INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO SERTÃO PERNAMBUCANO
CURSO DE LICENCIATURA EM FÍSICA

JOÃO VITOR DA SILVA SOUZA

ESTUDO DE ÓRBITAS PLANETÁRIAS UTILIZANDO PYTHON: UMA PROPOSTA PARA O ENSINO DE ASTRONOMIA

PETROLINA-PE
2024

JOÃO VITOR DA SILVA SOUZA

ESTUDO DE ÓRBITAS PLANETÁRIAS UTILIZANDO PYTHON: UMA PROPOSTA PARA O ENSINO DE ASTRONOMIA

Trabalho de conclusão de curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano, como parte dos requisitos para a conclusão do curso de Licenciatura em Física. Orientador: Prof. Dr. Erivelton Façanha da Costa.

PETROLINA-PE

2024

Dados Internacionais de Catalogação na Publicação (CIP)

S719 Souza, João Vitor da Silva.

Estudo de Órbitas Planetárias Utilizando Python: Uma Proposta Para O Ensino de Astronomia / João Vitor da Silva Souza. - Petrolina, 2025.
76 f. : il.

Trabalho de Conclusão de Curso (Licenciatura em Física) -Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano, Campus Petrolina, 2025.
Orientação: Prof. Dr. Erivelton Façanha da Costa.

1. Física. 2. Órbitas Planetárias. 3. Simulação em Python. 4. Ensino de Física. I. Título.

CDD 530

Gerado automaticamente pelo sistema Geficat, mediante dados fornecidos pelo(a) autor(a)

INSTITUTO FEDERAL
Sertão Pernambucano
Campus Petrolina

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SERTÃO PERNAMBUCANO
COORDENAÇÃO DE LICENCIATURA EM FÍSICA/ CAMPUS PETROLINA

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO (TCC)

Ata nº 01/2025 da sessão de defesa de Trabalho de Conclusão de Curso do(a) aluno(a) **JOÃO VITOR DA SILVA SOUZA**, do Curso Superior de Licenciatura em Física do Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano - IFSertãoPE, *Campus* Petrolina, realizada no dia **21 de MARÇO de 2025**.

Aos **vinte e um dias do mês de março de dois mil e vinte e cinco** (21/03/2025), na sala D08 do Campus Petrolina do IFSertãoPE, sob a presidência do(a) professor(a) **ERIVELTON FAÇANHA DA COSTA** (IFSertãoPE), reuniu-se a Banca Examinadora composta pelo presidente e pelos membros, os professores **Dr. PEDRO DE SIQUEIRA CAMPOS FILHO** (IFSertãoPE) e **Dr. BRUNO GOMES DA COSTA** (IFSertãoPE). Às **16 horas** (16h), o presidente abriu a sessão de defesa de Trabalho de Conclusão de Curso do(a) aluno(a) **JOÃO VITOR DA SILVA SOUZA**, intitulado **"ESTUDO DE ÓRBITAS PLANETÁRIAS UTILIZANDO PYTHON: UMA PROPOSTA PARA O ENSINO DE ASTRONOMIA",** orientado(a) pelo(a) professor(a) **Dr. ERIVELTON FAÇANHA DA COSTA.** Após a exposição do(a) aluno(a) e arguição da Banca, esta se reuniu reservadamente e decidiu pela **APROVAÇÃO** do aluno, com nota **NOVENTA E SEIS** (96). Nada mais havendo a tratar, foi encerrada a sessão, da qual eu, **ERIVELTON FAÇANHA DA COSTA**, professor(a), lavrei a presente ata que vai assinada por mim e pelos demais membros da Banca.

Petrolina, 21 de março de 2025.

Prof. Dr. ERIVELTON FAÇANHA DA COSTA

Prof. Dr. PEDRO DE SIQUEIRA CAMPOS FILHO

Prof. Dr. BRUNO GOMES DA COSTA

Aos meus pais, Marcelo Sidnei e Maria Lucélia

AGRADECIMENTOS

A Deus, por sempre iluminar e abençoar meu caminho. Se cheguei até aqui, é porque Ele esteve comigo.

Meus pais, pilares da minha jornada, a quem dedico este trabalho com todo o meu amor e gratidão. Nos momentos em que a dúvida me assombrava, vocês foram a luz que me guiou, com seu apoio afetivo, educacional e financeiro. Este trabalho é, em essência, um tributo a vocês.

À minha querida avó, meu eterno agradecimento por todo o cuidado e apoio financeiro que me proporcionou. Sua preocupação constante e generosidade foram fundamentais para que eu chegasse até aqui. Sou imensamente grato por tudo.

À minha irmã, por todas as conversas acadêmicas, que me incentivaram a amadurecer cada vez mais dentro do curso.

À minha amada namorada e futura esposa, que tem sido meu porto seguro desde o início. Com você, compartilho cada sonho, cada alegria e cada desafio. Obrigado por todo o amor, carinho e paciência, especialmente quando minha mente se aventura por universos distantes.

Ao Professor Erivelton Façanha, expresso minha mais profunda gratidão pela orientação exemplar, dedicação incansável e notável competência demonstradas ao longo deste trabalho. Sou eternamente agradecido pelos valiosos ensinamentos, conselhos e contribuições enriquecedoras que me proporcionou. Foi um privilégio desenvolver este projeto sob sua supervisão. Sua paixão pela Astronomia e compromisso com a excelência servem como um farol inspirador, e vejo-o como um modelo a ser seguido. Muito obrigado por tudo.

Agradeço individualmente aos professores Bruno Gomes, Berg Lima e Ericleiton Macedo, por suas contribuições únicas em me incentivar na produção de trabalhos científicos e me introduzir ao mundo acadêmico durante a Licenciatura em Física. Cada trabalho produzido foi crucial para o meu desenvolvimento ao longo do curso.

Ao meu amigo Reinan, por ter me apresentado, em 2022, à linguagem de programação Python e por toda a paciência em me ajudar a evoluir. Sou eternamente grato.

Ao meu amigo Josinaldo, que, de maneira direta ou indireta, sempre escutou minhas ideias e pelas várias horas de conversas construtivas. É um amigo para além do curso.

“Se cheguei até aqui foi porque me
apoiei no ombro dos gigantes."

Isaac Newton

RESUMO

Uma das áreas da Física que chamam grande atenção dos estudantes é a Astronomia. O estudo de órbitas planetárias requer certo grau de abstração, dado que não é possível simulá-las em laboratório, além de requerer certa complexidade na matemática e conceitos físicos envolvidos. Neste contexto, este trabalho discute o desenvolvimento de simulações de órbitas planetárias com programação em Python. A partir do estudo analítico das órbitas, as equações de movimento são transpostas para um código em Python. O objetivo é criar um ambiente virtual onde o estudo das órbitas dos planetas possa ser realizado através de simulações com o VPython, pacote de animações da linguagem. Também propõe-se uma oficina de Python para alunos de Licenciatura em Física do IFSertãoPE, a fim de que se familiarizem tanto com os conceitos físicos e matemáticos envolvidos nos problemas de órbitas planetárias, quanto com a programação em Python em si. Por fim, a partir dos conhecimentos adquiridos, os estudantes devem ser capazes de praticar e desenvolver suas próprias simulações.

Palavras-chave: Órbitas Planetárias; Simulação em Python; Ensino de Física.

ABSTRACT

One of the areas of Physics that attracts great attention from students is Astronomy. The study of planetary orbits requires a certain degree of abstraction, given that it is not possible to simulate them in a laboratory, in addition to requiring some complexity in the mathematics and physical concepts involved. In this context, this work discusses the development of planetary orbit simulations with Python programming. From the analytical study of orbits, the equations of motion are transposed into a Python code. The goal is to create a virtual environment where the study of planetary orbits can be carried out through simulations with VPython, an animation package of the language. It also proposes a Python workshop for Physics Licentiate students at IFSertãoPE, so that they become familiar with both the physical and mathematical concepts involved in planetary orbit problems, as well as with Python programming itself. Finally, from the acquired knowledge, students should be able to practice and develop their own simulations.

Keywords: Planetary Orbits; Python Simulation; Physics Teaching.

LISTA DE FIGURAS

Figura 1 – Simulação do lançamento oblíquo com Vpython. 8
Figura 2 – Gráfico gerado em Vpython da simulação do lançamento oblíquo. . . . 8
Figura 3 – Representação da geometria do modelo ptolomaico. Demonstrando a configuração dos deferentes e epciclos. 9
Figura 4 – Representação do modelo heliocêntrico de Copérnico. O Sol encontra-se no centro e os planetas em órbitas concêntricas ao Sol. 10
Figura 5 – Elementos da elipse. 11
Figura 6 – Lei das áreas. 12
Figura 7 – Representação do torque. 14
Figura 8 – Conservação do momento angular. 15
Figura 9 – Representação do centro de massa M entre dois corpos de massas m_1 e m_2. 16
Figura 10 – Centro de massa M localizado na origem. 17
Figura 11 – Uma órbita binária pode ser reduzida ao problema equivalente de cálculo do movimento da massa reduzida, μ, em torno da massa total, M, localizada na origem. 19
Figura 12 – Representação do infinitésimo de área dA varrida. 21
Figura 13 – Representação dos elementos de um ponto em coordendas polares. . . . 24
Figura 14 – Representação gráfica do potencial efetivo. 26
Figura 15 – Simulação da órbita da Terra e Marte. 36
Figura 16 – Simulação das órbitas de Mercúrio, Vênus, Terra e Marte. 37
Figura 17 – Simulação da órbita de Mercúrio. 37
Figura 18 – Simulação das órbitas de Marte, Júpiter e Saturno. 38
Figura 19 – Simulação das órbitas de Júpiter, Saturno, Urano e Netuno. 39
Figura 20 – Simulação do sistema Solar em Vpython. Branco - Mercúrio, verde - Vênus, azul - Terra, vermelho - Marte e laranja - Júpiter. 39
Figura 21 – Potencial efetivo da Terra simulado em Python. 40
Figura 22 – Potencial efetivo da Terra isolado, destacando o ponto minímo. 41
Figura 23 – Potencial efetivo de Vênus isolado, destacando o ponto minímo. 42
Figura 24 – Representação de r_p, r_a e E no potencial efetivo de mercúrio. 42

LISTA DE TABELAS

Tabela 1 – Terceira Lei de Kepler para os planetas visíveis a olho nu. 13
Tabela 2 – Classificação das órbitas em termos da E e e. 26
Tabela 3 – Massa e excentricidade dos planetas. 35
Tabela 4 – Energias obitidas para os planetas simulados. 38

