UNIVERSIDADE FEDERAL DE VIÇOSA
CAMPUS DE VIÇOSA
ENGENHARIA ELÉTRICA

DENISE CRISTINA HENRIQUE DE FREITAS

DESENVOLVIMENTO DE SOFTWARE PARA ANÁLISE DE SISTEMAS DE CONTROLE NO TEMPO DISCRETO

VIÇOSA
2021

DENISE CRISTINA HENRIQUE DE FREITAS

DESENVOLVIMENTO DE SOFTWARE PARA ANÁLISE DE SISTEMAS DE CONTROLE NO TEMPO DISCRETO

Monografia apresentada à Universidade Federal de Viçosa como parte das exigências para a aprovação na disciplina ELT 402 - Projeto de Engenharia II

Orientador: André Gomes Tôrres

VIÇOSA

2021

DENISE CRISTINA HENRIQUE DE FREITAS

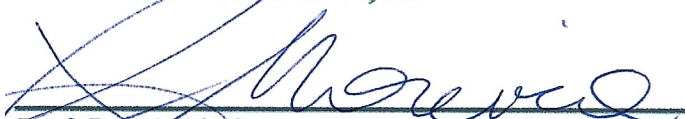
DESENVOLVIMENTO DE SOFTWARE PARA ANÁLISE DE SISTEMAS DE CONTROLE NO TEMPO DISCRETO

Monografia apresentada ao Departamento de Engenharia Elétrica do Centro de Ciências Exatas e Tecnológicas da Universidade Federal de Viçosa, para a obtenção dos créditos da disciplina ELT 402 – Projeto de Engenharia II e cumprimento do requisito parcial para obtenção do grau de Bacharel em Engenharia Elétrica.

Aprovada em 03 de novembro de 2021.

COMISSÃO EXAMINADORA

Prof. Dr. André Gomes Torres - Orientador
Universidade Federal de Viçosa



Prof. Dra. Kétia Soares Moreira - Membro
Universidade Federal de Viçosa

Prof. Dra. Mara Cristina da Silveira Coelho - Membro
CEFET - MG

Agradecimentos

Em primeiro lugar, agradeço a Deus, que permitiu que eu tivesse saúde determinação para ultrapassar todos os obstáculos encontrados ao longo desses anos de estudo.

Aos meus pais, Aparecida e Adirson, e a minha irmã, Letícia, por todo apoio e incentivo prestados durante a graduação.

Ao meu orientador, André Gomes Tôrres, por todo o apoio, paciência e direcionamento dado ao longo da elaboração deste trabalho.

Aos amigos que fiz durante os anos de graduação, por compartilharem momentos incríveis comigo e tornarem a trajetória muito mais leve e divertida.

E por fim, agradeço a Universidade Federal de Viçosa, seu corpo docente e administração que tão bem me receberam e que, através de suas diversas iniciativas, proporcionaram uma caminhada repleta de aprendizado e oportunidades.

Resumo

Este trabalho trata do desenvolvimento de um software que realiza a discretização de sistemas de controle através do método da Transformação Bilinear (Aproximação de Tustin). São abordados conceitos básicos referentes à sistemas de controle no tempo contínuo e discreto, às características da transformação bilinear, assim como alguns paradigmas da programação orientada a objetos, necessários para compreensão do projeto desenvolvido.

O software foi elaborado no ambiente de desenvolvimento da Microsoft Visual Studio, utilizando as distintas ferramentas disponíveis, a fim de criar um aplicativo de interface amigável e interativo, que possibilite a análise gráfica da resposta de um sistema no tempo discreto, a partir das informações de entrada no tempo contínuo fornecidas pelo usuário.

A validação do software foi realizada através da análise do erro entre a resposta do sistema discretizado e a resposta obtida pelo sistema no tempo contínuo. Tal comparação foi realizada através do *software* MATLAB. O aplicativo criado funciona como uma ferramenta de auxílio para estudantes, bem como projetistas de controladores digitais.

Palavras-chaves: software, Tranformação Bilinear, controle discreto, programação orientada a objetos.

Lista de ilustrações

Sumário

1 Introdução

Existem diversos registros que evidenciam a tentativa dos seres humanos em empregar fundamentos da teoria de controle em suas atividades cotidianas ao longo da história. No século XVIII, James Watt construiu um regulador centrífugo (Figura 1) para o controle de velocidade de uma máquina a vapor, sendo este o primeiro trabalho significativo relacionado ao controle automático. Minorsky, Hazen e Nyquist também foram responsáveis pelo desenvolvimento de trabalhos relevantes nos primeiros períodos da concepção da teoria de controle (OGATA, 2010).

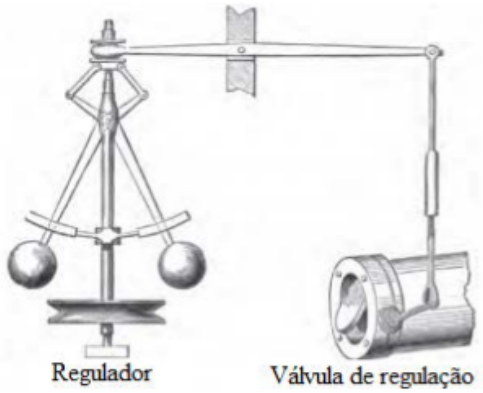


Figura 1 – Regulador centrífugo.
Fonte: (VILLAÇA; SILVEIRA, 2013).

Tendo em vista que, de um modo geral, os processos físicos são analógicos, grande parte das especificações, projetos e análise de comportamento dos processos, sistemas e plantas são realizadas em tempo contínuo. Entretanto, é notório que após a criação do transistor, as técnicas digitais vêm se tornando uma tendência nas áreas de automação e controle de processos. Este dispositivo consiste em um semicondutor de três camadas, composto principalmente de silício ou germânio, utilizado para diversas aplicações, tais como, amplificação e produção de sinais e em operações de chaveamento. Após sua inserção no mercado é possível observar a utilização cada vez mais frequente de equipamentos eletrônicos microprocessados (MOREIRA; SILVA, 2014).

Os controladores digitais são mais versáteis quando comparados com os analógicos, apresentando vantagens como maior flexibilidade na programação, menor custo, maior resistência à ruídos e distúrbios, maior confiabilidade e melhor sensibilidade (CAMPOS, 2006). Entretanto, a realização do projeto do controlador diretamente no domínio discreto pode ser uma tarefa bastante complicada. Tendo em vista que os processos reais normalmente são analógicos, as relações entre a representação matemática e a realidade física

são mais evidentes no tempo contínuo. Assim, a análise matemática no domínio discreto, em geral, é consideravelmente mais complexa.

Nesse cenário, surge a necessidade de realizar o processo de discretização, que possibilita a migração de um sistema de tempo contínuo para o tempo discreto. Em alguns casos procede-se a discretização apenas do controlador, mantendo a planta e outras partes do processo no tempo contínuo. Porém, quando deseja-se efetuar simulações computacionais antes da implementação efetiva do sistema real é mais vantajoso e, até mesmo necessário, realizar a simulação com a representação da planta discretizada (SOARES, 1996). Existem diversas técnicas que permitem obter esta representação discreta de um sistema contínuo, neste trabalho o método adotado é o da transformação bilinear, que apresenta como vantagem o fato de não causar distorções por sobreposição de espectro (aliasing) na resposta de um sistema discretizado (MOREIRA; SILVA, 2014).

Assim, a julgar pela importância da teoria de controle, bem como pelas vantagens advindas das técnicas de controle digital e, consequentemente, pela necessidade de migração de sistemas contínuos para o tempo discreto, este trabalho utiliza o ambiente de desenvolvimento Microsoft Visual Studio e suas ferramentas a fim de desenvolver um software que possibilite ao usuário a análise da resposta de um sistema no tempo discreto, fornecendo como entradas dados no tempo contínuo.

Este trabalho encontra-se dividido em cinco capítulos, incluindo o atual. No capítulo 2 é realizada uma revisão bibliográfica, de modo a apresentar alguns conceitos necessários para a compreensão do trabalho realizado. O capítulo 3 descreve a metodologia utilizada a fim de alcançar com êxito os objetivos propostos, isto é, desenvolver um aplicativo iterativo e útil a estudantes de graduação e outros usuários que desejem um suporte no projeto de controladores digitais. No capítulo 4 é realizada uma discussão acerca dos resultados obtidos. Por fim, o capítulo 5 apresenta as considerações finais deste trabalho.

1.1 Objetivo Geral

Desenvolver um software, através do ambiente de desenvolvimento da Microsoft Visual Studio, que realize a discretização de sistemas, empregando o método da transformação bilinear.

1.2 Objetivos Específicos

- Compreender a importância e as vantagens advindas da utilização de controladores digitais em relação aos analógicos;

- Empregar os conceitos de programação orientada a objetos como suporte para o desenvolvimento de um software no Microsoft Visual Studio;
- Estudar o comportamento de diferentes processos após a discretização, através do software desenvolvido;
- Verificar a eficiência da Transformação Bilinear como método de discretização de sistemas de controle.

2 Referencial Teórico

Para realização do trabalho proposto é essencial recorrer a literatura a fim de revisar e/ou definir alguns conceitos que serão empregados durante o seu desenvolvimento, bem como para verificar o resultado apresentado por outros trabalhos correlatos. (LEANDRO, 2004) realiza uma abordagem semelhante àquela aqui apresentada, desenvolvendo um aplicativo que permite simular a discretização de controladores analógicos, porém empregando distintos métodos de discretização, a validação do seu trabalho é realizada através de estudos de caso que permitem comprovar que os resultados apresentados pelo software desenvolvido são satisfatórios.

2.1 Sistemas de controle

Um sistema de controle consiste em um conjunto de componentes físicos, cuja associação permite gerenciar a si mesmo e a outros sistemas. A relação entre a entrada e a saída do mesmo pode ser descrita matematicamente por meio de equações diferenciais e funções de transferência, por exemplo. Assim, a questão básica de controle é sintetizar um sinal de entrada, de modo que a saída apresente um comportamento desejado (NISE; SILVA, 2002).

