UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE ENGENHARIA
CÂMPUS DE ILHA SOLTEIRA

MATEUS VALIM REIS

Análise de Desempenho de Algoritmos para Estimação de Fasores em Unidades de Medição nos Sistemas de Energia Elétrica

Ilha Solteira
2023

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de Ilha Solteira

MATEUS VALIM REIS

Análise de Desempenho de Algoritmos para Estimação de Fasores em Unidades de Medição nos Sistemas de Energia Elétrica.

Trabalho de conclusão de curso apresentado à Faculdade de Engenharia de Ilha Solteira – Unesp como parte dos requisitos para obtenção do título de Engenheiro Eletricista.

Nome do orientador
Prof. Dr. Jonatas Boas Leite

Ilha Solteira
2023

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

Reis, Mateus Valim.
R375a Análise de desempenho de algoritmos para estimação de fasores em unidades de medição nos sistemas de energia elétrica / Mateus Valim Reis. -- Ilha Solteira: [s.n.], 2023
43 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2023

Orientador: Jonatas Boas Leite

Inclui bibliografia

1. Medição de fasores. 2. Algoritmo de estimação fasorial. 3. Transformada discreta de fourier. 4. Método dos mínimos quadrados.

Amanda Sertori dos Santos
Bibliotecária - CRB/8-9061
Seção Técnica de Referência, Atendimento ao Usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação

unesp
UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

ATA DE DEFESA DE TRABALHO DE GRADUAÇÃO

Aos vinte e quatro dias do mês de novembro do ano de dois mil e vinte e três, o discente **Mateus Valim Reis,** matriculado sob o nº 161052517, tendo como banca examinadora o seu orientador, o Prof. Dr. Jonatas Boas Leite, o Doutoranda Elis Caroline de Souza Trindade e o Mestrando Vagner Antonio de Moraes da Cruz, apresentou o Trabalho de Graduação intitulado **"Análise de Desempenho de Algoritmos para Estimação de Fasores em Unidades de Medição nos Sistemas de Energia Elétrica",** obtendo a nota 10,0 (DEZ) e conceito APROVADO.

Jonatas Boas Leite
Prof. Dr. Jonatas Boas Leite
- Orientador-

Mateus Valim Reis
Mateus Valim Reis
- Discente-

Elis Caroline de Souza Trindade
Doutoranda Elis Caroline de Souza Trindade
- Membro da Banca -

Mestrando Vagner Antonio de Moraes da Cruz
- Membro da Banca -

DEDICATÓRIA

Neste momento de realização e conquista, quero dedicar meu trabalho de conclusão de curso a vocês, que sempre foram minha fonte de inspiração e apoio inabalável. Em especial, quero homenagear a memória de meu amado pai, cujo amor, sabedoria e encorajamento moldaram meu caminho acadêmico e pessoal.

Cada página deste TCC é um reflexo do apoio e amor que recebi ao longo desta jornada. Cada desafio superado foi possível graças à força que encontrei em cada um de vocês. E, embora meu pai já não esteja fisicamente presente, seu legado de determinação e paixão pela educação vive através de mim e deste trabalho.

Este TCC é um tributo ao nosso amor de família, à dedicação incansável de meus pais e meus irmãos ao legado que meu pai deixou. Agradeço a todos vocês por serem minha âncora nos momentos difíceis e minha celebração nos momentos de alegria.

AGRADECIMENTOS

Agradeço a Deus, pela minha vida e por ter possibilitado conviver com pessoas incríveis que me incentivaram muito pela busca do conhecimento, agradeço a Deus também pela saúde e também por ter me fortalecido nos piores momento. Agradeço a minha família de modo especial o meu pai Claudio Roberto que não está mais ao meu lado mas que tenho certeza que sempre olhou por mim, minha mãe Claudete que sempre orou muito por mim, aos meus irmãos, Lorena e Maxuel, meus cunhados Haiely e Cassiano e meus sobrinhos Lucas e Rafael que compõe a família maravilhosa que eu tenho.

Agradeço a Unesp e seus funcionários e professores de modo especial o Professor Dr. Jonatas Boas Leite por ter me orientado e por toda paciência que ele teve comigo.

E por fim agradeço todos meus amigos, da faculdade e da vida que me apoiaram em todos os momentos e incentivaram a terminar esse ciclo da melhor forma possível.

RESUMO

As unidades de medição fasorial (PMUs) são caracterizadas por sua capacidade de avaliar os fasores. As medidas desses fasores são então utilizadas em várias situações, como análise de sistema de potência, análise de estabilidade, estimativa de estado e projeto de esquemas de proteção de sistema de potência. Sincrofasores são fasores medidos no mesmo instante de tempo. Também é necessária a medição fasorial dinâmica para o funcionamento adequado de alguns equipamentos no sistema de energia elétrica. Componentes baseados em eletrônica de potência, como o sistema de transmissão de corrente alternada flexível (FACTS), precisam do conceito de fasores variáveis no tempo para serem usados na análise. A estimativa de fasores é significativamente usada em aplicações de monitoramento, proteção e controle de área ampla (WAMPAC) nos sistemas de energia. Portanto, há uma necessidade significativa de uma rápida e precisa estimação dos fasores capaz de auxiliar em uma tomada de decisão mais consistente relacionada à estabilidade e dinâmica de sistemas de potência. Este trabalho de graduação propõe, assim, a avaliação de desempenho em estado permanente de três algoritmos de estimação fasorial: transformada discreta de Fourier (DFT) recursiva; DFT não recursiva; e mínimo quadrado. O algoritmo DFT recursivo apresentou o melhor desempenho na estimação fasorial nos testes realizados.

Palavras-chave: Medição de fasores, algoritmo de estimação fasorial, transformada discreta de Fourier, Método dos mínimos quadrados.

ABSTRACT

Phasor measurement units (PMUs) are characterized by their ability to evaluate phasors. Measurements of these phasors are then used in various situations such as power system analysis, stability analysis, state estimation, and design of power system protection schemes. Synchrophasors are phasors measured at the same instant of time. Dynamic phasor measurement is also necessary for the proper functioning of some equipment in the electrical power system. Power electronics-based components, such as flexible alternating current transmission system (FACTS), need the concept of time-varying phasors to be used in analysis. Phasor estimation is significantly used in wide area monitoring, protection and control (WAMPAC) applications in power systems. Therefore, there is a significant need for fast and accurate phasor estimation capable of assisting in more consistent decision making related to the stability and dynamics of power systems. This undergraduate work therefore proposes the evaluation of the steady-state performance of three phasor estimation algorithms: recursive discrete Fourier transform (DFT); non-recursive DFT; and least square. The recursive DFT algorithm showed the best performance in phasor estimation in the tests carried out.

Keywords: Phasor measurement, phasor estimation algorithm, discrete Fourier transform, Least squares method.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

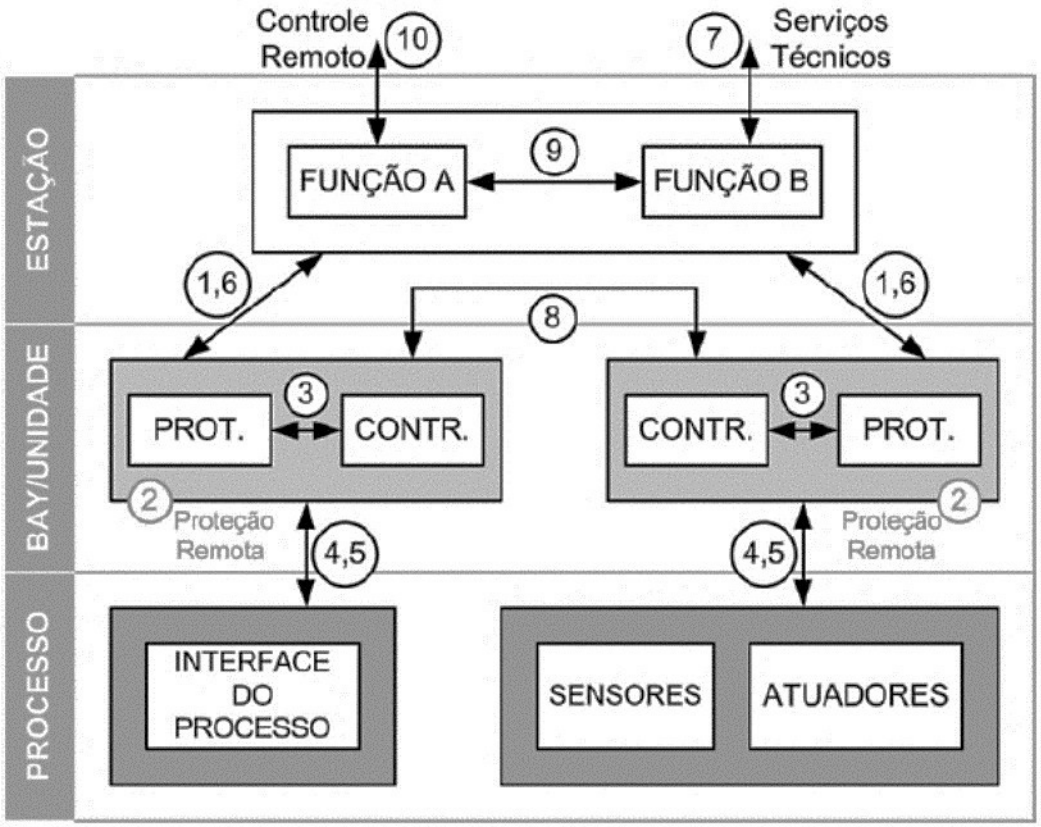
Transformadores convencionais de instrumentação, como transformadores de potencial (TPs) e transformadores de corrente (TCs), medem as altas tensões e correntes, respectivamente, que passam pelo circuito primário. Cabos de cobre conectam a saída analógica dos transformadores ao equipamento secundário, e o número de cabeamentos aumenta dependendo da aplicação (Ekanayake, et al, 2012).

Uma subestação digital faz parte do sistema secundário, incluindo todos os sistemas de proteção, controle, medição, monitoramento de condição, registro e supervisão associados ao processo primário. Nas subestações digitais, centenas (às vezes milhares) de metros de cabeamento de cobre, entre o pátio de manobras e os dispositivos eletrônicos inteligentes (IEDs) na sala de controle, são substituídos por alguns metros de cabos de fibra óptica. Usar menos cobre torna a subestação digital mais simples, mais compacta e mais eficiente (Northcote-Green & Wilson, 2017).

Conforme definido na norma 61850 do comitê internacional eletrotécnico, do Inglês *International Electrotechnical Committee* (IEC), a arquitetura digital da subestação compreende três níveis: um nível de processo, um nível de bay/unidade e um nível de estação, conforme mostrado na Figura 1. Cada um dos níveis desempenha funções específicas e as aplicações trabalham em conjunto para realizar as funcionalidades da subestação digital (IEC 61850-1, 2003).

Os transformadores de instrumentação, ou sensores, como TPs e TCs, e os equipamentos de comutação, ou atuadores, fazem parte do nível de processo. Esse nível atua como a ligação entre o equipamento primário e o secundário, que desempenham funções de proteção e controle. Em uma subestação tradicional, essa interface é estabelecida por meio de cabeamento de cobre, onde correntes e tensões são direcionadas para painéis de proteção e controle em níveis de sinais secundários padronizados aceitos, e os cabos de controle são usados para transmitir informações de *status*. Na subestação digital, ocorre a conversão de todos os dados, independentemente de serem analógicos ou binários, para o formato digital nas proximidades do equipamento condutor ou da fonte de medição, esses dados são enviados através de fibra ótica usando o protocolo IEC 61850-9-2.

