Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

# Estudo e cálculo da órbita lunar aplicado a comunicações EME

Filipe Carreiro Soares

Dissertação submetida como requisito parcial para obtenção do grau de

**Mestre em Engenharia de Telecomunicações e Informática**

Orientador:
Prof. Doutor Francisco António Bucho Cercas
ISCTE-IUL

Co-orientador:
Prof. Doutor Pedro Joaquim Amaro Sebastião
ISCTE-IUL

Outubro, 2017

# Resumo

Num mundo cada vez mais global, em que a rapidez e fiabilidade na troca de informação é cada vez mais crucial, verifica-se que existe uma centralização, e até monopolização, nos métodos e meios de partilha de informação. Quando se pretende efetuar uma comunicação via ondas rádio entre grandes distâncias, na qual existe a necessidade de se contornar os obstáculos naturais da Terra, são utilizados atualmente satélites artificiais que atuam como repetidores/refletores do sinal enviado novamente para o planeta Terra.

Durante a realização desta dissertação, procedeu-se ao estudo das comunicações via ondas rádios de longo alcance que utilizam a Lua como repetidor passivo de sinal. Foi também realizado o estudo aos elementos orbitais e métodos de cálculo da órbita lunar a partir da Terra, sendo deste modo possível efetuar o cálculo da posição da Lua face a uma localização geográfica na Terra para qualquer instante definido.

O estudo efetuado foi aplicado no desenvolvimento de um *software*, intitulado ItTracker, que é capaz de calcular a posição da Lua (e Sol), em tempo real ou definido, dada as coordenadas geográficas de uma localização específica na Terra. Esta dissertação apresenta soluções para que, se consiga efetuar uma comunicação via ondas rádio de longo alcance que utilize a Lua como repetidor/refletor passivo de sinal.

**Palavras Chave:** Lua, repetidor passivo de sinal, Elementos Orbitais, Cálculo, *Software*, ItTracker.

.

# Abstract

In an increasingly global world, where speed and reliability in the exchange of information is increasingly crucial, there is a centralization, and even monopolization, of the methods and means of information sharing. When it is intended to perform a radio waves communication between long distances, in which there is the need to circumvent the Earth natural obstacles, nowadays artificial satellites are used that act as repeaters/reflectors of the signal sent back to the planet Earth.

During the realization of this dissertation, the long-range radio wave communications using the Moon as a passive signal repeater was studied. The study of the orbital elements and methods of calculation of the lunar orbit from the Earth was also carried out, thus making it possible to calculate the position of the Moon given a geographic location on Earth at any given time.

The study was applied in the development of a *software*, named ItTracker, that can perform the calculation of the position of the Moon (and Sun), in real time or user defined, given the geographical coordinates of a specific location on Earth. This dissertation presents solutions so that it is possible to make a long-range radio wave communication using the Moons as a passive signal repeater/reflector

**Keywords:** Moon, Passive signal repeater, Orbital Elements, Calculation, *Software*, ItTracker.

# Agradecimentos

Em primeiro lugar gostaria de agradecer ao meu supervisor, Prof. Francisco Cercas, pelo contínuo apoio e motivação prestada ao longo do tempo em que se realizou esta dissertação. Gostaria também de agradecer ao Prof. José Sanguino, pela gentileza de ter disponibilizado a minha presença na cadeira de Sistemas de Posicionamento de Telecomunicações de Satélite, lecionada no IST Tagus Park, que serviu de base para os conhecimentos adquiridos.

De seguida, gostaria de agradecer aos meus irmãos, avós, tios e primos, por todo o apoio e motivação prestados ao longo dos anos.

Uma palavra também a todos os meus colegas de serviço, por me estarem sempre a motivar no sentido de terminar este capítulo importante na minha vida.

O meu agradecimento mais especial vai para o meu Pai, Mãe e Maria João. O vosso apoio, dedicação e confiança depositada em mim foi essencial e crucial para a realização desta dissertação. Sem vocês, não teria sido possível concluí-la.

# Índice

**Resumo** iii

**Abstract** v

**Agradecimentos** vii

**Lista de Figuras** xi

**Lista de Tabelas** xiii

**Lista de Acrónimos** xiv

**Lista de Símbolos** xv

**Introdução** 1

1.1 Motivação e Enquadramento ... 1

1.2 Objetivos ... 3

1.3 Organização ... 4

**Estado da Arte** 5

2.1 Comunicações via ondas rádio de longo alcance ... 5

2.1.1 Comunicações EME e rádio amadores ... 7

2.1.2 Implicações de uma comunicação EME a 10 GHz ... 7

2.2 Estudo Orbital ... 10

2.2.1 Leis de Kepler ... 10

2.2.2 Periélio/Perigeu e Afélio/Apogeu ... 12

2.2.3 Eclíptica da Terra ... 13

2.2.4 Eclíptica da Lua ... 14

2.2.5 Posição Heliocêntrica, Geocêntrica e Topocêntrica ... 14

2.2.6 Coordenadas Equatoriais (Hour Angle, Right Ascenscion e Declination) ... 16

2.2.7 Longitude e Latitude ... 18

2.2.8 Azimute e Elevação ... 18

2.2.9 Julian Day e Modified Julian Day ... 20

**Cálculo Orbital** 21

3.1 Elementos Orbitais ... 21

3.2 Algoritmos Computacionais ... 22

3.2.1 Especificações do software ItTracker ... 23

3.2.2 JD e MJD ... 24

3.2.3 Elementos Orbitais, Latitude e Longitude eclíptica ..... 25
3.2.4 Perturbações na Lua ..... 27
3.2.5 Right Ascenscion e Declination ..... 28
3.2.6 Hour Angle (GMST) ..... 29
3.2.7 Topocentric Right Ascension e Declination ..... 30
3.2.8 Azimute e Elevação ..... 31
3.2.9 Conversões e Normalizações ..... 32

**Software ItTracker e Resultados** **33**

4.1 Especificações e Diagrama de classes ..... 33
4.2 ItTracker GUI Inputs ..... 35
4.2.1 Latitude e Longitude ..... 35
4.2.2 Local Time e Set Time ..... 36
4.2.3 Check Connection ..... 38
4.2.4 Botões Calculate e Stop ..... 38
4.2.5 Visão Geral Inputs ..... 39
4.3 ItTracker GUI Output ..... 39
4.3.1 Output – Lua ..... 39
4.3.2 Output – Sol ..... 42
4.3.3 Ficheiro “input.dat” ..... 42
4.3.4 GUI Visão Geral ..... 43
4.4 Resultados ..... 44

**Conclusões** **51**

**Referências Bibliográficas** **53**

**Anexos** **55**

Anexo A Funções – Normalizações e Conversões ..... 56
Anexo B Manual de Instruções ..... 60
Anexo C Listagem de código fonte ..... 61

# Lista de Figuras

2.1: 1ª Lei de Kepler ..... 11
2.2: 2ª Lei de Kepler (lei das áreas) ..... 11
2.3: Posição de Periélio e Afélio ..... 12
2.4: Perigeu e Apogeu da Lua ..... 13
2.5: Obliquidade da Eclíptica da Terra ..... 13
2.6: Inclinação da Lua sobre a eclíptica da Terra ..... 14
2.7: Posição Topocêntrica e eixo de coordenadas ..... 15
2.8: Hour Angle (HA) e Declination (Dec) ..... 17
2.9: Right Ascension e Declination ..... 17
2.10: Azimute Cartográfico ..... 19
2.11: Ilustração de Elevação numa Parabólica ..... 19
2.12: Conjunto Azimute e Elevação ..... 20
3.1: Margem de Erro ..... 23
4.1: Versão Java ..... 33
4.2: Software Eclipse e Sistema Operativo Windows ..... 34
4.3: Diagrama de Classes ItTracker ..... 34
4.4: Input de Latitude e Longitude de posição na Terra ..... 35
4.5: Input correto de Latitude e Longitude ..... 36
4.6: Checkbox Local Time Selecionada ..... 36
4.7: Checkbox Set Time Selecionada ..... 37
4.8: Formato Input Hora e Data ..... 37
4.9: Input de Hora e Data modo Set Time ..... 37
4.10: Check Destination - Latitude e Longitude de Destino ..... 38
4.11: Botões Calculate e Stop ..... 38
4.12: Visão Geral Inputs ..... 39
4.13: Output painel Moon Orbital Values ..... 40
4.14: Output Azimuth and Elevation ..... 40
4.15: Representação de Azimute em Bussola ..... 41
4.16: Representação gráfica da Elevação ..... 41

4.17: Informação mostrada quando a ligação é possível ..... 41
4.18: Informação mostrada quando a ligação não é possível ..... 41
4.19: Output Sun painel Orbital Values ..... 42
4.20: Output Ficheiro "Input.dat" ..... 43
4.21: Visão Geral software ItTracker ..... 43
4.22: ItTracker resultados Lua - 1 ..... 45
4.23: MoonCalc.org resultado Lua - 1 ..... 45
4.24: ItTracker resultados Lua - 2 ..... 46
4.25: MoonCalc.org resultados Lua - 2 ..... 47
4.26: ItTracker resultado “Connection Impossible” ..... 48
4.27: ItTracker resultados Sol ..... 49
4.28: MoonCalc.org resultados Sol ..... 49

# Lista de Tabelas

1 Valores típicos de perdas de propagação. Valores de dimensão e tipo de antenas, G, e TxPwr ..... 9
2 Valores de ruído térmico provocados por diversas fontes associados a uma frequência de utilização. ..... 10

# Lista de Acrónimos

| | |
|---|---|
| 2D | Duas dimensões |
| 3D | Três dimensões |
| CW | Continuous Wave |
| UTC | Coordinated Universal Time |
| Dec | Declination |
| dB | Décibel |
| dBi | Décibel Isotrópico |
| EME | Earth-Moon-Earth |
| GHz | Giga Hertz |
| GUI | Graphical Unit Interface |
| HA | Hour Angle |
| JD | Julian Day |
| K | Kelvin |
| Kg | Quilogramas |
| MHz | Mega Hertz |
| MCW | Modulated Continuous Wave |
| MJD | Modified Julian Date |
| RA | Right Ascension |
| UHF | Ultra High Frequency band |
| VHF | Very High Frequency band |
| SHF | Super High Frequency band |
| W | Watt |
| WGS 84 | World Geodetic System 84 |

# Lista de Símbolos

| | |
|---|---|
| $p$ | Período orbital da Lua |
| $G$ | Constante gravitacional da Terra |
| $m_T$ | Massa da Terra |
| $m_L$ | Massa da Lua |
| $a$ | Distância média entre a Lua e a Terra |
| $X$ | Coordenada retangular no eixo dos X |
| $Y$ | Coordenada retangular no eixo dos Y |
| $Z$ | Coordenada retangular no eixo dos Z |
| θ | Tempo sideral local (HA) |
| ϕ | Latitude do observador (Dec) |
| $A$ | Semieixo maior da elipse orbital |
| $e$ | Excentricidade da órbita |
| $i$ | Inclinação da órbita |
| ω | Argumento do perigeu |
| Ω | Longitude do nó ascendente |
| $M$ | Anomalia Média |
| $v$ | Verdadeira Anomalia |
| $\varepsilon$ | Obliquidade da Eclíptica |
| $E$ | Anomalia Excêntrica |
| $r$ | Distância entre o centro da Terra e objeto celestial |
| $year$ | Variável com o valor de ano (input) |
| $month$ | Variável com o valor de mês (input) |
| $day$ | Variável com o valor de dia (input) |
| $UT$ | Variável com valor unitário da hora (input) |
| *Xeclip* | Coordenada eclíptica no eixo dos X |
| *Yeclip* | Coordenada eclíptica no eixo dos Y |
| *Zeclip* | Coordenada eclíptica no eixo dos Z |
| $LongiM$ | Longitude da Lua no plano da eclíptica |
| $LatM$ | Latitude da Lua no plano da eclíptica |

| | |
|---|---|
| $LongiS$ | Longitude do Sol no plano da eclíptica |
| $LatiS$ | Latitude do Sol no plano da eclíptica |
| $Lm$ | Longitude média da Lua |
| $F$ | Argumento da Latitude da Lua |
| $D$ | Alongamento médio da Lua |
| $Ms$ | Longitude média do Sol |
| $Xgeo$ | Coordenada geocêntrica retangular no eixo dos X |
| $Ygeo$ | Coordenada geocêntrica retangular no eixo dos Y |
| $Zgeo$ | Coordenada geocêntrica retangular no eixo dos Z |
| $Xgeo$ | Coordenada equatorial retangular no eixo dos X |
| $Ygeo$ | Coordenada equatorial retangular no eixo dos Y |
| $Zgeo$ | Coordenada equatorial retangular no eixo dos Z |
| $GMST0$ | Valor de Sideral Time no meridiano de Greenwich às 00:00 |
| $SidTime$ | Sideral Time |
| $LST$ | Local Sideral Time |
| $Longitude$ | Longitude do observador na Terra (valor de Input) |
| $Latitude$ | Latitude do observador na Terra (valor de Input) |
| $Latgeo$ | Latitude do com achatamento da Terra |
| $rgeo$ | Distância ao centro da Terra com achatamento |
| $g$ | Variável auxiliar de cálculo |
| $mpar$ | Moon Parallax |
| $topRA$ | RA Topocêntrico |
| $topDec$ | Dec Topocêntrica |
| $Azimute$ | Valor de Azimute da Lua (e Sol) a partir da Terra |
| $Elevação$ | Valor de Elevação da Lua (e Sol) a partir da Terra |
| $Xhor$ | Coordenada geocêntrica retangular *X* utilizada para rotação |
| $Yhor$ | Coordenada geocêntrica retangular *Y* utilizada para rotação |
| $Zhor$ | Coordenada geocêntrica retangular *Z* utilizada para rotação |
| $Elevacao_{Corrigida}$ | Valor de Elevação da Lua (e Sol) a partir da Terra após correções |

# Capítulo 1

# Introdução

## 1.1 Motivação e Enquadramento

Para se efetuar uma comunicação via ondas rádio que consiga ultrapassar obstáculos naturais como montanhas ou a própria curvatura da Terra, são utilizados satélites de telecomunicações artificiais, que têm como principal função a reflexão do sinal rádio enviado para o espaço de volta à Terra, mas agora para uma outra localização, contornando assim o problema criado pelos obstáculos naturais do nosso planeta.

Os satélites artificiais de telecomunicações acarretam custos muito elevados de manufaturação e de utilização. Esta dissertação pretende dar uma contribuição neste campo, na tentativa de facilitar todos os custos inerentes a uma comunicação efetuada através de um satélite de comunicações artificial, realizando a reflexão do sinal enviado com o auxílio de um recurso natural, e crucial para a vida no nosso planeta - a Lua.

As comunicações via ondas rádio, que utilizam a superfície da Lua como repetidor/refletor passivo do sinal enviado novamente para a Terra, não são nenhuma novidade. Na realidade, o tipo de comunicação *Earth-Moon-Earth* (EME), acabou por ser o ponto de partida para as comunicações de longo alcance que utilizam satélites artificiais como refletor de sinal, sendo que atualmente as comunicações do tipo EME são utilizadas, na sua maioria, pela comunidade académica e rádio amadores.

Devido aos avanços registados a nível do processamento digital de sinal, existe a motivação de se testarem comunicações de mais alto débito, o que só é possível com um valor de frequência portadora superior, face aos valores tipicamente utilizados em comunicações amadoras EME situados na banda *Very High Frequency* (VHF) e *Ultra High Frequency* (UHF) [1]. Este valor mais elevado de frequência da portadora, leva a que se consigam suportar

larguras de bandas superiores, tendo como consequência final um aumento no volume de informação partilhada.

Esta futura ligação deverá ser realizada através de um feixe de micro-ondas na banda *Super High Frequency* (SHF) com um valor de frequência na ordem dos 10 GHz. Este valor de frequência encontra-se muito acima dos valores tipicamente utilizados nas comunicações rádio amadoras de longo alcance do tipo EME [1].

Este tipo de ligação será útil a pessoas/entidades que queiram efetuar uma comunicação de custo reduzido sem fios, entre países muito distantes ou localizações remotas, quer seja através de uma ligação do tipo áudio ou mesmo de vídeo (caso se verifiquem débitos binários suficientes) em tempo real, ou até mesmo para a troca de dados interbancários, comerciais, sociais, científicos, académicos, etc.

À primeira vista este tipo de comunicação EME pode parecer bastante similar às comunicações de longo alcance efetuadas através de um satélite artificial. Porém, existem grandes diferenças entre ambas. Um satélite de telecomunicações contém uma antena emissora e recetora, e na superfície da Lua não existem antenas. Este fator leva a que se verifique uma enorme dispersão do sinal aquando da reflexão deste na sua superfície acidentada [2]. O facto de a Lua ter uma órbita em torno da Terra com um período de 27.3 dias [2], torna o problema ainda mais acentuado, face aos satélites artificiais geoestacionários que permanecem fixos face a uma dada localização na Terra.

Apesar das diferenças verificadas, o princípio básico é o mesmo em ambas as formas de comunicação via satélite (artificial ou natural), sendo este princípio a reflexão do sinal enviado da Terra novamente para esta. O grande problema é que a atenuação do sinal, devida à atenuação em espaço livre e à dispersão na Lua, é muito grande, mas não inviabiliza a comunicação como se provou na época que antecedeu a existência dos satélites artificiais. Deste modo, é possível enquadrar de forma natural esta dissertação no ramo das telecomunicações, mais precisamente em telecomunicações de longo alcance sem fios utilizando ondas rádio.

Esta dissertação tem como finalidade dar início ao estudo das comunicações via ondas rádio de longo alcance EME, bem como dos elementos orbitais que caracterizam a órbita da Lua. O resultado alcançado do estudo sobre a órbita lunar, será aplicado através da criação de

um *software* desenvolvido neste trabalho e designado por ItTracker, que é capaz de calcular a posição da Lua (e Sol) para uma determinada localização geográfica na Terra, num espaço de tempo pré-definido ou em tempo real. Deste modo, este trabalho contribui para que no futuro seja possível aperfeiçoar esta solução para efetuar uma ligação via ondas rádio utilizando a Lua como repetidor/refletor passivo do sinal enviado da Terra.

O sucesso do contexto no qual esta dissertação se insere, pretende abrir um novo leque de possibilidades (a entidades independentes e com recursos monetários mais reduzidos), nas ligações sem fios de longo curso, cujo monopólio se encontra muito centralizado e restrito.

## 1.2 Objetivos

Efetuar um estudo sobre as comunicações rádio de longo alcance utilizando a Lua como repetidor passivo (comunicações EME), realçando numa primeira fase, os aspetos que são considerados os mais importantes a se ter em conta.

Realizar um estudo aos elementos orbitais que caracterizam a órbita lunar. Posteriormente será aplicado o conhecimento adquirido através da criação de um *software* em linguagem Java, capaz de conseguir efetuar o cálculo final do Azimute e Elevação para a Lua (e Sol), a partir de uma localização geográfica na Terra, bem como o cálculo e demostração das variáveis intermédias mais relevantes, como Distance, Right Ascension (RA), Declination (Dec), etc.

O *software* será capaz de efetuar o cálculo com o grau de precisão (margem de erro) necessário e definido para esta dissertação (capacidade para efetuar o cálculo do passado, presente e até à data 31/12/2099). O *software* também será capaz de efetuar o cálculo em tempo real de todas as variáveis envolvidas autonomamente, até que seja dada ordem de interrupção por parte do utilizador. O resultado das variáveis finais de Azimute e Elevação da Lua observadas num ponto específico da Terra, será gravado num ficheiro (do tipo *.dat), para que no futuro seja possível controlar a orientação de uma antena parabólica através do output gerado pelo *software*.

## 1.3 Organização

O primeiro capítulo foca-se, numa fase introdutória, sobre o enquadramento, motivação e objetivos desta dissertação.

O capítulo 2 encontra-se dividido em duas partes distintas. Na primeira parte pode ser visto um pouco da história das comunicações via ondas rádio de longo alcance e implicações das comunicações EME. Na segunda parte, são descritos os conceitos orbitais mais importantes para o cálculo da órbita lunar a partir da Terra.

Na primeira parte do capítulo 3, são descritos os elementos orbitais utilizados e métodos de cálculo que foram realizados para o correto funcionamento do *software*. Numa segunda parte, são descritos os valores utilizados para as variáveis definidas no cálculo lunar, bem como todas as fórmulas matemáticas utilizadas e processo de cálculo sequencial utilizado pelo programa. Métodos de conversão e normalizações de valores matemáticos, também poderão ser observados neste capítulo.

No capítulo 4, numa primeira parte é feita a descrição de como se encontra organizado o ItTracker nas suas classes em Java. Na segunda parte deste capítulo, é descrita toda a componente visual do programa, aliado aos métodos e valores de input que terão que ser introduzidos para o correto funcionamento do *software*. Seguidamente também é descrito todo o output gerado pelo ItTracker. Na última parte deste capítulo, pode ser visto a comprovação dos resultados propostos e alcançados, através da comparação dos resultados obtidos com um outro *software* criado para o efeito de cálculo da órbita lunar a partir da Terra.

Por fim, no capítulo 5 são descritas as conclusões finais acerca da dissertação, bem como do trabalho futuro que poderá ser realizado.

# Capítulo 2

# Estado da Arte

Neste capítulo é possível ver a pesquisa e consulta efetuada para a realização dos objetivos propostos nesta dissertação. Este capítulo encontra-se dividido em duas partes distintas. Na parte 2.1 pode ser vista uma breve história das comunicações via ondas rádio de longo alcance bem como dos aspetos que deverão se ter em conta para a realização de uma comunicação EME. A parte 2.2 deste capítulo foca-se nos conceitos e fundamentos orbitais que são a base para a compreensão e realização do cálculo orbital da Lua do ItTracker.

## 2.1 Comunicações via ondas rádio de longo alcance

As comunicações de longo alcance de onda curta apareceram pela primeira vez em 1924 pela mão de Guglielmo Marconi [3], o também inventor das comunicações em onda longa, que até então e durante 30 anos, foram utilizadas para estabelecer uma comunicação via rádio frequência de longo alcance. O facto de a experiência ter sido um sucesso, aliado ao facto do custo de fabrico do equipamento ser muito mais acessível, bem como o seu consumo elétrico ser muito menor, levou a que a partir de 1926 passa-se a adotar globalmente a onda curta para telecomunicações de longo alcance [3].

A revolução da comunicação em onda curta levou a que em 1928 [4] fosse tentado, sem sucesso, obter um eco da Lua. A 10 de Janeiro de 1946 através do projeto da *Army Signal Corp's* do exército Norte-Americano *Project Diana*, foi registado com sucesso a primeira reflexão de um eco de sinal da Lua emitido através de uma antena de radar situada na Marconi Road, Wall, New Jersey [5]. O sucesso deste projeto teve como consequência a recomendação por parte do Dr. Donald Menzel da Harvard *College Observatory* deste tipo de comunicação ao departamento de investigação da marinha Norte-Americana [4].

O acentuado interesse por parte do exército Norte-Americano nas potencialidades que este tipo de comunicação poderia trazer em termos de segurança nacional [4], bem como os

problemas trazidos pelas tempestades na ionosfera de Terra, que tornavam impossível efetuar transmissões rádio de longo alcance entre os seus navios de porta-aviões a partir do reflexo do sinal na ionosfera da Terra, levaram a que no dia 21 de Outubro de 1951 [4], fosse possível ao engenheiro James H. Trexler, efetuar o primeiro teste com sucesso de transmissão de um pequeno impulso de 198Mhz através de radar, efetuado com uma potência de transmissão de 750 Watts.

Os resultados surpreenderam o próprio devido à fidelidade do eco do impulso recebido a partir da Lua, dando assim origem ao projeto *Communication Moon Relay* [3], também conhecido por *Operation Moon Bounce* por parte da marinha dos Estados Unidos.

Este sucesso levou a que testes [3] seguintes fossem efetuados através do envio de sinal em *Continuous Wave* (CW), e *Modulated Continuous Wave* (MCW), e sinal modulado em áudio frequência de modo a comprovar a possibilidade de se efetuar uma comunicação áudio com base na reflexão na Lua.

24 de julho de 1954, marca a data da primeira voz humana [3] (a voz de James H. Trexler) a ultrapassar a barreira da ionosfera e a regressar dois segundos e meio depois de volta à Terra após esta ter sido refletida na Lua.

