UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE TECNOLOGIA
CURSO DE ENGENHARIA ELÉTRICA

KAREN AKEMI KOBAYASHI

ESTUDO DE FONTES DE ERRO EM GEOLOCALIZAÇÃO ATRAVÉS DE SISTEMAS BASEADOS EM SATÉLITES E PROPOSTAS DE FORMAS DE MINIMIZAÇÃO DESTE ERRO

CURITIBA
2018

KAREN AKEMI KOBAYASHI

ESTUDO DE FONTES DE ERRO EM GEOLOCALIZAÇÃO ATRAVÉS DE SISTEMAS BASEADOS EM SATÉLITES E PROPOSTAS DE FORMAS DE MINIMIZAÇÃO DESTE ERRO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica da Universidade Federal do Paraná, como requisito parcial à obtenção do título de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Ewaldo Luiz de Mattos Mehl

CURITIBA

2019

KAREN AKEMI KOBAYASHI

ESTUDO DE FONTES DE ERRO EM GEOLOCALIZAÇÃO ATRAVÉS DE SISTEMAS BASEADOS EM SATÉLITES E PROPOSTAS DE FORMAS DE MINIMIZAÇÃO DESTE ERRO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica da Universidade Federal do Paraná, como requisito parcial à obtenção do título de bacharel em Engenharia Elétrica.

COMISSÃO EXAMINADORA

_____

Prof. Dr. Ewaldo Luiz de Mattos Mehl (Orientador)
Departamento de Engenharia Elétrica
Universidade Federal do Paraná (UFPR)

_____

Mestre Lucas Pioli Rehbein Kürten Ihlenfeld
Departamento de Engenharia Elétrica
Universidade Federal do Paraná (UFPR)

_____

Prof. Tibiriçá Krüger Moreira
Departamento de Engenharia Elétrica
Universidade Federal do Paraná (UFPR)

Curitiba, 04 de dezembro de 2018.

AGRADECIMENTOS

A minha religião, por me ajudar a enfrentar todos os dias e me guiar.

Ao meu orientador Ewaldo, pelo empenho e esforço dedicado a correções e incentivos que foram essenciais à elaboração deste trabalho de conclusão de curso.

Aos professores do Curso de Engenharia Elétrica da Universidade Federal do Paraná, que se comprometeram a me passar o conhecimento e educação e que assim, oportunizaram diversas portas para o meu caminho profissional.

À minha família, que me me apoiou e ajudou, nas horas que eu mais precisei de espaço e também quando eu precisei de um incentivo. Principalmente a Yumi por emprestar o celular para uso neste trabalho.

Ao Sá e Bolo por me apoiarem em todos os momentos mas mais precisamente para o desenvolvimento deste trabalho.

E aos demais amigos que tive a oportunidade de conhecer e partilhar momentos da vida, que tornaram os momentos difíceis em algo menor e menos assustador.

RESUMO

O GNSS (*Global Navigation Satellite Systems* - Sistemas Globais de Geolocalização por Satélites) referencia todos os satélites de geolocalização que possuem cobertura mundial. O mais conhecido deles é o GPS desenvolvido pelo Departamento de Defesa dos Estados Unidos - DoD (*Department of Defense*) e com o passar dos anos foi aberto para o uso civil. De maneira que seu uso se tornor muito popular e seu uso abrangeu para sistema para mapeamento e cartografia de precisão, navegação marítmica, agricultura, localização de automóveis e entre outras utilizações. Com isso, a necessidade de localizações precisas aumentou e forma a serem realizados vários estudos para mitigar os erros sofridos pelos receptores GNSS. Esses principais erros são são refração da troposfera e da ionosfera, absorção atmosférica, ruído do receptor, falta de precisão no relógio do satélite e eletrônicos, efeitos de multicaminhos e erros efemérides. Alguns países pelo mundo possuem um satelite geoestacionário que realiza a correção do erro atmosférico em tempo real, como: o sitado WAAS nos Estados Unidos, Egnos no continente Europeu e o MSAS no Japão, porém o Brasil não possui este tipo de satélite. Neste trabalho, foram apresentados os erros atmosféricos, compostos pelo atraso troposférico e ionosférico. Com o objetivo de realizar a mitigação destes erros para o território brasileiro sem a necessidade de estar conectado a internet. Para isso, foi utilizado o modelo de Hopfield e de Saastamoinen para a realização da mitigação do erro troposférico e mostrado o modelo de Klobuchar para o erro ionosférico. Com estes métodos, realizado os cálculos de maneira computacional por meio do software NetBeans IDE e pode obter uma distância de 12 metros entre o receptor e o ponto gerado computacionalmente.

Palavras-chave: GNSS. GPS. Erro troposférico hidrostático.

ABSTRACT

Global Navigation Satellite Systems (GNSS) refers to all geolocation satellites that have global coverage. The best known of these was GPS developed by the US Department of Defense (DoD) and over the years it has been opened for civilian use. GPS use became very popular and its use for system for mapping and cartography of precision, marítmica navigation, agriculture, localization of automobiles and among other uses. As a result, the need for precise locations has increased and several studies are needed to mitigate the errors of GNSS receivers. These main errors are refraction of troposphere and ionosphere, atmospheric absorption, receiver noise, lack of accuracy in the satellite and electronic clock, multipath effects and ephemeris errors. Some countries around the world have a geostationary satelite that performs real-time correction of atmospheric error, such as WAAS in the United States, Egnos in the European continent and MSAS in Japan, but Brazil does not have such satellite. In this essay, atmospheric errors composed of tropospheric and ionospheric delay were presented. With the objective of mitigating these errors for the Brazilian territory without the need to be connected to the Internet. For this, the Hopfield and Saastamoinen models were used to perform the tropospheric error mitigation and the Klobuchar model for the ionospheric error was shown. With these methods, calculation performed computationally using the NetBeans IDE software and they can achieve a distance of 12 meters between the receiver and the computationally generated point.

Key-words: GNSS. GPS. Hydrostatic tropospheric error.

LISTA DE ILUSTRAÇÕES

No table of figures entries found.

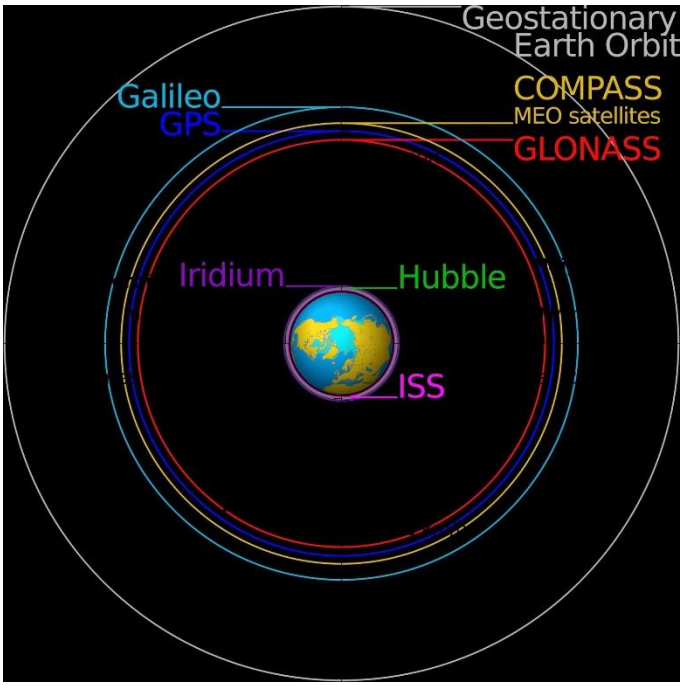
SUMÁRIO

1 INTRODUÇÃO

Sistemas Globais de Geolocalização por Satélites (GNSS – *Global Navigation Satellite Systems*) é o termo utilizado para referenciar todos os satélites de geolocalização que possuem cobertura global. Os sistemas mais conhecidos de GNSS são o GPS dos Estados Unidos, Galileo da Europa, GLONASS da Russia e COMPASS da China. O modo de operação de todos os sistemas que pertencem ao GNSS é baseado na distância relativa entre a posição do usuário e dos satélites. Para isso, utiliza-se o tempo de deslocamento das ondas de radio frequências transmitidas pelos satélites (KOS, BOTINČAN e DLESK, 2009).

A Figura 1 mostra orbitas de 4 sistemas mais conhecidos atualmente, US-GPS, EU-Galileo, Russia-GLONASS, China-Beidou/COMPASS comparadas a orbitas dos satellites Iridium, telefonia celular por satelite, do telescópio Hubble e da Estação Espacial Internacional (GROUNDS, 2014).

Figura 1 - Comparação de orbitas espaciais



Fonte: Grounds, 2014.

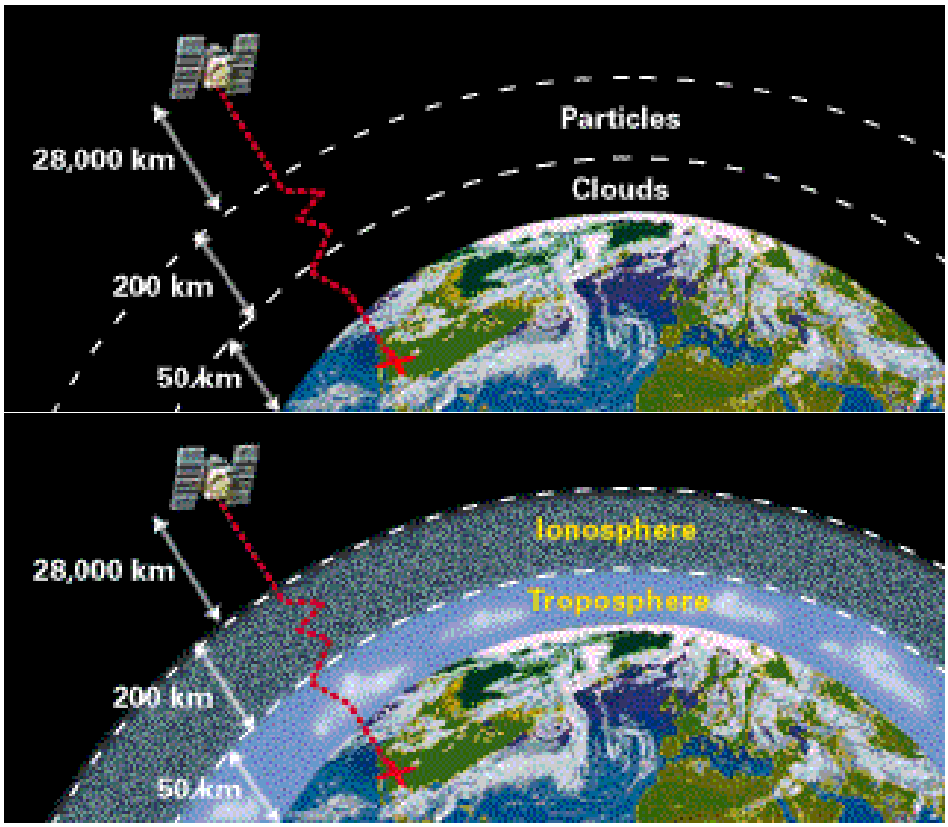
O GPS (*Global Positioning System* – Sistema de Posicionamento Global) foi desenvolvido pelo Departamento de Defesa dos Estados Unidos - DoD (*Department of Defense*) primeiramente para uso militar, para ser o método de nagevação principal das forças armadas americanas. (MONICO, 2000). Com o passar dos anos, o sistema GPS foi aberto para uso civil e desenvolveu-se receptores com melhor acurácia. De

modo que abrangeu-se a utilização desse sistema para mapeamento e cartografia de precisão, navegação marítmica, agricultura, localização de automóveis e entre outras utilizaçoes.

Desde 1995, o sistema de GPS consiste de 24 satélites posicionados de maneira que pelo menos quatro satélites estejam visíveis acima do horizonte, 24 horas por dia e em qualquer lugar do planeta Terra. Assim, o sistema consegue prover de informação de posição, navegação e tempo para se ter uma precisão tridimensional. Para isso, os satélites foram postos em órbitas a cerca de 20.200 km de altitude acima da superfície terrestre (SEEBER, 2003).

Os principais componente que contribuem para os erros ocorridos nas leituras de dados do GPS são refração da troposfera e da ionosfera, absorção atmosférica, ruído do receptor, falta de precisão no relógio do satélite e eletrônicos, efeitos de multicaminhos e erros efemérides. Além disso, existem erros que são inexplicáveis e imprevisíveis que podem ocorrer com o GPS, de modo que não são possíveis de serem calculados (US. ARMY CORPS OF ENGINEERS, 1996). A Figura 2 demostra os erros que os receptores GNSS enfrentam(www.trimble.com, acessado em 2018).

Figura 2 - Erros Atmosféricos



Fonte: www.trimble.com, acessado em 2018.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Obter uma solução viável para o Brasil de minimização do erro de refração atmosférico dos sinais GPS em tempo real, sem a necessidade de conexão com a internet. De modo a demonstrar a variação nas medições de localização de receptor de sinais GNSS fixado em um único ponto, através de um intervalo de semanas. Em seguida, verificar as ferramentas disponíveis no Brasil para a minimização do erro devido o atraso atmosférico e aplica-las por meio de ferramenta de software que será desenvolvida neste trabalho.

1.1.2 Objetivos Específicos

Os objetivos específicos do trabalho são:

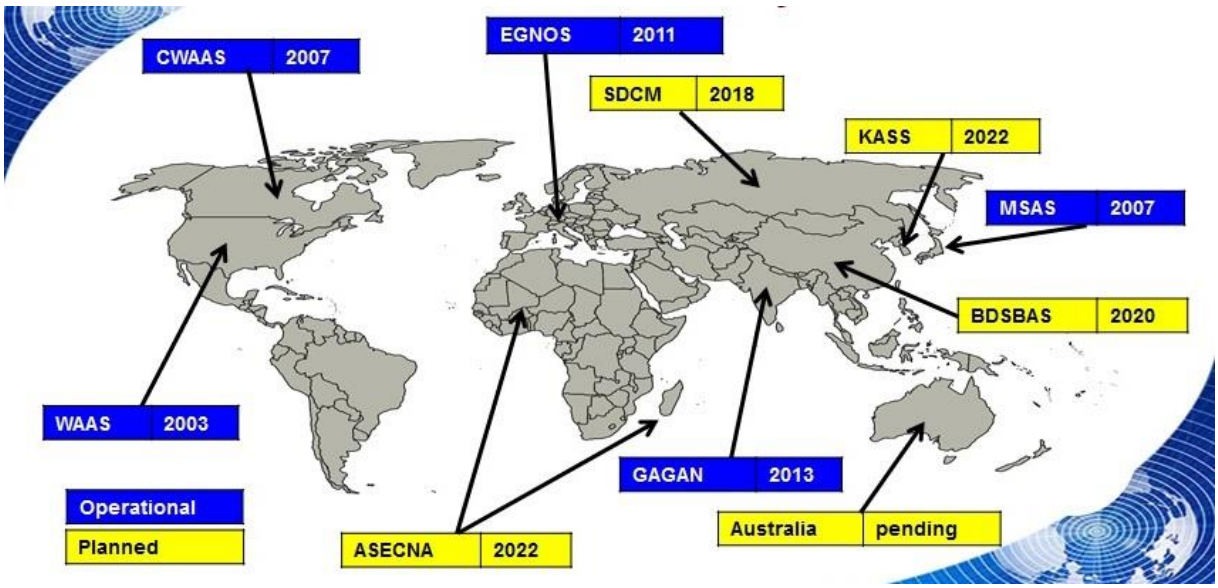
a) Obtenção de dados de geolocalização a partir do receptor de sinais GNSS;
b) Levantamento da metodologia de análise de dados para erros ocorridos devido a refração atmosférica;
c) Desenvolvimento de software de minimização do erro a partir dos dados obtidos.