Figura 1 - Arquitetura da subestação digital



Fonte: (LEITE, 2020).

O nível do bay engloba dispositivos secundários, conhecidos como IEDs, que incluem controladores de bay, relés de proteção, registradores de falhas e medidores de energia. Os IEDs dispensam entradas analógicas pois a coleta de dados se dá no nível do processo. Com a síntese de entradas, as entradas binárias se tornam menos necessárias ou até desnecessárias, permitindo, assim, que os aparelhos sejam mais compactos e ocupem apenas metade do espaço tradicional. Os IEDs processam os algoritmos e a lógica de proteção e controle, tomando decisões de *trip*/*notrip*, e também disponibilizam recursos de comunicação baseados no IEC 61850 para as redes *Ethernet* dos níveis inferior (processo) e superior (estação). A rede de comunicação tem redundância como uma exigência usual, garantindo a máxima disponibilidade e confiabilidade.

O nível da estação engloba elementos como computadores da estação, *switches* Ethernet e *gateways*. O barramento da estação oferece recursos de comunicação extras além do barramento do sistema de controle e aquisição de dados (SCADA), do Inglês *supervisory control and data acquisition*, que controla e adquire

dados, pois possibilita que diversos clientes compartilhem dados; suporta a comunicação entre dispositivos ponto a ponto; e conecta *gateways* em esquemas de comunicação entre subestações e a rede de área ampla. Os componentes presentes no nível da estação podem incluir interfaces homem-máquina (IHMs) para controle local da subestação, estações de trabalho de engenharia para acesso aos IEDs ou concentradores locais e sistemas de arquivamento de dados relacionados ao sistema de energia. Além disso, podem estar presentes *gateways* SCADA, *links* de servidor *proxy* para IHMs remotas ou controladores.

Os IEDs modernos integram diversas funções de proteção com medição, registro e monitoramento. Eles oferecem uma ampla gama de recursos para medição trifásica e parâmetros monofásicos. Eles também podem agregar outras funções, como fornecer as informações dos estados de disjuntores, chaves e medição sincronizada de fasores (PMU), do Inglês *phasor measurement unit*.

Esses novos dispositivos e funcionalidades têm contribuído para a rápida evolução da automação nas redes de distribuição de energia elétrica que tem se convertido em um tema de significativo interesse para a indústria do setor elétrico. O progresso do conceito de automação requer, muitas vezes, a formulação de novas tecnologias e recursos computacionais que permitam viabilizar a simulação em tempo real das redes de distribuição, acompanhadas de técnicas de *hardware-in-the-loop* (HIL), por exemplo, visando à redução dos prazos de projeto e testes, com custos mais acessíveis (Leite, 2020).

Simuladores em tempo real representam, de forma geral, ferramentas de imenso potencial para a análise, planejamento, controle, segurança e operação dos sistemas elétricos de energia. Uma característica comum dos simuladores digitais em tempo real consiste em sua capacidade de executar processamento paralelo para simular transitórios eletromagnéticos em extensas redes elétricas. Seus *designs* abrem oportunidades para uma ampla gama de testes em dispositivos elétricos, como relés digitais de proteção, IEDs e sistemas de controle. No entanto, a adoção dessa técnica ainda enfrenta grandes desafios pelo fato de se ter associado aos simuladores um elevado custo de fabricação, bem como às suas dimensões consideráveis, o que dificulta a mobilidade e o acesso (Leite, 2020).

A distinção entre um simulador HIL e os *softwares* de simulação convencionais consiste na capacidade dos simuladores HIL de gerar e medir sinais que podem atuar com um dispositivo externo, normalmente em fase de teste, por exemplo em

dispositivos de borda com modelos lógicos através dos padrões IEC 61970/61968 e 61850 (Leite & Kezunovic, 2023). Conforme definido por esses protocolos, a redelocal Ethernet deve assegurar a transmissão de mensagens críticas em termos de tempo, como os eventos orientados a objetos genéricos (GOOSE), do Inglês *genericobject oriented substation event*, e os valores brutos amostrados de dados (SV), do Inglês *sampled values*, dentro dos limites de tempo aceitáveis (Leite, 2020). Essas mensagens estão associadas, respectivamente, às interfaces 8 no nível de bay, e 4 e 5 entre o nível de processo e bay da arquitetura da subestação digital, conforme é mostrado na Figura 1.

Na subestação digital, entre os IEDs no nível de bay também é comum a transmissão de fasores através da função de PMU. A medição de fasores é uma prática essencial na engenharia, estruturada na representação complexa de sinais senoidais. As grandezas elétricas, como corrente e tensão alternada, são comumente modeladas como fasores complexos para simplificar a analise matemática, compreensão das suas componentes elétricas e a compreensão do sistema elétrico como um todo. A utilização de fasores desempenha um papel primordial na solução de problemas complexos em áreas como sistemas de potência, eletrônica, controle de processos industriais e telecomunicações (J Duncan Glover; Overbye; Sarma, 2017).

Para se representar um sinal senoidal a partir de um fasor é necessário encontrar parâmetros essenciais como a amplitude (magnitude), a fase e a frequência. Esses dados são encontrados por meio de dispositivos de medição especializados e processadas para análise. Além disso, a representação complexa dos fasores simplifica a realização de operações matemáticas em sistemas elétricos, tornando-a uma ferramenta indispensável para engenheiros e profissionais da área (Phadke, 2017).

Alguns conceitos são essenciais para se se medir e representar um fasor de maneira adequada, sendo um deles a transformada de Fourier que é uma aliada essencial e normalmente empregada na engenharia e na análise de sinais. Sua principal aplicação consiste em converter os sinais do domínio do tempo para o domínio da frequência. Esta transformada desmembra sinais complexos em suas componentes fundamentais de frequência, o que permite um estudo detalhado das grandezas elétricas e de outros fenômenos em termos de amplitude, fase e frequência. Ao decompor o sinal, a transformada de Fourier possibilita uma

compreensão mais profunda das características intrínsecas dos sinais, facilitando a detecção de padrões, a identificação de componentes harmônicas e a resolução de problemas em diversas áreas (J Duncan Glover; Overbye; Sarma, 2017).

Os sinais senoidais são amplamente representados como números complexos, o que simplifica a manipulação das operações matemáticas envolvidas em sua análise. Utilizando essa técnica a amplitude do sinal é representada pela parte real, enquanto a fase é representada pela parte imaginária. Essa prática facilita o acesso de algumas informações além das várias equações que se deseja realizar com os fasores.

1.1 OBJETIVOS

Assumindo a estrutura da subestação digital, este trabalho de graduação tem como objetivo principal implementar e comparar três algoritmos de estimação fasorial. Dois dos algoritmos estão fundamentados na transformada de Fourier, sendo eles o DFT recursivo e o DFT não-recursivo. O terceiro algoritmo faz a estimação dos fasores através dos mínimos quadrados. Os algoritmos são comparados em relação a sua precisão, quando se calcula magnitude e ângulo dos fasores, e também em relação ao tempo de processamento de cada método.

1.2. Estrutura do Trabalho de Graduação

Após a contextualização do trabalho de graduação neste primeiro capítulo, os capítulos seguintes estão estruturados da seguinte forma.

No capítulo 2 é apresentado a fundamentação teórica e os modelos matemáticos dos algoritmos estudados para estimação fasorial.

No Capítulo 3 são apresentados os códigos em linguagem de progração usados na implementação dos algoritmos.

No Capítulo 4 é verificado desempenho dos algoritmos implementados tanto na precisão dos fasores estimados quanto na tempo de processamento.

No Capítulo 5 são apresentadas as concluões.

2 MEDIÇÃO E ESTIMAÇÃO DE FASORES

Para realização desse trabalho de graduação é proposta uma metodologia dividida em três partes, sendo que a primeira descreve como obter as medidas dos parâmetros elétricos (tensão e corrente) no domínio do tempo usando dispositivos eletrônicos no nível de processo como especificado pela norma IEC 61850. A segunda parte consistiu na formulação matemática do algoritmo para estimar os fasores dos sinais no domínio do tempo encontrados na primeira parte. A terceira parte consiste na implementação ativa do algoritmo e análise comparativa entre os três métodos estudados. Neste capítulo, são apresentadas a primeira e segunda parte da metodologia.

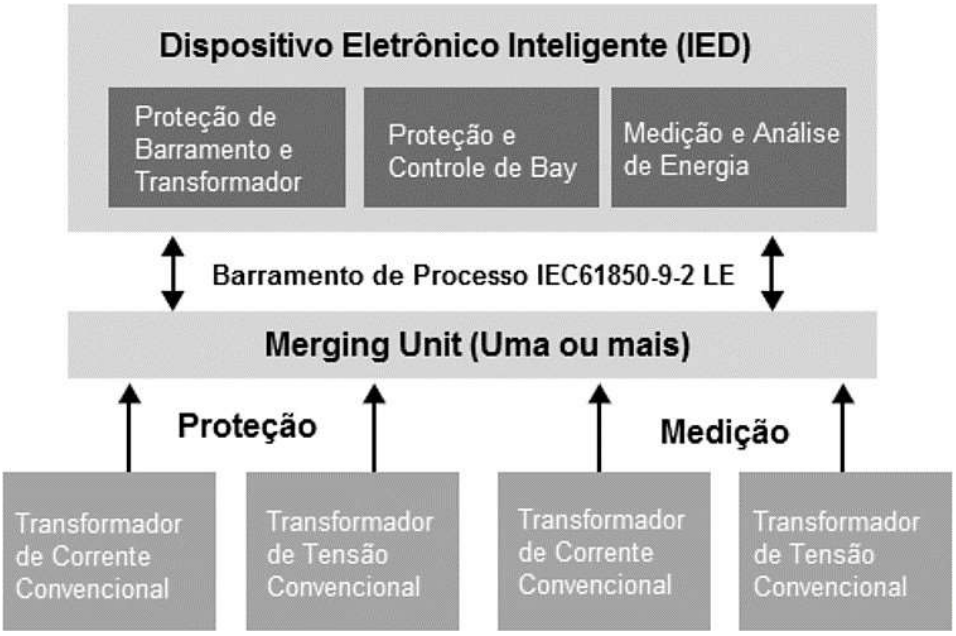
2.1 MEDIÇÕES DOS PARÂMETROS ELÉTRICOS

Uma *merging unit* (MU) transforma as saídas do transformador de instrumento em uma saída de dados padronizados baseada em Ethernet, isso viabiliza a aplicação das diretrizes estabelecidas pelo IEC 61850. Em subestações modernas, ou digitais em vez de direcionar diretamente as saídas dos sensores, como TPs ou TCs, para os dispositivos de proteção e controle situados no nível "bay", opta-se por posicionar MU em proximidade aos sensores que estão conectados aos equipamentos primários, operando no nível do processo (Wang, 2017).

Através da MU, os sinais analógicos que representam os parâmetros elétricos são convertidos em valores amostrados, seguindo as diretrizes do padrão IEC 61850-9-2, e são então empregados para as finalidades de proteção, medição e controle, estabelecendo comunicação digital com os IEDs presentes na subestação, conforme ilustrado na Figura 2.

Algumas das principais características da MU, incluem a conversão da informação analógica para formato digital, re-amostragem dos dados, sincronização com um relógio de referência global, adaptação das amostras ao protocolo IEC 61850-9-2 e estabelecimento de comunicação com os IEDs através de uma interfaceEthernet baseada em fibra óptica.

Figura 2 - MU conectadas a transformadores convencionais.



Fonte: (Leite, 2020).