SUMÁRIO

1 INTRODUÇÃO 3

2 FUNDAMENTAÇÃO TEÓRICA 5
2.1 TEORIA DA APRENDIZAGEM SIGNIFICATIVA (TAS) . . 5
2.1.1 Uso de TIC's no ensino 6
2.1.2 Linguagem de programação em Python no ensino de Física . 7
2.1.3 Simulação 3D com Vpython no ensino de Física 7
2.2 MODELOS GEOCÊNTRICO E HELIOCÊNTRICO 8
2.3 AS LEIS DE KEPLER 9
2.3.1 Primeira lei de Kepler 10
2.3.2 Segunda lei de Kepler 11
2.3.3 Terceira lei de Kepler 12
2.4 LEI DA GRAVITAÇÃO UNIVERSAL 13
2.5 CONSERVAÇÃO DO MOMENTO ANGULAR 14
2.6 DEDUÇÃO DAS LEIS DE KEPLER 15
2.6.1 Dedução da 1º lei de Kepler 15
2.6.1.1 Massa reduzida e momento angular 15
2.6.1.2 Equação da órbita 18
2.6.2 Dedução da 2º lei de Kepler 21
2.6.3 Dedução da 3º lei de Kepler 22
2.7 POTENCIAL EFETIVO 23
2.8 EQUAÇÃO VELOCIDADE 27

3 METODOLOGIA 29
3.1 LEVANTAMENTO BIBLIOGRÁFICO 29
3.2 EQUAÇÕES DO MOVIMENTO 29
3.3 SIMULAÇÃO DAS ÓRBITAS PLANETÁRIAS EM LINGUAGEM DE PROGRAMAÇÃO EM PYTHON 29
3.3.1 Pacotes e declaração de variáveis 29
3.3.2 Posição, velocidade, energia e aceleração 30
3.3.3 Plotagem das órbitas 32
3.3.4 Análise do Potencial efetivo da Terra 33
3.3.5 Plotagem do potencial efetivo 34
3.4 SIMULAÇÃO DAS ÓRBITAS PLANETÁRIAS EM VPYTHON 34

4 RESULTADOS 36

4.1 ÓRBITAS PLANETÁRIAS EM PYTHON 36
4.2 ÓRBITAS PLANETÁRIAS EM VPYTHON 39
4.3 POTENCIAL EFETIVO DA TERRA 40
4.4 PLANO DE AULA - MINICURSO 42
4.4.1 Plano de aula 1 43
4.4.2 Plano de aula 2 44

5 CONCLUSÃO 46

REFERÊNCIAS 47

APÊNDICE A – CÓDIGO DAS ÓRBITAS VIA GOOGLE COLAB 49

APÊNDICE B – CÓDIGO DO POTENCIAL EFETIVO . . 54

APÊNDICE C – CÓDIGO DAS ÓRBITAS VIA VPYTHON 56

APÊNDICE D – POTENCIAL EFETIVO E PONTO MÍNIMO 62

APÊNDICE E – POTENCIAL EFETIVO DE MERCÚRIO 64

1 INTRODUÇÃO

De acordo com Bernardes, Iachel e Scalvi (2008), podemos afirmar que uma das áreas que mais desperta a curiosidade dos alunos é a Astronomia, seja no ensino básico ou superior, servindo como ponto de partida para a investigação científica. Porém, estudar de maneira analítica as leis do movimento planetário exige certo grau de abstração matemática e conceitos físicos envolvidos.

A compreensão das órbitas planetárias é resultado de um longo processo de investigação científica. A evolução dos modelos do Sistema Solar, desde o geocentrismo até o heliocentrismo, demonstra como a ciência se adapta ao longo do tempo. Segundo Nussenzveig (2013), a obra de Copérnico, que foi baseada em dados da antiguidade, impulsionou a astronomia de observação. O modelo copernicano foi aceito pela comunidade científica após o físico alemão Johannes Kepler, utilizando os dados do astrônomo dinamarquês Tycho Brahe, desenvolver de forma empírica as leis para o movimento planetário. De acordo com Hewitt (2000), Kepler, após a morte de Brahe, converteu suas medidas em resultados que seriam coletados a partir de um observador estável (estacionário) externo ao sistema solar. Dessa forma, desenvolveu três leis que regem o movimento planetário.

Assim, como forma de investigação, contribuição ao ensino de astronomia e divulgação científica, este trabalho busca reproduzir o movimento planetário utilizando programação em Python e simulações através do VPython. Ou seja, serão analisadas as leis que regem o movimento dos corpos celestes utilizando Python. O movimento dos corpos sob ação de uma força central, as leis de Kepler e propriedade das órbitas elípticas, serão discutidas a partir de animações gráficas. De maneira geral, este trabalho lança mão das TICs (Tecnologia da Informação e Comunicação), em ambientes virtuais imersivos, de modo a combinar o mundo real com o virtual.

A rápida evolução das tecnologias digitais impõe novos desafios à educação, exigindo que professores e alunos se adaptem a um ambiente em constante mudança. De acordo com Faria (2004), quando tratamos de educação é necessário uma reflexão profunda sobre como encontrar novos métodos alternativos para aumentar a inspiração do professor e estimular os alunos no processo de ensino e aprendizagem. Dessa forma, o uso de simulações didáticas no ensino de Física pode ser uma abordagem pedagógica inovadora que, por meio de softwares, permite aos estudantes visualizar, explorar e compreender fenômenos físicos de forma mais interativa e engajadora (PINTO; HEREDIA; GONÇALVES, 2024).

Diante das dificuldades dos estudantes em compreender a Física, por achá-la difícil e abstrata, o estudo de órbitas planetárias através de linguagem de programação em Python é uma forma de tornar o ensino de Gravitação Universal mais atrativo, de modo a demonstrar visualmente as ideias teóricas, a partir de gráficos e simulação 3D. Segundo Araújo e Figueira (2022), por exemplo, as leis de Kepler são de grande importância

para o conhecimento do universo, embora a primeira lei seja de fácil entendimento, a segunda e terceira lei requerem um nível de abstração maior. Dessa forma, ferramentas computacionais podem facilitar no desenvolvimento do aluno,

> o fato é que a interação entre professor e aluno, através do modo experimental ou virtual, cria um ambiente favorável à aprendizagem, de modo que novas estratégias de ensino podem ser consideradas como oportunidade para serem aplicadas (JUNIOR; ALVES, 2019).

É interessante que o ensino de astronomia seja executado por meio de mecanismos ilustrativos, que permitam ao aluno observar melhor os fenômenos bem como sua matemática.

Este trabalho está estruturado em cinco etapas principais. No capítulo 2, apresenta-se a fundamentação teórica, explorando as teorias da aprendizagem, a evolução dos modelos cosmológicos, desde o geocêntrico até o heliocêntrico, e os conceitos chave da mecânica celeste, como as leis de Kepler, a lei da gravitação universal, a energia orbital e a equação da velocidade orbital.

No capítulo 3, descreve-se a metodologia adotada para atingir os objetivos propostos. Com comentários sobre a construção e funcionamento do código Python utilizado.

No capítulo 4, exibem-se os resultados obtidos na pesquisa, como as órbitas planetárias, gráficos e tabela das energias de cada planeta. Além disso, apresenta-se a proposta de uma oficina de órbitas planetárias para alunos ingressantes no curso de Licenciatura em Física do IFsertãoPE. Finalmente, no capítulo 5, na seção de conclusão, sintetizam-se as principais contribuições do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Diante das dificuldades dos estudantes de física em visualizar o fenômeno conforme o que é trabalhado teoricamente, principalmente em gravitação, torna-se necessária uma ferramenta que estimule a imaginação destes alunos. As simulações computacionais vêm sendo amplamente utilizadas em Física. De acordo com Medeiros e Medeiros (2002), os desenvolvimentos de simulações no ensino Física têm sido amplamente utilizados e podem ser facilmente encontrados na Internet. No IFsertãoPE, está é ainda uma linha de pesquisa em consolidação.

Com isso, é necessário oferecer um meio pelo qual estudantes de licenciatura em Física aprendam a criar simuladores, para que dessa forma possam melhorar o ensino de física na região. Segundo Debald (2007), a prática pedagógica universitária não pode se restringir a informar o acadêmico, pois existem vários espaços onde podem ser encontradas informações. Vasconcelos (2022) defende a utilização de simulações computacionais na aula de ciência por proporcionar um ambiente interativo entre o aluno e professores.

2.1 TEORIA DA APRENDIZAGEM SIGNIFICATIVA (TAS)

Grande parte dos profissionais da educação têm notado que são necessárias inovações na maneira de educar. Segundo Bezerra et al. (2009), atualmente os professores estão enfrentando grandes mudanças devido à inserção das novas tecnologias no meio educacional. O método tradicional de ensino se resume ao pincel e quadro branco. É comum professores escutarem algo como: “Onde irei usar isso em minha vida?” ou “Tenho 27 anos e nunca usei Bhaskara!”. Sendo assim, sobre aprendizagem significativa:

> é preciso entender que a aprendizagem é significativa quando novos conhecimentos (conceitos, ideias, proposições, modelos, fórmulas) passam a significar algo para o aprendiz, quando ele ou ela é capaz de explicar situações com suas próprias palavras, quando é capaz de resolver problemas novos, enfim, quando compreende. Essa aprendizagem se caracteriza pela interação entre os novos conhecimentos e aqueles especificamente relevantes já existentes na estrutura cognitiva do sujeito que aprende (MOREIRA, 2003).

Durante o processo de ensino aprendizagem, faz-se necessário observar de que forma o conteúdo está sendo transmitido, ou seja, como se dá a construção de tal conhecimento. Segundo Tavares (2004), é simples ocorrer a construção do conhecimento quando a discussão se inicia de uma ideia generalizada e inclusiva, se encaminhando para ideias menos inclusivas. Por exemplo, em uma aula de gravitação, seria mais interessante iniciar o debate com uma visão histórica geral e depois analisar detalhadamente os elementos do

conteúdo. Dessa forma, realiza-se uma organização de ideias de forma hierárquica, melhorando o processo de aprendizagem e a assimilação das informações na estrutura cognitiva do estudante.

De acordo com Ausubel (1982), essas estruturas cognitivas são um conjunto de conceitos organizados de maneira hierárquica que representam um indivíduo, formados pela quantidade de ideias e sua organização. Ausubel desenvolveu uma nova ideia de aprendizagem, saindo do método tradicional de ensino, em que o aluno tem a obrigação de memorizar o conteúdo e o professor ensinar o que já existe (ALMEIDA et al., 2016). Sendo assim, a aprendizagem significativa ocorre a partir da "incorporação não arbitrária". O estudante que entende algo é capaz de explicar com suas palavras, além disso, consegue interligar o conhecimento adquirido com outros tipos de conhecimento (AUSUBEL, 1982). Ainda, sobre aprendizagem significativa, Pelizzari et al. (2002) afirma que:

> efetivamente, a aprendizagem significativa tem vantagens notáveis, tanto do ponto de vista do enriquecimento da estrutura cognitiva do aluno, como do ponto de vista da lembrança posterior e da utilização para experimentar novas aprendizagens, fatores que a delimitam como sendo a aprendizagem mais adequada para ser promovida entre os alunos.

