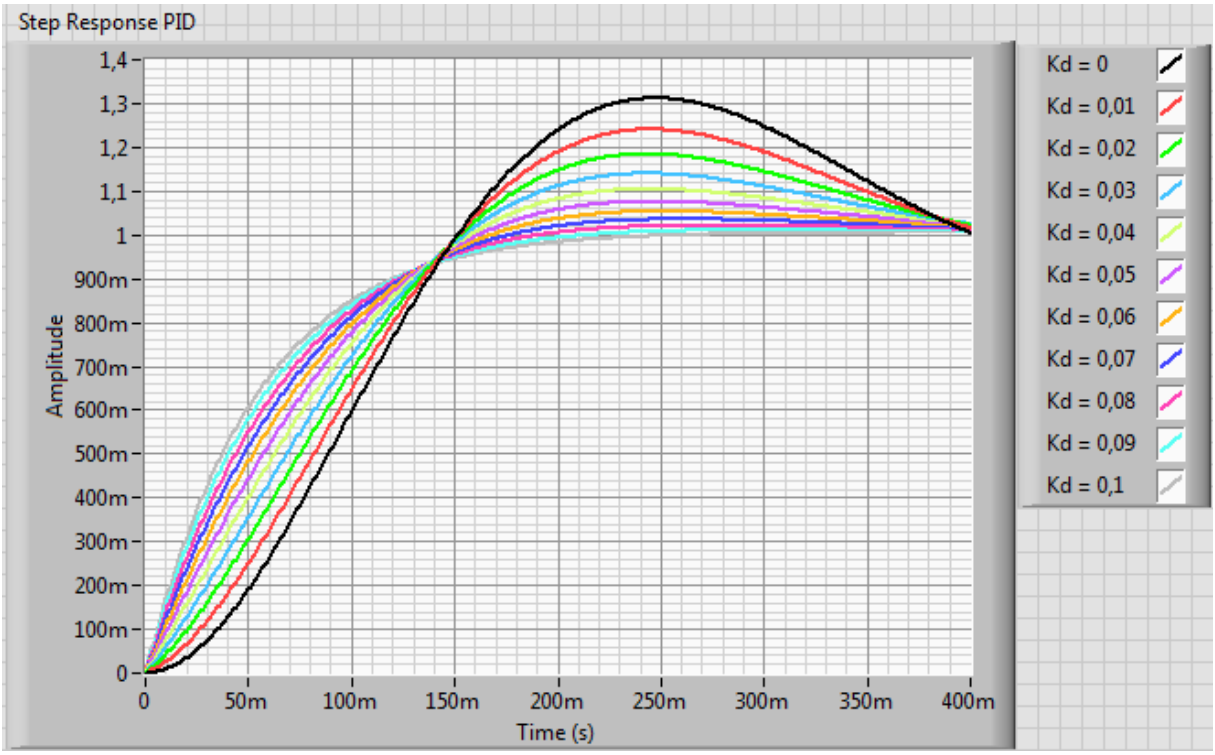

Figura 21 – Ampliação nas respostas ao degrau do sistema com PID para diferentes valores de K_d

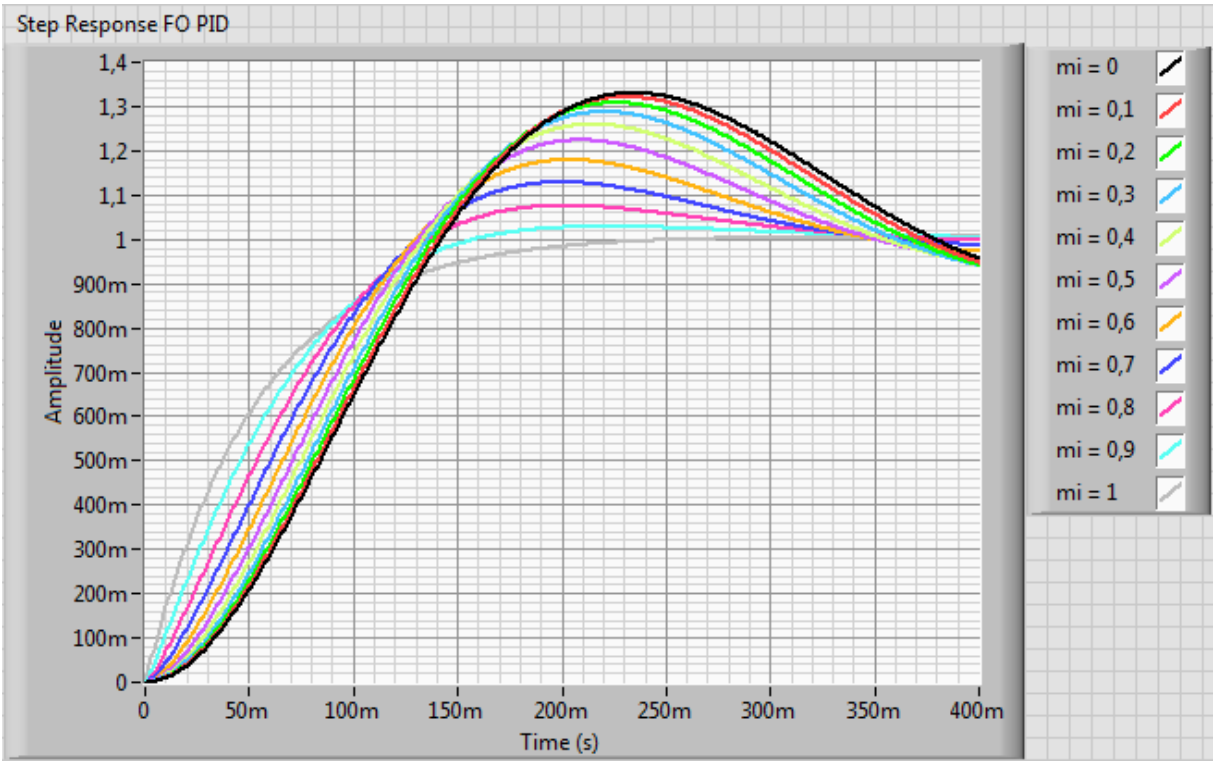

Figura 22 – Ampliação nas respostas ao degrau do sistema com $PI^{\lambda}D^{\mu}$ para diferentes valores de μ

Realizando esta ampliação é possível observar que a variação não é exatamente igual. A variação de K_d no primeiro gráfico implica numa variação do amortecimento do sistema, mas o pico de sobresinal ocorre sempre no mesmo instante de tempo, independente do valor de K_d. A variação de μ no segundo gráfico também implica numa variação do amortecimento do sistema, mas o pico de sobresinal é adiantado conforme o valor de μ é aumentado. Com isso, é possível observar que a transição de subida da resposta ao degrau sofre menos variação com alterações no valor de μ do que com alterações no valor de K_d.

O mesmo experimento é feito para observar o comportamento dos parâmetros integradores. Da mesma forma, foi construído um gráfico com as respostas do sistema com PID para diferentes valores de Ki (Figura 23) e outro com as respostas do sistema com $PI^{\lambda}D^{\mu}$ para diferentes valores de λ (Figura 24).