Apesar de já estar operacional em 1959, em Janeiro de 1960 foi inaugurada [3] a completamente operacional ligação entre Washington,D.C. e Hawaii, utilizando antenas motorizadas com transmissores de 100 Kilowatts à frequência de 400 MHz, com a capacidade de acomodar 16 canais *teleprinter* (teletexto) a operar a um rácio de 60 palavras por minuto e com a capacidade de processar fotografias em *facsimile* (telecópia) [3].

A complexidade envolvida e potências necessárias para transmissão, aliado ao aparecimento dos primeiros satélites artificiais de comunicação, levou a que este tipo de comunicação entrasse em desuso ao longo da década de 60 [3]. Contudo, convém realçar que se não fosse todo o estudo feito para as comunicações EME, provavelmente os satélites artificiais de telecomunicações não teriam surgido tão cedo.

### 2.1.1 Comunicações EME e rádio amadores

O interesse de rádio amadores pela tentativa de comunicação com reflexão na Lua começou desde cedo. Em 1953 foi registada [6] a primeira tentativa com sucesso de um rádio amador na obtenção de um eco através de um sinal enviado para a Lua.

Na madrugada do dia 17 de julho de 1960 [7] foi efetuada com sucesso a primeira transmissão transcontinental de duas vias por parte do grupo *Rhododendron Swamp* VHF *Society* (W1BU) em Medfield, Massachusetts e o *Eimac Gang Radio Club* (W6HB) em São Carlos da Califórnia. Este dia constitui um marco na história dos rádio amadores.

Entre 1959 e 1964 um grupo de estudantes da St. Joe's High School [8] tentaram sem êxito efetuar uma comunicação através da Lua. Este projeto académico teve o financiamento da *The Teletype Corporation* em Chicago, e consistia na implementação de raiz de todo o projeto, bem como na construção uma antena Yagi-Uda com uma dimensão de proporções "épicas".

A partir da década de 60 e até aos dias de hoje, muito do trabalho envolvido e inovação nesta área foi proposto e testado por rádio amadores. Estes registos de inovação foram trazidos em 1970 através da utilização de semicondutores de baixo ruido, bem como o melhoramento no *design* de antenas Yagi-Uda. Em 1980 ocorre a introdução dos pré-amplificadores e de antenas desenhadas com o auxílio de computadores.

Os avanços a nível do processamento de sinal digital já na década de 2000, veio trazer novas possibilidades a estas comunicações, o que fez renascer o interesse pelas comunicações EME na última década por parte da comunidade académica e rádio amadora, havendo mesmo registos de ecos efetuados com sucesso a partir de frequências tão elevada como os 120 GHz.

### 2.1.2 Implicações de uma comunicação EME a 10 GHz

As comunicações de longo alcance sem fios são possíveis nos dias que correm, devido ao recurso a satélites artificiais posicionados em órbitas geostacionárias, que se encontram a distâncias mais elevadas face à superfície da Terra, face a outro tipo de satélites que não sejam utilizados para telecomunicações via ondas rádio (ex. satélites meteorológicos).

A distância é um fator que terá que ser tido em conta. As ondas rádio propagam-se à velocidade da luz - acerca de $3x10^8$ m/s (c). A grande distância da Lua face à Terra, sendo esta a maior distância já efetuada para uma reflexão de sinal rádio a partir da ionosfera, leva a que o tempo de propagação de uma onda entre a Terra e a Lua, e de volta à Terra, possa ser visto como 2 x distância\c. Deste modo, obtêm-se valores de tempo de propagação de 2.4s em fase de perigeu e de 2,7 em fase de apogeu, e uma média de tempo de propagação de sinal na ordem dos 2,56s. O fator de distância acentuada é importante, e um aspeto sempre a ter em conta para comunicações via ondas rádio de longo alcance, visto que só com a distância elevada é que se conseguem obter longos alcances de reflexão do sinal. Porém, o facto de a distância ser bastante elevada no caso da Lua, leva a que também se verifique um aumento muito considerável da atenuação em espaço livre - cerca de 289.2 dB para uma frequência de 10 GHz [2]. Isto não é de todo desejável, visto que se irá verificar uma deterioração muito acentuada na relação sinal-ruído final.

As frequências mais utilizadas por parte dos rádio amadores na realização de comunicações EME tem recaído, ao longo dos anos, sobretudo em frequências na ordem dos 50, 144, 432, e 1296 MHz, nas bandas VHF e UHF [2] de modo a compensar as grandes perdas de sinal associadas às comunicações EME. Porém, para os valores de frequências utilizadas atualmente, só é possível obter-se valores de débito binários baixos ou médios, devido ao valor baixo da frequência da própria portadora. Estas frequências mais utilizadas atualmente, implicam também o recurso a uma potência de emissão na ordem da centena de watts (W), e que poderá ir até 1200 W [2]. Para o caso de Portugal, o valor máximo de potência a utilizar permitido para rádio amadores é na ordem dos 300W [9].

Uma das grandes barreiras à utilização de comunicações EME é de facto a elevada atenuação devido às perdas de propagação. Para frequências vez cada vez mais elevadas verifica-se um aumento cada vez mais acentuado nas perdas registadas. Porém a potência de transmissão necessária para a transmissão diminui com o aumento da frequência [2]. Outro fator que advém do aumento da frequência é a necessidade de se utilizar antenas parabólicas, em vez de antenas Yagi-Uda para valores de frequências acima dos 1296 MHz [2], como pode ser visto na Tabela seguinte 1.

Para a realização de uma comunicação futura utilizando a Lua como repetidor passivo de sinal, a escolha inicial recaiu sobre a utilização de uma frequência de micro-ondas SHF a 10

GHz. A escolha desta frequência não é ao acaso, visto que esta é uma frequência que se encontra no limite superior da gama ótima aconselhável à realização de uma comunicação EME [2], sendo que a partir deste valor de frequência começam-se a registar significantes componentes de reflexão difusa do sinal na superfície Lunar [2].

| Frequência (MHz) | Perdas médias de propagação (dB) | Tipo de Antena | G (dBi) | TxPwr (W) |
|---|---|---|---|---|
| 50 | 242.9 | 4x12m (Y) | 19.7 | 1200 |
| 144 | 252.1 | 4x6m (Y) | 21.0 | 500 |
| 432 | 261.6 | 4x6m (Y) | 25.0 | 250 |
| 1296 | 271.2 | 3m (P) | 29.5 | 160 |
| 2304 | 276.2 | 3m (P) | 34.5 | 60 |
| 3456 | 279.7 | 2m (P) | 34.8 | 120 |
| 5760 | 284.1 | 2m (P) | 39.2 | 60 |
| 10368 | 289.2 | 2m (P) | 44.3 | 25 |

**Tabela 0.1 – Valores típicos de perdas de propagação. Valores de dimensão e tipo de antenas, G, e TxPwr [2]**

Na tabela 1 pode ser visto que de facto a 10 GHz, as perdas médias de propagação aumentam, mas como já referido anteriormente, a potência de transmissão também diminui consideravelmente com o aumento da frequência. Um fator positivo que leva a que se efetuem novos testes a uma frequência de 10GHz para ligações futuras do tipo EME, é o facto de já existirem registos de testes realizados com sucesso a 10 GHz utilizando uma antena parabólica de 62cm e com uma potência de emissão de apenas 8W [10].

Outros aspetos a ter em conta são os do efeito de Doppler [1] (*Doppler Shift*) devido ao movimento constante da Terra e da Lua, bem como do efeito de *libration fading* [2], que consiste no atraso da receção das ondas rádio refletidas devido ao facto da superfície lunar ser bastante acidentada e provocar reflexões em diferentes fases.

O problema de *Faraday Rotation* [2], não é tido em conta visto que este só ocorre para frequências inferiores a 1296MHz [1], contudo, será necessário ter em atenção a atenuação provocada pela chuva na troposfera para uma frequência acima dos 5Ghz [2].

Outro fator a considerar é mudança de polarização de sinal [1] e as perturbações na órbita Lunar [2], sendo que outro fator de grande importância e bastante limitativo a se ter em conta, é o do ruído térmico produzido por variadíssimas fontes [2]. A própria Terra, a sua

atmosfera, a superfície da Lua, o cosmos, o Sol, bem como de outras fontes, são aspetos que têm que se ter em conta.

| Frequência (MHz) | CMB (K) | Atm (K) | Moon (K) | Gal (K) | Side (K) | $T_a$ (K) | $T_r$ (K) | $T_s$ (K) |
|---|---|---|---|---|---|---|---|---|
| 50 | 3 | 0 | 0 | 2400 | 1100 | 3300 | 50 | 3500 |
| 144 | 3 | 0 | 0 | 160 | 160 | 260 | 50 | 310 |
| 432 | 3 | 0 | 0 | 9 | 33 | 45 | 40 | 85 |
| 1296 | 3 | 0 | 0 | 0 | 30 | 35 | 35 | 70 |
| 2304 | 3 | 0 | 4 | 0 | 30 | 37 | 40 | 77 |
| 3456 | 3 | 1 | 5 | 0 | 30 | 40 | 50 | 90 |
| 5760 | 3 | 3 | 13 | 0 | 30 | 50 | 60 | 110 |
| 10368 | 3 | 10 | 42 | 0 | 30 | 85 | 75 | 160 |

**Tabela 0.2 – Valores de ruído térmico provocados por diversas fontes associados a uma frequência de utilização. [2]**

Na Tabela 2 podem ser vistos os valores das temperaturas de ruído que devem ser considerados para várias frequências de utilização associados a uma comunicação via ondas rádio do tipo EME.

## 2.2 Estudo Orbital

Nesta secção, podem ser vistos os conceitos considerados fundamentais acerca do estudo orbital efetuado para se conseguir calcular a posição da Lua, para uma localização na Terra. Este estudo foi a base do sucesso para o desenvolvimento do programa ItTracker.

### 2.2.1 Leis de Kepler

Para se alcançar o objetivo de se conseguir calcular o Azimute e Elevação da Lua face a uma posição na Terra, é necessário primeiro compreender as Leis de Kepler [11]. Estas leis são a base de qualquer cálculo orbital.

A **primeira lei de Kepler** define que: "As órbitas dos planetas descrevem uma elipse com o Sol a ocupar um dos focos (1602)" [11]. Esta lei definiu que as órbitas não eram círculos perfeitos, mas sim cónicas (círculos, elipses). Sendo que aquilo que distingue o tipo de órbitas é a sua excentricidade.

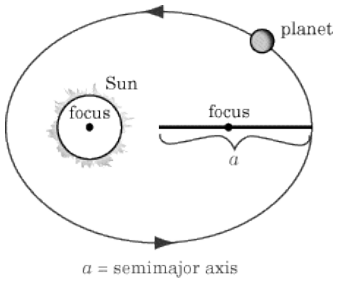


Figura 2.1: 1ª Lei de Kepler [12]

A **segunda lei de Kepler** define que: "O vetor que liga um planeta ao Sol, varre áreas iguais em tempos iguais (1605)." [11]. Esta é também conhecida como a lei das áreas, e determina que os planetas se movem com velocidades diferentes, dependendo da distância a que se encontram do astro situado num dos focos da elipse. Caso do Sol e de um planeta do sistema solar, como pode ser visto na figura 2.2.

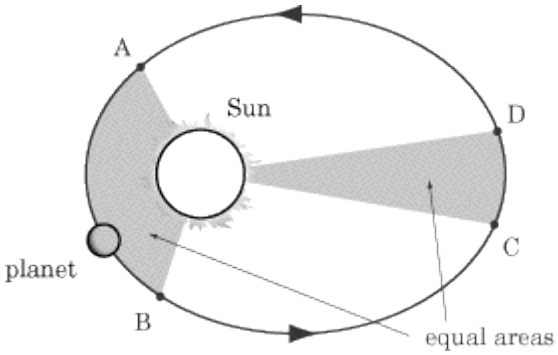


Figura 2.2: 2ª Lei de Kepler (lei das áreas) [12]

A **terceira lei de Kepler** define que: "O quadrado do período orbital de um planeta, é proporcional ao cubo do semieixo maior da órbita do planeta (1618)" [11]. Ou seja, se considerarmos um planeta em torno do Sol, esta lei define que quanto maior for a distância de um planeta ao Sol, maior vai ser o tempo que este planeta leva a completar uma volta em torno deste.

As leis de Kepler foram definidas tendo o Sol como ponto central, contudo, estas podem ser generalizadas para um sistema de coordenadas geocêntrico em que a Terra, ou qualquer outro objeto celeste é o ponto central. Como é o caso da Terra e da Lua que órbita à sua volta.

Antes de se efetuar cálculos acerca das coordenadas da Lua face a uma posição na Terra, é será efetuado o cálculo preciso do período da órbita Lunar (tempo que a Lua leva a dar uma volta completa sobre a órbita da Terra). Assim, aplica-se a versão de Newton sobre a formula resultante da 3ª lei de Kepler, de modo a ser possível efetuar este cálculo com exatidão.

$$p^2 = \frac{4\pi^2}{G.(m_T + m_L)} * a^3 \tag{2.1}$$

Para os valores de distância média entre a Lua e a Terra de $a = 3.844 * 10^8\ m$, e da massa da Terra $m_T = 5.974 * 10^{24}\ Kg$, da massa da Lua $m_L = 7.35 * 10^{22}\ Kg$ e da constante gravitacional $G = 6.67 * 10^{-11} N.m^3/Kg^2$ consegue-se finalmente obter que o período da Lua é:

$$\mathrm{p} = 2.36 * 10^6\ \mathrm{s} = 2.36 * 10^6\ \mathrm{s} * \frac{1\mathrm{Hora}}{3600} * \frac{1\mathrm{dia}}{24\mathrm{horas}} = 27.3\ \mathrm{Dias} \tag{2.2}$$

Através do auxílio da 3ª lei de Kepler, é agora possível afirmar com toda a exatidão que a Lua tem um período orbital de **27.3** dias em redor do planeta Terra.

## 2.2.2 Periélio/Perigeu e Afélio/Apogeu

A órbita da Terra em redor do Sol não é perfeitamente circular, mas sim elíptica, embora tendo um valor de excentricidade bastante baixo. A posição quando um planeta se encontra mais próximo do Sol, dá pelo nome de Periélio [11], e por sua vez, a posição em que um planeta se encontra mais afastado do Sol tem como nome Afélio [11]. Na figura 2.3 pode ser vista a posição de um planeta em ponto de periélio e afélio em torno do Sol para melhor compreensão.

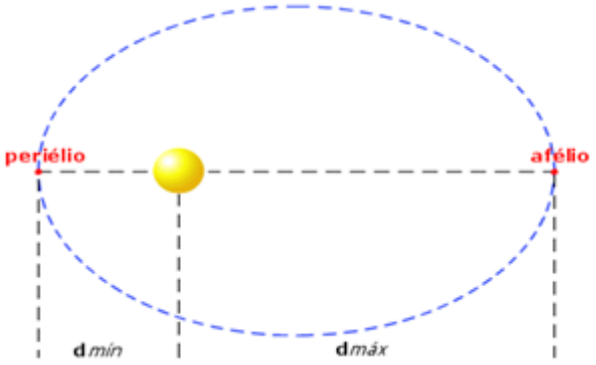

Figura 2.3: Posição de Periélio e Afélio [13]

É também de notar que é na posição de periélio que a velocidade linear de um planeta é máxima, e na posição afélio que a sua velocidade linear é mínima. Em órbitas Parabólicas só existe Periélio.

A orbita da Lua em torno da Terra, tal como a orbita da Terra em redor do Sol, é também uma orbita elíptica. Semelhante ao já observado para os conceitos de afélio e periélio, o nome dado para um objeto que gire em torno da Terra numa orbita elíptica e que se encontre na sua posição mais próxima da Terra, dá-se o nome de Perigeu [11]. Por sua vez, o Apogeu refere-se ao ponto em que o objeto se encontra mais afastado da Terra [11].

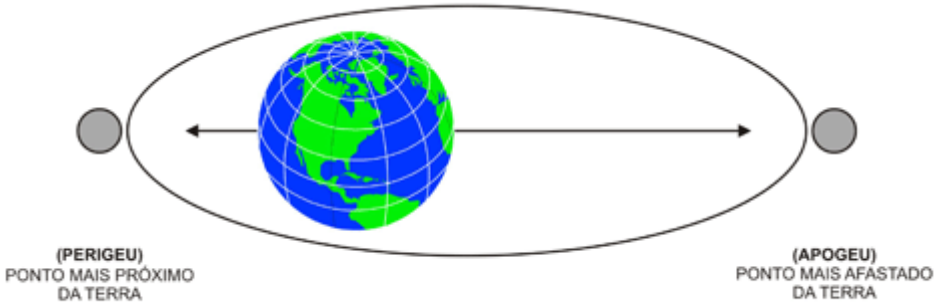


Figura 2.4: Perigeu e Apogeu da Lua [14]

É também de notar que o próprio ponto de perigeu da Lua anda para a "frente" numa revolução em cada 8,8 anos [15].

## 2.2.3 Eclíptica da Terra

A Terra encontra-se animada de um movimento de rotação em torno do seu eixo e de um movimento de translação em volta do Sol. O plano em que o centro da Terra se move no seu movimento anual em torno do Sol, é denominado de plano da eclíptica [15].

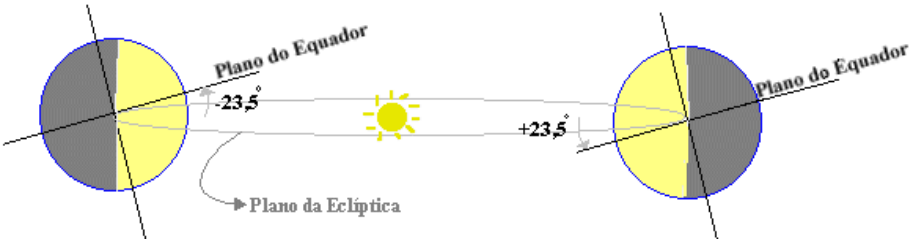


Figura 2.5: Obliquidade da Eclíptica da Terra [15]

O ângulo de inclinação da Terra, no plano da eclíptica dá-se pelo nome de obliquidade da eclíptica. A obliquidade da eclíptica mede o ângulo que a eclíptica faz com o plano do Equador, este ângulo tem um valor de 23,5º (valor arredondado às décimas) [15].

### 2.2.4 Eclíptica da Lua

Semelhante ao plano da Terra em relação ao Sol, a Lua também é caracterizada da mesma forma no seu plano de órbita face à Terra. O plano orbital da Lua em torno do plano da eclíptica da Terra tem uma inclinação de 5,14º [16]. Uma visualização dos diferentes planos pode ser vista na figura seguinte.


Figura 2.6: Inclinação da Lua sobre a eclíptica da Terra [16]

Apesar do ângulo de inclinação do plano da órbita em relação à eclíptica permanecer aproximadamente constante, este plano orbital não é fixo. Este move-se de tal maneira, que seu eixo descreve um círculo completo em torno do eixo da eclíptica num período de 18,6 anos [16]. Esta rotação para Oeste do plano orbital da Lua ocorre devido à força diferencial exercida pelo bojo equatorial da Terra, que é causado pela rotação da Terra.

### 2.2.5 Posição Heliocêntrica, Geocêntrica e Topocêntrica

Quando se está a efetuar um estudo sobre cálculos orbitais (e por consequente os próprios cálculos posteriormente), é necessário entender que para diferentes posições de ponto de observação, existem diferentes tipos de coordenadas que terão que ser utilizadas para o cálculo de diferentes pontos de observação. Deste modo, podem ser vistos de seguida 3 das posições utilizadas nesta dissertação para o cálculo da órbita lunar:

- A posição **Heliocêntrica** refere-se a um sistema de medida de posição que têm como referência e ponto **central o Sol** [11]. Um objecto (como a Terra) que gire em torno do Sol, têm como referência uma posição heliocêntrica.

- As posições **Geocêntricas**, tal como a heliocêntrica, têm como referência um ponto central. No caso da geocêntrica, e para esta dissertação, este ponto central é o **centro da Terra** [11]. Objectos como satélites artificias e a Lua, que giram em torno da Terra, têm como referência uma posição geocêntrica.

- Um sistema de posição **Topocêntrica,** tem como referência a posição de **um observador que se encontre na superfície da Terra** [11]. A diferença de valores de posição para um objeto no espaço, utilizando um sistema geocêntrico ou topocêntrico pode ser mínima conforme a distância do objeto, ou relevante (quanto mais distante, menos relevante).

Na figura 2.8 (abaixo ilustrada), podem ser vistas as coordenadas que servem para descrever a posição de um objeto no espaço utilizando um sistema de posição Topocêntrico.

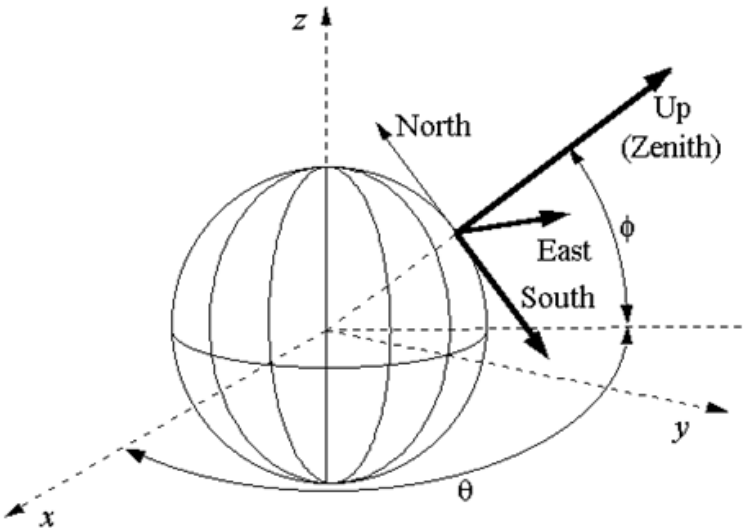

Figura 2.7: Posição Topocêntrica e eixo de coordenadas [17]

O sistema de eixo de coordenadas retangular é caracterizado nos eixos dos *X*, *Y*, *Z*. Para se efetuar a transformação do eixo de coordenadas cartesiano para um sistema de valores topocêntricos, é necessário efetuar a rotação por um angulo θ (tempo sideral local ou *Hour Angle*) sobre o eixo cartesiano dos Z. E depois a rotação de um angulo ϕ (latitude do observador ou *Declination*) sobre o eixo cartesiano dos X.

Para objetos que se encontrem muito próximos da Terra, como a Lua e satélites artificiais, é preciso ter em conta a diferença de distância de um objeto, pois os valores podem divergir num erro de quase 2º [17].

### 2.2.6 Coordenadas Equatoriais (*Hour Angle*, *Right Ascenscion* e *Declination*)

Dado só existir um plano equatorial no planeta Terra, quando se pretende representar um ponto de um observador que se encontra localizado no nosso planeta, utiliza-se este plano visto ser independente da superfície terrestre. Deste modo, o sistema de coordenadas equatoriais é um sistema de coordenadas que tem como plano fundamental o equador celeste (conhecido também como a linha do equador da Terra que separa o hemisfério norte do hemisfério sul), e que tem como finalidade, a representação da localização de corpos celestes a partir de um ponto na Terra [11].

O sistema equatorial local, que depende da localização do observador, tem como suas coordenadas o *Hour Angle* (HA) e Dec. O sistema equatorial universal tem como suas coordenadas a RA e Dec. De seguida, pode ser vista a descrição destas 3 coordenadas:

- **HA**, é a distância angular medida entre o meridiano de um objeto celestial e o meridiano de um observador na Terra ao longo do equador celeste [18]. Este valor é medido em horas, minutos, segundos de tempo sendo que uma revolução completa equivale a 24 horas, dado que a Terra efetua uma rotação completa de 360º em 24 Horas [18]. A contagem de tempo para a coordenada **HA** é feita no sentido dos ponteiros do relógio ao longo do equador celestial [18].

- **Dec** (**δ**) é o valor medido do equador celeste até ao objeto que se está a efetuar o cálculo. O valor é expresso em graus (º), minutos (´) segundos (´´), sendo positivo se estivermos a calcular algo a norte do equador celeste, e negativo se o objeto estiver a sul [11]. O seu valor varia entre +90º (valor do polo celeste norte), -90º (valor do polo celeste sul) e 0º na linha do equador celeste [18].