Sendo assim, a pergunta que os professores deveriam se fazer é: "De que forma é possível alcançar uma aprendizagem significativa?". Quais recursos são necessários a uma aprendizagem significativa? Um tema recorrente na literatura é a necessidade de preparação interdisciplinar do estudante, de modo a aumentar a construção das interligações entre as várias disciplinas (FRANCO; FILHO, 2017). Nesse sentido, a tecnologia como ferramenta pedagógica pode contribuir como prática inovadora para uma educação de qualidade, articulada com o conhecimento escolar e o currículo, conduzindo para uma aprendizagem significativa (NERLING; DARROZ, 2021).

2.1.1 Uso de TIC's no ensino

Durante os últimos anos, as tecnologias da informação e comunicação (TIC 's) têm ganhado espaço na educação. De acordo com Lobo e Maia (2015), atualmente não há dúvidas se as escolas devem utilizar recursos tecnológicos como ferramentas lúdicas para o ensino, já que a tecnologia é uma realidade diária no mundo. Sendo assim, é interessante analisar de que forma é possível utilizar estas tecnologias para um melhor desenvolvimento em sala de aula. De acordo com Debald (2007),

> é possível dizer que o uso das tecnologias em sala de aula não ocorre instantaneamente, por vontade exclusiva do professor. Para usar adequadamente as TICs em sala de aula torna-se necessário não só um processo de integração e domínio dos meios tecnológicos de computação, mas também um conhecimento de como estes meios podem ser utilizados

> para potencializar o processo de ensino. Este processo é lento e gradual.

Dessa forma, o professor deve estar familiarizado e preparado para trabalhar com estas ferramentas. Com isso, é possível dizer que a tecnologia tornou-se uma ferramenta útil para o processo de ensino e aprendizagem (LOBO; MAIA, 2015). As TICs são de grande relevância para ajudar professores, alunos, instituições e os demais que as utilizam no processo do desenvolvimento cognitivo do estudante (PESENTE; MATOS; AVELINO, 2023). Sobre isso, Nerling e Darroz (2021) argumenta:

> nesse sentido, a escola e o professor estão desafiados na construção e aplicação de novas metodologias de ensino, facilitadoras da aprendizagem e garantidoras da inclusão, e na utilização da tecnologia como ferramenta para complementar as práticas pedagógicas (NERLING; DARROZ, 2021).

2.1.2 Linguagem de programação em Python no ensino de Física

A linguagem de programação Python vem ganhando força no campo das ciências exatas (CONCEIÇÃO; ADMIRAL, 2022). Trata-se de uma linguagem de uso geral que se destacou pela facilidade de programação (VASCONCELOS, 2022).

No contexto educacional, o Python vem tendo um grande desenvolvimento em simulações. De fácil entendimento para o aluno, é mais simples trabalhar em sala de aula (ALMEIDA et al., 2016). Sendo assim, é uma ferramenta que pode ser explorada para tornar as aulas de física mais lúdicas e atrativas com simulações, visto que os conteúdos trabalhados podem ser demonstrados em espaços virtuais.

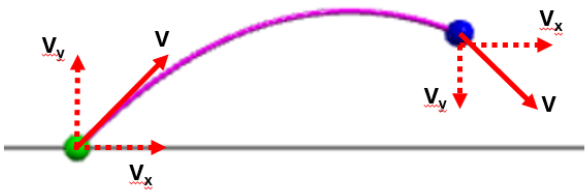
2.1.3 Simulação 3D com Vpython no ensino de Física

De maneira geral, o Vpython é um software que funciona com base na linguagem de programação em Python. De acordo com Silva (2021), o Vpython permite visualizar animações 3D e manipular sistemas físicos, mesmo para aqueles que não têm conhecimento de programação. Assim, é possível familiarizar facilmente os alunos com o software, o que permite que estes possam desenvolver seu próprio sistema físico de maneira rápida, fácil e interativa. Ainda sobre as contribuições da simulação no ensino de Física, Almeida et al. (2016) argumentam:

> as simulações contribuem para a modelagem dos sistemas físicos, facilitando a compreensão do aluno e possibilitando a construção do processo de ensino e aprendizagem. Se essa simulação é usada para representar um sistema físico não observável em um laboratório da escola, como, por exemplo, a órbita de um planeta, então ela poderá destruir conceitos prévios que foram construídos de forma errônea (ALMEIDA et al., 2016).

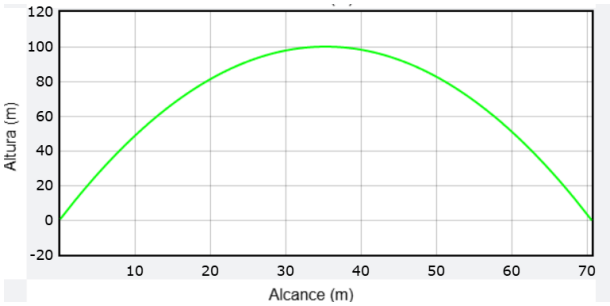
As figuras 1 e 2 são registros do simulador do movimento oblíquo construído a partir do Vpython. Nele é construído o movimento e ilustrado cada elemento importante no processo de aprendizagem do conteúdo, como os vetores velocidade nas direções x e y, a trajetória, o ângulo e alcance máximo atingido pela partícula.

Figura 1 – Simulação do lançamento oblíquo com Vpython.



Fonte: próprio autor.

Figura 2 – Gráfico gerado em Vpython da simulação do lançamento oblíquo.



Fonte: próprio autor.

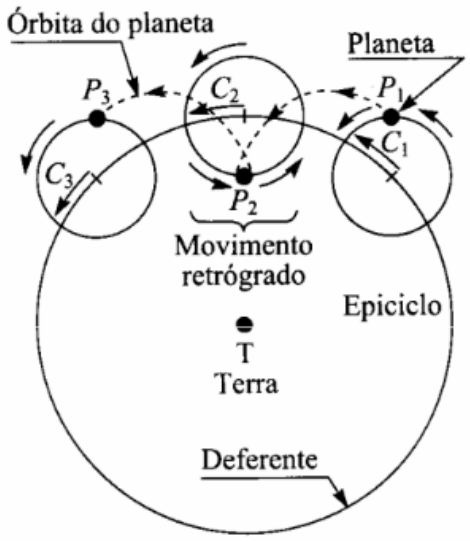
2.2 MODELOS GEOCÊNTRICO E HELIOCÊNTRICO

O modelo para o sistema solar desenvolvido por Ptolomeu foi o trabalho mais importante da Geometria e da Astronomia grega (NEVES, 2000). De acordo com Corcetti e Veraszto (2017), as ideias desenvolvidas por Ptolomeu permaneceram por mais de 1400 anos. Seu trabalho descrevia o movimento dos corpos celestes em órbitas circulares ao redor da Terra, em uma configuração geométrica de ciclos e epiciclos (figura 3), representando o primeiro sistema planetário geocêntrico no qual o movimento dos planetas foi descrito com precisão, levando em consideração a quantidade de informações disponíveis à época (CORCETTI E VERASZTO *apud* SILVA, 2017).

É interessante notar que a ideia de Ptolomeu não tinha nenhum problema matemático. Segundo Kepler e Saraiva (2004), o objetivo de Ptolomeu era de construir um modelo que

permitisse descrever a posição dos planetas de forma correta, o que foi razoavelmente bem-sucedido. A incoerência no modelo desenvolvido por Ptolomeu era que a interpretação física não fazia sentido. O simples fato de existir movimento dos planetas em séries de epiciclos em torno de nada não tem sentido físico (CORCETTI; VERASZTO, 2017). Para não falar da quantidade de epiciclos que surgiam no modelo a cada problema encontrado na teoria.

Figura 3 – Representação da geometria do modelo ptolomaico. Demonstrando a configuração dos deferentes e epciclos.



Fonte: Nussenzveig (2013).

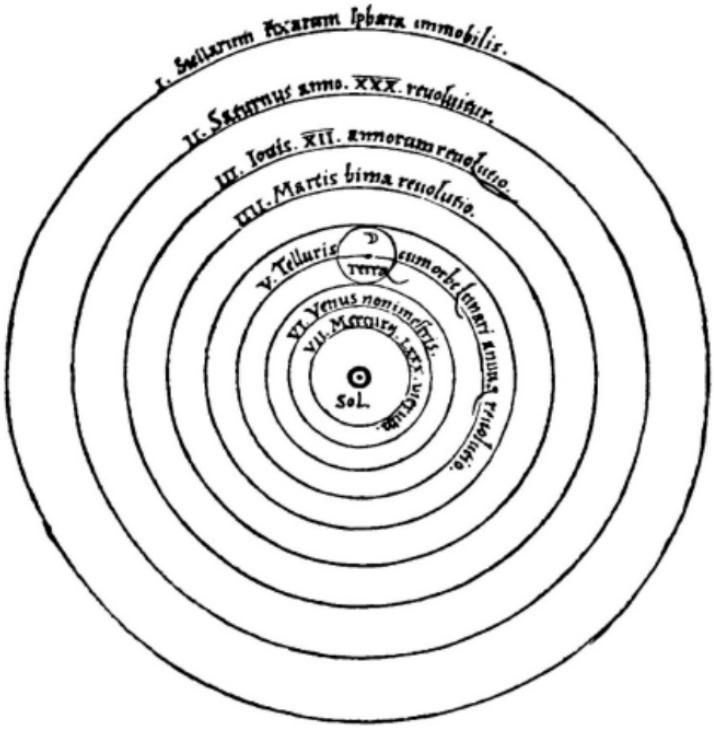
Sendo assim, em 1543, Copérnico em sua obra *De revolutionibus orbium coelestium* (Sobre as revoluções das esferas celestes) muda o referencial para o Sol, deixando a Terra como mais outro planeta a orbitar o "astro-rei"(NEVES, 2000). Essa teoria foi denominada de modelo heliocêntrico (figura 4). Ou seja, o ponto revolucionário das descobertas de Copérnico foi a teoria heliocêntrica, na qual houve a troca de posição entre a Terra e o Sol.

A teoria heliocêntrica já havia sido proposta na antiguidade por Aristarco de Samos (matemático e astrônomo grego). Ele chegou a tal conclusão com base nos tamanhos e distâncias do Sol e da Lua (STEINER, 2006). Naquela época, essa ideia ia contra o que pensava Aristóteles e logo foi abandonada. Com Copérnico, a teoria heliocêntrica conseguiu dar explicações objetivas para os fenômenos observados, como o movimento retrógrado dos planetas. Porém, Copérnico falhou em prever as posições dos planetas com precisão (KEPLER; SARAIVA, 2004).

