|  |  |
| --- | --- |
| http://noticias.paginas.ufsc.br/files/2019/12/60-anos.jpg | **UFSC – Universidade Federal de Santa Catarina**  **CTJ – Centro Tecnológico de Joinville**  **EMB5016 – Cálculo Numérico** |

**Relatório do Trabalho Parcial**

**Luís Eduardo Fernandes Costa Lima**

**20101965**

**Joinville**

**Outubro/2021**

1. **Determine os parâmetros da órbita**

Para determinar o que este item solicita, deve utilizar dos três pontos fornecidos no roteiro, junto com os conhecimentos das Leis de Kepler e da descrição analítica de uma elipse.

O primeiro passo para obter os parâmetros da órbita xc, a e b – parâmetros que possibilitam descrever qualquer ponto do movimento – é reescrever a equação geral da elipse afim de isolar a incógnita y. Dessa forma obtemos uma equação adequada para nosso objetivo, equação a qual foi fornecida pelo roteiro (equação 6); e então é possível montar um sistema linear com os três pontos fornecidos de forma que seja possível e determinado e sua solução corresponda aos parâmetros auxiliares A, B E C, os quais usaremos posteriormente para encontrar os parâmetros da órbita pela relação fornecida pelo roteiro (8) e para calcular a área do percorrida pela linha que liga o satélite e a terra, buscando verificar a 2◦

lei de Kepler, para armazenar esses dados para serem usados em um script diferente foram colocados no arquivo “ParametrosABC.bin”.

Em seguida, após termos montado o sistema linear, é preciso escrever um programa utilizando Python que tem a função de resolver o sistema linear, obtendo A, B e C e, após isso, determinar xc, a e b. Para tal função foi utilizado o método da eliminação gaussiana, o qual recebe a matriz aumentada e aplica a eliminação gaussiana, conseguindo então uma matriz escalonada, a partir dessa matriz é feito a regressão e então se obtém a solução para o sistema linear. Logo, feito o algoritmo da Eliminação Gaussiana, a próxima etapa é fazer um programa que utilizará o anterior e terá o papel de ter os pontos fornecidos, construir a matriz aumentada de acordo com a equação fornecida, utilizar a Eliminação Gaussiana para obter A, B e C e, por fim, determinar os parâmetros de órbita através dos resultados anteriores e então armazena-los em um arquivo externo. Dessa forma atendendo a solicitação.

Tabela 1 – Valores em km dos parâmetros de órbita obtidos

|  |  |
| --- | --- |
| xc | 17665.165095806275 |
| a | 21386.412945812823 |
| b | 5264.737829520467 |

Fonte: Elaborado pelo autor

1. **Faça uma figura com a órbita**

Para essa solicitação deve ser usado os parâmetros da órbita determinados na etapa anterior, e a partir delas e das equações (9 - 13) fornecidas pelo roteiro obter os pontos x e y para qualquer tempo t dado.

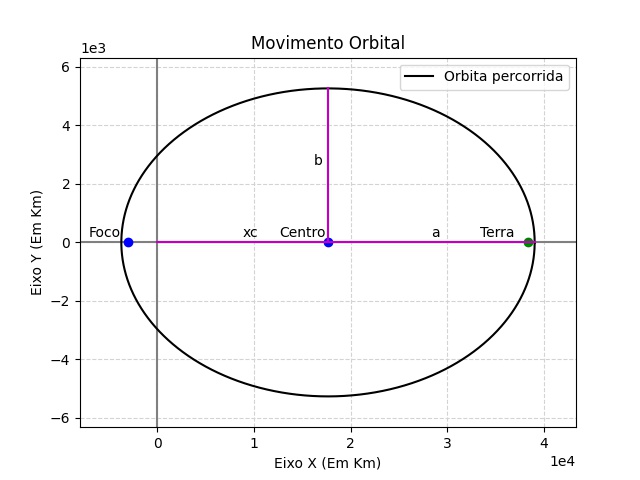
Então, o primeiro passo para plotar essa figura é escrever um programa que contém o método da Bissecção, o qual é responsável por obter a raiz aproximada de uma função dada de acordo com um certo erro de tolerância, esse programa será usado posteriormente para o cálculo da Anomalia excêntrica, já que não há como calcular de forma direta essa grandeza.