As MUs desempenham uma função essencial para gerar um fluxo de dados de saída que esteja perfeitamente alinhado com o tempo das amostras, seguindo o padrão IEC 61850-9-2. Este processamento abrange amostragem de valores analógicos; referência exata em tempo real; formatação de mensagem em valores de amostra; e publicação em uma única fonte de dados para equipamentos de medição, proteção e controle.

2.2 ESTIMAÇÃO E ATUALIZAÇÃO DE FASORES NA FREQUÊNCIA NOMINAL

Considerando $x(t)$, um sinal de entrada constante na frequência nominal do sistema de potência f_0, sendo Nf_0 sua frequência de amostragem. O ângulo de amostragem θ é definido como $2\pi/N$, sendo a estimativa do fasor realizada por meio da Transformada Discreta de Fourier (DFT) (Phadke, 2017). Uma vez que o principal objetivo nas medições fasoriais é calcular a componente da frequência fundamental, estabelece-se $k = 1$ nas equações da DFT para gerar o fasor correspondente à frequência fundamental a partir do conjunto de amostras x_n. O sobrescrito (*N – 1*) é empregado para encontrar o fasor como tendo a amostra (*N – 1*)-ésima como a última

amostra usada na estimativa. O cálculo da soma dos cossenos segue a formulação indicada por (01).

$$X_C^{N-1} = \frac{\sqrt{2}}{N}\sum_{n=0}^{N-1} x_n \cos(n\theta) = \frac{\sqrt{2}}{N}\sum_{n=0}^{N-1} x_n \cos(n\theta + \varphi) = \frac{X_m}{\sqrt{2}}\cos(\varphi) \quad (01)$$

A soma dos senos por sua vez é calculada de forma semelhante por (02).

$$X_S^{N-1} = \frac{\sqrt{2}}{N}\sum_{n=0}^{N-1} x_n \operatorname{sen}(n\theta) = \frac{\sqrt{2}}{N}\sum_{n=0}^{N-1} x_n \operatorname{sen}(n\theta + \varphi) = \frac{X_m}{\sqrt{2}}\operatorname{sen}(\varphi) \quad (02)$$

O fasor X^{N-1} é calculado por (03).

$$X^{N-1} = X_C^{N-1} - jX_S^{N-1} = \frac{X_m}{\sqrt{2}}[\cos(\varphi) + jsen(\varphi)\} = \frac{X_m}{\sqrt{2}}e^{-j\varphi} \quad (03)$$

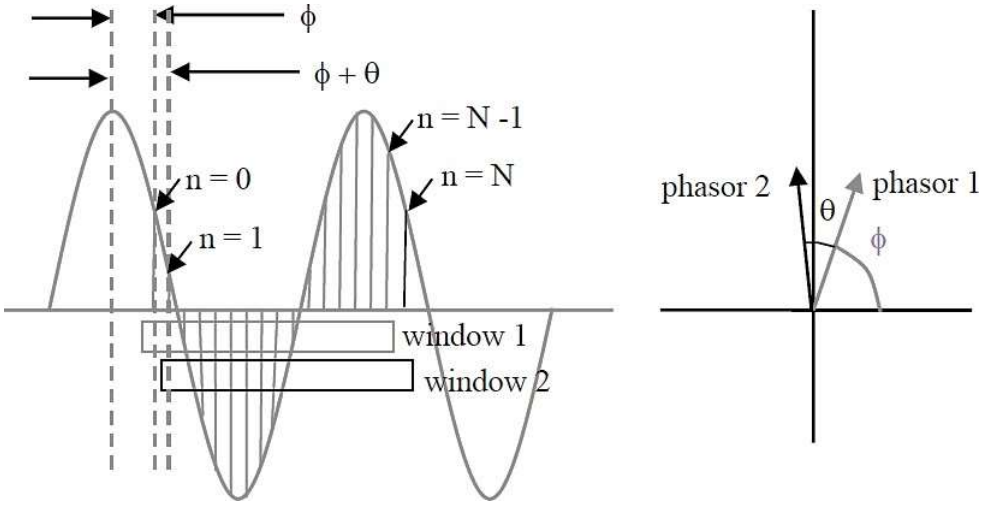
É importante notar que (03) fornece a estimativa do fasor na frequência fundamental, embora o subscrito $k = 1$ tenha sido omitido por questões de simplicidade. O resultado obtido segue a definição do fasor, e o ângulo de fase φ do fasor é o ângulo entre o instante em que a primeira amostra é coletada (correspondendo a $n = 0$) e o pico do sinal de entrada.

2.2.1 Atualização não recursiva

Para atualizar a estimativa fasorial de forma contínua, é preciso considerar algoritmos que levem em conta as novas amostras de dados adquiridas. Em um algoritmo não recursivo, a primeira janela de N amostras, denotada por $w_1 = \{x_n | n = 0 \ldots N-1\}$, é usada para estimar o fasor 1. A segunda janela de N amostras é de $w_2 = \{x_n | n = 1 \ldots N\}$, conforme apresentado na Figura 3.

Nesta janela, as leituras anteriores não são ignoradas e novos cálculos são feitos considerando as leituras anteriores. Assim, o fasor 2 estimado fica em um ângulo θ com o fasor 1. Este é um processo estável, mas lento, pois não aproveita os dados estimados da janela anterior. Cada iteração executa todos os cálculos (Phadke, 2017).

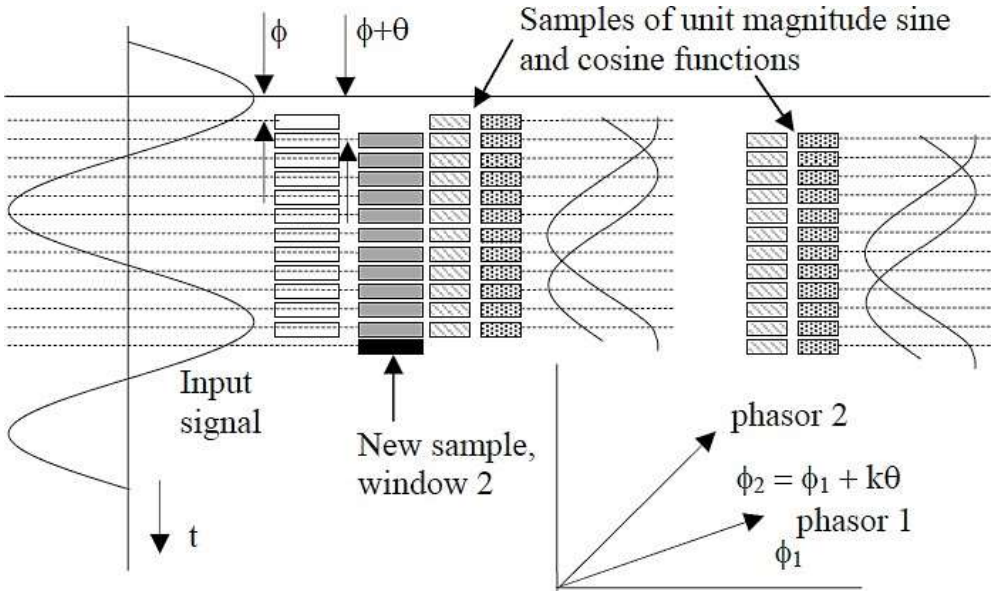
Figura 3 - Atualização de estimativas de fasores com janelas de *N* amostras.



Fonte: (PHADKE, 2017).

Na Figura 4 é apresentada outra visão do processo de estimativa fasorial não recursiva.

Figura 4 – Estimação de fasores não-recursiva.



Fonte: (PHADKE, 2017).

À medida que novas amostras são obtidas, a tabela de multiplicadores de seno e cosseno é movida para baixo para corresponder à nova janela de dados. Na Figura 4, os multiplicadores são vistos como amostras de ondas senoidais e cosseno de unidade de magnitude na frequência nominal do sistema de potência. A nova janela

de dados tem $N - 1$ amostras em comum com a janela de dados antiga. Na computação real, eles são simplesmente armazenados como tabelas de seno e cosseno, que são usadas repetidamente em cada janela, conforme necessário.

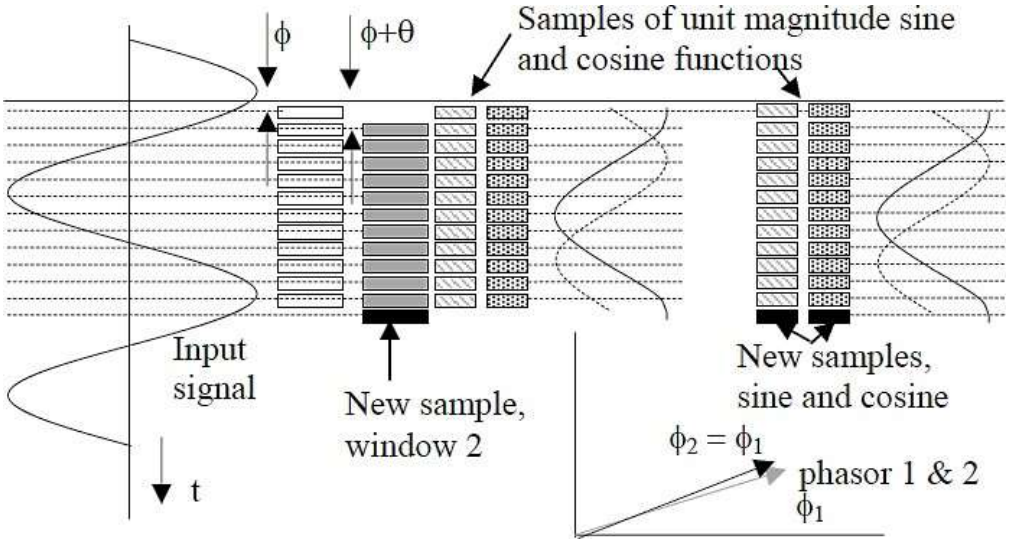
2.2.2 Atualização recursiva

Um processo de atualização recursiva envolve a modificação dos valores dos fasores anteriores, ou seja, as estimativas dos fasores da janela anterior são obtidas e ajustadas para uma nova janela, em contraste com o método não recursivo, onde apenas incorporamos o novo valor da amostra, no processo recursivo, as amostras mais antigas são retiradas da janela anterior. Isso o faz muito mais rápido do que o método não recursivo. Porém, este método pode ser instável, pois qualquer erro nos cálculos anteriores é propagado. Se $(N + r)$ representa a última amostra desta janela, então a estimativa fasorial pode ser expressa como:

$$X^{N+r} = X^{N+r-1} + \frac{\sqrt{2}}{N}(x_{N+r} - x_r)e^{-j(r)\theta} \quad (04)$$

Na Figura 5 é apresentada uma visão do processo de estimativa fasorial recursiva.

Figura 5 – Estimação de fasores recursiva.



Fonte: (PHADKE, 2017).

Quando o sinal de entrada é uma senóide constante, $x_N + r$ é igual a x_r, dessa forma, a estimativa fasorial com dados da nova janela é a mesma que a estimativa fasorial com dados da janela.

