UNIVERSIDADE FEDERAL DE VIÇOSA
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

JUVENAL CABRAL PIRES

SIMULADOR DE RESPOSTA DO MOTOR CC USANDO WINDOWS FORMS NA LINGUAGEM C#

VIÇOSA
2023

JUVENAL CABRAL PIRES

SIMULADOR DE RESPOSTA DO MOTOR CC USANDO WINDOWS FORMS NA LINGUAGEM C#

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.

Orientador: Prof. Dr. André Gomes Torres.

VIÇOSA
2023

JUVENAL CABRAL PIRES

SIMULADOR DE RESPOSTA DO MOTOR CC USANDO WINDOWS FORMS NA LINGUAGEM C#

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 06 de Julho de 2023.

COMISSÃO EXAMINADORA

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

Prof. Dr. Denilson Eduardo Rodrigues - Membro
Universidade Federal de Viçosa

Prof. Dr. José Tarcísio de Resende - Membro
Universidade Federal de Viçosa

“A natureza não é benévola, e é com determinada indiferença
que de tudo se vale para os seus fins.”
(Lao-Tsé)

Agradeço a todos de maneira geral, pois cada pessoa é resultado de inúmeras contribuições e influências de outras, sozinhos pouco fazemos.

Agradecimentos

Agradeço de forma geral a todos que participaram e contribuíram de alguma forma na minha jornada até aqui. Cada pessoa deixa algo de si, seja algo bom ou ruim, que cause incômodo ou apreço, mas acima disso tudo e a despeito de qualquer mero julgamento deixam aquilo que podem deixar e dessa forma deixam conhecimento e crescimento pessoal. Seja como forma de inspiração ou como exemplo do que não reproduzir.

De forma particular e pertinente a esse trabalho agradeço enormemente ao Prof. André pela paciência e inspiração.

Resumo

Sabendo da importância do uso de máquinas elétricas tanto no dia-a-dia de pessoas e empresas, quanto para o estudo em engenharia, e dado também a importância de simulações para o aprendizado, esse trabalho se propõe a construir um simulador de máquinas CC. Simulador que será construído para trabalhar no ambiente windows, trabalhado na IDE Visual Studio, utilizando a linguagem de programação C#, com intuito de estudar os conceitos de máquinas elétricas, utilização do Visual studio e linguagem C# e por último contribuir com a aprendizagem de futuros usuários.

Palavras-chave: Microsoft Visual Studio, Windows Forms, C#, Curvas de resposta de motores CC.

Abstract

Knowing the importance of the use of electric machines in both the daily lives of individuals and businesses, as well as for engineering studies, and considering the significance of simulations for learning, this project aims to build a DC machine simulator. The simulator will be developed to work in the Windows environment, using the Visual Studio IDE and the C# programming language, with the purpose of studying electrical machine concepts, utilizing Visual Studio and the C# language, and ultimately contributing to the learning of future users.

Keywords: Microsoft Visual Studio, Windows Forms, C#, DC motor response curves.

Sumário

Lista de Figuras

1 Introdução do Trabalho

O uso de simuladores foi abrangido por várias áreas de trabalho e conhecimento a partir do ponto em que os sistemas computacionais se desenvolveram de maneira a propiciar o processamento de uma quantidade considerável de informação. A simulação de diversos sistemas passou a ser uma ferramenta essencial, seja por questão de segurança, acessibilidade, custo, maleabilidade, tempo, melhor controle e monitoramento do que se propõe a simular.

Uma atividade em que se faz extremamente necessário o uso de simulações é o ensino, principalmente de engenharia, visto que alguns sistemas de engenharia são caros, outros perigosos, e às vezes ambos. Especificamente na área de Engenharia Elétrica, no ramo de máquinas elétricas, a simulação além de proporcionar segurança e ausência da necessidade de se ter vários equipamentos de medição, ela contribui com informações que às vezes não são de fácil aquisição.

Dado a importância das máquinas de corrente contínua(máquinas CC) tanto para aplicações como objeto de estudo para engenharia elétrica, e visando propiciar o acesso a um simulador de fácil interação e pouca complexidade de manuseio para simulações de máquinas CC para uso no aprendizado desse assunto, este trabalho se propõe a construir um simulador de máquinas CC no formato Windows forms utilizando o ambiente de desenvolvimento Visual Studio da Microsoft™, utilizando a linguagem de programação C# que fornecerá as curvas de comportamento e resposta das máquinas CC propostas.

1.1 Conceitualização de programação

1.1.2 Algoritmo: Definições

Dentre várias definições podemos destacar as seguintes, a fim de delimitarmos o conceito:

Um algoritmo pode ser definido como uma sequência finita de passos (instruções) para resolver um determinado problema. Sempre que desenvolvemos um algoritmo estamos estabelecendo um padrão de comportamento que deverá ser seguido (uma norma de execução de ações) para alcançar o resultado de um problema. (INTRODUÇÃO A ALGORITMOS E PROGRAMAÇÃO FABRICIO FERRARI)

“Algoritmo é uma sequência de passos que visa atingir um objetivo bem definido” (FORBELLONE, 1999).

"Algoritmo é a descrição de uma sequência de passos que deve ser seguida para a realização de uma tarefa" (ASCENCIO, 1999).

"Algoritmo é uma sequência finita de instruções ou operações cuja execução, em tempo finito, resolve um problema computacional, qualquer que seja sua instância" (SALVETTI, 1999).

"Algoritmo são regras formais para a obtenção de um resultado ou da solução de um problema, englobando fórmulas de expressões aritméticas" (MANZANO, 1997).

"Ação é um acontecimento que, a partir de um estado inicial, após um período de tempo finito, produz um estado final previsível e bem-definido. Portanto, um algoritmo é a descrição de um conjunto de comandos que, obedecidos, resultam numa sucessão finita de ações" (FARRER, 1999)

"Um algoritmo é uma série ordenada de passos não-ambíguos, executáveis." (BROOKSHEAR, 2003, p.151)

"Algoritmo é um processo de cálculo matemático ou de resolução de um grupo de problemas semelhantes". (MANZANO e OLIVEIRA, 2000, p.6)

Algoritmo é uma sequência ordenada e finita de operações para a realização de uma tarefa qualquer (Ascênsio, 1999)

1.1.2 Exemplos de algoritmos

Alguns exemplos de algoritmos são:

Algoritmo 1: Troca de pneu do carro.

1: desligar o carro

2: pegar as ferramentas (chave e macaco)

3: pegar o estepe

4: suspender o carro com o macaco

5: desenroscar os 4 parafusos do pneu furado

6: colocar o estepe

7: enroscar os 4 parafusos

8: baixar o carro com o macaco

9: guardar as ferramentas

Algoritmo 2: Calcula Área de uma Circunferência.

1: $\pi \leftarrow 3.14$ {entrada para o processamento}

2: leia R {entrada para o processamento}

3: $A \leftarrow \pi * R2$ {processamento}

4: escreva A {saída}

1.1.3 Partes Constituintes e Estrutura de Um Algoritmo

Um algoritmo quando programado num computador é constituído pelo menos das 3 partes, sendo elas: 1. Entrada de dados; 2. Processamento de dados; 3. Saída de dados; Na parte de entrada, são fornecidas as informações necessárias para que o algoritmo possa ser executado. Estas informações podem ser fornecidas no momento em que o programa está sendo executado ou podem estar embutidas dentro do mesmo. Na parte do processamento são avaliadas todas as expressões algébricas, relacionais e lógicas, assim como todas as estruturas de controle existentes no algoritmo (condição e/ou repetição). Na parte de saída, todos os resultados do processamento (ou parte deles) são enviados para um ou mais dispositivos de saída, como: monitor, impressora, ou até mesmo a própria memória do computador.

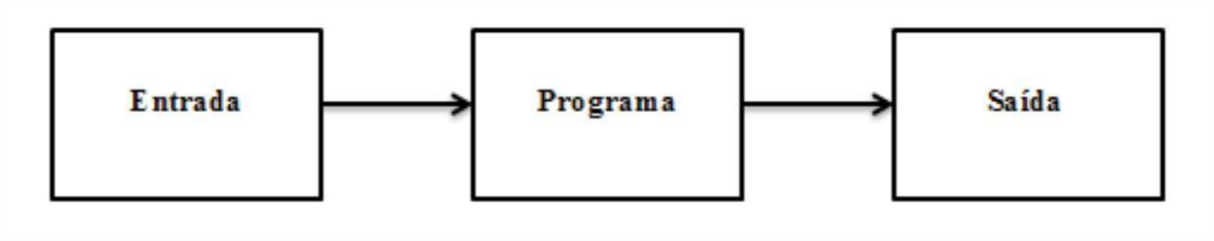


Figura 1 - Bloco de Algoritmo (Fonte: Autor).

1.1.4 Software

Um software, normalmente, é composto por diversas funções, bibliotecas e módulos que geram um programa executável ao final do processo de desenvolvimento e este que, quando executado, recebe algum tipo de “entrada” de dados (input), processa as informações segundo uma série de algoritmos ou sequências de instruções lógicas e libera uma saída (output), como

resultado deste processamento. Um software bem desenvolvido é, normalmente, criado pela área engenharia de software e inclui não apenas o programa de computador, em si, mas, também, manuais, especificações e configurações.

1.2 Máquinas elétricas

As máquinas CC caracterizam-se por sua versatilidade. Por meio das diversas combinações de enrolamentos de campo, excitados em derivação, série ou independentemente, elas podem ser projetadas de modo a apresentar uma ampla variedade de características de tensão versus corrente ou de velocidade versus conjugado, para operações dinâmicas e em regime permanente. Devido à facilidade com que podem ser controladas, sistemas de máquinas CC têm sido usados com frequência em aplicações que exigem uma ampla faixa de velocidades ou de controle preciso da saída do motor. Nos últimos anos, a tecnologia de estado sólido que é utilizada nos sistemas de acionamento CA desenvolveu-se o suficiente para que esses sistemas estejam substituindo as máquinas CC em aplicações antes associadas quase exclusivamente às máquinas CC. Entretanto, a versatilidade das máquinas CC, em combinação com a relativa simplicidade dos seus sistemas de acionamento, irá assegurar o seu uso continuado em uma ampla variedade de aplicações. (Fitzgerald)

Os motores de corrente contínua possuem enrolamentos de campo localizados no estator que são alimentados por corrente contínua, criando um campo magnético fixo. Os enrolamentos de armadura também são alimentados com corrente contínua, e a interação entre os condutores da armadura com o campo magnético fixo criado no estator produz o torque mecânico que é disponibilizado no eixo da máquina (KOSOW, 2000).

As principais configurações de excitação das máquinas CC são: série, paralelo e composta, sendo a forma como o enrolamento de campo é excitado, a diferença para cada configuração. No gerador CC, o enrolamento de campo é excitado com corrente contínua. Quando o enrolamento de armadura, presente no rotor e submetido à rotação do torque de entrada, corta as linhas de força geradas no estator, uma força eletromotriz com característica senoidal é induzida na armadura. Por meio da retificação mecânica realizada pelo comutador, a tensão alternada é transformada em contínua e entregue à carga (CREPPE; SIMONE, 2002).

1.2.1 O Circuito Equivalente De Um Motor CC

O circuito equivalente de um motor CC está mostrado na (FIGURA 2). Nessa figura, o circuito de armadura é representado por uma fonte de tensão ideal EA e um resistor RA. Essa representação é na realidade o equivalente Thévenin da estrutura completa do rotor, incluindo as bobinas do rotor, os interpolos e os enrolamentos de compensação, se presentes. A queda de tensão nas escovas é representada por uma pequena bateria Vescova que se opõe à corrente que circula na máquina. As bobinas de campo, que produzem o fluxo magnético do gerador, são representadas pelo indutor LF e pelo resistor RF. O resistor separado Raj representa um resistor externo variável, usado para controlar a corrente que circula no circuito de campo. Há algumas variações e simplificações desse circuito equivalente básico. A queda de tensão nas escovas é frequentemente apenas uma fração mínima da tensão gerada em uma máquina. Portanto, em casos não muito críticos, a queda de tensão nas escovas pode ser desprezada ou incluída de forma aproximada no valor de RA. Além disso, algumas vezes a resistência interna das bobinas de campo é combinada com o resistor variável e a resistência total é denominada RF (FIGURA 2b). Uma terceira variação é que alguns geradores têm mais do que uma bobina de campo, todas as quais são incluídas no circuito equivalente. (Chapman, 2013).