Após a construção desse programa, o próximo passo é construir outro programa principal que tem a função de ler os valores dos parâmetros de órbita obtidos anteriormente, obter os pontos que formam a elipse e então, com uso da biblioteca matplotlib, plotar a imagem solicitada. Para executar a primeira parte, o programa busca por um arquivo nomeado de “Parametros.bin”, criado pelo programa que encontrou xc, a, b e então lê linha a linha, cada um contendo um dos parâmetros e os salva em variáveis. Então, após fazer a leitura, o programa faz a chamada de uma função chamada “PontosElipse” que recebe o número de pontos equidistantes em tempo que será retornado; isso se dá, pois, a órbita elíptica é plotada através da ligação de vários pontos consecutivos pertencentes a ela e não por uma infinidade de pontos, logo quanto maior esse número, mais perfeita será a elipse; e recebe os três parâmetros salvos anteriormente. Então, dentro da função, a primeira etapa é calcular o período orbital, a excentricidade e a distância focal, sendo essas obtidas utilizando os parâmetros xc, a e b, após isso é criado uma lista de elementos que vai de 0 até o período orbital, os números de elementos é igual ao número de pontos fornecidos a função e todos eles são espaçados igualmente, essa lista corresponde a um tempo que vai desde do inicio da órbita (t=0) até o ponto final (t=P), nesse caso temos que o período vale 0.9843.

Então, ainda dentro da função, para cada elemento t da lista citada é obtido uma coordenada de um ponto, para isso é usado a função “MValor” que retorna um M (Anomalia média), então com esse valor é calculado um E (Anomalia excêntrica), que utiliza do método da bissecção e a função “EValor” para encontrar uma raiz, então é obtido o ângulo e a distancia em relação a terra pelas funções “AnguloValor” e “DistValor”, respectivamente, sendo essas funções as equações fornecidas no roteiro. Após obter a distância e o ângulo é utilizado a relação fornecida em 13 para se obter as coordenadas dos pontos. Por fim, essa função coloca cada uma dessas coordenadas em listar e as retorna.

O último passo para plotar a figura é, utilizando as listar obtidas na função e a biblioteca matplotlib, plotar a elipse em uma figura através da função plot, e então foi plotado também, usando os parâmetros, o semieixo maior(a), o semieixo menor (b) e o deslocamento (xc). Então é estilizada as cores, legendas e escala da figura e então ela é salva em um arquivo jpg.

Figura 1- Movimento Orbital



Fonte: Elaborado pelo autor

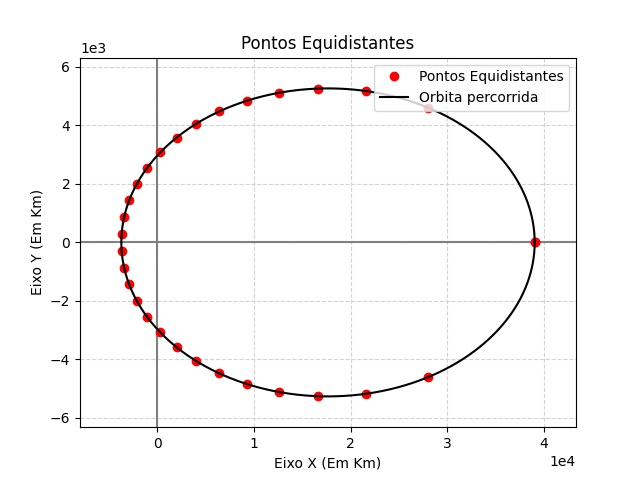
1. **Faça uma figura mostrando a órbita com 30 posições equidistantes em tempo deste satélite**

Está solicitação usa do programa escrito no item passado com algumas implementações, sendo que todos seus requisitos já foram adicionados pelo item anterior.

Então, a primeira etapa para plotar essa imagem é adicionar no corpo principal do programa anterior, comandos que utilizaram da função “PontosElipse”, anteriormente explicada, para produzir um conjunto de 30 pontos equidistante em tempo. Logo é chamado essa função e é colocado como 30 os números de pontos a serem calculador, já que, como citado, a lista de elementos de t é construída de forma que todos os seus elementos tenham o mesmo módulo da diferença entre seu ele e seus vizinhos. E através dessa chamada é obtido as listas que terão as coordenadas x e y dos pontos.

Após isso, é iniciado a plotagem da figura, que consistem em usar a função “clf” da biblioteca matplotlib para limpar os dados da figura anterior, começando do zero uma nova figura, e então plotar os pontos 30 pontos obtidos na última chamada da função “PontosElipse” e também plotar, novamente, a elipse como foi feito no item anterior. Em seguida é feito a estilização das cores, legenda e escala da figura e então ela é salva em um arquivo jpg.

Figura 2- Pontos Equidistantes em tempo



Fonte: Elaborado pelo autor

1. **Verifique a segunda lei de Kepler**

Está última etapa tem como objetivo calcular as áreas feitas pela linha que liga o satélite e a terra em uma determinada variação de tempo para dois tempos iniciais diferentes, verificando se essas áreas são iguais, assim como prevê a segunda lei de Kepler.

Antes da criação do programa que calculará essa área, foi feita a análise da Figura 5 fornecida pelo roteiro. Dessa análise é possível observar que a área a qual se pretende calcular é a área abaixo da curva no ponto inicial ao final somada a área abaixo da reta 2 e subtraída da área abaixo da reta 1, sendo essas áreas, formadas pelas retas, triângulos retângulos.