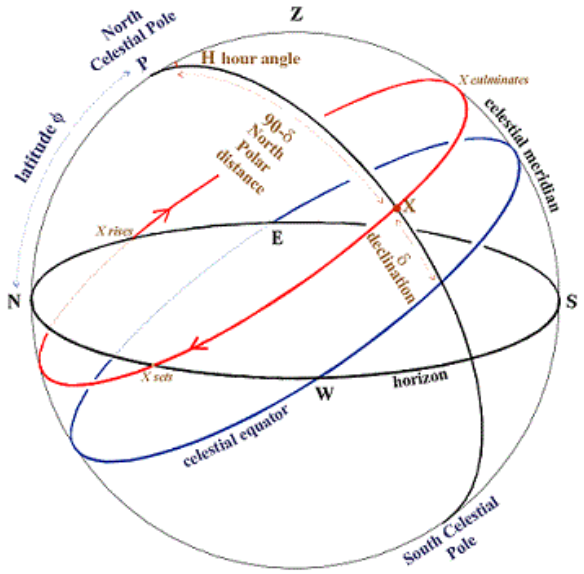


Figura 2.8: *Hour Angle* (HA) e *Declination* (Dec) [18]

- **RA** (α) é uma coordenada do sistema equatorial, similar à HA**.** É medida ao longo do equador celeste e no ponto vernal, posição do Sol na altura do equinócio vernal, o seu valor é igual a zero. A maneira mais usual de expressar o valor de **RA** é também em horas, minutos, segundos de tempo, sendo que uma revolução completa também equivale a 24 horas (360º) [19]. Para **RA**, a contagem de tempo é efetuada no sentido contrário ao dos ponteiros do relógio ao longo equador celestial [19]. É de realçar que os polos celestes não têm valores de **RA** definidos.
-

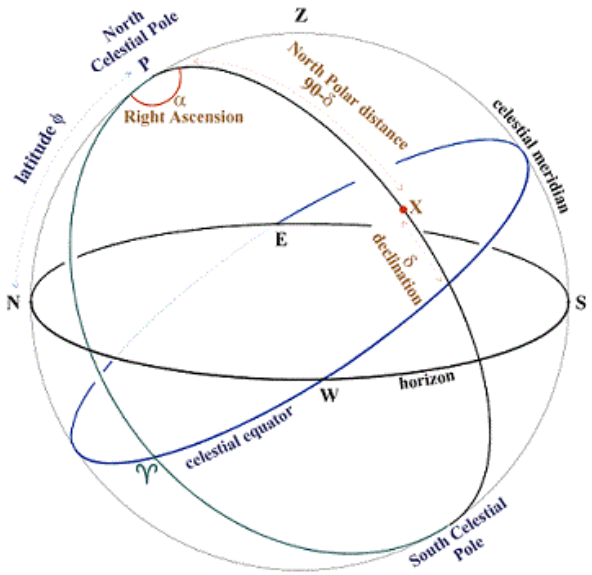


Figura 2.9: *Right Ascension* e *Declination* [19]

### 2.2.7 Longitude e Latitude

A **Latitude** na Terra, é a distância a que o ponto de referência se encontra do Equador medida ao longo do meridiano de Greenwich [11]. Esta distância mede-se em graus, e varia entre 0º a 90º para Norte (Hemisfério Norte) e 0º a -90º para Sul (Hemisfério Sul) [11].

A **Longitude** na Terra é a distância a que o ponto de referência se encontra do meridiano de Greenwich ao longo do Equador [11]. A distância é também medida em graus, e o seu valor varia entre 0º e 180º, para Este (valor positivo) ou Oeste (valor negativo) do meridiano [11].

É necessário realçar que os valores de **Latitude** e **Longitude** podem variar conforme o sistema de coordenadas (ou plano) em que se está a efetuar o cálculo orbital. Por exemplo, para um sistema de coordenadas eclípticas referentes ao Sol e Lua, no plano orbital da eclíptica, estes valores são medidos em graus, e com valor de 0º no ponto Vernal. A **Latitude** é positiva a norte da eclíptica e negativa para sul. A **Longitude** por sua vez, varia entre 0º e 360º (revolução completa), e é medida no sentido contrário ao dos ponteiros do relógio ao longo da eclíptica.

### 2.2.8 Azimute e Elevação

O valor de **Azimute** calculado nesta dissertação, é o valor em graus contado a partir do Norte, no sentido dos ponteiros dos relógios, e que aponta para um ponto no horizonte. Os valores de **Azimute** variam entre 0º e 360º. O requisito básico para se conseguir localizar um determinado **Azimute** é saber onde fica o Norte. Existem diferentes maneiras de representar os valores de **Azimutes**. Estas são as seguintes:

- **Azimute Magnético**: medido a partir do Norte Magnético.
- **Azimute Geográfico**: medido a partir do Norte Geográfico.
- **Azimute Cartográfico**: medido a partir do Norte Cartográfico.

Para esta dissertação foi escolhido o **Azimute Geográfico** como referência. Esta será a referência que servirá para localizar a Lua ( ou Sol) ao longo de um círculo horizontal que vai

do norte geográfico até ao ponto em que este se encontra. Um exemplo de o azimute referente à Lua (ou Sol) é localizado, utilizando uma antena parabólica, pode ser visto na figura 2.11.

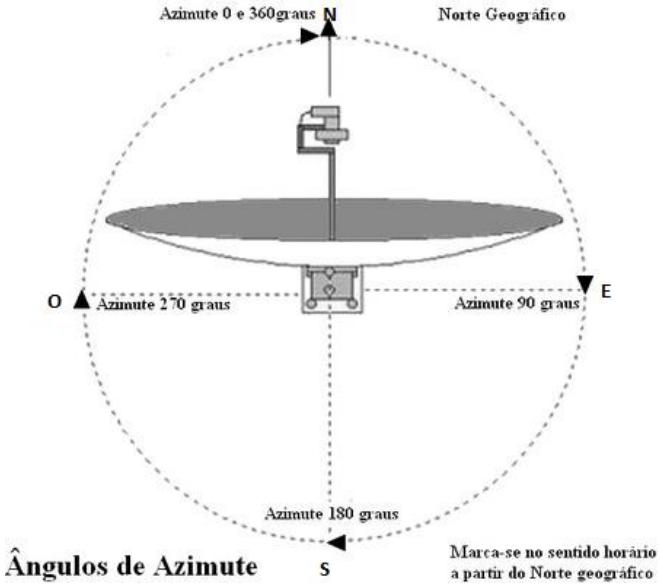

Figura 2.10: Azimute Cartográfico [20]

A **Elevação** de um objeto no espaço é também medida em graus e é nada mais do que a altura a que a Lua (ou Sol) se encontram na linha do horizonte [21]. Os valores do ângulo de elevação variam entre 0º e -90º quando o objeto se encontra abaixo da linha do horizonte (Lua ou Sol não se encontram visíveis) e entre 0º e 90º quando estes se encontram acima da linha do horizonte [21], ou seja, a Lua (ou Sol) encontram-se visíveis para a localização em que se encontra o observador (ou ponto de cálculo).

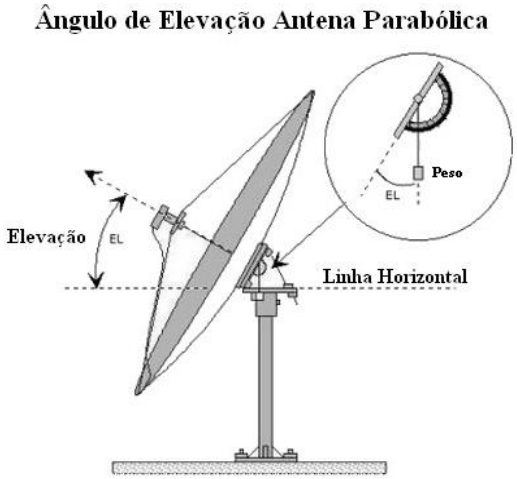

Figura 2.11: Ilustração de Elevação numa Parabólica [20]

Na figura 2.11, é possível ver a linha horizontal, referente à linha do horizonte, e o ângulo de elevação para uma antena parabólica. Esta figura, ilustra o conceito de elevação que uma antena deverá a ter para localizar um objeto que se encontre no espaço.

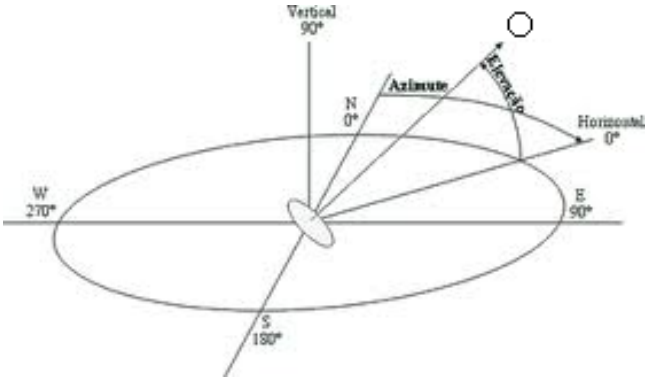

Figura 2.12: Conjunto Azimute e Elevação [20]

Na figura 2.12, observa-se uma visão geral da conjunção de valores de Azimute e Elevação. Este é o resultado final dos valores que irão permitir efetuar a localização da Lua, a partir da Terra, utilizando o programa ItTracker desenvolvido para esta dissertação.

### 2.2.9 Julian Day e Modified Julian Day

O *Julian Day* (JD) é uma contagem do número de dias contínuos com início no dia 1° de janeiro de 4713 a.C. [22]. O dia é caracterizado como tendo o seu inicio ao meio-dia e dura até ao meio-dia seguinte. Deste modo, todo o período noturno fica convenientemente inserido em um único dia. Este método de contagem de dias é fundamental para o cálculo de posição de objetos no espaço. Por exemplo, para o dia 20 de julho de 1999, cujo JD é o número 2.451.380, isto significa que se passaram 2.451.380 dias desde o dia 1 de janeiro de 4713 a.C.

O *Modified Julian Date* **(**MJD**)**, considera que os dias julianos, entre 1859 e 2130 [22], começam sempre com "24". Assim, em 1975 foi adotada a convenção do MJD, definida como:

$$MJD = JD - 2451543.5 \qquad (2.3)$$

O MJD de valor igual a 0 corresponde ao JD 2451543.5 (12 horas após o meio-dia de 16 de novembro de 1858 UTC). Ou seja, o MJD = 0 designa a meia-noite entre 16 e 17 de novembro de 1858. Pode-se inserir quantas casas decimais após a vírgula se achar necessário para representar a fração do dia com a precisão desejada.

# Capítulo 3

# Cálculo Orbital

Este capítulo é a extensão do estudo orbital apresentado na segunda parte do capítulo anterior. O capítulo encontra-se dividido em duas partes distintas. Na primeira parte, são abordadas as fórmulas matemáticas dos elementos orbitais utilizados para o cálculo da posição da Lua (e Sol). Na segunda parte, é feita a descrição dos algoritmos matemáticos utilizados na realização do ItTracker para o cálculo orbital, bem como as considerações que foram tidas em conta, para se conseguir obter o resultado final do valor de azimute e elevação da Lua (e Sol) para uma localização específica na Terra.

## 3.1 Elementos Orbitais

Após a perceção dos conceitos orbitais mais relevantes para se conseguir efetuar o cálculo orbital de um objeto no espaço, contexto principal no qual se encontra inserida esta dissertação, será efetuada de seguida, a descrição daqueles que são considerados os **6 elementos orbitais primários** [22]**.** Estes elementos definem a órbita de um corpo celeste no espaço e são usualmente denominados de **Elementos Orbitais Keplerianos** [22]**.**

Os primeiros dois elementos descrevem a **forma da órbita** num sistema de coordenadas em 2D (*X*, *Y*). Os elementos são os seguintes:

- A - Semieixo maior ou Distância média.
- $e$ - Excentricidade (circulo = 0; elipse entre 0 e 1; parábola = 1; hipérbole > 1).

Os elementos seguintes permitem definir a **orientação da órbita** num sistema a 3D (*X*, *Y*, *Z*). Os elementos são os seguintes:

- $i$ - Inclinação da orbita.
- $\omega$ - Argumento do perigeu

O próximo elemento serve de “ligação”, da órbita do elemento que se encontra a ser calculado, ao planeta Terra.

- $\Omega$ - Longitude da node ascendente.

O último elemento serve para identificar aonde se encontra o objeto, que se encontra a ser calculado, ao longo da sua órbita.

- $M$ - Anomalia Média (0 = Perigeu; 180 = Apogeu).

Os elementos acima descritos, são o “ponto de partida” para se começar a efetuar o cálculo da órbita da Lua (e Sol). Contudo, existem outros elementos orbitais que também são utilizados durante o cálculo orbital do *software* ItTracker. Os restantes elementos orbitais utilizados nos algoritmos de cálculo da posição lunar (e solar) são os seguintes:

- $v$ - Verdadeira Anomalia.
- $\varepsilon$ - Obliquidade da Eclíptica.
- E - Anomalia Excêntrica.
- $r$ – Distância entre a Terra e o objeto celestial.

## 3.2 Algoritmos Computacionais

Nesta secção, é descrita numa primeira parte as especificações que foram tidas em contas para os algoritmos matemáticos elaborados no cálculo da órbita lunar. Numa segunda parte, são descritos os algoritmos matemáticos e método sequencial de cálculo. Esta descrição tem como base os cálculos efetuados pelo ItTracker.

Para o caso da posição do Sol foram efetuadas simplificações aos elementos orbitais. Os elementos orbitais utilizados para o cálculo da posição do Sol encontram-se descritos neste capítulo. Mas como a forma de cálculo é muito similar à da efetuada para a localização da Lua, a partir da secção 3.2.5, a descrição dos algoritmos é “praticamente” só feita para o cálculo da órbita da Lua. Contudo, toda a sua forma e método de cálculo pode ser vista em Anexo C.

### 3.2.1 Especificações do *software* ItTracker

De seguida podem ser vistas as especificações que foram tidas em conta para se alcançar o resultado final de se conseguir obter o valor final de azimute e elevação com a margem de erro definida.

- Todos os valores calculados, para os algoritmos existentes no ItTracker foram calculados/convertidos para º graus e no formato de *Java double* (número real), para se obter uma maior precisão de resultados.
- Existem algoritmos que são uma simplificação das fórmulas genéricas para cálculos orbitais, de modo a melhorar a performance e facilidade de cálculo [23].
- Todas as simplificações têm em conta a precisão requerida para o projeto.
- Os algoritmos simplificados foram testados sem simplificações, obtendo-se em nos algoritmos testados, o mesmo resultado para o grau de precisão (margem de erro) definida para o cálculo da posição da Lua.

Os algoritmos e valores das variáveis utilizadas para os cálculos realizados pelo programa desenvolvido para esta dissertação, permitem que a margem de erro de cálculo se encontre sempre situada entre 1 a 2 arco minutos para o cálculo da órbita da Lua. Como 1 arco minuto equivale a 0.016º (1 minuto/60º), os valores apresentados terão um erro entre 0.0166º e 0.033º.


Figura 3.1: Margem de Erro

Os valores orbitais, bem como a fórmula de cálculo do JD encontra-se restringida a uma data de validade que vai até 31/12/2099. Esta restrição serve para garantir que a margem de erro se mantenha sempre dentro dos valores definidos para o cálculo da órbita lunar. Os fatores que influenciam a margem de erro, e que não foram tidos em conta [22], são os seguintes:

- Nutação, Aberração, e pequenas perturbações planetárias são ignoradas.
- A precessão é ignorada.
- Termos de ordem superior nos elementos orbitais são ignorados.

- A diferença entre a hora terrestre/hora nas efemérides (TT/ET) e tempo universal coordenado (UTC) é ignorada.

De seguida podem ser vistos os algoritmos matemáticos responsáveis pelos cálculos efetuados. As variáveis e fórmulas utilizadas no código fonte do ItTracker, foram "simplificadas" (no seu contexto em linguagem Java) para uma melhor compreensão, e visualização, das próprias nesta dissertação. A explicação sobre os métodos/algoritmos responsáveis por normalizações e conversões de valores podem ser vistos na secção 3.2.9 deste capítulo. Todo o código em linguagem Java pode ser verificado no Anexo C sem as simplificações de leitura descritas.

## 3.2.2 JD e MJD

Como já referido no capítulo anterior, trata-se de uma sequência de números inteiros, um para cada dia. Este método de cálculo simplifica a tarefa de determinar o número de dias transcorridos entre duas datas.

Para se começar a efetuar os cálculos orbitais, é necessário primeiro obter os valores de **Data** e **Hora** em formato unitário. Após se obter os valores de data e hora pretendidos, é possível começar a efetuar o cálculo do **JD** e **MDJ**. O cálculo é efetuado pelo algoritmo da seguinte forma:

$$\mathrm{JD} = (365.25 * year) + \big(30.6001 * (month + 1)\big) - 15 + 1720996.5 + day + \left(\frac{UT}{24}\right) \quad (3.1)$$

Caso o valor do mês seja menor ou igual a 2 é necessário somar 12 ao valor de mês e subtrair um ano para que a coerência e fiabilidade dos algoritmos se mantenha.

$$\mathrm{year} = \mathrm{year} - 1 \quad (3.2)$$

$$month = month + 12 \quad (3.3)$$

Depois de ser efetuado o cálculo do JD, é necessário efetuar o cálculo do MJD. Como, já referido no capítulo 2, o cálculo é efetuado da seguinte forma:

$$\mathrm{MJD} = \mathrm{JD} - 2451543.5 \quad (3.4)$$

### 3.2.3 Elementos Orbitais, Latitude e Longitude eclíptica

Os valores dos elementos orbitais para a Lua, estão dimensionados para uma precisão e validade de data de cálculo definidas. Nesta secção, pode ser visto como é efetuado o cálculo (e o valor) de cada um dos elementos orbitais, da Lua e Sol, calculados nesta dissertação.

- Cálculo e valores dos **elementos orbitais** da **Lua**:

$$\Omega = 125.1228 - 0.0529538083 * \text{MJD} \quad (3.5)$$
$$\omega = 318.0634 - 0.1643573223 * \text{MJD} \quad (3.6)$$
$$M = 115.3654 + 13.0649929509 * \text{MJD} \quad (3.7)$$
$$i = 5.1454 \quad (3.8)$$
$$A = 60.2666 \quad (3.9)$$
$$e = 0.054900 \quad (3.10)$$
$$\varepsilon = 23.4393 - 3.56 * 10^{-7} * \text{MJD} \quad (3.11)$$

- Cálculo e valores dos **elementos orbitais** do **Sol:**

$$\Omega = 0 \quad (3.12)$$
$$\omega = 282.9404 + 4.70935 * 10^{-5} * \text{MJD} \quad (3.13)$$
$$M = 356.0470 + 0.9856002585 * \text{MJD} \quad (3.14)$$
$$i = 0 \quad (3.15)$$
$$A = 1.0 \quad (3.16)$$
$$e = 0.016709 - 1.15 * 10^{-9} * \text{MJD} \quad (3.17)$$
$$\varepsilon = 23.4393 - 3.56 * 10^{-7} * \text{MJD} \quad (3.18)$$

Após o cálculo, ou definição dos valores para os elementos orbitais (utilizado $A$ =1.0 AU para o Sol como simplificação), é necessário efetuar o cálculo da anomalia excêntrica. O cálculo deste elemento é efetuado através de um processo iterativo. Foram utilizadas duas variáveis para o cálculo iterativo da **Anomalia Excêntrica** que podem ser vistas de seguida:

$$E0 = M + 180/\pi * e * \sin M * (1 + e * \cos M) \quad (3.19)$$
$$E1 = E0 - (E0 - 180/\pi * e * \sin e - M)/(1 - e * \cos E0) \quad (3.20)$$

Para que se possa efetuar o cálculo iterativo é verificado se a diferença entre $E0$ e $E1$ é inferior a $1 * 10^{-8}$. Até que este valor seja superior à diferença entre $E0$ e $E1$, o algoritmo efetua a operação $E0 = E1$, e volta novamente a correr a fórmula 3.20. Quando a condição for atingida, o algoritmo devolve a variável $E1$ como sendo este o valor iterado da anomalia excêntrica.

Após se achar o valor da anomalia excêntrica através do método de cálculo iterativo, é efetuado o cálculo das **coordenadas retangulares *X*** e ***Y***, no **plano orbital da eclíptica**, aonde o valor da coordenada *X* aponta para o ponto de perigeu/periélio (Lua/Sol).

$$X = A * (\cos(E) - e) \tag{3.21}$$

$$Y = A * (\sqrt{(1 - e * e)} * \sin E) \tag{3.22}$$

Agora que já foram calculadas as coordenadas retangulares no plano da eclíptica, é possível agora obter os últimos elementos orbitais $\boldsymbol{r}$ (**Distância**) e $\boldsymbol{v}$ (**Verdadeira Anomalia** em gaus).

$$r = \sqrt{(X * X + Y * Y)} \tag{3.23}$$

$$v = \text{atan2}(\text{Y}\,,\ \text{X}) \tag{3.24}$$

Com os elementos orbitais já todos calculados, é possível então saber onde se encontra a Lua no seu plano orbital. Para se obter esta posição, calcula-se a **posição geocêntrica** da Lua num sistema de coordenadas eclípticas (***Xeclip***, ***Yeclip***, ***Zeclip***).

$$Xeclip = r * (\cos\Omega * \cos(v + \omega) - \sin\Omega * \sin(v + \omega) * \cos i \tag{3.25}$$

$$Yeclip = r * (\sin\Omega * \cos(v + \omega) + \cos\Omega * \sin(v + \omega) * \cos i \tag{3.26}$$

$$Zeclip = r * \sin(v + \omega) * \sin i \tag{3.27}$$

E por fim é efetuado o cálculo da **Longitude** e **Latitude** da Lua no plano da eclíptica.

$$LongiM = \text{atan2}(\text{Yeclip}\,,\ \text{Xeclip}) \tag{3.28}$$

$$LatM = \text{atan2}\big(\text{Zeclip}\,, \sqrt{(\text{Xeclip} * \text{Xeclip} + \text{Yeclip} * \text{Yeclip})}\big) \tag{3.29}$$

No caso do Sol, não foi necessário efetuar o cálculo em coordenadas eclípticas. Efetuou-se somente seguintes cálculos:

$$LongiS = v + \omega \tag{3.30}$$

$$LatiS = 0 \tag{3.31}$$

O valor de Latitude do Sol é igualado a 0 para simplificar, visto que o seu valor é tão reduzido que para o seu cálculo, e objetivo de cálculo, se torna "irrelevante".

### 3.2.4 Perturbações na Lua

O valor de Latitude e Longitude calculados anteriormente para a Lua não se encontram dentro da margem de erro requerida. Para que o valor alcançado seja aceitável, existe a necessidade de ter em conta certos termos de perturbação [22]. Primeiro é necessário efetuar o cálculo das seguintes variáveis.

$Lm = \Omega + \omega + M$ (Longitude média da Lua) (3.32)

$F = Lm - \Omega$ (Argumento da Latitude da Lua) (3.33)

$D = Lm - \omega + M$ (Alongmento Médio da Lua) (3.34)

$M$ (Anomalia média do Sol) (3.35)

$Ms = \omega + M$ (Longitude média do Sol) (3.36)

Com estes valores já calculados, efetua-se a soma dos seguintes valores de perturbação para a **Longitude** da Lua:

$$LongiM = LongiM + (-1.274 * \sin(M - 2 * D) + 0.658 * \sin(2 * D) - 0.186 * \sin Ms - 0.059 * \sin(2 * M - 2 * D) - 0.057 * \sin(M - 2 * D + Ms) + 0.053 * \sin(M + 2 * D) + 0.046 * \sin(2 * D - Ms) + 0.041 * \sin(M - Ms) - 0.035 * \sin D - 0.031 * \sin(M + Ms) - 0.015 * \sin(2 * F - 2 * D) + 0.011 * \sin(M - 4 * D)) \tag{3.37}$$

Sendo também necessário efetuar a soma do valor das perturbações para a **Latitude** da Lua:

$$LatM = LatM + (-0.173 * \sin(F - 2 * D) - 0.055 * \sin(M - F - 2 * D) - 0.046 * \sin(M + F - 2 * D) + 0.033 * \sin(F + 2 * D) + 0.017 * \sin(2 * M + F)) \tag{3.38}$$

E por fim, efetuar também a soma dos valores de perturbação para **Distância** da Lua:

$$r = r + (-0.58 * \cos(M - 2 * D) - 0.46 * \cos(2 * D) \quad (3.39)$$

Após a adição dos valores de perturbação para a Longitude, Latitude e Distância da Lua, os valores de margem de erro voltam novamente a ficar dentro da margem definida.