1.2.2 Os Motores De Excitação Independente e Em Derivação

Um motor CC de excitação independente é um motor cujo circuito de campo é alimentado a partir de uma fonte isolada de tensão constante, ao passo que um motor CC em derivação é um motor cujo circuito de campo é alimentado diretamente dos terminais de armadura do próprio motor. Na prática, quando a tensão da fonte de alimentação de um motor é constante, não há nenhuma diferença de comportamento entre esses dois tipos de máquinas. A não ser que seja especificado em contrário, sempre que o comportamento de um motor em derivação for descrito, também estaremos incluindo o motor de excitação independente. (Chapman, 2013). A equação da lei de Kirchhoff das tensões (LKT) para o circuito de armadura desses motores é:

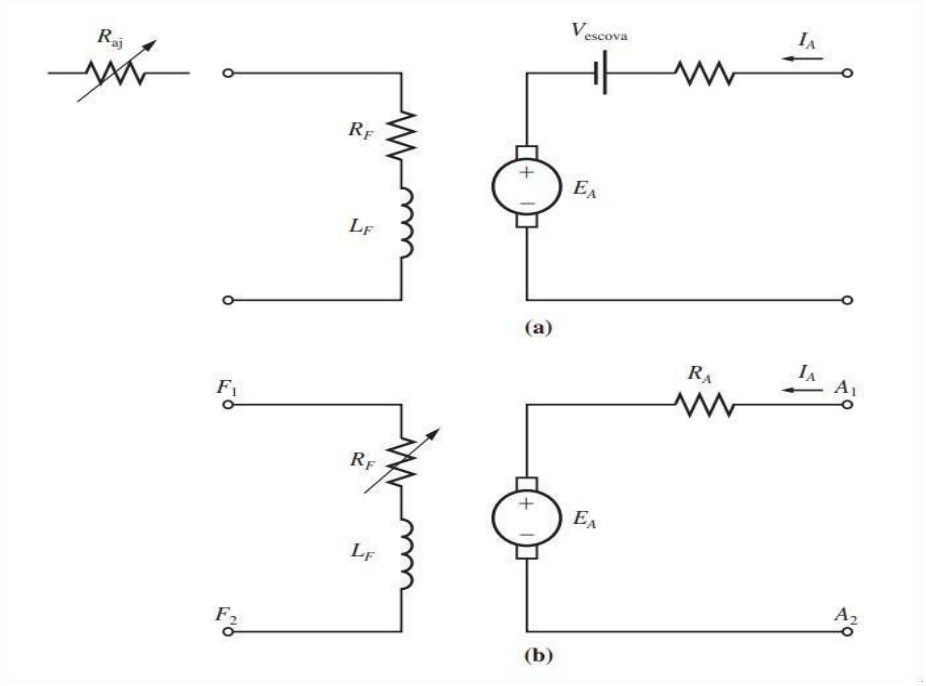
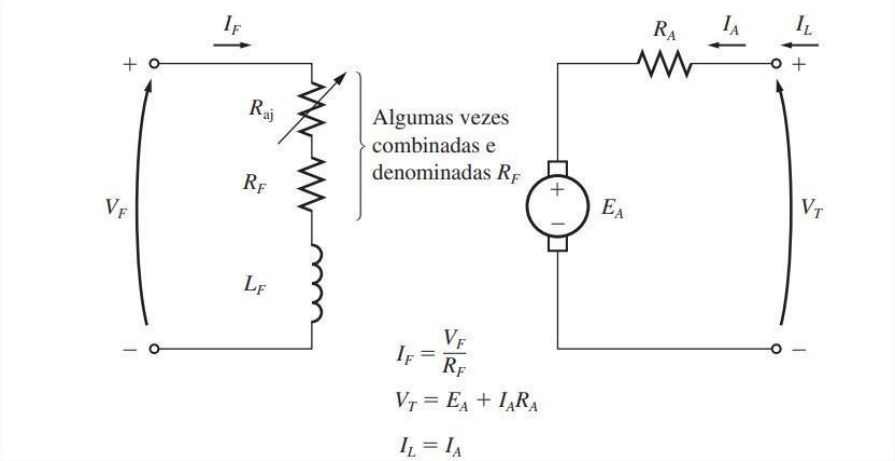


Figura 2 - (a) O circuito equivalente de um motor CC. (b) Um circuito equivalente simplificado em que a queda de tensão nas escovas foi eliminada e Raj foi combinada com a resistência de campo. Fonte: Chapman, 2013.



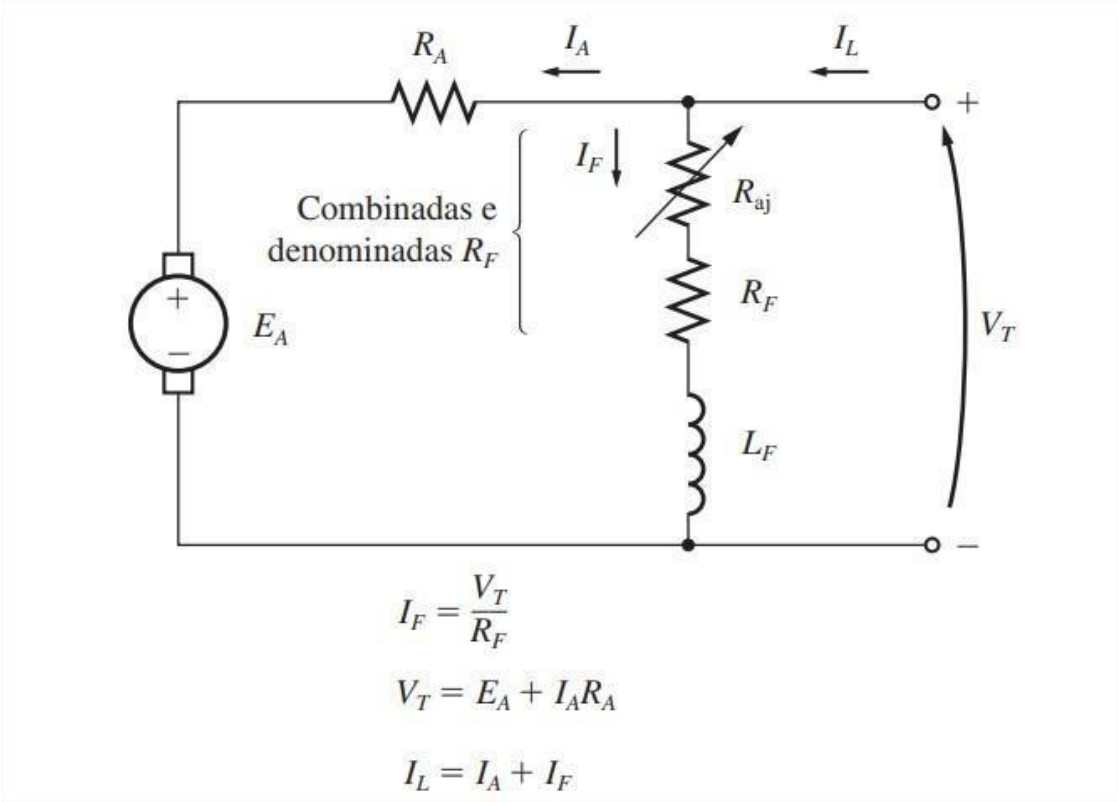


Figura 3 - (a) O circuito equivalente de um motor CC de excitação independente. (b) O circuito equivalente de um motor CC em derivação (shunt). (Chapman, 2012).

A característica de terminal de um motor CC em derivação

A característica de terminal de uma máquina é um gráfico que envolve as grandezas de saída da máquina. Para um motor, as grandezas de saída são o conjugado no eixo e a velocidade. Assim, a característica de terminal de um motor é um gráfico do seu conjugado de saída versus a velocidade. (Chapman, 2013).

A característica de saída de um motor CC em derivação pode ser obtida a partir das equações da tensão induzida e do conjugado mais a lei de Kirchhoff das tensões (LKT). Resultando na expressão:

$$\omega_m = \frac{V_T}{K\phi} - \frac{R_A}{(K\phi)^2}\,\tau_{\text{ind}} \quad (1)$$

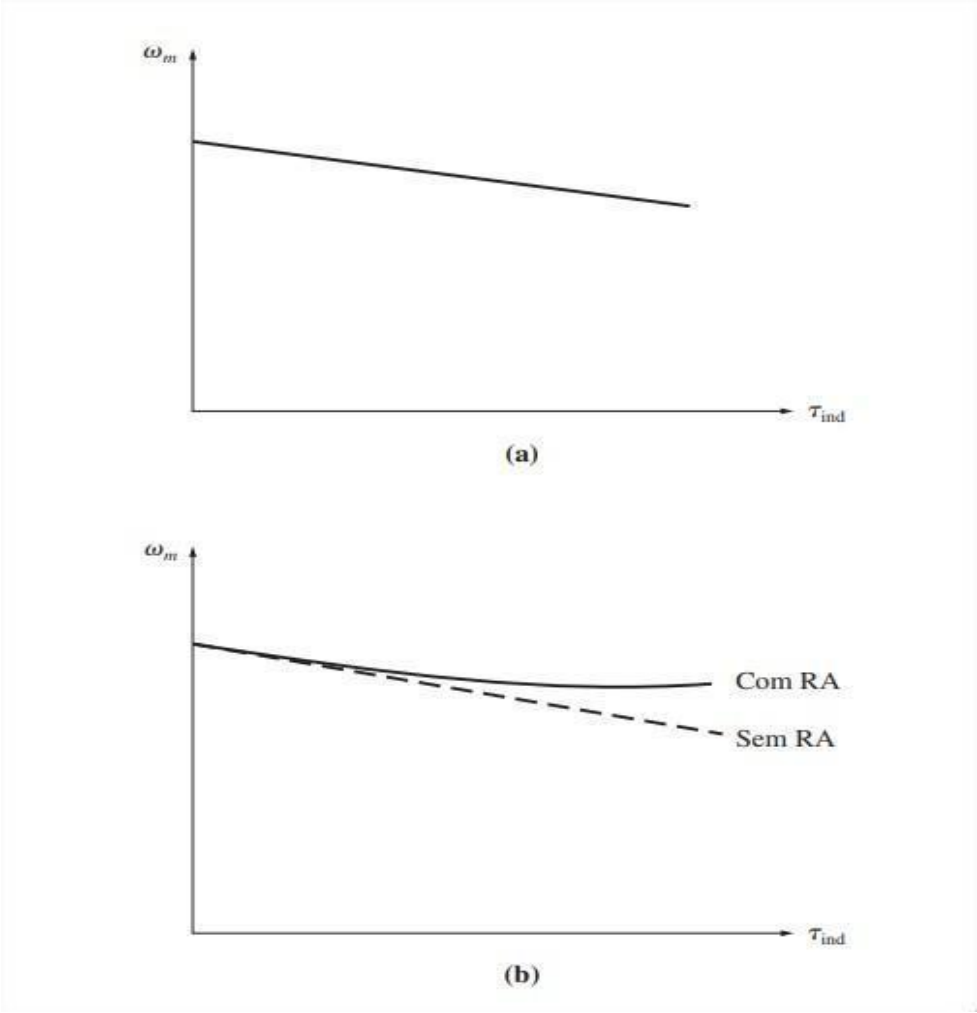


Figura 4- (a) Característica de conjugado versus velocidade de um motor CC em derivação ou de excitação independente, com enrolamentos de compensação para eliminar a reação de armadura. (b) Característica de conjugado versus velocidade de um motor em que a reação de armadura está presente (Chapman, 2012).

Um efeito interno do motor que dentre outros que pode afetar a forma da curva de conjugado versus velocidade é a reação de armadura. Se um motor apresentar reação de armadura, então os efeitos de enfraquecimento de fluxo reduzirão o seu fluxo quando a carga aumentar. Como em (1). mostra, para qualquer carga, o efeito de uma redução de fluxo é o aumento da velocidade do motor em relação à velocidade na qual o motor giraria se não houvesse a reação de armadura. A característica de conjugado versus velocidade de um motor CC em derivação com reação de armadura está mostrada na (FIGURA 4b). Naturalmente, se um motor tiver enrolamentos de compensação, não haverá problemas de enfraquecimento de fluxo na máquina, o qual será constante.

Se houver enrolamentos de compensação em um motor CC em derivação, de modo que seu fluxo seja constante independentemente da carga, e se a velocidade e a corrente de armadura

do motor forem conhecidas para qualquer valor de carga, então sua velocidade poderá ser calculada para qualquer outro valor de carga, desde que a corrente de armadura para aquela carga seja conhecida ou possa ser determinada. (Chapman, 2013).

1.2.3 O Motor CC Série

Um motor CC série é um motor CC cujos enrolamentos de campo consistem em relativamente poucas espiras conectadas em série com o circuito de armadura. Em um motor série, a corrente de armadura, a corrente de campo e a corrente de linha são todas a mesma.

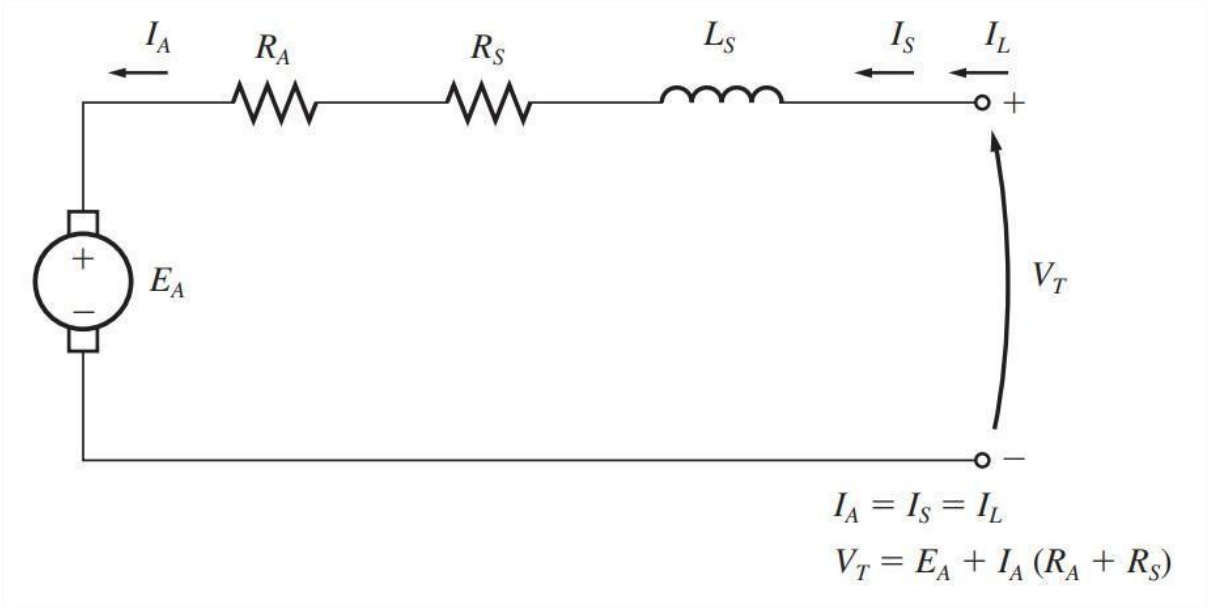


Figura 5- O circuito equivalente de um motor CC série.Fonte (Chapman, 2012).