2.3 AS LEIS DE KEPLER

Nascido no sudoeste da Alemanha, em um vilarejo chamado Weil-der-Stadt, o astrônomo e matemático Johannes Kepler (1571-1630) era um rapaz dedicado, tímido, pobre

Figura 4 – Representação do modelo heliocêntrico de Copérnico. O Sol encontra-se no centro e os planetas em órbitas concêntricas ao Sol.



Fonte: Carroll e Ostlie (2017).

que passou a vida enfrentando problemas de saúde (AVILA, 1989). Mesmo vivendo em um período conturbado na Europa devido às guerras religiosas, Kepler não tinha problemas em expor suas ideias na Alemanha, dessa forma, depois de longos anos de estudo conseguiu desenvolver as três leis que regem o movimento dos astros (BALDOW; SILVA, 2014).

Quando Kepler teve acesso ao Almagesto de Ptolomeu, julgou o trabalho como sendo um sistema muito complicado. Os epiciclos serviam para explicar os movimentos dos planetas, mas tornava o sistema incompreensível (CONTADOR, 2022). Segundo Almeida et al. (2016), os estudos de Kepler sobre as órbitas planetárias alteram completamente e contrariam o geocentrismo. Com Kepler, as órbitas dos planetas deixam de ser circulares e passam a ter uma geometria elíptica.

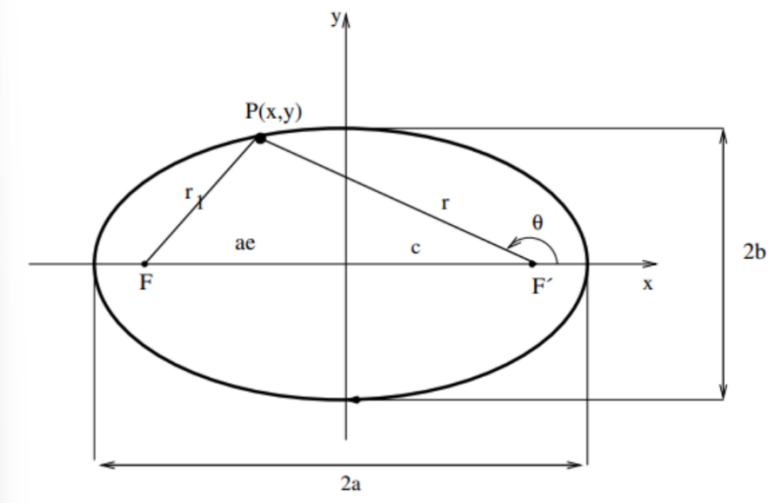
2.3.1 Primeira lei de Kepler

De acordo com Nussenzveig (2013), para Kepler, as órbitas descritas pelos planetas ao redor do Sol são elipses com o Sol em um dos focos.

A partir da fig.5, aplicando a lei dos cossenos ao triângulo FPF´, podemos chegar a

$$r_1^2 = r^2 + (2ae)^2 + 2r(2ae)cos(\theta).$$

Figura 5 – Elementos da elipse.



Fonte: Kepler e Saraiva (2004).

Como descrito por Kepler e Saraiva (2004), podemos definir:

$$r_1 + r = 2a$$

o que nos leva a

$$(2a - r)^2 = r^2 + 4a^2e^2 + 4raecos(\theta)$$

$$4a^2 - 4ar + r^2 = r^2 + 4a^2e^2 + 4raecos(\theta)$$

$$4a^2 - 4ar = 4a^2e^2 + 4raecos(\theta)$$

$$ar + raecos(\theta) = a^2 - a^2e^2 \quad \Rightarrow \quad r(1 + ecos(\theta)) = a(1 - e^2)$$

$$\therefore r = \frac{a(1 - e^2)}{(1 + ecos(\theta))}, \tag{2.1}$$

em que r é a equação da elipse na forma polar, e é a excentricidade da órbita e a o semi-eixo maior. Note que se $e = 0$, temos $r = a$. Ou seja, a órbita passa a ser circular.

2.3.2 Segunda lei de Kepler

Segundo Contador (2022), a segunda lei de Kepler afirma que o raio vetor que liga o planeta ao Sol varre áreas iguais em tempos iguais. Segundo Kepler e Saraiva (2004):

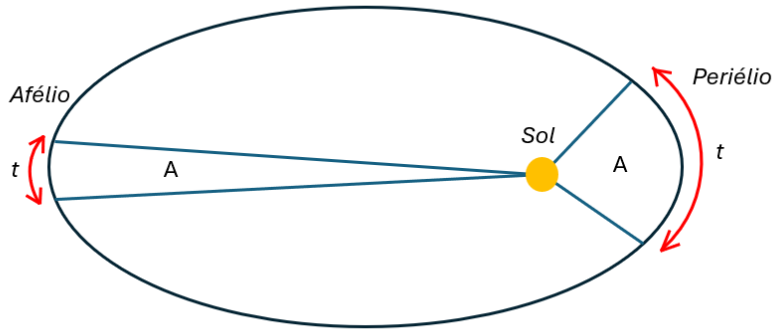
> o significado físico dessa lei é que a velocidade orbital não é uniforme, mas varia de forma regular: quanto mais distante o planeta do Sol, mais devagar ele se move. Dizendo de outra maneira, essa lei estabelece que

a velocidade areal é constante (KEPLER; SARAIVA, 2004).

Então, é possível afirmar que a taxa de variação da área varrida em relação ao tempo é igual a uma constante (figura 6):

$$\frac{dA}{dt} = constante \tag{2.2}$$

Figura 6 – Lei das áreas.



Fonte: Adaptado de Nussenzveig (2013).

2.3.3 Terceira lei de Kepler

A terceira e última lei de Kepler diz que: os quadrados dos períodos dos planetas são proporcionais ao cubo de sua distância média ao Sol, ou seja:

$$T^2 = Ka^3 \tag{2.3}$$

em que T é o período orbital, tempo necessário para um planeta dar uma volta completa em torno do Sol. Na tabela 1, é possível observar o período de cada planeta do sistema solar visíveis à época de Kepler.

Em outras palavras, essa lei estabelece que planetas com órbitas maiores se movam mais lentamente em torno do Sol e, portanto, isso implica que a força entre o Sol e o planeta decresce com a distância (KEPLER; SARAIVA, 2004).

Tabela 1 – Terceira Lei de Kepler para os planetas visíveis a olho nu.

Planeta	Semi-eixo Maior (UA)	Período (anos)	a^3	T^2
Mercúrio	0,387	0,241	0,058	0,058
Vênus	0,723	0,615	0,378	0,378
Terra	1,000	1,000	1,000	1,000
Marte	1,524	1,881	3,537	3,537
Júpiter	5,203	11,862	140,8	140,7
Saturno	9,534	29,456	867,9	867,7

Fonte: Kepler e Saraiva (2004).

2.4 LEI DA GRAVITAÇÃO UNIVERSAL

Das leis da dinâmica, sabemos que em um movimento circular uniforme a força resultante será a força centrípeta, o que nos permite escrever

$$F = \frac{mv^2}{R} \tag{2.4}$$

em que a velocidade linear pode ser escrita como $v = wR$, sendo w a velocidade angular do corpo, expressa por $w = \frac{2\pi}{T}$, onde T é o período do movimento. Dessa forma, temos

$$v^2 = \frac{4\pi^2 R^2}{T^2} \tag{2.5}$$

então

$$F = 4\pi^2 \frac{mR}{T^2}. \tag{2.6}$$

Segundo Nussenzveig (2013), pela terceira lei de Kepler, $\frac{R^3}{T^2} = C$. Substituindo na eq.(2.6), temos

$$F = 4\pi^2 C \frac{m}{R^2}. \tag{2.7}$$

A lei dos períodos nos permite concluir que a força gravitacional varia inversamente com o quadrado da distância do planeta ao Sol, $F\alpha\frac{m}{R^2}$. Observando a eq.(2.7) e pela terceira lei de Newton, conclui-se que a força que o planeta faz sobre o Sol é $F \propto \frac{M}{R^2}$, onde M é a massa solar. O que leva a:

$$\vec{F} = -\frac{GMm}{R^2}\hat{r} \tag{2.8}$$

A equação 2.8 é a lei da gravitação universal de Isaac Newton, onde G é a constante gravitacional universal. Dessa maneira, Newton conclui que, se a força gravitacional estiver correta, deve existir uma força atrativa entre dois corpos em qualquer lugar no universo. Tal força é proporcional ao produto das massas e inversamente proporcional ao quadrado da distância (KEPLER; SARAIVA, 2004).

2.5 CONSERVAÇÃO DO MOMENTO ANGULAR

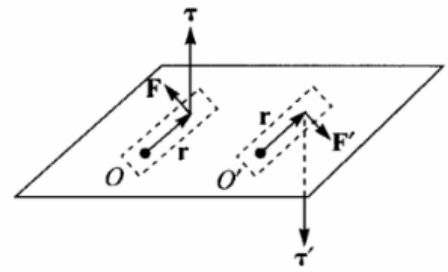
É interessante compreender o que é o momento angular. Enquanto a cinemática translacional descreve o movimento linear de um corpo sob a ação de forças, a cinemática rotacional analisa o movimento de rotação em torno de um eixo fixo. Neste contexto, o conceito de torque substitui a força como a grandeza fundamental. O torque, ou momento de força, é definido como o produto vetorial da força aplicada pelo seu braço de alavanca, que representa a distância perpendicular entre a linha de ação da força e o eixo de rotação (NUSSENZVEIG, 2013).

Definimos torque como

$$\vec{\tau} = \vec{r} \times \vec{F} \therefore \tau = rFsen(\theta), \tag{2.9}$$

em que r é a distância entre o eixo de rotação e o ponto de aplicação da força, F a força aplicada e θ é o ângulo entre r e F. Note que se $r \parallel F$, então $\tau = 0$. Na figura 7 é possível observar que o torque é perpendicular ao plano que contém os vetores $\vec{r}$ e $\vec{F}$.

Figura 7 – Representação do torque.



Fonte: Nussenzveig (2013).