### 3.2.5 *Right Ascenscion* e *Declination*

Como já foi calculado anteriormente os valores das coordenadas eclípticas, para a Latitude e Longitude da Lua numa posição geocêntrica do plano da eclíptica, é agora necessário efetuar o cálculo das **coordenadas geocêntricas retangulares *Xgeo*, *Ygeo*** e ***Zgeo*** (para o plano geocêntrico). No caso do Sol terá que ser efetuar o cálculo extra para posição geocêntrica (a Lua já se encontra num plano geocêntrico, mas o Sol encontra-se numa posição heliocêntrica). Deste modo, o cálculo destas coordenadas para a Lua é efetuado da seguinte forma:

$$Xgeo = r * \cos(LongiM) * \cos(LatM) \quad (3.40)$$
$$Ygeo = r * \sin(LongiM) * \cos(LatM) \quad (3.41)$$
$$Zgeo = r * \sin(LatM) \quad (3.42)$$

Efetuado este cálculo, efetua-se a rotação no plano entre os eixos *Ygeo* e *Zgeo*, pelo valor da obliquidade da eclíptica, de modo a obter as **coordenadas retangulares equatoriais** (para o plano equatorial):

$$Xequat = Xgeo \quad (3.43)$$
$$Yequat = Ygeo * \cos(\varepsilon) - Zgeo * \sin(\varepsilon) \quad (3.44)$$
$$Zequat = Ygeo * \sin(\varepsilon) + Zgeo * \cos(\epsilon) \quad (3.45)$$

Após o cálculo dos valores das coordenadas equatoriais ***Xequat***, ***Yequat***, ***Zequat***, já é possível calcular os valores das coordenadas equatoriais **RA** e **Dec**.

$$\mathrm{RA} = \mathrm{atan2}(\mathrm{Yequa}, \mathrm{Xequat}) \quad (3.46)$$
$$\mathrm{Dec} = \mathrm{atan2}\left(\mathrm{Zequat}, \sqrt{(\mathrm{Xequat} * \mathrm{Xequat} + \mathrm{Yequat} * \mathrm{Yequat}}\right) \quad (3.47)$$

### 3.2.6 Hour Angle (GMST)

O cálculo do tempo sideral ($SidTime$), serve para que o valor de RA saiba aonde está a parte do céu que se encontra precisamente a sul. O valor de GMST0 é o valor do Sideral Time no meridiano de Greenwich às 00:00, para o momento do cálculo. A *Longitude* é o valor da posição do observador na Terra que queremos calcular.

Sendo assim, primeiro efetua-se o cálculo do valor de $GMST0$. É feita uma divisão do valor da longitude média do Sol ($LongiS$) por 15, para se efetuar a passagem do valor de um formato graus para horas.

$$GMST0 = LongiS \ / \ 15 \ + \ 12 \tag{3.48}$$

Seguidamente efetua-se o cálculo do valor de $SidTime$. É novamente efetuada uma divisão por 15 ao valor da $Longitude$ do observador local, para se efetuar uma passagem de um formato de graus para horas.

$$SidTime = \ GMST0 + UT + Longitude/15 \tag{3.49}$$

Com o valor de $SidTime$ calculado, calcula-se agora o valor $LST$ (Local Sideral Time), que corresponde nada mais que ao valor de RA no meridiano local para a $Longitude$ introduzida pelo utilizador ($Longitude$ referente ao ponto de observação na Terra).

$$LST = SidTime * 15 \tag{3.50}$$

E por fim, subtrai-se o valor de $LST$ pelo valor RA, calculado anteriormente, de modo a se obter o **HA**, que corresponde ao valor do $LST$ no meridiano de Greenwich.

$$\mathrm{HA} = LST - \mathrm{RA} \tag{3.51}$$

### 3.2.7 *Topocentric Right Ascension* e *Declination*

Para se efetuar o cálculo na **posição Topocêntrica** de RA e Dec da Lua (este cálculo não é efetuado para o Sol dada a distância), é necessário ter em conta o achatamento da Terra, visto que a Terra não é perfeitamente redonda. Começa-se então por transformar o valor da $Latitude$ (latitude do observador na Terra) para $Latgeo$, e do valor de distância ao centro da Terra para $rgeo$:

$$Latgeo = Latitude - 0.1924 * \sin(2 * Latitude) \quad (3.52)$$

$$rgeo = 0.99833 + 0.00167 * \cos(2 * Latitude) \quad (3.53)$$

Utiliza-se também um ângulo auxiliar $g$, que irá ser utilizado para o caso do valor de $Latgeo$ ser diferente de 0.

$$g = \text{atan}(\tan(Latgeo) / \cos(\text{HA})) \quad (3.54)$$

Efetua-se o cálculo da variável $mpar$ que equivale ao Horizontal Parallax (no caso da Lua Moon Parallax), que designa o tamanho aparente do raio equatorial da Terra, visto da Lua de modo a corrigir o valor de $Elevacao$.

$$mpar = \text{asin}(1/r) \quad (3.55)$$

Agora que já foram calculadas todas as variáveis intermédias, que permitem que o grau de precisão seja mantido dada a proximidade da Lua, é então possível agora efetuar o cálculo do valor **Topocêntrico** de **RA** ($topRA$).

$$topRA = \text{RA} - mpar * rgeo * \cos(Latgeo) * \sin(HA) / \cos(\text{Dec}) \quad (3.56)$$

Para se efetuar o cálculo da $topDec$, e como já referido anteriormente, terá que se ter em conta o facto do valor de $Latgeo$ ter de ser diferente de 0. Se o valor for diferente de 0 (valor de input de *Latitude* é diferente de 0), utiliza-se a fórmula 3.56. Caso seja igual a 0, terá de se utilizar a fórmula 3.57, visto que o $\sin(0) = 0$ (o algoritmo prevê este caso).

$$topDec = \text{Dec} - mpar * rgeo * \sin(Latgeo) * \sin(g - \text{Dec}) / \sin(g) \quad (3.57)$$

$$topDec = \text{Dec} - mpar * rgeo * \sin(-\text{Dec}) * \cos(\text{HA}) \quad (3.58)$$

Durante uma última fase de testes de cálculo do *software*, verificou-se que os valores topocêntricos ($topRA$, $topDec$) não estavam dentro do valor de margem de erro definida (<0.033º). Deste modo, e na versão final do ItTracker que é descrito no capítulo seguinte desta dissertação, foi utilizada a conversão de coordenadas com valores geocêntricos, em vez de se utilizar a conversão com valores topocêntricos.

### 3.2.8 Azimute e Elevação

Para o cálculo final de *Azimute* e *Elevação*, foi necessário primeiro converter os valores das variáveis HA e Decl para um sistema de coordenadas retangulares no plano geocêntrico.

- A coordenada ***X*** aponta para o equador celestial no **Sul**.

$$X = \cos(\text{HA}) * \cos(\text{Dec}) \quad (3.59)$$

- A coordenada ***Y*** aponta para o horizonte que se encontra a **Oeste**.

$$Y = \sin(\text{HA}) * \cos(\text{Dec}) \quad (3.60)$$

- A coordenada ***Z*** aponta para o polo celestial no **Norte**.

$$Z = \sin(\text{Dec}) \quad (3.61)$$

De seguida é necessário efetuar a rotação das coordenadas calculadas previamente, ao longo de um eixo que vai de **Este** para **Oeste**, ou seja uma rotação ao longo do eixo dos *Y*. Assim, o eixo dos *Z* passa agora a apontar para o ponto de Zénite.

$$Xhor = X * \sin(Latitude) - Z * \cos(Latitude) \quad (3.62)$$

$$Yhor = Y \quad (3.63)$$

$$Zhor = X * \cos(Latitude) + Z * \sin(Latitude) \quad (3.64)$$

Agora já se pode finalmente obter os de *Elevação* e *Azimute*, sendo este o objetivo principal de todos os cálculos realizados anteriormente. Efetua-se a adição de 180 ao valor de $Azimute$, para que este valor seja lido no formato de Azimute Geográfico (como pretendido).

$$Azimute = atan2(Yhor, Xhor) + 180 \tag{3.65}$$

$$Elevacao = atan2(Zhor, \sqrt{(Xhor * Xhor + Yhor * Yhor)}) \tag{3.66}$$

Por fim, efetua-se uma correção final ao valor de *Elevação.* Esta correção é efetuada, para que o valor final da $Elevação$ da Lua mantenha o grau de precisão definido. Após uma série de testes efetuados, verificou-se que não era necessário efetuar a correção ao valor de $Azimute$ para se manter o valor de margem de erro. Sendo assim, o valor da correção final do valor de *Elevação* é feito através do cálculo da fórmula 3.66.

$$Elevacao_{Corrigida} = Elevacao - mpar * \cos(elevacao) \tag{3.67}$$

### 3.2.9 Conversões e Normalizações

Durante a realização do cálculo dos algoritmos descritos para cálculo da órbita da Lua (e Sol), foram realizados cálculos intermédios de normalizações e conversões de valores. Estas, normalizações e conversões, são fundamentais para o cálculo de todos os algoritmos descritos na secção 3.2 do capítulo 3.. Todas as conversões e normalizações efetuadas podem ser vistas no Anexo A. As normalizações e conversões efetuadas foram as seguintes:

- Conversão de valor formato *double* para Graus (entre 0º e 360º).
- HA (garante que o valor de HA fica entre -180º e 180º).
- Graus, Minutos, Segundos (transforma um valor no formato *double* para o formato graus (º), minutos (‘), segundos (‘’)).
- Minutos, Segundos (transforma um valor no formato *double* para o formato minutos (‘), segundos (‘’)).
- Horas, Minutos, Segundos (transforma um valor no formato *double* para o formato horas (h), minutos (m), segundos (s)).
- Arredondamentos a 4 casas decimais.
- Conversão para radianos $(\pi/180)$ e Conversão para graus $(180/\pi)$

# Capítulo 4

# *Software* ItTracker e Resultados

No capítulo 3 foram descritos os elementos orbitais bem como a forma sequencial e métodos de cálculo efetuados pelos algoritmos computacionais do ItTracker, de modo a se obter o valor final do Azimute e Elevação da Lua (e Sol), para um observador que se encontra localizado num ponto específico na Terra.

No capítulo 4, e numa primeira parte, pode ser visto como se encontra organizado o código fonte dentro das suas classes do *software* ItTracker, compilado em linguagem de programação Java. Numa segunda parte pode ser visto como, e aonde, deverão ser realizados todos os *inputs* necessários para que o programa funcione corretamente. Numa terceira parte, será demostrando os valores de *output* calculados, e na parte final deste capítulo, será comprovada a veracidade dos cálculos efetuados, efetuando uma comparação dos resultados obtidos com os resultados de um outro *software* (disponibilizado *online*), para o efeito de cálculo orbital lunar.

## 4.1 Especificações e Diagrama de classes

Para a realização do ItTracker, *software* encarregue de efetuar os cálculos orbitais, foi escolhida a linguagem de programação Java, na sua versão JDK7.


Figura 4.1: Versão Java

O código fonte compilado, e criado, através da utilização do *software* Eclipse. O ficheiro "ItTracker.Jar" (ficheiro executável,) foi compilado para correr em sistema operativo Windows (versão 7 ou superior).


Figura 4.2: *Software* Eclipse e Sistema Operativo Windows

Os *links* para *download* do compilador Eclipse e das bibliotecas Java JDK (na sua versão mais recente) podem ser vistos em Anexo B.

Na figura 4.3, que pode ser vista de seguida, é possível observar como se encontram organizadas as classes compiladas, em Java, para o programa ItTracker.

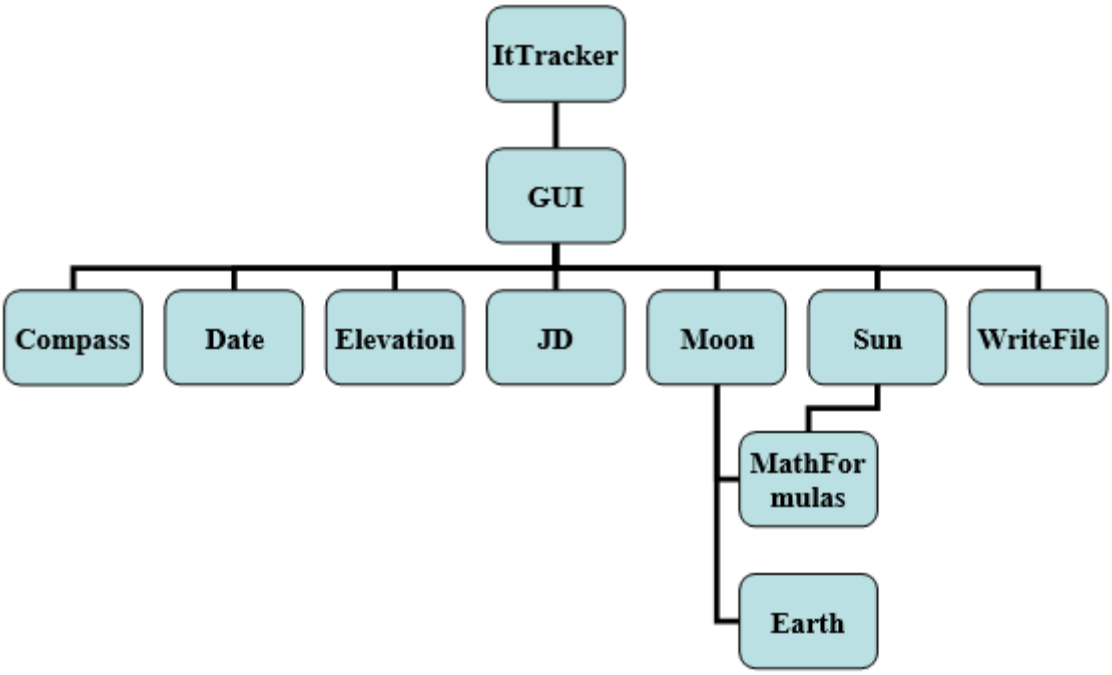


Figura 4.3: Diagrama de Classes ItTracker

Seguidamente pode se observar uma breve descrição do que cada uma das classes criadas é responsável por efetuar no seu código fonte em Java, de modo a que o programa desenvolvido funcione corretamente.

Descrição das classes em Java:

- **ItTracker:** Classe *Main*, responsável pela gestão e execução do *software*.
- **GUI:** Classe responsável por criar e gerir de todas as funcionalidades gráficas.
- **Compass:** Classe responsável pelo desenho e funcionalidades do compasso gráfico.
- **Elevation:** Classe responsável pelo desenho e funcionalidades da Elevação gráfica.
- **JD:** Classe responsável pela conversão para JD e MJD.
- **Date:** Classe responsável por obter a data e hora do PC e conversões do valor de UTC para valor unitário.
- **Moon:** Classe responsável por todos os elementos e algoritmos de cálculo orbital da Lua.
- **Sun:** Classe responsável por todos os elementos e algoritmos de cálculo orbital do Sol.
- **WritheFile:** Classe responsável por criação e escrita de ficheiro "input.dat"
- **MathFormulas:** Classe que contém várias fórmulas matemáticas (conversões, normalizações, etc.) comuns aos algoritmos da Lua e Sol.
- **Earth:** Classe com valores de elementos da Terra.

## 4.2 ItTracker GUI *Inputs*

Seguidamente, podem ser visualizados todos os *Inputs* necessários para a realização dos algoritmos efetuados, bem como o formato em que estes devem ser inseridos pelo utilizador, de modo a que o *software* funcione corretamente.

### 4.2.1 Latitude e Longitude

Estes campos são referentes ao valores de **Latitude** e **Logitude** local na Terra à qual se pretende efectuar a verificação do valor de Azimute e Elevação da Lua e Sol.


Figura 4.4:Input de Latitude e Longitude de posição na Terra

O valor de Latitude introduzido deverá ser positivo e entre "0º e 90º" para uma localização que se encontre no hemisfério norte. Este valor deverá ser negativo de "-90º a 0º" para o valor de uma localização que se encontre a sul do hemisfério.

O valor de Longitude introduzido para uma localização que se encontre a Este do meridiano de Greenwich terá que ter um valor positivo entre "0º e 180º". Este valor terá que ser negativo e entre "-180º e 0º" e para Oeste do meridiano de Greenwich.

A separação de valores em casas decimais para a *Latitude* e *Longitude*, terá que ser efetuada com um ponto "." e não com uma vírgula ",". Ambos os valores não poderão conter o símbolo de graus "º".

De seguida, pode ser visto um exemplo da introdução correta de valores de *input* para os campos **Latitude** e **Longitude** de uma localização geográfica na Terra (ex. ISCTE-IUL em Portugal) com com as coordenadas:

- **Latitude** = 38.74879º
- **Longitude** = -9.15357º

Figura 4.5:*Input* correto de Latitude e Longitude

## 4.2.2 *Local Time* e *Set Time*

A *checkbox* **Local Time**, quando selecionada, permite efectuar os cálculos a cada segundo, para o valor atual da hora, minutos e segundos, que o PC usado pelo utilizador se encontra configurado, até que o utilizador decida parar a operação.

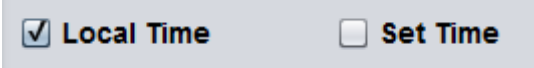

Figura 4.6: *Checkbox* Local Time Selecionada

A hora do PC terá que se encontrar no fuso horário UTC, de modo a que os resultados calculados em tempo real se encontrem correctos. O código atual encontra-se configurado com a hora UTC para um utilizador que utilize um fuso horário UTC +1. Pode ser vista no Anexo A, aonde se encontra esta configuração.

A *checkbox* **Set Time**, quando selecionada, permite que o utilizador defina uma data e hora para a qual pretende efectuar os cálculos. A seleção de cada uma das checkbox, desactiva a que se encontrava selecionada anteriormente.

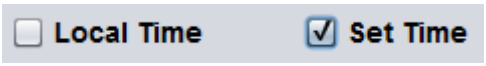


Figura 4.7: *Checkbox* Set Time Selecionada

A hora terá que ser inserida consoante o fuso horário UTC na secção "Hour(UTC)". O campo "HH" corresponde ao valor das horas, o campo "MM" ao valor dos minutos, e "SS" corresponde ao valor dos segundos. Para os valores da secção "Date", "DD" corresponde ao dia, "MM" ao mês e "YYYY" corresponde ao ano.


Figura 4.8: Formato *Input* Hora e Data

Na figura 4.9 pode ser visto um exemplo da introdução correta dos valores de hora e data nos seus respetivos campos e seleção de uma das *checkboxes*.

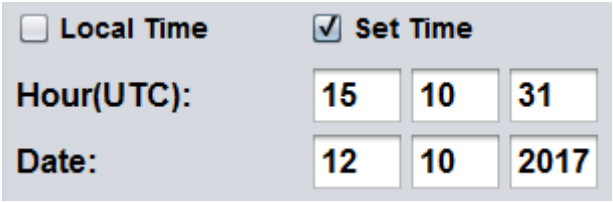


Figura 4.9: *Input* de Hora e Data modo Set Time

### 4.2.3 *Check Connection*

Quando a *checkbox* **Check Connection** se encontra selecionada, é possível ao utilizador introduzir as coordenadas para uma segunda localização. O preenchimento correto dos valores é idêntico ao já verificado no preenchimento dos valores dos campos **Latitude** e **Longitude** vistos em 4.2.1. Esta opção serve para verificar se duas localizações se encontram com a Lua visível (no seu horizonte), para o espaço de tempo definido anteriormente ("*Local Time*" ou "*Set Time*"). O algoritmo (que pode ser visto em Anexo A) implícito nesta *checkbox*, dará a "conexão" com o resultado de possível, sempre que o valor da Elevação da Lua, para as duas localidades distintas, seja superior a 5º.

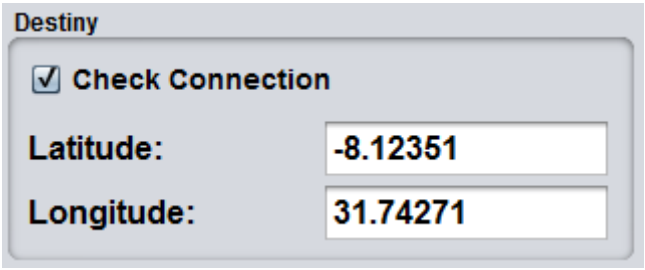


Figura 4.10: *Check Destination* - Latitude e Longitude de Destino

Um exemplo do correto preenchimento de valores, e seleção da *checkbox* **Check Connection** pode ser visto na figura 4.10.

### 4.2.4 Botões Calculate e Stop

O botão **Calculate**, inicia o cálculo dos algoritmos orbitais, conforme as informações de *input* selecionadas anteriormente. Se a *checkbox* "*Local Time*" estiver selecionada, todos os campos de *Input* (bem como o botão Calculate) ficarão descativos durante o decorrer desta operação.


Figura 4.11: Botões Calculate e Stop

O botão **Stop**, serve parar a operação de cálculo, quando esta é realizada com a seleção da *checkbox* "*Local Time*" ativa. Após o *click* neste botão, todos os campos de *Input* (bem como o botão "*Calculate*") ficarão novamente ativos.

### 4.2.5 Visão Geral *Inputs*

Na figura 4.12, pode ser vista uma visão geral do painel de *inputs* do programa ItTracker.

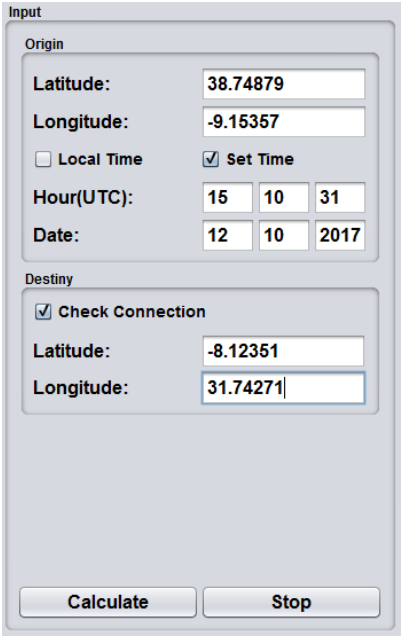


Figura 4.12: Visão Geral *Inputs*

## 4.3 ItTracker GUI *Output*

Seguidamente, podem ser vistos os elementos de *output* do ItTracker, aquando da realização de uma operação de cálculo.

### 4.3.1 *Output* – Lua

O painel "*Moon Orbital Values*" permite a visualização dos valores das variáveis consideradas mais relevantes para o cálculo da posição da órbita da Lua, efetuada a partir de um observador que se localizado encontra na Terra e como pode ser visto na figura 4.13.

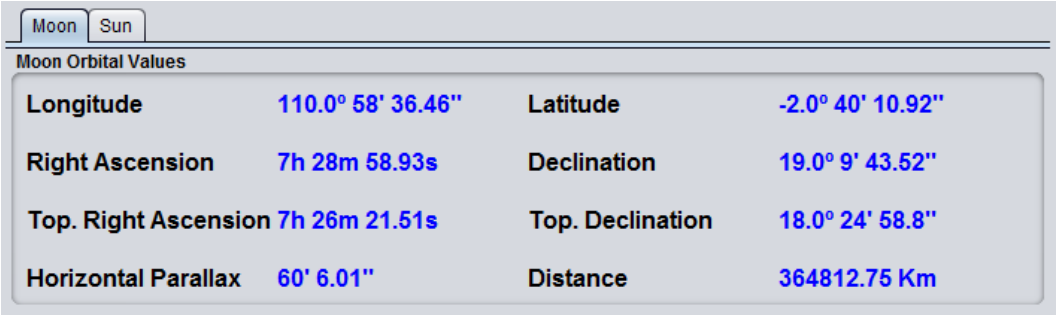


Figura 4.13: *Output* painel *Moon Orbital Values*

De seguida, pode ser vista uma breve descrição das unidades em que se encontram as variáveis de *output* vistas na figura 4.13:

- Os valores de Longitude, Latitude, *Declination* e *Top. Declination*, encontram-se no formato graus (º), minutos (´) e segundos (´´).
- Os valores de *Right Ascension* e *Top. Right Ascension* encontram-se no formato horas (h), minutos(m), segundos(s).
- O valor de *Horizontal Parallax* encontra-se no formato minutos (´), segundos (´´).
- O valor de *Distance* encontra-se em quilómetros (Km).