Nesse âmbito, é importante diferenciar sistemas de controle de malha aberta e malha fechada. O primeiro, ilustrado pela Figura 2, se refere a sistemas nos quais a saída (y) não apresenta um efeito sobre a ação de controle (u). Portanto, não há correção do sinal atuante através da utilização do valor obtido pela medição da saída, de modo que a entrada do controlador é composta apenas pelo valor de referência (r).

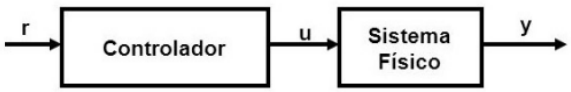


Figura 2 – Sistema de controle de malha aberta.

Fonte: (AFONSO, 2009).

Um sistema de controle de malha fechada, por sua vez, diz respeito a sistemas nos quais a diferença resultante da comparação entre a entrada de referência e a saída é considerada para geração do sinal de controle, isto é possível através de uma realimentação da saída para a entrada, conforme ilustrado na Figura 3 (OLIVEIRA et al.,).

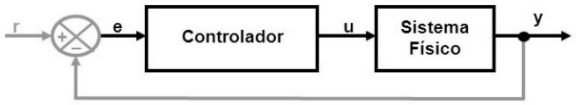


Figura 3 – Sistema de controle de malha fechada.

Fonte: (AFONSO, 2009).

O emprego da realimentação resulta em vantagens significativas tais como maior precisão, minimização do efeito de perturbações externas, além da diminuição de sensibilidade do sistema a variações dos parâmetros do processo. Por outro lado, ela pode resultar em menor estabilidade. Desse modo, a escolha correta do controlador a ser empregado no projeto é determinante para um bom desempenho do sistema (NISE; SILVA, 2002).

2.2 Controlador PID

O controlador Proporcional Integral Derivativo (PID) é atualmente o mais empregado industrialmente, fato este que é uma consequência do seu desempenho robusto e da diversidade de condições de funcionamento que o mesmo proporciona. Ele resulta da combinação das ações proporcional, integral e derivativa (Figura 4) , cujos parâmetros devem ser sintonizados a fim de obter a resposta desejada do sistema (FACCIN, 2004).

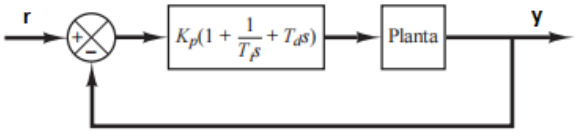


Figura 4 – Controle PID de uma planta.

Fonte: (OGATA, 2010).

A ação proporcional (P), conforme o nome sugere, varia proporcionalmente em relação ao erro resultante da diferença entre a saída (y) e a entrada de referência (r). A ação integral (I), por sua vez, é a responsável pela eliminação do offset (erro em regime permanente), assim está diretamente relacionada com a precisão do sistema de controle, porém ela pode provocar instabilidade. Por fim, a ação derivativa (D) é caracterizada pelo efeito antecipatório, de modo a aumentar a estabilidade do sistema (MOREIRA; SILVA, 2014).

Matematicamente, o controlador PID pode ser expresso por:

$$g_{PID}(t) = K_P + K_I \times \int e(t)dt + K_D \times \frac{de(t)}{dt}, \tag{2.1}$$

Aplicando a Transformada de Laplace, segue que:

$$G_{PID}(s) = K_P + K_I \times \frac{E(s)}{s} + K_D \times s \times E(s). \tag{2.2}$$

2.3 Transformação Bilinear

A Transformação Bilinear consiste em um mapeamento não linear entre o plano s e o plano z (figura 5). Este mapeamento é realizado, de modo que:

- O eixo imaginário do plano s é mapeado sobre a circunferência unitária no plano z.
- O semiplano lateral esquerdo do plano s é mapeado dentro do círculo unitário no plano z.
- O semiplano lateral direito do plano s é mapeado fora do círculo unitário no plano z.

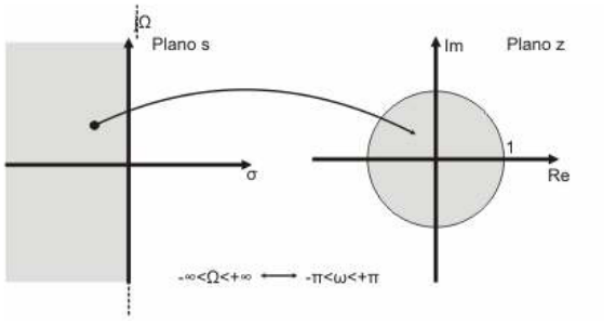


Figura 5 – Mapeamento do plano s no plano z.

Fonte: (OLIVEIRA et al.,).

Esta técnica de discretização tem a propriedade de transformar uma função contínua estável em uma função discreta também estável. Além disso, o *aliasing* produzido pela mesma é pequeno o suficiente para ser desconsiderado (OLIVEIRA et al.,).

Considerando um sistema de controle em malha fechada no qual r(t) é a entrada, e(t) o erro, u(t) o sinal de controle e y(t) a saída do sistema, tem-se que R(s), E(s), U(s) e Y(s) são as transformadas de Laplace dos sinais temporais correspondentes. Assim, sendo G(s) a função de transferência do controlador e H(s) a função da planta, e fazendo:

$$G(s) = \tfrac{1}{s},$$

Então,

$$\frac{U(s)}{E(s)} = \frac{1}{s}.$$

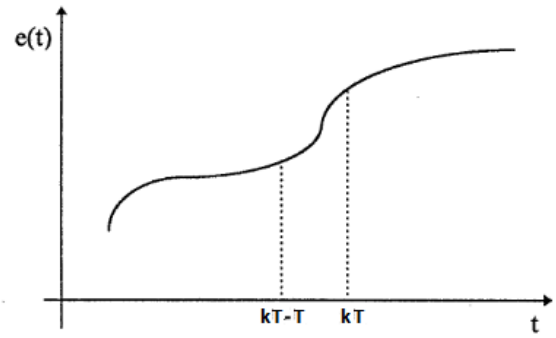


Figura 6 – Gráfico de e(t).

Fonte:(SOARES, 1996).

Posto isto, tomando a Figura 6, como o fator $\frac{1}{s}$ representa uma integração no tempo, obtém-se:

$$u(kT) = \int_0^{kT-T} e(t)dt + \int_{kT-T}^{kT} e(t)dt, \tag{2.3}$$

Isto corresponde a dizer que o valor de u(t) no instante t=kT é equivalente a área da função e(t) até o instante t=$kT - T$ mais a área entre os instantes t=$kT - T$ e t=kT, ou seja,

$$u(kT) = u(kT - T) + (\acute{a}rea\ entre\ kT - T\ e\ kT), \tag{2.4}$$

Aproximando o sinal e(t) entre os pontos t=$kT - T$ e t=kT por um segmento de reta (Figura 7) é possível reescrever o valor de $u(kT)$.

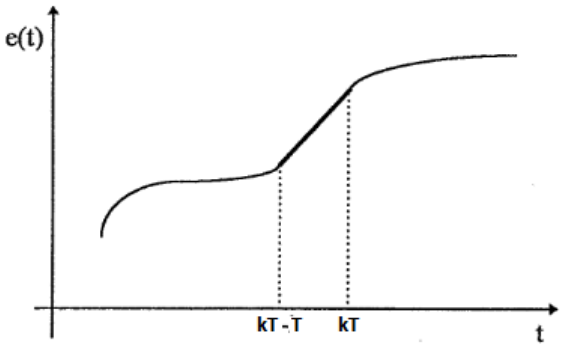


Figura 7 – Aproximação do intervalo entre $t = kT - T$ e $t = kT$ por um segmento de reta.

Fonte:(SOARES, 1996).

Assim,

$$u(kT) = u(kT - T) + \frac{T}{2}(e(kT - T) + e(kT)), \tag{2.5}$$

Nota-se que o segundo termo da equação (2.5) corresponde a área do trapézio formado entre t=$kT - T$ e t=kT. Desse modo, aplicando a Transformada Z à equação anterior e tendo em vista que

$$\mathcal{Z}\{f(t - kT)\} = z^{-k}F(z), \tag{2.6}$$

com k inteiro

Obtém-se

$$U(z) = z^{-1}U(z) + \frac{T}{2}(z^{-1}E(z) + E(z)), \tag{2.7}$$

$$U(z)(1 - z^{-1}) = \frac{T}{2}(z^{-1} + 1)E(z), \tag{2.8}$$

Realizando algumas manipulações matemáticas, chega-se a relação entre as transformadas Z dos sinais e(t) e u(t), dada por

$$\frac{U(z)}{E(z)} = \frac{T}{2}\frac{z^{-1} + 1}{1 - z^{-1}}, \tag{2.9}$$

Portando, de acordo com a Transformação Bilinear, a relação entre as variáveis s e z é dada por:

$$s = \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}}, \tag{2.10}$$

onde, T é o período de amostragem.

Assim, sendo conhecida a função de transferência no tempo contínuo, é possível obter sua correspondente no tempo discreto, uma vez que,

$$G(z) = G(s)|_{s=\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}}, \tag{2.11}$$

2.4 Programação orientada a objetos (POO)

A POO é um paradigma de programação que surgiu alternativamente a programação estruturada. Sua criação visa uma aproximação das estruturas de um programa

com o mundo real. Todos os conceitos da programação orientada a objetos são construídos a partir de dois conceitos chaves: classes e objetos. As classes referem-se a um conjunto de objetos que compartilham características semelhantes. Uma classe determina o comportamento de seus objetos através de métodos, e os possíveis estados através de atributos. A criação de um objeto torna a abstração da classe algo concreto e, a esse procedimento dá-se o nome de instanciação da classe. Desse modo, extensas linhas de códigos são divididas em vários objetos, permitindo que o desenvolvedor pense um objeto por vez, facilitando o desenvolvimento de softwares e tornando mais fácil a reutilização de códigos (GONÇALVES, 2018).

Além dos conceitos base definidos anteriormente, é válido destacar as principais características desse paradigma de programação, sendo elas, abstração, encapsulamento, herança e polimorfismo. A abstração implica em ocultar detalhes desnecessários ao cliente. O encapsulamento, por sua vez, diz respeito ao fato de um grupo de propriedades e métodos serem abordados como um único objeto. Já a herança, refere-se à capacidade de criar novas classes a partir de uma classe já existente. Por fim, o polimorfismo determina que várias classes podem ser utilizadas de modo intercambiável, mesmo que cada uma delas implemente de forma distinta as mesmas propriedades ou métodos (LEANDRO, 2004).