Se o torque for nulo, podemos fazer

$$\tau = 0 \therefore m(\vec{r} \times \vec{a}) = m(\vec{r} \times \frac{d\vec{v}}{dt}) = 0. \tag{2.10}$$

Logo, pela regra do produto

$$\frac{d}{dt}(\vec{r} \times \vec{v}) = \frac{d\vec{r}}{dt} \times \vec{v} + \vec{r} \times \frac{d\vec{v}}{dt} \tag{2.11}$$

$$\frac{d}{dt}(\vec{r} \times \vec{v}) = \vec{r} \times \frac{d\vec{v}}{dt}, \tag{2.12}$$

em que $d\vec{r}/dt \times \vec{v} = \vec{v} \times \vec{v} = 0$. Então, substituindo na eq.(2.10), temos

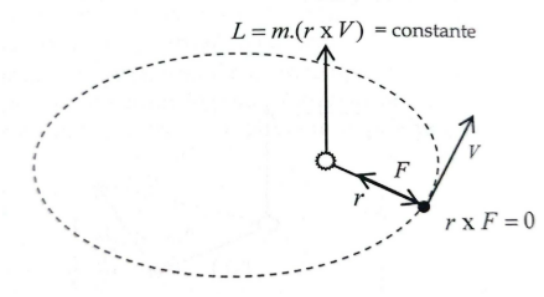
$$m\frac{d}{dt}(\vec{r} \times \vec{v}) = 0 \tag{2.13}$$

ou seja, chegamos a conclusão de:

$$m(\vec{r} \times \vec{v}) = Constante = \vec{l}. \tag{2.14}$$

Note que a eq.(2.14) é o momento angular. O resultado aqui obtido prova a conservação do momento angular em sistemas de forças centrais. Assim, fica provado que se não existir torque (força central), então, o momento angular é conservado (CONTADOR, 2022). A figura 8, ilustra a conservação do momento angular e sua relação com a força gravitacional.

Figura 8 – Conservação do momento angular.



Fonte: Contador (2022).

2.6 DEDUÇÃO DAS LEIS DE KEPLER

Nesta seção será deduzido as Leis de Kepler a partir da gravitação universal de Newton. Sendo assim, os resultados serão comparados com o que foi visto nas seções 2.3, 2.4 e 2.5.

2.6.1 Dedução da 1º lei de Kepler

Para entender a dedução da primeira lei de Kepler a partir da lei da gravitação universal é fundamental começarmos pelo conceito de centro de massa em sistemas de dois corpos sujeitos à interação gravitacional.

2.6.1.1 Massa reduzida e momento angular

Observando a fig.(9), é possível escrever

$$\vec{r} = \vec{r_2} - \vec{r_1} \tag{2.15}$$

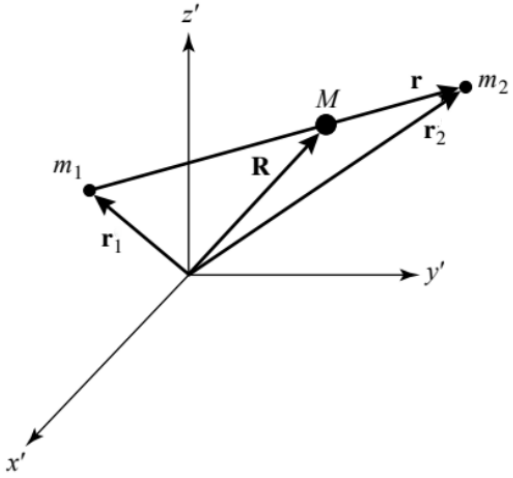
Com isso, escreve-se a posição do centro de massa $\vec{R}$ em termos de m_1 e m_2:

$$\vec{R} = \frac{m_1\vec{r_1} + m_2\vec{r_2}}{m_1 m_2} \tag{2.16}$$

que podemos generalizar

$$\vec{R} = \frac{\sum_{i=1}^{n} m_i\vec{r_i}}{\sum_{i=1}^{n} m_i} \therefore \sum_{i=1}^{n} m_i\vec{R} = \sum_{i=1}^{n} m_i\vec{r_i}. \tag{2.17}$$

Figura 9 – Representação do centro de massa M entre dois corpos de massas m_1 e m_2.



Fonte: Carroll e Ostlie (2017).

Note que $\sum_{i=1}^{n} m_i$ é a massa total do sistema, o que permite definir

$$M\vec{R} = \sum_{i=1}^{n} m_i \vec{r_i}. \tag{2.18}$$

Considerando a massa total do sistema M constante e derivando a eq.(2.18) em relação ao tempo, é possível escrever

$$M\frac{d\vec{R}}{dt} = \sum_{i=1}^{n} m_i \frac{d\vec{r_i}}{dt} \tag{2.19}$$

$$\therefore \vec{p_t} = \sum_{i=1}^{n} \vec{p_i} \tag{2.20}$$

em que $\vec{p_t}$ é o momento linar total do sistema. Veja que é possível, ainda, derivar ambos os lados da eq.(2.20) em relação ao tempo:

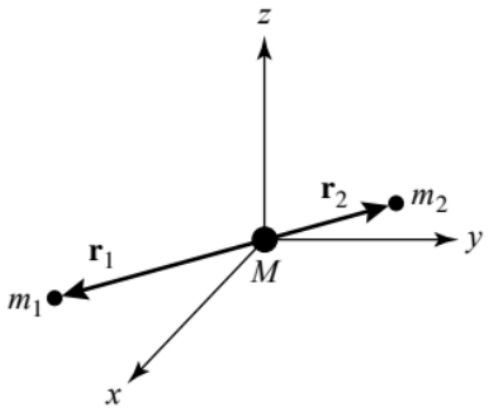
$$\frac{d\vec{p_t}}{dt} = \sum_{i=1}^{n} \frac{d\vec{p_i}}{dt} \tag{2.21}$$

que é justamente a definição de força. De acordo com Carroll e Ostlie (2017), a terceira lei de Newton garante que a soma das forças internas ao sistema é nula,

$$F = M\frac{d^2\vec{R}}{dt} = 0, \tag{2.22}$$

ou seja, se não existe forças externas sobre o sistema, a aceleração do centro de massa é nula, o que significa que ele permanece em repouso ou se move com velocidade constante. Essa propriedade nos permite usar o centro de massa como referencial, simplificando a análise do movimento (figura 10).

Figura 10 – Centro de massa M localizado na origem.



Fonte: Carroll e Ostlie (2017).

Sendo assim, é possível definir a massa reduzida do sistema. Observando a fig.(10), é possível escrever

$$\vec{R} = \frac{m_1\vec{r_1} + m_2\vec{r_2}}{m_1 + m_2} = 0. \tag{2.23}$$

Se a distância entre as duas massas passa a ser

$$\vec{r} = \vec{r_2} - \vec{r_1}, \tag{2.24}$$

na eq.(2.23), escreve-se

$$m_1\vec{r_1} + m_2\vec{r_2} = 0. \tag{2.25}$$

Substituindo $\vec{r_2}$, tem-se

$$m_1\vec{r_1} + m_2(\vec{r_1} + \vec{r}) = m_1\vec{r_1} + m_2\vec{r_1} + m_2\vec{r} = 0. \tag{2.26}$$

Colocando $\vec{r_1}$ em evidência, escreve-se

$$\vec{r_1}(m_1 + m_2) = -m_2\vec{r} \therefore \vec{r_1} = -\frac{m_2}{m_1 + m_2}\vec{r}. \tag{2.27}$$

De maneira análoga, pode-se encontrar $\vec{r_2}$. Logo, fazendo a susbtituição de $\vec{r_1}$ na eq.(2.25), nos leva a

$$m_1(\vec{r_2} - \vec{r}) + m_2\vec{r_2} = m_1\vec{r_2} - m_1\vec{r} + m_2\vec{r_2} = 0. \tag{2.28}$$

Colocando r_2 em evidência, tem-se

$$\vec{r_2}(m_1 + m_2) = m_1\vec{r} \therefore \vec{r_2} = \frac{m_1}{m_1 + m_2}\vec{r} \tag{2.29}$$

Então, a massa reduzida do sistema é

$$\mu = \frac{m_1 m_2}{m_1 + m_2} \tag{2.30}$$

a massa reduzida é fundamental no estudo de sistemas de dois corpos. Ao invés de analisar o movimento de ambos os corpos em torno do centro de massa, pode-se tratar o problema como se um único corpo, com massa reduzida, orbitasse ao redor de uma massa localizada no centro. Sendo assim, tem-se que

$$\begin{cases} r_1 = -\frac{\mu}{m_1}\vec{r} \\ r_2 = \frac{\mu}{m_2}\vec{r} \end{cases}. \tag{2.31}$$

Dessa forma, pode-se reescrever a eq.(2.14) como

$$\vec{l} = m_1(\vec{r_1} \times \vec{v_1}) + m_2(\vec{r_2} \times \vec{v_2}) \tag{2.32}$$

que a partir da eq.(2.31), escreve-se

$$\vec{l} = m_1 \left(-\frac{m_2}{m_1 + m_2}\right)\vec{r} \times \left(-\frac{m_2}{m_1 + m_2}\right)\vec{v} + m_2 \left(\frac{m_1}{m_1 + m_2}\right)\vec{r} \times \left(\frac{m_1}{m_1 + m_2}\right)\vec{v} \tag{2.33}$$

que passa a ser

$$\vec{l} = \mu(\vec{r} \times \vec{v}) \left(\frac{m_1 + m_2}{m_1 + m_2}\right) \tag{2.34}$$

logo,

$$\vec{l} = \mu(\vec{r} \times \vec{v}) \tag{2.35}$$

que é o momento angular em termos da massa reduzida do sistema. Dessa forma, pode-se subistituir o problema de dois corpos por um problema de um corpo com massa μ orbitando a massa M que está localizada no centro de massa do sistema (figura 11).

2.6.1.2 Equação da órbita

É possível reescrever a eq.(2.35) da seguinte maneira:

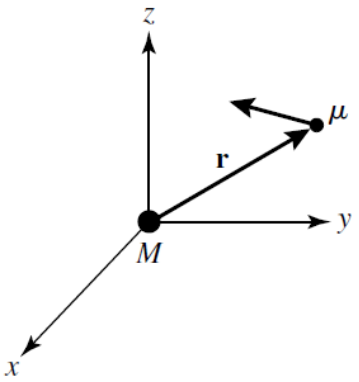
$$\vec{l} = \mu r\hat{r} \times \vec{v} \because \vec{r} = r\hat{r} \tag{2.36}$$

em que

$$\vec{l} = \mu r\hat{r} \times \frac{d}{dt}(r\hat{r}) = \mu r\hat{r} \times \left(\frac{dr}{dt}\hat{r} + r\frac{d\hat{r}}{dt}\right)$$

$$\therefore \vec{l} = \mu r^2 \hat{r} \times \frac{d\hat{r}}{dt} \because \hat{r} \times \hat{r} = 0. \tag{2.37}$$

Figura 11 – Uma órbita binária pode ser reduzida ao problema equivalente de cálculo do movimento da massa reduzida, μ, em torno da massa total, M, localizada na origem.



Fonte: Carroll e Ostlie (2017).