O painel "*Azimuth and Elevation*", permite visualizar os valores finais de *Azimute* e *Elevação* da Lua (valores em inglês no *output* do *software*). Ambos os valores se encontram numa unidade em graus (º), como pode ser visto na figura 4.14

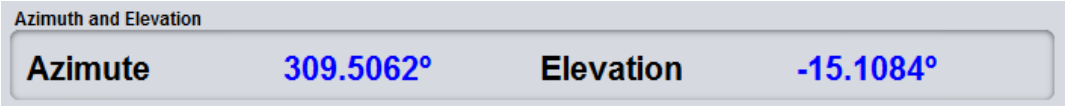


Figura 4.14: *Output Azimuth and Elevation*

Para uma compreensão mais simples do valor de *Azimute* da Lua, face à localização de um observador na Terra, foi criada uma bússola que aponta para o valor de *Azimute* calculado. Um valor de *Azimute* igual a 0º = Norte (N), 90º = Este (E), 180º = Sul (S) e 270º = Oeste (W), como pode ser visto na figura seguinte.

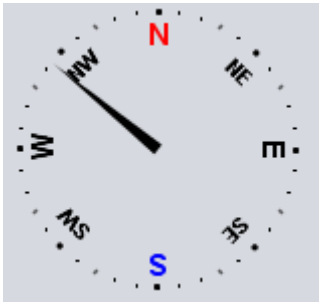

Figura 4.15: Representação de Azimute em Bussola

Foi também desenvolvido uma forma de representar visualmente o valor de *Elevação* da Lua. Caso o ângulo de *Elevação*, seja inferior a 0º, então a Lua encontra-se abaixo da linha do horizonte, ou seja, a Lua não se encontra visível para a localização pretendida. Caso este valor seja superior a 0º, então a Lua encontra-se visível e acima da linha do horizonte.

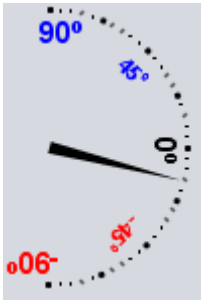

Figura 4.16: Representação gráfica da Elevação

Por fim, quando a *checkbox* "***Check Connection***" se encontra selecionada (secção 4.2.3), o resultado obtido quando ambas as localizações se encontram com a Lua visível, sendo a conexão possível, é o seguinte:

Figura 4.17: Informação mostrada quando a ligação é possível

Quando ambas as localizações não se encontram com o valor de *Elevação* da Lua acima da linha do horizonte, o resultado gráfico é o que pode ser visto na figura 4.18.

Figura 4.18: Informação mostrada quando a ligação não é possível

### 4.3.2 *Output* – Sol

O painel "***Sun Orbital Values***", permite visualizar os valores das variáveis consideradas mais relevantes para o cálculo da posição do Sol, dada a localização de um observador que se encontra na Terra, e que pode ser visto na figura 4.19.

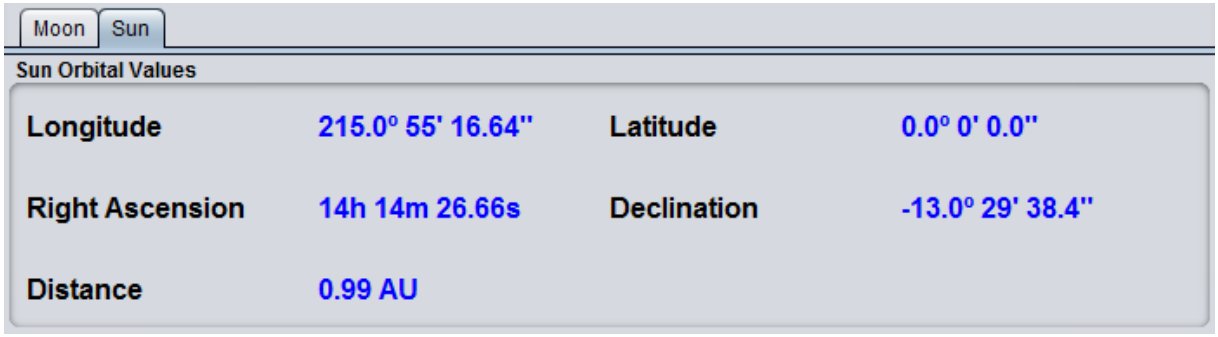


Figura 4.19: *Output Sun* painel *Orbital Values*

Visto que o Sol se encontra mais distante da Terra do que da Lua, não existe a necessidade de se calcular, e representar, os valores de *Top. Right Ascension* e *Top. Declination*, de modo a se obter valores mais precisos.

As unidades das variáveis de *output* do Sol, são idênticas às já verificadas para os valores de *output* da Lua, com a exceção à variável *Distance* que se encontra na unidade Astro Units (AU). Os valores de *Azimute* e *Elevação*, bem como as suas representações gráficas, são idênticos aos do *output* para a Lua vistos em 4.3.1.

### 4.3.3 Ficheiro "input.dat"

O ficheiro "input.dat", regista os valores de *Azimute* e *Elevação*, de modo a que seja possível a iteração do programa desenvolvido, com um hardware que fique encarregue de efetuar a ligação a uma antena parabólica.

O ficheiro é criado de raiz sempre que o utilizador utiliza a opção "*Local Time"*, e fica localizado no diretório aonde se encontra a ser corrido o ItTracker. Este ficheiro, regista os valores de *Azimute* e *Elevação* ao segundo calculados pelo programa.

O primeiro valor registado é meramente indicativo e tem o valor “1”, o segundo valor é referente ao valor de *Azimute*, e o último valor é referente ao valor da *Elevação*. Na figura 4.19, pode ser visto um exemplo do *output* gerado por este ficheiro.

```
1 188.3614 54.6725
```

Figura 4.20: Output Ficheiro "Input.dat"

### 4.3.4 GUI Visão Geral

Por fim, pode ser observada uma visão geral de toda a interface do programa ItTracker, incluindo os valores de *output* e *input*, bem como todas as representações gráficas auxiliares e o painel “*Local Time and* Date”, que corresponde à hora e data que o PC se encontra configurado e aonde o *software* está a ser executado. A visão geral do ItTracker é vista na figura 4.2.

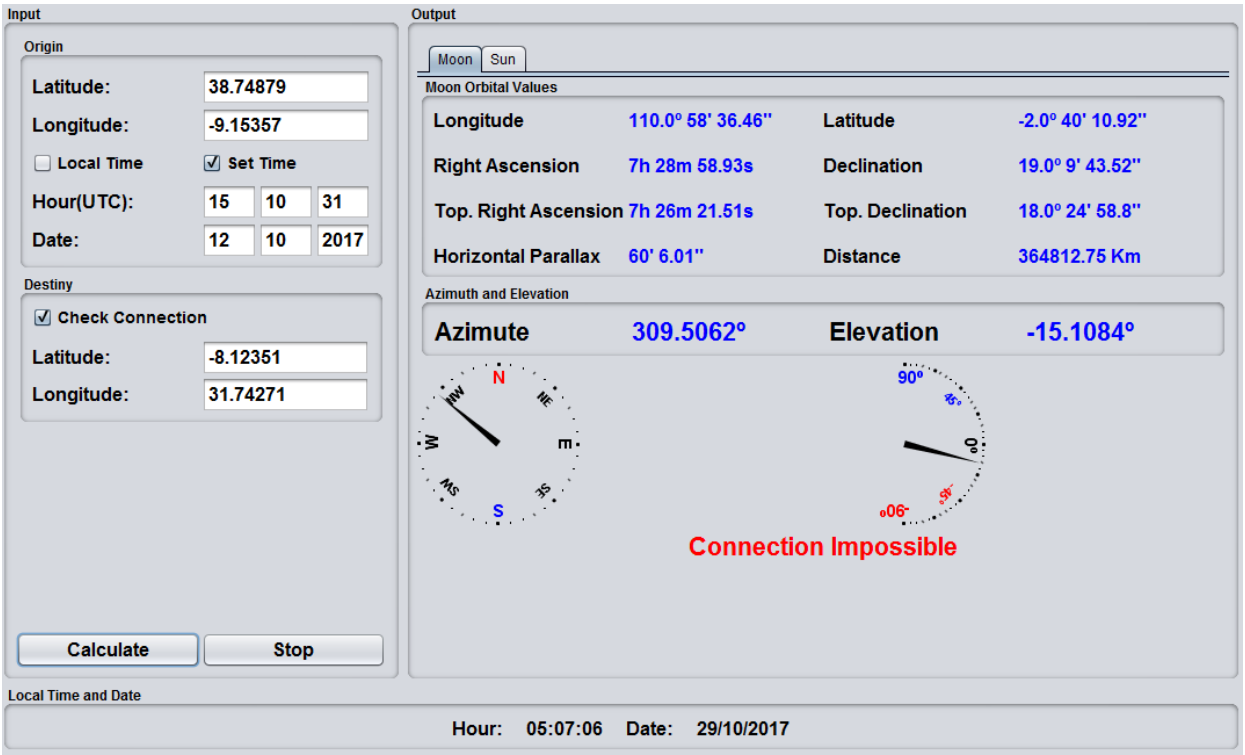


Figura 4.21: Visão Geral *software* ItTracker

## 4.4 Resultados

De modo a ser possível comprovar o correto funcionamento do programa, nomeadamente dos algoritmos matemáticos responsáveis pelo cálculo da órbita lunar dada a localização de um observador que se encontra na Terra, foi utilizado o *software* "MoonCalc.org" desenvolvido por Torsten Hoffmann 2015-2017 [24]. Este *software* encontra-se disponível nas versões online, e para aplicações Android, e ambas as versões se encontram disponíveis gratuitamente. O *software* desenvolvido por Torsten Hoffmann, encontra-se muito bem classificado, obtendo *reviews* muito positivas, dado o seu grau de fiabilidade.

Deste modo, e como método de comprovação de resultados obtidos, a escolha recaiu sobre o programa MoonCalc.org.

Foram efetuados arredondamentos aos resultados do ItTracker a duas casas decimais para os valores de *Azimute* e *Elevação*, dado que os resultados do MoonCalc.org se encontram neste formato. Este arredondamento irá facilitar a compreensão e comparação da fiabilidade dos resultados obtidos.

O primeiro passo para a comprovação de resultados obtidos, será então escolher uma localização e data para um ponto no planeta Terra. A escolha para este exemplo de comprovação de resultados foi a seguinte:

- **Latitude** = 37.74406
- **Longitude** = -25.57223
- **Data** = 15/10/2017
- **Hora** = 05:10 UTC

O passo seguinte será então introduzir as variáveis descritas acima em ambos os programas, e efetuar a comparação de resultados entre ambos.

Na figura 4.21, podem ser vistos os resultados alcançados através do cálculo orbital efetuado pelo ItTracker, desenvolvido no âmbito desta dissertação, e cujo objetivo principal é

o de se efetuar o cálculo dos valores de *Azimute* e *Elevação* da Lua, mantendo o resultado final de ambos os valores, dentro da margem de erro definida.

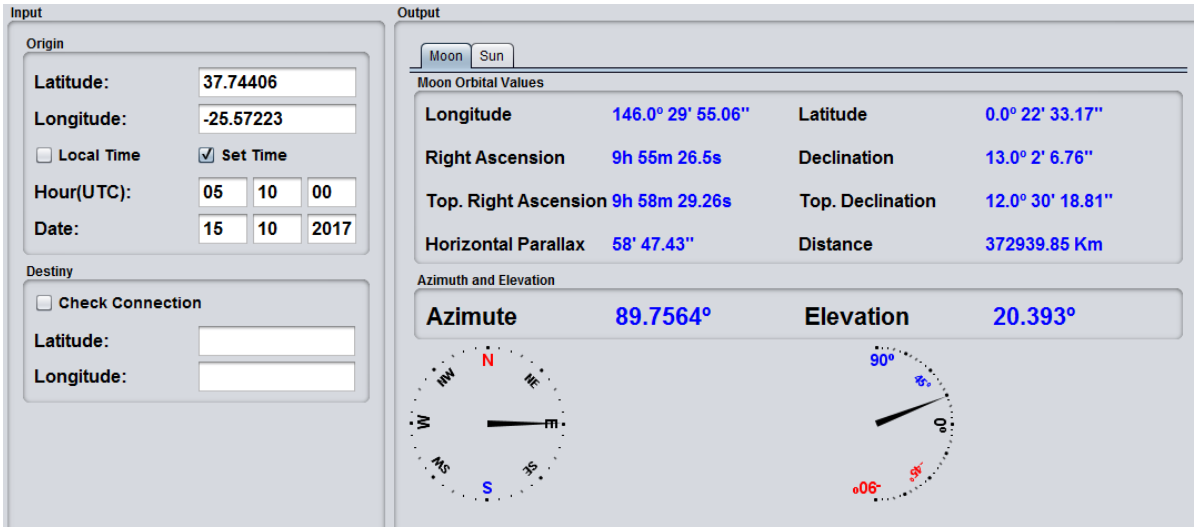

Figura 4.22: ItTracker resultados Lua - 1

Observaram-se os seguintes valores de Azimute e Elevação para o **ItTracker**:

- **Azimute** = 89.76º
- **Elevação** = 20.39º

Na figura 4.22, podem ser vistos os resultados obtidos através do *software* MoonCalc.org.

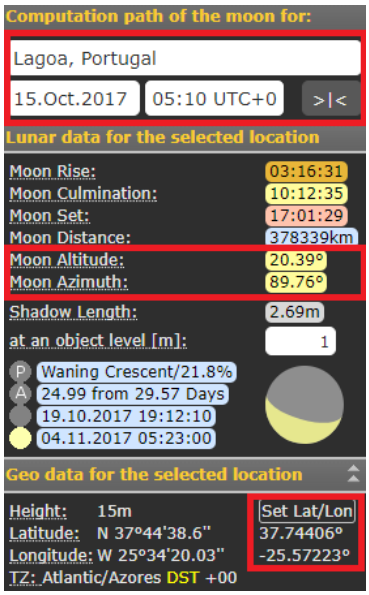

Figura 4.23: MoonCalc.org resultado Lua - 1

Sendo os valores de Azimute e Elevação obtidos pelo **MoonCalc.org** os seguintes:

- **Azimute** = 89.76º
- **Elevação** = 20.39º

Após comparação entre ambos os resultados calculados nos diferentes programas, é facilmente verificado que os resultados obtidos pelo ItTracker, *software* desenvolvido no âmbito desta dissertação, se encontram corretos e dentro da margem de erro definida.

De seguida, será efetuado um teste ao algoritmo que se encontra encarregue de efetuar o "Connection Check", explicado em 4.3.1, para o caso do cálculo da órbita Lunar.

Dado que a primeira localização escolhida já se encontrava com a Lua visível, vai ser agora escolhida uma segunda localização em que a Lua não se encontre visível para a mesma data e hora.

Foram escolhidas as seguintes coordenadas:

- **Latitude** = -56.07204
- **Longitude** = 156.97266

Após introdução dos valores, verifica-se o seguinte *output* de resultados para o **ItTracker**:

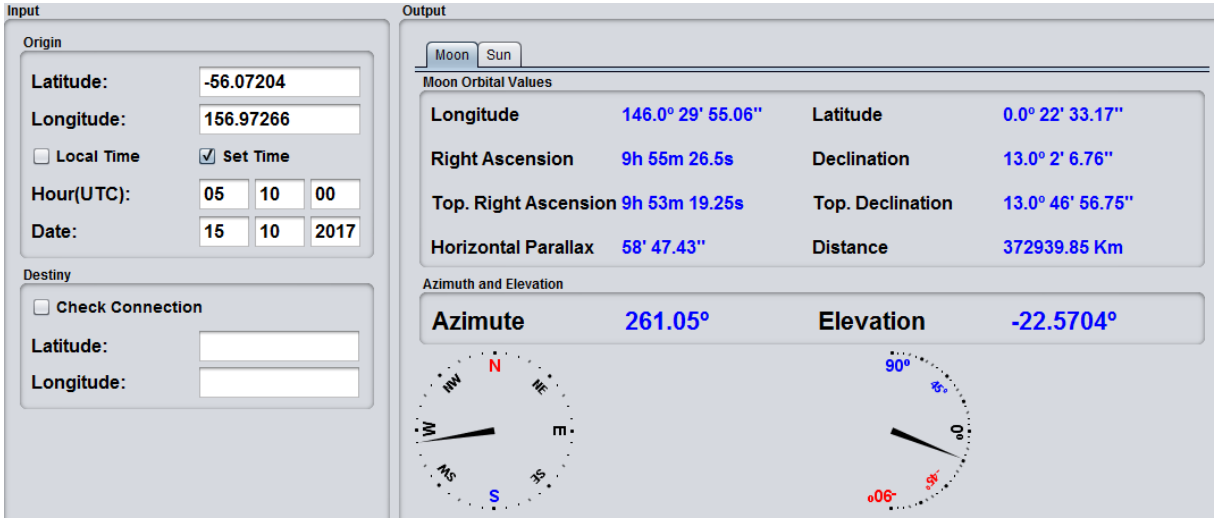

Figura 4.24: ItTracker resultados Lua - 2

Observou-se que os valores de Azimute e Elevação do **ItTracker** para a segunda localização são os seguintes:

- **Azimute** = 261.05º
- **Elevação** = -22.57º

E os seguintes resultados obtidos no **MoonCalc.org**:

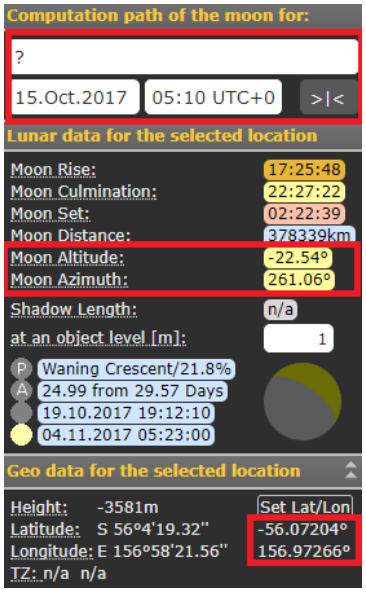

Figura 4.25: MoonCalc.org resultados Lua - 2

- **Azimute** = 261.06º
- **Elevação** = -22.54º

Verifica-se que existe uma ligeira diferença entre ambos os resultados obtidos, contudo, basta efetuar uma subtração entre ambos os resultados, para se constar que os resultados relativos ao **ItTracker** se encontram dentro da margem de erro definida para esta dissertação.

***Azimute*** (MoonCalc.org) - ***Azimute*** (ItTracker): 261.06 - 261.05 = **0.01º** (4.1)

***Elevação*** (ItTracker) - ***Elevação*** (MoonCalc.org): 22.57 - 22.54 = **0.03º** (4.2)

Dado que os resultados obtidos, para as novas coordenadas aonde a localização da Lua não se encontra visível se encontram corretos, e dentro da margem de erro definida, irá ser então

verificado, e comprovado de seguida o resultado do algoritmo “Check Connection”, podendo ser visto o *output* gerado pelo **ItTrack**, de seguida e na figura 4.25.

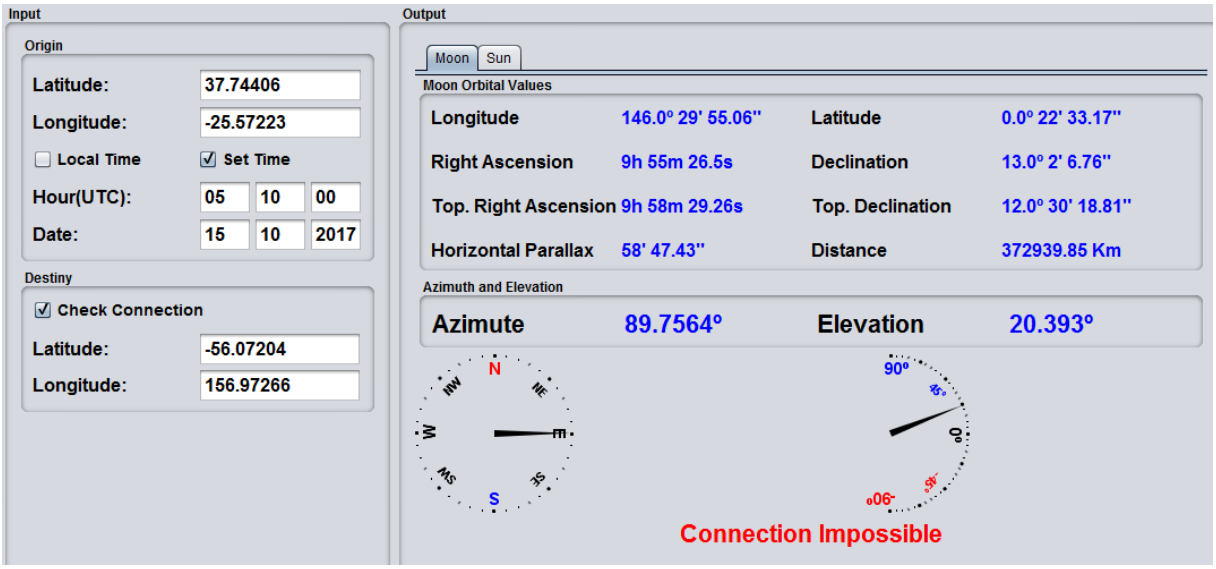

Figura 4.26: ItTracker resultado “Connection Impossible”

Para o caso do cálculo da posição do Sol, dada a localização de um observador na Terra, foi utilizado também um *software* desenvolvido por Torsten Hoffmann [25]. Deste modo, foi utilizado o “SunCalc.org”, que é em tudo semelhante ao programa MoonCalc.org, sendo a diferença o facto de que este efetua o cálculo da posição do Sol, para um ponto na Terra.

O primeiro passo será novamente efetuar a escolha da data e coordenadas, para se efetuar a verificação do cálculo da posição do Sol:

- **Hora**: 05:54 UTC
- **Data:** 18/11/2020
- **Latitude**: 39.15645º
- **Longitude**: -8.04699º

Desta vez, foi escolhida uma data no ano de 2020 de modo a comprovar a capacidade do **ItTracker** em efetuar cálculos no futuro.

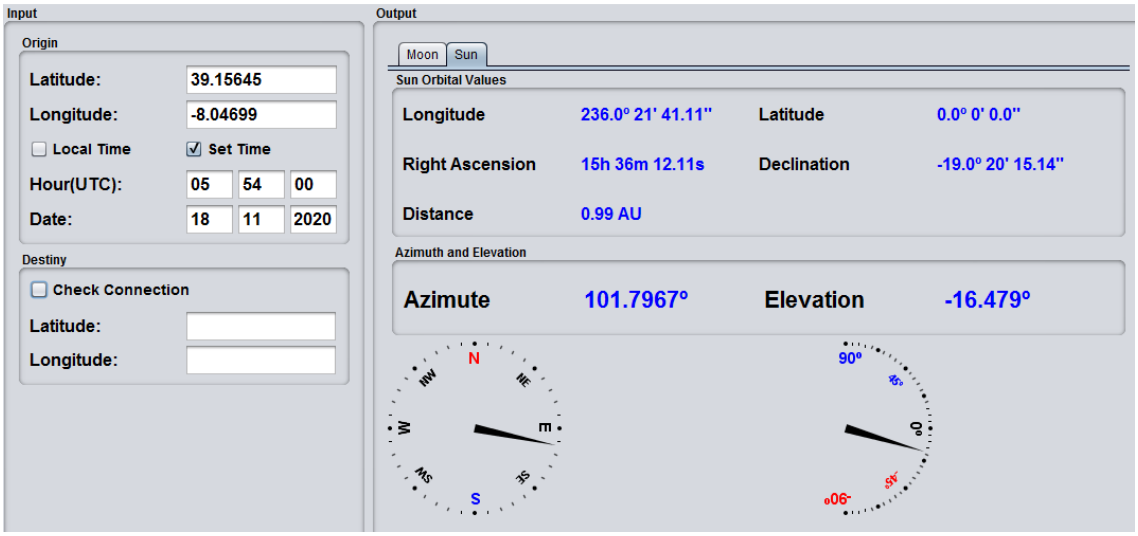


Figura 4.27: ItTracker resultados Sol

Obtiveram-se os seguintes resultados, que podem ser vistos na figura 4.26, para o **ItTracker**:

- **Azimute** = 101.80º
- **Elevação** = -16.48º

E os seguintes resultados para o **SunCalc.org**:

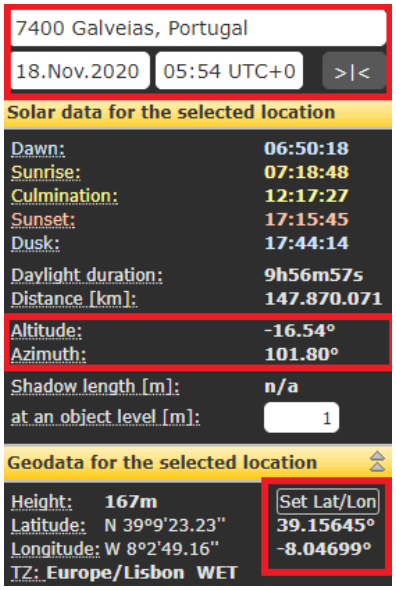


Figura 4.28: MoonCalc.org resultados Sol

- **Azimute** = 101.80º
- **Elevação** = -16.54º

Para o caso do cálculo da posição do Sol dada a localização de um observador na Terra, verifica-se que existe uma diferença de 0.06º para os valores de *Elevação* entre ambos os programas. Esta situação acaba por ser normal, dado que foram efetuadas muitas simplificações para o cálculo da posição do Sol.