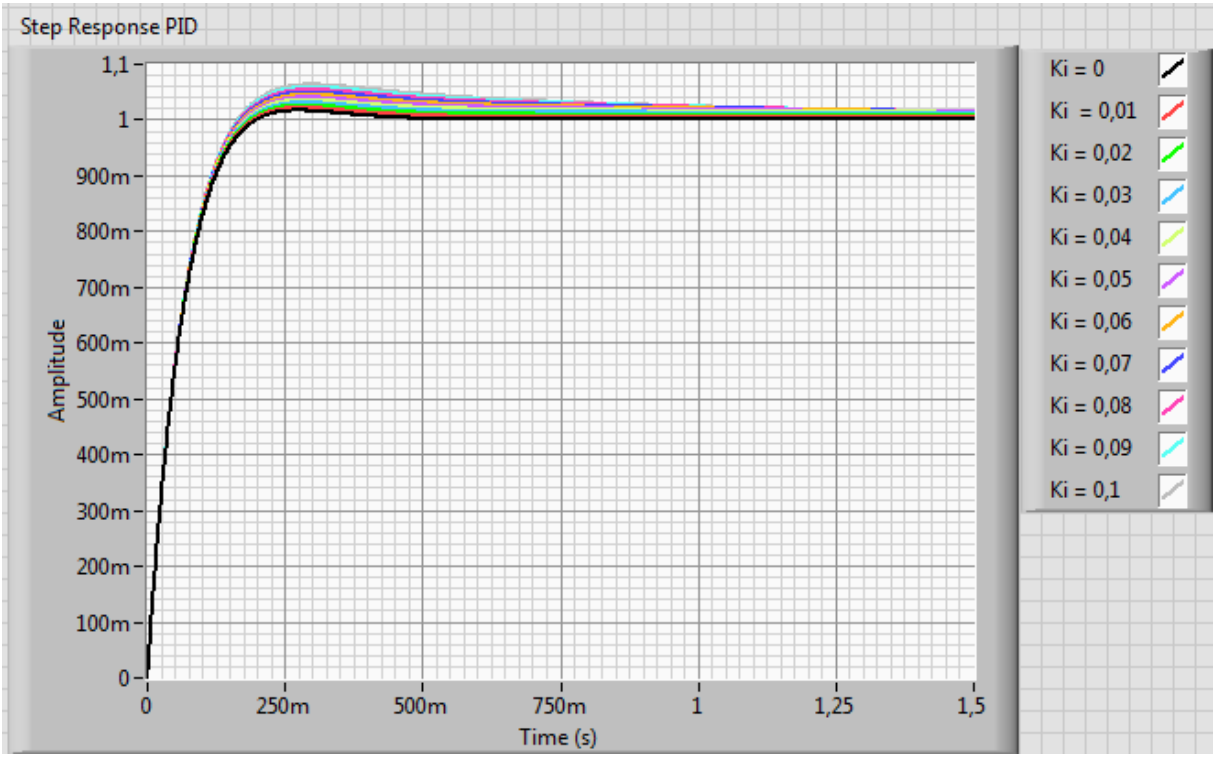

Figura 23 – Respostas ao degrau do sistema com PID para diferentes valores de Ki

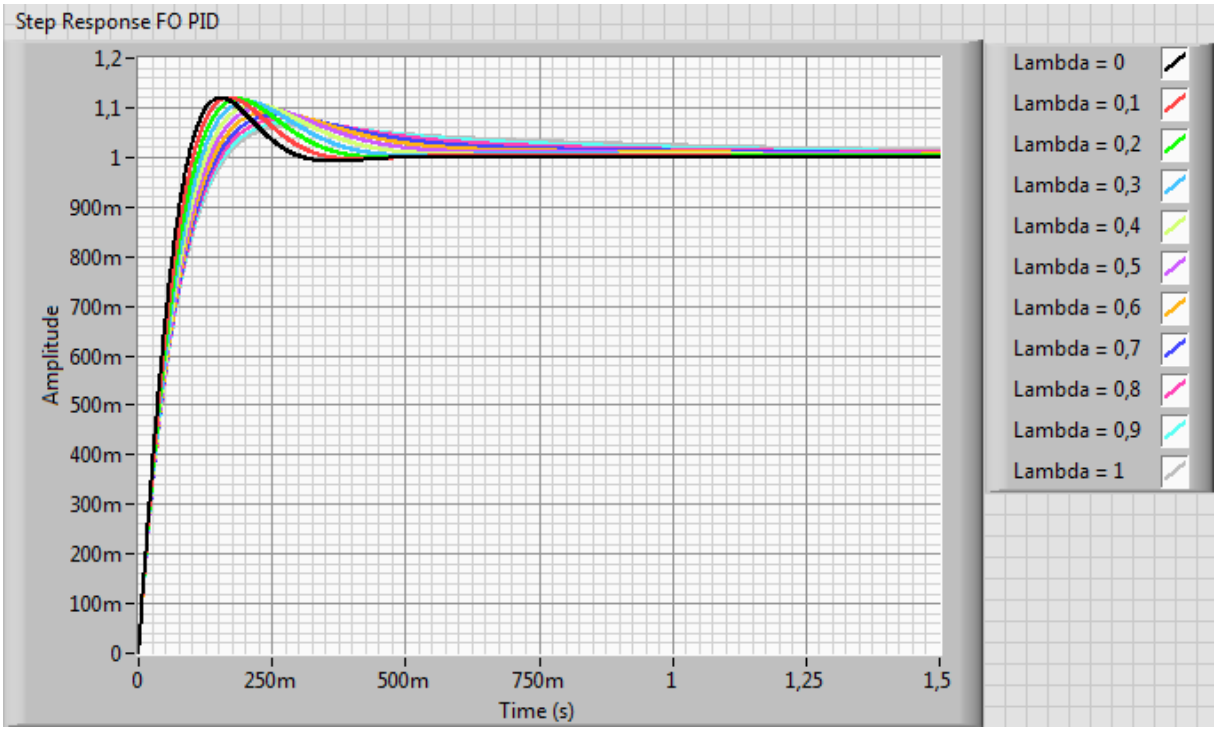

Figura 24 – Respostas ao degrau do sistema com $PI^{\lambda}D^{\mu}$ para diferentes valores de μ

Este experimento foi realizado da seguinte forma:

Para o primeiro gráfico, foi usado o PID convencional em malha fechada com o modelo da planta. Foram mantidos $K_p = 1$ e $K_d = 0,08$ enquanto K_i foi alterado de 0 a 1 em passos de 0,1.

Para o segundo gráfico, foi usado o $PI^{\lambda}D^{\mu}$ em malha fechada com o modelo da planta. Foram mantidos $K_p = 1$, $K_i = 1$, $K_d = 0,8$ e $\mu = 0,999$ enquanto λ foi alterado de 0 a 1 em passos de 0,1.

É possível observar que no primeiro gráfico, conforme K_i aumenta, o tempo de amortecimento do sistema aumenta com o efeito de integração se prolongando por mais tempo. No segundo caso a observação é parecida, pois o tempo de amortecimento também aumenta com o aumento de λ, porém para valores baixos de λ é observado um aumento do sobresinal. Isto por conta do efeito de uma ordem fracionária baixa se aproximar do efeito do ganho proporcional.

5.2 Experimento simulado em *software*

O próximo experimento foi feito por meio do módulo de simulação do LabVIEW. Neste modo, o LabVIEW usa um algoritmo para resolver o sistema que permite a inserção de não-linearidades ao sistema ainda em ambiente de *software*. Isto aproxima o experimento ao que será encontrado em experimentos reais, sem a necessidade de montar todo o aparato e permitindo extrapolações que o sistema real pode não permitir.

Neste experimento, foi inserida saturação à saída do controlador. Esta é uma não-linearidade que estará presente pois na avaliação teórica, explorada no capítulo anterior, a saída do controlador possui amplitude ilimitada, o que não é verdade em experimentos reais. A saturação inserida foi de -10V e 10V, que é a limitação do módulo de saída analógica que será usado no experimento real e também da entrada analógica de controle da planta. Esta saturação modifica a característica do sistema e pode alterar bastante a resposta do sistema quando a saída do controlador alcança a limitação de amplitude máxima na geração de sinal. Neste caso, a resposta ficou muito mais lenta. Para resolver isso, foi aumentado bastante o parâmetro proporcional do controlador.

Foram avaliadas as respostas do sistema a partir de duas entradas diferentes: o degrau unitário e uma entrada senoidal com frequência de 5Hz e amplitude 1.

Em todas as avaliações, foi utilizado o método de Runge-Kutta 23, que é um solver que permite passo de simulação variável, se ajustando ao melhor tempo de simulação para o sistema.

Para o degrau unitário, o controlador PID foi ajustado com os seguintes parâmetros: $K_p = 70$, $K_i = 1$, $K_d = 0,9$. Os resultados da resposta ao degrau e a saída do controlador podem ser observados nas figuras 25 e 26.

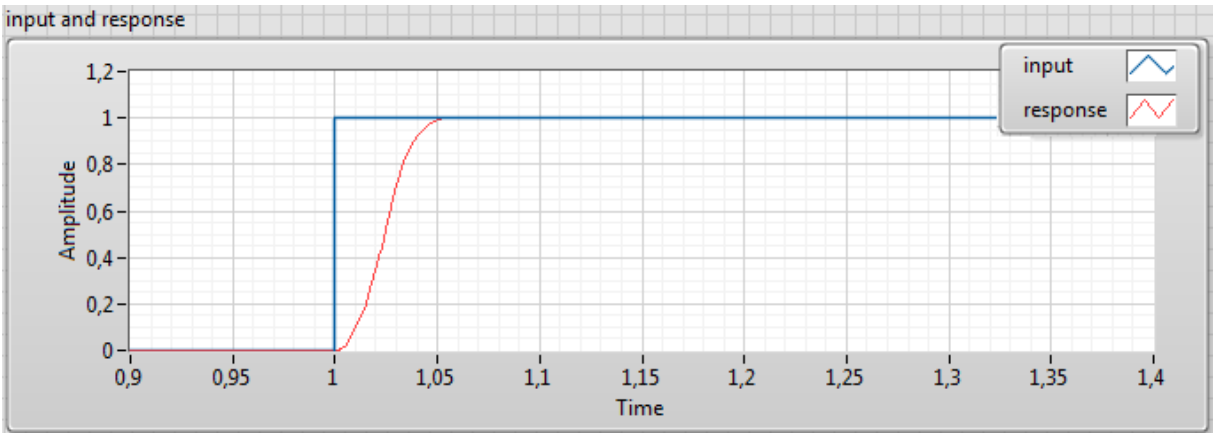

Figura 25 – Resposta ao degrau do controlador PID com saturação

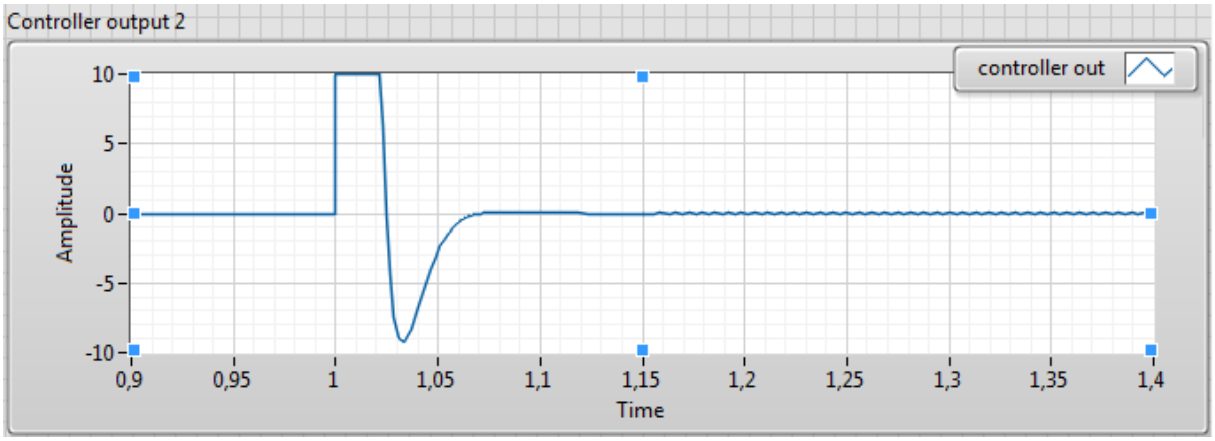

Figura 26 – Saída do controlador PID para a resposta ao degrau

A mesma avaliação foi feita com o controlador $PI^{\lambda}D^{\mu}$, que foi ajustado da seguinte forma: $K_p = 70$, $K_i = 1$, $\lambda = 0,9$, $K_d = 3,5$ e $\mu = 0,7$. Os resultados podem ser observados nas fugras 26 e 27.

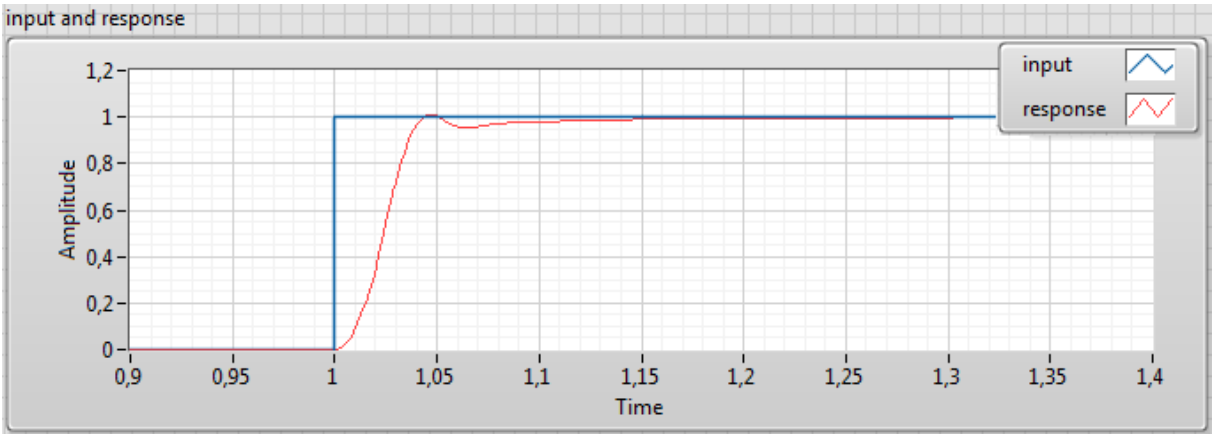

Figura 27 – Resposta ao degrau do controlador $PI^{\lambda}D^{\mu}$ com saturação