A massa reduzida está sujeita a uma aceleração, que pode ser escrita a partir da eq.(2.8),

$$\vec{a} = -\frac{GM}{r^2}\hat{r} \tag{2.38}$$

sendo assim, o produto $\vec{a} \times \vec{l}$ nos leva a

$$\vec{a} \times \vec{l} = -\frac{GM}{r^2}\hat{r} \times \left(\mu r^2 \hat{r} \times \frac{d\hat{r}}{dt}\right) = -GM\mu\hat{r} \times \left(\hat{r} \times \frac{d\hat{r}}{dt}\right). \tag{2.39}$$

Aplicando a identidade vetorial:

$$\vec{A} \times \left(\vec{B} \times \vec{C}\right) = \left(\vec{A}.\vec{C}\right)\vec{B} - \left(\vec{A}.\vec{B}\right)\vec{C} \tag{2.40}$$

o que nos leva a

$$\vec{a} \times \vec{l} = -GM\mu\left[\left(\hat{r}.\frac{d\hat{r}}{dt}\hat{r} - (\hat{r}.\hat{r})\frac{d\hat{r}}{dt}\right)\right]. \tag{2.41}$$

Sabendo que $\hat{r}.\hat{r} = 1$ e

$$\frac{d}{dt}(\hat{r}.\hat{r}) = 2\hat{r}\frac{d\hat{r}}{dt} = 0$$

escreve-se

$$\frac{d\vec{v}}{dt} \times \vec{l} = GM\mu\frac{d\hat{r}}{dt}, \tag{2.42}$$

que é o produto vetorial da aceleração e o momento angular do sistema. Observe que é possível ainda reescrever a eq.(2.42) na forma

$$\frac{d\vec{v}}{dt} \times \vec{l} + \vec{v} \times \frac{d\vec{l}}{dt} = GM\mu\frac{d\hat{r}}{dt}, \tag{2.43}$$

pois pela eq.(2.14) a derivada do momento angular em relação ao tempo é nulo, o que leva a

$$\frac{d}{dt}(\vec{v} \times \vec{l}) = \frac{d}{dt}(GM\mu\hat{r}) \tag{2.44}$$

integrando ambos os lados indefinidamente, resulta em

$$\vec{v} \times \vec{l} = GM\mu\hat{r} + \vec{D} \tag{2.45}$$

onde $\vec{D}$ é um vetor constante que está na mesma direção que $\vec{v} \times \vec{l}$ e $\hat{r}$.

Multiplicando escalarmente $\vec{r} = r\hat{r}$ a eq.(2.45), pode-se escrever

$$\vec{r}(\vec{v} \times \vec{l}) = GM\mu r\hat{r}.\hat{r} + \vec{D}.\vec{r}. \tag{2.46}$$

A partir da identidade vetorial:

$$\vec{A}.(\vec{B} \times \vec{C}) = (\vec{A} \times \vec{B}).\vec{C} \tag{2.47}$$

é possível escrever

$$(\vec{r} \times \vec{v}).\vec{l} = GM\mu r + rDcos(\theta). \tag{2.48}$$

Note que pela eq.(2.35), teremos

$$\frac{l^2}{\mu} = GM\mu r\left(1 + \frac{Dcos(\theta)}{GM\mu}\right) \tag{2.49}$$

em que θ é o ângulo da massa reduzida a partir do periélio.

Segundo Carroll e Ostlie (2017), pode-se definir

$$e = \frac{D}{GM\mu} \tag{2.50}$$

logo,

$$r = \frac{l^2/\mu^2}{GM(1 + ecos(\theta))} \tag{2.51}$$

que é análoga a eq.(2.1). Comparando a eq.2.1 com eq.(2.51), concluímos que

$$\frac{l^2}{GM\mu^2} = a(1 - e^2) \therefore l = \mu\sqrt{GMa(1 - e^2)} \tag{2.52}$$

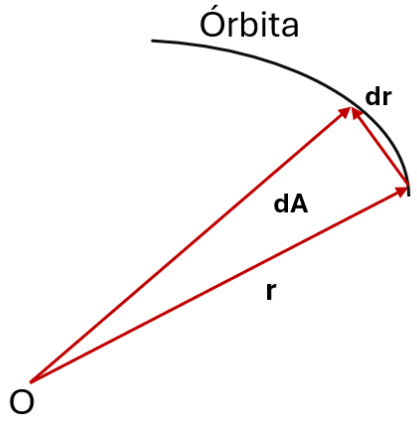
que é válido para órbitas fechadas.

2.6.2 Dedução da 2º lei de Kepler

De acordo com Nussenzveig (2013), se observarmos a fig.12, podemos escrever o infinitésimo dA como

$$dA = \frac{1}{2} \left| \vec{r} \times d\vec{r} \right| . \tag{2.53}$$

Figura 12 – Representação do infinitésimo de área dA varrida.



Fonte: Adaptado de Nussenzveig (2013).

A taxa de variação de dA em relação ao tempo é

$$\frac{dA}{dt} = \frac{1}{2} \left| \vec{r} \times \frac{d\vec{r}}{dt} \right| = \frac{1}{2} \left| \vec{r} \times \vec{v} \right| = \frac{1}{2m} \left| \vec{r} \times \vec{p} \right| \tag{2.54}$$

logo,

$$\frac{dA}{dt} = \frac{|l|}{2m} \tag{2.55}$$

que é a velocidade areal, ou seja, a área varrida pelo raio que liga o planeta ao Sol. Como foi visto, essa quantidade é *constante*, de acordo com a eq.(2.2) e eq.(2.14). Sendo assim,

$$\frac{dA}{dt} = \frac{|l|}{2m} = constante. \tag{2.56}$$

A área varrida pela linha que conecta um planeta ao Sol em um dado intervalo de tempo é constante. Isto é precisamente a segunda lei de Kepler. O que significa que, para varrer a mesma área em um tempo igual, o planeta precisa se mover mais rápido quando está mais próximo do Sol (periélio) e mais lentamente quando está mais distante (afélio). Como o momento angular $\vec{l}$ é uma grandeza conservada, sua direção, módulo e sentido não se modificam com o tempo. Isso significa que a órbita do planeta não oscila, permanecendo sempre no mesmo plano em torno do corpo central.

2.6.3 Dedução da 3º lei de Kepler

De acordo com Winterle e Steinbruch (2000), a equação que descreve uma elipse em coordenadas cartesianas é dada por:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{2.57}$$

em que a e b são, respectivamente, os semieixos maior e menor. Logo, é possível fazer

$$y = b\sqrt{1 - \frac{x^2}{a^2}} = \frac{b}{a}\sqrt{a^2 - x^2}. \tag{2.58}$$

Sendo assim, encontra-se a área da elipse a partir de:

$$A_{elipse} = \int y dx = \frac{b}{a}\int \sqrt{a^2 - x^2} dx. \tag{2.59}$$

A partir de uma substituição trigonométrica simples em que $x = asen(\theta)$ e $dx = acos(\theta)d\theta$, escreve-se

$$A_{elipse} = \frac{b}{a}\int_0^{2\pi} acos(\theta)\sqrt{a^2 - a^2 sen^2(\theta)} d\theta \tag{2.60}$$

colocando a^2 em evidência, a integral torna-se

$$A_{elipse} = \frac{b}{a}a^2\int_0^{2\pi} cos(\theta)\sqrt{1 - sen^2(\theta)}. \tag{2.61}$$

A partir da identidade trigonométrica $cos^2(\theta) = 1 - sen^2(\theta)$, escreve-se

$$A_{elipse} = ba\int_0^{2\pi} cos^2(\theta) d\theta. \tag{2.62}$$

Resolvendo a integral, é possível substituir $cos^2(\theta) = 1/2(1 + sen^2(\theta))$. Sendo assim,

$$A_{elipse} = \frac{1}{2}ba\int_0^{2\pi} (1 + sen(2\theta))\, d\theta = \frac{1}{2}ba\left[\theta + \frac{1}{2}sen(2\theta)\right]_0^{2\pi} \tag{2.63}$$

$$\therefore A_{elipse} = ba\pi \tag{2.64}$$

note que se $b = a = r$ a área da elipse torna-se a área de um círculo de raio r:

$$A_{crculo} = \pi r^2. \tag{2.65}$$

Observe que a eq.(2.56) pode ser escrita em função de μ, que nos leva a

$$\frac{dA}{dt} = \frac{1}{2}\frac{l}{\mu} \tag{2.66}$$

logo,

$$dA = \frac{1}{2}\frac{l}{\mu}dt = \frac{1}{2}\frac{l}{\mu}\int_0^T dt \tag{2.67}$$

$$\therefore A = \frac{1}{2}\frac{l}{\mu}T \tag{2.68}$$

que é a área da elipse. Comparando a eq.(2.64) com eq.(2.68), escreve-se

$$\pi ab = \frac{1}{2}\frac{l}{\mu}T \tag{2.69}$$

elevando ambos os lados ao quadrado, tem-se

$$\pi^2 a^2 b^2 = \left(\frac{1}{2}\frac{l}{\mu}T\right)^2 = \frac{1}{4}\frac{l^2}{\mu^2}T^2. \tag{2.70}$$

Sendo assim, isolando T^2, escreve-se

$$T^2 = \frac{4\mu^2\pi^2 a^2 b^2}{l^2}. \tag{2.71}$$

De acordo com Kepler e Saraiva (2004), é possível escrever

$$b^2 = a^2(1 - e^2), \tag{2.72}$$

dessa forma, combinando a eq.(2.68), eq.(2.72) e eq.(2.52), tem-se

$$T^2 = \frac{4\mu^2\pi^2 a^2 a^2(1-e^2)}{\mu^2 GMa(1-e^2)} = \frac{4\pi^2}{GM}a^3 \tag{2.73}$$

em que M é a massa do sistema, ou seja:

$$T^2 = \frac{4\pi^2}{G(m_1 + m_2)}a^3 \tag{2.74}$$

que é a terceira lei de Kepler definida na seção 2.3.

2.7 POTENCIAL EFETIVO

De acordo com Nussenzveig (2013), a força gravitacional é conservativa, ou seja, a energia total não varia com o tempo. Sendo assim, podemos escrever:

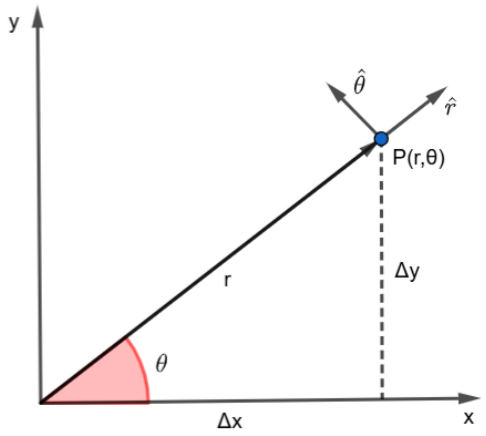
$$E = K + U = Constante \tag{2.75}$$

onde E é a anergia mecânica, K é a energia cinética e U a energia potencial gravitacional, logo

$$E = \frac{1}{2}mv^2 - \frac{GMm}{r}. \tag{2.76}$$

Observe que podemos reescrever a velocidade na eq.(2.76) em coordenadas polares,

Figura 13 – Representação dos elementos de um ponto em coordendas polares.