A linguagem C#, utilizada para o desenvolvimento desse trabalho foi criada pela Microsoft como parte da plataforma .NET. Ela apresenta similaridades com outras linguagens, tais como JAVA e C++, acrescida de uma melhor implementação e novos recursos. Além disso, fornece suporte completo para a programação orientada a objeto, incluindo abstração, encapsulamento, herança e polimorfismo (MICROSOFT, 2020).

3 Metodologia

A seguir encontram-se detalhadas as etapas sucedidas para o desenvolvimento deste trabalho.

3.1 Modelagem matemática

Tendo em vista, o que foi apresentado em relação a Transformação Bilinear no capítulo anterior e adotando o sistema de controle de malha fechada indicado na Figura 8, é possível realizar manipulações matemáticas a fim de obter o equivalente no tempo discreto das funções de transferência representadas.

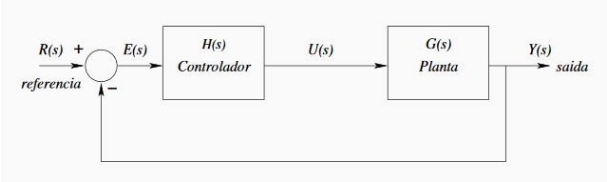


Figura 8 – Sistema de controle.

Fonte: (MARUYAMA, 2017).

3.1.1 Sistemas de primeira ordem

Seja um sistema de primeira ordem, cuja função de transferência é dada por

$$G(s) = \frac{Y(s)}{U(s)} = \frac{A}{Bs + C}. \tag{3.1}$$

Assim, através da substituição da variável s, conforme determina o método da Transformação Bilinear, tem-se que

$$G(z) = \frac{A}{B\left(\frac{2}{T}\frac{z-1}{z+1}\right) + C}. \tag{3.2}$$

Após algumas manipulações matemáticas, obtém-se:

$$\frac{Y(s)}{U(s)} = \frac{AT(z+1)}{z(2B + CT) + CT - 2B}. \tag{3.3}$$

Segundo a propriedade de deslocamento temporal da Transformada Z, sendo $g[k]$ um sinal discreto, tem-se que:

$$z^{-k0}G(z) = \mathcal{Z}\left\{g[k-k0]\right\}. \tag{3.4}$$

Portanto, aplicando a propriedade descrita na Equação (3.4) na Equação (3.3) e isolando o valor da saída atual do sistema, o valor $y[k]$ pode ser calculado por

$$y[k] = \frac{AT(u[k]+u[k-1]) + (2B - CT)y[k-1]}{2B + CT}. \tag{3.5}$$

Desse modo, é possível obter uma equação de diferença que represente a saída do sistema em tempo discreto.

3.1.2 Sistemas de segunda ordem

No caso de uma planta de segunda ordem, a função de transferência pode ser indicada por

$$G(s) = \frac{Y(s)}{U(s)} = \frac{As + B}{Cs^2 + Ds + E}. \tag{3.6}$$

Assim, realizando a substituição da variável s, de acordo com a Transformação Bilinear, obtém-se

$$G(z) = \frac{A\left(\frac{2}{T}\frac{z-1}{z+1}\right) + B}{C\left(\frac{2}{T}\frac{z-1}{z+1}\right)^2 + D\left(\frac{2}{T}\frac{z-1}{z+1}\right) + E}. \tag{3.7}$$

De modo que, aplicando a propriedade de deslocamento no tempo, indicada na Equação (3.4), a saída correspondente em tempo discreto é determinada através de:

$$\begin{aligned} y[k] &= \frac{(2AT + BT^2)u[k] + 2BT^2u[k-1] + (BT^2 - 2AT)u[k-2]}{4C + 2DT + ET^2} \\ &+ \frac{-(2ET^2 - 8C)y[k-1] - (4C - 2DT + ET^2)y[k-2]}{4C + 2DT + ET^2}. \end{aligned} \tag{3.8}$$

3.1.3 Controlador PID

Conforme abordado anteriormente, um controlador PID é dado por

$$H(s) = \frac{U(s)}{E(s)} = K_P\left(1 + \frac{1}{T_I s} + T_D s\right) = K_P + \frac{K_I}{s} + K_D s. \tag{3.9}$$

Consequentemente, a substituição de acordo com a Transformação Bilinear, resulta em

$$H(z) = \frac{K_P\left(\frac{2}{T}\frac{z-1}{z+1}\right) + K_I + K_D\left(\frac{2}{T}\frac{z-1}{z+1}\right)^2}{\left(\frac{2}{T}\frac{z-1}{z+1}\right)}. \tag{3.10}$$

Portanto, aplicando novamente a propriedade de deslocamento temporal, obtém-se

$$\begin{aligned} u[k] &= \frac{(2TK_P + K_IT^2 + 4K_D)e[k] + (2K_IT^2 - 8K_D)e[k-1]}{2T} \\ &+ \frac{(K_IT^2 + 4K_D - 2TK_P)e[k-2] + 2Tu[k-2]}{2T}. \end{aligned} \tag{3.11}$$

3.1.4 Realimentação

Nesse trabalho, considerou-se duas possibilidades de realimentação para o sistema de controle, sendo elas um valor constante k ou uma função de transferência de primeira ordem, cuja modelagem matemática é idêntica àquela apresentada na subseção 3.1.1.

3.2 Interface com o usuário

Para a construção da interface com o usuário utilizou-se o modelo do Aplicativo de formulários do Windows (.NET Framework) para C# no Visual Studio 2019.

A fim de obter as funcionalidades desejadas para o software, tornou-se necessária a utilização de três formulários. O primeiro é referente a janela inicial, na qual o usuário informará os dados de entrada (Figura 9). Para isso, através da caixa de ferramentas disponível, os seguintes elementos foram introduzidos ao projeto: *radioButton*, *textBox*, *folderBrowserDialog*,*GroupBox*, *buttons*, *pictureBox*, *timer*, *vScrollBar* e *labels*. Para ilustrar a malha de controle, adicionou-se uma imagem com um modelo padrão ao formulário através da *pictureBox* e utilizou-se recursos de sobreposição para organizar os demais elementos sobre a mesma.

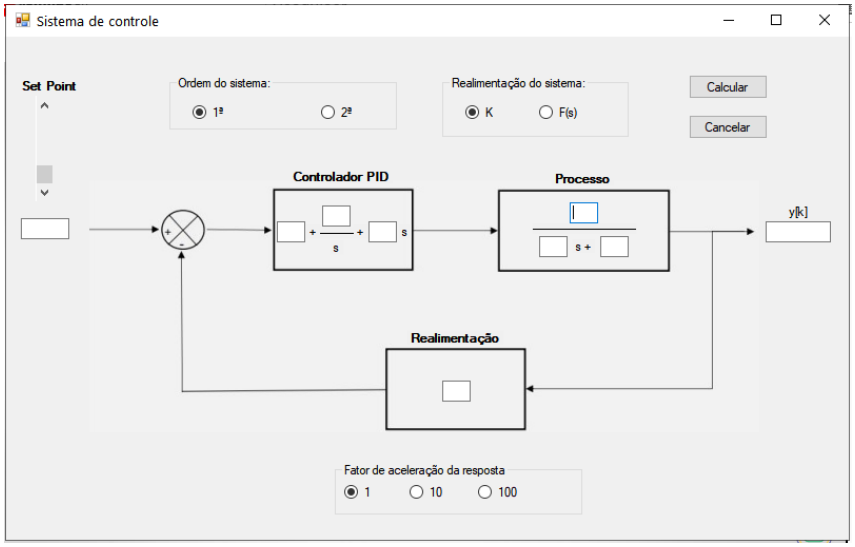


Figura 9 – Formulário 1.

Fonte:Próprio Autor.

O segundo formulário, representado na Figura 10, destinou-se a exibição do gráfico, nesse caso o elemento introduzido foi o *zedGraphControl*. Para ajustar o gráfico ao formulário, permitindo que ele também altere seu tamanho de acordo com a maximização e minimização da janela foi necessário alterar a propriedade *Anchor*, de modo a ancorar o controle com todas as bordas do recipiente.

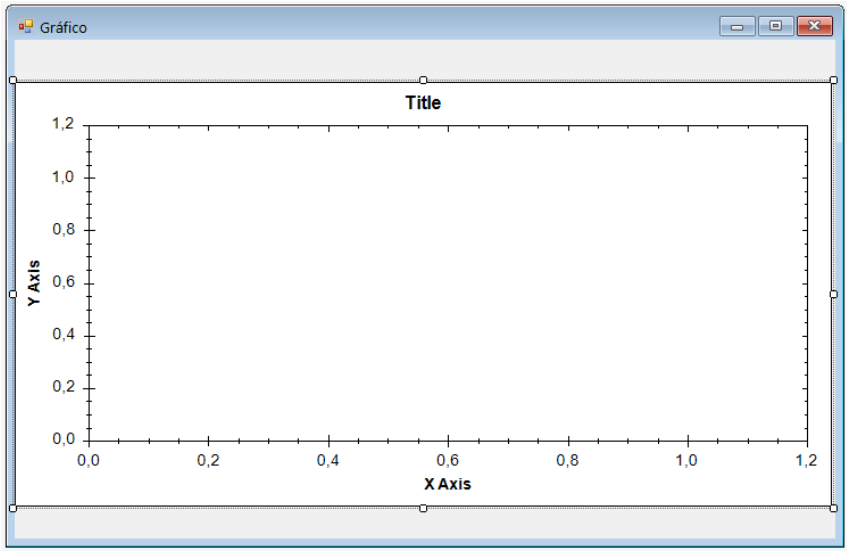


Figura 10 – Formulário 2.

Fonte:Próprio Autor.

Por fim, no terceiro formulário foram inseridos os elementos *label*, *textBox* e *button*, de modo a representar uma caixa de diálogo na qual o usuário poderá informar um nome

para o arquivo, conforme ilustrado pela Figura 11. Tendo em vista, a necessidade de compartilhar informações entre os formulários, alguns elementos tiveram a propriedade *Modifiers* alterada para *public*.