Realça-se, novamente, o facto de o objetivo principal do **ItTracker** ser o de manter a margem de erro definida para o cálculo da posição da Lua a partir de uma localização na Terra, e não do Sol, visto que o *software* foi desenvolvido no âmbito de uma comunicação EME.

Neste capítulo não foram demonstrados os resultados para o caso do algoritmo "Check Connection" se encontrar no modo "Connection Possible". Contudo, este encontra-se a funcionar corretamente e como previsto, visto que a diferença entre estes estados (possível ou impossível), se baseia simplesmente no facto de uma das localizações não obter o valor de *Elevação* da Lua superior a 5º durante os cálculos efetuados.

Não foi demonstrado o facto de o **ItTracker** efetuar cálculos em tempo real (*checkbox* "Local Time" selecionada) dada a própria implicação deste tipo de ação (em tempo real). Contudo, estes também se encontram a funcionar corretamente dado que as diferenças não são em termos de algoritmos do cálculo orbital utilizados, mas sim em termos computacionais (Anexo A). As diferenças prendem-se com os factos de o algoritmo utilizar a hora definida no PC do utilizador, bem como de os cálculos orbitais serem calculados ao segundo, e autonomamente, até ordem de interrupção do utilizador visto este se encontrar a ser corrido por *threads* criadas em Java. No modo de cálculo em tempo real, é também gravado no ficheiro "input.dat" o resultado do valor de *Azimute* e *Elevação* da Lua para as coordenadas introduzidas e que permitirá uma interação em tempo real com diferentes dispositivos de controlo de posição (ex. antenas parabólicas).

Todo o código fonte referente ao software **ItTracker** pode ser visto em Aneco C para consulta ou testes futuros.

# Capítulo 5

# Conclusões

As comunicações rádio de longo alcance são uma parte fundamental no ramo das telecomunicações. Os satélites de comunicação artificiais são, nos dias que correm, a única forma de se conseguir efetuar uma telecomunicação "instantânea" quando nos encontramos em localizações remotas, onde a conexão por cabo ainda não se encontra distribuída entre diversos pontos do planeta.

O uso da Lua como repetidor/refletor passivo de sinal, foi o primeiro passo para o uso de satélites artificiais como meio fundamental de comunicação entre localizações distantes que não se encontram interligadas via cabo. Contudo, todas as complicações inerentes a uma comunicação EME fizeram com que esta entrasse em desuso, logo após o sucesso alcançado pelos primeiros satélites artificiais de telecomunicações.

Muito do trabalho desenvolvido ao longo dos anos na área das comunicações EME, após o aparecimento dos satélites artificiais, foi efetuado por rádio amadores e comunidades académicas que não abdicaram da noção de se conseguir utilizar o nosso, e único, satélite natural da Terra, como meio de reflexão de sinal, para se estabelecer uma comunicação via ondas rádio entre localizações distantes no nosso planeta - a Lua.

As novas capacidades em termos de modulação de sinal via rádio alcançadas na era digital, nomeadamente ao longo dos últimos anos, levam a que se testem novas soluções, novas ideias e novos conceitos, de modo a que se consigam criar novas oportunidades.

Durante a fase inicial do trabalho realizado para o desenvolvimento desta dissertação, foi também realizado o estudo relativo às noções, elementos e cálculos orbitais. A complexidade algorítmica inerente ao estudo realizado dos cálculos orbitais, foi aplicado na criação de um *software*, denominado ItTracker, que consegue localizar a posição da Lua, para um observador que se encontra localizado na Terra e dado um espaço temporal real ou definido pelo utilizador.

O programa tem a capacidade de conseguir comunicar com diferentes tipos de hardware, nomeadamente e no caso específico do contexto desta dissertação, controladores de antenas, visto que o resultado do cálculo orbital dos valores de Azimute e Elevação da Lua efetuado, é exportado para um ficheiro de simples leitura e interação, e que torna fácil a interligação entre diferentes dispositivos.

O programa ItTracker desenvolvido nesta dissertação apresenta uma série de funcionalidades que permitem determinar com sucesso, e em tempo real para qualquer utilizador na Terra, as coordenadas que permitem apontar uma antena à Lua, podendo-se igualmente determinar a posição do Sol a partir daquele ponto. Os resultados obtidos têm uma margem de erro inferior a 0.0333º, para o cálculo da órbita da Lua, e foram comparados com outros programas disponíveis que o atestam.

Este é um estudo e *software* que deverá ser continuado. Esta dissertação é apenas mais um contributo para compreender e implementar toda a complexidade necessária para se conseguir estabelecer uma comunicação via rádio entre duas localizações remotas, que utilize a Lua como refletor/repetidor passivo de sinal.

# Trabalho Futuro

Para trabalho futuro, recomenda-se que se adicionem novas funcionalidades e capacidades realizados pós estudo orbital. Por exemplo, adicionar ao programa a capacidade de conseguir localizar todos os planetas do sistema solar a partir de uma localização na Terra, em qualquer espaço de tempo. O passo seguinte, seria o de se efetuar a ligação do resultado exportado pelo ItTracker a um hardware que seja responsável pelo controlo de uma antena. Introduzir ao *software* a capacidade de se conseguir operar remotamente através de uma ligação à internet, que não necessite que o utilizador se encontre fisicamente localizado na estação de satélites. E por fim, estudar e implementar uma ligação na gama dos 10GHz, aplicando este conhecimento adquirido ao ItTracker, através da introdução da capacidade de este conseguir medir, calcular e prever todos os aspetos rádio necessários, e que estão implícitos, no estabelecimento com sucesso de uma comunicação via ondas rádio EME. Com o funcionamento da estação poderão ser efetuados estudos vários sobre o canal, atrasos de propagação e atenuações, entre outros, de forma a melhorar este tipo de ligações.

# Referências Bibliográficas

[1] E-zine, “Moonbounce, EME propagation - an overview, summary or tutorial about the basics of Moonbounce or EME radio signal propagation as it is used by radio hams or radio amateurs on the amateur radio bands.” 2009 Disponível na World Wide Web: http://www.electronics-radio.com/articles/ham_radio/amateur-propagation/moonbounce-propagation-eme.php à data de 21/10/2017

[2] ARRL, Chapter 30 “Earth-Moon-Earth (EME) Communication.”, “Handbook for Radio Communications”, edt. by J. Taylor, K1JT, United States of America, 2010.

[3] Daniel R. Headrick, Chapter 1 “Radio Versus Cable: International Telecommunications Before Satellites”, “Beyond the Ionosphere: The development of Satellite Communications”, edt. by Andrew J. Butrica, United States of America, 1997.

[4] David K. Van Keuren, Chapter 2 "Moon in Their Eyes: Moon Communication Relay at the Naval Research Laboratory, 1951-1962", “Beyond the Ionosphere: The development of Satellite Communications”, edt. by Andrew J. Butrica, United States of America, 1997.

[5] Herbert Kaufman, “A DX Record: To the Moon and Back,” QST 30, United States of America, pp.65-68, May 1946.

[6] “Project Moonbeam”, QST, United States of America, pp.11-12, March 1953.

[7] William ORR, W6SAI, “The Story of the First Successful Two-Way Amateur Communication via the Moon!”, QST, United States of America, September 1960.

[8] E-zine, St. Joe’s High School Radio Club, “Operation MOONBOUNCE” 1959-1964 Available: http://sjhrc.org/moonbounce.html à data de 24/07/2017.

[9] Utilização de Frequências pelos serviços de amador e amador por satélite, ICP-Anacom Autoridade Nacional de Comunicações, Anexo 6, pp. 219, 2011.

[10] Rex Moncur VK7MO and Alan Devlin VK3XPD, “10 GHz EME QSO with 64 cm (~2 ft) Dish and 8 Watts” Disponível na World Wide Web: http://www.vk3hz.net/microwave/10-GHz-EME-QSO-with-64-cm.pdf à data de 25/10/2017

[11] Material das aulas leccionadas pelo Professor José Sanguino, da cadeira de Sistemas de Posicionamento de Telecomunicações de Satélite, lecionada no IST Tagus Park, Satellite Ephemeris and Coordinates, 2013/2014.

[12] Sparknotes, Chapter 11.5, Kepler’s Law, “Sat Physics” Disponível na World Wide Web: http://www.sparknotes.com/testprep/books/sat2/physics/chapter11section5.rhtml à data de 25/10/2017

[13]. Flutuante, “Perceptos por perto afectos ao longe” 2008, disponível na World Wide Web: https://flutuante.files.wordpress.com/2008/12/afelio-perielio.gif à data de 21/10/2017

[14] “O ceú da minha terra” 2012, disponível na World Wide Web: http://www.asemana.publ.cv/IMG/jpg_1-153.jpg à data de 14/09/2016

[15] Cláudio B. Amador, “As estações do Ano X Periélio e Afélio”, 2009 disponível na World Wide Web: http://www.oocities.org/br/hpclaudioweb/astroerros.html à data de 21/10/2017

[16] Fernando Lang da Silveira, Res. Bra. Ensino Fis.vol. 23 “The variations within main phases of the Moon time intervals” 2001, disponível na World Wide Web: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1806-11172001000300008 à data de 22/10/2017

[17] Dr. T.S. Kelso, Part II, “Satellite Times” Orbital Coordinate System 1995, Available: https://celestrak.com/columns/v02n02/ à data de 25/10/2017

[18] Fiona Vincent, Chapter 5, “AS 2001 Positional Astronomy” Coordinates Systems: the first equatorial or “HA-Dec.” system 1998, Available: http://star-www.st-and.ac.uk/~fv/webnotes/index.html à data de 25/10/2017

[19] Fiona Vincent, Chapter 4, “AS 2001 Positional Astronomy” Coordinates Systems: the second equatorial or “RA-Dec. system” 1998, disponível em World Wide Web: http://star-www.st-and.ac.uk/~fv/webnotes/index.html à data de 25/10/2017

[20] Electrónica PT, “Como apontar uma antena parabólica” 2017, disponível em World Wide Web: https://www.electronica-pt.com/satelite-parabolicas/apontar-parabolica à data de 22/10/2017

[21] Material da cadeira de Sistemas de Comunicação Digital por Satélite, orientada pelo Professor Francisco Cercas do Mestrado em Engenharia de Telecomunicações e Informática no ISCTE-IUL 2013/2014

[22] Jean H. Meeus, “Astronomical Algorithms”, pub. Willmann-Bell, Incorporated 1991.

[23] Paul Schlyter, “How to compute planetary positions” 1997, Disponível em World Wide Web: http://stjarnhimlen.se/comp/ppcomp.html à data de 25/10/2017

[24] “MoonCalc.org” 2015-2017, disponível em World Wide Web: https://www.mooncalc.org à data de 28/10/2017

[25] “SunCalc.org” 2015-2017, disponível em World Wide Web: https://www.suncalc.org à data de 28/10/2017

# Anexos

# Anexo A Funções – Normalizações e Conversões

Pode ser visto de seguida, as funções computacionais criadas em Java e consideradas mais relevantes para o s*oftware* ItTracker. As restantes funções e métodos de conversões (ex. de graus para radianos, etc.), podem ser vistas em Anexo C, dado que todo o código se encontra devidamente comentado para utilização futura.

- Função norma - Normalização entre 0º e 360º

A função **norma**, da classe Java "MathFormulas.Java", tem como parâmetro de entrada/saída um valor do tipo *double*, e é responsável pela normalização de um valor entre 0º e 360º. Se o valor for negativo, é retirada a parte inteira da divisão do valor por 360 e soma-se 1 à parte inteira. Seguidamente, soma-se o valor negativo à multiplicação da parte inteira a multiplicar por 360.

Se o valor for positivo e maior que 360, retira-se a parte inteira da divisão do valor por 360, e depois é efetuado a subtração do valor pela multiplicação da parte inteira, e multiplica-se por 360. Este método garante a normalização de qualquer valor negativo ou positivo entre 0 e 360.

```
public double norma(double a){
      int b = 0;
      if (a<0){
            b = (int) (-a/360);
            b = b + 1;
            a =  a + b*360;
      }else if(a>360){
            b = (int) (a/360);
            a = a - b*360;
      }return a;
}
```

- Função HA

O código que pode ser visto de seguida, da classe Java "moon.java", garante que o valor do HA é mantido na sua forma mais comum de utilização quando se pretende trabalhar em graus, ou seja entre -180º e 180º

```
        if(HA>180)
           HA = HA - 360;
         else if(HA<-180)
           HA = HA + 360;
```

- Função - Graus, Minutos, Segundos

A função ***DegreesToHours***, da classe Java "MathFormulas.Java", têm como parâmetro de entrada um valor *double* de um angulo e retornando o valor no tipo *string*. Esta *string* é o resultado da conversão de um angulo em graus, para o formato Graus(º), Minutos (´), Segundos (´´).

```
public String DegreesToHours(double angle){

      String s="";
            double degrees = (int) angle;
            double minutes = Math.abs(angle) - Math.abs(degrees);
                   minutes = minutes *60;
            double seconds = Math.abs(minutes - (int)minutes);
                   seconds = seconds*60;

            if(angle<0 && angle >-1)
            s = "-"+(int)angle;
            else s=""+degrees;

            return s+"° "+(int)minutes+"' "+(Math.round(seconds * 100.0) /
100.0)+"''";
      }
```

- Função DegreesToHours1 - Minutos, Segundos

A função **DegreesToHours**1, têm como parâmetro de entrada um valor *double* de um angulo e retornando o valor no tipo *string*. Esta *string* é o resultado da conversão de um angulo em graus para Minutos(´), Segundos(´´).

```
public String DegreesToHours1(double angle){


            double minutes = angle *60;

            double seconds = Math.abs(minutes - (int)minutes);
                   seconds = (seconds*60);

            return (int)minutes+"' "+(Math.round(seconds * 100.0) /
100.0)+"''";
      }
```

- Função DegreesToHours2 - Horas, Minutos, Segundos

A função **DegreesToHours2,** da classe Java "MathFormulas.Java", têm como parâmetro de entrada um valor *double* de um angulo e retornando o valor no tipo *string*. Esta *string* é o resultado da conversão de um angulo em graus para o formato de tempo em Horas(h), Minutos(m), Segundos(s).

```
public String DegreesToHours2(double angle){
            double aux=0;
            aux=angle/15;
            int   horas = (int) aux;

            double minutes =Math.abs(aux - horas);
                   minutes = minutes *60;
            double seconds = Math.abs(minutes - (int)minutes);
                   seconds = (seconds*60);

            return (int)horas+"h "+(int)minutes+"m "+(Math.round(seconds *
100.0) / 100.0)+"s'";
}
```

- Funções de conversão e arredondamento

A funções que podem ser vistas de seguida, são responsáveis pelo arredondamento a 4 casas decimais dos valores calculados, conversão de um valor de radianos para graus e conversão de graus para radianos, respetivamente.

```
/Arrendonda a 4 Casas decimais

      public double arredonda(double a){
            return (Math.round(a * 10000.0) / 10000.0) ;
      }

      double convR = (PI/180); //Conversor para Radianos
      double convD = (180/PI); //Conversor para Graus
```

- Função UTConv

A função **UTCconv,** da classe "Date.java", tem como parâmetros de entrada a hora, minutos e segundos, e esta função retorna o valor unitário das horas inseridas no campo de *input* pelo utilizador.

```
  public double UTCconv(double hour, double minute, double second){

            return hour + (double)minute/60 + (double)second/3600;        }
```

- Função getUTCtime

A função **getUTCtime,** da classe "Date.java", funciona da mesma forma que a função anterior, contudo esta é utilizada quando o valor que se quer converter são as horas atuais e definidas no PC responsável pela execução do ItTracker.

```
public double getUTCtime(){
      return (getHour() + (double)getMinute()/60 +
(double)getSecond()/3600;
     }
```

Os métodos getHour(), getMinute() e getSecond() servem para captar a hora/minutos/segundos locais do Pc, e fazem também parte da classe "*Date.java*".

É subtraído o valor unitário de 1 ao valor das horas, visto para que o valor das horas se mantenha no fuso horário UTC para um fuso horário UTC+1. Caso se mude de fuso horário, deverá ter-se em consideração que terá que ser ajustado este valor para correto funcionamento do ItTracker.

- Função *Check Connection*

A função **Check Connection**, da classe "GUI.Java" serve para que quando a opção da checkbox "check connection" é selecionada, esta verifica se ambos os valores de coordenadas introduzidos se encontram com a Lua em linha de vista. Como referido no capítulo 4, o resultado é possível para um angulo de visão da lua superior a 5º (ou seja, elevação >5º em ambos).Resultado não possível dá-se caso um dos valores de elevação se encontre abaixo de 5º.

```
            if(cc.isSelected() == true && (m.getAltitudeCorrected() > 5.0
&& m1.getAltitudeCorrected() > 5.0)){
                  check1.setText("Conection Possible");
                  check1.setForeground(Color.blue);

            }else if(cc.isSelected() == true && (m.getAltitudeCorrected() <
5.0 || m1.getAltitudeCorrected() < 5.0)){
                  check1.setText("Conection Impossible");
                  check1.setForeground(Color.red);
            }}
```

- Função *WriteFile*

Esta parte do código, da classe Java "WriteFile.Java", permite que um ficheiro do tipo "*.dat" seja criado quando se efetua o calculo em tempo real dos algoritmos. O ficheiro criado pode ser visto no diretório principal das classes java, sempre que este é criado ou no diretório aonde se encontra o ficheiro executável do java "ItTracker.jar".

```
public static void Run(double Azimute, double Elevation) throws IOException
{    File fileOne = new File("input.dat");
    fileOne.delete();
    File newFile = new File("input.dat");
    newFile.createNewFile();

      PrintWriter writer = new PrintWriter(new
      FileWriter("input.dat"));
      writer.print("1 "+Azimute+" "+Elevation);
      writer.close();
      }
public static void write(double Azimute, double Elevation){

               try {
                        Run(Azimute,Elevation);
               } catch (IOException e) {
                        System.out.println("Erro ao Escrever no ficheiro");
      e.printStackTrace();    }  }
```

# Anexo B Manual de Instruções

Se o utilizador desejar executar o ItTracker sem o auxílio do *software* Eclipse, basta que este execute o ficheiro "ItTracker.Jar". O utilizador terá que executar o *software* através do Sistema Operativo Windows, e terá que ter o Java Runtime Enviroment atualizado no seu computador. O Java JDK8 e Java Runtime Enviroment, podem ser descarregados de forma gratuita através dos websites oficiais da Oracle:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

O *software* compilador Ecplise, pode ser também descarregado de forma gratuita no website oficial:

http://www.eclipse.org/downloads/eclipse-packages/

# Anexo C Listagem de código fonte

Moon.Java

```
import java.util.Calendar;

public class Moon {


	private double N;                //Long da Node Asc em graus
	private double i = 5.1454;       //Inclinação em graus
	private double w;                //Arg. do perigeo em graus
	private double a = 60.2666;      //Distância Média
	private double e = 0.054900;     //Exentricidade
	private double M;                //Anómalia média 0 perigeo e 180 apogeu
	private double ecl;              //Obliquidade da ecliptica
	private double OP = 27.322;      //Periodo Orbital m.arredondado à centena


	private double r;                //Distância
	private double v;                //Anomalia verdadeira

	private double E;                //Anomalia Excentrica

	private double RA;               //Right Ascension
	private double Decl;             //Declination
	private double topRA;            //Topocentric Right Ascension
	private double topDecl;          //Topocentric Declination

	private double azimute;          //Azimute face a uma coordenada local
	private double elevacao;         //Altitude face a uma coordenada local
	private double elevacaocorrigida;//Altitude corrigida

	private double latM;             //Latitude da Lua +90º e -90º
	private double longiM;           //Longitude da Lua 0º a 360º

	private double latitude;         //Latitude Local -90º(S) a 90º(N)
	private double longitude;        //Longitude Local -180º(O) a 180º(E)

	private double xrect;            //Coordenada Rectangular X
	private double yrect;            //Coordenada Rectangular Y
	private double zrect;            //Coordenada Rectangular Z

	private double xrect1;           //Coordenada Rectangular X
	private double yrect1;           //Coordenada Rectangular Y
	private double zrect1;           //Coordenada Rectangular Z

	private double xhor;             //Coordenada Rectangular X
	private double yhor;             //Coordenada Rectangular Y
	private double zhor;             //Coordenada Rectangular Z

	private double xgeo;             //Coordenada Geocêntrica X
	private double ygeo;             //Coordenada Geocêntrica Y
	private double zgeo;             //Coordenada Geocêntrica Z
```

```
	private double xeclip;                  //Coordenada Ecliptica X
	private double yeclip;                  //Coordenada Ecliptica Y
	private double zeclip;                  //Coordenada Ecliptica Z

	private double xequat;                  //Coordenada Equatorial X
	private double yequat;                  //Coordenada Equatorial Y
	private double zequat;                  //Coordenada Equatorial Z

	private double Lm;                      //Distância média da lua
	private double D;                       //Elongation média da lua
	private double F;                       //Argumento de latitude da lua
	private double Ms;                      //Anómalia média do Sol

	private double UT;                      //Universal Time
	private double GMST0;                   //Sidime no meridiano de Greenwich
	private double SidTime;                 //Local Sidereal Time
	private double LST;                     //Conversão de SidTime para graus
	private double HA;                      //Hour Angle (GMST)
	private double latgeo;                  //Latitude Geocêntrica
	private double rgeo;                    //Distância do centro da Terra à Lua
	private double g;                       //Angulo Auxiliar
	private double mpar;                    //Moon's Parallax

	private String coord;

	private MathFormulas m = new MathFormulas();

	private Sun sun = new Sun();
	private Earth earth = new Earth();

	public void CalcMoon(double day, double UT, double latitude, double
longitude){

		sun.CalcSun(day, UT, latitude, longitude);

 //Calculo Elementos orbitais

			N = m.norma(125.1228 - 0.0529538083 * day);
			w = m.norma(318.0634 + 0.1643573223 * day);
			M = m.norma(115.3654 + 13.0649929509 * day);

			Lm = m.norma(N+w+M);
			F =  m.norma(Lm - N);
			D =  m.norma(Lm - sun.getSunL());
			Ms = m.norma(sun.getSunM());
			ecl = 23.4393 - 3.563*Math.pow(10, -7) * day;
			E = m.calcE(M, e);

//Calculo Coordenadas Rectangulares no plano orbital da Lua
			xrect = a * (m.cos(E) - e);
			yrect = a * (m.sqrt(1-e*e) * m.sin(E));

//Calculo Distância e Verdadeira Anomalia da Lua
			r = m.calcDistance(xrect, yrect);
			v =m.calcTAnomaly(xrect, yrect);

//Calculo Coordenadas Eclipticas da Lua
			 xeclip = r * (m.cos(N) * m.cos((v+w)) - m.sin(N) * m.sin((v+w)) *
m.cos(i));
```

```
           yeclip = r * (m.sin(N) * m.cos((v+w)) + m.cos(N) * m.sin((v+w)) *
m.cos(i));
           zeclip = r * m.sin((v+w)) * m.sin(i);

//Calculo Latitude e Longitude
           longiM = m.norma((Math.atan2(yeclip, xeclip)*m.convD) + longP());
           latM =  Math.atan2(zeclip, m.sqrt(xeclip*xeclip+yeclip*yeclip))*m.convD
+ latP();

//Conversão de Long e Lat para Coordenadas rectangulares x,y,z Geocêntricas
           xgeo = m.calcEclCoordX(r, longiM,  latM);
           ygeo = m.calcEclCoordY(r, longiM, latM);
           zgeo = m.calcEclCoordZ(r, latM);

 //Conversão de Coordenadas Rectangulares Geocêntricas para Coordenadas
Equatoriais Right Ascension e Declination
          xequat= xgeo;
           yequat = m.calcEquaCoordY(ecl, ygeo, zgeo);
           zequat = m.calcEquaCoordZ(ecl, ygeo, zgeo);

           RA   = m.norma(m.atan2( yequat, xequat ) * m.convD);
           Decl = m.atan2( zequat, m.sqrt( xequat * xequat + yequat * yequat ) ) *
m.convD;


//Sideral Time e Hour Angle (GMST)
          GMST0 = sun.getSunL()/15 + 12;
          SidTime = GMST0 + UT + longitude/15;
          LST = m.norma(SidTime*15); //Para Graus multiplica por 15
          HA = m.norma(LST - RA);    //Valor entre -180 e 180

          if(HA>180)
             HA = HA - 360;
           else if(HA<-180)
             HA = HA + 360;

//Moon Paralax  ou Horizontal Paralax
           mpar = Math.sin(1/r)*m.convD;

//Flatening da Terra
           latgeo = latitude - 0.1924 * m.sin(2*latitude);

           rgeo = 0.99833 + 0.00167 * m.cos(2*latitude);

//Topocentric RA e Decl
           g = Math.atan( m.tan(latgeo) / m.cos(HA))*m.convD;
           g = Math.round(g * 1000.0) / 1000.0;
           topRA = RA  - mpar * rgeo * m.cos(latgeo) * m.sin(HA) / m.cos(Decl);
//Formula pode falhar se Decl = 0 Lua perto dos polos

           if(latgeo!=0)
           topDecl = Decl - mpar * rgeo * m.sin(latgeo) * m.sin(g - Decl) /
m.sin(g);
           else topDecl = Decl - mpar * rgeo * m.sin(-Decl) * m.cos(HA);

//Geocentric Altitude e Azimute
           xrect1 = m.cos(HA) * m.cos(Decl);
           yrect1 = m.sin(HA) * m.cos(Decl);
           zrect1 = m.sin(Decl);
```

```
            xhor = xrect1 * m.sin(latitude) - zrect1 * m.cos(latitude);
            yhor = yrect1;
            zhor = xrect1 * m.cos(latitude) + zrect1 * m.sin(latitude);
            azimute  = m.atan2( yhor, xhor )*m.convD + 180 ; //+180 para ser lido
como 0ºN 90ºE 180ºS 270ªO
            elevacao = m.atan2( zhor, m.sqrt(xhor*xhor+yhor*yhor) ) * m.convD;
            elevacaocorrigida = elevacao - mpar * m.cos(elevacao);  //Correção de
Altitude

}

        public String getCord(){
                return coord;
        }

//Perturbações na Lua

        public double longP(){
          return -1.274 * m.sin(M - 2*D)
                  +0.658 * m.sin(2*D)
                  -0.186 * m.sin(Ms)
                  -0.059 * m.sin(2*M - 2*D)
                  -0.057 * m.sin(M - 2*D + Ms)
                  +0.053 * m.sin(M + 2*D)
                  +0.046 * m.sin(2*D - Ms)
                  +0.041 * m.sin(M - Ms)
                  -0.035 * m.sin(D)
                  -0.031 * m.sin(M + Ms)
                  -0.015 * m.sin(2*F - 2*D)
                  +0.011 * m.sin(M - 4*D);
        }
        public double latP(){
                return -0.173 * m.sin(F - 2*D)
                          -0.055 * m.sin(M - F - 2*D)
                          -0.046 * m.sin(M + F - 2*D)
                          +0.033 * m.sin(F + 2*D)
                          +0.017 * m.sin(2*M + F);
        }
        public double distanceP(){
                return  -0.58 * m.cos(M - 2*D)
                              -0.46 * m.cos(2*D);
        }

//Métodos para retirar valores os valores das variáveis m.arredondados (para
Display)
        public double getN(){
        return m.arredonda(N);
        }
        public double geti(){
                return m.arredonda(i);
        }
        public double getw(){
                return m.arredonda(w);
        }
        public double geta(){
                return m.arredonda(a);
```

```
}
public double gete(){
	return m.arredonda(e);
}
public double getM(){
	return m.arredonda(M);
}
public double getOP(){
	return OP;
}
public double getE(){
	return m.arredonda(E);
}
public double getDistance(){ //Adicionar altura?
	return (Math.round(r*earth.getEarthRad() * 100.0) / 100.0);
}
public double getV(){
	return m.arredonda(v);
}
public double getXrect(){
  return m.arredonda(xrect);
}
public double getYrect(){
	return m.arredonda(yrect);
}
public double getXeclip(){
	return m.arredonda(xeclip);
}
public double getYeclip(){
  return m.arredonda(yeclip);
}
public double getZeclip(){
	return m.arredonda(zeclip);
}
public double getXgeo(){
	return m.arredonda(xgeo);
}
public double getYgeo(){
  return m.arredonda(ygeo);
}
public double getZgeo(){
	return m.arredonda(zgeo);
}
public double getXequat(){
	return m.arredonda(xequat);
}
public double getYequat(){
  return m.arredonda(yequat);
}
public double getZequat(){
	return m.arredonda(zequat);
}
public double getlongiV(){
	return m.arredonda(longiM);
}
public double getlatV(){
	return m.arredonda(latM);
}
public String getlongi(){
```

```
            return m.DegreesToHours(longiM);
      }
      public String getlat(){
            return m.DegreesToHours(latM);
      }
      public String getRA(){
            return m.DegreesToHours2(RA);
      }
      public String getDecl(){
            return m.DegreesToHours(Decl);
      }
      public String getMPar(){

            return m.DegreesToHours1(mpar);
      }
      public double getHA(){
            return m.arredonda(HA);
      }
      public String gettopRA(){
            return m.DegreesToHours2(topRA);
      }
      public String gettopDecl(){
            return m.DegreesToHours(topDecl);
      }
      public double getAzimute(){
            return m.arredonda(azimute);
      }
      public double getAltitude(){
            return m.arredonda(elevacao);
      }
      public double getAltitudeCorrected(){
            return m.arredonda(elevacaocorrigida);
      }
}
```

Sun.Java

```
public class Sun {

      private int N = 0;               //Long da Node Asc em graus
      private int i = 0;               //Inclinação em graus
      private double w ;               //Arg. do perigeo em graus
      private double a = 1.00000;      //Semi eixo maior/Distância Média (AU)
      private double e ;               //Exentricidade
      private double M ;         //Anómalia média 0 perélio and 180 afélio
      private double oblecl;            //Obliquidade da Ecliptica
      private double E;
      private double q;
      private double Q;
       private double azimute;
       private double altitude;
      private double xrect;
      private double yrect;
      private double zrect;

      private double r;
      private double v;
      private double longitudeS;
      private int latitudeS;
```

```
        private double xhor;
        private double yhor;
        private double zhor;
        private double xgeo;
        private double ygeo;
        private double zgeo;
        private double xequat;
        private double yequat;
        private double zequat;
        private double RA;
        private double Decl;
        private double GMST0;
        private double SidTime;
        private double LST;
        private double HA;
        private String coord;
        private MathFormulas m = new MathFormulas();

        public Sun(){ }

        public void CalcSun(double day, double UT, double latitude, double
longitude){

//Calculo de Elementos Orbitais do Sol
            w = 282.9404 + 4.70935 * Math.pow(10,  -5) * day;
            M = m.norma(356.0470 + 0.9856002585 * day);
            e = 0.016709 - 1.15 * Math.pow(10, -9) * day;
            oblecl = 23.4393 - 3.56*Math.pow(10, -7) * day;
            E = m.calcE(M, e);
            q  = a * (1 - e);
            Q  = a * (1 + e);

//Calculo Coordenadas Rectangulares no plano orbital da Ecliptica
            xrect = (m.cos(E) - e);
            yrect = (m.sqrt(1-e*e) * m.sin(E));

//Calculo Distância e Verdadeira Anomalia do Sol
            r = m.calcDistance(xrect, yrect);
            v =m.calcTAnomaly(xrect, yrect);

//Calculo Longitude e Latitude
             longitudeS = m.norma(v+w);
             latitudeS = 0;

//Calculo RA e DEcl
             xgeo = m.calcEclCoordX(r, longitudeS,  latitudeS);
             ygeo = m.calcEclCoordY(r, longitudeS, latitudeS);
             zgeo = m.calcEclCoordZ(r, latitudeS);

             xequat = xgeo;
             yequat = m.calcEquaCoordY(oblecl, ygeo, zgeo);
             zequat = m.calcEquaCoordZ(oblecl, ygeo, zgeo);

             RA   = m.norma(m.atan2( yequat, xequat ) * m.convD);
             Decl = m.atan2( zequat, m.sqrt( xequat * xequat + yequat * yequat )
) * m.convD;


//Sideral Time e Hour Angle
```

```
                GMST0 = getSunL()/15 + 12;
                SidTime = GMST0 + UT + longitude/15;
                LST = m.norma(SidTime*15);
                HA = m.norma(LST - RA);
                    //Altitude e Azimute
                xrect = m.cos(HA) * m.cos(Decl);
                yrect = m.sin(HA) * m.cos(Decl);
                zrect = m.sin(Decl);

                xhor = xrect * m.sin(latitude) - zrect * m.cos(latitude);
                yhor = yrect;
                zhor = xrect * m.cos(latitude) + zrect * m.sin(latitude);

             azimute  = m.atan2( yhor, xhor )*m.convD + 180 ; //+180 para ser lido
como 0ºN 90ºE 180ºS 270ªO
                altitude = m.atan2( zhor, m.sqrt(xhor*xhor+yhor*yhor) ) * m.convD;


      }

      public String getCord(){
            return coord;}

      public double getSunW(){
            return m.norma(w);}

      public double getSunM(){
            return m.norma(M);}
      public double getSunL(){
            return m.norma(w+M);}
      private double getSunoblecl(){
            return m.norma(oblecl);}
      private double getSune(){
            return m.norma(e);}
      public double getDistance(){
            return (Math.round(r * 100.0) / 100.0);}
      public String getlongi(){
            return m.DegreesToHours(longitudeS);}
      public String getlat(){
            return m.DegreesToHours(latitudeS);}
      public String getRA(){
                  return m.DegreesToHours2(RA);}
      public String getDecl(){
            return m.DegreesToHours(Decl);}
      public double getAzimute(){
            return m.arredonda(azimute);}
      public double getAltitude(){
            return m.arredonda(altitude);}
      public double getPerihelion(){
             return q;    }
    public double getAphelion(){
      return Q;    }
}
```

GUI.Java

```
import java.awt.BorderLayout;
import java.awt.Color;
```

```
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;
import javax.swing.JTextField;
import javax.swing.border.TitledBorder;
import javax.swing.event.ChangeEvent;
//Coordenadas 38.73753,-9.13863  Lisboa
public class GUI extends JComponent implements ActionListener {

	private int hour;
	private int minute;
	private int second;
	private int day;
	private int month;
	private int year;
	private double UT;
	private double latitude;
	private double longitude;
	private double latitudeC;
	private double longitudeC;
	private  JLabel dataShow = new JLabel();
	private  JLabel horaShow = new JLabel();
	private JLabel latitudeM = new JLabel();
	private JLabel longitudeM = new JLabel();
	private JLabel distanciaM = new JLabel();
	private JLabel RAM = new JLabel();
	private JLabel topRA1 = new JLabel();
	private JLabel topDecl1 = new JLabel();
	private JLabel DeclM = new JLabel();
	private JLabel MparM = new JLabel();
	private JLabel azimuteM = new JLabel();
	private JLabel altitudeM = new JLabel();
	private JLabel riseM = new JLabel();
	private JLabel setM = new JLabel();
	private JLabel check1 = new JLabel();
	private JLabel latitudeS = new JLabel();
	private JLabel distanciaS = new JLabel();
	private JLabel RAS = new JLabel();
	private JLabel DeclS = new JLabel();
	private JLabel azimuteS = new JLabel();
	private JLabel altitudeS = new JLabel();
	private JLabel riseS = new JLabel();
	private JLabel setS = new JLabel();
	private  JTextField latiI = new JTextField("");
	private  JTextField longI = new JTextField("");
```

```
        private  JTextField latiC = new JTextField("");
        private  JTextField longiC = new JTextField("");
       private  JTextField hourI = new JTextField("00");
        private  JTextField minuteI = new JTextField("00");
        private  JTextField secondI = new JTextField("00");
        private  JTextField dayI = new JTextField("DD");
        private  JTextField monthI = new JTextField("MM");
        private  JTextField yearI = new JTextField("YYYY");
        JLabel longitudeS = new JLabel();
        private  JFrame janela = new JFrame("ItTracker");
        private  JPanel gui = new JPanel(new BorderLayout(0,0));
       private  JCheckBox rt = new JCheckBox("Local Time", true);
       private  JCheckBox st = new JCheckBox("Set Time", false);
       private  JCheckBox cc = new JCheckBox("Check Connection", false);
        private  JButton buttonCalc = new JButton("Calculate");
        private  JButton buttonStop = new JButton("Stop");

        private  Moon m = new Moon();
        private  Moon m1 = new Moon();
        private  Sun s = new Sun();

        private Compass c = new Compass();
        private Compass c1 = new Compass();
        private Elevation e = new Elevation();
        private Elevation e1 = new Elevation();
        private WorldMap w = new WorldMap();
        private JD j;
        private final Date d= new Date();
        private final sentinela sen = new sentinela();
        private final sentinelaCB cb = new sentinelaCB();
        private boolean lock=false;
         Font font1 = new Font("Arial", Font.BOLD, 16);
        Font font = new Font("Arial", Font.BOLD, 14);
        Font font2 = new Font("Arial", Font.BOLD, 22);
//Contrutor GUI
public GUI() {

         javax.swing.Timer timer = new javax.swing.Timer(1000,this);
         timer.start();
         latiC.setEditable(false);
              longiC.setEditable(false);
              setNEditable();
}
//Valores de Input
public JPanel InputValues(){

          JPanel input = new JPanel();
          input.setLayout(new BoxLayout(input, BoxLayout.Y_AXIS));
          input.setBorder(new TitledBorder("Input"));
          JPanel insertValues = new JPanel(new FlowLayout(FlowLayout.LEFT,0,0));
          insertValues.setBorder(new TitledBorder("Origin"));

        final JPanel insert = new JPanel(new GridLayout(0,2,0,2));

          JLabel latiin = new JLabel("Latitude:");
          JLabel longiin = new JLabel("Longitude:");
          JLabel datain = new JLabel("Date:");
          JLabel horain = new JLabel("Hour(UTC):");
```

```
        latiin.setFont(font1);
        latiI.setFont(font1);
        longiin.setFont(font1);
        longI.setFont(font1);
        horain.setFont(font1);
        datain.setFont(font1);
        rt.setFont(font);
        st.setFont(font);
        buttonCalc.setFont(font1);
        buttonStop.setFont(font1);
        buttonCalc.addActionListener(sen);
        buttonStop.addActionListener(sen);
        rt.addActionListener(cb);
        st.addActionListener(cb);
        insert.add(latiin);
        insert.add(latiI);
        insert.add(longiin);
        insert.add(longI);
        insert.add(rt);
        insert.add(st);
        insert.add(horain);
        insert.add(hourIn());
        insert.add(datain);
        insert.add(dateIn());
        insertValues.add(insert);
        JPanel insertValues1 = new JPanel(new BorderLayout(0,0));
        insertValues1.add(checkCalc(), BorderLayout.NORTH);
        insertValues1.add(buttonsCalc(), BorderLayout.SOUTH);
        input.add(insertValues, BorderLayout.NORTH);
        input.add(insertValues1, BorderLayout.SOUTH);
        return input;
}

//Painel Check Destiny Input
public JPanel checkCalc(){

             JPanel value = new JPanel(new BorderLayout(0,5));
             value.setBorder(new TitledBorder("Destiny"));
             JPanel checkcalc = new JPanel(new GridLayout(2,2,1,1));
             JLabel latiin = new JLabel("Latitude:");
             JLabel longiin = new JLabel("Longitude:");
             cc.setFont(font);
             latiin.setFont(font1);
             latiC.setFont(font1);
             longiin.setFont(font1);
             longiC.setFont(font1);
             checkcalc.add(latiin);
             checkcalc.add(latiC);
             checkcalc.add(longiin);
             checkcalc.add(longiC);
             cc.addActionListener(cb);
             value.add(cc, BorderLayout.NORTH);
             value.add(checkcalc, BorderLayout.SOUTH);

      return value;
}

//Adiciona Botões
public JPanel buttonsCalc(){
```

```
                JPanel buttons = new JPanel(new GridLayout(1,2,1,1));
                buttons.add(buttonCalc);
                buttons.add(buttonStop);

        return buttons;}
    //Painel Hora de Input
    public JPanel hourIn(){

        JPanel hour = new JPanel(new GridLayout(1,3,1,1));
        hourI.setFont(font1);
        minuteI.setFont(font1);
        secondI.setFont(font1);
        JLabel l = new JLabel(":");
        JLabel l1 = new JLabel(":");
         l.setFont(font1);
        l1.setFont(font1);
        hour.add(hourI);
        hour.add(minuteI);
        hour.add(secondI);
       return hour;}
    /Painel data Input
    public JPanel dateIn(){

        JPanel date = new JPanel(new GridLayout(1,5,1,1));
        JLabel l = new JLabel("/");
        JLabel l1 = new JLabel("/");
        l.setFont(font1);
        l1.setFont(font1);
        dayI.setFont(font1);
        monthI.setFont(font1);
        yearI.setFont(font1);
        date.add(dayI);
        date.add(monthI);
        date.add(yearI);

        return date;}

    //Hora e Data Local
    public JPanel TimeDate1(){

         JPanel timedate = new JPanel(new FlowLayout(FlowLayout.CENTER,20,0));
         timedate.setBorder(new TitledBorder("Local Time and Date"));
         JLabel data = new JLabel("Date:");
         JLabel hora = new JLabel("Hour:");
         dataShow.setText(d.toStringDay());
          horaShow.setText(d.toStringHour());
         hora.setFont(font1);
         horaShow.setFont(font1);
         data.setFont(font1);
         dataShow.setFont(font1);
         timedate.add(hora);
         timedate.add(horaShow);
         timedate.add(data);
         timedate.add(dataShow);
         return timedate;
     }
    //Output LUA
    public JPanel outputMoon() {
```

```
JPanel outputMoon = new JPanel();
outputMoon.setLayout(new BoxLayout(outputMoon, BoxLayout.Y_AXIS));
final JPanel see = new JPanel(new GridLayout(0,4,0,20));
see.setBorder(new TitledBorder("Moon Orbital Values"));

final JPanel seeMain = new JPanel(new GridLayout(0,4,10,0));
seeMain.setBorder(new TitledBorder("Azimuth and Elevation"));
outputMoon.setFont(font1);

JLabel azimute = new JLabel("Azimute");
JLabel altitude = new JLabel("Elevation");
JLabel lati = new JLabel("Latitude");
JLabel longi = new JLabel("Longitude");
JLabel distancia = new JLabel("Distance");
JLabel RA = new JLabel("Right Ascension");
JLabel Decl = new JLabel("Declination");
JLabel Mpar = new JLabel("Horizontal Parallax");
JLabel Rise = new JLabel("Rising Time");
JLabel Set = new JLabel("Set Time");
JLabel topRA = new JLabel("Top. Right Ascension ");
JLabel topDecl = new JLabel("Top. Declination");
lati.setFont(font1);
longi.setFont(font1);
distancia.setFont(font1);
RA.setFont(font1);
Decl.setFont(font1);
Mpar.setFont(font1);
topRA.setFont(font1);
topDecl.setFont(font1);
latitudeM.setFont(font1);
longitudeM.setFont(font1);
distanciaM.setFont(font1);
RAM.setFont(font1);
DeclM.setFont(font1);
MparM.setFont(font1);
Rise.setFont(font1);
Set.setFont(font1);
riseM.setFont(font1);
setM.setFont(font1);
topRA1.setFont(font1);
topDecl1.setFont(font1);
see.add(longi);
see.add(longitudeM);
see.add(lati);
see.add(latitudeM);
see.add(RA);
see.add(RAM);
see.add(Decl);
see.add(DeclM);
see.add(topRA);
see.add(topRA1);
see.add(topDecl);
see.add(topDecl1);
see.add(Mpar);
see.add(MparM);
see.add(distancia);
see.add(distanciaM);
//Azimute e elevacao
             azimuteM.setFont(font2);
```

```
altitudeM.setFont(font2);
azimute.setFont(font2);
altitude.setFont(font2);
check1.setFont(font2);
seeMain.add(azimute);
seeMain.add(azimuteM);
seeMain.add(altitude);
seeMain.add(altitudeM);
outputMoon.add(see, BorderLayout.NORTH);
outputMoon.add(seeMain, BorderLayout.SOUTH);
JPanel check = new JPanel();
check.add(check1);
JPanel graph = new JPanel(new GridLayout(0,2,0,0));
e.setPreferredSize(new Dimension(140,140));
e.setOpaque(false);
c.setPreferredSize(new Dimension(140,140));
c.setOpaque(false);
graph.add(c,new FlowLayout(FlowLayout.CENTER));
graph.add(e);
outputMoon.add(graph);
outputMoon.add(check);
return outputMoon ;  }

//OUTPUT SUN
public JPanel outputSun(){
    JPanel outputSun = new JPanel();
    outputSun.setLayout(new BoxLayout(outputSun, BoxLayout.Y_AXIS));

    final JPanel seeOrbital = new JPanel(new GridLayout(0,4,0,20));
    seeOrbital.setBorder(new TitledBorder("Sun Orbital Values"));

    final JPanel seeAE = new JPanel(new GridLayout(0,4,10,0));
    seeAE.setBorder(new TitledBorder("Azimuth and Elevation"));

              JLabel azimute = new JLabel("Azimute");
                   JLabel altitude = new JLabel("Elevation");
                   JLabel lati = new JLabel("Latitude");
                   JLabel longi = new JLabel("Longitude");
                   JLabel distancia = new JLabel("Distance");
                   JLabel RA = new JLabel("Right Ascension ");
                   JLabel Decl = new JLabel("Declination");

                latitudeS.setFont(font1);
                longitudeS.setFont(font1);
                distanciaS.setFont(font1);
                RAS.setFont(font1);
                DeclS.setFont(font1);
                lati.setFont(font1);
                longi.setFont(font1);
                distancia.setFont(font1);
                RA.setFont(font1);
                Decl.setFont(font1);

                riseS.setFont(font1);
                setS.setFont(font1);


                seeOrbital.add(longi);
            seeOrbital.add(longitudeS);
```

```
seeOrbital.add(lati);
seeOrbital.add(latitudeS);
seeOrbital.add(RA);
seeOrbital.add(RAS);
seeOrbital.add(Decl);
seeOrbital.add(DeclS);

seeOrbital.add(distancia);
seeOrbital.add(distanciaS);



//Azimute e Altitude

azimuteS.setFont(font2);
altitudeS.setFont(font2);
azimute.setFont(font2);
altitude.setFont(font2);

seeAE.add(azimute);
seeAE.add(azimuteS);
seeAE.add(altitude);
seeAE.add(altitudeS);

JPanel graph = new JPanel(new GridLayout(0,2,0,0));

c1.setPreferredSize(new Dimension(140,140));
c1.setOpaque(false);

e1.setPreferredSize(new Dimension(140,140));
e1.setOpaque(false);

graph.add(c1);
graph.add(e1);

outputSun.add(seeOrbital, BorderLayout.NORTH);
outputSun.add(seeAE, BorderLayout.SOUTH);
outputSun.add(graph);

return outputSun ;
}

//OUTPUT

public JPanel output(){

JTabbedPane outputPane = new JTabbedPane();

//Lua Pane
outputPane.addTab("Moon", null, outputMoon(),"Moon Output Values");
outputPane.setMnemonicAt(0, KeyEvent.VK_1);
//Sol Pane
outputPane.addTab("Sun", null, outputSun() , "Sun Output Values");
outputPane.setMnemonicAt(1, KeyEvent.VK_2);
/*   //World Map Pane
outputPane.addTab("World Map", null, WorldMap(),"World Map");
outputPane.setMnemonicAt(2, KeyEvent.VK_3);     */
```

```
      //Adiciona Paineis

              JPanel output = new JPanel();

              output.setBorder(new TitledBorder("Output"));
                output.add(outputPane);

                return output;
 }




//Sentinelas de Escuta checkbox's
public class sentinelaCB implements ActionListener {

@Override
public void actionPerformed(ActionEvent e) {

JCheckBox cb = (JCheckBox) e.getSource();

if(cb == rt){

dayI.setText("DD");
monthI.setText("MM");
yearI.setText("YYYY");
hourI.setText("HH");
minuteI.setText("MM");
secondI.setText("SS");
setNEditable();
st.setSelected(false);
rt.setSelected(true);
}else if (cb == st){
dayI.setText("01");
monthI.setText("10");
yearI.setText("2017");
hourI.setText("00");
minuteI.setText("00");
secondI.setText("00");
setEditable();
rt.setSelected(false);
st.setSelected(true);
}else if (cb == cc){
if(cc.isSelected() == true){

                    latiC.setEditable(true);
                    longiC.setEditable(true);

             }else if (cc.isSelected()== false){

                    latiC.setEditable(false);
                    longiC.setEditable(false);


                    }
             }
    }
}
```

```
public void actionPerformed(ActionEvent e) {
    dataShow.setText(d.toStringDay());
    horaShow.setText(d.toStringHour());

}

public void stateChanged(ChangeEvent e) {
}


PausableThread p = new PausableThread();

//Sentinela para Botoes

public class sentinela implements ActionListener {

            @Override
  public void actionPerformed(ActionEvent e) {

       JButton button = (JButton) e.getSource();

            if(button == buttonCalc){

                        if (rt.isSelected()== true && cc.isSelected()==false){

                              latitude = Double.parseDouble(latiI.getText());
                              longitude =Double.parseDouble(longI.getText());

                              if(lock!=true)
                              p.start();
                              else p.Resume();
                              setDisable();

                     }else if(st.isSelected()==true && cc.isSelected()==false){

                              latitude = Double.parseDouble(latiI.getText());
                              longitude =Double.parseDouble(longI.getText());
                              hour = Integer.parseInt(hourI.getText());;
                              minute =Integer.parseInt(minuteI.getText());;
                              second = Integer.parseInt(secondI.getText());;
                              day = Integer.parseInt(dayI.getText());;
                              month = Integer.parseInt(monthI.getText());;
                              year = Integer.parseInt(yearI.getText());;
                              UT = d.UTCconv(hour, minute, second);
                              j = new JD(day, month, year, UT);
                              s.CalcSun (j.MJDCalcD(),UT,latitude,longitude);
                              m.CalcMoon(j.MJDCalcD(),UT,latitude,longitude);
                              setText();
                    }else if(rt.isSelected()==true && cc.isSelected()==true){
                              latitude = Double.parseDouble(latiI.getText());
                              longitude =Double.parseDouble(longI.getText());
                              latitudeC = Double.parseDouble(latiC.getText());
                              longitudeC =
Double.parseDouble(longiC.getText());
```

```
                                    if(lock!=true)
                                    p.start();
                                    else p.Resume();


                                    setDisable();


                            }else if(st.isSelected()==true && cc.isSelected()==true){

                                    latitude = Double.parseDouble(latiI.getText());
                                    longitude = Double.parseDouble(longI.getText());

                                    latitudeC = Double.parseDouble(latiC.getText());
                                    longitudeC = Double.parseDouble(longiC.getText());

                                    hour = Integer.parseInt(hourI.getText());;
                                    minute =Integer.parseInt(minuteI.getText());;
                                    second = Integer.parseInt(secondI.getText());;

                                    day = Integer.parseInt(dayI.getText());;
                                    month = Integer.parseInt(monthI.getText());;
                                    year = Integer.parseInt(yearI.getText());;

                                    UT = d.UTCconv(hour, minute, second);

                                    j = new JD(day, month, year, UT);

                                    s.CalcSun (j.MJDCalcD(),UT,latitude,longitude);
                                    m.CalcMoon(j.MJDCalcD(),UT,latitude,longitude);

                                    m1.CalcMoon(j.MJDCalcD(),UT,latitudeC,
longitudeC);

                                setText();
                            }
                        }else if(button == buttonStop){

                                p.Pause();

                            lock=true;

                                setEnable();

                    }
                 }
                }

                //Thread responsavel pelo calculo em tempo real
                private class PausableThread extends Thread {

                    private boolean paused;
                    private boolean running;

                    public PausableThread() {
                        paused = false;
                        running = true;
                    }
```

```
    @Override
    public void run() {
        while (running) {
            while (!paused) {
                try {
                    j = new JD(d.getDay(), d.getMonth(),
d.getYear(),d.getUTCtime());
                                s.CalcSun (j.MJDCalcD(), d.getUTCtime(),latitude,
longitude);
                            m.CalcMoon(j.MJDCalcD(),d.getUTCtime(),latitude,
longitude);

WriteFile.write(m.getAzimute(),m.getAltitudeCorrected());
                                if(cc.isSelected()){
m1.CalcMoon(j.MJDCalcD(),d.getUTCtime(),latitudeC, longitudeC);
                                    }
                                     setText();
                                   Thread.sleep(1000);
                } catch (InterruptedException e) {
                    paused = true;
                }
            }

            try {
                Thread.sleep(Long.MAX_VALUE);
            } catch (InterruptedException e) {
                paused = false;
            }
          }
         System.out.println("stopped");
    }

    public void Pause() {
        paused = true;
    }

    public void Resume() {
        interrupt();
        paused = false;
    }
}


public void setText(){

//Lua
latitudeM.setText(""+m.getlat());
longitudeM.setText(""+m.getlongi());
distanciaM.setText(""+m.getDistance()+" Km");
RAM.setText(""+m.getRA());
DeclM.setText(""+m.getDecl());
topRA1.setText(""+m.gettopRA());
topDecl1.setText(""+m.gettopDecl());
MparM.setText(""+m.getMPar());
azimuteM.setText(""+m.getAzimute()+"º");
altitudeM.setText(""+m.getAltitudeCorrected()+"º");

latitudeM.setForeground(Color.blue);
longitudeM.setForeground(Color.blue);
```

```
distanciaM.setForeground(Color.blue);
RAM.setForeground(Color.blue);
topRA1.setForeground(Color.blue);
topDecl1.setForeground(Color.blue);
DeclM.setForeground(Color.blue);
MparM.setForeground(Color.blue);
azimuteM.setForeground(Color.blue);
altitudeM.setForeground(Color.blue);

//Sol
longitudeS.setText(""+s.getlongi());
latitudeS.setText(""+s.getlat());
distanciaS.setText(""+s.getDistance()+" AU");
RAS.setText(""+s.getRA());
DeclS.setText(""+s.getDecl());

azimuteS.setText(""+s.getAzimute()+"º");
altitudeS.setText(""+s.getAltitude()+"º");


longitudeS.setForeground(Color.blue);
latitudeS.setForeground(Color.blue);
distanciaS.setForeground(Color.blue);
RAS.setForeground(Color.blue);
DeclS.setForeground(Color.blue);

azimuteS.setForeground(Color.blue);
altitudeS.setForeground(Color.blue);

//Bussola elevação
c.setAzimute(m.getAzimute());
e.setElevation(m.getAltitude());
c1.setAzimute(s.getAzimute());
e1.setElevation(s.getAltitude());

//Check Conection
if(cc.isSelected() == true && (m.getAltitudeCorrected() > 5.0 &&
m1.getAltitudeCorrected() > 5.0)){
check1.setText("Connection Possible");
check1.setForeground(Color.blue);

}else if(cc.isSelected() == true && (m.getAltitudeCorrected() < 5.0 ||
m1.getAltitudeCorrected() < 5.0)){
check1.setText("Connection Impossible");
check1.setForeground(Color.red);
}
}
public void setEditable(){
dayI.setEditable(true);
monthI.setEditable(true);
yearI.setEditable(true);

hourI.setEditable(true);
             minuteI.setEditable(true);
             secondI.setEditable(true);
}

public void setNEditable(){
             dayI.setEditable(false);
```

```
            monthI.setEditable(false);
            yearI.setEditable(false);

            hourI.setEditable(false);
            minuteI.setEditable(false);
            secondI.setEditable(false);
}
public void setEnable(){
          buttonCalc.setEnabled(true);
            cc.setEnabled(true);
            st.setEnabled(true);
            rt.setEnabled(true);
            dayI.setEnabled(true);
            monthI.setEnabled(true);
            yearI.setEnabled(true);
            hourI.setEnabled(true);
            minuteI.setEnabled(true);
            secondI.setEnabled(true);
            latiI.setEnabled(true);
            longI.setEnabled(true);
            latiC.setEnabled(true);
            longiC.setEnabled(true);
}
public void setDisable(){
            buttonCalc.setEnabled(false);
            cc.setEnabled(false);
            st.setEnabled(false);
            rt.setEnabled(false);
            dayI.setEnabled(false);
            monthI.setEnabled(false);
            yearI.setEnabled(false);
            hourI.setEnabled(false);
            minuteI.setEnabled(false);
            secondI.setEnabled(false);
            latiI.setEnabled(false);
            longI.setEnabled(false);
            latiC.setEnabled(false);
            longiC.setEnabled(false);
}
//Criação Janela

public void createAndShowGUI() {


        gui.add(TimeDate1(), BorderLayout.SOUTH);
        gui.add(InputValues(), BorderLayout.WEST);
        gui.add(output(), BorderLayout.CENTER);

            janela.setContentPane(gui);
          janela.pack();
            janela.setSize(1100, 654);
            janela.setResizable(true);

            janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            janela.setVisible(true);
}


}
```