Fonte: Próprio autor.

A partir da figura 13, podemos extrair as seguintes relações:

$$\begin{cases} x = rcos(\theta) \\ y = rsen(\theta) \end{cases}. \tag{2.77}$$

Sabendo que $x^2 + y^2 = r^2$, o que nos permite escrever

$$\vec{r} = x\hat{i} + y\hat{j} \therefore \vec{r} = r\hat{r}, \tag{2.78}$$

em que $\hat{r} = cos(\theta)\hat{i} + sen(\theta)\hat{j}$. Logo se fizermos a derivada de $\vec{r}$ em relação ao tempo, iremos obter a velocidade v:

$$\vec{v} = \frac{d\vec{r}}{dt} = \frac{dr}{dt}\hat{r} + r\frac{d\hat{r}}{dt}, \tag{2.79}$$

em que podemos escrever $d\hat{r}/dt$ utilizando a regra da cadeia

$$\frac{d\hat{r}}{dt} = \frac{d\hat{r}}{d\theta}\frac{d\theta}{dt}, \tag{2.80}$$

em que

$$\frac{d\hat{r}}{d\theta} = -sen(\theta)\hat{i} + cos(\theta)\hat{j} = \hat{\theta} \tag{2.81}$$

$$\therefore \frac{d\hat{r}}{dt} = \frac{d\theta}{dt}\hat{\theta} = \dot{\theta}\hat{\theta}. \tag{2.82}$$

Com isso, podemos escrever

$$\vec{v} = \dot{r}\hat{r} + r\dot{\theta}\hat{\theta}. \tag{2.83}$$

Note que a velocidade possui componentes na direção radial e angular. Logo, a equação da velocidade pode ser escrita como

$$\vec{v} = v_r\hat{r} + v_\theta\hat{\theta} \Rightarrow \begin{cases} v_r = \dot{r} \\ v_\theta = r\dot{\theta} \end{cases}$$

$$\therefore v^2 = \dot{r}^2 + r^2\dot{\theta}^2. \tag{2.84}$$

Sendo assim, podemos substituir a eq. na eq.(2.84) na (2.76), o que nos leva a

$$E = \frac{1}{2}m(\dot{r}^2 + r^2\dot{\theta}^2) + U(r),$$

$$\therefore E = \frac{1}{2}m\dot{r}^2 + \frac{1}{2}mr^2\dot{\theta}^2 - \frac{GMm}{r}. \tag{2.85}$$

Segundo Nussenzveig (2013), podemos destacar que $mr^2\dot{\theta}$ é o que definimos como o momento angular l do planeta em função da velocidade angular $\dot{\theta}$, ou seja:

$$\vec{l} = \vec{r} \times \vec{p} = rpsen(\theta). \tag{2.86}$$

Levando em consideração que $r \perp p$ e que $p = mv$, temos

$$l = rmv_\theta,$$

$$\therefore l = mr^2\dot{\theta}. \tag{2.87}$$

Com isso, podemos isolar $\dot{\theta}$ e elevar ambos os lados ao quadrado, o que nos leva a

$$\left(\frac{l}{mr^2}\right)^2 = \dot{\theta}^2 \tag{2.88}$$

que, substituindo na eq.(2.85), resulta em

$$E = \frac{1}{2}m\dot{r}^2 + \frac{1}{2}\frac{l^2}{mr^2} - \frac{GMm}{r}, \tag{2.89}$$

em que E é a energia mecânica total. É muito importante se atentar ao comportamento da energia mecânica, pois se tivermos $E < 0$ temos um estado ligado, assim, a eq.(2.1) descreverá uma elipse. Porém, se tivermos $E = 0$ ou $E > 0$, trata-se de estado de espalhamento, ou seja, teremos uma parábola e uma hipérbole, respectivamente (THORNTON; MARION, 2011). Na tabela 2, temos a relação entre o comportamento das órbitas, as energias e a excentrencidade.

Veja que é possível escrever:

$$E = \frac{1}{2}m\dot{r}^2 + V_{eff}(r) \tag{2.90}$$

onde,

$$V_{eff}(r) = \frac{1}{2}\frac{l^2}{mr^2} - \frac{GMm}{r}, \tag{2.91}$$

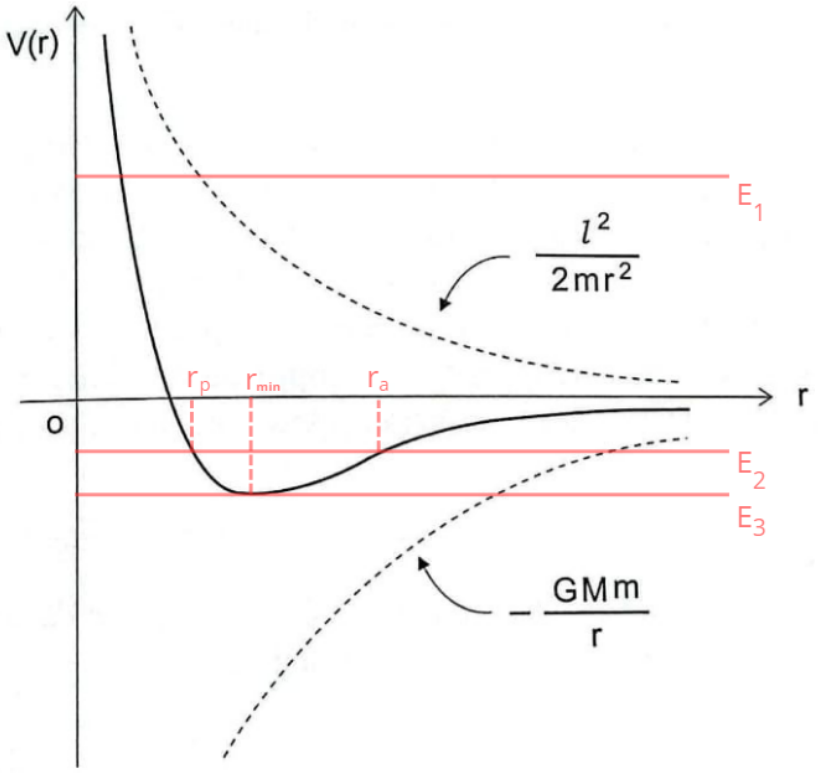
em que $V_{eff}(r)$ é o *potencial efetivo* e o termo $l^2/2mr^2$ é identificado como energia potencial centrífuga. É possível observar na fig.(14) o comportamento do potencial efetivo, a energia potencial e a energia potencial centrífuga. Note que quando $r \to \infty$ o potencial tente a zero. Além disso, é possível observar na fig.(14) as energias E_1, E_2 e E_3. A energia E_1 representa um estado de espalhamento, pois $E_1 > 0$. Já a energia E_2 trata-se de um estado ligado, em que a partícula oscila entre r_p e r_a, que são justamente o periélio e afélio de uma órbita elíptica. Por fim, a energia E_3 representa uma partícula em movimento circular com raio orbital $r = r_{min}$.

Tabela 2 – Classificação das órbitas em termos da E e e.

Órbita	Energia	Excentricidade
Hipérbole	$E > 0$	$e > 1$
Parábola	$E = 0$	$e = 1$
Elipse	$V_{min} < E < 0$	$0 < e < 1$
Círculo	E $= V_{min}$	$e = 0$

Fonte: Thornton e Marion (2011).

Figura 14 – Representação gráfica do potencial efetivo.



Fonte: Adaptdado de Neto (2020).

2.8 EQUAÇÃO VELOCIDADE

É possível escrever a energia mecânica em função do semi-eixo maior da órbita. A fim de demonstrar isso, retoma-se a eq.(2.1), porém agora na forma Kepler e Saraiva (2004),

$$r = \frac{P}{1 + ecos(\theta)} \tag{2.92}$$

em que P é chamado de *semi-lactus rectum*,

$$\frac{l^2}{GM} = P. \tag{2.93}$$

Comparando a eq.(2.93) com a eq.(2.1), é possível escrever

$$l^2 = GMa(1 - e^2) \because P = \frac{l^2}{GM} = a(1 - e^2) \tag{2.94}$$

em que

$$l^2 = GMa(1 - e)(1 + e). \tag{2.95}$$

Na eq.(2.14) a segunda lei de Kepler foi escrita em termos do momento angular. Além disso, chegou-se a conclusão de que a velocidade do corpo varia entre o periélio e o afélio. Sendo assim, considerando o momento angular no periélio:

$$l_p = r_p v_p \tag{2.96}$$

em que, observando a figura 5, escreve-se o raio vetor no períelio como,

$$r_p = a - ae = a(1 - e). \tag{2.97}$$

Logo, isolando v_p na eq.(2.96) e substituindo substituindo r_p, tem-se

$$v_p = \frac{l_p}{a(1 - e)}. \tag{2.98}$$

Substituindo v_p e r_p na eq.(2.76), leva a

$$E = \frac{1}{2} m \frac{l_p^2}{a^2(1 - e)^2} - \frac{GMm}{a(1 - e)}. \tag{2.99}$$

Substituindo a eq.(2.95) em eq.(2.99),

$$E = \frac{GMma(1 - e)(1 + e)}{2a^2(1 - e)^2} - \frac{GMm}{a(1 - e)}$$

$$E = \frac{GMm}{a(1 - e)} \left(\frac{1 + e}{2} - 1 \right) = -\frac{GMm}{a(1 - e)} \frac{(1 - e)}{2}$$

$$\therefore E = -\frac{GMm}{2a}, \tag{2.100}$$

ou seja, existe uma relação entre a energia do sistema e o semi-eixo maior.

Das equações 2.76 e 2.100, escreve-se

$$-\frac{GMm}{2a} = \frac{1}{2}mv^2 - \frac{GMm}{r}$$

$$-\frac{GM}{2a} + \frac{GM}{r} = \frac{1}{2}v^2$$

$$\therefore v = \sqrt{GM\left(\frac{2}{r} - \frac{1}{a}\right)} \tag{2.101}$$

que é a equação da velocidade do planeta. Note que v depende apenas de r e a. Pode-se generalizar a equação da velocidade, considerando-se o movimento de dois corpos em torno de um centro de massa. Assim, uma maneira alternativa de escrever a eq.(2.100) é:

$$E = -\frac{GM\mu}{2a} \tag{2.102}$$

o que leva a

$$v = \sqrt{G(m_1 + m_2)\left(\frac{2}{r} - \frac{1}{a}\right)} \tag{2.103}$$

que é a equação da velocidade generalizada (CARROLL; OSTLIE, 2017). A eq.(2.48) é uma forma simples e aproximadamente equivalente de escrevê-la.