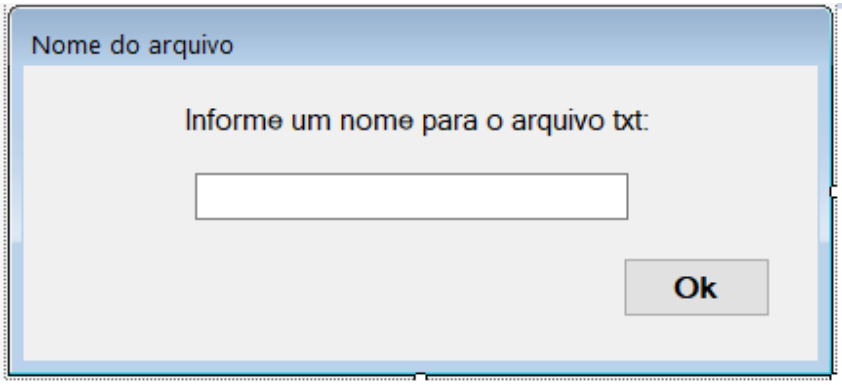


Figura 11 – Formulário 3.

Fonte: Próprio Autor.

3.3 Estruturação do código

Baseando-se nos conceitos relacionados a programação orientada a objetos, cada formulário corresponde a uma classe, dentro da qual são definidos métodos responsáveis pela manipulação de eventos. Os eventos são gerados quando o usuário executa alguma ação no formulário ou em um de seus controles (elementos discretos de interface com o usuário que exibe ou aceita informações).

3.3.1 Classe Form1

No que se refere a classe *Form 1*, para tornar possível a troca de informações com os outros formulários, foi necessário utilizar o método construtor, permitindo instanciar essa classe, inicializando os formulários dois e três, conforme as linhas de código abaixo:

```
form2 form2 = new form2();
Form3 form3 = new Form3();
```

Os eventos para os quais foram determinados métodos manipuladores foram o *CheckedChanged* do *radioButton*, o *Tick* do *timer*, o *Load* do *form*, o *Click* dos *buttons*, o *Scroll* do *vScrollBar* e o *Leave* do *textBox*.

O *CheckedChanged* ocorre toda vez que a propriedade *Checked*, responsável por indicar se um botão está ou não selecionado, é alterada. Através da manipulação deste evento é possível realizar um controle de visibilidade entre os elementos adicionados ao

formulário. Desse modo, as fórmulas da função de transferência do processo, bem como da realimentação exibidas ao usuário serão alternadas de acordo com a opção selecionada pelo mesmo. Essa estratégia permite a criação de um design mais harmonioso e iterativo. Nas linhas de código apresentadas abaixo é possível observar um exemplo, que resultará na exibição de determinados *textBoxs* e *labels* e ocultação de outros quando o botão 1 for selecionado pelo usuário.

```
private void radioButton1_CheckedChanged(object sender, EventArgs e)
{
    label3.Visible = true;
    textBox1.Visible = true;
    textBox2.Visible = true;
    textBox3.Visible = true;
    pictureBox5.Visible = true;
    label6.Visible = false;
    label8.Visible = false;
    label9.Visible = false;
    textBox6.Visible = false;
    textBox7.Visible = false;
    textBox8.Visible = false;
    textBox9.Visible = false;
    textBox10.Visible = false;
    pictureBox2.Visible = false;

}
```

Após o temporizador ser iniciado, a cada intervalo de tempo definido, neste caso 100 ms, é gerado o evento *Tick*. No manipulador do evento em questão, encontra-se desenvolvido todo código referente aos cálculos necessários para discretização pelo método da Transformação Bilinear, obtenção da resposta do sistema em tempo discreto, bem como para tracejar o gráfico. Ao trabalhar com este evento todas as operações citadas são repetidas a cada *Tick* do *timer*, isso permite, por exemplo, que o usuário altere dados ao longo do processamento, obtendo uma resposta correspondente as atualizações no próximo *Tick*.

Pelo método de manipulação do evento *Load*, que ocorre toda vez que o usuário carrega o formulário, realizou-se o desenho dos divisores das funções e da malha de controle, além de nomear o gráfico, bem como seus eixos. Ao pressionar algum dos botões é gerado o evento *Click*. No manipulador do *Click* do botão calcular o temporizador é iniciado, são realizadas algumas configurações referentes ao gráfico, tais como habilitação do *zoom* e da barra de rolagem do *zedGraphControl* e é feita a verificação se o usuário deseja ou não gerar um arquivo de texto. Já no manipulador do *Click* do botão cancelar é determinada a interrupção do temporizador, de acordo com o trecho do código apresentado abaixo, encerrando, pois, toda a rotina referente aos cálculos e plotagem executadas na ocorrência do *Tick*.

```
private void button2_Click(object sender, EventArgs e)
{
    timer1.Stop();
}
```

As rotinas desenvolvidas nos manipuladores dos eventos *Scroll*, que ocorre quando o usuário move a caixa de rolagem, e *Leave*, gerado quando o controle do *textBox* deixa de ser o controle ativo, são responsáveis pela vinculação entre o valor de *Setpoint* exibido na caixa de texto e aquele correspondente a barra de rolagem. Além disso, vale ressaltar que há tratamento de exceções no código através dos blocos de funções *try* e *catch*, informando ao usuário a ocorrência de um erro devido ao formato incorreto das entradas ou a ausência delas. As linhas a seguir ilustram a rotina executada quando o usuário retira o controle ativo do *textBox 4* (ocorrência do evento *Leave*), assim, caso tenha sido informado um valor válido, a posição da barra de rolagem será alterada de acordo com o valor informado na caixa de texto, caso contrário, será exibida uma mensagem solicitando que o usuário forneça uma entrada antes de passar para o próximo campo.

```
private void textBox4_Leave(object sender, EventArgs e)
{
    try
    {
        //Relaciona a posição da barra de rolagem com o valor informado pelo usuário
        vScrollBar1.Value = 100 - Convert.ToInt32(textBox4.Text);
    }
    catch
    {
        MessageBox.Show("Informe a entrada do processo (U(s))!");
    }
}
```

3.3.2 Classe Form2

A classe *Form 2* foi criada exclusivamente para exibição do gráfico, consequentemente foi necessário um número consideravelmente inferior de métodos manipuladores de eventos em comparação com aqueles utilizados no formulário 1. Mais especificamente foram trabalhados o evento *ZoomEvent* do *zedGraphControl*, bem como o *FormClosing* do *form*. Pelo controle do *ZoomEvent* é habilitada a barra de rolagem, assim é possível uma análise mais detalhada do gráfico exibido pelo usuário. Já no método manipulador do *FormClosing*, que ocorre sempre que o usuário fecha o formulário, antes do seu fechamento, é utilizada uma estratégia que permite cancelar o seu fechamento e alterar a propriedade de visibilidade do *form*, dando a impressão que o mesmo foi fechado, porém evitando a geração de um erro caso o usuário gere um novo evento *Click* no botão calcular. Esta última estratégia empregada pode ser visualizada nas linhas de código abaixo.

```
private void Form2_FormClosing(object sender, FormClosingEventArgs e)
{
    //Cancela a operação de fechar o form e tira sua visibilidade para o usuário
    e.Cancel = true;
    ActiveForm.Visible = false;
}
```

3.3.3 Classe Form3

Na classe *Form 3*, são declaradas duas propriedades e especificado o nível de acesso do campo, isto possibilita que a *string* referente ao nome do arquivo definido pelo usuário, bem como o valor booleano indicando a ocorrência do evento *Click* do botão do formulário 3 sejam utilizados pela classe *form1*. O procedimento de propriedade *GET* retorna o valor de uma propriedade, sendo utilizado quando trata-se de uma propriedade apenas de leitura. Abaixo é possível observar as linhas do código que desempenham esta funcionalidade.

```
public string Nome
{
    get { return X; }
}
2 referências
public bool p
{
    get { return b; }
}
```

3.4 Execução do programa

Ao iniciar o software, o usuário deverá informar os dados do sistema de controle contínuo que será discretizado utilizando o método da Transformação Bilinear. O valor do *SetPoint* poderá ser indicado tanto pela barra de rolagem quanto digitado na caixa de texto, conforme destacado na Figura 12.

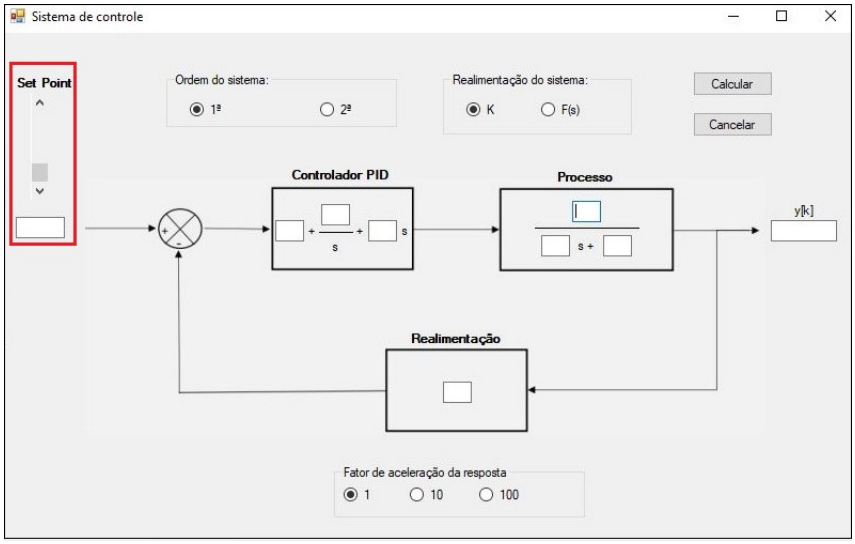


Figura 12 – Local para indicação do SetPoint.

Fonte:Próprio Autor.

Além disso, através dos botões de seleção, o usuário deverá indicar se a função de transferência do processo é de 1ª ordem (Figura 13) ou 2ª ordem (Figura 14). O tipo de realimentação também deverá ser selecionado, sendo possível escolher entre um valor constante (K), de acordo com a Figura 15, ou uma função de transferência (F(s)), conforme Figura 16. Assim, de acordo com a opção selecionada, serão exibidos os campos adequados na malha de controle para que o usuário informe os coeficientes das funções.