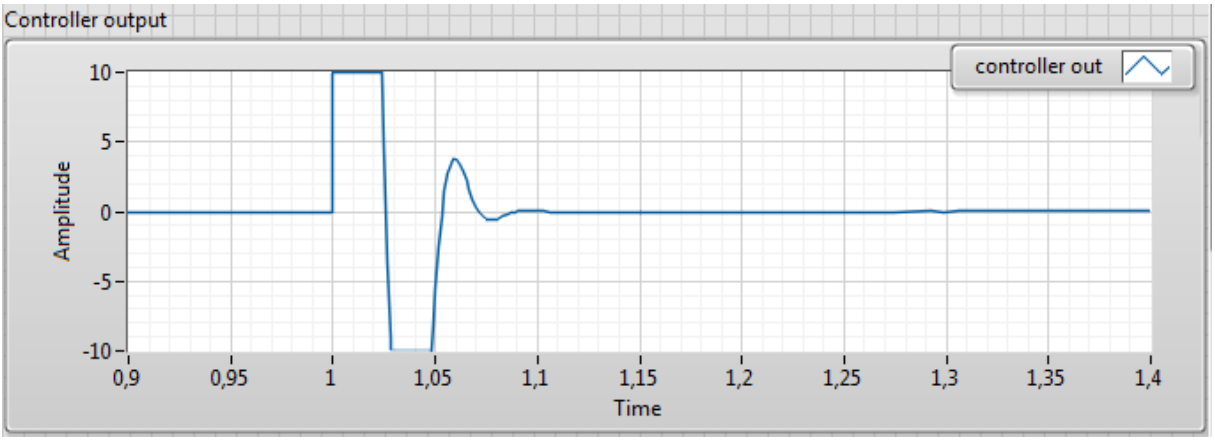

Figura 28 – Saída do controlador $PI^{\lambda}D^{\mu}$ para a resposta ao degrau

É possível observar que a resposta do controlador fracionário foi 10ms mais rápida que a do PID convencional, mas apresentou uma leve oscilação antes de estabilizar. Em casos que esta oscilação não impossibilite o uso do controlador, é possível considerar que a resposta do controlador fracionário foi melhor que a do PID convencional nesta condição. A saída do controlador fracionário também foi um pouco mais agressiva que a do PID convencional, aplicando sinal positivo e negativo ao motor e chegando à saturação em ambos antes do sistema chegar à acomodação. O PID convencional saturou apenas quando aplicou o sinal positivo.

O mesmo experimento foi realizado aplicando-se à entrada do sistema um sinal senoidal com frequência de 5Hz e amplitude 1.

Para a senoide, o controlador PID foi ajustado com os seguintes parâmetros: K_p = 100, K_i = 1, K_d = 10. Os resultados da resposta ao degrau e a saída do controlador podem ser observados nas figuras 29 e 30. Como a resposta ficou muito próxima ao sinal de entrada, também foi analisado o sinal de erro (Figura 31).

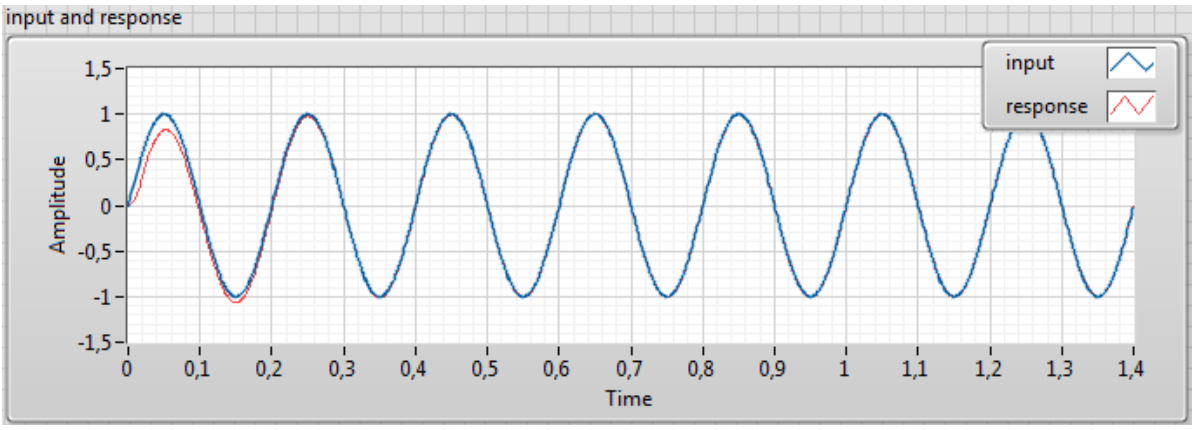

Figura 29 – Resposta à senoide do controlador PID com saturação

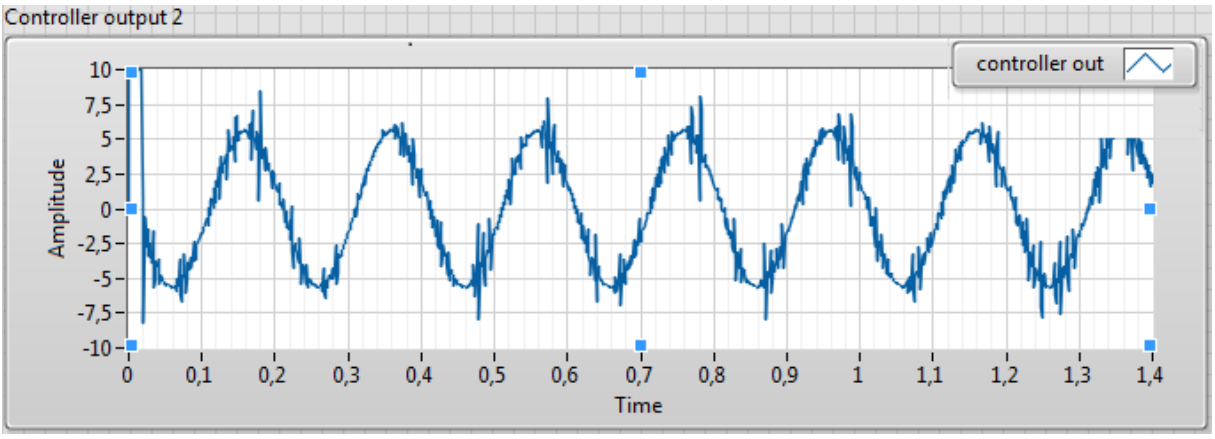

Figura 30 – Saída do controlador PID para a resposta à senoide

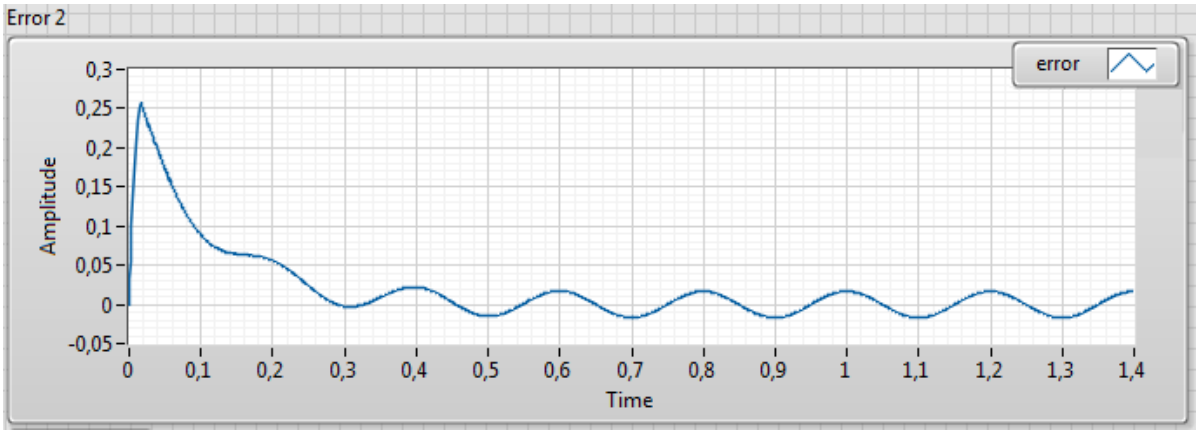

Figura 31 – Sinal de erro do controlador PID para resposta à senoide

A mesma avaliação foi feita com o controlador $PI^{\lambda}D^{\mu}$, que foi ajustado da seguinte forma: $K_p = 70$, $K_i = 1$, $\lambda = 0{,}9$, $K_d = 50$ e $\mu = 0{,}7$. Os resultados podem ser observados nas fugras 31, 32 e 33.

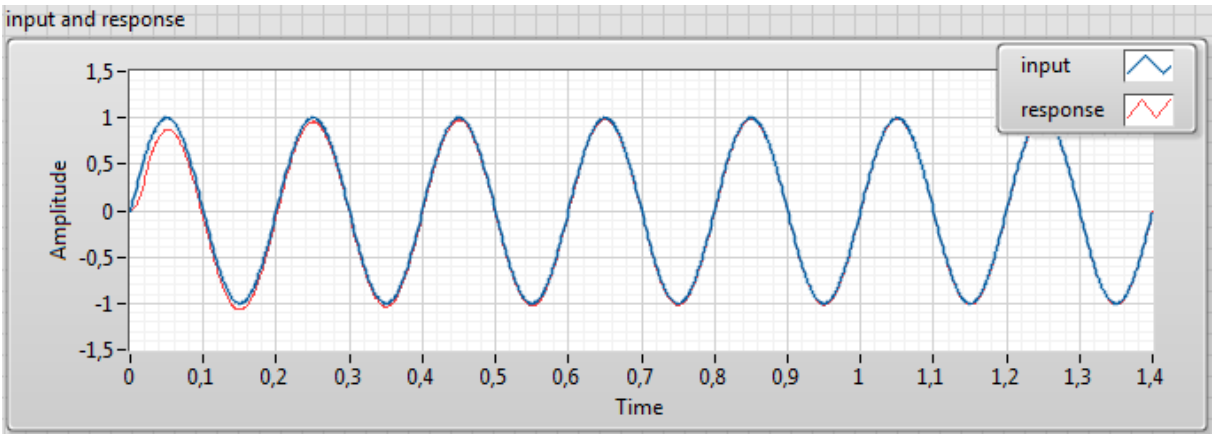

Figura 32 – Resposta à senoide do controlador $PI^{\lambda}D^{\mu}$ com saturação