A relação resultante de conjugado versus velocidade é:

$$\omega_m = \frac{V_T}{\sqrt{Kc}} \frac{1}{\sqrt{\tau_{\text{ind}}}} - \frac{R_A + R_S}{Kc} \quad (2)$$

Observamos que, para um motor série não saturado, a velocidade do motor varia com o inverso da raiz quadrada do conjugado. Trata-se de uma relação bem incomum! Essa característica de conjugado versus velocidade ideal está plotada na (FIGURA 6). Examinando essa equação, pode-se ver imediatamente uma das desvantagens dos motores série. Quando o conjugado desse motor vai a zero, sua velocidade vai a infinito. Na prática, o conjugado nunca pode ser inteiramente zero devido às perdas mecânicas, no núcleo e suplementares.

Entretanto, se nenhuma outra carga mecânica for acoplada ao motor, ele poderá girar suficientemente rápido para se danificar seriamente. Por isso nunca deve-se deixar um motor CC série completamente sem carga e nunca acoplar a carga mecânica por meio de uma correia ou outro mecanismo que possa se romper. Se isso acontecesse e o motor ficasse sem carga enquanto estivesse em funcionamento, os resultados poderiam ser muito graves. (Chapman, 2013).

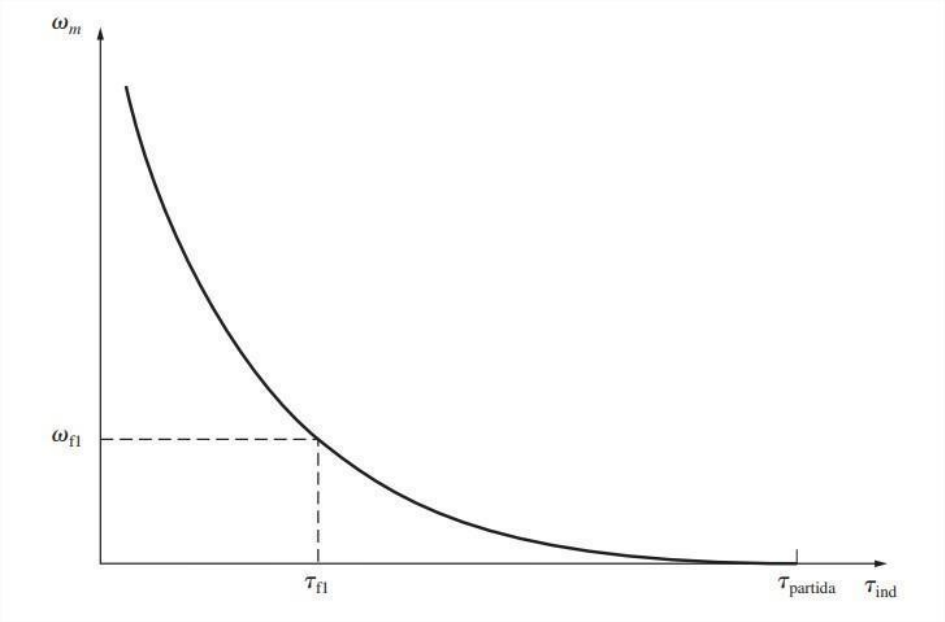


Figura 6- A característica de conjugado versus velocidade de um motor CC série (Chapman, 2012).

1.2.3 O Motor CC Composto

Um motor CC composto é um motor que tem campos em derivação e em série. Esse motor é mostrado na (FIGURA 7). Os pontos ou marcas que aparecem nas bobinas dos dois campos têm o mesmo significado que os pontos ou as marcas em um transformador: uma corrente que entra no terminal com marca produz uma força magnetomotriz positiva. Se a corrente entrar nos terminais com marcas de ambas as bobinas de campo, as forças magnetomotrizes resultantes combinam-se, produzindo uma força magnetomotriz total maior. Essa situação é conhecida como composição cumulativa ou aditiva. Se a corrente entrar no terminal com marca de uma bobina de campo e sair pelo terminal com marca da outra bobina de campo, as forças magnetomotrizes resultantes subtraem-se.(Chapman, 2013). Na

(FIGURA 7), as marcas circulares correspondem à composição cumulativa do motor e as marcas quadradas correspondem à composição diferencial.

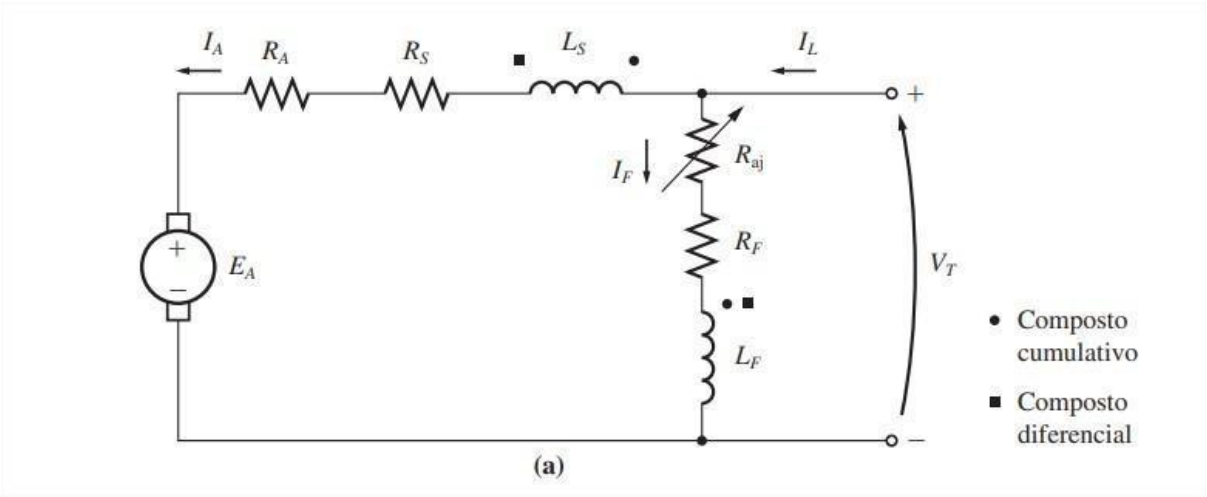


Figura 7- O circuito equivalente de motores CC compostos: (a) ligação em derivação longa; (b) ligação em derivação curta. (Chapman, 2012).

A característica de conjugado x velocidade de um motor CC composto cumulativo

No motor CC composto cumulativo (ou aditivo), há uma componente de fluxo que é constante e outra que é proporcional à sua corrente de armadura (e portanto à sua carga). Dessa forma, o motor composto cumulativo tem um conjugado de partida mais elevado do que um motor em derivação (cujo fluxo é constante), mas um conjugado de partida mais baixo do que o de um motor série (cujo fluxo inteiro é proporcional à corrente de armadura). De certa forma, o motor CC composto cumulativo combina as melhores características de ambos os motores em derivação e série. Como em um motor série, ele apresenta um conjugado extra para a partida e, como um motor em derivação, a velocidade não dispara quando ele está sem carga. Com cargas leves, o campo em série tem um efeito muito pequeno, o que leva o motor a comportar-se aproximadamente como um motor CC em derivação. Quando a carga torna-se muito grande, o fluxo do enrolamento em série torna-se bem importante e a característica de conjugado versus velocidade começa a se tornar semelhante à curva característica de um motor série. (Chapman, 2013). Uma comparação das características de conjugado versus velocidades de cada um desses tipos de máquinas está mostrada na (FIGURA 8).

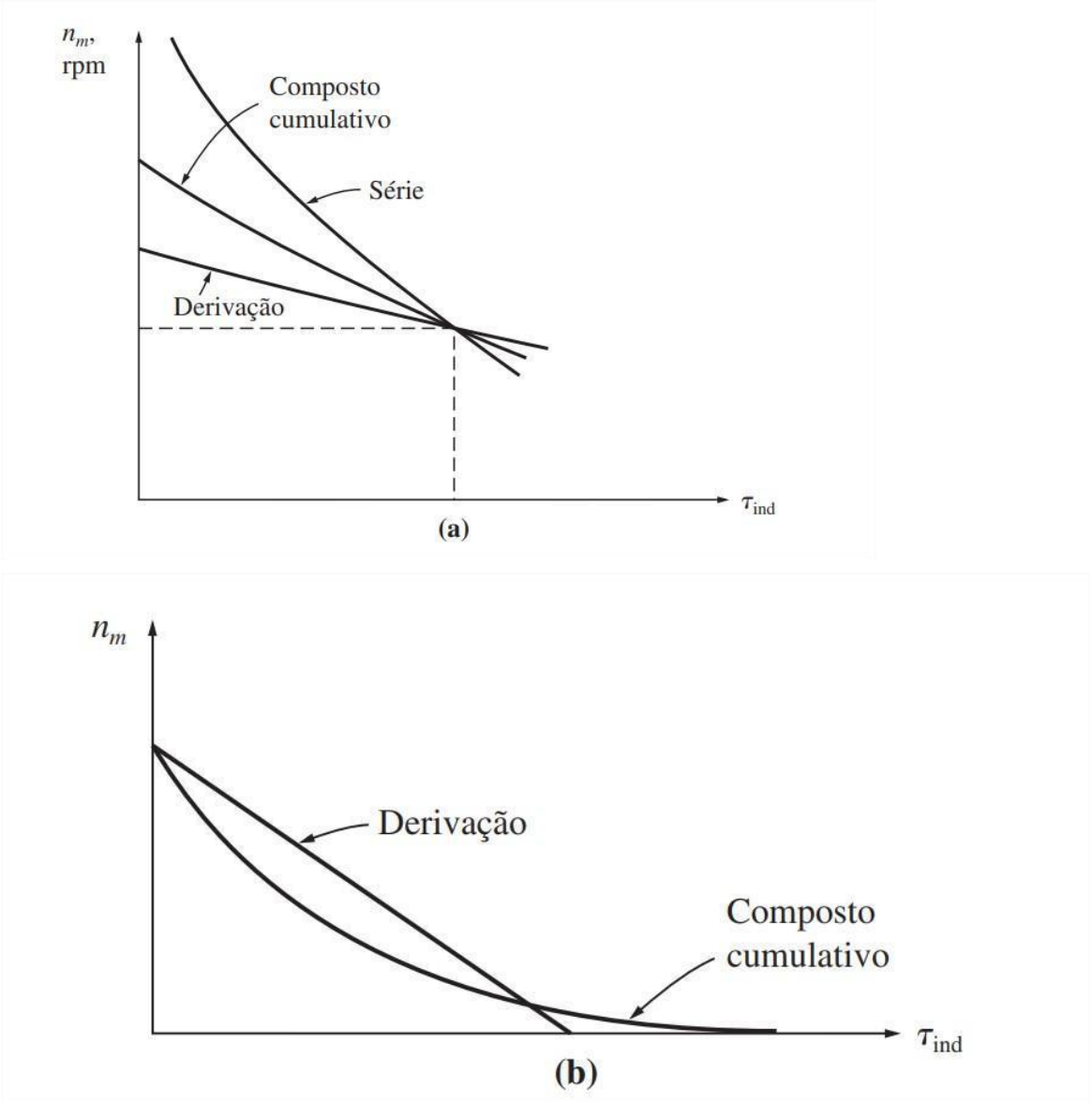


Figura 8- (a) A característica de conjugado versus velocidade de um motor CC composto cumulativo comparada com a de motores série e em derivação, com a mesma carga plena nominal. (b) A característica de conjugado versus velocidade de um motor CC composto cumulativo comparada com a de um motor em derivação, com a mesma velocidade a vazio (Chapman, 2012).

.

A característica de conjugado velocidade de um motor CC composto diferencial

Em um motor CC composto diferencial, a força magnetomotriz em derivação e a força magnetomotriz em série subtraem-se entre si. Isso significa que, quando a carga no motor aumenta, a IA aumenta e o fluxo no motor diminui. Entretanto, quando o fluxo diminui, a velocidade do motor aumenta. Essa elevação de velocidade causa outro aumento de carga, o que por sua vez aumenta a IA e diminui mais o fluxo, aumentando novamente a velocidade. O

resultado é que um motor CC composto diferencial é instável e sua velocidade tende a disparar. Essa instabilidade é muito pior do que a de um motor em derivação com reação de armadura. É tão ruim que um motor CC composto diferencial não é adequado para nenhuma aplicação. Para tornar as coisas piores, é impossível dar partida a esse motor. Nas condições de partida, a corrente de armadura e a corrente do campo em série são muito elevadas. Como o fluxo em série é subtraído do fluxo em derivação, o campo em série pode na realidade inverter a polaridade magnética dos pólos da máquina. Tipicamente, o motor permanece imóvel ou gira lentamente no sentido errôneo, ao mesmo tempo que os enrolamentos queimam-se, devido à excessiva corrente de armadura. Quando é necessário dar partida a um motor como esse, seu campo em série deve ser curto-circuitado, de modo que durante a partida ele se comporte como um motor comum em derivação.

Devido aos problemas de estabilidade do motor CC composto diferencial, ele quase nunca é usado intencionalmente. Entretanto, poderá resultar um motor composto diferencial se o sentido do fluxo de potência for invertido em um gerador composto cumulativo. Por essa razão, quando um gerador CC composto cumulativo é usado para alimentar um sistema com potência elétrica, ele terá um circuito de proteção de inversão de potência que o desligará da linha se houver uma inversão no fluxo de potência. Em nenhum conjunto de motor e gerador, no qual se espera que a potência possa fluir em ambos os sentidos, pode-se usar um motor composto diferencial e, consequentemente, não se pode usar um gerador composto cumulativo. (Chapman, 2013). Uma característica de terminal típica para um motor CC composto diferencial está mostrada na (FIGURA 9).