1.2 JUSTIFICATIVA

A Administração de Aviação Federal Americana (FAA – *Federal Aviation Administration*) desenvolveu o sistema chamado WAAS (*Wide Area Augmentation System*) com o intuito de utilizar o sistema GPS para a realização rotas em navegação e de pousos e abordagens de alta precisão. O WAAS possui cobertura da superfície terrestre até 30.500 metros acima do nível do mar sobre os Estados Unidos e incluindo, Alaska, Havaí, Porto Rico e boa parte do Golfo do México. Para isso, consiste em satélites geoestacionários GE0, os quais transmitem os dados para estações de referência na superfície terrestre (WRSs – *Wide Area Reference Stations*). seguidamente, as WRSs conduzem os dados para as estações controladoras (WMSs – *Wide Area Master Stations*), que determinam as correções diferenciais e devolvem para os satélites GE0 onde os sinais DGPS serão retransmitidos para o usuário por meio da frequência L1 (FRENCH, 1996).

Alguns países pelo mundo possuem um satelite geoestacionário que realiza a correção do erro atmosférico em tempo real, como: o sitado WAAS nos Estados Unidos, Egnos no continente Europeu e o MSAS no Japão, porém, como pode ser visto na immagem o Brasil não possui este tipo de satélite. Além disso, pode ser visto que existem outros países realizando investimentos para realizar esta correção via satélite, demonstrado na Figura 3.

Figura 3 - Satélites geoestacionários



Fonte: *European Space Agency, 2018*

1.3 ESTRUTURA DO TRABALHO

Este trablaho foi dividido em 5 capítulos.

O primeiro capítulo realiza uma introdução ao tema abordado, objetivos a serem atingidos e a justificativa.

No capítulo seguinte, é apresentado a revisão bibliográfica sobre GNSS, GPS e erros atmosféricos, com o intuito de realizar a explicação teórica do trabalho.

No capítulo três são apresentados os matérias utilizados para realizar a mitigação dos erros atmosféricos e os métodos utilizados para isso.

No quarto capítulo é feito a análise dos materiais, métodos e dos dados adquiridos computacionalmente.

Por fim, no último capítulo, é apresentada a conclusão a partir dos resultados obtidos e sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

Os satélites contidos na constelação do GPS utilizam duas frequências, L1 e L2, para a transmissão de informação. As portadoras podem ser moduladas através do esquema BPSK (*binary phase shift keying*) por códigos pseudorandômicos, também conhecidos como códigos *Gold*, os quais possuem um comportamento spectral semelhante ao ruído, porém possuem boa autocorreção e propriedades de intercorrelação. Existem dois tipos de códigos que os satélites do GPS utilizam, o C/A, o qual está disponível par uso civil e modula a portadora L1, e o código P, que está disponível apenas para uso militar e modula as portadoras L1 e L2. Os sinais podem ser separados e detectados pela técnica CDMA (*code division multiple access*), pois códigos diferentes e não correlacionados são usados por cada satélite. Novos sinais estão sendo estudados para uma versão atualizada do GPS, com a inclusão de um sinal L1C modernizado, compatível com QZSS, uma segunda frequência civil L2C que irá proporcionar uma melhor correção dos erros ionosféricos, e sinais L5 e M, de modo que o último será de uso exclusivamente militar (SILVA, 2007).

Segundo (DIXON, 1991), para se determinar a localização de um observador no planeta Terra pode ser feita através da determinação da distância entre ele e apenas três satélites, os quais possuem as orbitas conhecidas. Em relação ao GPS, a medida de distância se baseia no tempo de viagem τ do sinal do satélite, adquirido pela medição da diferença entre os tempos de transmissão (t_s) e recebimento (t_r) no receptor de GPS de código de distância especial. Se for desconsiderado qualquer erro na precisão dos relógios e efeitos da transmissão de dados na velocidade da luz c, pode-se calcular a como a distância real ρ entre o satélite e o receptor como $c(t_r - t_s)$ apenas. Na distância estimada, estão presentes erros dos relógios do satélite ou do receptor, dessa maneira é referenciado como pseudodistancia R, definida na equação 1:

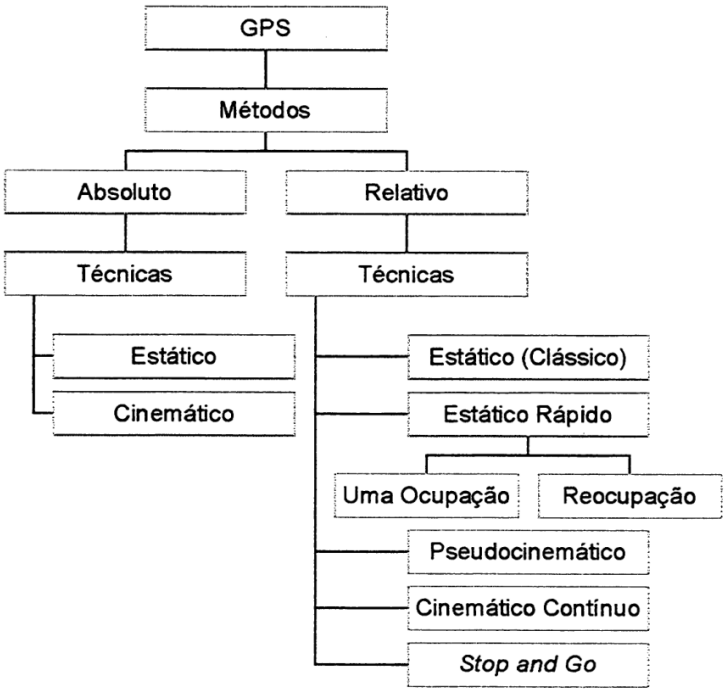
$$R = \rho + c\left(\Delta t_r - \Delta t_s + \Delta t_p\right) \quad (1)$$

onde Δt_r é o offset do “verdadeiro” horário pertencente os offset do relógio do receptor referente ao sistema GPS, qualquer outro erro induzido pelo receptor é desprezado, o offset do relógio do satélite é representado por Δt_s, e Δt_p é o atraso associado a

todos as outras fontes de erros, principalmente pelo efeito da propagação atmosférica. A utilização de um quarto satélite permite a correção de primeira ordem do relógio $(\Delta t_r - \Delta t_s)$, e métodos existentes podem ser aplicados para corrigir e estimar Δt_p, de modo a possibilitar posicionamento em nível de metros, sobre condições ideais.

Com GNSS se tornou possível a realização de maneira simples do geoposicionamento para usuários que possuíssem de um receptor. No início, era empregado o método absoluto, através de códigos. Com o passar dos anos, desenvolveram-se outras metodologias para se efetuar levantamentos. As atividades geodésicas, as quais necessitam de alta precisão, começaram a ser atendidas por estas novas metodologias. Assim, para a modelagem de distintas fontes de erros existentes nas observações, começou a ser utilizado os métodos relativos, os quais eram empregados anteriormente no VLBI (*Very Long Baseline Interferometry*) (LAGO, 2001). A Figura 4 expõe métodos e técnicas de posicionamento GPS através de um organograma sintetizado (CARVALHO, 1999).

Figura 4 - Organograma simplificado dos métodos e técnicas de posicionamento com o GPS



Fonte: Carvalho, 1999.

As técnicas de segurança, denominadas SA (*Selective Availability* – disponibilidade seletiva) e AS (*Anti-Spoofing*), ocasionam erros na definição de coordenadas das observações de GPS. O desligamento da disponibilidade seletiva foi

realizado em 2 de maio de 2000 e era aplicada para controle dos dados das efemérides. O intuito da AS era danificar a exatidão no posicionamento absoluto, pelo meio de erro artificial e intencional. No posicionamento relativo, consegue-se extinguir maior parte deste erro. A técnica AS se limita na modificação do código P por um código preciso Y, o qual acesso é concedido para usuários autorizados. Com o auxílio de técnicas especias, pode-se atualmente recuperar a portadora L2 por alguns receptores geodésicos. Assim, evidencia-se os Serviços de Posicionamento Padrão (SPS) e Serviço de Posicionamento Preciso (PPS). O SPS possui acessibilidade sem restrição a todos usuários, porém possui erros causados pelo *Anti-Spoofing* e pode-se atingir precisões cerca de 30 metros tanto no posicionamento vertical quanto no horizontal. O segundo serviço, PPS, disponibiliza o modelo de precisão e posicionamento aos usuários autorizados pelo DoD, de maneira que não ocorre sofre efeitos do *Anti-Spoofing* (LAGO, 2001).

A exatidão de posicionamento dos sistemas de GNSS podem variar de poucos milímetros para décimo de metros, de maneira a variar com o tipo de observáveis, como medições de código ou fase, e o modo de posicionamento, como receptores sozinhos ou em método relativo. Para a maioria dos receptores de GPS comerciais se consegue apenas exatidão de décimo de metros e com medições de código. De maneira que é necessário medições da fase da portadora de rádio frequência e modo de operação diferencial para se obter precisão de centímetros, o que é necessário para levantamentos RTK (*Real Time Kinematic* - Posicionamento Cinemático em Tempo Real). Além disso, é possível se obter precisão de milímetros através de pós-processamento de dados para aplicações de levantamentos estáticos precisos (KOS, BOTINČAN e DLESK, 2009).

2.1 POSICIONAMENTO ABSOLUTO

O posicionamento absoluto utiliza um único receptor passivo para determinar a localização de uma estação, para isso, ele se localiza na estação e realiza a coleta de dados de múltiplos satélites. Este método não possui precisão satisfatória para levantamentos de precisão ou utilização hidrográfica. Contudo, para navegação e localização, é o método mais utilizado para uso de posicionamento GPS militar e comercial (US. ARMY CORPS OF ENGINEERS, 1996). Com o aperfeiçoamento dos

métodos de correção dos erros de propagação do sinal e erros efeméricos, surgiu o Posicionamento por Ponto Preciso (PPP) (ALVES, MONICO e ROMÃO, 2011). O posicionamento por ponto pode ser dividido em duas componentes, baseado na precisão pretendida, posicionamento por ponto convencional, possui qualidade de coordenadas menores que 10 metros, e o PPP, o qual possui qualidade de coordenadas menores que 50 centímetros (POLEZEL, 2010).

O princípio fundamental do posicionamento absoluto é fundado na medida de pseudodistancias, por meio de aplicação das portadoras ou dos códigos, e possui aplicação tanto em levantamentos estáticos como cinemáticos (LAGO, 2001).

2.2 POSICIONAMENTO RELATIVO

Para o método de posicionamento relativo, necessita-se de pelo menos dois receptores para realizar a coleta de dados, de maneira que um deles se localize em uma estação com coordenadas conhecidas, chamada de estação de referência. Este método se baseia na diferença de informações entre a estação de referência e as estações de interesse, as quais possuem as coordenadas a serem determinadas. Para isso, todas as estações envolvidas no levantamento devem realizar observações ao mesmo tempo. Este método permite por meio de combinações lineares a minimização ou eliminação de várias fontes de erros (LAGO, 2001). Com o surgimento dos Sistemas de controle Ativos (SCA), usuários que possuem apenas um receptor podem realizar o posicionamento relativo. Para isso, devem realizar o acesso aos dados de estação que pertence ao SCA, as mais conhecidas são a Rede Brasileira de Monitoramento Contínuo (RBMC) e a Rede GNSS Ativa do estado de São Paulo (POLEZEL, 2010).

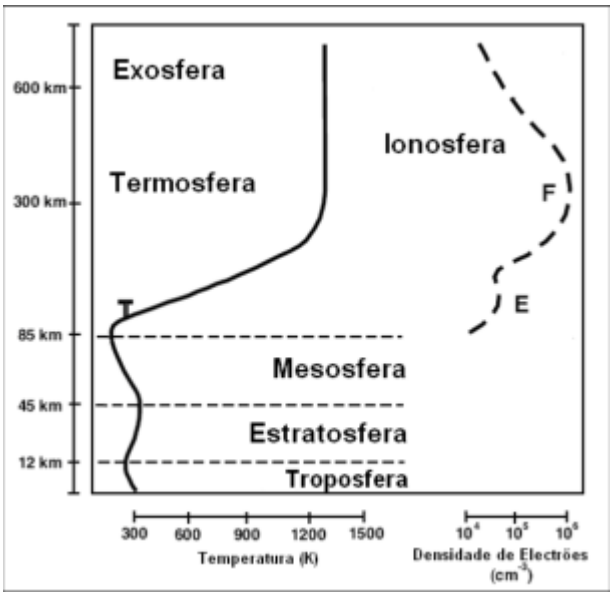
2.3 ATMOSFERA

A composição da atmosfera é de vários gases, de maneira que cerca de 99,98% é de nitrogênio, oxigênio, argônio e dióxido de carbono. Outro elemento relevante presente principalmente nas camadas mais baixas da atmosfera é o vapor d’água. O ciclo hidrilógico que acontece no planeta, evaporação, condensação e precipitação, torna imensamente variável espacial e temporalmente o conteúdo de

vapor d'água na atmosfera, de modo que a quantificação na constituição atmosférica se torna complexa (DA SILVA, DOS SANTOS e DE OLIVEIRA, 1999).

A atmosfera terrestre pode ser dividida em várias camadas, troposfera, estratosfera, mesosfera, termosfera e exosfera, conforme apresenta a Figura 5, (SILVA, 2007).

Figura 5 – Camadas da Atmosfera



Fonte: Silva (2007).

Os dois componentes da atmosfera que proporcionam mais problemas são a troposfera e a ionosfera (FRENCH, 1996). Atrasos atmosféricos são erros que necessitam ser corrigidos ou mitigados para poder se obter alta precisão de posicionamento ou tempo. A refração atmosférica causa erros nos sinais de GNSS em torno de 2 a 10 metros, causados pela Troposfera, e em torno de 10 a 30 metros, até perda de sinal, pela Ionosfera. Em contrapartida, atrasos atmosféricos estimados através de processamento de dados de GNSS são importantes fontes de informação para aplicações científica, como também, para pesquisas fundamentais sobre a atmosfera terrestre (ROYAL OBSERVATORY OF BELGIUM (ROB), 2010).

2.3.1 Refração Troposférica

Os sinais GNSS propagados pela atmosfera recebem interferência de vários erros, como o erro pertencente à troposfera. A determinação deste erro se deve as condições meteorológicas como umidade, pressão e temperatura, as quais

encontram-se no trajeto do satélite até o receptor percorrido pelo sinal. O efeito descrito anteriormente é denominado atraso zenital troposférico (ZTD – *Zenith Tropospheric Delay*) (CHUERUBIM, SGANTINE e DA SILVA). Para usuários sem acesso a dados meteorológicos, necessita-se da utilização de modelos que usam como base valores estatísticos (SILVA, 2007).