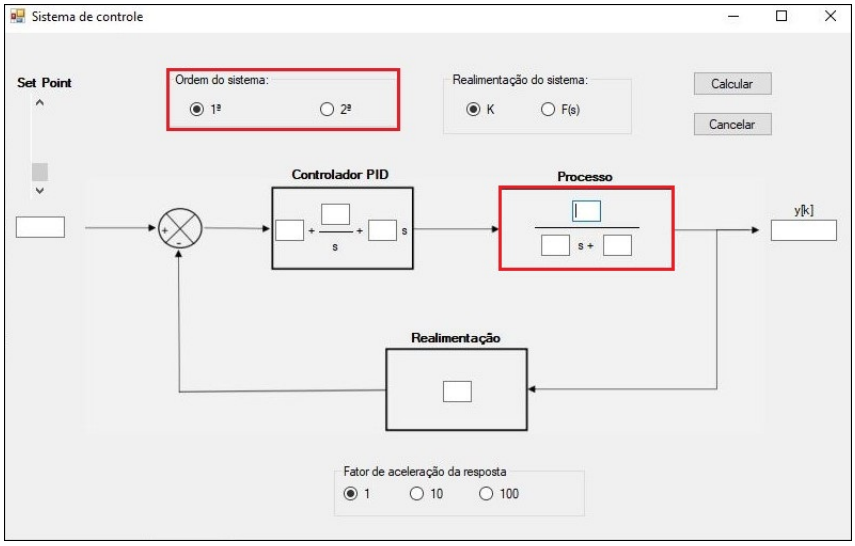


Figura 13 – Seleção da Função de Transferência de 1ª ordem.

Fonte:Próprio Autor.

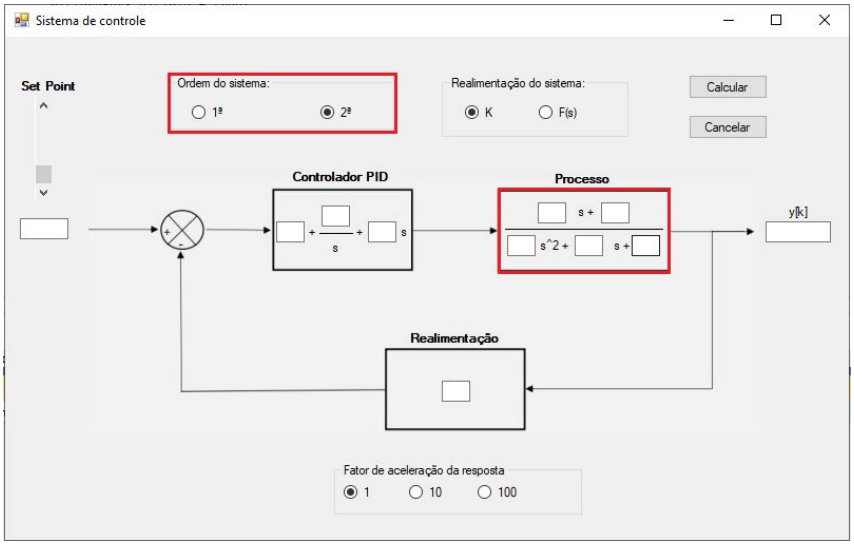


Figura 14 – Seleção da Função de Transferência de 2ª ordem.

Fonte:Próprio Autor.

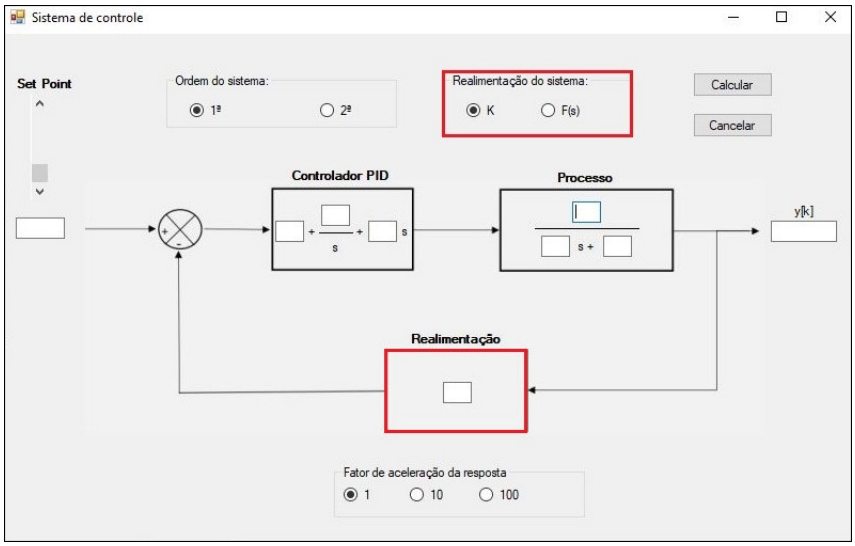


Figura 15 – Seleção da realimentação constante.

Fonte:Próprio Autor.

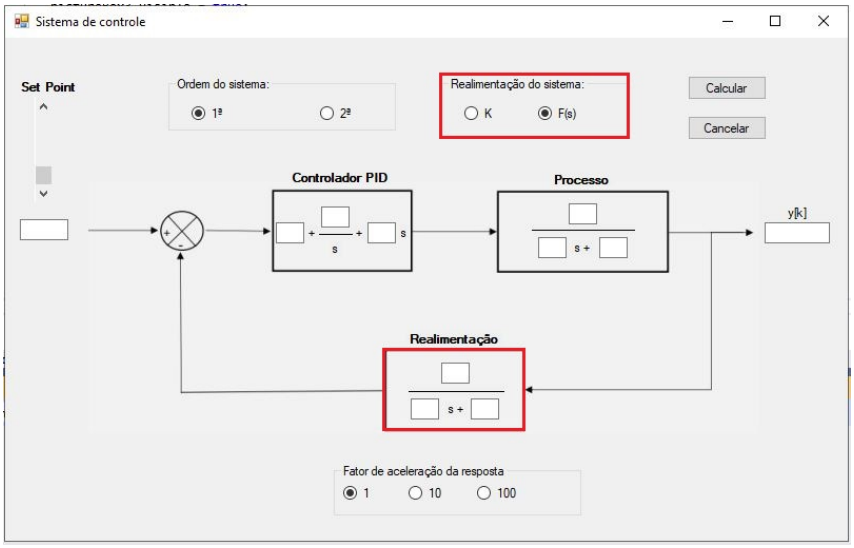


Figura 16 – Seleção da realimentação F(s).

Fonte:Próprio Autor.

Na área destacada na Figura 17 será possível alterar o fator de aceleração da resposta. Assim, considerando que o período de amostragem padrão definido no código é 100ms, modificando este fator, será possível reduzir este valor em 10 ou 100 vezes, aumentando, pois, a frequência de amostragem.

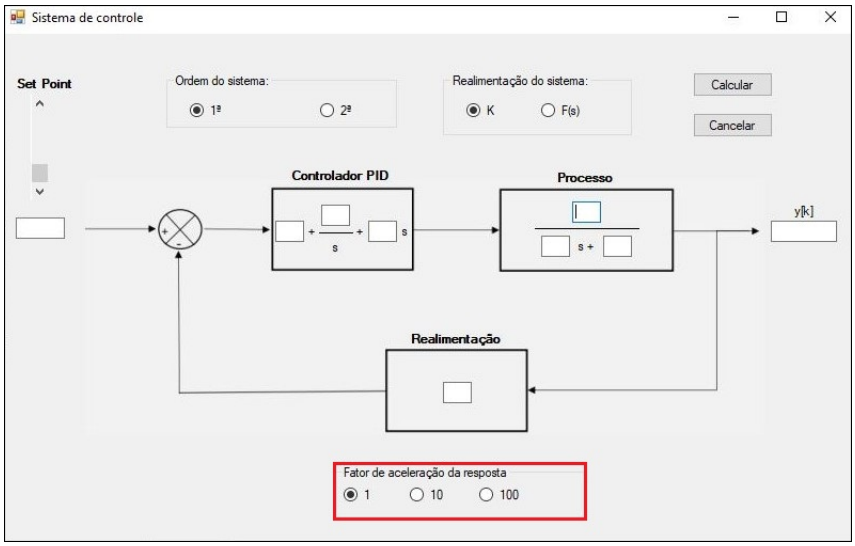


Figura 17 – Seleção do fator de aceleração.

Fonte:Próprio Autor.

Após fornecer os parâmetros de entrada necessários e pressionar o botão calcular (Figura 18), é fornecida ao usuário a opção de salvar os dados em formato *txt* (Figura 19), caso ele tenha esse interesse deverá selecionar a pasta na qual o novo arquivo será

armazenado (Figura 20) e nomeá-lo. Caso contrário, o programa salta essa etapa e segue para o cálculo do valor de saída.

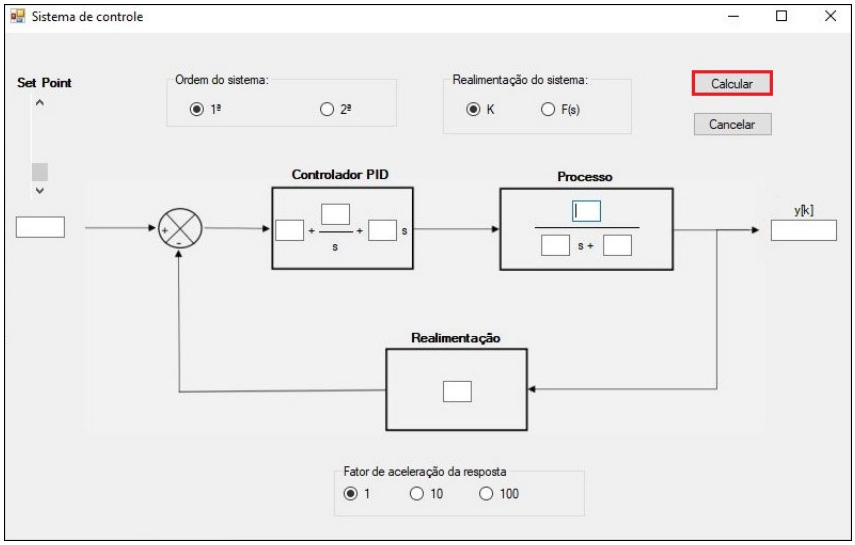


Figura 18 – Botão Calcular.