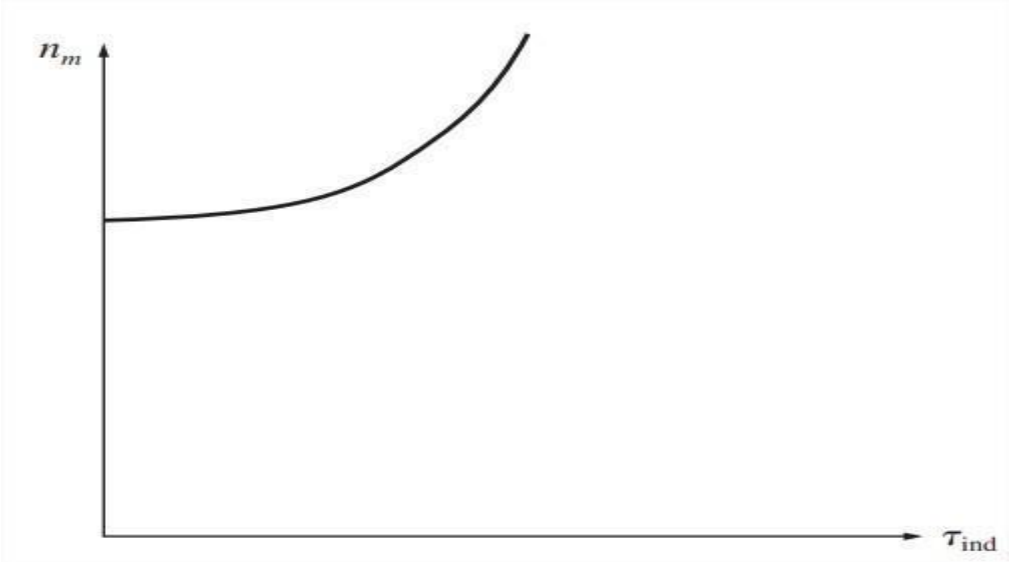


Figura 9- A característica de conjugado versus velocidade de um motor CC composto diferencial (Chapman, 2012)

1.3 Objetivo Geral

Este trabalho tem como objetivo construir um simulador do tipo Windows forms

Dado o objetivo geral, têm-se como objetivos específicos:

- Estudar lógica de programação e a linguagem C# e o uso da IDE Visual Studio;
- Estudar as máquinas elétricas CC;
- Disponibilizar um simulador para uso na aprendizagem de engenharia.

2 Materiais e Métodos

2.1 Ferramentas

Sabendo da aderência do sistema operacional windows optamos pela criação de um simulador para trabalhar nesse ambiente, e para isso vimos na utilização das ferramentas descritas a seguir a melhor opção.

2.1.1 Visual Studio

Microsoft Visual Studio é um ambiente de desenvolvimento integrado (IDE) da Microsoft™ para desenvolvimento de software especialmente dedicado ao .NET Framework e às linguagens Visual Basic (VB), C, C++, C# (C Sharp) e F# (F Sharp). Também é um produto de desenvolvimento na área web, usando a plataforma do ASP.NET, como websites, aplicativos web, serviços web e aplicativos móveis.

IDE, do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo.[1][2]
Geralmente os IDEs facilitam a técnica de RAD (de Rapid Application Development, ou "Desenvolvimento Rápido de Aplicações"), que visa a maior produtividade dos desenvolvedores.
As características e ferramentas mais comuns encontradas nos IDEs são:

- Editor - edita o código-fonte do programa escrito na(s) linguagem(ns) suportada(s) pela IDE;
- Compilador (compiler) - compila o código-fonte do programa, editado em uma linguagem específica e a transforma em linguagem de máquina;
- Linker - liga (linka) os vários "pedaços" de código-fonte, compilados em linguagem de máquina, em um programa executável que pode ser executado em um computador ou outro dispositivo computacional;
- Depurador (debugger) - auxilia no processo de encontrar e corrigir defeitos no código-fonte do programa, na tentativa de aprimorar a qualidade de software;

- Modelagem de dados (modeling) - criação do modelo de classes, objetos, interfaces, associações e interações dos artefatos envolvidos no software com o objetivo de solucionar as necessidades-alvo do software final;
- Geração de código - característica mais explorada em Ferramentas CASE, a geração de código também é encontrada em IDEs, contudo com um escopo mais direcionado a templates de código comumente utilizados para solucionar problemas rotineiros. Todavia, em conjunto com ferramentas de modelagem, a geração pode gerar praticamente todo o código-fonte do programa com base no modelo proposto, tornando muito mais rápido o processo de desenvolvimento e distribuição do software;
- Distribuição (deploy) - auxilia no processo de criação do instalador do software, ou outra forma de distribuição, seja discos ou via internet;

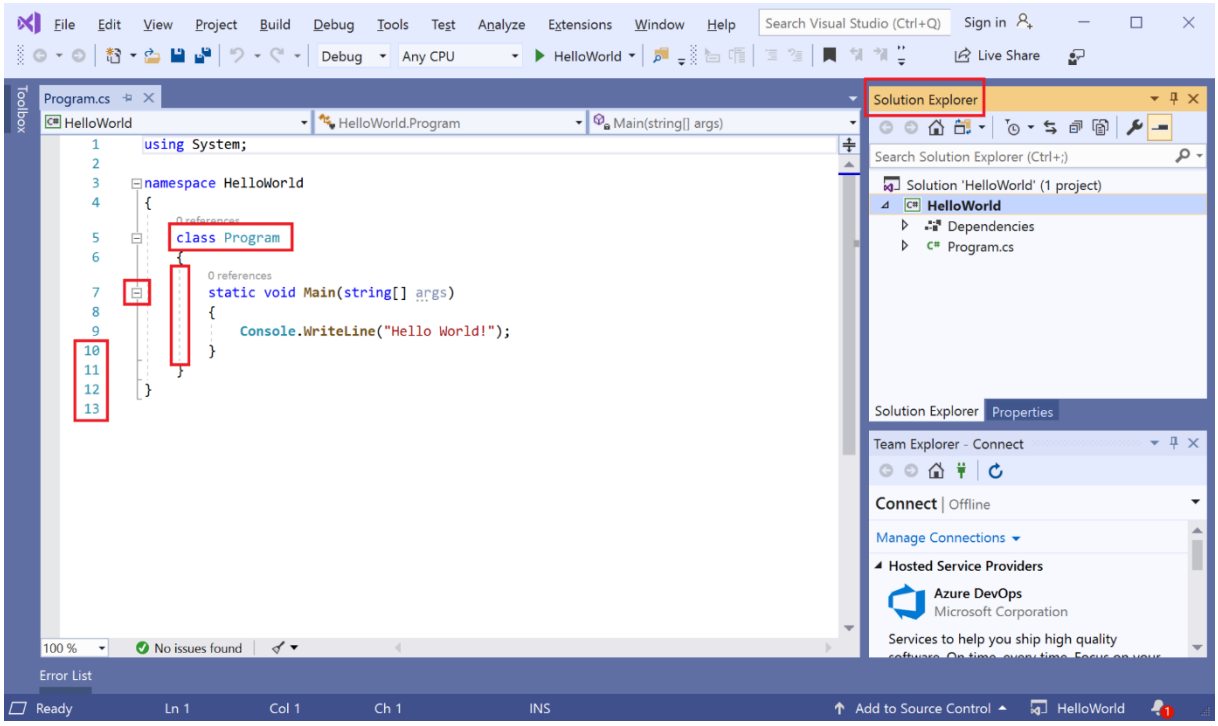


Figura 10 - Ambiente de desenvolvimento da Microsoft™, Visual Studio. (Fonte: Autor).

- Testes Automatizados (automated tests) - realiza testes no software de forma automatizada, com base em scripts ou programas de testes previamente especificados, gerando um relatório, assim auxiliando na análise do impacto das alterações no código-fonte. Ferramentas deste tipo mais comuns no mercado são chamadas robôs de testes;

- Refatoração (refactoring) - consiste na melhoria constante do código-fonte do software, seja na construção de código mais otimizado, mais limpo e/ou com melhor entendimento pelos envolvidos no desenvolvimento do software. A refatoração, em

  conjunto com os testes automatizados, é uma poderosa ferramenta no processo de erradicação de "bugs", tendo em vista que os testes "garantem" o mesmo comportamento externo do software ou da característica sendo reconstruída.(Microsoft,2020).

2.1.2 Windows Forms

Windows Forms é uma estrutura de interface do usuário para criar aplicativos da área de trabalho do Windows. Ele fornece uma das maneiras mais produtivas de criar aplicativos de área de trabalho com base no designer visual fornecido no Visual Studio. Funcionalidades como o posicionamento do tipo "arrastar e soltar" de controles visuais facilitam a criação de aplicativos da área de trabalho.

Com o Windows Forms, se desenvolve aplicativos graficamente ricos que são fáceis de implantar, atualizar e trabalhar enquanto estiverem offline ou conectados à Internet. Windows Forms aplicativos podem acessar o hardware local e o sistema de arquivos do computador em que o aplicativo está sendo executado.

Windows Forms é uma tecnologia de interface do usuário para .NET, um conjunto de bibliotecas gerenciadas que simplificam tarefas comuns de aplicativos, como leitura e gravação no sistema de arquivos. Ao usar um ambiente de desenvolvimento como o Visual Studio, se pode criar Windows Forms aplicativos de cliente inteligente que exibem informações, solicitam a entrada de usuários e se comunicam com computadores remotos em uma rede.

Nos Windows Forms, um formulário é uma superfície visual na qual são exibidas informações para o usuário. Normalmente, você cria Windows Forms aplicativos adicionando controles a formulários e desenvolvendo respostas para ações do usuário, como cliques do mouse ou pressionamentos de tecla. Um controle é um elemento de interface do usuário discreto que exibe dados ou aceita entrada de dados.

Quando um usuário executa alguma ação em seu formulário ou em um de seus controles, a ação gera um evento. Seu aplicativo reage a esses eventos com o código e processa os eventos quando eles ocorrem.

O Windows Forms contém uma variedade de controles que se pode adicionar a formulários: controles que exibem caixas de texto, botões, caixas suspensas, botões de opção e até mesmo páginas da Web. Se um controle existente não atender às necessidades, Windows Forms também dará suporte à criação de controles próprios personalizados usando a UserControl classe.

Windows Forms tem controles avançados de interface do usuário que emulam recursos em aplicativos de alto nível, como Microsoft Office. Ao usar os ToolStrip controles e MenuStrip , pode-se criar barras de ferramentas e menus que contêm texto e imagens, exibir submenus e hospedar outros controles, como caixas de texto e caixas de combinação.(Microsoft,2020)

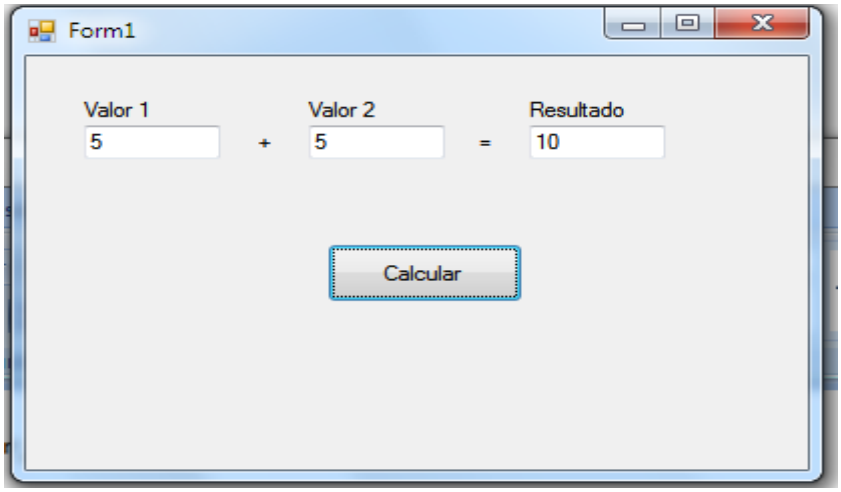


Figura 11- Exemplo de formulário do Windows forms, para se fazer o cálculo de adição de dois valores. (Fonte: Autor).

Com o Designer de formulários do Windows do tipo "arrastar e soltar" no Visual Studio, pode-se facilmente criar Windows Forms aplicativos. Basta selecionar os controles com o cursor e colocá-los onde deseja no formulário. O designer oferece ferramentas como linhas de grade e linhas de alinhamento para facilitar o alinhamento dos controles. Pode-se usar os FlowLayoutPanel TableLayoutPanel controles, e SplitContainer para criar layouts de formulário avançados em menos tempo.

Por fim, se for preciso criar elementos próprios personalizados de interface do usuário, o System.Drawing namespace conterá uma grande seleção de classes para processar linhas, círculos e outras formas diretamente em um formulário.

2.1.3 C#

O C# (pronuncia-se "C Sharp") é uma linguagem de programação simples, moderna, orientada a objeto e fortemente tipada. O C# tem suas raízes na família C de idiomas e estará imediatamente familiarizado com os programadores C, C++ e Java. O C# é padronizado pela ECMA International como o *ECMA-334 _ Standard e por ISO/IEC como o padrão _ *iso/IEC 23270**. O compilador C# da Microsoft™ para o .NET Framework é uma implementação em conformidade desses dois padrões.

O C# é uma linguagem orientada a objeto, mas inclui ainda suporte para programação orientada a componentes. O design de software atual depende cada vez mais dos componentes de software na forma de pacotes independentes e auto descritivos de funcionalidade. O principal é que esses componentes apresentam um modelo de programação com propriedades, métodos e eventos; eles têm atributos que fornecem informações declarativas sobre o componente; e incorporam sua própria documentação. O c# fornece construções de linguagem para dar suporte direto a esses conceitos, tornando o C# uma linguagem muito natural para criar e usar componentes de software.

Vários recursos do C# auxiliam na construção de aplicativos robustos e duráveis: coleta de lixo _ recupera automaticamente a memória ocupada por objetos não utilizados; a _manipulação de exceção_ fornece uma abordagem estruturada e extensível para detecção e recuperação de erros; e o design _ tipo-seguro da linguagem torna impossível a leitura de variáveis não inicializadas, para indexar matrizes além dos limites ou para executar conversões de tipo desmarcadas.