A troposfera é a camada mais baixa da atmosfera, com extensão da superfície terrestre até cerca de 16 km de altitude no equador e de 8 km nos pólos. Além disso, é composta de vapor d'água e aerossóis (KOS, BOTINČAN e DLESK, 2009). A espessura da camada da troposfera chega a medir até 50 km, de modo que os sinais GPS ao atravessarem a camada possuem sua velocidade de propagação reduzidas. Dessa forma, as pseudodistancias recebem um aumento em com referência à distância geométrica (CARVALHO, 1999). Para a faixa de frequência do GNSS, a Troposfera é um meio não dispersivo. O atraso troposférico pode ser separado em dois elementos: hidrostático e úmido. O primeiro é provocado pela influência da atmosfera hidrostática (ZHD – *Zenithal Hydrostatic Delay*) e reflete em cerca de 90% do atraso troposférico total. Como o ZHD possui pouca variação, ele pode ser estabelecido com precisão razoável. O elemento úmido se deve pela ação do vapor d'água atmosférico (ZWD – *Zenital Wet Delay*) e corresponde a cerca de 10% do atraso. No entanto, possui alta variação temporal e espacial, de maneira a tormar uma previsão apropriada a partir de medidas da umidade na superfície terrestre impossível (ALVES, ABREU e SOUZA, 2013).

Segundo (SAPUCCI e MONICO), a divisão da modelagem do atraso troposférico é feita em duas partes. Primeiramente, determina-se o atraso produzido no sinal GPS ao se propagar através da troposfera na direção zenital, chamado de atraso zenital troposférico. A seguir, associa-se o atraso zenital com o ângulo de elevação do satélite observado ao usar as qualificadas funções de mapeamento.

A variação do atraso troposférico nos sinais de RF do GNSS pode ser de 2 metros, na direção zenital, até 20 metros para ângulos menores que 10°. Para a redução dos erros referentes a troposfera, utilizam-se modelos globais, originados empiricamente de dados de radiossondas, os quais os modelos de Hopfield e Saastamoinen se destacam (CHUERUBIM, SGANTINE e DA SILVA).

2.3.2 Refração Ionosférica

O atraso ionosférico do GNSS é causado por processos físicos e químicos que ocorrem no espaço entre o Sol e a Terra. Radiações e Partículas lançados pelo Sol, chamado de vento solar, pode causar perturbações do campo magnético da Terra e alteração da distribuição vertical das partículas ionizadas na ionosfera. A Radiação do Sol provoca ionização das moléculas de gases, de maneira que libera elétrons livres. A proporção do atraso ionosférico é em relação a quantidade de elétrons livres (TEC – *total eléctron content*) detectado pelo sinal viajado do satélite até o receptor (KOS, BOTINČAN e DLESK, 2009). Em períodos de alta atividade solar ou com os satélites próximos ao horizonte são gerados os maiores erros nos resultados do GPS. Para períodos com baixa atividade solar, durante a noite ou com um satélite próximo ao zênite são gerados os menores erros ionosféricos (US. ARMY CORPS OF ENGINEERS, 1996).

O TEC depende de diferentes variáveis como mudanças no fluxo solar ionizante, hora, estações do ano, atividade magnética, ângulo de elevação e localização do usuário. Entre as latitudes de $\pm 15°$ estão os valores mais altos de TEC, utilizando como referência o equador geomagnético. Além disso, estes valores são dependentes do ângulo de elevação dos satélites, de maneira que se o ângulo de elevação diminui, o TEC aumenta (SILVA, 2007).

3 MATERIAIS E MÉTODOS

3.1 RECEPTOR GNSS

Existem vários tipos de equipamentos que realizam a função de receptor GNSS e que variam com a aplicatividade, como foi descrito nas seções anteriores. Devido a disponibilidade e praticidade, foi selecionado como receptor GNSS deste trabalho um aparelho celular. Dentre os aparelhos existentes, necessitava-se de amplo espaço de armazenamento, disponibilidade de aplicativo gratuito para realizar a coleta de dados e que pudesse permanecer por vários dias fixo. Com estas especificações, foi selecionado o Iphone 5s. A figura 6 a seguir exemplifica o celular mencionado.

Figura 6 - Iphone 5s

Fonte: www.zoom.com.br, 2018.

Na Figura 7, pode ser visualizado como ele foi instalado para poder realizar esta pesquisa. Para a melhor coleta de dados, foi selecionado um local em que recebesse o melhor sinal de GPS a todo momento e que não sofresse distúrbios que variassem a posição do celular. Desse modo, foi necessário fixa-lo com fitas, em cima de algumas caixas e estivesse no segundo pavimento da residência.

Figura 7 - Posicionamento do receptor GNSS

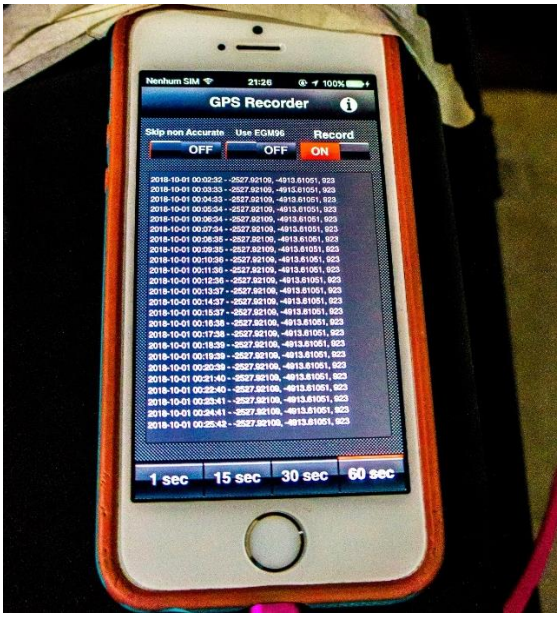
Fonte: A autora, 2018.

3.2 APLICATIVO PARA COLETA DE DADOS

Para realizar a coleta de dados pelo Iphone 5s, foi selecionado um aplicativo de GPS, gratuito, que coletasse e armazenasse dados no celular por um longo período e que estes dados possuíssem data, horário, latitude e longitude. No aplicativo da Apple Store foi encontrado o GPS Recorder, o qual cumpria com estas especificações.

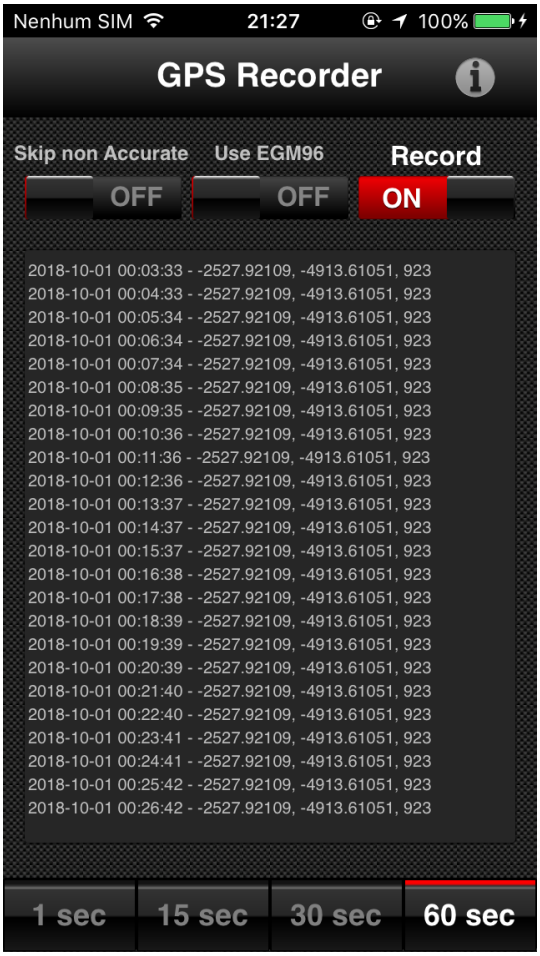
Neste aplicativo, como mostrado nas Figuras 8 e 9, pode ser escolhido o intervalo de tempo em que será coletado os dados, a cada $1\,s$, $15\,s$, $30\,s$ ou $60\,s$, como não era necessário coletar dados em intervalos pequenos devido a análise ser de várias semanas, foi selecionado o intervalo de $60\,s$. Como o objetivo deste trabalho é mitigar os erros no sinal GNSS, foi desabilitado a opção de não coletar dados não precisos (*Skip non Accurate*). A opção *Recorder* quando habilitada coleta os dados e depois quando desabilitada, salva os dados na nuvem do aparelho (*ICloud*), de maneira que possa ser analisada posteriormente.

Figura 8 - GPS Recorder no receptor GNSS



Fonte: A autora, 2018.

Figura 9 - GPS Recorder detalhado



Fonte: A autora, 2018.

Os dados adquiridos pelo GPS Recorder são no formato NMEA.log, o qual informa o tipo da mensagem, data, horário (UTC), latitude, longitude, altitude, número de satélites, entre outros dados, como pode ser evidenciado na Figura 10.

Figura 10 - Formato de dados NMEA

```
@Phoenix-GPSRecorder/v1.0.2/EGM96
$GPGGA,153424,2527.92405,S,4913.60419,W,1,04,1.0,908.73999,M,000.0,M,,*4C
$GPRMC,153424,A,2527.92405,S,4913.60419,W,000.0,000.0,290818,,,A*4D
$GPGGA,153439,2527.92405,S,4913.60419,W,1,04,1.0,904.78186,M,000.0,M,,*41
$GPRMC,153439,A,2527.92405,S,4913.60419,W,000.0,000.0,290818,,,A*41
$GPGGA,153454,2527.92405,S,4913.60419,W,1,04,1.0,904.501221,M,000.0,M,,*7F
$GPRMC,153454,A,2527.92405,S,4913.60419,W,000.0,000.0,290818,,,A*4A
```

Fonte: A autora, 2018.

3.3 BANCO DE DADO METEREOLOÓGICO

Para poder realizar as equações de mitigação de erros atmosféricos, utilizou-se um banco meteorológico do INMET (Instituto Nacional de Meteorologia) da cidade de Curitiba. A coleta de dados é realizada no campus Centro Politécnico da UFPR e são disponibilizados ao público no site do istituto (http://www.inmet.gov.br). Para isso, necessita-se especificar o período que deseja solicitar os dados. A Figura 11 mostra como os dados são fornecidos, nota-se que existe um aviso sobre a qualidade dos dados. Além disso, há períodos que ocorrem problemas na leitura de dados, como pode ser visto nos espaços em brancos nas colunas de umidade e ponto de orvalho. Como não são utilizados estes dados para os cálculos, não houve adversidades.

Figura 11 - Dados meteorológicos de Curitiba

Consulta Dados da Estação Automática: Curitiba (PR) Fechar

Observação: Estes são dados brutos e sem consistência com o único objetivo de deixá-los disponíveis de forma imediata.

Data Inicial: 20/08/2018 Data Final: 11/11/2018 Nova Pesquisa Download de Dados

Data	Hora	Temperatura (°C)			Umidade (%)			Pto. Orvalho (°C)			Pressão (hPa)			Vento (m/s)			Radiação	Chuva
	UTC	Inst.	Máx.	Mín.	Inst.	Máx.	Mín.	Inst.	Máx.	Mín.	Inst.	Máx.	Mín.	Vel.	Dir. (°)	Raj.	(kJ/m²)	(mm)
20/08/2018	00	16.2	17.9	16.2	70	70	61	10.7	10.8	10.2	917.3	917.3	916.9	0.7	60	2.6	-3.54	0.0
20/08/2018	01	15.3	16.2	15.3	77	77	70	11.3	11.3	10.8	917.2	917.3	917.1	0.6	49	2.4	-3.54	0.0
20/08/2018	02	15.1	15.4	15.1	84	84	77	12.4	12.4	11.3	917.4	917.4	917.2	0.8	44	2.7	-3.54	0.0
20/08/2018	03	14.0	15.1	14.0							917.2	917.5	917.2	0.0	54	2.0	-3.54	0.0
20/08/2018	04	13.3	14.1	13.2							917.0	917.2	917.0	0.1	75	1.6	-3.54	0.0
20/08/2018	05	12.7	13.3	12.7							916.6	917.0	916.5	0.0	149	2.3	-3.54	0.0
20/08/2018	06	12.1	12.7	11.7							916.4	916.7	916.3	0.0	190	0.4	-3.54	0.0
20/08/2018	07	11.7	12.3	11.5							916.5	916.5	916.4	0.0	35	0.9	-3.54	0.0
20/08/2018	08	11.5	11.9	11.2							916.9	916.9	916.5	0.0	355	0.5	-3.54	0.0
20/08/2018	09	11.6	11.9	11.3							917.4	917.5	916.8	0.0	59	1.3	1.695	0.0
20/08/2018	10	14.4	14.4	11.6	93			13.3			917.8	917.8	917.4	0.6	51	1.6	172.8	0.0
20/08/2018	11	19.0	19.0	14.4	61	93	60	11.2	12.4	11.0	918.1	918.1	917.8	1.3	55	2.6	1052.	0.0
20/08/2018	12	22.1	22.2	19.0	45	62	44	9.8	11.8	9.4	918.4	918.4	918.0	2.5	24	4.3	1772.	0.0
20/08/2018	13	24.5	25.0	22.1	34	45	33	7.6	10.2	7.2	918.2	918.4	918.2	2.4	43	5.3	2346.	0.0
20/08/2018	14	25.2	25.6	24.2	29	36	29	5.8	8.5	5.8	917.8	918.2	917.8	2.2	26	5.7	2676.	0.0
20/08/2018	15	25.6	26.1	24.3	29	33	28	6.4	7.7	5.7	917.2	917.8	917.2	3.0	291	5.7	2508.	0.0
20/08/2018	16	26.1	26.4	25.2	27	30	27	5.8	7.5	5.5	916.4	917.2	916.3	2.2	294	7.5	2713.	0.0
20/08/2018	17	26.3	26.8	25.7	27	28	25	6.1	6.3	5.0	916.0	916.4	916.0	2.9	258	5.8	2251.	0.0
20/08/2018	18	25.9	27.0	25.5	27	30	26	5.6	6.7	5.1	915.8	916.0	915.7	2.9	307	6.0	1604.	0.0
20/08/2018	19	24.9	26.1	24.9	33	33	27	7.4	7.7	5.6	916.2	916.2	915.8	2.3	312	6.0	1013.	0.0
20/08/2018	20	18.6	25.4	18.6	60	60	33	10.8	10.9	7.4	916.8	916.8	916.2	3.1	108	6.0	295.2	0.0
20/08/2018	21	15.9	18.6	15.9	79	79	60	12.2	12.2	10.8	917.4	917.4	916.8	1.9	111	6.0	-0.85	0.0
20/08/2018	22	14.8	16.0	14.8	85	85	78	12.4	12.4	12.0	917.9	917.9	917.3	1.1	99	4.2	-3.54	0.0
20/08/2018	23	14.4	15.0	14.4		100	85		14.5	12.3	918.2	918.2	917.9	2.3	44	5.7	-3.54	0.0
21/08/2018	00	14.4	14.5	14.3							918.3	918.5	918.2	2.4	48	5.7	-2.68	0.0
21/08/2018	01	13.9	14.4	13.9							918.3	918.4	918.3	2.7	45	6.4	-1.86	0.0
21/08/2018	02	13.8	14.0	13.8							918.3	918.5	918.2	1.1	48	5.5	-1.72	0.0
21/08/2018	03	13.5	13.7	13.4							917.7	918.4	917.7	0.8	59	4.2	-1.46	0.0
21/08/2018	04	13.5	13.6	13.4							917.2	917.8	917.2	1.2	66	4.5	-1.58	0.0
21/08/2018	05	13.3	13.5	13.2							917.2	917.3	917.1	1.0	32	3.9	-1.33	0.0

Fonte INMET, 2018.