Fonte:Próprio Autor.

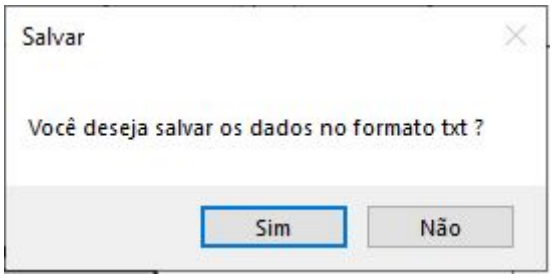


Figura 19 – Opção para salvar os dados em formato txt.

Fonte:Próprio Autor.

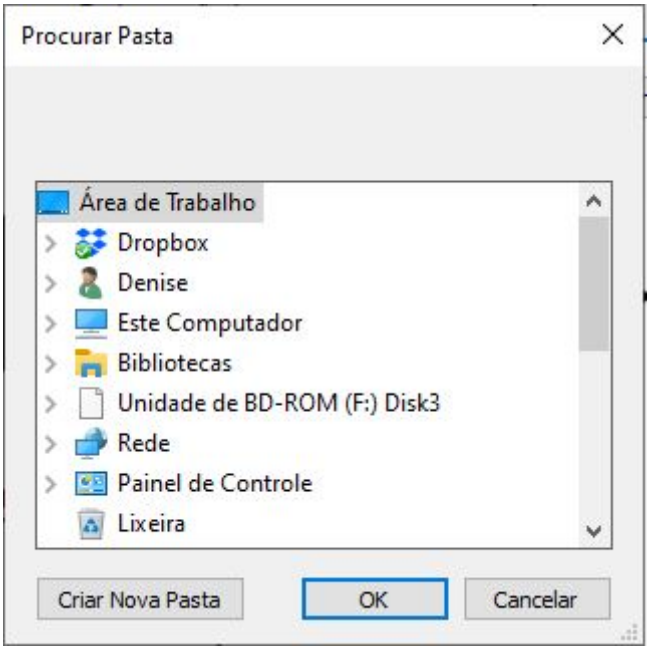


Figura 20 – Procurar pasta.

Fonte:Próprio Autor.

Se o usuário pressionar o botão calcular e alguma entrada estiver em formato incorreto ou ausente será exibido um aviso informando o mesmo sobre a ocorrência do erro (Figura 21).

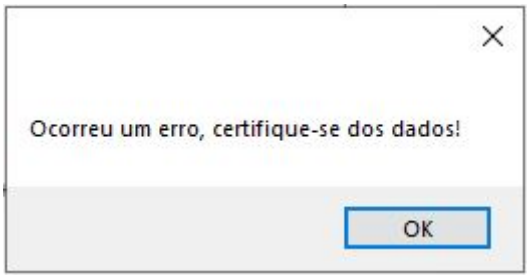


Figura 21 – Aviso de erro.

Fonte:Próprio Autor.

A resposta do sistema discretizado será exibida dinâmicamente na caixa destacada na Figura 22. Além disso, a representação gráfica, cuja exibição ocorre em uma janela a parte, também se dará maneira dinâmica.

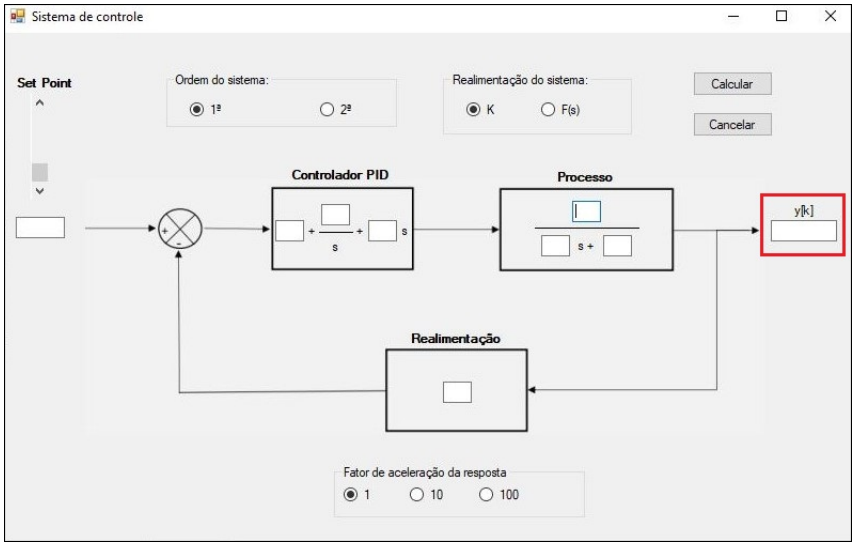


Figura 22 – Saída do Sistema.

Fonte:Próprio Autor.

Quando houver necessidade de interromper o processo, o usuário deverá pressionar o botão cancelar (Figura 23), nesse momento os cálculos são interrompidos e o gráfico gerado poderá ser analisado criteriosamente através do recurso de *zoom* e da barra de rolagem.

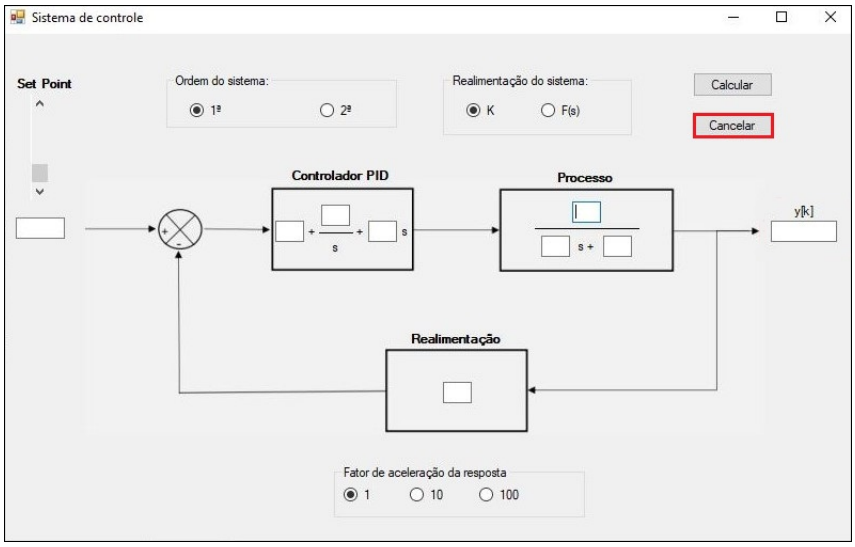


Figura 23 – Botão Cancelar.

Fonte:Próprio Autor.

4 Resultados

Após o desenvolvimento do software, foram analisados alguns exemplos de sistemas de controle com o objetivo de verificar o comportamento depois da discretização. É importante destacar que, conforme apresentado anteriormente, a Transformação Bilinear permite a discretização de um sistema contínuo de forma aproximada. Assim, para validar os resultados, utilizou-se o Matlab. Desse modo, os valores da resposta do sistema em relação a uma dada entrada, salvos no formato txt, foram utilizados a fim de comparar o comportamento do sistema discretizado com o comportamento apresentado pelo sistema no tempo contínuo.

Inicialmente considerou-se uma planta com função de transferência de primeira ordem, dada por

$$G(s) = \frac{1}{10\ s + 1}, \tag{4.1}$$

Empregou-se as constantes $k_p = 10$, $k_i = 1$ e $k_d = 2$ para o controlador PID e uma realimentação negativa e unitária.

O gráfico obtido pelo aplicativo desenvolvido (utilizando a biblioteca *Zedgraph*), para este sistema de controle, considerando um período de amostragem de 100ms (intervalo padrão do *Tick* do evento *timer*), encontra-se representado na Figura 24.

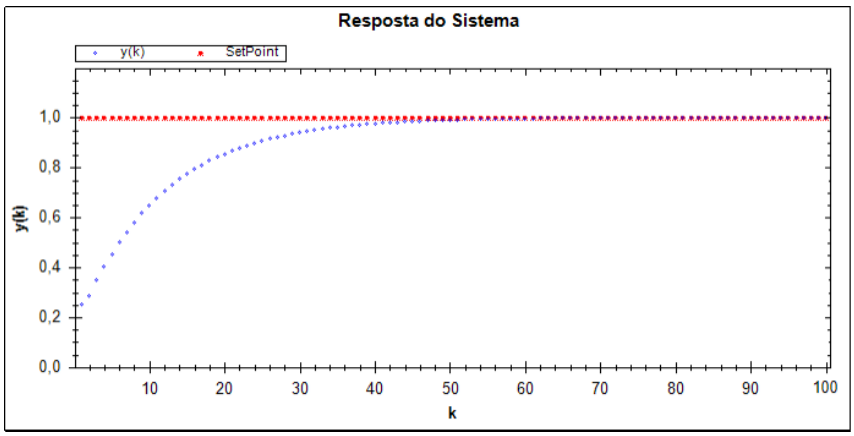


Figura 24 – Resposta discretizada de um Sistema de 1ª ordem com realimentação unitária (T=100ms).

Fonte:Próprio Autor.

É importante destacar que a escolha adequada do período de amostragem (T) é determinante para a obtenção de uma resposta satisfatória. Assim, através do Matlab

foram analisadas as respostas discretizadas considerando T=100ms e T=10ms, cujos gráficos podem ser analisados, respectivamente nas Figuras 25 e 26.