C# tem um sistema de tipo unificado. Todos os tipos do C#, incluindo tipos primitivos, como int e double, herdam de um único tipo de object raiz. Assim, todos os tipos compartilham um conjunto de operações comuns, e valores de qualquer tipo podem ser armazenados, transportados e operados de maneira consistente. Além disso, C# oferece

suporte a tipos de referência e tipos de valor definidos pelo usuário, permitindo a alocação dinâmica de objetos, bem como o armazenamento em linha de estruturas leves.

Para garantir que os programas e bibliotecas C# possam evoluir ao longo do tempo de maneira compatível, muito ênfase foi colocado no controle de versão no design do C#. Muitas linguagens de programação prestam pouca atenção a esse problema e, como resultado, programas escritos nessas linguagens quebram com mais frequência do que o necessário quando versões mais recentes das bibliotecas dependentes são introduzidas. Aspectos do design do C# que foram influenciados diretamente pelas considerações de controle de versão incluem os virtual override modificadores and separados, as regras para resolução de sobrecarga de método e suporte para declarações de membro de interface explícitas. (Microsoft,2017)

2.2 Métodos

2.2.1 Sem usar a biblioteca de plotagem

Visando plotar diversas curvas em função das variáveis de entrada da máquina CC será utilizado a ferramenta charts para Windows forms no Visual Studio. Para tanto, será criado funções que a partir dos dados a serem plotados, façam todo esse ajuste.

Os dados de plotagem serão obtidos através da computação dos parâmetros de entrada usando os modelos matemáticos das diversas topologias de motores CC, em uma faixa de operação pré-estabelecida.

A princípio foi feito a plotagem dos gráficos sem o uso da biblioteca de plotagem ScottPlot, obtendo-se gráficos satisfatórios, porém se fazendo dificultoso a questão de encapsular em uma função devido aos comandos utilizados, para tornar o código mais limpo e fácil de manipular.

```
public void SetupChart2()
{
chart2.Tag = new ChartScaleData(chart2);
        chart2.Series.Clear();
        double xMax, yMax, xMin, yMin;
        xMax = yMax = xMin = yMin = 0;
        {
```

```
Series s = new Series();
s.IsVisibleInLegend = true;

s.IsXValueIndexed = false;
s.ChartType = SeriesChartType.FastLine;
s.BorderWidth = 2;
//s.MarkerStyle = (MarkerStyle)(j - jMin + 1); //0 = MarkerStyle.None
s.MarkerBorderWidth = 3;
s.MarkerStep = 1;
s.MarkerSize = 10;
chart2.Series.Add(s);
 for (int i = 0; i < iMax; i++)
{
   i_l += 10;
   i_a[i] = i_l - v_t / r_f;
   e_a[i] = v_t - i_a[i] * r_a;
   f_ar[i] = (i_a[i] / 200) * f_ar0;
   i_f[i] = v_t / r_f - f_ar[i] / n_f;
}
e_a0 = interpola(e_avalues, if_values, i_f, e_a0);
for (int i = 0; i < iMax; i++)
{
    n[i] = (e_a[i] / e_a0[i]) * n_0;
   t_ind[i] = e_a[i] * i_a[i] / (n[0] * 2 * Math.PI / 60);
   double Y = n[i];
   double X = t_ind[i];
   if (i == 0)
   {
      xMin = xMax = X;
      yMin = yMax = Y;
   }
   else
```

```
            {
                if (Y < yMin) yMin = Y;

                if (Y > yMax) yMax = Y;
                if (X < xMin) xMin = X;
                if (X > xMax) xMax = X;
            }
            Chart chart;
            chart2.Series[0].Points.AddXY(X, Y);
        }
    }
    double xInt, xMinInt, yInt, yMinInt;
    xInt = xMinInt = yInt = yMinInt = 0;
    if (useNiceRoundNumbers)
    {
        GetNiceRoundNumbers(ref xMin, ref xMax, ref xInt, ref xMinInt);
        GetNiceRoundNumbers(ref yMin, ref yMax, ref yInt, ref yMinInt);
    }
    parametros_grafico2(ref xMin, ref xMax, ref xInt, ref xMinInt, ref yMin, ref yMax,
ref yInt, ref yMinInt);
}
```

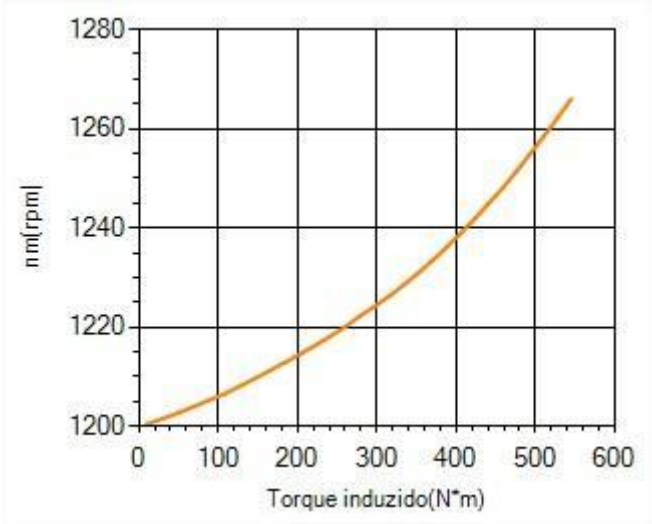


Figura 12 - Plotagem sem utilizar uma biblioteca de plot (Fonte: Autor).

Posteriormente ao se fazer a plotagem com a biblioteca ScottPlot o código pra cada gráfico se reduziu a:

```
for (int i = 0; i < iMax; i++)
        {

           i_l += 10;
           i_a[i] = i_l - v_t / r_f;
           e_a[i] = v_t - i_a[i] * r_a;
           f_ar[i] = (i_a[i] / 200) * f_ar0;
           i_f[i] = v_t / r_f - f_ar[i] / n_f;
        }
        e_a0 = interpola(arrlista, arrlista1, i_f, e_a0);

        for (int i = 0; i < iMax; i++)
        {
           n[i] = (e_a[i] / e_a0[i]) * n_0;
           t_ind[i] = e_a[i] * i_a[i] / (n[0] * 2 * Math.PI / 60);
           Y[i] = n[i];
           X[i] = t_ind[i];
        }
        formsPlot1.Plot.Clear();
        formsPlot1.Plot.AddScatter(X, Y);
        formsPlot1.Refresh();
```

Podemos destacar também o uso de uma função de interpolação de valores, devido a característica de a curva de magnetização não conter pontos, ou seja, valores específicos necessários para o cálculo e plotagem das curvas de respostas.

A interpolação é um cálculo numérico que se faz usando alguma dos métodos existentes na literatura de modo a estimar um valor não aferido na construção da curva de magnetização da máquina.

2.2.2 **Com a biblioteca de plotagem: Scott Plot**

ScottPlot é uma biblioteca de plotagem gratuita e de código aberto para .NET que facilita a exibição interativa de grandes conjuntos de dados. Gráficos de linhas, gráficos de barras, gráficos de pizza, gráficos de dispersão e muito mais podem ser criados com apenas algumas linhas de código.

Posteriormente utilizando-se a biblioteca podemos notar uma diferença aparente que se manifesta pela harmonia do gráfico, o que poderia ser feito acima. A despeito disso não se faz mais necessário que estabeleçamos todos os parâmetros de construção do gráfico, deixando dessa forma o código mais fácil de manusear.

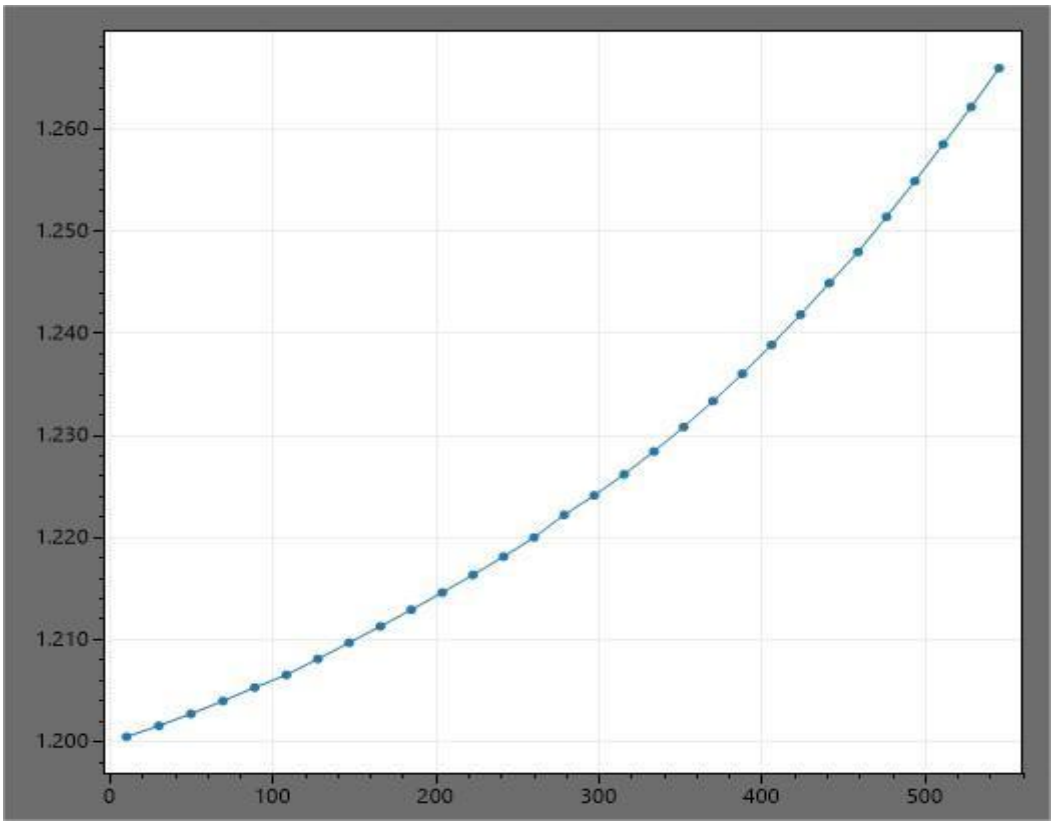


Figura 13- Plotagem utilizando a biblioteca ScottPlot (Fonte: Autor).

Foi implementado um método para busca do arquivo .txt que contém a curva de magnetização da máquina a qual se quer obter o plot do comportamento.

Além disso, podemos destacar a aplicação do evento `timer_Tick` que foi o que possibilitou o auto preenchimento das caixas de entrada, após um tempo pré-determinado após estar vazia de entrada, com um valor pré-testado.

O evento `timer_Tick` funciona da seguinte forma, é ativado periodicamente por um timer vinculado a ele com periodicidade pré-determinada, e dentro do evento podemos colocar o que queremos que seja executado.

E o resultado completo da interface pode ser visto abaixo:

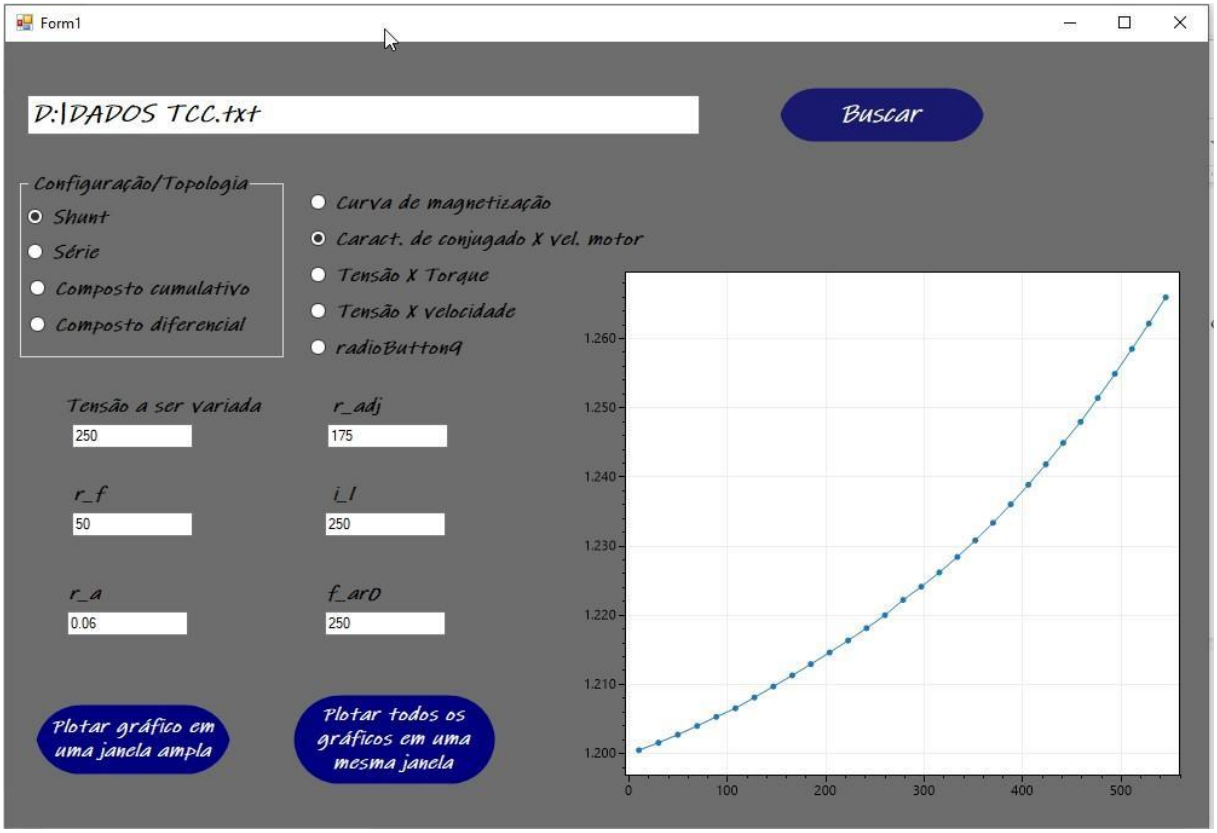


Figura 14- Interface do App para simulação das curvas de comportamento da máquina CC (Fonte: Autor).