3.4 SOFTWARES

Para este trabalho foi utilizado dois softwares: Netbeans, para realizar a programação de análise, e mitigação dos erros atmosféricos e QGIS, para demonstrar a localização dos pontos coletados pelo receptor. Estes softwares serão descritos a seguir.

3.4.1 NetBeans IDE

O software escolhido para ser feita a programação foi o NetBeans IDE. De acordo com o próprio website do aplicativo (netbeans.org), é uma ferramenta que proporciona desenvolvimento rápido e fácil de aplicativos em Java para *desktop*, celular e *web*, como também, aplicações HTML5 por meio de HTML, JavaScript e CSS. Além disso, é um software gratuito e livre, com ampla comunidade de usuários e desenvolvedores ao redor do mundo.

Este software foi utilizado para desenvolver um aplicativo para *desktop* em linguagem Java, de maneira que o usuário pudesse inserir dados na interface e recebesse a localização do receptor com os erros atmosféricos mitigados. Além disso, transmitisse ao usuário uma tabela no formato .csv com dados filtrados. Nesta tabela seriam desprezados os pontos que, de acordo com a autora, fossem erros de leitura, como o surgimento da localização de um ponto uma única vez.

3.4.2 QGIS

O QGIS é um Sistema de Informações Geográficas (GIS - *Geographic Information System*) de código aberto e de fácil utilização, licenciado sob o *GNU General Public License*. O software é um projeto oficial da *Open Source Geospatial Foundation* (OSGeo). Pode ser executado em Linux, Unix, Mac OSX, Windows e Android, além de suportar vários formatos e funcionalidades de vetor, raster e banco de dados. O QGIS não é apenas um GIS de *desktop*, ele também fornece um navegador de arquivos espaciais, um aplicativo de servidor e *web* aplicativos.

Neste trabalho, o QGIS foi utilizado para demonstrar no mapa os pontos coletados pelo receptor GNSS e os pontos gerados pelo software NetBeans IDE.

3.5 MITIGAÇÃO DO ERRO TROPOSFÉRICO

Para realizar a mitigação do erro troposférico, foi selecionado os dois modelos que receberam mais destaque durante a realização da revisão bibliográfica. Os modelos são de Hopfield e Saastamoinen, os quais serão descritos nas seções seguintes.

3.5.1 Modelo de Hopfield

Hopfield, em 1969, desenvolveu sua teoria com base em uma atmosfera com índice constante de declive da temperatura em relação ao aumento da altitude, para expressar a refratividade troposférica (DA SILVA, DOS SANTOS e DE OLIVEIRA, 1999). As equações a seguir demonstram este modelo (Monico, 2000 *apud* Seeber, 1993) (citado por (SAPUCCI e MONICO)):

$$D_{zh} = 155{,}2 \,.\, 10^{-7} . \frac{P_0}{T_0} \,.\, H_H \tag{2}$$

$$D_{zw} = 155{,}2 \,.\, 10^{-7} . \frac{4810 \,.\, e_0}{T_0^2} . H_W \tag{3}$$

onde o as medidas efetuadas na superfície são caracterizadas pelo sub índice 0. As unidades de medida que devem ser empregadas para a pressão e temperatura são hPa e $Kelvins$, respectivamente. A constante de refratividade do ar (k_1) e um coeficiente, o qual relaciona o atraso zenital com a espessura da camada atmosférica, são reprentadas pela constante comum nas equações 2 e 3. Os termos H_H e H_W representam as alturas em metros das camadas atmosféricas das componentes hidrostática e úmida, respectivamente, dados como:

$$H_H = 40136 + 148{,}72(T_0 - 273{,}16) \tag{4}$$

$$H_W = 11000 \quad (5)$$

Como demonstrado nas equações acima, H_H é dependente da temperatura e H_W é constante. Porém, conforme a latitude do local a espessura da camada que compreende o vapor d'água atmosférico pode mudar. Para locais próximos ao equador se atribui o valor mencionado na equação 5, de modo que para lugares perto dos pólos pode ser de 7000 metros (Hartman, 1994) (citado por (SAPUCCI e MONICO)). Com isso, pode ser realizada uma correção, ao utilizar a taxa de variação de H_W constante em relação às variações da latitude (φ) da localidade. Assim, a equação pode ser escrita como:

$$H_W = 11000 - 44{,}44 \,.\, \varphi \quad (6)$$

onde deve ser aplicado os valores da latitude em módulo e em unidade de arco decimal (SAPUCCI e MONICO).

3.5.2 Modelo de Saastamoinen

Saastamoinen, em 1973, desenvolveu um modelo fundamentado no pressuposto do decréscimo linear da temperatura até a tropopausa, a partir de teorias referentes a refração atmosférica. De modo a caracterizar a estratosfera como um modelo isotérmico, pois a temperatura adquiria um valor constante acima da tropopausa (DA SILVA, DOS SANTOS e DE OLIVEIRA, 1999). Segundo (SAPUCCI e MONICO), a equação do modelo de Saastamoinen pode ser descrito por (Saastamoinen, 1973):

$$D_{zh} = 0{,}002277 \,.\, D \,.\, P_0 \quad (7)$$

$$D_{zw} = 0{,}002277 \,.\, D \left(\frac{1255}{T_0} + 0{,}05\right) e_0 \quad (8)$$

onde as unidades devem ser dadas em $Kelvins$ para T, e para P_0 e e_0 em hPa. Os valores da constante k_1, constante gravitacional efetiva e constante específica dos gases hidrostáticos R_h estão contidos no termo constante $(0{,}002277)$ encontrado nas equações 7 e 8. A constante gravitacional depende da altura e da latitude do local. A aplicação do fator D se deve as correções para as variações da constante gravitacional efetiva, o qual é descrito como:

$$D = (1 + 0{,}0026 \,.\cos(2\varphi) + 0.00028 \,.\, h_0) \tag{9}$$

onde deve ser aplicado em quilômetros os valores referentes ao h_0.

3.6 MITIGAÇÃO DO ERRO IONOSFÉRICO

Segundo (SILVA, 2007), em receptores de apenas frequência L1, é necessário um algoritmo para mitigar o erro ionosférico. Para isso, pode-se utilizar o modelo de Klobuchar para receptores de GPS, o qual baseia-se numa representação em forma de cosseno da curva diurna, a qual pode variar em período e amplitude em função da latitude do usuário, e supõe o atraso constante no período noturno.

3.6.1.1 Modelo Klobuchar

Segundo (SILVA, 2007), o modelo Klobuchar apresenta quatro parâmetros (Kaplan e Hegarty, 2006), (Parkinson e Spilker, 1996):

- A_1 – Constante Noturna - $5\ ns$;
- A_2 – Amplitude da função cosseno;
- A_3 – Fase - $14\ horas$ (hora local);
- A_4 – Período da função cosseno - $72000\ s$.

No zênite o atraso ionosférico para a hora local t é dado pela equação a seguir:

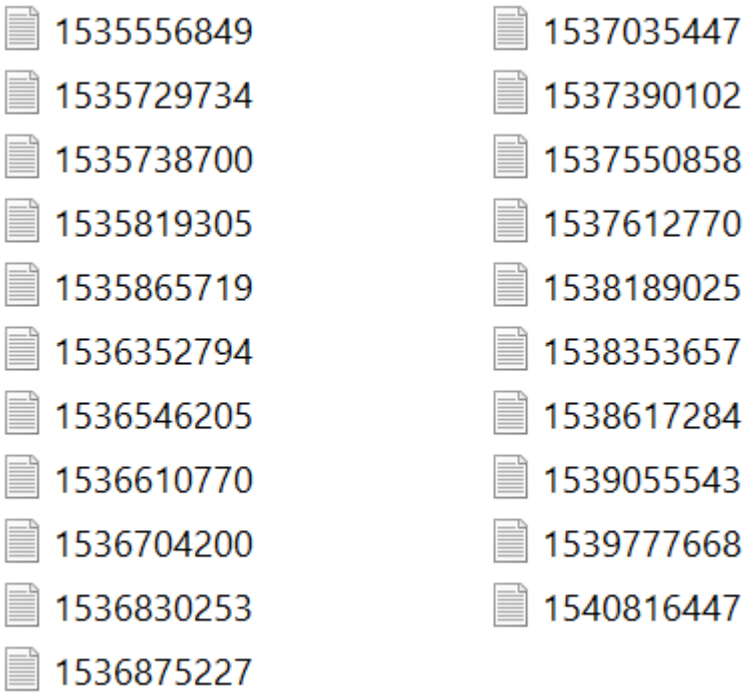
$$\frac{I_z, L1}{c} = \begin{cases} A_1 + A_2\ .\ cos\left(\dfrac{2\pi(t - A_3)}{A_4}\right), & se\ |t - A_3| < \dfrac{A_4}{4} \\ A_1, & caso\ contrário \end{cases} \tag{10}$$

4 ANÁLISE DOS RESULTADOS

4.1 DADOS DO RECEPTOR GNSS E METEOROLÓGICOS

Os dados gerados pelo GPS Recorder foram coletados do dia 28/08/2018 ao dia 05/11/2018 e para não ocorrer problemas com corrupção de arquivos, os dados foram divididos em vários arquivos. Para isso, após a coleta de dados por alguns dias, era salvo o arquivo e iniciado nova coleta de dados. A Figura 12 demonstra os arquivos que foram gerados neste processo.

Figura 12 - Arquivos com os dados coletados



Fonte: A autora, 2018.

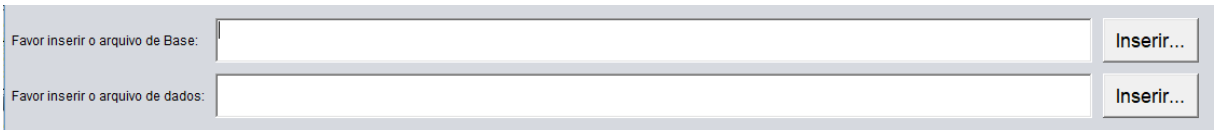
Para poder ser posteriormente analisados no programa NetBeans IDE, os dados coletados foram reunidos em um único arquivo no formato txt.

4.2 INFORMAÇÕES DO NETBEANS IDE

A primeira parte do desenvolvimento do aplicativo de *desktop* foi realizar a interface do usuário. A intenção principal é coletar os arquivos do usuário e fornecer ao final as coordenadas do ponto final, que haveria a mitigação dos erros

atmosféricos. Para isso, foi posicionado dois botões na parte superior, um para o usuário inserir o arquivo com a base de dados, que seria os dados meteorológicos, e outro para poder inserir os dados coletados pelo receptor GNSS, como pode ser visto na Figura 13.

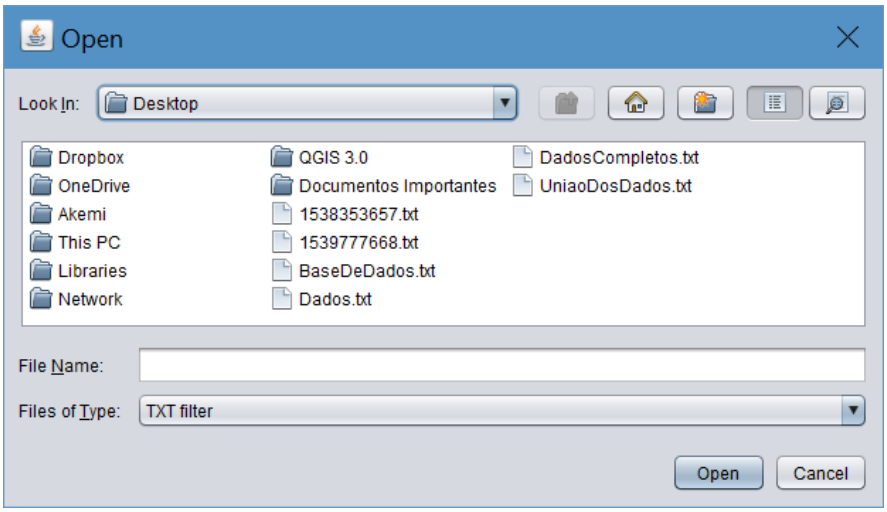
Figura 13 - Segmento para inserir os arquivos de dados



Fonte: A autora, 2018.

Como cada usuário vai possuir um caminho de diretório e nome de arquivo diferente, foi utilizado o comando *JFileChooser* que permite o usuário escolher o arquivo na pasta do computador, como é mostrado pela Figura 14. Além disso, para facilitar para o usuário encontrar o arquivo, foi realizado um filtro para mostrar apenas os arquivos no formato txt.

Figura 14 - Janela de seleção de arquivo



Fonte: A autora, 2018.

Para poder mostrar para o usuário que o arquivo foi selecionado, foi inserido duas janelas ao lado dos botões. As mensagens que são mostradas são referentes ao arquivo ter sido aberto corretamente, o usuário não ter selecionado algum arquivo ou se ocorreu algum erro ao tentar abrir o arquivo selecionado. Um exemplo destas opções é demonstrado na Figura 15 a seguir.

Figura 15 - Demonstração opção de mensagem

Fonte: A autora, 2018.

Após selecionar os arquivos de dados, foi realizada a transferência dos arquivos para tabelas, chamadas de *JTables*. A Tabela com os dados meteorológicos, foi chamada de Tabela de Base e pode ser vista na Figura 16. Nela foi realizada a adição de zeros nas células em que os dados não foram coletados, como descrito na seção 3.3. Para o arquivo com todos os dados coletados pelo receptor GNSS, foi separado em duas tabelas, uma contendo os dados referentes as leituras do tipo RMC e outra para as leituras do tipo GGA. As leituras do tipo RMC, chamado de *The Remommended Minimum* – o mínimo recomendado, é a própria versão do NMEA para os dados essenciais de GPS, como posição, velocidade e tempo, e as leituras GGA são *Global Positioning System Fix Data*, ou seja, sistema global de correção de dados de posicionamento. Além disso, foi imposta nas duas tabelas a correção descrita anteriormente de desconsiderar leituras únicas de pontos. Por fim, foi realizada as transformações de dados do modelo NMEA para os necessários para realizar as equações posteriormente, como pode ser visto na Figura 16. A seguir são descritas as colunas e demonstradas as transformações realizadas:

- Horário: formato 153424 para 15:34:24;
- Latitude: formato 2527.92405, S para -25.46540083;
- Longitude: formato 4913.60419, W para -49.2267365;
- Data: formato 290818 para 29/08/2018.