2.2.3 Método do mínimo quadrado

O método de mínimos quadrados (LSE), do Inglês, *least square estimation*, é uma abordagem de ajuste de curva amplamente adotada e proeminente na comunidade acadêmica, muito pela facilidade de entendimento do método. Utilizando esse método é possível estimar o fasor com menos leituras, ou amostras (Sachdev& Nagpal, 1991). A equação (05) representa um sinal senoidal.

$$x(t) = X_m sen(\omega t + \varphi) = X_d sen(\omega t) + X_q \cos(\omega t) \quad (05)$$

Considerando $X_q = X_M sen(\varphi)$ e $X_d = X_M cos(\varphi)$ o valor da n-ésima é dado por (06).

$$x_n = [sen(n\omega\Delta t) \quad cos(n\omega\Delta t)] \begin{bmatrix} X_d \\ X_q \end{bmatrix} \quad (06)$$

Considerando N equações e N amostras no tempo específico Δt, é possível se obter.

$$[\boldsymbol{x}_n]_{Nx1} = [\boldsymbol{A}]_{Nx2}[\boldsymbol{X}]_{2x1} \quad (07)$$

Para $N > 2$, tem-se:

$$\boldsymbol{X} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{x}_n \quad (08)$$

3 O ALGORITMO

Nesse capítulo é descrita implementação dos três algoritmos estudados, algoritmo não-recursivo, recursivo e mínimos quadrados, assim como o algoritmo principal usado para a leitura do sinal de entrada. Esses algoritmos sãoimplementados em linguagem de progrmação C-Sharp (C#).

3.1 O ALGORITMO PRINCIPAL

O algoritmo principal é a base para a leitura do sinal de entrada que é obtido através de um arquivo XML, e também para a impressão dos resultados finais, tais como magnitude e ângulo de fase para cada fasor estimado usando os métodos propostos. Na Figura 6 é mostrada parte do código fonte do algoritmo principal que é responsável pela leitura das amostras dos sinais de entrada.

Figura 6 - Algoritmo de leitura.

```
//Código.
bool nao_recursiva = false;
bool recursiva = true;
bool minquadrado = false;
// Variáveis declaradas.
double[] corrente_instantanea; // Vetor com as medições de corrente instantânea.
int N; // Quantidade de amostras por ciclo.
double NP = 10000; // Quantidade de pontos usados para se calcular o tempo de execução.
double teta; // Ângulo.
double somaC, XC; // Soma do cosseno.
double somaS, XS; // Soma de seno.
double[] X, phi; // Módulo e ângulo estimado do fasor.

XmlDocument doc = new XmlDocument();
doc.Load(System.IO.Directory.GetParent(@"../../").FullName + "\\CorrenteInstantanea.xml");

XmlNode nodelist = doc.SelectSingleNode("InstantaneousMagnitude");
corrente_instantanea = new double[nodelist.ChildNodes.Count];
X = new double[corrente_instantanea.Length];
phi = new double[corrente_instantanea.Length];
N = Convert.ToInt32(nodelist.Attributes["N"].Value);
for (int i = 0; i < corrente_instantanea.Length; i++)
{
    corrente_instantanea[i] = Convert.ToDouble(nodelist.ChildNodes[i].Attributes["InstMag"].Value);
}
```

Fonte: Produzida pelo próprio autor

Nas primeiras linhas de códigos pode se perceber três variáveis booleanas, essa estrutura tem como finalidade alternar entre os métodos não-recursivo, recursivo e mínimos quadrados. Quando se é atribuído o valor *false* para as variáveis bolenas, ou seja é como se fosse a entrada 0 (zero), e quando se atribui *true*, ou seja é como

fosse a entrada 1 (um). Na ativação de um método é atribuida uma entrada *true* para o seu boleano correspondente e *false* para os outros.

Depois de se escolher o método é a etapa de se declarar as variáveis que são usadas ao longo de todas as operações, dentre todas as variáveis declaradas vale destacar a *NP*, que é o número de repetições usados para se calcular o tempo de processamento de cada método. O valor de *NP* é alterado previamente para as simulações. Nesse trabalho é adotado três valores diferentes de *NP*, mil, dez mil e cinquenta mil.

Logo em seguida se observa a estrutura usada para ler o sinal de entrada em formato XML, sendo possível atribuir os valores instantâneos do sinal para a variável *Corrente_Instantanea* e também é possível atribuir à variável *N* o número de pontos do sinal, que neste trabalho é de 50 pontos por ciclo.

Um laço *for{.}* é utilizado para atribuir cada valor instantâneo ao seu devido ponto e, assim, criar o vetor *Corrente_Instantanea*, dessa forma é possível realizar os cálculos para os três métodos utilizados.

Continuando a análise do algoritmo principal, é utilizada uma estrutura para imprimir os resultados obtidos, essa estrutura é observada na Figura 7.

Figura 7 - Estrutura de impressão.

```
using (StreamWriter sw = new StreamWriter(System.IO.Directory.GetParent(@"../../").FullName + "\\Estimativadefasores.txt"))...


Console.Write("O fasor estimado possui o modulo =" + X[49]);
Console.Write("O fasor estimado possui o angulo =" + phi[49]);
Console.ReadLine();
```

Fonte: Produzida pelo próprio autor

3.2 O ALGORITMO NÃO-RECURSIVO

O algoritmo não-recursivo é construído seguindo a lógica matemática descrita anteriormente no capítulo 2, a estrutura de cálculo pode ser observada na Figura 8.

No início do código se atribui 0 para as variáveis *somaC* e *somaS*, para que não haja a possibilidade de que essas variáveis já possuam outros valores atribuídos a elas, logo em seguida dois laços *for{.}* são abertos.

O segundo laço *for{.}* tem a finalidade de realizar a contagem para todos os pontos do ciclo do sinal de entrada, para que estes pontos sejam contabilizados

durante o cálculo e o terceiro laço *for{.}* é responsável por realizar o somatório como dado em (01) e (02).

Figura 8 – Impelementação do algoritmo não-recursivo.

```
if (nao_recursiva)
{

    teta = (2 * Math.PI) / N;
    somaC = 0;
    somaS = 0;

    tempo.Start();
    for (int np = 0; np < NP; np++)
    {
        for (int w = N - 1; w < (corrente_instantanea.Length); w++)
        {
            somaC = 0;
            somaS = 0;
            for (int n = w - (N - 1); n <= w; n++)
            {
                somaC = somaC + (corrente_instantanea[n] * Math.Cos(n * teta));
                somaS = somaS + (corrente_instantanea[n] * Math.Sin(n * teta));
            }
            XC = (Math.Sqrt(2.0) / N) * somaC;
            XS = (Math.Sqrt(2.0) / N) * somaS;

            X[w] = Math.Sqrt((XC * XC) + (XS * XS));
            phi[w] = Math.Atan(XS / XC);
        }
    }
    tempo.Stop();
    Console.WriteLine("Tempo gasto não recursivo : " + (tempo.ElapsedMilliseconds / NP).ToString() + " milisegundos");
}
```

Fonte: Produzida pelo próprio autor

Desse modo, são realizados os cálculos e é possível obter o valor de *X[.]* que é equivalente a magnitude do fasor e o valor de *phi[.]* que é equivalente ao ângulo de fase do fasor.

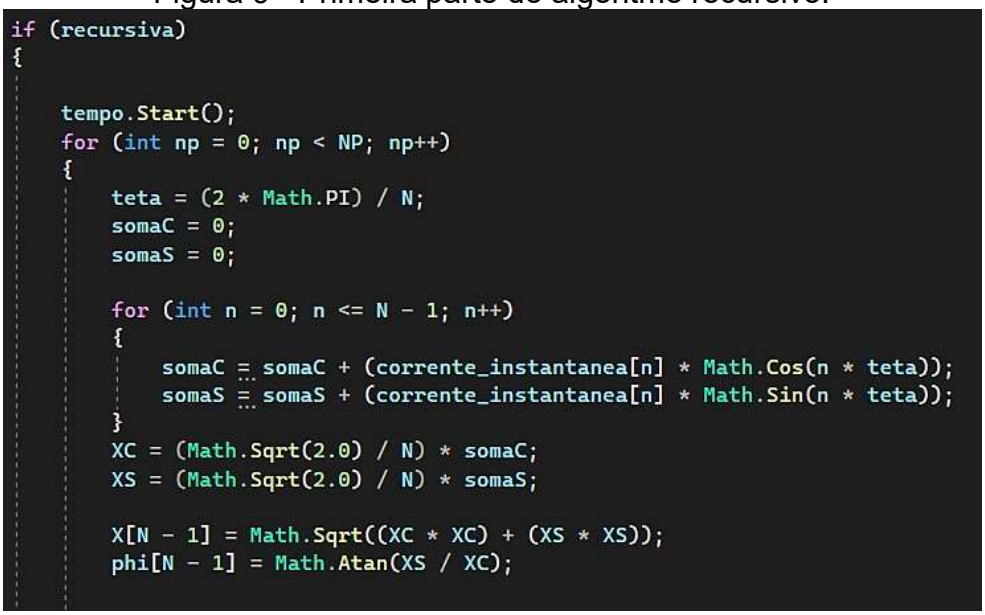
Um contador é inserido no começo e fim de cada estrutura de contagem para obter o tempo de processamento, porém para melhor a precisão nessa cronometragem um linha de código é inserida um laço *for{.}* que permitia realizar a simulação repetidas vezes e se tirar a média dos resultados, a variável que comanda o número de repetições é a *NP*, apresentado anteriormente.

Observando essa estrutura de cálculo, nota-se que os pontos são somados conforme a progressão da janela, o contador *w* representa esse movimento por todas as amostras representadas por *N*, esse fator confere a esse método uma maior estabilidade, porém uma velocidade menor por estar considerando todas as amostras no final da estimativa.

3.3 O ALGORITMO RECURSIVO

Seguindo a lógica matemática descrita no capítulo 3 é construido o algoritmo recursivo e, para o maior entendimento da estrutura, na Figura 9 é ilustrada na primeira parte da sua configuração.

Figura 9 - Primeira parte do algoritmo recursivo.



```
if (recursiva)
{

    tempo.Start();
    for (int np = 0; np < NP; np++)
    {
        teta = (2 * Math.PI) / N;
        somaC = 0;
        somaS = 0;

        for (int n = 0; n <= N - 1; n++)
        {
            somaC = somaC + (corrente_instantanea[n] * Math.Cos(n * teta));
            somaS = somaS + (corrente_instantanea[n] * Math.Sin(n * teta));
        }
        XC = (Math.Sqrt(2.0) / N) * somaC;
        XS = (Math.Sqrt(2.0) / N) * somaS;

        X[N - 1] = Math.Sqrt((XC * XC) + (XS * XS));
        phi[N - 1] = Math.Atan(XS / XC);
```

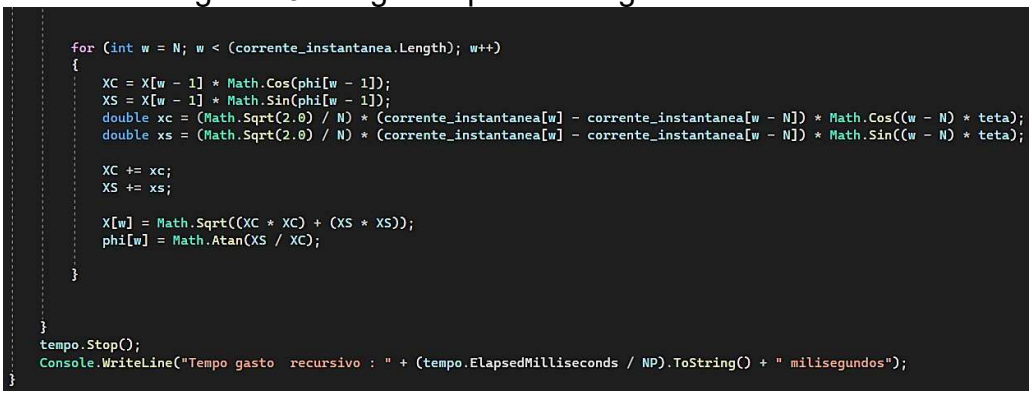
Fonte: Produzida pelo próprio autor

De maneira semelhante ao código do algoritmo não-recursivo, o primeiro laço *for{.}* é introduzido para se medir a média de tempo considerando as repetições como *NP*, também é necessário zerar as entradas de *somaC* e *somaS* pelos mesmos motivos citados na subsseção anterior.