Ao se variar os valores dos parâmetros de entrada temos automaticamente a alteração do gráfico.

3 Resultados e Discussão

3.1 Motor shunt

Podemos notar que da proposta inicial foi acrescida mais algumas funcionalidades dentre as quais a inclusão da opção de plotar dados pré-testados presentes internamente no código do aplicativo, facilitando no sentido de não necessitar incluir o arquivo com a curva de magnetização e os parâmetros também necessários para plotagem.

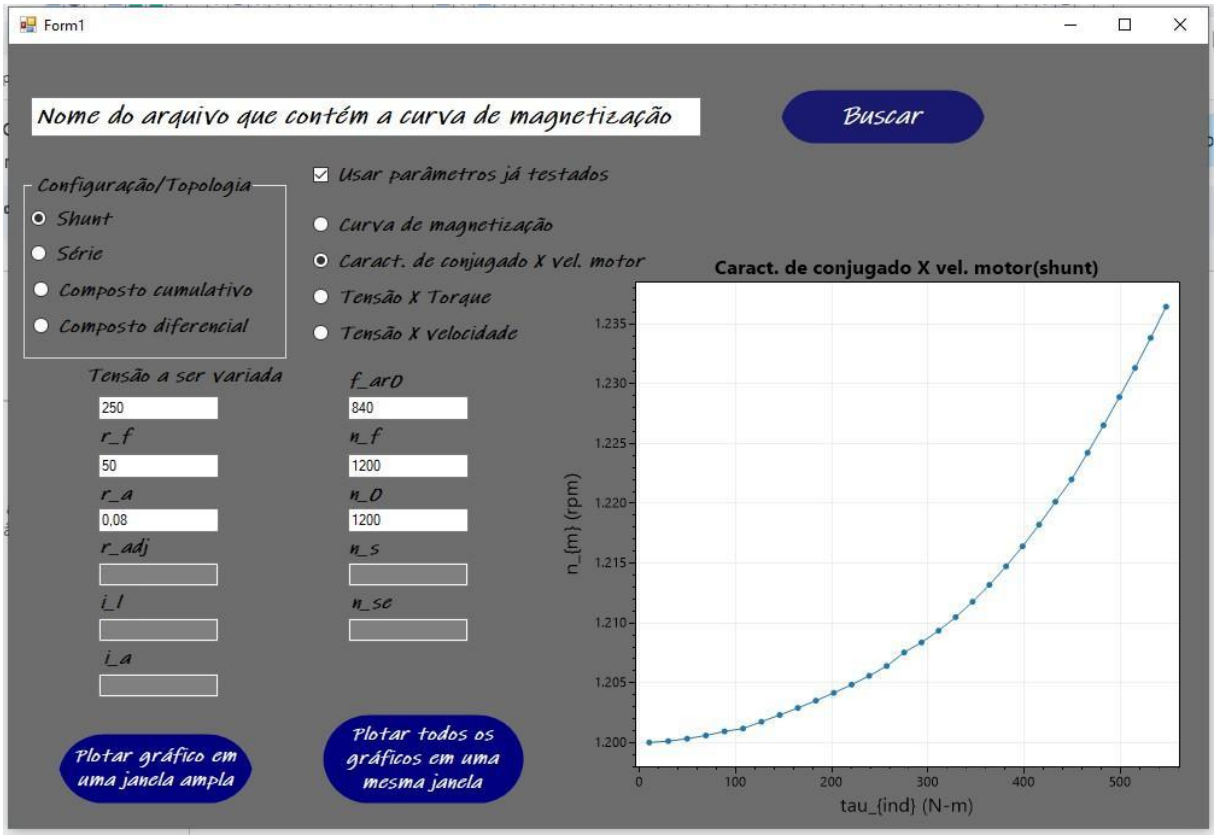


Figura 15- Curva de resposta do motor CC shunt, característica de conjugado X velocidade, plotadas na interface do programa (Fonte: Autor).

Como iremos especificar logo abaixo, os campos(as textBox), com os valores são os parâmetros que influenciam na resposta da topologia, assim o restante dos campos correspondentes aos parâmetros não utilizados estão impedidos de modificação para uma melhor usabilidade e situabilidade do resultado plotado.

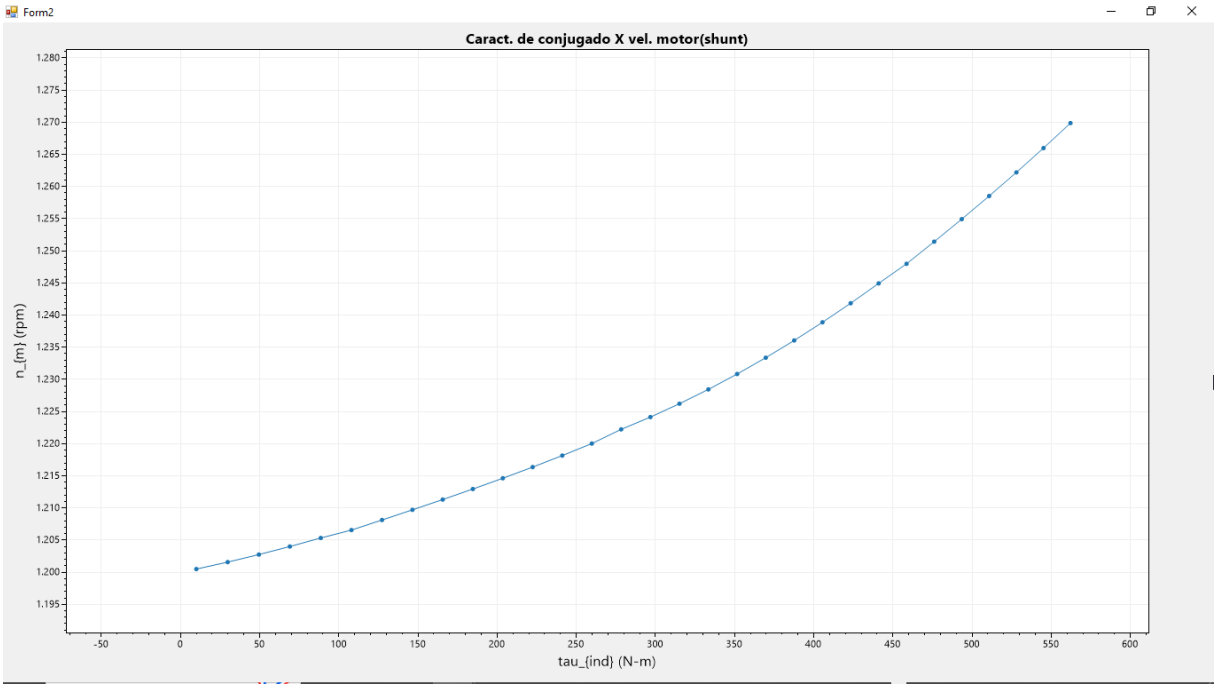


Figura 16- Curva de resposta do motor CC shunt, característica de conjugado X velocidade, plotadas a parte da interface (Fonte: Autor).

A curva de característica de conjugado versus velocidade do motor com reação de armadura é obtida variando-se a corrente de linha para os demais parâmetros fixos, que no caso são a tensão de terminal, a resistência de campo, a resistência de armadura, o número de espiras do campo e a reação de armadura para 200 A. À medida que a corrente de linha aumenta consequentemente tanto a velocidade quanto a característica de conjugado também aumentam com um comportamento próprio da máquina.

Na figura X podemos observar a correspondência do valor plotado pelo aplicativo objeto de trato com uma curva presente na literatura para os mesmos valores de parâmetros. O ajuste de escala e enquadramento perceptivelmente não é o mesmo, mas verificando ponto a ponto nota-se que a correspondência entre os gráficos é verdadeira.

Comparando-se o aplicativo com o ambiente do matlab para casos em que se vai fazer estudos, testes e comparações em uma topologia dado a variação exaustiva de um parâmetro como por exemplo a resistência de campo objetivando-se ver a variação do comportamento das curvas implementadas, pode ser percebido a principal vantagem do aplicativo no sentido de fazer atualizações imediatas mediante a alteração dos parâmetros.

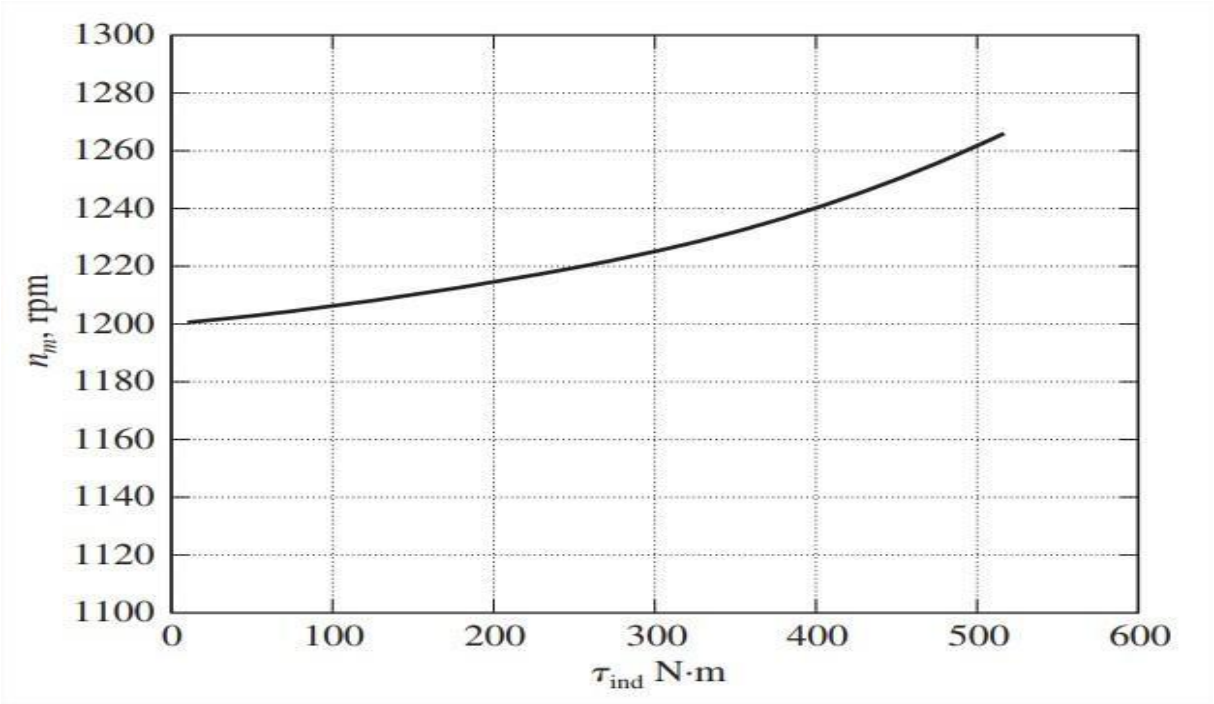


Figura 17- Curva de resposta do motor CC shunt, característica de conjugado X velocidade, plotada no matlab (Chapman, 2012).

3.2 Motor série

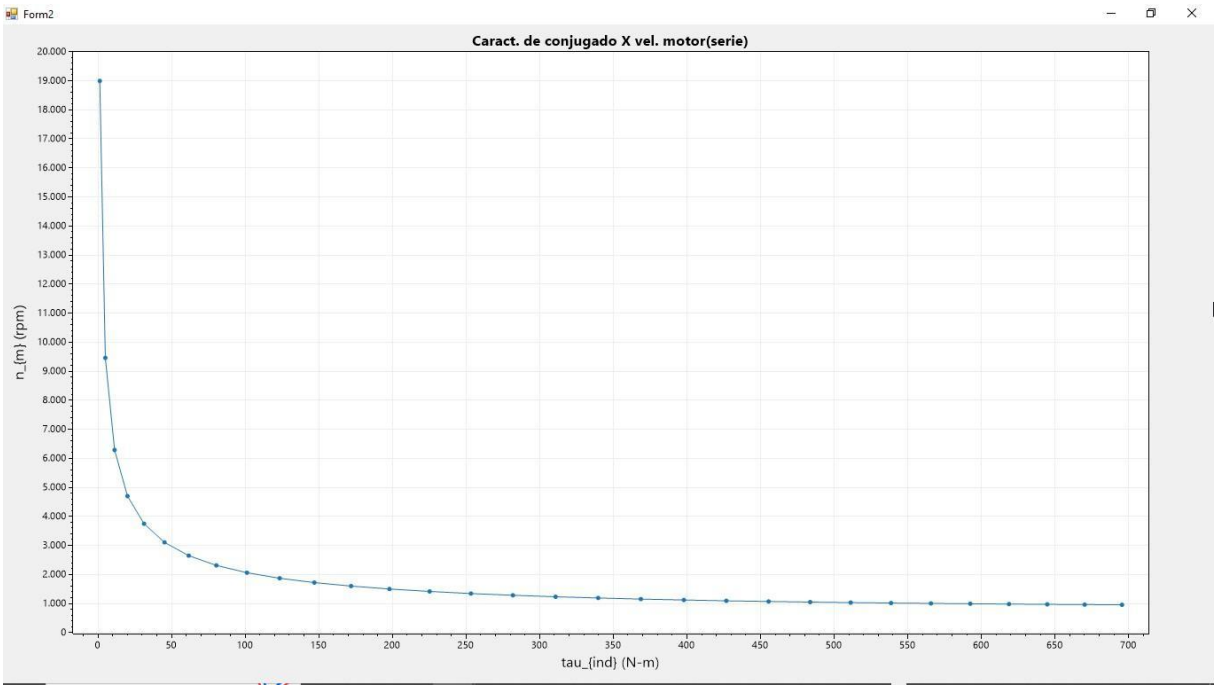


Figura 18- Curva de resposta do motor CC série, característica de conjugado X velocidade (Fonte: Autor).