Figura 16 - Tabelas de Dados RMC e GGA

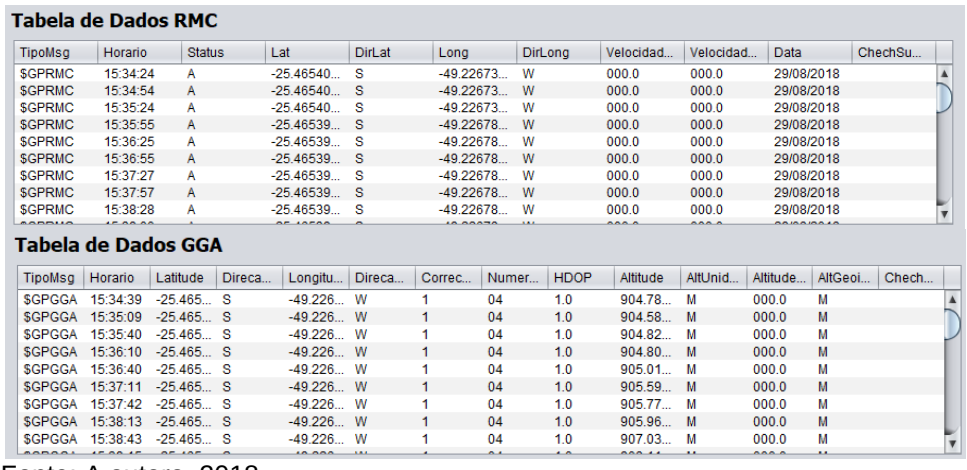


Tabela de Dados RMC

TipoMsg	Horario	Status	Lat	DirLat	Long	DirLong	Velocidad...	Velocidad...	Data	ChechSu...
$GPRMC	15:34:24	A	-25.46540...	S	-49.22673...	W	000.0	000.0	29/08/2018	
$GPRMC	15:34:54	A	-25.46540...	S	-49.22673...	W	000.0	000.0	29/08/2018	
$GPRMC	15:35:24	A	-25.46540...	S	-49.22673...	W	000.0	000.0	29/08/2018	
$GPRMC	15:35:55	A	-25.46539...	S	-49.22678...	W	000.0	000.0	29/08/2018	
$GPRMC	15:36:25	A	-25.46539...	S	-49.22678...	W	000.0	000.0	29/08/2018	
$GPRMC	15:36:55	A	-25.46539...	S	-49.22678...	W	000.0	000.0	29/08/2018	
$GPRMC	15:37:27	A	-25.46539...	S	-49.22678...	W	000.0	000.0	29/08/2018	
$GPRMC	15:37:57	A	-25.46539...	S	-49.22678...	W	000.0	000.0	29/08/2018	
$GPRMC	15:38:28	A	-25.46539...	S	-49.22678...	W	000.0	000.0	29/08/2018	

Tabela de Dados GGA

TipoMsg	Horario	Latitude	Direca...	Longitu...	Direca...	Correc...	Numer...	HDOP	Altitude	AltUnid...	Altitude...	AltGeoi...	Chech...
$GPGGA	15:34:39	-25.465...	S	-49.226...	W	1	04	1.0	904.78...	M	000.0	M	
$GPGGA	15:35:09	-25.465...	S	-49.226...	W	1	04	1.0	904.58...	M	000.0	M	
$GPGGA	15:35:40	-25.465...	S	-49.226...	W	1	04	1.0	904.82...	M	000.0	M	
$GPGGA	15:36:10	-25.465...	S	-49.226...	W	1	04	1.0	904.80...	M	000.0	M	
$GPGGA	15:36:40	-25.465...	S	-49.226...	W	1	04	1.0	905.01...	M	000.0	M	
$GPGGA	15:37:11	-25.465...	S	-49.226...	W	1	04	1.0	905.59...	M	000.0	M	
$GPGGA	15:37:42	-25.465...	S	-49.226...	W	1	04	1.0	905.77...	M	000.0	M	
$GPGGA	15:38:13	-25.465...	S	-49.226...	W	1	04	1.0	905.96...	M	000.0	M	
$GPGGA	15:38:43	-25.465...	S	-49.226...	W	1	04	1.0	907.03...	M	000.0	M	

Fonte: A autora, 2018.

Com isso, pode-se realizar os cálculos para a mitigação dos erros atmosféricos. Desse modo foi realizada outra tabela, chamada de Tabela de Resultados, na qual foram inseridas as colunas de data, horário, latitude e longitude da Tabela de Dados RMC, altitude da Tabela GGA, e temperatura e pressão da Tabela de Base. A seguir foi inserida duas novas colunas, uma para os resultados dos erros troposféricos hidrostráticos para o modelo de Hopfield e uma para o modelo de Saastamoinen, como pode ser evidenciado na Figura 17.

Figura 17 - Tabela de Resultados

Tabela de Resultados

Data	Horario	Lat	Long	Altitude	temp_inst	pressao	DzhHopfi...	DzhSaas...
29/08/2018	15:34:24	-25.4654...	-49.2267...	904.78186	24.0	917.2	2.093627...	2.092219...
29/08/2018	15:34:54	-25.4654...	-49.2267...	904.5832...	24.0	917.2	2.093627...	2.092218...
29/08/2018	15:35:24	-25.4654...	-49.2267...	904.8291...	24.0	917.2	2.093627...	2.092219...
29/08/2018	15:35:55	-25.4653...	-49.2267...	904.8071...	24.0	917.2	2.093627...	2.092219...
29/08/2018	15:36:25	-25.4653...	-49.2267...	905.0108...	24.0	917.2	2.093627...	2.092219...
29/08/2018	15:36:55	-25.4653...	-49.2267...	905.5905...	24.0	917.2	2.093627...	2.092219...

Fonte: A autora, 2018.

Para isso, foi necessário transformar as equações 2, 4, 7 e 9, para a linguagem Java, como demonstrado na Figura 18. Como pode-se notar, não foi realizado os cálculos para as componentes úmidas do erro troposférico e para o erro ionosférico. Isso se deve ao fato de não haver dados suficientes para realizar os cálculos, com o uso apenas das tabelas de dados coletados e meteorológicos.

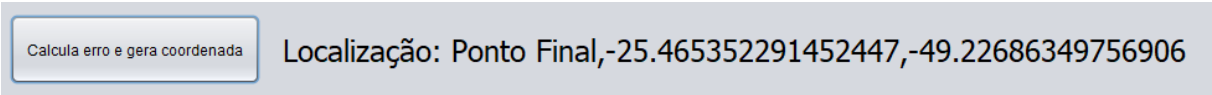
Figura 18 - Código para os modelos Hopfield e Saastamoinen

```
for(int i = 0; i < jTableMitErro.getRowCount(); i++)
{
    double pressao = Double.parseDouble(jTableMitErro.getValueAt(i, 6)+"");
    double temperatura = Double.parseDouble(jTableMitErro.getValueAt(i, 5)+"");
    double tempKelvin = temperatura+273.16;
    double altitude = Double.parseDouble(jTableMitErro.getValueAt(i, 4)+"");
    double h = altitude/1000;
    double angulo = Double.parseDouble(jTableMitErro.getValueAt(i, 2)+"");
    double fi = Math.toRadians(angulo);
    double doisfi = 2*fi;

     //Modelo Hopfield
    double Hh = 40136+(148.72*(tempKelvin-273.16));
    double DzhHopf = 155.2*(Math.pow(10, -7)*pressao*Hh)/tempKelvin ;
    jTableMitErro.setValueAt(DzhHopf, i, 7);

    //Modelo Saastamoinen
    double D = 1+(0.00266*Math.cos(doisfi))+(0.00028*h);
    double DzhSaas = 0.0022767*D*pressao;
    jTableMitErro.setValueAt(DzhSaas, i, 8);
}
```

Fonte: A autora, 2018.

Para poder ser gerada a Tabela de Resultados, foi feito um botão chamado “Calcula erro e gera coordenada”, o qual calcula e gera uma coordenada final para o usuário em formato CSV e mostra ela na caixa ao lado do botão, como exemplifica a Figura 19. Para realizar a coordenada final, foram utilizados os valores de latitude e longitude coletados e os valores de erros calculados no programa. Como os erros do modelo Hopfield e Saastamoinen possuem valores muito similares, foi escolhido o modelo Saastamoinen por utilizar mais dados em sua equação.

Figura 19 - Botão para cálculo de erro e da geração da coordenada ao lado



Fonte: A autora, 2018.

Para o cálculo da coordenada final foi inserido o erro troposférico hidrostático na latitude e na longitude. De maneira que a cada ponto de latitude teria um valor acima dele, ao somar o erro, e outro abaixo, ao diminuir o erro, e para cada ponto de longitude teria um valor a esquerda, ao diminuir o erro, e outro à direita, ao somar o erro. Seguidamente foi realizada a média das latitudes, através da soma todas as latitudes, as coletadas e as com erros, e divisão da quantidade total de latitudes. O mesmo foi realizado para a longitude, demonstrado na Figura 20.

Figura 20 - Código para calcular a coordenada final

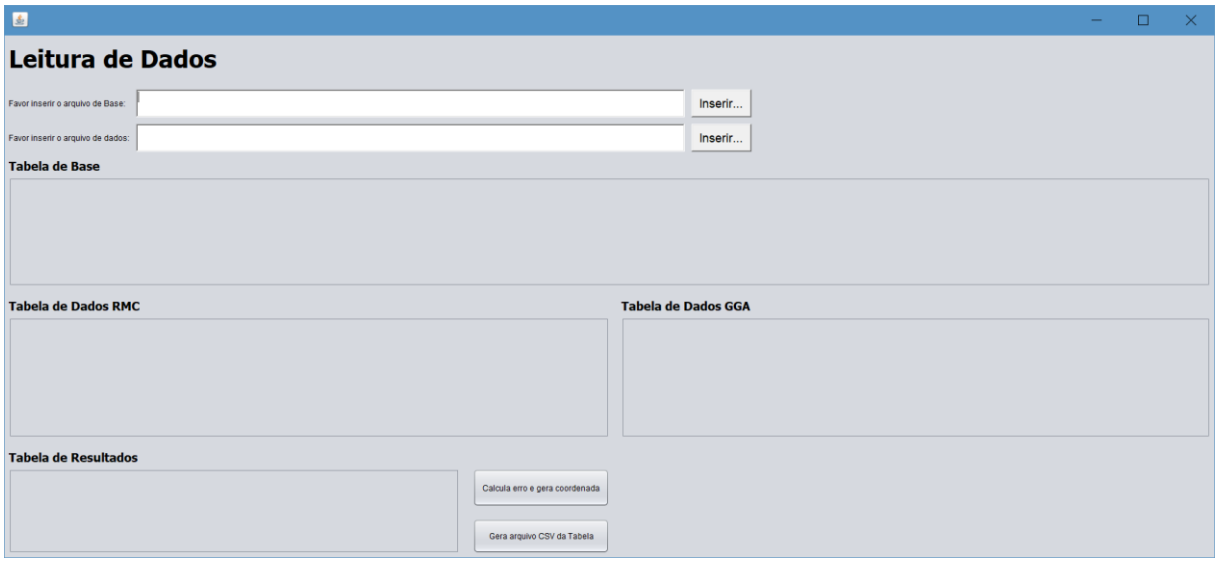
```
double somalat =0;
for(int i = 0; i < jTableMitErro.getRowCount(); i++)
{
    double lat = Double.parseDouble(jTableMitErro.getValueAt(i, 2)+"");
    double erroTropo = Double.parseDouble(jTableMitErro.getValueAt(i, 8)+"");
    double topo = lat+erroTropo;
    double baixo = lat-erroTropo;
    somalat = lat+somalat+topo+baixo;
}

double somalong =0;
for(int i = 0; i < jTableMitErro.getRowCount(); i++)
{
    double longi = Double.parseDouble(jTableMitErro.getValueAt(i, 3)+"");
    double erroTropo = Double.parseDouble(jTableMitErro.getValueAt(i, 8)+"");
    double topo = longi+erroTropo;
    double baixo = longi-erroTropo;
    somalong = longi+somalong+topo+baixo;
}

int numeroLinha = jTableMitErro.getRowCount();
int linhaTotal = numeroLinha*3;
double latTotal = somalat/linhaTotal;
double longTotal = somalong/linhaTotal;
```

Fonte: A autora, 2018.

Para poder verificar os pontos que foram utilizados para o cálculo do ponto com a mitigação do erro troposférico hidrostático, foi posto um botão para gerar a Tabela Resultados em arquivo CSV. As Figuras 21 e 21 demonstram a interface de usuário ao ser aberta e ao ser utilizada para a mitigação, respectivamente.

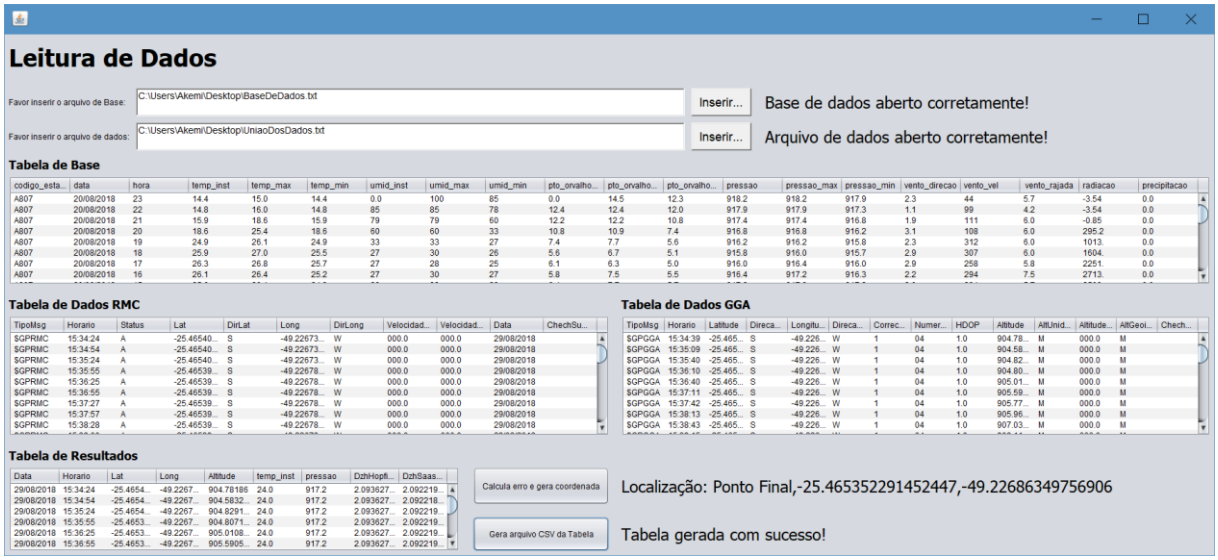
Além disso, o código completo desenvolvido para realizar a mitigação do erro troposférico hidrostático pode ser observado no Apêndice A.

Figura 21 - Interface de usuário ao iniciar



Fonte: A autora, 2018.

Figura 22 - Interface de usuário com dados



Fonte: A autora, 2018.

4.3 INFORMAÇÕES DO QGIS

O software QGIS foi empregado para demonstrar a localização dos pontos comentados durante este trabalho. Para isso, utilizou-se a camada Bing Satellite para adquirir a imagem de satélite da localização do receptor GNSS e camadas com os pontos em questão. Primeiramente a Figura 23 demonstra a localização do receptor

GNSS, exibido no formato de um celular, e de todos os pontos coletados pelo receptor, círculos vermelhos.

Figura 23 - Receptor e todos os pontos coletados

Fonte: A autora, 2018.