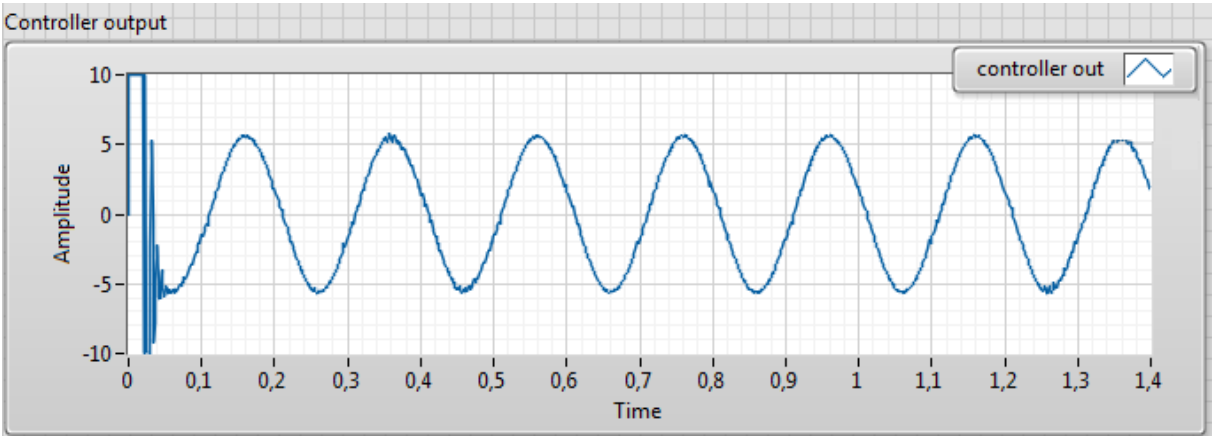

Figura 33 – Saída do controlador $PI^{\lambda}D^{\mu}$ para a resposta à senoide

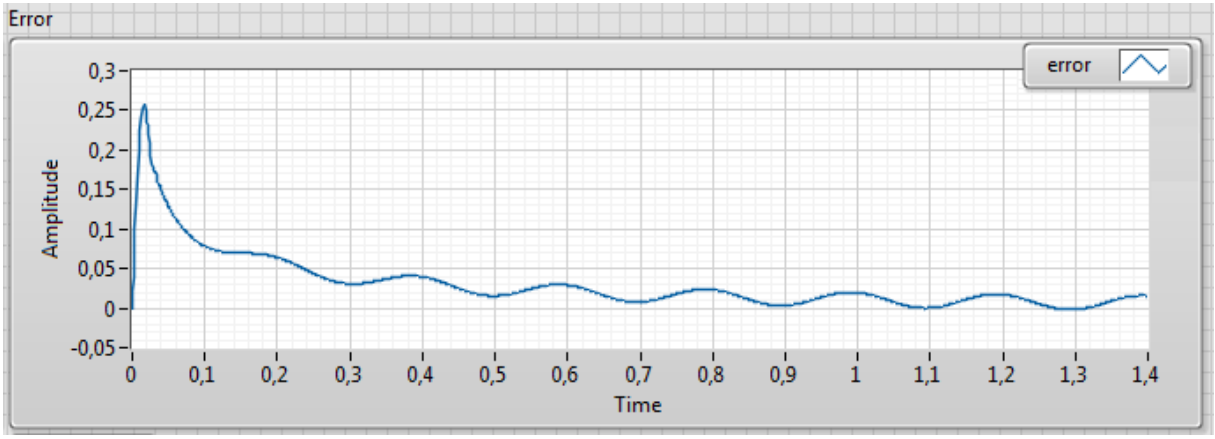

Figura 34 – Sinal de erro do controlador $PI^{\lambda}D^{\mu}$ para resposta à senoide

É possível observar pelas respostas à senoide (Figuras 29 e 32) e pelos sinais de erro (Figuras 31 e 34) que tanto na fase transitória quanto em regime o controlador $PI^{\lambda}D^{\mu}$ apresenta resultados melhores, reduzindo o pico inicial de erro mais rapidamente e mantendo um erro em regime com amplitude próxima à metade do erro apresentado pelo PID convencional.

5.3 Experimento com dispositivos reais.

O experimento foi realizado também em implementação real. A planta de controle de motor CC foi conectada ao CompactRIO e os algoritmos de controle foram descarregados na sua CPU para execução em tempo real. O diagrama da figura 35 mostra como foram feitas as conexões do experimento.

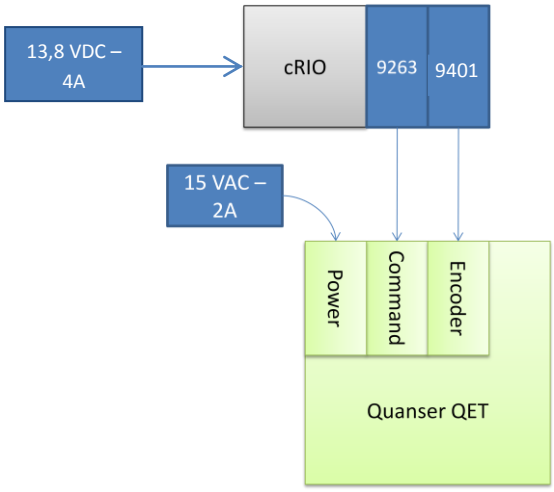

Figura 35 – Diagrama de conexões do sistema real

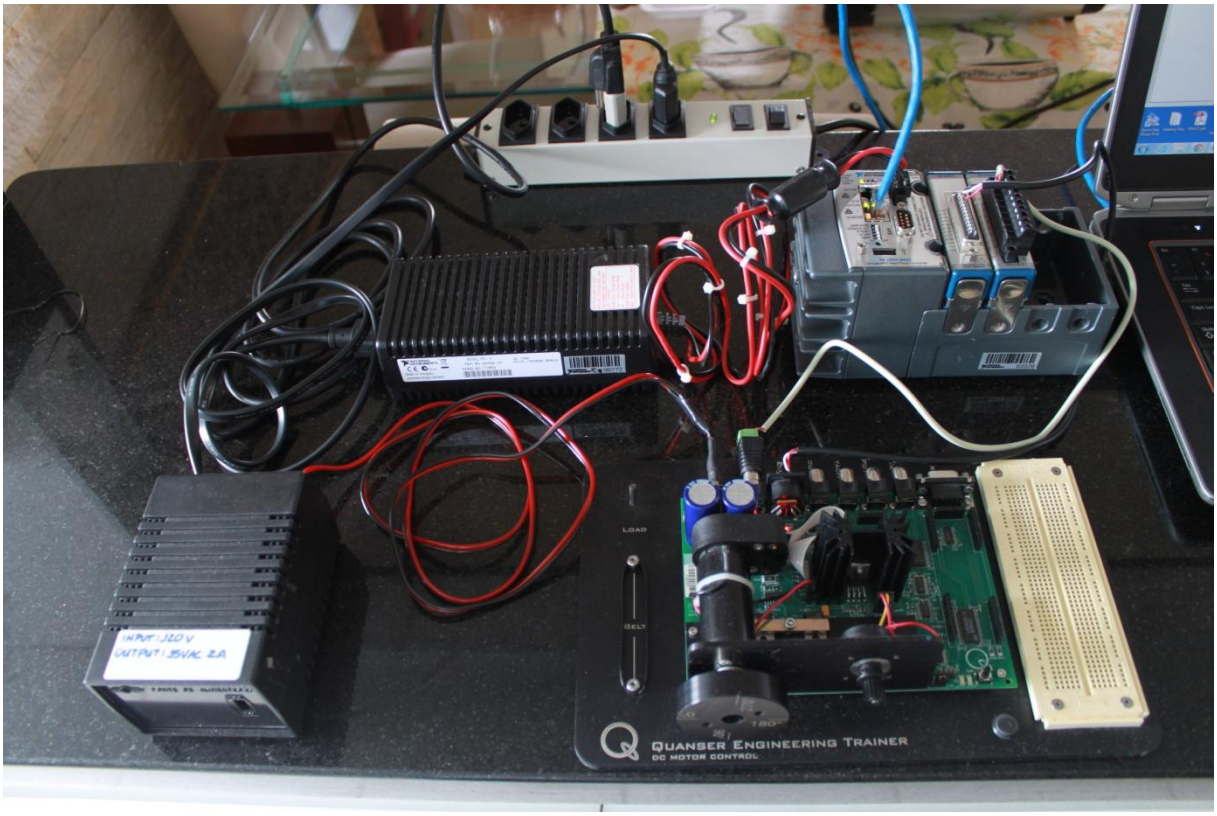

Figura 36 – Fotografia do sistema montado

Inicialmente foi executado um algoritmo de identificação de sistemas para confirmar a validade do modelo utilizado. Como foi utilizado um controlador diferente do controlador considerado pelo fabricante para estimativa do modelo (o fabricante descreve o uso de uma placa de controle com comunicação RS-232 com o computador), o ganho do sistema foi alterado, mas as demais constantes apresentaram resultados muito próximos. Para um período de amostragem de 0,005s, o modelo extraído da identificação de sistema foi:

$$F(s) = \frac{56{,}2}{0{,}105s^2 + s} \tag{21}$$

O experimento foi programado também com o módulo de simulação do LabVIEW, que permite sincronizar o loop de simulação com a varredura dos sinais de entrada e saída. Esta varredura foi programada com esse mesmo período de amostragem, de 0,005s. O loop de simulação foi configurado para usar o algoritmo

Runge-Kutta 1 (Euler) que mantém o passo de simulação constante (neste caso, configurado para 0,005s). Dessa forma, a mesma programação usada no experimento em simulação, foi usada no experimento prático, apenas ajustando estas configurações.

No experimento prático, mantendo-se os mesmos ajustes de K_p, K_i, λ, K_d e μ daqueles ajustados para a simulação de resposta ao degrau, foi observado que o sistema não apresentava os mesmos resultados da simulação em *software*, especialmente quando $K_d \neq 0$ impondo um erro muito elevado à resposta do sistema ou até instabilidade. Foram identificados dois pontos importantes:

- O primeiro foi que o passo de simulação variável com Runge-Kutta 23 estava fazendo a simulação executar com um passo menor que 0,001s. Forçando a simulação em *software* a executar com passo fixo de 0,005s era observada os mesmos erros elevados e instabilidade do experimento prático.

- O segundo foi que o parâmetro *High Frequency Time Constant*, Tf, do controlador PID não estava sendo usado.

O parâmetro Tf é um filtro passa baixas aplicado à ação derivativa do controlador PID para tornar o modelo do sistema realizável. Este parâmetro também resolve outro problema atenuando ruído de alta frequência, já que a ação derivativa é muito sensível a ruídos de sinal podendo levar o sistema à instabilidade ao responder a ruído.

Aplicando um valor de 0,01s a Tf, foi possível resolver os problemas no PID convencional e a resposta se tornou compatível com os resultados observados na simulação em *software*.

No $PI^{\lambda}D^{\mu}$ o mesmo cálculo de Tf foi inserido no programa, mas não foram observados alterações nos resultados.