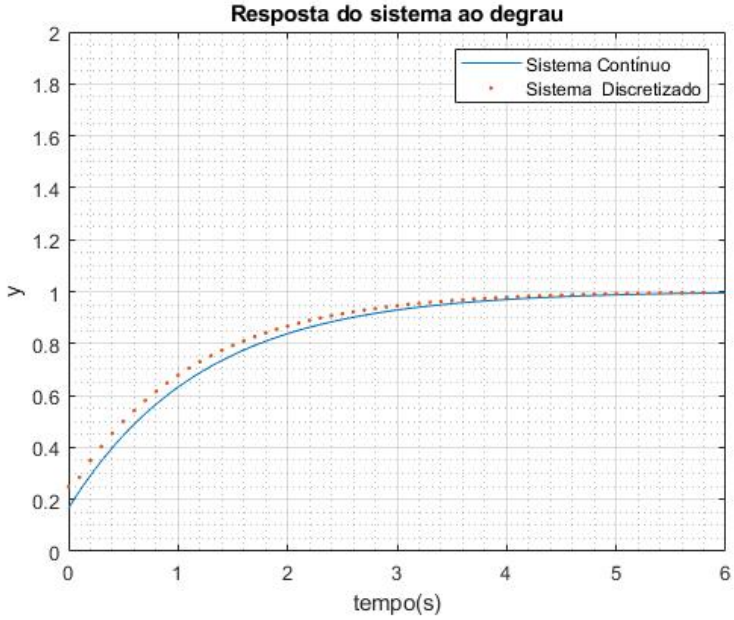


Figura 25 – Resposta ao degrau de um Sistema de 1ª ordem (T=100ms).

Fonte:Próprio Autor.

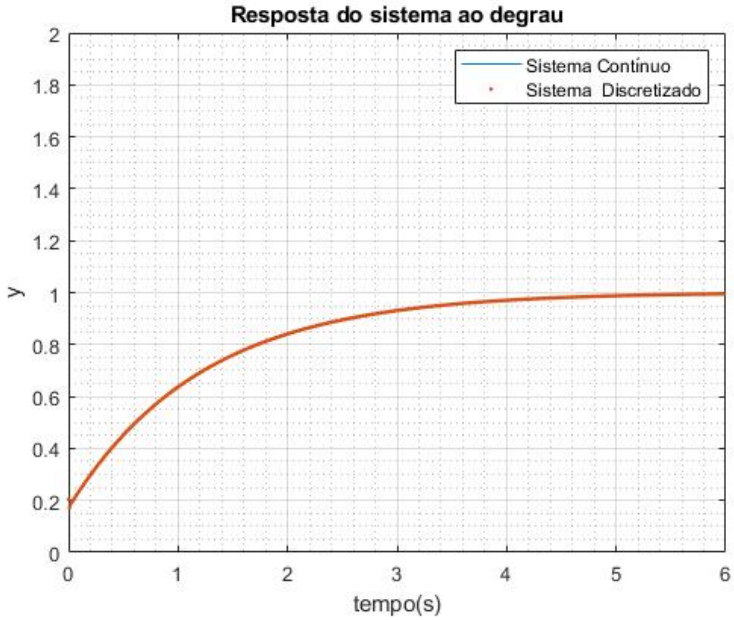


Figura 26 – Resposta ao degrau de um Sistema de 1ª ordem (T=10ms).

Fonte:Próprio Autor.

Portanto, analisando a resposta discretizada nas Figuras 25 e 26 e comparando com o comportamento do sistema contínuo, é possível observar que, quando T=100ms,

embora a resposta discretizada aproxime-se consideravelmente da resposta contínua, é visível a diferença em alguns instantes entre as curvas apresentadas. Na discretização com T=10ms, por sua vez, é evidente que o sistema discretizado consiste em uma representação mais confiável do sistema de controle analisado. Para o primeiro caso (T=100 ms), o erro médio quadrático entre os valores das 50 primeiras amostras em tempo discreto e o correspondente no tempo contínuo foi de 0.0011. Já no segundo caso (T=10 ms) esse valor foi reduzido para 5.9097×10^{-5}.

Alterando a função de transferência de realimentação para

$$F(s) = \frac{1}{s+1}, \tag{4.2}$$

e mantendo-se, a função de transferência da planta, bem como as constantes do controlador PID, a nova resposta do sistema a entrada unitária, para T=100ms, obtida pelo software desenvolvido pode ser observada na Figura 27.

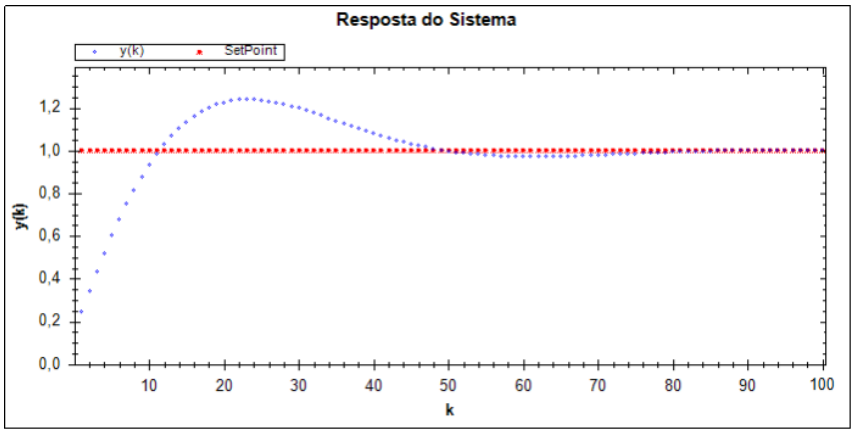


Figura 27 – Resposta discretizada de um Sistema de 1ª ordem (T=100ms).

Fonte:Próprio Autor.

A comparação entre as respostas do sistema discretizado e sistema contínuo através do Matlab, para T=100ms e T=10ms, podem ser observadas, respectivamente nas Figuras 28 e 29.

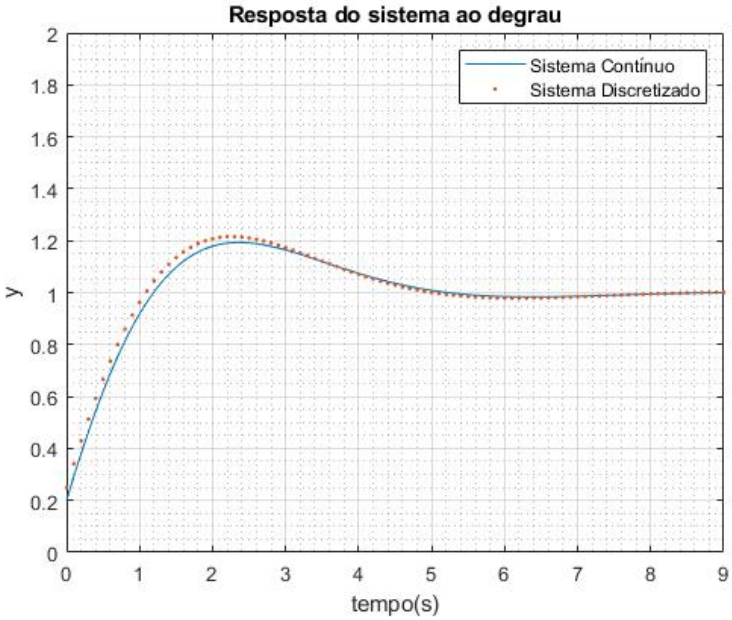


Figura 28 – Resposta ao degrau de um Sistema de 1ª ordem (T=100ms).

Fonte:Próprio Autor.

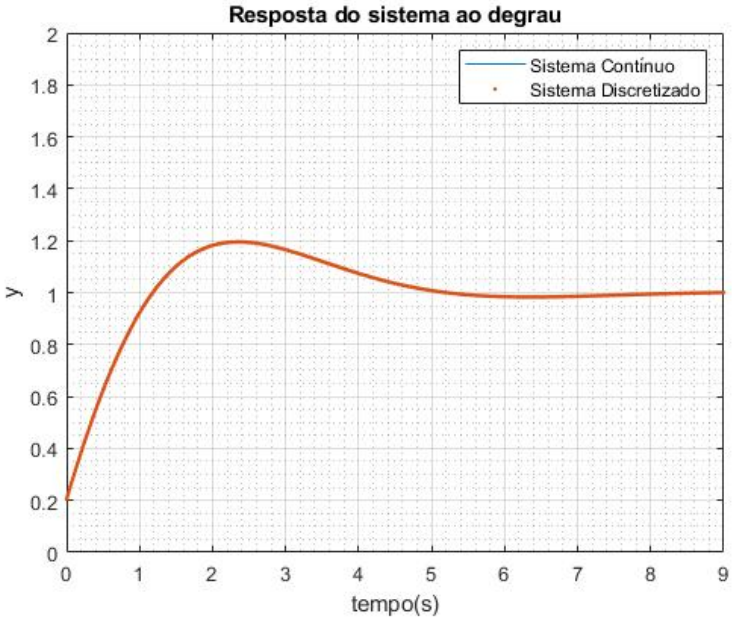


Figura 29 – Resposta ao degrau de um Sistema de 1ª ordem (T=10ms).

Fonte:Próprio Autor.

Assim como no exemplo anterior, observa-se que a redução do período de amostragem permite a obtenção de uma resposta discretizada mais próxima daquela obtida no sistema contínuo. O cálculo do erro médio quadrático entre os valores das 50 primeiras amostras em tempo discreto e o correspondente no tempo contínuo mostrou que para o

primeiro caso (T=100 ms), o erro foi de 0.0019. Enquanto no segundo caso (T=10 ms) esse valor foi reduzido para 3.0537×10^{-5}.

Por fim, analisou-se um sistema de 2ª ordem, cujo processo é representado pela seguinte função de transferência

$$G(s) = \frac{10}{s^2 + 3\ s + 2}, \tag{4.3}$$

Empregou-se o controlador PID com os parâmetros $k_p = 5$, $k_i = 2$ e $k_d = 1$ e realimentação unitária negativa. O gráfico obtido para T=100ms pode ser observado na Figura 30.

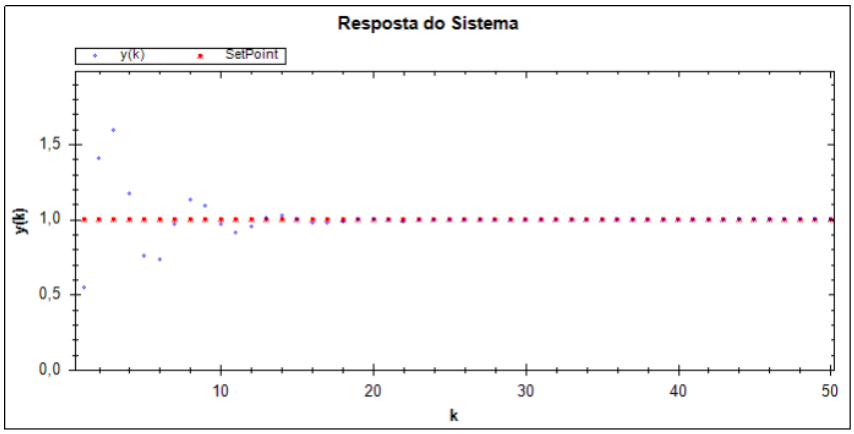


Figura 30 – Resposta discretizada de um Sistema de 2ª ordem (T=100ms).