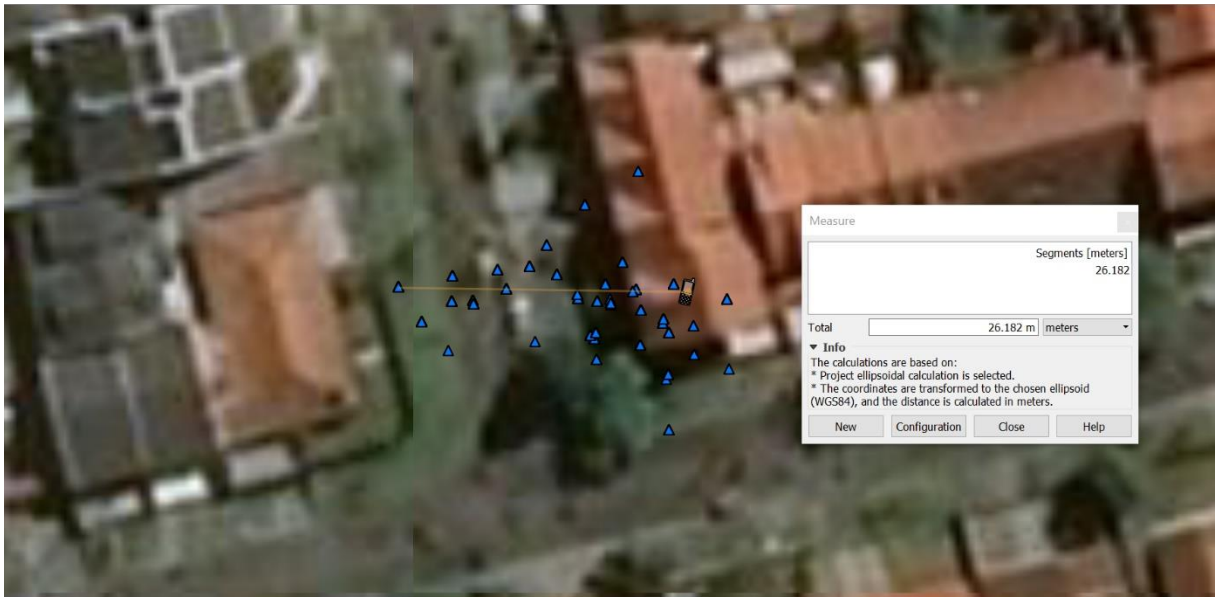
A seguir, foi realizado a comparação dos pontos coletados com os pontos gerados no NetBeans IDE, adquiridos da Tabela Resultados, como pode ser visto na Figura 24. Como pode ser analisado, foram desprezados três pontos nos cálculos.

Figura 24 - Receptor com pontos coletados e pontos da Tabela Resultados

Fonte: A autora, 2018.

Para demonstrar a necessidade da mitigação de erros dos dados coletados por receptores GNSS, foi realizada a medida de distância entre o ponto mais distante e o receptor, como mostra a Figura 25, através dos dados da Tabela Resultados.

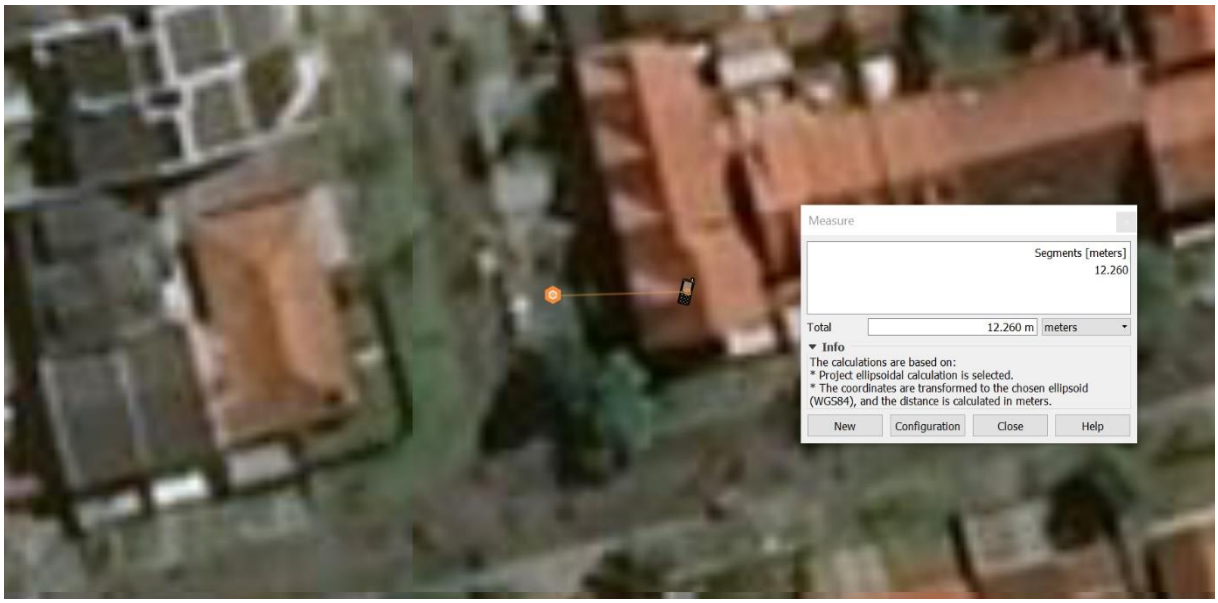
Figura 25 - Distância do ponto mais distante e receptor GNSS



Fonte: A autora, 2018.

Por fim, a Figura 26 mostra a localização do ponto em que foi realizada a mitigação de erro troposférico hidrostático e a distância até o receptor GNSS.

Figura 26 - Ponto com erro mitigado e distância até o receptor GNSS



Fonte: A autora, 2018.

5 CONCLUSÕES E TRABALHOS FUTUROS

A utilização de aparelho celular como receptor GNSS foi a forma encontrada para poder realizar a coleta de dados necessários para a realização deste trabalho. Dentro dos requisitos inicias considerados, ele conseguiu cumprir sem acarretar problemas. Porém, ao decorrer do levantamento de artigos e livros, notou-se que este aparelho não possui condições de gerar dados confiáveis.

Os dados meteorológicos do INMET, os quais foram utilizados como base para este trabalho, foram suficientes apenas para os cálculos de erro troposférico hidrostático. Ao levar em consideração que para o erro troposférico úmido, ao utilizar os modelos Hopfield e Saastamoinen, e para o erro ionosférico, pelo modelo Klobuchar, são necessárias outras variáveis, que não são coletadas. De modo que não foi possível realizar estes cálculos e gerar um ponto mais preciso.

O software NetBeans IDE foi a componente mais importante para a realização deste trabalho, de maneira que foi possível realizar uma interface de usuário simples e fácil, inserir todos os dados dispostos e calcular todas as variáveis necessárias de maneira clara. Ele cumpriu com todas as expectativas e tornou o desenvolvimento dos códigos o direto possível.

As imagens, pelo software QGIS, utilizadas para demonstrar a localização dos pontos importantes citados neste trabalho foram realizadas de forma simples e fácil. Não houve nenhum problema ao serem geradas e foi possível alterar os ícones dos pontos de maneira clara para melhor exemplificar os dados.

Os modelos de mitigação de erro troposférico hidrostático de Hopfield e de Saastamoinen mostraram ser viáveis e de resultados similares. De modo que foi possível ter uma distância entre o receptor e o ponto final de apenas 12 metros, praticamente metade do valor em relação ao ponto mais distante, de 26 metros. Assim, pode-se notar que os efeitos meteorológicos realmente afetam a leitura dos receptores GNSS, principalmente os de uso diário.

Para trabalhos futuros, deve-se avaliar a utilização de um receptor GNSS que gere dados de confiança maior e leve em consideração o emprego de outros dados como base para os cálculos necessários para realizar a mitigação dos erros atmosféricos.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, C. M. D.; MONICO, J. F. G.; ROMÃO, V. M. C. ANÁLISE DA ACURÁCIA NO PPP A PARTIR DA SOLUÇÃO DE AMBIGUIDADES GPS EM CURTOS PERÍODOS DE OCUPAÇÃO. **Revista Brasileira de Cartografia (2011) N 0 63/5: 589-600**, p. 585-600, 26 Fevereiro 2011.

ALVES, D. B. M.; ABREU, P. A. G. D.; SOUZA, J. S. GNSS: status, modelagem atmosférica e métodos de posicionamento, Pato Branco, p. 13, 2013.

CARVALHO, L. D. ANÁLISE DAS TÉCNICAS GPS ATUAIS PARA OS POSICIONAMENTOS ESTÁTICOS E CINEMÁTICOS EM BASES CURTAS, p. 158, 1999.

CHUERUBIM, M. L.; SGANTINE, P. C.; DA SILVA, I. O IMPACTO DOS MODELOS TROPOSFÉRICOS NO POSICIONAMENTO GNSS DE ALTA PRECISÃO EM DIFERENTES ÉPOCAS E ESTAÇÕES DO ANO. **Revista de Engenharia e Tecnologia**, v. 8, n. 2016.

DA SILVA, N. C. C.; DOS SANTOS, M. C.; DE OLIVEIRA, L. C. EFEITO DA REFRAÇÃO TROPOSFÉRICA NO POSICIONAMENTO GEODÉSICO COM GPS. **Brazilian Journal of Geophysics**, v. 17(2,3), 1999.

DIXON, T. H. AN INTRODUCTION TO THE GLOBAL POSITIONING SYSTEM AND SOME GEOLOGICAL APPLICATIONS. **American Geophysical Union**, Pasadena, California, p. 249-276, 1991.

ERROR Correction. **Trimble**. Disponivel em: <https://www.trimble.com/gps_tutorial/howgps-error.aspx>. Acesso em: 28 Setembro 2018.

ESA. Satellite-based augmentation systems worldwide. **ESA**, 26 Janeiro 2018. Disponivel em: <https://www.esa.int/spaceinimages/Images/2018/01/Satellite-based_augmentation_systems_worldwide>. Acesso em: 28 Setembro 2018.

FRENCH, G. T. **UNDERSTANDING THE GPS - An Introduction to the Global Positioning System**. [S.l.]: GeoResearch, Inc., 1996.

GROUNDS, T. US Official: Foreign GNSS Signals Need FCC Authorization for Use in United States. **the good word groundswell**, 2014. Disponivel em: <http://unbonmotgroundswell.blogspot.com/2014/12/us-official-foreign-gnss-signals-need.html>. Acesso em: 27 Setembro 2018.

KOS, T.; BOTINČAN, M.; DLESK, A. MITIGATING GNSS POSITIONING ERRORS DUE TO ATMOSPHERIC SIGNAL DELAYS, Zagreb, Croacia, p. 495 - 513, 02 Novembro 2009.

LAGO, I. F. D. INTEGRAÇÃO GPS E GLONASS APLICADA AOS LEVANTAMENTOS GEODÉSICOS, Curitiba, n. Universidade Federal do Paraná, p. 101, 2001.

MONICO, J. F. G. Posicionamento pelo NAVSTAR-GPS Descrição, fundamentos e aplicações. 1. ed. [S.l.]: UNESP, 2000. Cap. 1.

POLEZEL, W. G. C. INVESTIGAÇÕES SOBRE O IMPACTO DA MODERNIZAÇÃO DO GNSS NO POSICIONAMENTO, Presidente Prudente, 2010.

ROYAL OBSERVATORY OF BELGIUM (ROB). ATMOSPHERE. **Royal Observatory of Belgium GNSS Research Group**, 2010. Disponivel em: <http://gnss.be/atmosphere_tutorial.php>. Acesso em: 03 Novembro 2018.

SAPUCCI, L. F.; MONICO, J. F. G. AVALIAÇÃO DOS MODELOS DE HOPFIELD E DE SAASTAMOINEN PARA A MODELAGEM DO ATRASO ZENITAL TROPOSFÉRICO EM TERRITÓRIO BRASILEIRO UTILIZANDO GPS.

SEEBER, G. Satellite Geodesy. [S.l.]: de Gruyter; 2nd Compl. REV. and Extend. ed. edition (June 19, 2003), 2003. p. 612.

SILVA, T. M. L. D. S. Análise de Erros em Receptores de GNSS, Outubro 2007.

SILVA, T. S. Error Analysus in GNSS receivers, Outubro 2007.

US. ARMY CORPS OF ENGINEERS. GPS Absolute Positioning Determination Concepts, Errors, and Accuracies. In: _____ **NAVSTAR GLOBAL POSITIONING SYSTEM SURVEYING**. [S.l.]: [s.n.], 1996. Cap. 5.