Além do problema observado por conta de Tf, foi possível observar que o resultado do sistema em malha fechada variava bastante com a variação da velocidade de execução da malha de controle, mesmo na simulação em software. As figuras 37 e 38 mostram o resultado da simulação em *software* para o $PI^{\lambda}D^{\mu}$ com passo de simulação em 0,001s e em 0,005s, aplicando-se Tf = 0,01s.

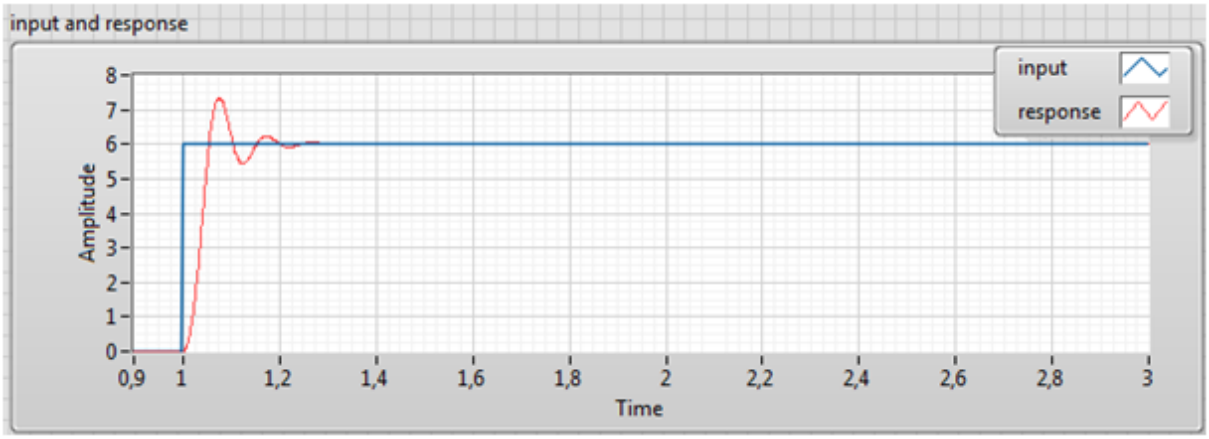

Figura 37 – Resposta ao degrau com simulação em 0,001s

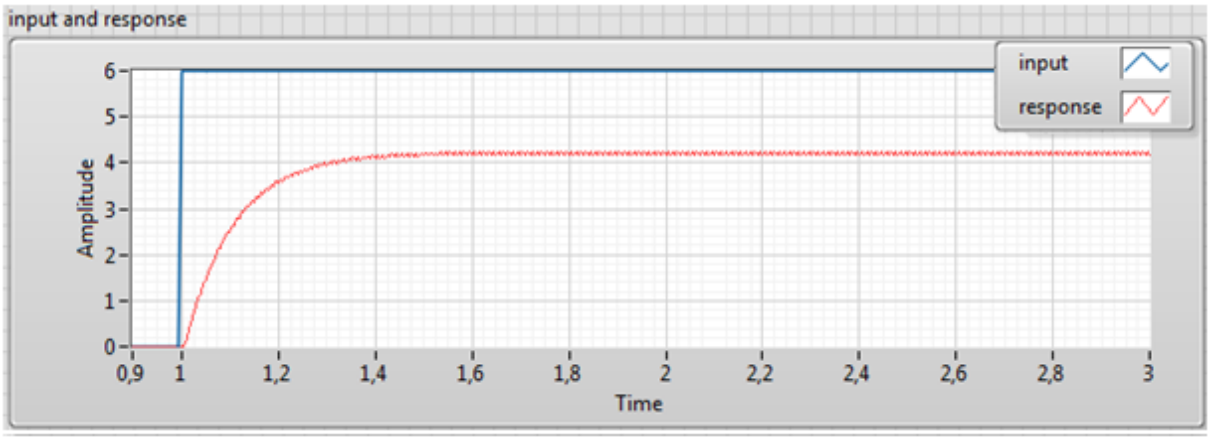

Figura 38 – Resposta ao degrau com simulação em 0,005s

A CPU de tempo real do compactRIO utilizada neste experimento não alcança o processamento necessário para executar o sistema de controle com período de 0,001s, sendo que o passo mínimo alcançado foi de 0,005s, o que impossibilitou a verificação prática se o aumento da frequência do loop de controle resolveria o problema da ação derivativa.

Por outro lado, quando a ação derivativa é nula, $K_d = 0$, os resultados são compatíveis com os da simulação e é possível alcançar a estabilidade com sobresinal e oscilação antes da acomodação (Figura 39).

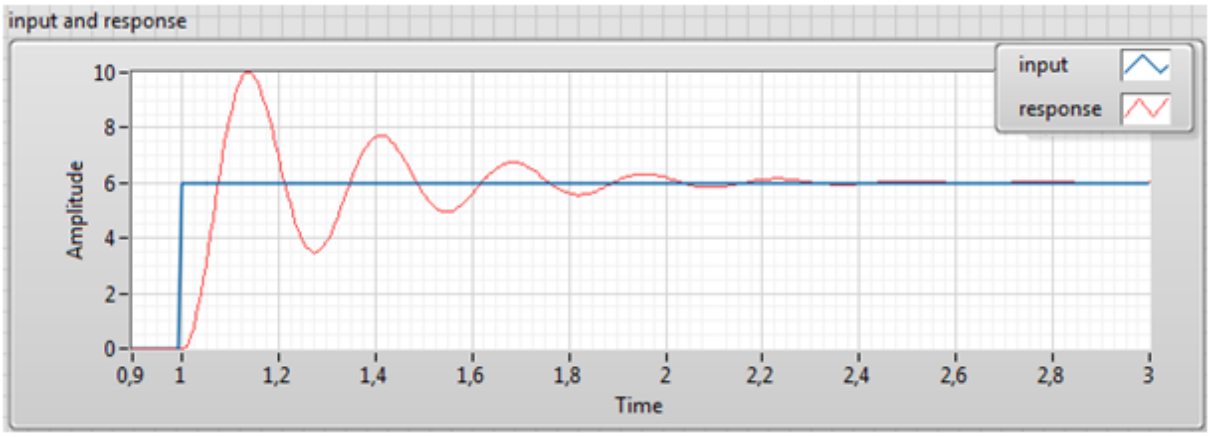

Figura 39 – Resposta ao degrau sem atuação derivativa

6 CONCLUSÕES E PRÓXIMAS ETAPAS

6.1 Conclusões

Alguns objetivos inicialmente propostos foram atingidos e é possível pontuar uma série de resultados importantes alcançados:

- Foi apresentado, testado e validado um programa desenvolvido em LabVIEW para a execução do operador fracionário s^r;

- Foi identificada uma relação entre os limites de frequência ω_A e ω_B usados no método de discretização indireta que torna a aproximação válida (equação 18);

- Foi implementado o controlador $PI^{\lambda}D^{\mu}$ e foram realizados diversos experimentos simulados e reais que agregaram muita informação sobre o comportamento do operador fracionário e de variações de resposta impostas por variações dos parâmetros do controlador;

- Foi identificado por meio das simulações em *software* que o $PI^{\lambda}D^{\mu}$ pode apresentar resultados melhores que os obtidos com o PID convencional e que os ajustes λ e μ podem alterar a forma de resposta do sistema, não sendo apenas ajustes finos dos parâmetros integrativo e derivativo.

6.2 Próximas etapas

Tendo em vista a construção do conhecimento científico, este trabalho pode ser usado como guia para explorar alguns pontos com maior profundidade.

É importante estudar a implementação do método de discretização direta para a realização do operador fracionário s^r e implementações de filtro passa baixas na atuação derivativa do controlador $PI^{\lambda}D^{\mu}$ afim de resolver os problemas de execução prática com sinais reais. Também pode ser alvo de estudos futuros a implementação deste mesmo sistema de controle com uma controladora de tempo real com maior capacidade de processamento afim de identificar se a execução com taxa de amostragem mais alta resolveria os problemas da atuação derivativa.

REFERÊNCIAS

Barbosa, R. S.; Machado, J. A. T. **Implementation of discrete-time fractional-order controllers based on LS approximation.** Department of Electrotechnical Engineering, Institute of Engineering of Porto, Acta Polytechnica Hungarica, Vol. 3, No. 4, Porto, Portugal, 2006

Manabe, S. **The non-integer integral and its application to control systems.** UDC 621-501.2:517.39, Vol. 6, N°3/4, 1961

Podlubny, I.; Dorcak, L.; Kostial, I. **On Fractional Derivatives, Fractional-Order Dynamic Systems and $PI^{\lambda}D^{\mu}$-controllers.** Proceendings of the 36th Conference on Decision and Control, San Diego, California, USA, Dezembro, 1997

Podlubny, I. **Fractional-Order Systems and $PI^{\lambda}D^{\mu}$-Controllers.** IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 44, NO. 1, JANUARY 1999

Xue, D; Chen, Y. **A Comparative Introduction of Four Fractional Order Controllers.** Proceedings of the 4th World Congress on Intelligent Control and Automation, Shanghai, China, June, 2002

Chen, Y. Q.; Moore, K. L. **Discretization Schemes for Fractional-Order Differentiators and Integrators**. IEEE Transactions on Circuits and Systems—I: Fundamental Theory and Applications, VOL. 49, NO. 3, 2002

Al-Alaoui, M. A. **Simulation and Discretization of Fractional Order Systems**. Department of Electrical and Computer Engineering, American University of Beirut, Beirut, Lebanon, 2009.

Jin, Y. et. al. **LabVIEW Based Experimental Validation of Fractional Order Motion Controllers.** Control and Decision Conference, 2009. CCDC '09. Chinese, vol., no., pp.323,328, 17-19 June 2009

Li, H.; Luo, Y.; Chen, Y. **A Fractional Order Proportional and Derivative (FOPD) Motion Controller: Tuning Rule and Experiments.** IEEE Transactions on Control Systems Technology, VOL. 18, NO. 2, Março 2010

ANEXO 1 - Desenvolvimento do programa que resolve s^r pela discretização direta

Neste anexo poderão ser encontrados os esforços de desenvolvimento do controlador por meio do método de discretização direta, o qual resultaria na função de transferência no domínio discreto. Este desenvolvimento não foi concluído de forma satisfatória e por isso, o trabalho é apresentado apenas com a implementação por discretização indireta. O anexo é apresentado para poder contribuir como guia parcial para trabalhos futuros.

1.1 Desenvolvimento

Para a implementação deste programa, foram adotados os métodos propostos por Barbosa e Machado (2006), Chen e Moore (2002) e Al-Alaoui (2009). Conforme apresentado nestes trabalhos, um dos métodos mais eficientes para a implementação do operador s^r para r não inteiro é a discretização direta de s^r, usando-se a transformada de Al-Alaoui, que resulta numa função de transferência não-racional no domínio z, e então aplicando-se o método de mínimos quadrados pelo método de Prony para, finalmente, obter a função de transferência racional no domínio z.