O segundo laço *for{.}* é utilizado para calcular o somatório em (01) e (02) e, igual ao feito no primeiro algoritmo, encontra-se o valor para a magnitude e ângulo do fasor, porém esse valor é apenas considerando a primeira janela.

Na Figura 10 mostra-se o modo para fazer a janela se mover excluindo o último valor calculado.

Figura 10 - Segunda parte do algoritmo recursivo.



```
        for (int w = N; w < (corrente_instantanea.Length); w++)
        {
            XC = X[w - 1] * Math.Cos(phi[w - 1]);
            XS = X[w - 1] * Math.Sin(phi[w - 1]);
            double xc = (Math.Sqrt(2.0) / N) * (corrente_instantanea[w] - corrente_instantanea[w - N]) * Math.Cos((w - N) * teta);
            double xs = (Math.Sqrt(2.0) / N) * (corrente_instantanea[w] - corrente_instantanea[w - N]) * Math.Sin((w - N) * teta);

            XC += xc;
            XS += xs;

            X[w] = Math.Sqrt((XC * XC) + (XS * XS));
            phi[w] = Math.Atan(XS / XC);

        }


    }
    tempo.Stop();
    Console.WriteLine("Tempo gasto  recursivo : " + (tempo.ElapsedMilliseconds / NP).ToString() + " milisegundos");
}
```

Fonte: Produzida pelo próprio autor

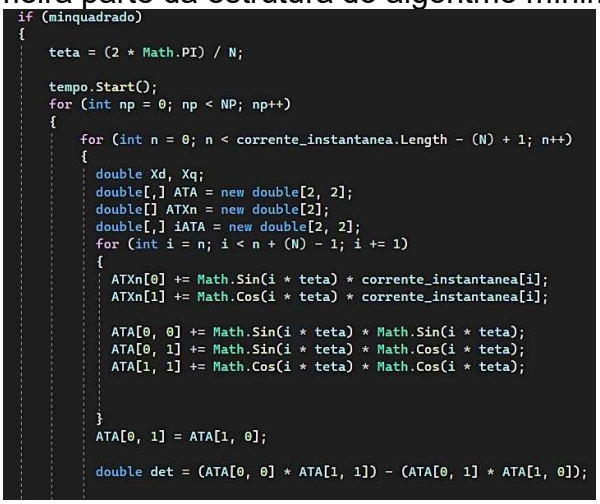
O laço *for{.}* mostrado na Figura 10, possibilitava a movimentação da janela por toda extensão da amostragem enquanto o vetor *XC[.]* e *XS[.]* são atualizados pelos valores da variáveis *xc* e *xs*, dessa maneira é possível excluir a última amostra da janela enquanto a janela se deslocava por todo o conjunto amostral. Assim, é possível calcular a magnitude e ângulo do fasor utilizando o método recursivo.

O fato de se excluir a última amostra da janela pode ocasionar uma certa instabilidade pois se qualquer erro acontecer durante o processamento, o erro é propagado para todos os outros cálculos, porém isso também o torna bem mais rápido se comparado ao não-recursivo.

3.4 O ALGORITMO DO MÍNIMO QUADRADO

Para o melhor entendimento em relação a sua estrutura, na Figura 11 é apresentada a primeira parte do código para o algoritmo mínimo quadrado.

Figura 3 - Primeira parte da estrutura do algoritmo mínimo quadrado.



```
if (minquadrado)
{
    teta = (2 * Math.PI) / N;

    tempo.Start();
    for (int np = 0; np < NP; np++)
    {
        for (int n = 0; n < corrente_instantanea.Length - (N) + 1; n++)
        {
          double Xd, Xq;
          double[,] ATA = new double[2, 2];
          double[] ATXn = new double[2];
          double[,] iATA = new double[2, 2];
          for (int i = n; i < n + (N) - 1; i += 1)
          {
            ATXn[0] += Math.Sin(i * teta) * corrente_instantanea[i];
            ATXn[1] += Math.Cos(i * teta) * corrente_instantanea[i];

            ATA[0, 0] += Math.Sin(i * teta) * Math.Sin(i * teta);
            ATA[0, 1] += Math.Sin(i * teta) * Math.Cos(i * teta);
            ATA[1, 1] += Math.Cos(i * teta) * Math.Cos(i * teta);


          }
          ATA[0, 1] = ATA[1, 0];

          double det = (ATA[0, 0] * ATA[1, 1]) - (ATA[0, 1] * ATA[1, 0]);
```

Fonte: Produzida pelo próprio autor

Da mesma maneira que a implemetação dos outros algoritmos, o primeiro laço *for{.}* é introduzido para realizar as repetições de simulações e encontrar a média do tempo de processamento.

Logo depois disso, o algoritmo começa ficar bem diferente dos outros dois, uma vez que precisa-se declarar algumas variáveis que não são declaradas nos algoritmos

fundamentados na DFT. As duas primeiras variáveis declaradas são *Xd* e *Xq*, conforme é apresentada em (05).

As outras três variáveis são matrizes, a matriz *ATA[,]* representa a matriz $\boldsymbol{x}_n$ em (06), a matriz *ATXn[.]* representa a matriz composta por seno e cosseno em (06),e a matriz *iATA[,]* representa a matriz inversa de $\boldsymbol{x}_n$ representada na equação (07).

Logo após declarar essas variáveis é o momento de se atribuir os valores de *ATA[,]* e *ATXn[.]* e, desse modo, construir a matriz que é utilizada nos cálculos feitos no decorrer do algoritmo. O segundo laço *for{.}* tem a finalidade de atribuir os valores mencionados.

A última linha de código observada na Figura 11, trata-se do cálculo do determinante da matriz *ATA[,]*, que é usado no cálculo da matriz inversa de $\boldsymbol{x}_n$ (*iATA[,]*). O código para esses cálculos podem ser observados na Figura 12.

Figura 12 - Segunda parte da estrutura do algoritmo mínimo quadrado.

```
            iATA[0, 0] = (ATA[1, 1] / det);
            iATA[0, 1] = (-ATA[0, 1] / det);
            iATA[1, 0] = (-ATA[1, 0] / det);
            iATA[1, 1] = (ATA[0, 0] / det);



            Xd = iATA[0, 0] * ATXn[0] + iATA[0, 1] * ATXn[1];
            Xq = iATA[1, 0] * ATXn[0] + iATA[1, 1] * ATXn[1];

            // Xd / Xq = Math.Tan(phi);
            phi[n+N-1] = Math.Atan(Xd / Xq);

            X[n+N-1] = Math.Sqrt((Xd * Xd) + (Xq * Xq)) / (Math.Sqrt(2));




        }


    }

    tempo.Stop();
    Console.WriteLine("Tempo gasto minquadrado : " + (tempo.ElapsedMilliseconds / NP).ToString() + " milisegundos");
}
```

Fonte: Produzida pelo próprio autor

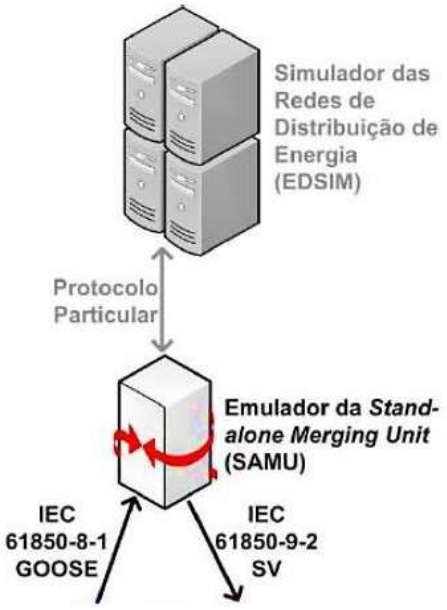
É calculada a matriz *iATA[,]* como pode ser observado no início da Figura 12, desse modo é possível encontrar os valores de *xd* e *xq* conforme (08) e encontrar os valores de magnitude e ângulo do fasor assumindo o método do mínimo quadrado.

4 RESULTADOS E DISCUSSÃO

4.1 PARTE EXPERIMENTAL

O algoritmo de estimativa de fasores requer operação e interação em tempo real com os dados, portanto, sua eficácia precisa ser validada em um ambiente de simulação de *hardware-in-the-loop* em tempo real (Leite, 2020). Na Figura 13, ressalta-se a estrutura do simulador em tempo real. O emulador da *standalone merging unit* (SAMU) obtém medições de corrente e tensão do simulador de rede de distribuição de energia (EDSIM) como em Leite e Mantovani, 2015, e transmite esses dados para o dispositivo em teste usando valores de amostra (SV) padronizado pela IEC 61850-9-2. Após o processamento dessas medições, o dispositivo em teste pode operar no sistema de distribuição simulado, emitindo comandos GOOSE, em vez de afetar seu funcionamento como ocorreria em uma rede elétrica real.

Figura 14 - Simulador em tempo real.

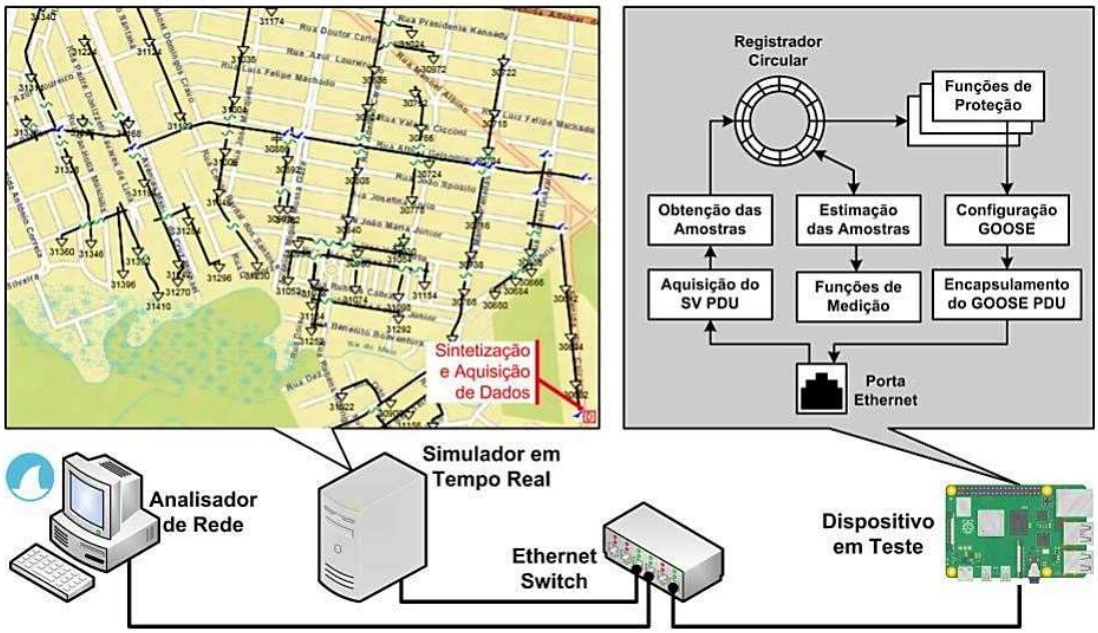


Fonte: (LEITE, 2020).