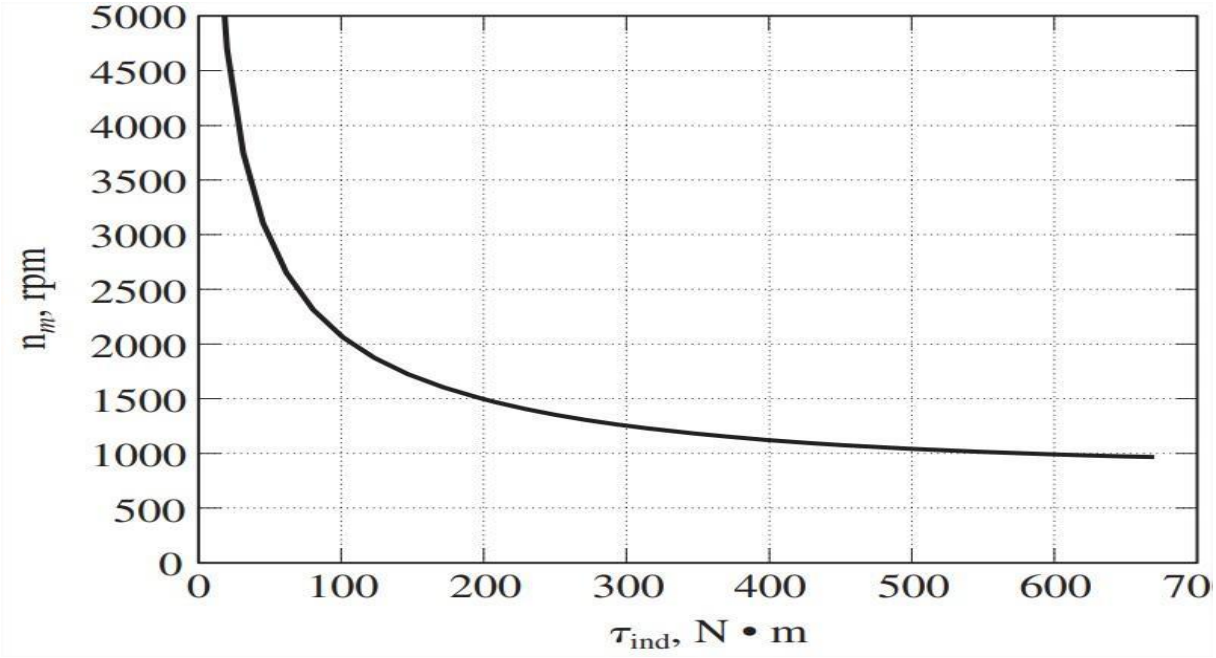


Figura 19- Curva de resposta do motor CC série, característica de conjugado X velocidade, plotada no matlab (Chapman, 2012).

Podemos reparar uma certa diferença entre as curvas plotadas para o caso da topologia série no simulador e no matlab, mas tal fato é devido à configuração de exibição dos gráficos, visível pelo fato de que o início da curva plotada no matlab ao contrário do final está recortada. Mas vale ressaltar que a curva teórica ideal mostrada na figura exibe a tendência assintótica percebida também na curva simulada com mais evidência de tal comportamento.

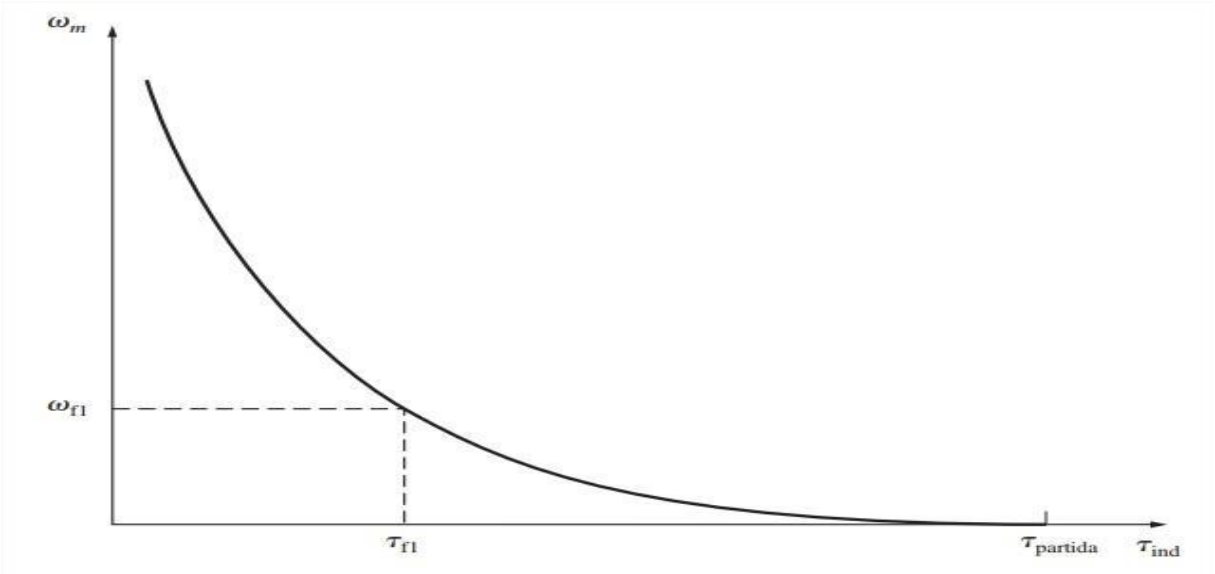


Figura 20- Curva teórica de resposta do motor CC série, característica de conjugado X velocidade (Chapman, 2012).

Vemos acima que o comportamento do motor série é o contrário do motor shunt, pois a cada incremento no valor da corrente de linha tanto a velocidade quanto a característica de conjugado diminui.

A curva de resposta da velocidade em relação a tensão também reforça esse comportamento oposto entre a topologia shunt e série, como podemos observar nas figuras abaixo.

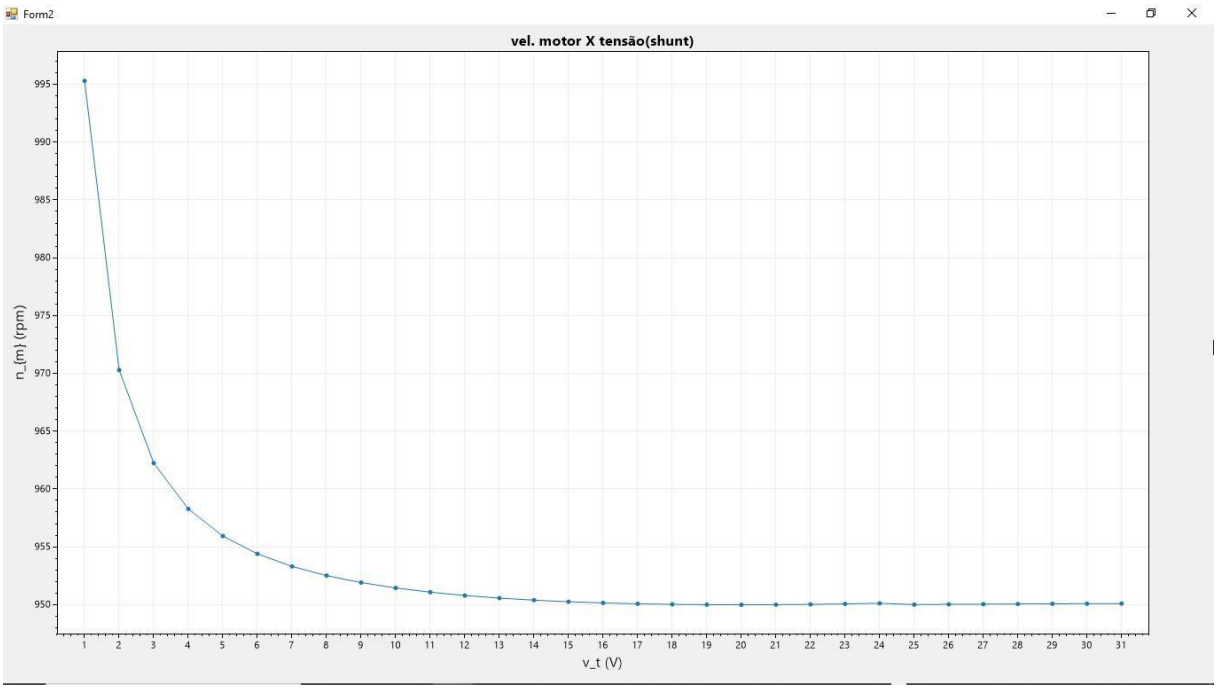


Figura 21- Velocidade em função da variação da tensão na topologia shunt. (Fonte: Autor).

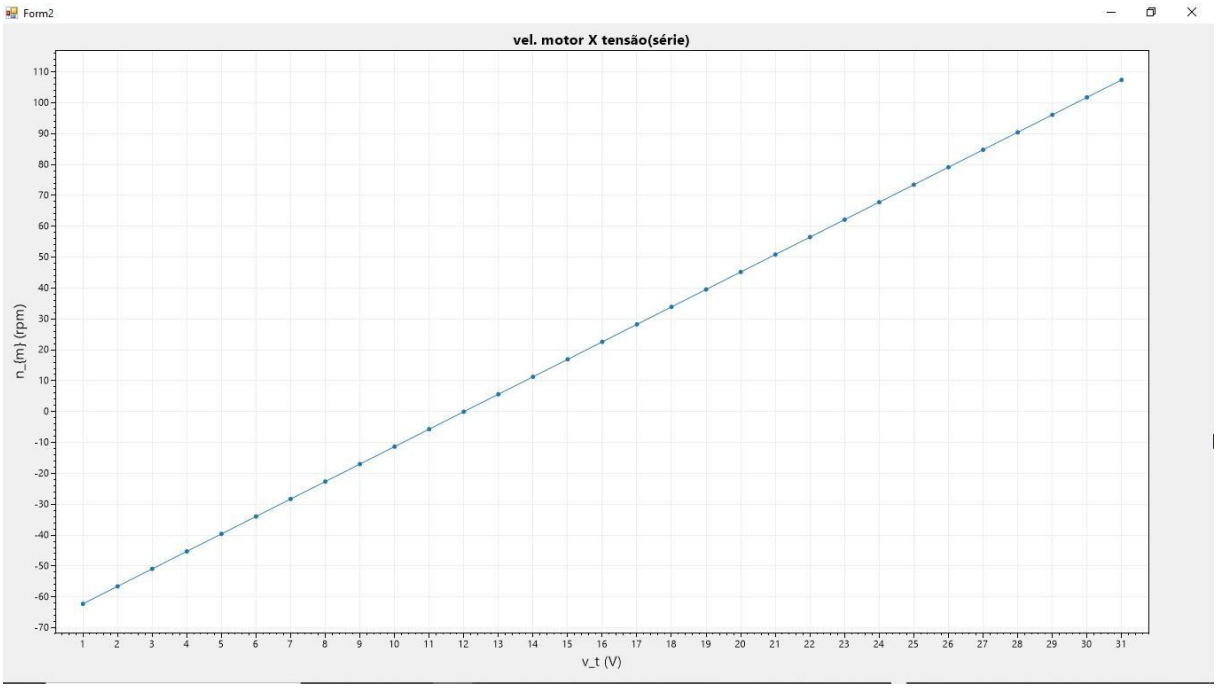


Figura 22- Velocidade em função da variação da tensão na topologia série (Fonte: Autor).

3.3 Motor composto cumulativo e composto diferencial

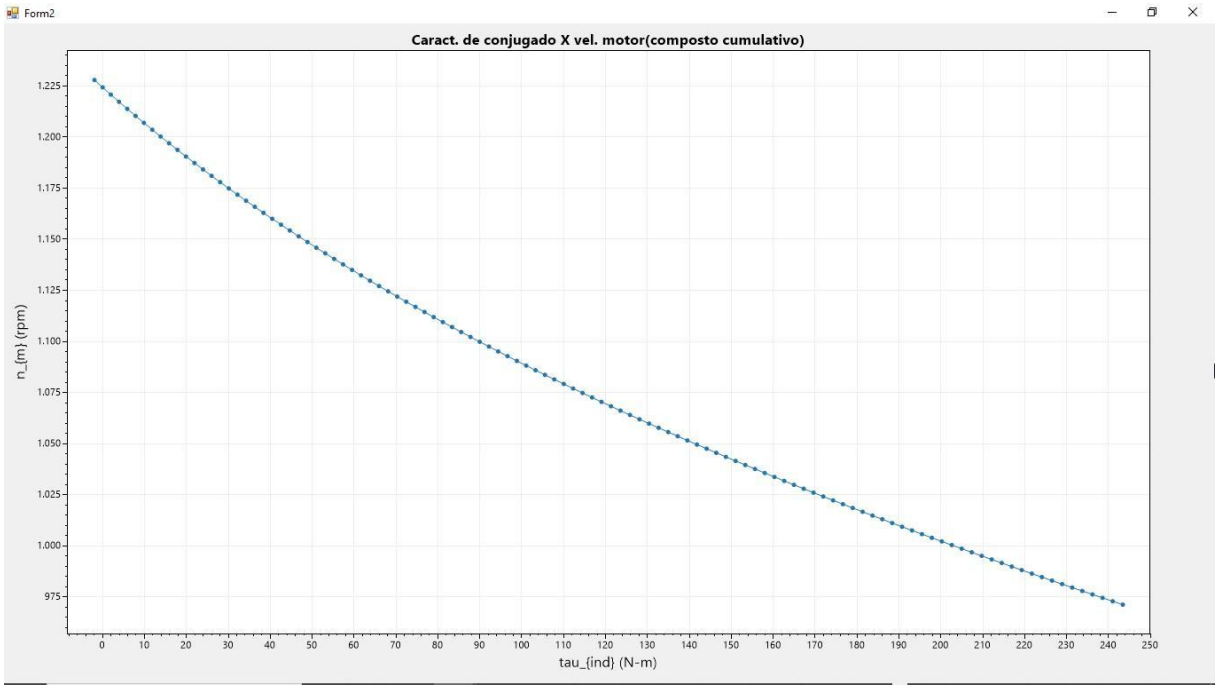


Figura 23- Resposta do motor CC composto cumulativo, característica de conjugado X velocidade, plotada no aplicativo (Fonte: Autor).