APÊNDICE A – CÓDIGO DESENVOLVIDO NO NETBEANS IDE

C:\Users\Akemi\Documents\NetBeansProjects\TCC_GPS_testedeloucura\src\MitigacaoErro.java

```
 1
 2 import java.awt.Color;
 3 import java.io.BufferedReader;
 4 import java.io.File;
 5 import java.io.FileReader;
 6 import java.io.FileWriter;
 7 import java.io.IOException;
 8 import java.util.logging.Level;
 9 import java.util.logging.Logger;
10 import javax.swing.JFileChooser;
11 import javax.swing.filechooser.FileNameExtensionFilter;
12 import javax.swing.table.DefaultTableModel;
13
14 /*
15  * To change this license header, choose License Headers in Project Properties.
16  * To change this template file, choose Tools | Templates
17  * and open the template in the editor.
18  */
19
20 /**
21  *
22  * @author Karen Akemi Kobayashi
23  */
24 public class MitigacaoErro extends javax.swing.JFrame {
25
26     JFileChooser chooserDados;
27     JFileChooser chooserBase;
28     JFileChooser saveFileChooser;
29     /**
30      * Creates new form MitigacaoErro
31      */
32     public MitigacaoErro() {
33         initComponents();
34         chooserDados = new JFileChooser();
35         chooserDados.setFileFilter(new FileNameExtensionFilter("TXT filter","txt"));
36
37         chooserBase = new JFileChooser();
```

```
38        chooserBase.setFileFilter(new FileNameExtensionFilter("TXT
filter","txt"));
39
40        saveFileChooser = new JFileChooser();
41        saveFileChooser.setFileFilter(new FileNameExtensionFilter("CSV
filter","csv"));
42    }
43
44    /**
45     * This method is called from within the constructor to initialize the
form.
46     * WARNING: Do NOT modify this code. The content of this method is
always
47     * regenerated by the Form Editor.
48     */
49    @SuppressWarnings("unchecked")
50    // <editor-fold defaultstate="collapsed" desc="Generated Code">
51    private void initComponents() {
52
53        jScrollPane3 = new javax.swing.JScrollPane();
54        jTable1 = new javax.swing.JTable();
55        jLabel1 = new javax.swing.JLabel();
56        jLabel2 = new javax.swing.JLabel();
57        CaminhoDados = new java.awt.TextField();
58        InsereDados = new java.awt.Button();
59        CaixaDeMensagemCSV = new javax.swing.JLabel();
60        jScrollPane1 = new javax.swing.JScrollPane();
61        jTableRMC = new javax.swing.JTable();
62        jScrollPane2 = new javax.swing.JScrollPane();
63        jTableGGA = new javax.swing.JTable();
64        jLabel3 = new javax.swing.JLabel();
65        CaminhoBase = new java.awt.TextField();
66        InsereBase = new java.awt.Button();
67        jScrollPane4 = new javax.swing.JScrollPane();
68        jTableBase = new javax.swing.JTable();
69        CaixaDeMensagemBase = new javax.swing.JLabel();
70        jScrollPane5 = new javax.swing.JScrollPane();
71        jTableMitErro = new javax.swing.JTable();
72        CalculaMitErro = new javax.swing.JButton();
73        jLabel4 = new javax.swing.JLabel();
74        jLabel5 = new javax.swing.JLabel();
75        jLabel6 = new javax.swing.JLabel();
76        jLabel7 = new javax.swing.JLabel();
77        GeraArquivoCSV = new javax.swing.JButton();
```

```
78      CaixaDeMensagemPontoFinal = new javax.swing.JLabel();
79      CaixaDeMensagemDados = new javax.swing.JLabel();
80
81      jTable1.setModel(new javax.swing.table.DefaultTableModel(
82          new Object [][] {
83              {null, null, null, null},
84              {null, null, null, null},
85              {null, null, null, null},
86              {null, null, null, null}
87          },
88          new String [] {
89              "Title 1", "Title 2", "Title 3", "Title 4"
90          }
91      ));
92      jScrollPane3.setViewportView(jTable1);
93
94
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
95
96      jLabel1.setFont(new java.awt.Font("Tahoma", 1, 36)); // NOI18N
97      jLabel1.setText("Leitura de Dados");
98
99      jLabel2.setText("Favor inserir o arquivo de dados:");
100
101      CaminhoDados.setBackground(new java.awt.Color(255, 255, 255));
102      CaminhoDados.setEditable(false);
103      CaminhoDados.setFont(new java.awt.Font("Dialog", 0, 14)); //
NOI18N
104      CaminhoDados.addActionListener(new
java.awt.event.ActionListener() {
105          public void actionPerformed(java.awt.event.ActionEvent evt) {
106              CaminhoDadosActionPerformed(evt);
107          }
108      });
109
110      InsereDados.setFont(new java.awt.Font("Dialog", 0, 18)); // NOI18N
111      InsereDados.setLabel("Inserir...");
112      InsereDados.addActionListener(new java.awt.event.ActionListener() {
113          public void actionPerformed(java.awt.event.ActionEvent evt) {
114              InsereDadosActionPerformed(evt);
115          }
116      });
117
```

```
118         CaixaDeMensagemCSV.setFont(new java.awt.Font("Tahoma", 0, 24));
// NOI18N
119
120         jTableRMC.setModel(new javax.swing.table.DefaultTableModel(
121             new Object [][] {
122
123             },
124             new String [] {
125
126             }
127         ));
128         jTableRMC.setEnabled(false);
129         jScrollPane1.setViewportView(jTableRMC);
130
131         jTableGGA.setModel(new javax.swing.table.DefaultTableModel(
132             new Object [][] {
133
134             },
135             new String [] {
136
137             }
138         ));
139         jTableGGA.setEnabled(false);
140         jScrollPane2.setViewportView(jTableGGA);
141
142         jLabel3.setText("Favor inserir o arquivo de Base:");
143
144         CaminhoBase.setBackground(new java.awt.Color(255, 255, 255));
145         CaminhoBase.setEditable(false);
146         CaminhoBase.setFont(new java.awt.Font("Dialog", 0, 14)); // NOI18N
147         CaminhoBase.addActionListener(new java.awt.event.ActionListener()
{
148             public void actionPerformed(java.awt.event.ActionEvent evt) {
149                 CaminhoBaseActionPerformed(evt);
150             }
151         });
152
153         InsereBase.setFont(new java.awt.Font("Dialog", 0, 18)); // NOI18N
154         InsereBase.setLabel("Inserir...");
155         InsereBase.addActionListener(new java.awt.event.ActionListener() {
156             public void actionPerformed(java.awt.event.ActionEvent evt) {
157                 InsereBaseActionPerformed(evt);
158             }
159         });
```

```
160
161         jTableBase.setModel(new javax.swing.table.DefaultTableModel(
162             new Object [][] {
163
164             },
165             new String [] {
166
167             }
168         ));
169         jTableBase.setEnabled(false);
170         jScrollPane4.setViewportView(jTableBase);
171
172         CaixaDeMensagemBase.setFont(new java.awt.Font("Tahoma", 0, 24));
// NOI18N
173
174         jTableMitErro.setEnabled(false);
175         jScrollPane5.setViewportView(jTableMitErro);
176
177         CalculaMitErro.setText("Calcula erro e gera coordenada");
178         CalculaMitErro.addActionListener(new
java.awt.event.ActionListener() {
179             public void actionPerformed(java.awt.event.ActionEvent evt) {
180                 CalculaMitErroActionPerformed(evt);
181             }
182         });
183
184         jLabel4.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
185         jLabel4.setText("Tabela de Base");
186
187         jLabel5.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
188         jLabel5.setText("Tabela de Dados RMC");
189
190         jLabel6.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
191         jLabel6.setText("Tabela de Dados GGA");
192
193         jLabel7.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
194         jLabel7.setText("Tabela de Resultados");
195
196         GeraArquivoCSV.setText("Gera arquivo CSV da Tabela");
197         GeraArquivoCSV.addActionListener(new
java.awt.event.ActionListener() {
198             public void actionPerformed(java.awt.event.ActionEvent evt) {
199                 GeraArquivoCSVActionPerformed(evt);
200             }
```

```
201         });
202
203         CaixaDeMensagemPontoFinal.setFont(new java.awt.Font("Tahoma",
0, 24)); // NOI18N
204
205         CaixaDeMensagemDados.setFont(new java.awt.Font("Tahoma", 0,
24)); // NOI18N
206
207         javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
208         getContentPane().setLayout(layout);
209         layout.setHorizontalGroup(
210
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
211             .addGroup(layout.createSequentialGroup()
212                 .addContainerGap()
213
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
214                     .addComponent(jScrollPane4)
215                     .addComponent(jLabel1,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
216                     .addGroup(layout.createSequentialGroup()
217
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
218                         .addComponent(jLabel3)
219                         .addComponent(jLabel2))
220
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
221
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING, false)
222                         .addComponent(CaminhoDados,
javax.swing.GroupLayout.DEFAULT_SIZE, 813, Short.MAX_VALUE)
223                         .addComponent(CaminhoBase,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
224
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
225
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
```

```
226                     .addComponent(InsereBase,
javax.swing.GroupLayout.PREFERRED_SIZE, 89,
javax.swing.GroupLayout.PREFERRED_SIZE)
227                     .addComponent(InsereDados,
javax.swing.GroupLayout.PREFERRED_SIZE, 89,
javax.swing.GroupLayout.PREFERRED_SIZE))
228                 .addGap(20, 20, 20)
229
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
230                     .addComponent(CaixaDeMensagemBase,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
231                     .addComponent(CaixaDeMensagemDados,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
232                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
233
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
234                     .addGroup(layout.createSequentialGroup()
235
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TR
AILING)
236                         .addGroup(layout.createSequentialGroup()
237                             .addComponent(jScrollPane5)
238                             .addGap(20, 20, 20)
239
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING, false)
240                                 .addComponent(CalculaMitErro,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
241                                 .addComponent(GeraArquivoCSV,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
242                         .addComponent(jScrollPane1))
243                     .addGap(18, 18, 18))
244                 .addGroup(layout.createSequentialGroup()
245                     .addComponent(jLabel5,
javax.swing.GroupLayout.PREFERRED_SIZE, 600,
javax.swing.GroupLayout.PREFERRED_SIZE)
246                     .addGap(308, 308, 308)))
```

```
247
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
248                     .addComponent(jLabel6,
javax.swing.GroupLayout.PREFERRED_SIZE, 570,
javax.swing.GroupLayout.PREFERRED_SIZE)
249                     .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 873,
javax.swing.GroupLayout.PREFERRED_SIZE)
250                     .addComponent(CaixaDeMensagemPontoFinal,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
251                     .addComponent(CaixaDeMensagemCSV,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
252                 .addGroup(layout.createSequentialGroup()
253
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
254                     .addComponent(jLabel4,
javax.swing.GroupLayout.PREFERRED_SIZE, 1107,
javax.swing.GroupLayout.PREFERRED_SIZE)
255                     .addComponent(jLabel7))
256                     .addGap(0, 0, Short.MAX_VALUE)))
257             .addContainerGap())
258         );
259         layout.setVerticalGroup(
260
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
261             .addGroup(layout.createSequentialGroup()
262                 .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 68,
javax.swing.GroupLayout.PREFERRED_SIZE)
263
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
264
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
265                 .addComponent(CaminhoBase,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

```
266                 .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
267                 .addComponent(InsereBase,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
268                 .addComponent(CaixaDeMensagemBase,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE))
269
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
270
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
271                 .addComponent(CaminhoDados,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
272                 .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
273                 .addComponent(InsereDados,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE)
274                 .addComponent(CaixaDeMensagemDados,
javax.swing.GroupLayout.PREFERRED_SIZE, 41,
javax.swing.GroupLayout.PREFERRED_SIZE))
275
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
276                 .addComponent(jLabel4)
277
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
278                 .addComponent(jScrollPane4,
javax.swing.GroupLayout.PREFERRED_SIZE, 161,
javax.swing.GroupLayout.PREFERRED_SIZE)
279                 .addGap(18, 18, 18)
280
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
281                 .addComponent(jLabel5)
282                 .addComponent(jLabel6))
283
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
284
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
285                 .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 178,
javax.swing.GroupLayout.PREFERRED_SIZE)
286                 .addComponent(jScrollPane2,
javax.swing.GroupLayout.PREFERRED_SIZE, 178,
javax.swing.GroupLayout.PREFERRED_SIZE))
287             .addGap(18, 18, 18)
288             .addComponent(jLabel7)
289
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
290
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
291                 .addComponent(jScrollPane5,
javax.swing.GroupLayout.PREFERRED_SIZE, 125,
javax.swing.GroupLayout.PREFERRED_SIZE)
292                 .addGroup(layout.createSequentialGroup()
293
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
294                     .addComponent(CalculaMitErro,
javax.swing.GroupLayout.PREFERRED_SIZE, 56,
javax.swing.GroupLayout.PREFERRED_SIZE)
295                     .addComponent(CaixaDeMensagemPontoFinal,
javax.swing.GroupLayout.PREFERRED_SIZE, 56,
javax.swing.GroupLayout.PREFERRED_SIZE))
296                 .addGap(18, 18, 18)
297
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
298                     .addComponent(GeraArquivoCSV,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
299                     .addComponent(CaixaDeMensagemCSV,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))
300             .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
301         );
302
303         pack();
```

```
304    }// </editor-fold>
305
306    private void
InsereDadosActionPerformed(java.awt.event.ActionEvent evt) {
307        // TODO add your handling code here:
308        int returnValue = chooserDados.showOpenDialog(this);
309
310        if (returnValue == JFileChooser.APPROVE_OPTION) {
311          try
312          {
313             File inputFileDados = chooserDados.getSelectedFile();
314             FileReader inputReaderDados = new FileReader(inputFileDados);
315             BufferedReader inputBRDados = new
BufferedReader(inputReaderDados);
316             CaminhoDados.setText(inputFileDados.toString());
317
318             String[] cabecalhoRMC = new String[]
319             {
320                "TipoMsg", "Horario","Status", "Lat","DirLat", "Long","DirLong",
"VelocidadeNos", "VelocidadeKm/h", "Data", "ChechSumData"
321             };
322             String[] cabecalhoGGA = new String[]
323             {
324                "TipoMsg", "Horario", "Latitude","DirecaoLat",
"Longitude","DirecaoLong", "CorrecaoGPS", "NumeroSatelites", "HDOP",
"Altitude", "AltUnidade", "AltitudeGeoide", "AltGeoideUnidade",
"ChechSumData"
325             };
326
327             DefaultTableModel modeloRMC =
(DefaultTableModel)jTableRMC.getModel();
328             modeloRMC.setColumnIdentifiers(cabecalhoRMC);
329             DefaultTableModel modeloGGA =
(DefaultTableModel)jTableGGA.getModel();
330             modeloGGA.setColumnIdentifiers(cabecalhoGGA);
331
332             // get lines from txt file
333             Object[] tableLinhas = inputBRDados.lines().toArray();
334
335             // extratct data from lines
336             // set data to jtable model
337             for (Object tableLinha : tableLinhas)
338             {
339                String linha = tableLinha.toString().trim();
```

```
340             if (linha.startsWith("$GPRMC"))
341             {
342                 String[] dadosLinhaRMC = linha.split(",");
343                 modeloRMC.addRow(dadosLinhaRMC);
344             }
345             if (linha.startsWith("$GPGGA"))
346             {
347                 String[] dadosLinhaGGA = linha.split(",");
348                 modeloGGA.addRow(dadosLinhaGGA);
349             }
350         }
351 
352         for(int i = 1; i < jTableRMC.getRowCount(); i++)
353         {
354             int linhaAnterior = i-1;
355             Object latitude = jTableRMC.getValueAt(i, 2);
356             Object latAnt = jTableRMC.getValueAt(linhaAnterior, 2);
357             Object longitude = jTableRMC.getValueAt(i, 3);
358             Object longAnt = jTableRMC.getValueAt(linhaAnterior, 3);
359 
360             if( (latitude != latAnt) && (longitude != longAnt)  )
361             {
362                 modeloRMC.removeRow(i);
363             }
364 
365         }
366         for(int i = 1; i < jTableGGA.getRowCount(); i++)
367         {
368             int linhaAnterior = i-1;
369             Object latitude = jTableGGA.getValueAt(i, 2);
370             Object latAnt = jTableGGA.getValueAt(linhaAnterior, 2);
371             Object longitude = jTableGGA.getValueAt(i, 3);
372             Object longAnt = jTableGGA.getValueAt(linhaAnterior, 3);
373 
374             if( (latitude != latAnt) && (longitude != longAnt)  )
375             {
376                 modeloGGA.removeRow(i);
377             }
378         }
379 
380         for(int i = 0; i < jTableRMC.getRowCount(); i++)
381         {
382             String horario = jTableRMC.getValueAt(i, 1).toString();
383             String hora = horario.substring(0, 2);
```

```
384             String min = horario.substring(2, 4);
385             String seg = horario.substring(4, 6);
386             String hm = hora + ":" + min;
387             String hms = hm + ":" + seg;
388             jTableRMC.setValueAt(hms, i, 1);
389
390             String calendario = jTableRMC.getValueAt(i, 9).toString();
391             String dia = calendario.substring(0, 2);
392             String mes = calendario.substring(2, 4);
393             String ano = calendario.substring(4, 6);
394             String dm = dia + "/" + mes;
395             String dma = dm + "/20" + ano;
396             jTableRMC.setValueAt(dma, i, 9);
397
398
399             double latitude = Double.parseDouble(jTableRMC.getValueAt(i,
3)+"");
400             int grausLat = (int)(latitude*0.01);
401             double minLat = latitude - (grausLat*100);
402             double decLat = minLat/60;
403             double grausDecLat = grausLat + decLat;
404
405             double longitude =
Double.parseDouble(jTableRMC.getValueAt(i, 5)+"");
406             int grausLong = (int)(longitude*0.01);
407             double minLong = longitude - (grausLong*100);
408             double decLong = minLong/60;
409             double grausDecLong = grausLong + decLong;
410
411             String dirLat = jTableRMC.getValueAt(i, 4).toString();
412             String dirLong = jTableRMC.getValueAt(i, 6).toString();
413
414             if("S".equals(dirLat))
415             {
416                 double negLat = -grausDecLat;
417                 jTableRMC.setValueAt(negLat, i, 3);
418             }
419             if("N".equals(dirLat))
420             {
421                 jTableRMC.setValueAt(grausDecLat, i, 3);
422             }
423             if("W".equals(dirLong))
424             {
425                 double negLong = -grausDecLong;
```