Para a implementação do método descrito acima, a primeira tarefa é realizar a discretização direta usando-se a transformada de Al-Alaoui. A transformada de Al-Alaoui é a seguinte:

$$s^r \approx \left(\frac{8}{7T} \frac{1 - z^{-1}}{1 + z^{-1}/_7} \right)^r \tag{22}$$

Onde T é o período de amostragem

A expansão desta transformada em série de potências é feita usando-se a resposta ao impulso da função:

$$H_A^r(z^{-1}) = \left(\frac{8}{7T} \frac{1 - z^{-1}}{1 + z^{-1}/_7} \right)^r = \left(\frac{8}{7T} \right)^r \sum_{k=0}^{\infty} \left[\sum_{j=0}^{k} (-1)^j \left(\frac{1}{7} \right)^{k-j} \binom{r}{j} \binom{-r}{k-j} \right] z^{-k} \tag{23}$$

$$H_A^r(z^{-1}) = \sum_{k=0}^{\infty} h_A^r(k) z^{-k}$$

$$h_A^r(k) = \left(\frac{8}{7T} \right)^r \sum_{j=0}^{k} (-1)^j \left(\frac{1}{7} \right)^{k-j} \binom{r}{j} \binom{-r}{k-j} \tag{24}$$

Lembrando que conforme a equação (3):

$$\binom{r}{k} = \frac{\Gamma(r+1)}{\Gamma(k+1)\Gamma(r-k+1)}$$

Dessa forma o primeiro passo é implementar um programa que realize $\binom{r}{k}$. Como o LabVIEW já possui a função Gamma (Γ) na sua biblioteca de matemática (Mathmatics>>Elementary & Special Functions>>Gamma Functions), a implementação do programa se tornou bastante simples, conforme é possível observar o diagrama de blocos na figura 40.

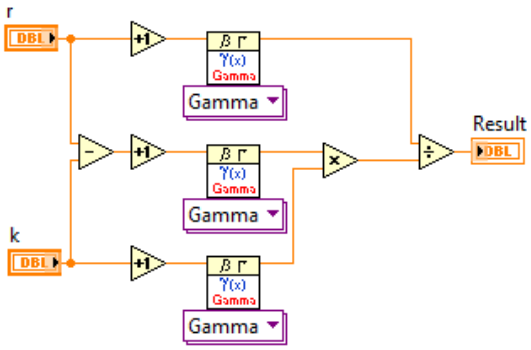

Figura 40 – Diagrama de blocos do programa que calcula $\binom{r}{k}$

O próximo passo é desenvolver o programa que irá calcular $h_A^r(k)$. Para isso, é necessário estruturar um programa que calcule a equação (24):

$$h_A^r(k) = \left(\frac{8}{7T}\right)^{\alpha} \sum_{j=0}^{k} (-1)^j \left(\frac{1}{7}\right)^{k-j} \binom{\alpha}{j} \binom{-\alpha}{k-j}$$

Para este programa, não há recursos especiais a serem usados. É necessário estruturar o algoritmo que calcule a equação passo a passo. Este algoritmo é apresentado no diagrama de blocos da figura 41. Note-se que o SubVI utilizado, é exatamente o programa que apresentamos anteriormente na figura 40.

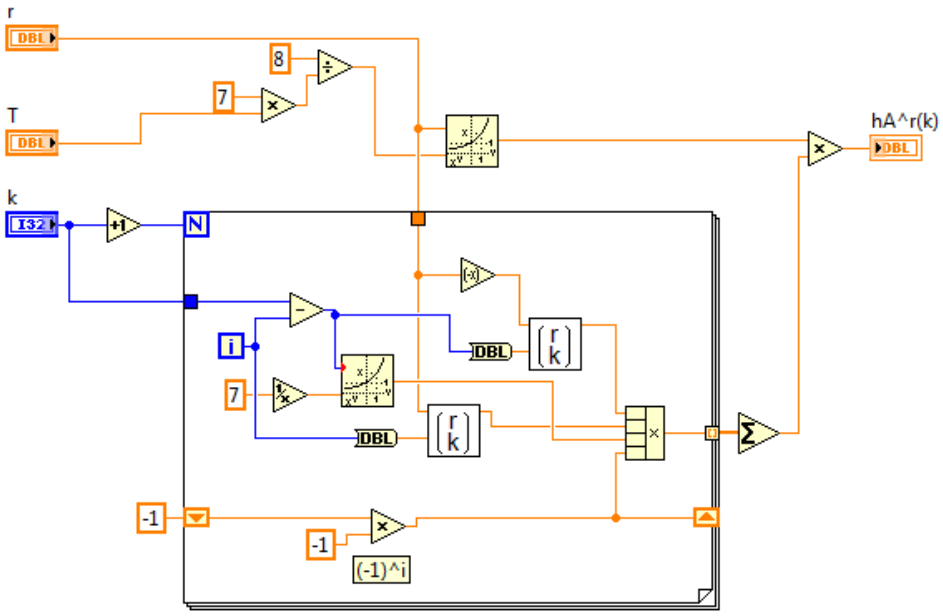

Figura 41 – Diagrama de blocos do programa que calcula $h_A^r(k)$

Finalmente, usa-se o método de Prony para obter a função de transferência no domínio discreto, z, que se aproxima ao operador fracionário s^r. BARBOSA e MACHADO (2006) apresentam o método de Prony em alguns passos. Trata-se de uma evolução da aproximação de Padé para o projeto de filtros IIR (*Infinite Impulse Response*) baseado no método de mínimos quadrados. Com este método, a partir da função resposta ao impulso $h_A^r(k)$, é possível extrair uma função de transferência racional no domínio do tempo discreto, z, do tipo:

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_m z^{-m}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} = \sum_{k=0}^{\infty} h(k) z^{-k} \qquad (25)$$

Onde $h(k)$ é a função resposta ao impulso e $m \leq n$.

O método de Prony baseia-se em estruturar um sistema linear de equações para determinar os coeficientes $a(k)(k = 1, 2, \ldots, n)$ e $b(k)(k = 0, 1, 2, \ldots, m)$. O diagrama de blocos do programa que aplica o método de Prony para retornar os coeficientes $a(k)$ e $b(k)$ é apresentado na figura 42.

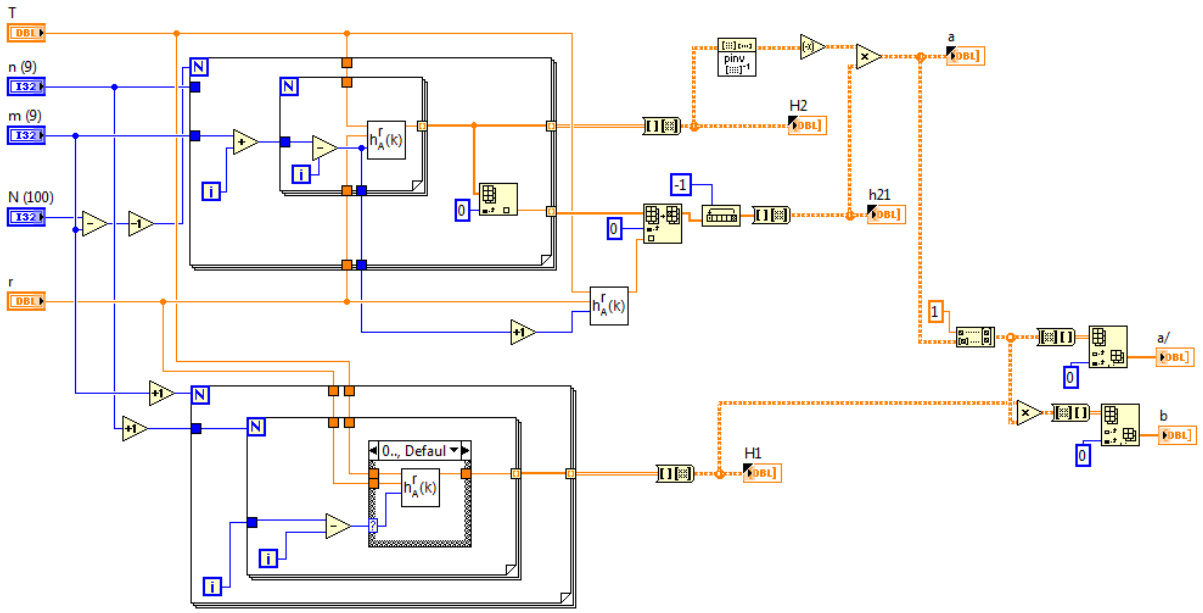

Figura 42 – Diagrama de blocos do programa que aplica Prony e retorna $a(k)$ e $b(k)$

Com isto, temos o programa que calcula s^r e fornece uma função de transferência aproximada no domínio do tempo discreto, z.

Durante o desenvolvimento e testes, foi observado que este algoritmo funciona bem para valores de r não inteiros. Para r de valor inteiro ele não consegue calcular os coeficientes da função de transferência e retorna apenas uma função constante = 1. Também é observado que conforme o valor de r se distancia de 0, a fidelidade da aproximação cai. Assim, para obter os melhores resultados, limitaremos o uso desta função a

$$-1 < r < 1 \tag{26}$$

Dessa forma, trabalhamos na faixa onde a aproximação é a mais exata possível e para o único valor inteiro, a saída é uma função constante unitária. Isto simplifica bastante o desenvolvimento do próximo passo que é a implementação do controlador $PI^{\lambda}D^{\mu}$. Por outro lado, representa uma limitação do controlador à medida que ordens de integração maior que |1| poderiam ser verificadas, por exemplo.

1.2 Resultados Obtidos

As primeiras análises foram feitas no programa que realiza s^r. Neste caso, foram experimentados diversos valores em r e observado o comportamento de resposta em frequência do modelo discreto de s^r. Para isto, foi construído um programa com o diagrama de blocos apresentado na figura 43.

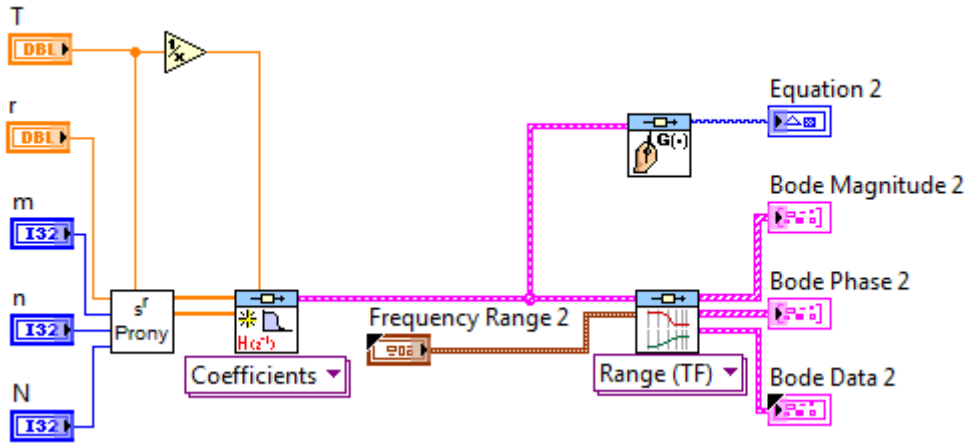

Figura 43 – Diagrama de blocos do programa que analisa s^r

Para o experimento, foram adotados os seguintes parâmetros:

T = 0,01

m = 9

n = 9

N = 100

Para r foram experimentados diversos valores afim de analisar o comportamento da função.