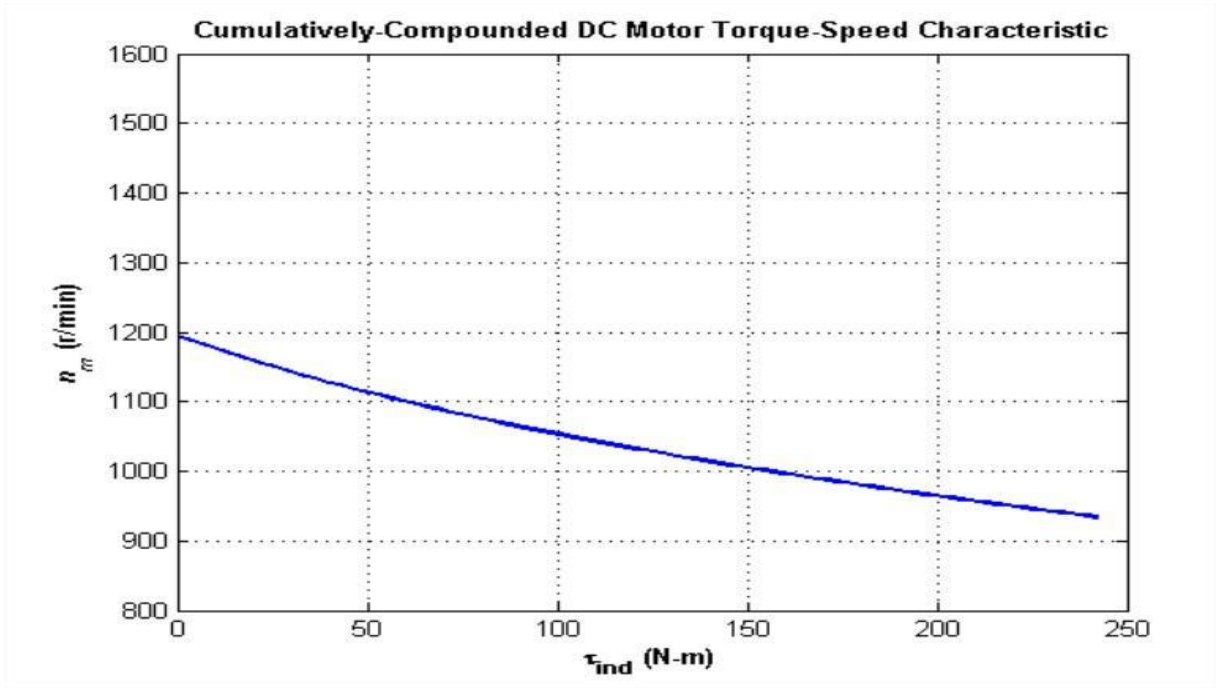


Figura 24- Curva de resposta do motor CC composto cumulativo, característica de conjugado X velocidade, plotada no matlab (Chapman, 2012).

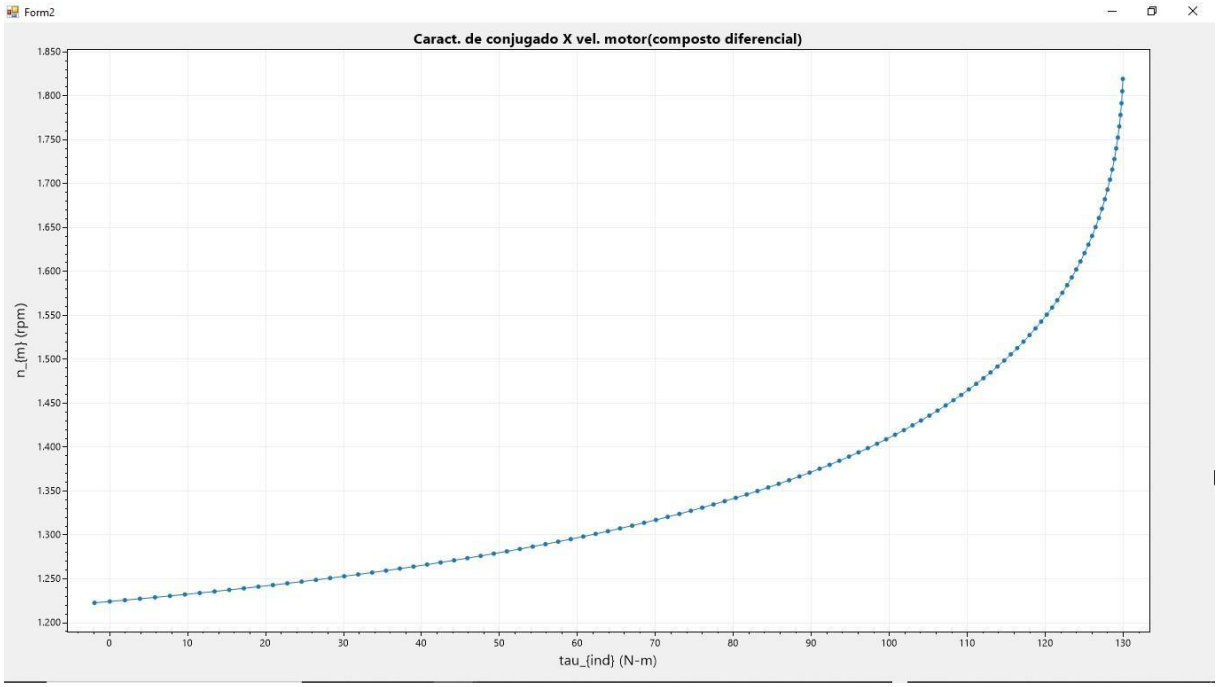


Figura 25- Curva de resposta do motor CC composto diferencial, característica de conjugado X velocidade, plotada no aplicativo. (Fonte: Autor)

Figura 26- Curva de resposta do motor CC composto diferencial, característica de conjugado X velocidade, plotada no matlab (Chapman, 2012).

Como podemos reparar o motor composto cumulativo tem um conjugado de partida mais elevado do que um motor em derivação, em que o fluxo é constante, mas um conjugado de partida mais baixo do que um motor série (cujo fluxo inteiro é proporcional à corrente de armadura). Isso ocorre porque no motor CC composto cumulativo há uma componente de fluxo constante e outra proporcional a corrente de armadura, logo a sua carga.

Como podemos ver pela curva de resposta o motor composto diferencial é instável e com o aumento de sua carga tende a disparar a velocidade, e por isso não tem aplicações.

3.4 Performance ou gasto computacional do aplicativo

O aplicativo tem um custo computacional muito baixo considerando uma máquina com processador Intel Core i7-3610QM CPU @ 2.30GHz, 64 bits com 8.0 GB de memória RAM, no período de espera, que é depois de estar totalmente inicializado é de 0,4 MB. Já para a execução mais complexa disponível que é quando se plota os 4 gráficos conforme figura abaixo o uso de memória chega a aproximadamente no máximo 17 MB, o que é insignificante perante as capacidades de memórias atuais. Em questão de processamento o máximo utilizado foi 6% da capacidade de processamento no caso citado acima.

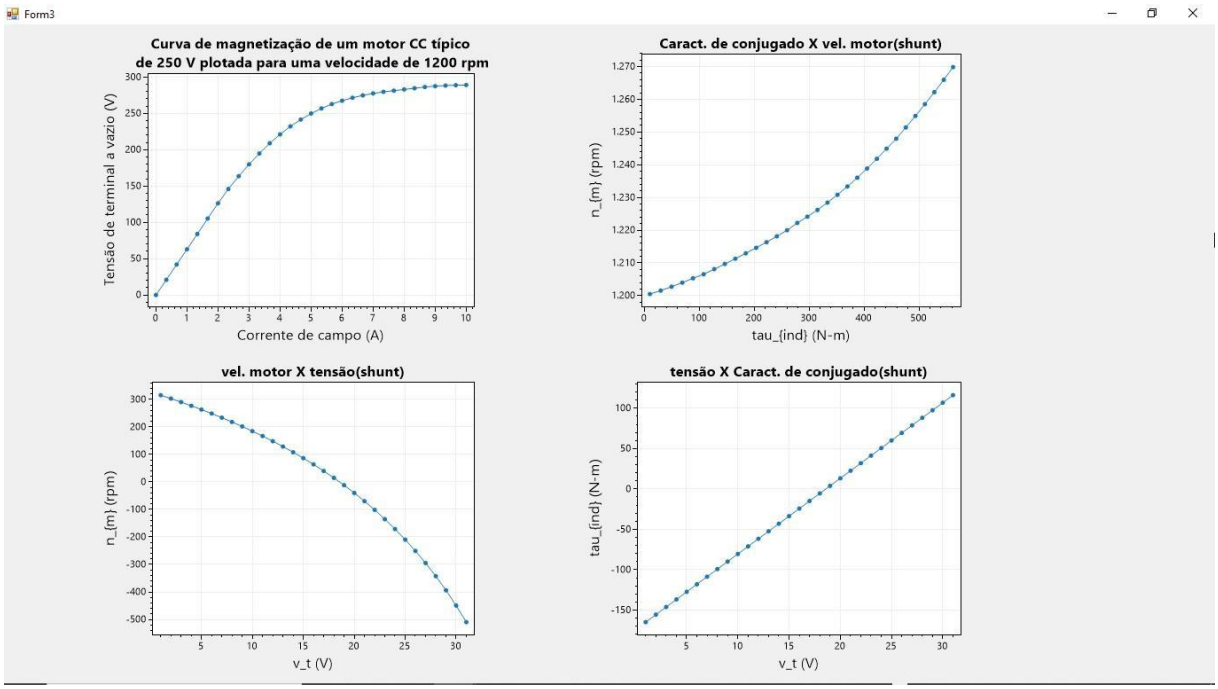


Figura 27- Curvas de resposta do motor CC shunt, plotada no aplicativo. (Fonte: Autor)

3.5 Correlação e desvio padrão entre valores do Matlab e do aplicativo

O Matlab utiliza o cálculo com 16 casas decimais, o aplicativo construído utiliza o tipo double que na linguagem C# apresenta 11 casas decimais. Ao se fazer uma comparação numérica encontramos uma correlação de 0.9999987345184779, já quanto ao erro médio foi encontrado uma discrepância de 0.002671% em média, tanto depois de obter os valores interpolados, quanto ao final do cálculo para encontrar os pontos a serem plotados.

4 Conclusões

Podemos dizer com parcimônia que atendo-se ao proposto inicialmente cumpriu-se de modo satisfatório. E em alguns aspectos até além, como por exemplo a usabilidade do aplicativo quanto a alteração dos valores, parâmetros de entrada. E esse aspecto juntamente com o custo computacional se faz o diferencial de usabilidade, pois além da facilidade de ser executado, também auto preenche as entradas caso não haja valor algum.

No aspecto de processamento interno dos dados e plotagem das curvas de comportamento das topologias pudemos verificar que o resultado foi muito satisfatório, pois a biblioteca utilizada permite que o usuário dê zoom em algum ponto específico de maneira a obter as coordenadas dele.

Toda a estrutura do aplicativo pode ser aproveitada e expandida de modo a plotar outras curvas em algum assunto onde se deseje verificar a diferença entre comportamentos de sistemas. Pois a maneira como foi construída possibilita uma fácil adaptação.

Referências Bibliográficas

CHAPMAN, S. J. **Fundamentos de Máquinas Elétricas**. Porto Alegre: AMGH, 2013.

DEL TORO, V. **Fundamentos de Máquinas Elétricas**. São Paulo: LTC, 1999.

FITZGERALD, A. E. KINGSLEY JR., C.; UMANS, S. D. **Máquinas Elétricas**: Com introdução à eletrônica de potência. Porto Alegre: AMGH, 2014.

KOSOW, I. L. **Máquinas Elétricas e Transformadores**. 14. ed. Rio de Janeiro: Globo, 2000.

CREPPE, R.; SIMONE, G. A. **Conversão Eletromecânica de Energia**. São Paulo: Érica, 2002.

BELHOT, R. V.; FIGUEIREDO, R. S.; MALAVÉ, C. O. **O uso da simulação no ensino de engenharia**. Congresso Brasileiro de Educação em Engenharia, 2001.

Albahari, B.; Drayton, P.; Merril, B. **C# Essentials** 2.ed. Sebastopol: O'Reilly, 2002.

Dos Santos, L. C. **Microsoft Visual C# 2010 Express**: Aprenda a Programar na Prática. 1.ed. São Paulo: Editora Érica Ltda, 2011.

Klein dos Santos, Alexandro. **«Os IDE's (Ambientes de Desenvolvimento Integrado) como ferramentas de trabalho em informática»** (PDF). Universidade Federal de Santa Maria. Consultado em 5 de maio de 2016

↑ Casavella, Eduardo. **«Ambientes Integrados de Desenvolvimento em Linguagem C». Intellectuale.** Consultado em 5 de maio de 2016

Microsoft (26 de setembro de 2020). **Windows forms.** Fonte: docs.microsoft *https://docs.microsoft.com/pt-br/dotnet/desktop/winforms/overview/?view=netdesktop-5.0*

Microsoft ((1 de julho de 2017)). **C#.** Fonte:docs.microsoft *https://docs.microsoft.com/pt-br/dotnet/csharp/language-reference/language-specification/introductio*n.