```
426                 jTableRMC.setValueAt(negLong, i, 5);
427               }
428               else
429               {
430                 jTableRMC.setValueAt(grausDecLong, i, 5);
431               }
432             }
433
434             for(int k = 0; k < jTableGGA.getRowCount(); k++)
435             {
436               String horario = jTableGGA.getValueAt(k, 1).toString();
437               String hora = horario.substring(0, 2);
438               String min = horario.substring(2, 4);
439               String seg = horario.substring(4, 6);
440               String hm = hora + ":" + min;
441               String hms = hm + ":" + seg;
442               jTableGGA.setValueAt(hms, k, 1);
443
444
445               double latitude = Double.parseDouble(jTableGGA.getValueAt(k,
2)+"");
446               int grausLat = (int)(latitude*0.01);
447               double minLat = latitude - (grausLat*100);
448               double decLat = minLat/60;
449               double grausDecLat = grausLat + decLat;
450
451               double longitude =
Double.parseDouble(jTableGGA.getValueAt(k, 4)+"");
452               int grausLong = (int)(longitude*0.01);
453               double minLong = longitude - (grausLong*100);
454               double decLong = minLong/60;
455               double grausDecLong = grausLong + decLong;
456
457               String dirLat = jTableGGA.getValueAt(k, 3).toString();
458               String dirLong = jTableGGA.getValueAt(k, 5).toString();
459
460               if("S".equals(dirLat))
461               {
462                 double negLat = -grausDecLat;
463                 jTableGGA.setValueAt(negLat, k, 2);
464               }
465               if("N".equals(dirLat))
466               {
467                 jTableGGA.setValueAt(grausDecLat, k, 2);
```

```
468             }
469             if("W".equals(dirLong))
470             {
471                double negLong = -grausDecLong;
472                jTableGGA.setValueAt(negLong, k, 4);
473             }
474             else
475             {
476                jTableGGA.setValueAt(grausDecLong, k, 4);
477             }
478          }
479
480          inputBRDados.close();
481          inputReaderDados.close();
482          mensagemDados("Arquivo de dados aberto corretamente!",
false);
483       }
484       catch (IOException ioe) {
485          mensagemDados("Erro ao abrir arquivo de dados!", true);
486       }
487    }
488    else
489    {
490       mensagemDados("Arquivo de dados nao escolhido!",true);
491    }
492 }
493
494 private void
CaminhoDadosActionPerformed(java.awt.event.ActionEvent evt) {
495    // TODO add your handling code here:
496 }
497
498 private void
CaminhoBaseActionPerformed(java.awt.event.ActionEvent evt) {
499    // TODO add your handling code here:
500 }
501
502 private void InsereBaseActionPerformed(java.awt.event.ActionEvent
evt) {
503    // TODO add your handling code here:
504    int returnValue = chooserBase.showOpenDialog(this);
505
506    if (returnValue == JFileChooser.APPROVE_OPTION)
507    {
```

```
508         try
509         {
510            File inputFileBase = chooserBase.getSelectedFile();
511            FileReader inputReaderBase = new FileReader(inputFileBase);
512            BufferedReader inputBRBase = new
BufferedReader(inputReaderBase);
513            CaminhoBase.setText(inputFileBase.toString());
514
515            String firstLine = inputBRBase.readLine().trim();
516            String[] cabecalhoBase = firstLine.split(",");
517            DefaultTableModel modeloBase =
(DefaultTableModel)jTableBase.getModel();
518            modeloBase.setColumnIdentifiers(cabecalhoBase);
519
520            // get lines from txt file
521            Object[] tableLines = inputBRBase.lines().toArray();
522
523            // extratct data from lines
524            // set data to jtable model
525            for (Object tableLine : tableLines) {
526               String line = tableLine.toString().trim();
527               String[] dataRow = line.split(",");
528               modeloBase.addRow(dataRow);
529            }
530
531            for(int i = 0; i < jTableBase.getRowCount(); i++)
532            {
533               for(int j = 0; j < jTableBase.getColumnCount(); j++)
534               {
535                  if(jTableBase.getValueAt(i, j).toString().equals(""))
536                  {
537                     double a = 0.0;
538                     jTableBase.setValueAt(a, i, j);
539                  }
540               }
541            }
542
543            inputBRBase.close();
544            inputReaderBase.close();
545            mensagemBase("Base de dados aberto corretamente!", false);
546         }
547         catch (IOException ioe)
548         {
549            mensagemBase("Erro ao abrir Base de dados!", true);
```

```
550         }
551       }
552       else
553       {
554         mensagemBase("Base de dados não escolhida!",true);
555       }
556     }
557
558     private void
CalculaMitErroActionPerformed(java.awt.event.ActionEvent evt) {
559
560       DefaultTableModel modeloMitErro =
(DefaultTableModel)jTableMitErro.getModel();
561       String[] cabecalhoMitErro = new String[]
562       {
563         jTableRMC.getColumnName(9), jTableRMC.getColumnName(1),
jTableRMC.getColumnName(3), jTableRMC.getColumnName(5),
jTableGGA.getColumnName(9), jTableBase.getColumnName(3),
jTableBase.getColumnName(12), "DzhHopfield", "DzhSaastamoinen"
564       };
565       modeloMitErro.setColumnIdentifiers(cabecalhoMitErro);
566       jTableMitErro.setAutoCreateRowSorter(true);
567
568       for(int i = 0; i < jTableRMC.getRowCount(); i++)
569       {
570         String[] conteudoRMCA = new String[]
571         {
572           jTableRMC.getValueAt(i, 9).toString(), jTableRMC.getValueAt(i,
1).toString(), jTableRMC.getValueAt(i, 3).toString(), jTableRMC.getValueAt(i,
5).toString()
573         };
574         modeloMitErro.addRow(conteudoRMCA);
575       }
576
577       for(int i = 0; i < jTableMitErro.getRowCount(); i++)
578       {
579         Object colunaAltitude = jTableGGA.getValueAt(i, 9);
580         jTableMitErro.setValueAt(colunaAltitude, i, 4);
581       }
582
583       for(int i = 0; i < jTableMitErro.getRowCount(); i++)
584       {
585         for(int j = 0; j < jTableBase.getRowCount(); j++)
586         {
```

```
587             String dataMitErro = jTableMitErro.getValueAt(i, 0).toString();
588             String horarioMitErro = jTableMitErro.getValueAt(i, 1).toString();
589             String horaMitErro = horarioMitErro.substring(0, 2);
590 
591             String dataBase = jTableBase.getValueAt(j, 1).toString();
592             String horaBase = jTableBase.getValueAt(j, 2).toString();
593 
594             if( (dataBase.equals(dataMitErro)) &&
(horaBase.equals(horaMitErro)) )
595             {
596                Object colunaTemperatura = jTableBase.getValueAt(j, 3);
597                Object colunaPressao = jTableBase.getValueAt(j, 12);
598                jTableMitErro.setValueAt(colunaTemperatura, i, 5);
599                jTableMitErro.setValueAt(colunaPressao, i, 6);
600             }
601          }
602       }
603 
604       for(int i = 0; i < jTableMitErro.getRowCount(); i++)
605       {
606          double b = 0.0;
607          if(jTableMitErro.getValueAt(i, 5) == null)
608          {
609             jTableMitErro.setValueAt(b, i, 5);
610          }
611          if(jTableMitErro.getValueAt(i, 6) == null)
612          {
613             jTableMitErro.setValueAt(b, i, 6);
614          }
615       }
616 
617       for(int i = 0; i < jTableMitErro.getRowCount(); i++)
618       {
619          double pressao = Double.parseDouble(jTableMitErro.getValueAt(i,
6)+"");
620          double temperatura =
Double.parseDouble(jTableMitErro.getValueAt(i, 5)+"");
621          double tempKelvin = temperatura+273.16;
622          double altitude = Double.parseDouble(jTableMitErro.getValueAt(i,
4)+"");
623          double h = altitude/1000;
624          double angulo = Double.parseDouble(jTableMitErro.getValueAt(i,
2)+"");
625          double fi = Math.toRadians(angulo);
```

```
626         double doisfi = 2*fi;
627
628          //Modelo Hopfield
629         double Hh = 40136+(148.72*(tempKelvin-273.16));
630         double DzhHopf = 155.2*(Math.pow(10, -
7)*pressao*Hh)/tempKelvin ;
631         jTableMitErro.setValueAt(DzhHopf, i, 7);
632
633         //Modelo Saastamoinen
634         double D = 1+(0.00266*Math.cos(doisfi))+(0.00028*h);
635         double DzhSaas = 0.0022767*D*pressao;
636         jTableMitErro.setValueAt(DzhSaas, i, 8);
637       }
638
639
640       double somalat =0;
641       for(int i = 0; i < jTableMitErro.getRowCount(); i++)
642       {
643         double lat = Double.parseDouble(jTableMitErro.getValueAt(i, 2)+"");
644         double erroTropo =
Double.parseDouble(jTableMitErro.getValueAt(i, 8)+"");
645         double topo = lat+erroTropo;
646         double baixo = lat-erroTropo;
647         somalat = lat+somalat+topo+baixo;
648       }
649
650       double somalong =0;
651       for(int i = 0; i < jTableMitErro.getRowCount(); i++)
652       {
653         double longi = Double.parseDouble(jTableMitErro.getValueAt(i,
3)+"");
654         double erroTropo =
Double.parseDouble(jTableMitErro.getValueAt(i, 8)+"");
655         double topo = longi+erroTropo;
656         double baixo = longi-erroTropo;
657         somalong = longi+somalong+topo+baixo;
658       }
659
660       int numeroLinha = jTableMitErro.getRowCount();
661       int linhaTotal = numeroLinha*3;
662       double latTotal = somalat/linhaTotal;
663       double longTotal = somalong/linhaTotal;
664
```

```
665         String caminhoCSVPF =
"C:\\Users\\Akemi\\Desktop\\PontoFinal.csv";
666         String latString = Double.toString(latTotal);
667         String longString = Double.toString(longTotal);
668         String ponto = "Ponto Final,"+latString+","+longString;
669         try {
670             FileWriter csvPF = new FileWriter(new File(caminhoCSVPF));
671             csvPF.write(ponto);
672             csvPF.close();
673             mensagemMitErro("Localização: "+ponto, false);
674         }
675         catch (IOException ex)
676         {
677
Logger.getLogger(MitigacaoErro.class.getName()).log(Level.SEVERE, null, ex);
678             mensagemMitErro("Problema ao realizar a mitigação de erros!",
true);
679         }
680
681     }
682
683     private void
GeraArquivoCSVActionPerformed(java.awt.event.ActionEvent evt) {
684
685         String pathToExportTo =
"C:\\Users\\Akemi\\Desktop\\TxtsCombinados.csv";
686
687         try {
688             FileWriter csv = new FileWriter(new File(pathToExportTo));
689
690             for(int i = 0; i < jTableMitErro.getColumnCount(); i++)
691             {
692                 csv.write(jTableMitErro.getColumnName(i) + ",");
693             }
694
695             csv.write("\n");
696
697             for (int i = 0; i < jTableMitErro.getRowCount(); i++) {
698                 for (int j = 0; j < jTableMitErro.getColumnCount(); j++) {
699                     csv.write(jTableMitErro.getValueAt(i, j).toString() + ",");
700                 }
701                 csv.write("\n");
702             }
703
```

```
704             csv.close();
705             mensagemCSV("Tabela gerada com sucesso!", false);
706         }
707         catch (IOException ex)
708         {
709 
Logger.getLogger(MitigacaoErro.class.getName()).log(Level.SEVERE, null, ex);
710             mensagemCSV("Erro ao gerar tabela!", true);
711         }
712 
713     }
714 
715     private void mensagemDados(String msg, boolean isError)
716     {
717         if (isError)
718         {
719             CaixaDeMensagemDados.setText(msg);
720             CaixaDeMensagemDados.setForeground(Color.RED);
721         }
722         else
723         {
724             CaixaDeMensagemDados.setText(msg);
725             CaixaDeMensagemDados.setForeground(Color.BLACK);
726         }
727     }
728 
729     private void mensagemBase(String msg, boolean isError)
730     {
731         if (isError)
732         {
733             CaixaDeMensagemBase.setText(msg);
734             CaixaDeMensagemBase.setForeground(Color.RED);
735         }
736         else
737         {
738             CaixaDeMensagemBase.setText(msg);
739             CaixaDeMensagemBase.setForeground(Color.BLACK);
740         }
741     }
742 
743     private void mensagemCSV(String msg, boolean isError)
744     {
745         if (isError)
746         {
```

```
747         CaixaDeMensagemCSV.setText(msg);
748         CaixaDeMensagemCSV.setForeground(Color.RED);
749       }
750       else
751       {
752         CaixaDeMensagemCSV.setText(msg);
753         CaixaDeMensagemCSV.setForeground(Color.BLACK);
754       }
755     }
756 
757     private void mensagemMitErro(String msg, boolean isError)
758     {
759       if (isError)
760       {
761         CaixaDeMensagemPontoFinal.setText(msg);
762         CaixaDeMensagemPontoFinal.setForeground(Color.RED);
763       }
764       else
765       {
766         CaixaDeMensagemPontoFinal.setText(msg);
767         CaixaDeMensagemPontoFinal.setForeground(Color.BLACK);
768       }
769     }
770 
771     /**
772      * @param args the command line arguments
773      */
774     public static void main(String args[]) {
775       /* Set the Nimbus look and feel */
776       //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">
777       /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
778        * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
779        */
780       try {
781         for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
782           if ("Nimbus".equals(info.getName())) {
783             javax.swing.UIManager.setLookAndFeel(info.getClassName());
784             break;
785           }
786         }
```

```
787         } catch (ClassNotFoundException ex) {
788 
java.util.logging.Logger.getLogger(MitigacaoErro.class.getName()).log(java.util
.logging.Level.SEVERE, null, ex);
789         } catch (InstantiationException ex) {
790 
java.util.logging.Logger.getLogger(MitigacaoErro.class.getName()).log(java.util
.logging.Level.SEVERE, null, ex);
791         } catch (IllegalAccessException ex) {
792 
java.util.logging.Logger.getLogger(MitigacaoErro.class.getName()).log(java.util
.logging.Level.SEVERE, null, ex);
793         } catch (javax.swing.UnsupportedLookAndFeelException ex) {
794 
java.util.logging.Logger.getLogger(MitigacaoErro.class.getName()).log(java.util
.logging.Level.SEVERE, null, ex);
795         }
796         //</editor-fold>
797 
798         /* Create and display the form */
799         java.awt.EventQueue.invokeLater(new Runnable() {
800             public void run() {
801                 new MitigacaoErro().setVisible(true);
802             }
803         });
804     }
805 
806     // Variables declaration - do not modify
807     private javax.swing.JLabel CaixaDeMensagemBase;
808     private javax.swing.JLabel CaixaDeMensagemCSV;
809     private javax.swing.JLabel CaixaDeMensagemDados;
810     private javax.swing.JLabel CaixaDeMensagemPontoFinal;
811     private javax.swing.JButton CalculaMitErro;
812     private java.awt.TextField CaminhoBase;
813     private java.awt.TextField CaminhoDados;
814     private javax.swing.JButton GeraArquivoCSV;
815     private java.awt.Button InsereBase;
816     private java.awt.Button InsereDados;
817     private javax.swing.JLabel jLabel1;
818     private javax.swing.JLabel jLabel2;
819     private javax.swing.JLabel jLabel3;
820     private javax.swing.JLabel jLabel4;
821     private javax.swing.JLabel jLabel5;
822     private javax.swing.JLabel jLabel6;
```

```
823     private javax.swing.JLabel jLabel7;
824     private javax.swing.JScrollPane jScrollPane1;
825     private javax.swing.JScrollPane jScrollPane2;
826     private javax.swing.JScrollPane jScrollPane3;
827     private javax.swing.JScrollPane jScrollPane4;
828     private javax.swing.JScrollPane jScrollPane5;
829     private javax.swing.JTable jTable1;
830     public javax.swing.JTable jTableBase;
831     public javax.swing.JTable jTableGGA;
832     public javax.swing.JTable jTableMitErro;
833     public javax.swing.JTable jTableRMC;
834     // End of variables declaration
835
836
837 }
838
```