Assim, foi possível o desenvolvimento do programa responsável por calcular essas áreas, o qual faz uso de técnicas de integração numérica, nesse caso foi utilizado o método de Newton-Cotes. Logo o primeiro passo a ser tomado foi a codificação de um módulo que contém duas funções que usam esse método, porém umas delas é adaptada ao uso da Spline Cúbica; método numérico de interpolação de pontos que também foi desenvolvido nesse processo.

Então, o próximo passo foi o desenvolvimento do programa principal, o “LeiDeKepler.py”. Ele importa os módulos citados anteriormente e também faz uso de parte do código dos itens 2 e 3, pois ele utiliza as funções que calculam os pontos da elipse para ser possível plotar a trajetória da orbita do satélite. No começo do programa são lidos os dados dos parâmetros de orbita escritos no primeiro item e então são usados para calcular o período de órbita e, a partir desse valor, é escolhido um intervalo de tempo e os tempos inicias os quais serão calculadas as respectivas áreas. Então usando as equações dadas (9-13) foi criado a função que calcula a coordenadas de um ponto de órbita para dado tempo e com essa função calculamos os pontos iniciais e os pontos finais dos intervalos de tempo.

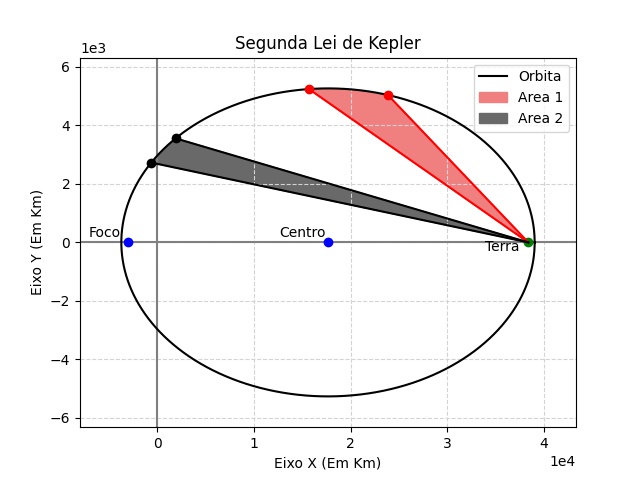
Logo o próximo passo foi a passagem desses pontos para a função que tem como papel o cálculo da área feita entre esses pontos, como citado anteriormente. Essa função recebe como parâmetro uma matriz que contém em uma linha as coordenadas do ponto inicial e na segunda as coordenadas do ponto final e, também, recebe a coordenada x do planeta terra e o números de pontos que serão usados para a interpolação. Então a função verifica quais desses pontos estão mais distantes da terra no eixo x, e o ponto mais distante é aquele o qual a reta 1 liga ele e a terra e o outro ponto forma a reta 2. Em seguida, com essa informação e relembrando que as retas formam triângulos retângulos e usando as bases e alturas, são calculadas as áreas abaixo das duas retas e armazenadas em variáveis. Após isso, é calculada a área abaixo da curva da elipse de duas formas diferentes, a primeira utiliza a equação 6 dada e os valores dos parâmetros armazenados no arquivo “ParametrosABC.bin” no item 1 e a segunda usa um conjunto de pontos dentro desse intervalo para fazer a interpolação, ambas formas utilizam o módulo de Newton-Cotes citados anteriormente e através das funções disponíveis nele calculam a área das curvas, logo é calculado e retornando as áreas da trajetória das duas formas somando a respectiva área da curva a do triângulo 2 e subtraindo da do triângulo.

Os dados obtidos utilizando a equação 6 que descreve a elipse foram de 19651331.44794 km² e 19651332.47002 km², uma diferença menor que 0,0001% em relação à média dessas áreas, logo é possível inferir que essa pequena diferença é fruto dos erros de cálculo causados pelas aproximações do computador e então verificar, através dos resultados, a segunda lei de Kepler.

Já utilizando a Spline Cúbica para descrever a trajetória do satélite nos pontos, os resultados de área foram 19651331.44799 km² e 19651332.47004 km², uma diferença de 5 casas decimais em relação ao cálculo anterior, essa diferença pode ser maior dependendo dos intervalos escolhidos e de quantos pontos foram usados para a interpolação, para atingir essa precisão foram usados 100 pontos. E desses dados podemos verificar, também, a segunda lei de Kepler levando em consideração ao erro computacional, como feito com os dados anteriores.

Esses dados são escritos e armazenados no arquivo “Area.bin” e, em seguida, usando eles e a biblioteca “matplotlib”, é construído a figura q mostras as duas áreas e a trajetória da elipse, assim como os focos da elipse e o centro dela.

Figura 4- Segunda Lei de Kepler



Fonte: Elaborado pelo autor