A estrutura experimental a ser utilizada é mostrada em detalhes na Figura 14. O simulador de tempo real e o dispositivo testado são conectados pelo nó de comunicação (Ethernet Switch). Um analisador de rede, como o Wireshark®

(Chappel, 2017), monitora o fluxo de dados. O diagrama funcional do dispositivo em teste ilustra seus principais blocos lógicos, incluindo o bloco de aquisição do SV PDU (IEC, 2004) e o encapsulamento do GOOSE PDU para comunicação Ethernet. As amostras encontradas dos valores de tensão e corrente são coletadas em um registrador circular, que fornece as informações de entrada para as funções de proteção e medição, com destaque para o algoritmo de estimação de fasores a ser avaliado neste trabalho de graduação.

Figura 15 - Diagrama Funcional.



Fonte: (LEITE, 2020).

Para o maior entendimento do funcionamento dos algoritmos algumas variáveis importantes, que foram obtidas através de um laboratório, são apresentadas. A frequência de amostragem escolhida foi de 500 amostras divididas em 10 ciclos, 50 amostras por ciclo, sendo que cada ponto era o valor instantâneo de um sinal (tensão ou corrente) medida no laboratório. Com essa frequência de amostragemfoi realizado uma série de simulações utilizando os três algoritmos: não recursivo, recursivo e mínimos quadrados.

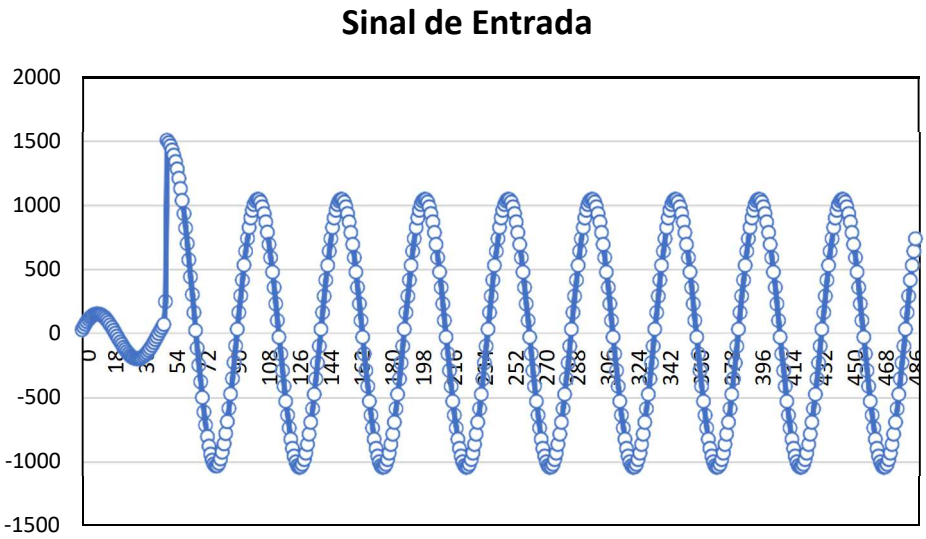
Após realizar as simulações com cada algoritmo listado, foi possível obter alguns dados importantes para conseguir analisar o desempenho dos algoritmos, esses dados demonstram características como: velocidade de processamento e assertividade dos cálculos, desse modo pode-se analisar qual dos algoritmos se

comporta melhor quando se deseja uma maior exatidão nos cálculos e maior velocidade no processamento de cada ação dentro do código.

Uma curva característica foi levantada para cada grandeza obtida, sendo elas a magnitude, o ângulo e o tempo de processamento. Além de se encontrar essas grandezas para cada algoritmo, foi realizado repetições de cada algoritmo para se encontrar os valores mais precisos, desse modo notou-se que quanto mais repetições cada algoritmo era submetido maior era a sua precisão.

Outro aspecto essencial para se analisar é o sinal de entrada do algoritmo, pois a partir dessa curva foi possível retirar todos os pontos que seriam usados para o processamento dos três algoritmos, o sinal foi medido, conforme a estrutura experimentas nas Figuras 13 e 14, e convertido em arquivo XML para ser lido por um algoritmo principal e assim ser possível retirar os dados necessários. Na Figura 15 é mostrado a curva desse sinal.

Figura 15 - Curva do sinal de entrada

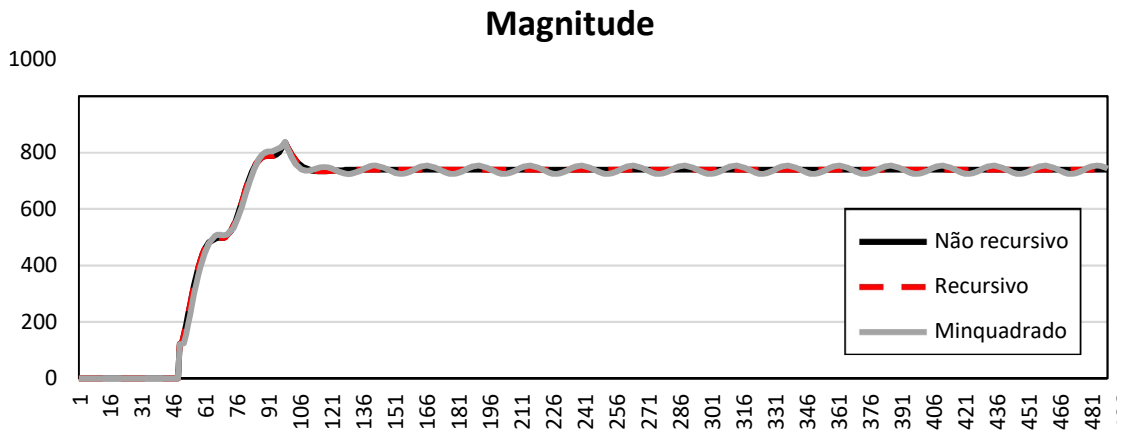


Fonte: Produzida pelo próprio autor

4.2 CURVAS CARACTERÍSTICA DA MAGNITUDE DOS FASORE ESTIMADOS.

Tendo sido realizadas as simulações referentes aos três algoritmos desenvolvidos, não-recursivo, recursivo e mínimos quadrados, foi plotado três curvas de magnitude, de maneira sobreposta para facilitar a analise, como ilustrado na Figura 16.

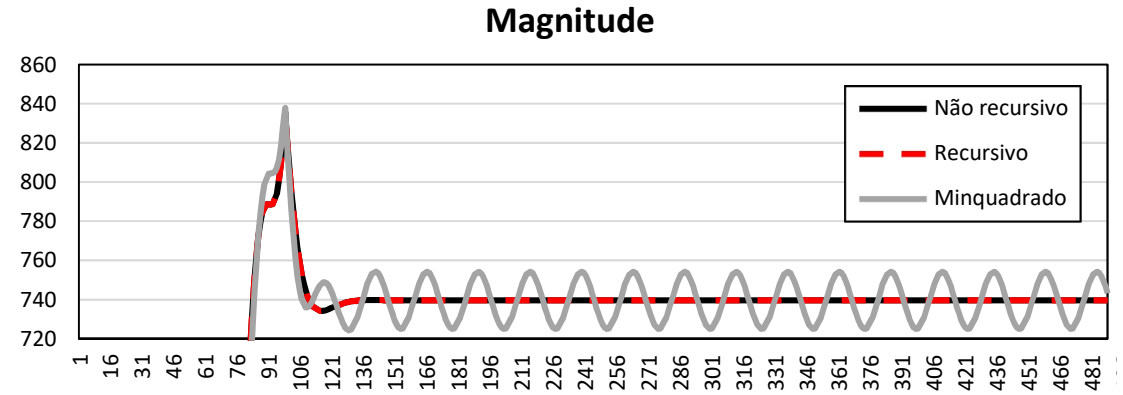
Figura 16 - Magnitude dos 3 métodos sobrepostos.



Fonte: Produzida pelo próprio autor

Para melhorar a visualização, foi feito um corte na área de estabilização dessa maneira pode se obter a Figura 17.

Figura 17 - Corte magnitude dos 3 métodos sobrepostos.



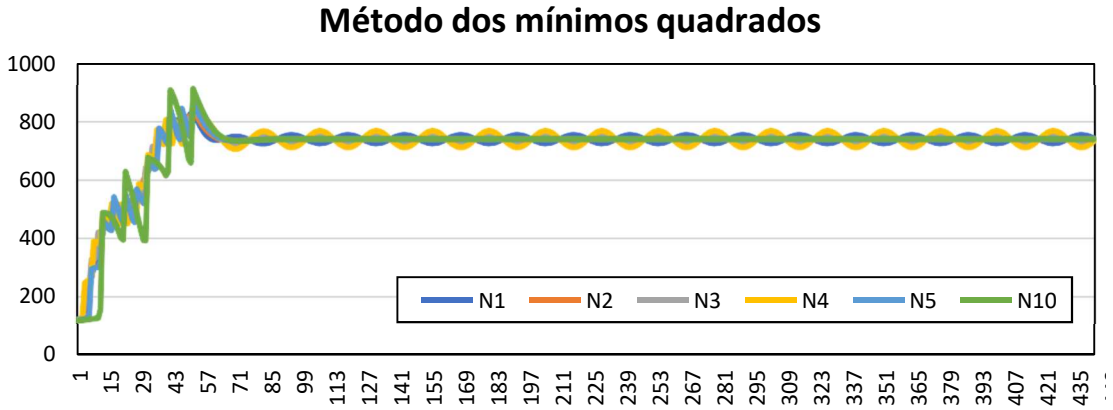
Fonte: Produzida pelo próprio autor

Observando a Figura 16 pode se perceber que os algoritmos recursivo e não-recursivo se comportaram de maneira muito parecida, mesmo que um deles seja muito mais estável que outro. Isso se deve pela quantidade de pontos do janelamento, de qualquer forma a magnitude encontrado pelos dois métodos foram iguais.

Já o método dos mínimos quadrados apresentou um comportamento bem diferente dos demais, sua curva além de ser diferente não se estabilizou quando chegou em um ponto de convergência e ficou oscilando em torno do valor da magnitude do fasor até o fim do processamento do algoritmo.

Quando o algoritmo dos mínimos quadrados foi submetido a uma mudança da contagem em um dos laços *for{.}*, foi possível obter resultados mais estáveis iguais aos apresentados pelos outros dois métodos de DFT. Foram realizadas mais cinco simulações mudando a frequência da contagem do algoritmo, e foram nomeados com forme a mudança era feita. Para distinguir a mudança na frequência da contagem, foi adotado a nomenclatura *N1*, para o caso que não houve nenhuma mudança; *N2*, para quando a contagem do laço *for{.}* era de 2 pontos em 2 pontos, ou seja metade da frequência de amostragem; *N3*, para quando a contagem do laço *for{.}* era de 3 pontos em 3 pontos; *N4*, para quando a contagem do laço *for{.}* era de 4 pontos em 4 pontos; *N5*, para quando a contagem do laço *for{.}* era de 5 pontos em 5 pontos; e *N10*, para quando a contagem do laço *for{.}* era de 10 pontos em 10 pontos. As magnitudes estimaas nessas execuções podem ser observadas na Figura 18.

Figura 18 - Magnitude de *N1* até *N10* par o método do mínimo quadrados.