O primeiro passo foi observar se para valores muito próximos de |1| a função se aproximaria das funções integrador e derivador. Nestes casos, a resposta em frequência apresentou um resultado em ganho satisfatório para uma faixa de frequência até 5 décadas inferior à frequência de amostragem (1/T). Para fase, os resultados são satisfatórios para uma faixa entre 4 e 1 década inferior à ferquência de amostragem. Os mesmos resultados foram observados variando o valor de T.

A figura 44 apresenta o resultado para um valor de r = -0,999 que se aproxima a um integrador (1/s). para a faixa de frequência entre 10mHz e 10Hz

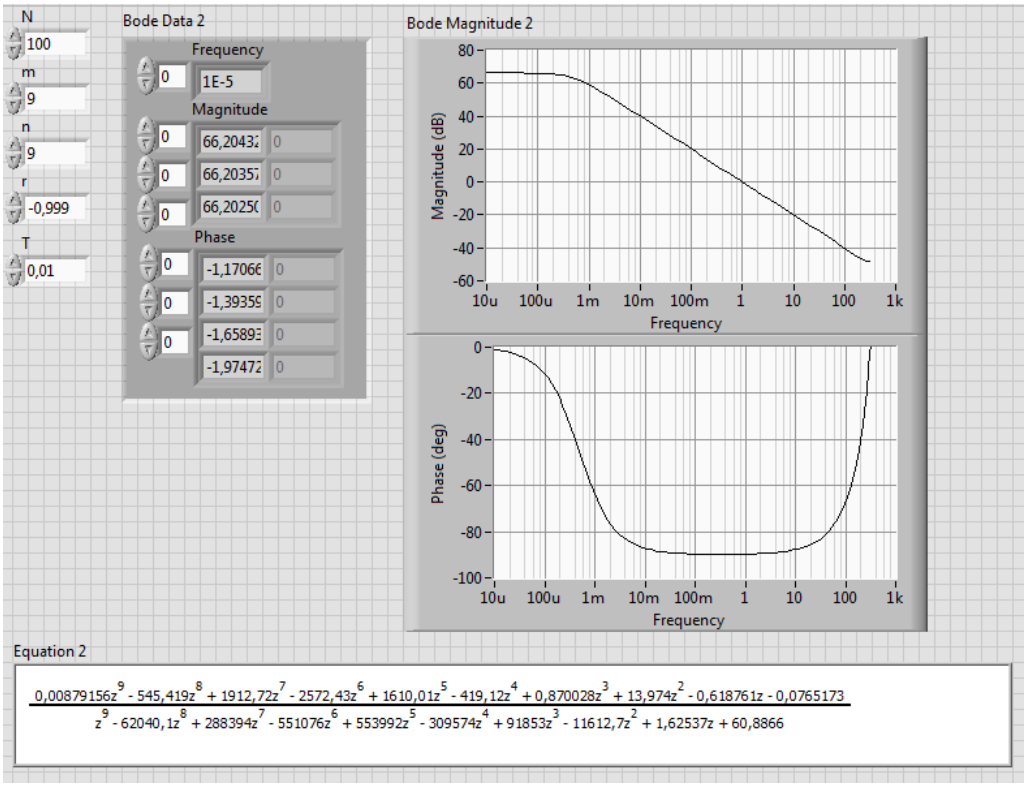

Figura 44 – Resposta em frequência de s^r para r = -0,999

Resultado semelhante foi observado para r = 0,999, onde ocorre a aproximação a um derivador (s) (Figura 45).

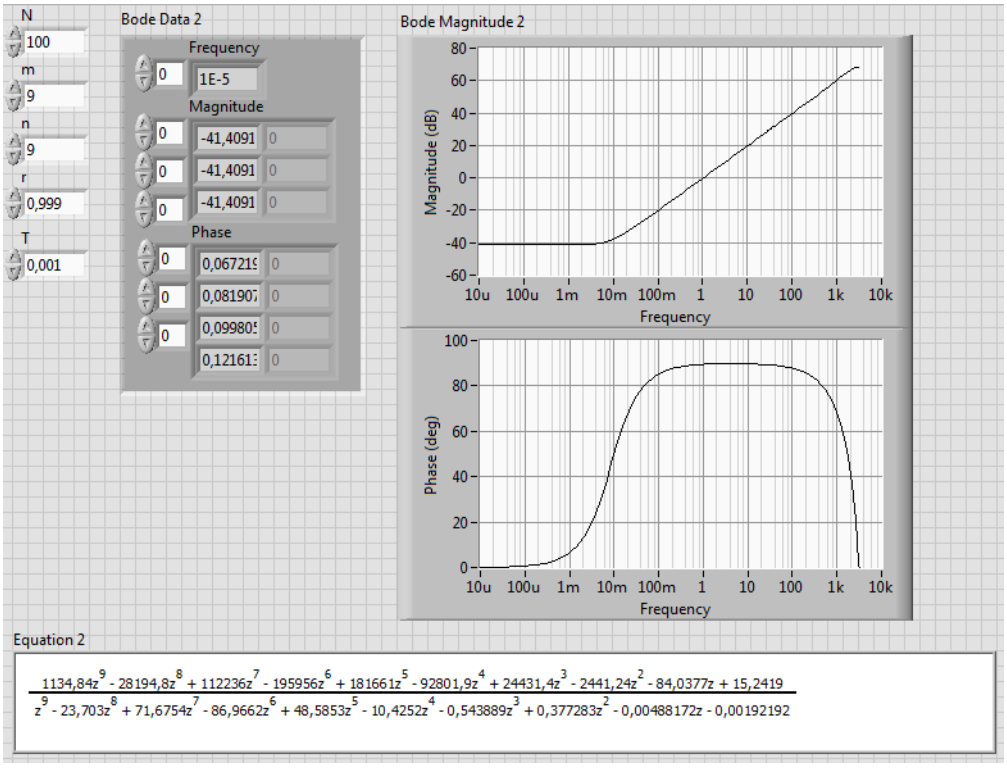

Figura 45 – Resposta em frequência de s^r para r = 0,999

Para valores intermediários de r, é possível observar uma variação entre a resposta do integrador e a resposta do derivador, reduzindo a inclinação da curva de ganho e reduzindo-se a fase do sistema. Na Figura 46 é possível observar a resposta para r = 0,5.

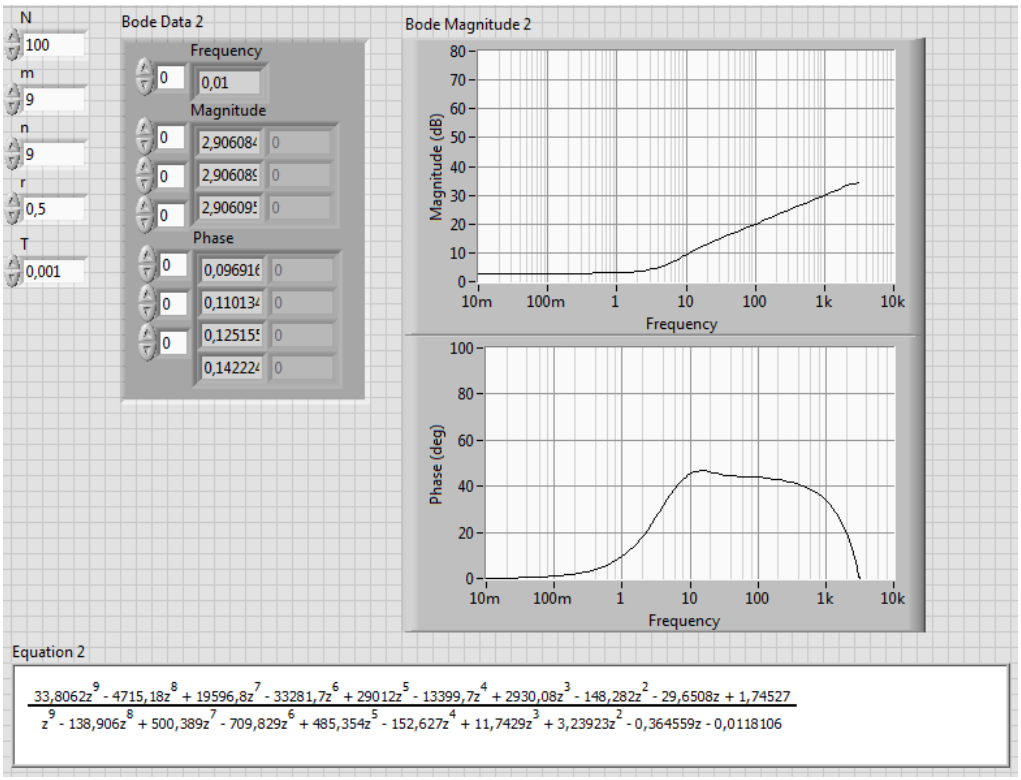

Figura 46 – Resposta em frequência de s^r para r = 0,5

Os resultados de resposta em frequência de sr podem ser considerados satisfatórios e o passo seguinte foi a implementação do controlador $PI^{\lambda}D^{\mu}$, usando-se o programa s^r como SubVI conforme apresentado no na figura 47.

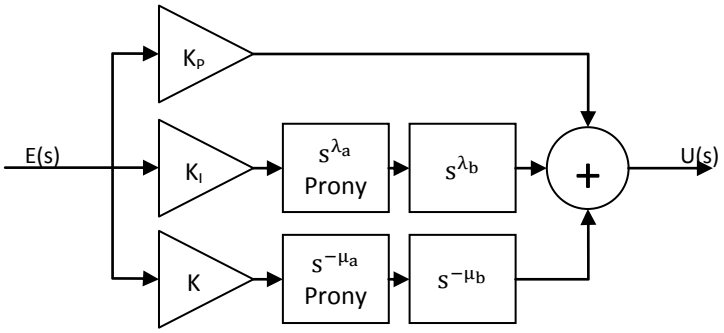

Figura 47 – Diagrama de blocos do controlador $PI^{\lambda}D^{\mu}$ (Discretização Direta)

Com o controlador $PI^{\lambda}D^{\mu}$ implementado foram realizadas as mesmas análises de reposta em frequência em malha aberta, usando-se valores de r muito próximos a |1| e comparando-se ao controlador PID convencional.