Fonte:Próprio Autor.

A resposta do sistema ao degrau no tempo contínuo juntamente com a resposta discretizada pelo método da Transformada de Tustin, considerando os períodos de amostragem 100ms,10ms e 1ms, encontram-se representadas, respectivamente, nas Figuras 31, 32 e 33.

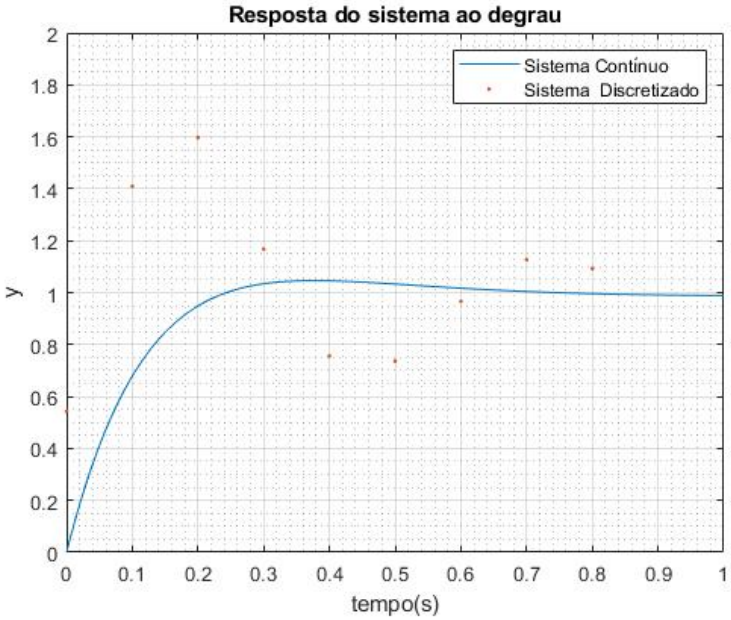


Figura 31 – Resposta ao degrau de um Sistema de 2ª ordem (T=100ms).

Fonte:Próprio Autor.

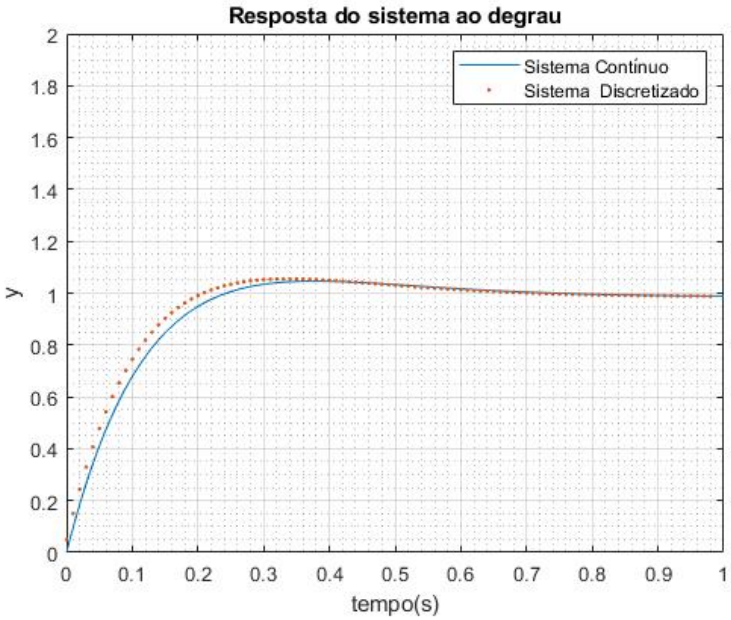


Figura 32 – Resposta ao degrau de um Sistema de 2ª ordem (T=10ms).

Fonte:Próprio Autor.

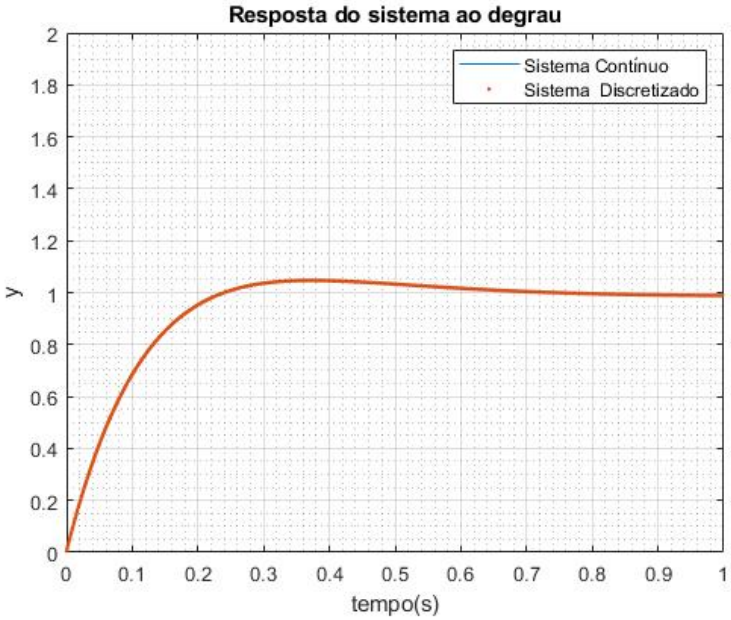


Figura 33 – Resposta ao degrau de um Sistema de 2ª ordem (T=1ms).

Fonte:Próprio Autor.

Neste exemplo, percebe-se que a discretização empregando um período de amostragem de T=100ms (Figura 31) difere consideravelmente da curva correspondente ao comportamento do sistema no tempo contínuo durante o regime transitório, apresentando um erro médio quadrático de 0.0291. A resposta obtida através da redução deste período por um fator de 10 (Figura 32) mostra uma melhora da resposta discretizada, neste caso o erro foi reduzido para 0.0020. Porém o melhor resultado pode ser observado no terceiro caso (Figura 32), utilizando um período de amostragem de 1ms, isto gera um erro de 3.6812×10^{-5}.

Com relação ao desempenho do programa durante sua execução, considerando um processador Intel Core i5-5200U, 2.20 GHz, 64 bits com 8.0 GB de memória RAM, o programa, na versão executável, apresentou no pior caso, um consumo de memória de 15300 KB e 33,6 % de utilização do processador.

5 Conclusão

O crescimento significativo do uso de técnicas digitais evidencia a necessidade de estudar soluções de discretização de sistemas contínuos para posterior implementação em computadores. A Transformação Bilinear, abordada nesse trabalho, consiste em uma técnica largamente empregada, este fato justifica-se pelos resultados satisfatório obtidos através do seu emprego.

O software desenvolvido permite que o usuário, conhecendo as informações da planta, bem como os parâmetros do controlador projetado no tempo contínuo, obtenha a resposta do sistema discretizado para uma entrada fornecida. Além disso, o aplicativo permite que o usuário salve os valores de saída do sistema discretizado, isto pode ser útil, por exemplo, para gerar curvas em outros aplicativos, contrastando com as informações do sistema no domínio contínuo. Além disso, pode ser importante para comparação entre o comportamento do sistema discretizado empregando-se distintos controladores.

Os resultados obtidos mostraram que, a Transformação Bilinear consiste em um método bastante eficiente de discretização, desde que seja selecionado um período de amostragem adequado, uma vez que este influencia diretamente no comportamento do sistema discretizado. O período de amostragem pode ser alterado no software através do campo de seleção denominado fator de aceleração da resposta.

Assim, este trabalho cumpriu com êxito os objetivos propostos, desenvolvendo um software iterativo que possibilita a discretização de sistemas de controle através da Transformação Bilinear, permitindo a implementação computacional de controladores inicialmente projetados no tempo contínuo.

Referências

AFONSO, A. P. **Introdução aos sistemas dinâmicos**. 2009. <https://slideplayer.com.br/slide/334154/>. [Online; acesso: 19 de Agosto de 2020].

CAMPOS, P. R. B. Análise de sistemas contínuos amostrados. Material de apoio à disciplina Controle Digital, lecionada no CEFET-PR, 2006.

FACCIN, F. Abordagem inovadora no projeto de controladores pid. Dissertação de mestrado. Universidade Federal do Rio Grande do Sul. Escola de Engenharia. Programa de Pós-Graduação em Engenharia Química, p.145, 2004.

GONÇALVES, M. V. Programação orientada a objetos. **Revista Ada Lovelace**, v. 2, p. 106–110, 2018.

LEANDRO, H. Ambiente digital para discretização de controladores analógicos. COBENGE, Brasília, 2004.

MARUYAMA, N. Introdução aos sistemas de controle. 2017.

MICROSOFT. **Documentação do C**. 2020. <https://docs.microsoft.com/pt-br/dotnet/csharp>. [Online; acesso: 13 de Agosto de 2020].

MOREIRA, C. N.; SILVA, F. S. da. Automação da discretização de controladores pid e filtros digitais. **Revista Ciência e Tecnologia**, v. 17, n. 31, 2014.

NISE, N. S.; SILVA, F. R. da. **Engenharia de sistemas de controle**. [S.l.]: LTC, 2002. v. 3.

OGATA, K. **Modern control engineering**. [S.l.]: Prentice hall, 2010.

OLIVEIRA, E. E. C. de et al. Projeto e análise do desempenho dos filtros iir por meio da técnica de invariância ao impulso e transformação bilinear. **V CONNEPI**, Maceió, Alagoas.

SOARES, P. M. O. d. R. Discretização de controladores contínuos. Dissertação de mestrado, Universidade do Porto. Porto, p.130, 1996.

VILLAÇA, M. V. M.; SILVEIRA, J. L. Uma breve história do controle automático. **Revista Ilha Digital**, v. 4, p. 3–12, 2013.