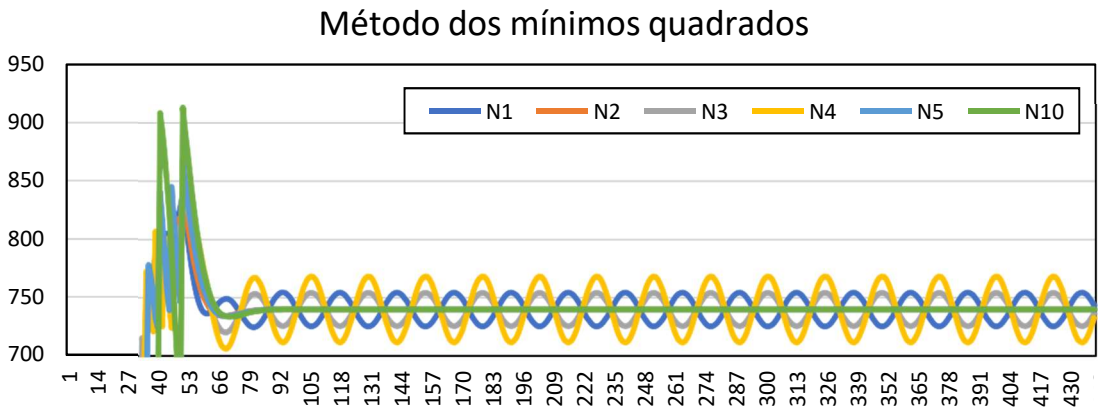


Fonte: Produzida pelo próprio autor

Da mesma maneira foi realizado um corte na área de estabilidade, criando a Figura 19.

Como visto anteriormente a execução *N1* possui uma grande variação e pouca estabilidade, e cada simulação que se aumentava o intervalo de contagem os resultados obtidos começavam a se aproximar dos métodos recursivo e não-recursivo, dessa maneira as curvas referente a *N2* foi mais estável que *N1*, *N3* foi mais estável que *N2*, *N4* mais estável que *N3*, *N5* mais estável que *N4* e por fim *N10* mais estável que *N5*.

Figura 19 - Corte magnitude de *N1* até *N10*.



Fonte: Produzida pelo próprio autor.

De imediato pensou-se que simplesmente aumentando o intervalo da contagem os resultados seriam estáveis, porém quando tentou-se executar curvas para *N6*, *N7*, *N8*, *N9* e todos as outras acima de *N10* notou-se um resultados pior que os anteriormente vistos, de maneira que estes foram tão aleatórios que não foi possível montar um gráfico de acordo com o modelo selecionado para mostrar os dados calculados. Deduziu-se que o sucesso de *N5* e *N10*, as simulações que mais chegaram perto da estabilidade desejada, se deu pelo fato do intervalo em questão serem múltiplos de 50.

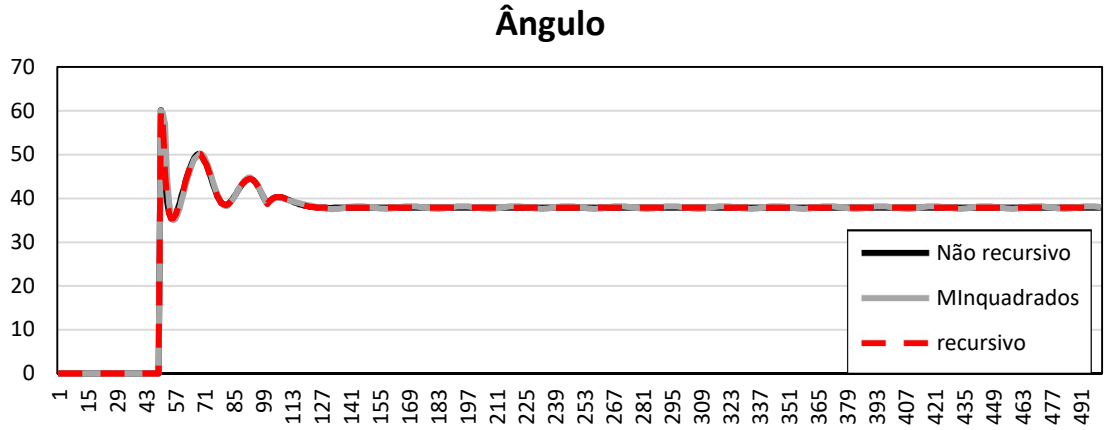
4.3 CURVAS CARACTERÍSTICA DO ANGULO DOS FASORES ESTIMADOS.

As curvas características referentes ao ângulo foram levantadas juntamente com as magnitudes e de maneira semelhante foram sobrepostas para serem analisadas, essas curvas são ilustradas na Figura 20.

Foi realizado um corte na área de instabilidade para melhorar a visualização, formando a Figura 21.

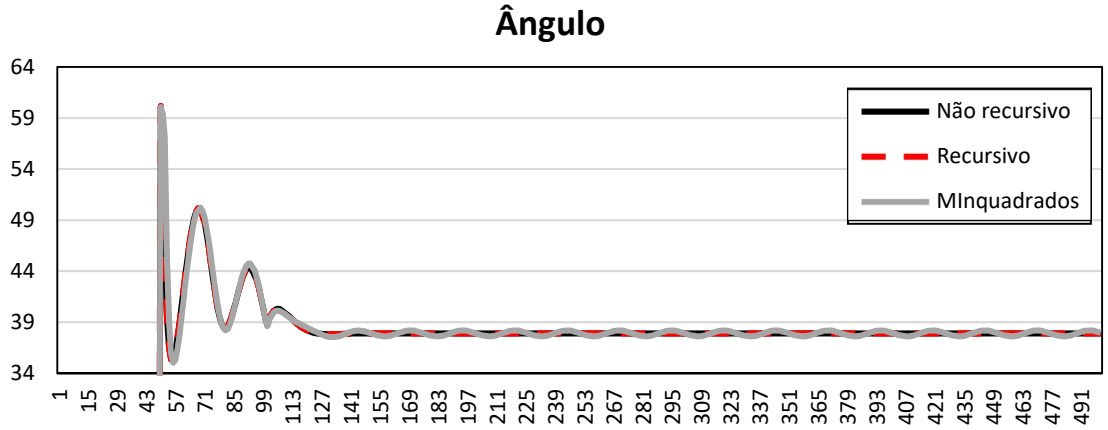
As execuções revelaram que, a curva característica referente ao ângulo, se comportaram da mesma forma que as curvas de magnitudes. Desse modo, os métodos recursivo e não-recursivo tiveram resultados iguais e estáveis e o método dos mínimos quadrados apresentou um resultado diferente e com certa variação, porém a variação foi ligeiramente menor que o encontrado na magnitude.

Figura 20 - Curva de ângulos dos 3 métodos.



Fonte: Produzida pelo próprio autor

Figura 21 - Corte curva de ângulos dos 3 métodos.



Fonte: Produzida pelo próprio autor

4.4 TEMPO DE PROCESSAMENTO

Após a conclusão dos cálculos das curvas de magnitude e do ângulo assim como a analise respectiva considerando cada método estudado, surgiu a necessidade de se encontrar outros meios para se comparar os algoritmos, então decidiu-se por cronometrar o tempo de processamento de cada estimativa.

Como o tempo para executar cada algoritmo estava na casa dos milissegundos não era possível perceber qualquer diferença no desempenho dos três métodos em questão e também não era possível calcular de maneira analógica utilizando qualquer aparelho de medição então um pequeno comando foi introduzido no inicio e fim de

cada algoritmo contabilizando assim o tempo demorado para que cada operação fosse realizada.

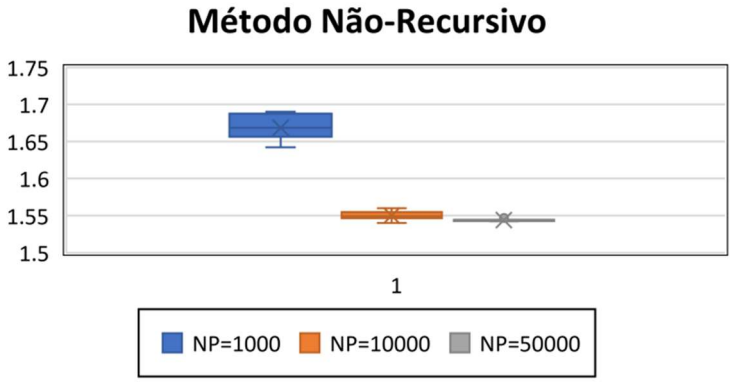
Logo pode-se perceber que a diferença relativa entre os métodos era, de certa forma, muito grande e isso ocorre principalmente pelo formato de cálculo de cada algoritmo. A diferença na estrutura de cálculo é explicada com o decorrer das análises dos tempos de processamento.

Outro aspecto importante que se notou ao realizar as execuções do mesmo algoritmo várias vezes foi que o tempo de processamento possuía uma variação conforme a contagem se repetia, essa variação era significativa e poderia atrapalhar na análise entre os algoritmos, então decidiu-se criar outro comando que repetisse o mesmo calculo uma quantidade de vezes e depois tirasse a média aritmética desses valores de tempo para que esse tempo fosse um pouco mais preciso em relação aos primeiro valores de tempo encontrado. Decidiu-se também observar a maior precisão conforme era aumentado o número de vezes que o cálculo era repetido, uma variável foi atribuída para o número de repetições essa variável era *NP*, para *NP* foi atribuído três valores diferentes, 1.000, 10.000 e 50.000.

4.4.1 Tempo de processamento método não-recursivo.

Realizando as simulações para o método não-recursivo e cronometrando através do algoritmo foi possível encontrar os valores para o tempo de processamento considerando as repetições *NP*, plotou-se o gráfico da Figura 22 que permite uma melhor visualização do tempo da operação não-recursiva.

Figura 22 - Tempo de processamento método não-recursivo.



Fonte: Produzida pelo próprio autor

De acordo com o gráfico estima-se que utilizando *NP*=1.000 o resultado teve um resultado bem mais instável sendo que a variação máxima ficou próxima a 1,7 milissegundos (ms), a variação máxima utilizando *NP*=10.000 ficou próximo a 1,57 ms e a variação máxima utilizando NP=50.000 ficou próximo a 1,54 ms. Concluiu-se que a medida que o número de repetições aumentou a variação máxima diminuiu e aumentou a exatidão e a precisão dos valores de tempos de processamento.

Os tempos de processamento evidenciados nesse gráfico nos mostra que o método não-recursivo é relativamente lento se comparados aos outros algoritmos, esse fato ocorre devido a estrutura de cálculo do algoritmo.

O algoritmo não-recursivo inclui cada ponto novo na janela e o mantém até a finalização de todos os pontos, isso confere uma maior estabilidade para o algoritmo na hora de realizar a operação, porém reduz a velocidade de processamento consideravelmente, porém a variação máxima considerando cada *NP* se manteve baixa por causa da estabilidade atribuída ao algoritmo pela sua estrutura de cálculo. Caso o sinal de entrada fosse maior seria ainda mais evidente a sua disparidade em relação ao tempo de execução total do algoritmo

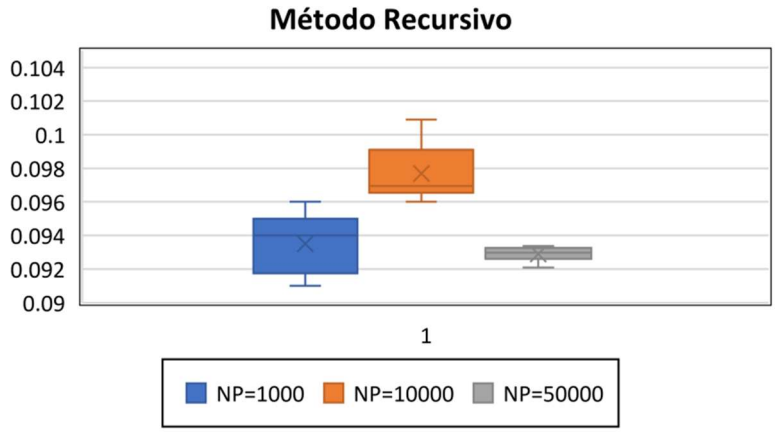
4.4.2 Tempo de processamento método recursivo.