Para este teste, foram usados os seguintes parâmetros:

T = 0,01

$K_p = 1$

$K_i = 0,9$

$\lambda = 0,999$

$K_d = 0,01$

$\mu = 0,999$

Os parâmetros m, n, e N foram mantidos nos mesmos valores que no experimento anterior e como constantes no SubVI s^r.

Os resultados observados na simulação podem ser vistos na figura 48.

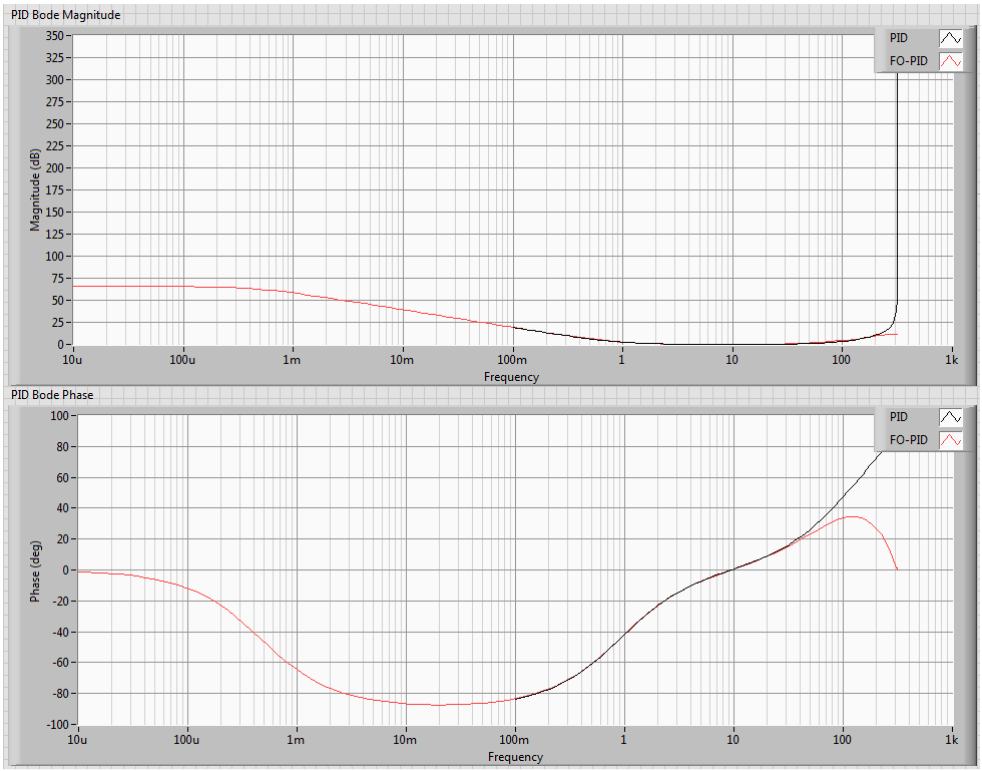

Figura 48 – Resposta em frequência PID e $PI^{\lambda}D^{\mu}$

Inicialmente o resultado parece discrepante, mas quando são ajustadas as escalas do gráfico para valores de frequência compatíveis com o período de amostragem T, o resultado fica muito próximo (Figura 49).

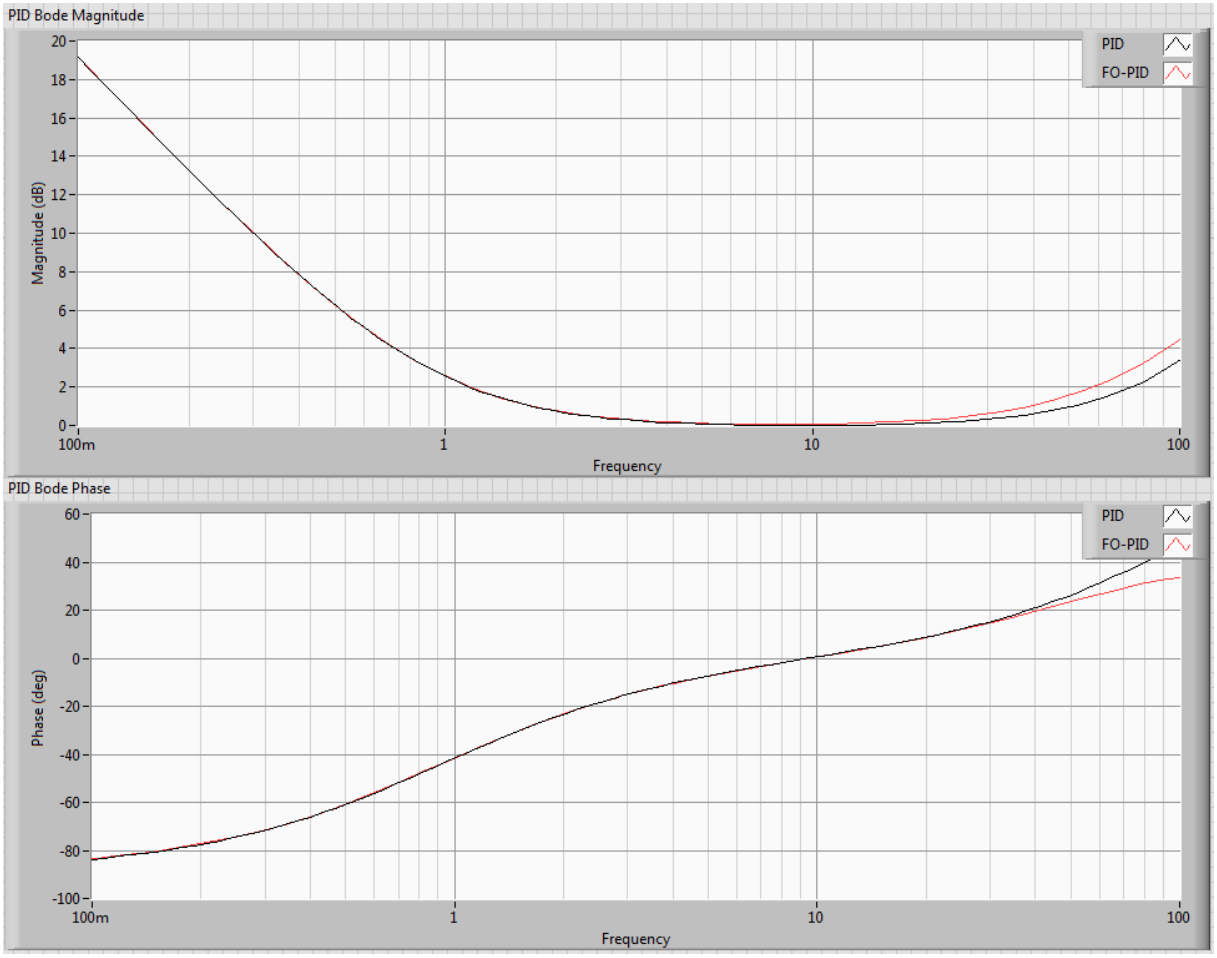

Figura 49 – Resposta em frequência PID e $PI^{\lambda}D^{\mu}$, escalas ajustadas

A partir daí, foram realizados experimentos de resposta no tempo dos controladores, ainda em malha aberta, e neste ponto foram observados os problemas que impediram o uso deste método no trabalho.

A figura 50 apresenta respectivamente a resposta ao degrau dos controladores PID e $PI^{\lambda}D^{\mu}$ em malha aberta usando-se os mesmos parâmetros descritos anteriormente:

T = 0,01

$K_p = 1$

$K_i = 0,9$

$\lambda = 0,999$

$K_d = 0,01$

$\mu = 0,999$

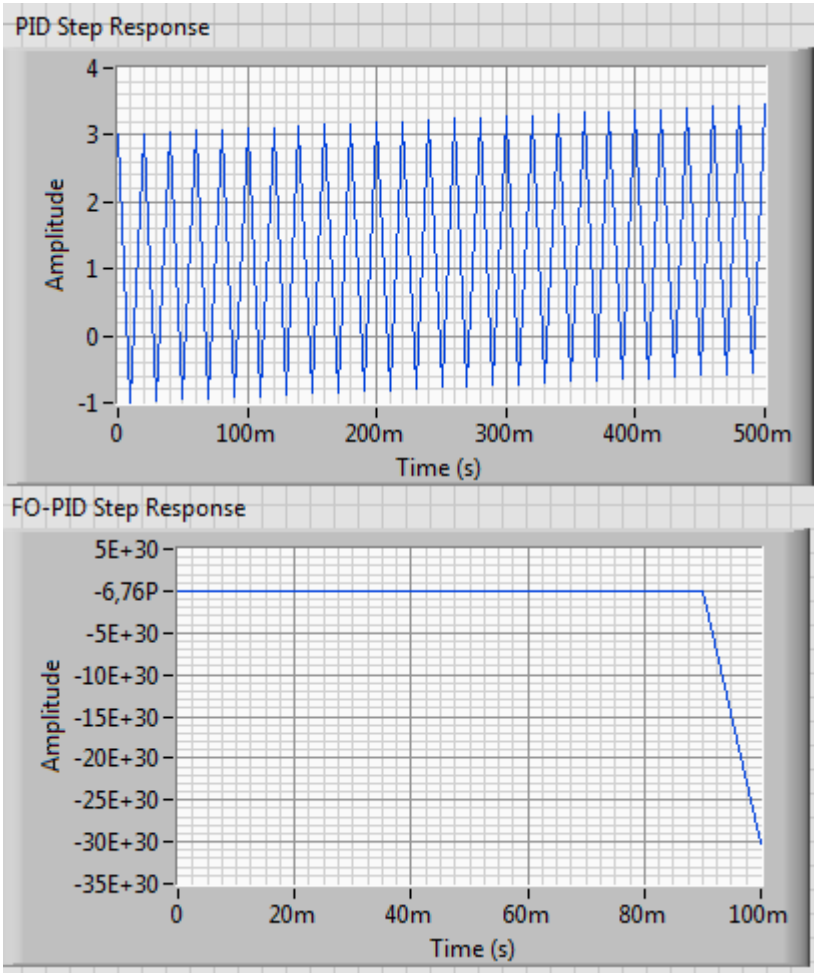

Figura 50 – Resposta ao degrau dos controladores PID e $PI^{\lambda}D^{\mu}$

Esta resposta instável inviabiliza o uso deste controlador em aplicações.

Para resposta no domínio do tempo, independente da entrada aplicada e se o controlador está sendo analisado individualmente em malha aberta ou em malha fechada com a planta, os resultados são instáveis.

Dessa forma, futuros trabalhos podem usar o trabalho descrito neste anexo como um guia inicial para o desenvolvimento de um controlador fracionário usando o método de discretização direta.