Date.Java

```
import java.util.Calendar;

public class Date {

//Utilizar formato de mês em letras
      private String months[] = {" ",
                  "JAN", "FEB", "MAR", "APR",
                  "MAY", "JUN", "JUL", "AUG",
                  "SEP", "OCT", "NOV", "DEC"};
//Retorna o Dia
     public int getDay(){
        Calendar now = Calendar.getInstance();
          return  now.get(Calendar.DATE);
     }
//Retorna o Mês
     public int getMonth(){
        Calendar now = Calendar.getInstance();
       return  1+now.get(Calendar.MONTH);
     }
//Retorna o Ano
     public int getYear(){
        Calendar now = Calendar.getInstance();
       return  now.get(Calendar.YEAR);
     }
//Retorna as horas
     public int getHour(){
         Calendar now = Calendar.getInstance();
       return  now.get(Calendar.HOUR_OF_DAY);
     }
 //Retorna as horas UTC
     public int getHourUTC(){
         Calendar now = Calendar.getInstance();
       return  now.get(Calendar.HOUR_OF_DAY)-1;
     }
//Retorna os minutos
     public int getMinute(){
        Calendar now = Calendar.getInstance();
       return  now.get(Calendar.MINUTE);
     }
//Retorna os segundos
     public int getSecond(){
        Calendar now = Calendar.getInstance();
        return now.get(Calendar.SECOND);
     }
//Conversão das horas locais para formato unitário
     public double getUTCtime(){
       return getHourUTC() + (double)getMinute()/60 + (double)getSecond()/3600;
}
 //Conversão das horas de input para formato unitário
    public double UTCconv(double hour, double minute, double second){
              return (hour-1) + (double)minute/60 + (double)second/3600;
         }
//Converte data para output (não utilizado)
    public String toStringDayConv(int day, int month, int year){
        return day+"/"+month+"/"+year;
    }
```

```
 //Converte a Data numa String para output
    public String toStringDay(){
      String day= ""+getDay();
      String month=""+getMonth();


            if(getDay()<10)
                  day="0"+getDay();
            if(getMonth()<10)
                  month="0"+getMonth();

        return day+"/"+ month+"/"+getYear();
    }

//Converte Horas numa string para output
    public String toStringHour(){
      String hour= ""+getHour();
      String minute=""+getMinute();
      String second=""+getSecond();

            if(getHour()<10)
                  hour="0"+getHour();
            if(getMinute()<10)
                  minute="0"+getMinute();
            if(getSecond()<10)
                  second="0"+getSecond();

        return hour+":"+ minute+":"+second;
    }

}
```

Compass.Java

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Polygon;
import java.awt.RenderingHints;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.AffineTransform;
import javax.swing.JPanel;
public class Compass extends JPanel implements ActionListener {
Moon m = new Moon();
private double azimute;
public void paintComponent(Graphics graphics) {
          Graphics2D g = (Graphics2D)graphics;
          g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
          RenderingHints.VALUE_ANTIALIAS_ON);
          int size = Math.min(getWidth(),getHeight());
          int center = size/2;
          AffineTransform at = g.getTransform();
           //Desenha Traços
          g.setColor(Color.BLACK);
          for (int i=0; i<40; i++) {
              g.drawLine(center,center/25,center,center/50);
              g.rotate((Math.PI*2) * (9 / 360.0),center,center);
          }
```

```
            g.setTransform(at);

//Desenha Pontos
            g.setColor(Color.BLACK);
            for (int i=0; i<8; i++) {
               g.fillOval(center-center/40,center/100,center/20,center/20);
                g.rotate((Math.PI*2) * (45 / 360.0),center,center);
            }
            g.setTransform(at);

//Adiciona Coordenadas
            Font font = new Font("Arial", Font.BOLD, 15);
            Font font1 = new Font("Arial", Font.BOLD, 11);
            g.setColor(Color.RED);
            g.setFont(font);
            for (int i=0; i<8; i++) {
               if(i==0)g.drawString("N",center-center/13,(center/2)-center/4);
               g.setColor(Color.BLACK);
               g.setFont(font1);
               if(i==1)g.drawString("NE",center-center/12,(center/2)-center/4);
               g.setFont(font);
               if(i==2)g.drawString("E",center-center/13,(center/2)-center/4);
               g.setFont(font1);
               if(i==3)g.drawString("SE",center-center/12,(center/2)-center/4);
               g.setFont(font);
               g.setColor(Color.BLUE);
               if(i==4)g.drawString("S",center-center/13,(center/2)-center/4);
               g.setColor(Color.BLACK);
               g.setFont(font1);
               if(i==5)g.drawString("SW",center-center/12,(center/2)-center/4);
               g.setFont(font);
               if(i==6)g.drawString("W",center-center/13,(center/2)-center/4);
               g.setFont(font1);
               if(i==7)g.drawString("NW",center-center/12,(center/2)-center/4);
g.rotate((Math.PI*2) * (45 / 360.0),center,center);
            }
            g.setTransform(at);
            int[] x,y;
// Desenha ponteiro
            g.setColor(new Color(0,0,0));
            g.rotate(azimute*(Math.PI/180) ,center,center);
            x = new int[] { center,center+center/20,center,center-center/20 };
            y = new int[] { center,center,center/20,center };
            g.fill(new Polygon(x,y,x.length));
            g.setTransform(at);   }

         public void setAzimute(Double Azimute){
               azimute = Azimute;
               repaint();   }
       @Override
       public void actionPerformed(ActionEvent arg0) {}
}
```

Elevation.Java

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
```

```
import java.awt.Polygon;
import java.awt.RenderingHints;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.geom.AffineTransform;
import javax.swing.JPanel;
public class Elevation extends JPanel implements ActionListener {
	Moon m = new Moon();
	rivate double elevation;
	public void paintComponent(Graphics graphics) {

	     Graphics2D g = (Graphics2D)graphics;
	     g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
	     RenderingHints.VALUE_ANTIALIAS_ON);
	    int size = Math.min(getWidth(),getHeight());
	    int center = size/2;

	    AffineTransform at = g.getTransform();

	    //Desenha Traços
	    g.setColor(Color.BLACK);
	    for (int i=0; i<40; i++) {
	        g.drawLine(center,center/25,center,center/50);
	        g.rotate((Math.PI) * (9 / 360.0),center,center);
	    }
	    g.setTransform(at);
	    //Desenha Pontos
	    g.setColor(Color.BLACK);
	    for (int i=0; i<9; i++) {
	       g.fillOval(center-center/40,center/100,center/20,center/20);
	        g.rotate((Math.PI) * (45 / 360.0),center,center);
	    }
	    g.setTransform(at);

	    Font font = new Font("Arial", Font.BOLD, 15);
	    Font font1 = new Font("Arial", Font.BOLD, 11);

	   //Desenha Numeros
	    for (int i=0; i<5; i++) {
	       g.setFont(font);
	       g.setColor(Color.blue);
	       if(i==0)g.drawString("90º",center-center/13,(center/2)-center/4);

	       g.setFont(font1);
	       if(i==1)g.drawString("45º",center-center/12,(center/2)-center/4);
	       g.setFont(font);
	       g.setColor(Color.black);
	       if(i==2)g.drawString("0º",center-center/13,(center/2)-center/4);
	       g.setFont(font1);
	       g.setColor(Color.red);
	       if(i==3)g.drawString("-45º",center-center/12,(center/2)-center/4);
	       g.setFont(font);
	             if(i==4)g.drawString("-90º",center-center/13,(center/2)-
center/4);

	        g.rotate((Math.PI*2) * (45 / 360.0),center,center);
	    }
	    g.setTransform(at);
```

```
                int[] x,y;

            // Desenha ponteiro
            g.setColor(new Color(0,0,0));
            g.rotate(elevation*(Math.PI/180) ,center,center);
            x = new int[] { center,center+center/20,center,center-center/20 };
            y = new int[] { center,center,center/20,center };
            g.fill(new Polygon(x,y,x.length));
            g.setTransform(at);

            // Desenha Traço

        }

         public void setElevation(Double Elevation){
                elevation=Elevation;
                if(Elevation<0)elevation=Math.abs(elevation)+ 90;
                if(Elevation>0)elevation=90-elevation;
                repaint();
         }

        @Override
        public void actionPerformed(ActionEvent arg0) {
        }
}
```

Elevation.Java

```
public class Earth {
        private double EarthRadKM = 6378.1370;
        MathFormulas m = new MathFormulas();
        public double getEarthRad(){
                return EarthRadKM;
        }
}
```

WriteFile.Java

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;


public class WriteFile {
        public static void Run(double Azimute, double Elevation) throws IOException
{
 //Limpa o Conteudo do ficheiro, se ele existir, e cria um novo ficheiro
    File fileOne = new File("input.dat");
    fileOne.delete();
    File newFile = new File("input.dat");
    newFile.createNewFile();

    //Cria o objecto writer
    PrintWriter writer = new PrintWriter(new FileWriter("input.dat"));
    writer.print("1 "+Azimute+" "+Elevation);
    writer.close();
```

```
        }
        public static void write(double Azimute, double Elevation){
 try {
                        Run(Azimute,Elevation);
                } catch (IOException e) {
                        System.out.println("Erro ao Escrever no ficheiro");
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
          }


}
```

MathFormulas.Java

```
public class MathFormulas {

        double PI = Math.PI;        //PI
        double convR = (PI/180); //Conversor para Radianos
        double convD = (180/PI); //Conversor para Graus
        private double E0;         //Variáveis auxiliares de E (processo Iterativo)
        private double E1;          //Variáveis auxiliares de E (processo Iterativo)
        //Calculo Iterativo para a Eccentric Anomaly
        public double calcE(double M, double e){
              E0 = M + convD * e * sin(M) * (1 + e * cos(M));
             E1 = E0 - (E0 - convD * e * sin(E0) - M) / (1 - e * cos(E0));
         while(E0-E1 > 1 * Math.pow(10,-8)){
            E0 = E1;
                  E1 = E0 - (E0 - convD * e * sin(E0) - M) / (1 - e * cos(E0));
           }
              return norma(E1);   }
//Calculo Distancia através de coordenadas rectangulares
        public double calcDistance(double xrect, double yrect){
              return sqrt(xrect*xrect + yrect*yrect);
        }
//Calculo True Anomaly através de coordenadas rectangulares
        public double calcTAnomaly(double xrect, double yrect){
        return norma(atan2(yrect , xrect )*convD);
        }
//Calculo Coordenadas rectangulares Eclipticas
    public double calcEclCoordX(double r, double longi, double lat){
              return r * cos(longi) * cos(lat);
              }
    public double calcEclCoordY(double r, double longi, double lat){
              return r * sin(longi) * cos(lat);
            }
    public double calcEclCoordZ(double r, double lat){
              return r * sin(lat);
            }
//Calculo rotação de coordenadas rectangulares para equatoriais
    public double calcEquaCoordX(double xgeo){
              return xgeo;
              }
   public double calcEquaCoordY( double ecl, double ygeo , double zgeo){
              return ygeo * cos(ecl) - zgeo * sin(ecl);
            }
   public double calcEquaCoordZ(double ecl, double ygeo, double zgeo){
              return ygeo * sin(ecl) + zgeo * cos(ecl);
            }
```

```
//Arrendonda a 4 Casas decimais
	public double arredonda(double a){
		return (Math.round(a * 10000.0) / 10000.0) ;
	}
//Normalização
	public double norma(double a){
		int b = 0;
		if (a<0){
			b = (int) (-a/360);
			b = b + 1;
			a =  a + b*360;
		}else if(a>360){
			b = (int) (a/360);
			a = a - b*360;
		}return a;
	}
//Converte Graus para Graus, Minutos, Segundos
public String DegreesToHours(double angle){

	String s="";
		double degrees = (int) angle;
		double minutes = Math.abs(angle) - Math.abs(degrees);
			minutes = minutes *60;
		double seconds = Math.abs(minutes - (int)minutes);
			seconds = seconds*60;

		if(angle<0 && angle >-1)
		s = "-"+(int)angle;
		else s=""+degrees;

		return s+"º "+(int)minutes+"' "+(Math.round(seconds * 100.0) /
100.0)+"''";
	}
//Converte de Graus para minutos segundos
public String DegreesToHours1(double angle){
		double minutes = angle *60;
		double seconds = Math.abs(minutes - (int)minutes);
			seconds = (seconds*60);
		return (int)minutes+"' "+(Math.round(seconds * 100.0) / 100.0)+"''";
	}

// Converte de Graus para Horas Minutos Segundos RA
public String DegreesToHours2(double angle){
		double aux=0;
		aux=angle/15;
		int   horas = (int) aux;
		double minutes =Math.abs(aux - horas);
			minutes = minutes *60;
		double seconds = Math.abs(minutes - (int)minutes);
			seconds = (seconds*60);
		return (int)horas+"h "+(int)minutes+"m "+(Math.round(seconds * 100.0)
/ 100.0)+"s";

}
//Funções Trignométricas
	public double cos(double a){
		return Math.cos(a*convR);
		}
```

```
	public double sin(double b){
		return Math.sin(b*convR);
		}
	public double tan(double c){
		return Math.tan(c*convR);
		}
	public double sqrt(double d){
		return Math.sqrt(d);
		}
	public double asin(double g){
		return Math.asin(g);
	    }
	public double atan2(double e, double f){
		return Math.atan2(e*convR, f*convR);
		}
	public double atan(double h){
		return Math.atan(h*convR);
		}
}
```

JD.Java

```
public class JD {
	private int day;
	private int year;
	private int month;
	private double UT;
	private double JD;
	private double T;
	public JD (int day, int month, int year, double UT){
		this.day = day;
		this.month = month;
		this.year = year;
		this.UT = UT;}

//Calculo do JD
	public double JDCalc(){

		if (month<=2){
			month = month +12; year = year-1;
			}
		return JD = (int)(365.25*year) + (int)(30.6001*(month+1)) - 15 +
1720996.5 + day + (UT/24);
	}
//Calculo do MJD
	public double MJDCalcD(){

		return  JDCalc() - 2451543.5;
		}
//Conversão de JD para string
	public String toStringJD(){
		 return ":"+JDCalc();
	    }
//Epoch time
	public double getT(){
			return T = (JD - 451545.0)/36525.0;
	   }

}
```

ItTracker.Java

```
import javax.swing.SwingUtilities;
import javax.swing.UIManager;
import javax.swing.UnsupportedLookAndFeelException;
import javax.swing.UIManager.LookAndFeelInfo;

public class ItTracker {
	public static void main(String[] args) throws InterruptedException {

		try {
			for (LookAndFeelInfo info :
UIManager.getInstalledLookAndFeels()) {
			 if ("Nimbus".equals(info.getName())) {
			     UIManager.setLookAndFeel(info.getClassName());
			        break;
			   }
			 }
		}catch (Exception e) {
			try {

	UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
			 }catch (ClassNotFoundException e1) {
			  // TODO Auto-generated catch block
			  e1.printStackTrace();
			 } catch (InstantiationException e1) {
			  // TODO Auto-generated catch block
			  e1.printStackTrace();
			 } catch (IllegalAccessException e1) {
			  // TODO Auto-generated catch block
			  e1.printStackTrace();
			 } catch (UnsupportedLookAndFeelException e1) {
			  // TODO Auto-generated catch block
			  e1.printStackTrace();
			 }
		}

		final GUI gui = new GUI();

		SwingUtilities.invokeLater(new Runnable() {
		   @Override
		   public void run() {
		    gui.createAndShowGUI();
		   }
		});
}
}
```