Ao realizar as execuções para o método recursivo e cronometrando através do algoritmo encontrou-se o tempo de processamento considerando as repetições *NP*=1.000, *NP*=10.000 e *NP*=50.000, montou-se o gráfico da Figura 23 permitindo uma melhor visualização do tempo da operação recursiva.

Observando o gráfico estima-se que a variação máxima considerando *NP*=1.000 ficou próxima a 0,096 ms e também pode ser considerado a maior variação em relação as outras repetições do mesmo algoritmo, para *NP*=10.000 a variação máxima ficou próxima a 0,101 ms e para *NP*=50.000 a variação ficou próxima a 0,0935 ms. Dessa maneira ele se comportou de maneira semelhante ao método não-recursivo, e à medida que se aumentava as repetições a variação máxima diminuiu consideravelmente.

Também se percebeu um tempo de processamento bem menor em relação ao algoritmo anterior, o método recursivo foi bem mais rápido devido a sua estrutura de cálculo, porém a instabilidade que era esperada na variação não foi tão relevante.

Figura 23 - Tempo de processamento método recursivo.



Fonte: Produzida pelo próprio autor

O método recursivo tem por característica excluir o último ponto da amostragem quando a janela se desloca entre os 50 pontos do ciclo e adicionar o próximo ponto, dessa maneira ele consegue ser bem mais rápido no seu processamento, porém corre o risco de perpetuar caso esse venha a ocorrer durante o cálculo, por isso se esperava uma menor eficiência na sua exatidão e precisão quando se calcula a magnitude e ângulo. Conclui-se que o método recursivo não apresentou nenhuma incoerência de cálculos pelo fato do sinal de entrada possuir poucos pontos de amostragem.

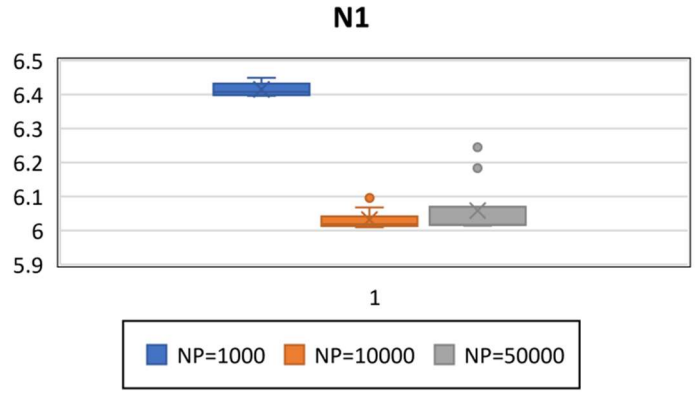
4.4.3 Tempo de processamento método mínimos quadrados.

Para o método dos mínimos quadrados foi proposto um comparativo a mais, uma vez que quando se executou o algoritmo considerando *N1* o resultado tanto para magnitude e ângulo foi discrepante em relação aos outros métodos, então para cronometrar o tempo de processamento foi considerado *N1*, *N5* e *N10*. A Figura 24 a seguir representa *N1*, para as repetições *NP*=1.000, *NP*=10.000 e *NP*=50.000.

Conforme mostra a figura os valores do tempo de processamento não seguiram os mesmos princípios dos outros métodos, uma vez que, aumentando as repetições não interferiu na variação máxima e na precisão dos dados encontrado, aconteceu de maneira contrária que a prevista, com NP=50000 ocorreram pontos fora da variação.

Outro ponto a se destacar foi o tempo encontrado considerando *N1* foi alto, tornando o método mais lento e menos eficiente tanto na precisão quanto na velocidade.

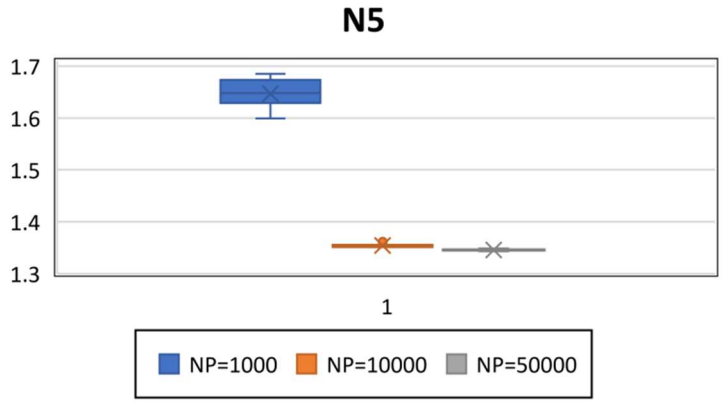
Figura 24 - Tempo de processamento N1.



Fonte: Produzida pelo próprio autor

O próximo gráfico plotado foi para *N5*, considerando *NP*=1.000, *NP*=10.000 e *NP*=50.000, ilustrado na Figura 25.

Figura 6 - Tempo de processamento *N5*.

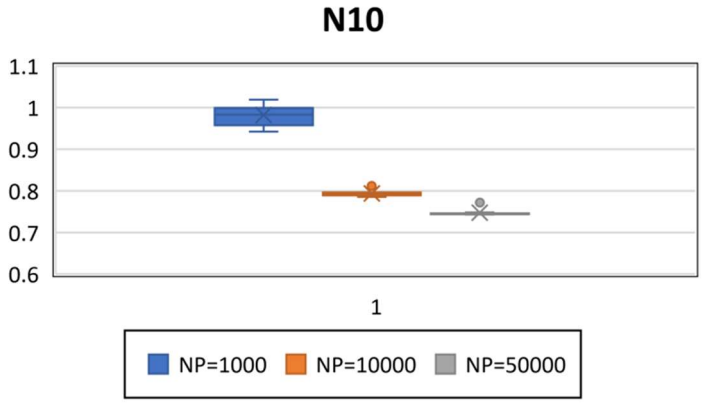


Fonte: Produzida pelo próprio autor

Analisando o gráfico percebe-se a grande mudança em relação ao gráfico anterior, a variação máxima foi menor que o a simulação anterior e o tempo de processamento também, a variação máxima para *NP*=50.000 foi tão pequena que permitiu atingir um grau de precisão interessante a ponto de ficar próxima aos dois primeiros métodos.

Para a próxima execução considerou *N10*, para *NP*=1.000, *NP*=10.000 e *NP*=50.000, mostrado na Figura 26.

Figura 26 - Tempo de processamento *N10*.



Fonte: Produzida pelo próprio autor

Essa execução foi a que apresentou o melhor resultado em relação as outras execuções do método mínimos quadrados. Conseguiu uma maior estabilidade pelo fato de sua variação máxima ser a menor e conseguiu uma maior velocidade pelo fato de ter o menor tempo de processamento.

Para ilustrar melhor o comparativo entre as simulações *N1*, *N5* e *N10* do método dos mínimos quadrados, montou-se uma tabela para cada simulação considerando *NP*=50.000 para se obter o maior grau de estabilidade. A Tabela 1 ilustra as diferenças de variações e de valores absolutos obtidos nas três execuções.

Tabela 1 - Tempo de processamento (ms) min, med e máx para NP=50.000.

N1		***N5***		***N10***	
***NP*=50.000**		***NP*=50.000**		***NP*=50.000**	
1	6,24456	1	1,34838	1	0,77136
2	6,0297	2	1,3451	2	0,7479
3	6,0313	3	1,34618	3	0,74384
4	6,1832	4	1,34484	4	0,744
5	6,01662	5	1,3489	5	0,74488
6	6,01398	6	1,34538	6	0,74326
7	6,01632	7	1,34444	7	0,74548
8	6,01672	8	1,34348	8	0,74536
9	6,01716	9	1,34278	9	0,74506
10	6,01468	10	1,34468	10	0,74312

Fonte: Produzida pelo próprio autor

5 CONCLUSÃO

Ao término de todas as execuções e analisando comparativamente os três algoritmos estudados sendo eles, o método não recursivo, recursivo e mínimo quadrado, foi possível tirar conclusões importantes a respeito da sua efetividade considerando a precisão e a velocidade de processamento. Desse modo o método recursivo se mostrou como o mais promissor quando se olha para a velocidade de processamento, já que este obteve o menor tempo de operação, e quando se olha para a precisão ele surpreendeu positivamente, uma vez que se esperava uma instabilidade muito alta, tendo em vista sua estrutura de cálculo, seus resultados foram iguais aos do método não-recursivo provavelmente pelo número de amostra se baixo. O método não-recursivo se mostrou bem lento se comparado ao recursivo, porém isso já era esperado pela sua estrutura de cálculo, e a precisão se igualou ao método recursivo. O método do mínimo quadrado se mostrou bem interessante uma vez que pode-se observar alguns comportamentos incomuns, quando se calculava a magnitude e ângulo os resultados se mantiveram instáveis e teve uma variação considerável observados através de um gráfico e o tempo de processamento foi bem alto e totalmente aleatório se comparados com os outros métodos, porém adotou-se uma forma de se corrigir essa instabilidade basicamente pulando as amostras que estavam sendo lidas e dessa forma melhorou-se bastante os resultados, fazendo que com a precisão fosse aceitável e o tempo fosse diminuído ficando mais rápido que o método não-recursivo. Com isso pode-se confirmar que o método do mínimo quadrado precisa de menos amostras para conseguir fazer uma estimativa mais precisa que as outros métodos.

6 REFERÊNCIAS

CHAPPELL, L., ***Wireshark 101: Essential Skills for Network Analysis***, 2nd Ed., pages 408, 2017.

EKANAYAKE, J., LIYANAGE, K., WU, J., YOKOYAMA, A., JENKINS, N., "***Smart Grid: Technology and Applications***"- Wiley, 2012.

IEC 61850-1. ***Communication networks and systems in substation, Part 1: Introduction and Overview***. Geneva, Switzerland, 2003.

IEC, ***Communication Networks and Systems in Substations – Part 9-2: Specific Communication Service Mapping (SCSM) – Sampled Value over ISO/IEC8802-3***. First Edition, pages 1-34, 2004.

LEITE, J. B.. ***Simulação de Hardware-In-the-Loop para Testar Dispositivos de Automação Avançada em Redes de Distribuição de Energia Elétrica***. Anais do VIII Simpósio Brasileiro de Sistemas Elétricos (SBSE 2020), Santo André - SP. p. 1-6. 2020.

NORTHCOTE-GREEN, J., WILSON, R. G., "***Control and Automation of Electrical Power Distribution Systems***." Reino Unido: CRC Press, 2017.

PHADKE, A., "***Synchronized Phasor Measurements and Their Applications.***" Springer International PU, 2017.

SACHDEV, M.S., AND M. NAGPAL. "***A Recursive Least Error Squares Algorithm for Power System Relaying and Measurement Applications***." IEEE Transactions on Power Delivery 6, no. 3, 1008-015, 1991.

WANG, C., "***Smart Electricity Distribution Networks.***", Reino Unido: CRC Press, 2017.

J DUNCAN GLOVER; OVERBYE, T. J.; SARMA, M. S. Power system analysis & design. Boston, Ma: Cengage Learning, 2017.

REIZ, C. e LEITE, J. B., "***Hardware-In-the-Loop Simulation to Test Advanced Automation Devices in Power Distribution Networks***," in IEEE Transactions on Power Delivery, vol. 36, no. 4, pp. 2194-2203, Aug. 2021, doi: 10.1109/TPWRD.2020.3022333.