3 METODOLOGIA

O presente trabalho trata-se de uma pesquisa com caráter exploratório, e posterior utilização de método experimental em ambiente virtual. A seguir estão listados os passos desenvolvidos durante o curso da pesquisa.

3.1 LEVANTAMENTO BIBLIOGRÁFICO

Foi feito levantamento bibliográfico sobre as bases da mecânica envolvida no movimento dos planetas (este levantamento encontram-se nas seções 2.2 a 2.8);

3.2 EQUAÇÕES DO MOVIMENTO

Obtenção das equações de movimento que definem a órbita de um planeta (equações 2.1 e 2.23);

$$\begin{cases} r = \frac{a(1-e^2)}{(1+ecos(\theta))} & \textbf{(posição)} \\ E = \frac{1}{2}m\dot{r}^2 + \frac{1}{2}\frac{l^2}{mr^2} - \frac{GMm}{r} & \textbf{(energia)} \end{cases};$$

3.3 SIMULAÇÃO DAS ÓRBITAS PLANETÁRIAS EM LINGUAGEM DE PROGRAMAÇÃO EM PYTHON

Foi realizada a transposição das equações de movimento de um planeta para a linguagem de programação em Python. Para construção do código utilizou-se o Google Colab, uma ferramenta online popular e gratuita, oferecida pelo Google. O Colab funciona como um notebook digital, onde é possível escrever e executar códigos diretamente no navegador, sem precisar instalar nenhum software no computador do usuário. O Colab está integrado ao Google Drive, o que facilita muito o acesso e o compartilhamento de projetos.

3.3.1 Pacotes e declaração de variáveis

O código é iniciado da seguinte maneira:

```
>>> !apt-get install libx264-155
>>> !pip install pyffmpeg
>>> import matplotlib.pyplot as plt
>>> import matplotlib.animation as animation
>>> import numpy as np
```

Ou seja, inicialmente instala-se alguns pacotes no ambiente, como *!apt-get install libx264-155*, que instala uma biblioteca necessária para gerar vídeos com o codec H264. O

!pip install pyffmpeg, que instala a biblioteca *pyffmpeg*, relacionada à animação. Importamos também as bibliotecas *matplotlib.pyplot* para visualização, *matplotlib.animation* para criação de animações e *numpy* para operações numéricas.

Em seguida, adicionam-se:

```
>>> plt.rcParams['animation.ffmpeg_path'] = '/usr/bin/ffmpeg'
>>> plt.rcParams['animation.codec'] = 'h264'
>>> from IPython.display import HTML,
```

no qual utilizou-se a função *HTML* do *IPython.display* para gerar a aniamação no Google Colab e *plt.rcParams* para personalizar a aparência dos gráficos. Com isso, pensando em simular incialmente apenas a órbita da Terra, define-se as constantes e parâmetros necessários:

```
# Constantes

>>> G = 6.67430e-11
>>> M_sol = 1.98847e30
>>> AU = 1.495978707e11

# Parâmetros da Terra
>>> M_terra = 5.9722e24
>>> A = 1.0 * AU
>>> epsilon = 0.0167
>>> theta = -np.pi / 2,
```

em que G constante gravitacional, a M_{Sol} massa do Sol, a M_{Terra} massa da Terra, e excentricidade da órbita da Terra, A o semi-eixo maior, θ uma posição inicial e define-se a escala da simulação utilizando a Unidade Astronômica (UA) como referência.

3.3.2 Posição, velocidade, energia e aceleração

Dessa forma, escreve-se a equação 2.1 no código,

```
>>> r = A * (1 - epsilon**2) / (1 + epsilon * np.cos(theta))
```

que é a equação da órbita. Como se está trabalhando em coordendas polares, atualiza-se os valores da posição a cada intervalo de tempo, escrevemos também:

```
>>> x = r * np.cos(theta)
>>> y = r * np.sin(theta)
```

De acordo com a eq.(2.103), é possível escrever a equação da velocidade do sistema em função do semi-eixo maior a e r. Sendo assim, escrevemos

```
>>> v = np.sqrt(G *( M_sol+M_terra) * (2 / r - 1 / A))
```

que é a equação velociade do sistema Terra-Sol. Após inserir a equação 2.103 no código, escreve-se a equação 2.76(energia) no seguinte formato:

```
>>> E = 0.5*M_terra*(vx**2+vy**2)-G*M_sol*M_terra/r
```

em que

```
>>> vx = v * np.cos(theta+np.pi/2)
>>> vy = v * np.sin(theta+np.pi/2)
```

em que adicionamos $\pi/2$ a θ para garantir que o vetor velocidade seja perdendicular ao vetor posição no periélio e afélio.

Importante é definir um passo de tempo para a simulação:

```
# Simulação

>>> dt = 0.001 * 365.25 * 24 * 3600  # Passo de tempo em segundos

>>> t_end = 10 * 365.25 * 24 * 3600  # Tempo total da simulação em segundos

>>> t = np.arange(0, t_end, dt)
>>> x_pos = []
>>> y_pos = [].
```

Com isso, é possível cirar um *loop* em que calcula-se a aceleração gravitacional e a atualização das posições e velocidades. A equação da aceleração gravitacional foi escrita, a partir da eq.(2.38),

```
# Calcula a aceleração gravitacional

>>> r_vec = np.array([x, y])  #vetor com valores de x e y
>>> a = -G * M_sol * r_vec / np.linalg.norm(r1_vec)**3 #aceleração

# Atualiza a velocidade e a posição

>>>    vx += a[0] * dt #velocidade na componente x
>>>    vy += a[1] * dt #velocidade na componente y
```

```
>>>    x1 += vx * dt  #posição em x
>>>    y1 += vy * dt  #posição em y
>>>    x1_pos.append(x1) #adicionar valores x
>>>    y1_pos.append(y1) #adiconar valores y
```

3.3.3 Plotagem das órbitas

Dessa meneira, a visualização dos resultados é crucial para a compreensão da simulação. Asssim, antes de executar o código inserimos algumas linhas para configurar nossa animação. Ou seja, a última parte do código gera a animação da órbita e o valor da energia mecânica:

```
>>> print("Energia da órbita da Terra",E)  #Imprimir o valor da energia

# Plotagem e Animação
%matplotlib inline

>>> plt.style.use('dark_background') #fundo preto
>>> fig, ax = plt.subplots() #configuração da figura
>>> ax.set_aspect('equal') #ajuste na proprorção nos eixos x e y
>>> ax.set_xlim([-2.5 * AU, 2.5 * AU]) #limites no eixo x
>>> ax.set_ylim([-2.5 * AU, 2.5 * AU]) #limites no eixo y
>>> ax.set_xlabel('Distância radial (m)') # definição do eixo x
>>> ax.set_ylabel('Distância radial (m)') # definição do eixo y


>>> line, = ax.plot([], [], 'b-', label='Terra') #exibir rastro
>>> sol, = ax.plot([0], [0], 'yo', markersize=10, label='Sol') # adicionando o sol

>>> ax.legend() # exibir legenda

>>> def animate(i):
    line1.set_data(x_pos[:i], y_pos[:i])
    return line, sol # retornando o sol também

>>> ani = animation.FuncAnimation(fig,
animate, frames=len(t), interval=10, blit=True) #animação

# Gerando um vídeo HTML5
```

```
>>> html = ani.to_html5_video()

# Exibindo a animação no Colab

>>> HTML(html).
```

Após se observar que a órbita da Terra foi gerada com sucesso, passa-se a inserir os dados de outros planetas como Mercúrio, Vênus e Marte (a tabela 3 mostra alguns dados importantes). Assim, apenas repete-se o processo para cada planeta. O código completo pode ser encontrado no Apêndice A. Um detalhe importante é não esquecer de ajustar o passo de tempo dos outros planetas, uma boa estimativa seria:

```
# Passos de tempo individuais

>>> dt_mercurio = 0.0005 * 365.25 * 24 * 3600  # Menor dt para Mercúrio
>>> dt_venus = 0.0008 * 365.25 * 24 * 3600
>>> dt_terra = 0.001 * 365.25 * 24 * 3600
>>> dt_marte = 0.0012 * 365.25 * 24 * 3600
```

3.3.4 Análise do Potencial efetivo da Terra

A fim de simular o gráfico do potencial efetivo da Terra, foi utilizada a equação 2.25, em uma nova seção de códigos. Foram definidas as mesmas bibliotecas da seção 3.4, constantes e parâmetros da Terra. Dessa forma, escreve-se uma nova função para descrever a equação 2.25:

```
# Cálculo do momento angular da Terra

>>> L = M_terra * np.sqrt(G * M_sol * A * (1 - epsilon**2)) #momento angular

# Função para calcular o potencial efetivo

>>> def potencial_efetivo(r):
    U_gravitacional = -G * M_sol * M_terra / r #Potencial gravitacional
    U_centrifugo = L**2 / (2 * M_terra * r**2) #potencial centrífugo
    return U_gravitacional + U_centrifugo  #retorna a soma dos potenciais
```

3.3.5 Plotagem do potencial efetivo

Foram necessários alguns ajustes para plotar separadamente a energia potencial centrífuga, a energia potencial gravitacional e o potencial efetivo. De maneira similar a plotagem das órbitas, é possível implementar:

```
# Plota o gráfico do potencial efetivo

>>> plt.figure(figsize=(10, 6)) #dimensão da figura
>>> plt.plot(r_values, Ueff_values, label='Potencial Efetivo') #plota potencial eff
>>> plt.plot(r_values, U_centrifugo, label='Energia centrífuga') #potencial cent.
>>> plt.plot(r_values, U1_values, label='Potencial gravitacional') #potencial grav
>>> plt.xlabel('r (m)')  #eixo x é r em metros
>>> plt.ylabel('V (J)') #eixo y é V em joules
>>> plt.title('Potencial Efetivo da Terra') #título
>>> plt.savefig('potencial_terra.png', dpi = 500) #salvar figura em qualidade de 500 d
>>> plt.legend() #exibir legenda
>>> plt.grid(True) #grade
>>> plt.show() #executar o gráfico
```

Para plotar separadamente os potenciais, basta inserir "#"na linha que deseja não plotar temporariamente, como por exemplo:

```
#plt.plot(r_values, U_centrifugo, label='Energia Potencial centrífuga')

#plt.plot(r_values, U1_values, label='Energia Potencial')
```

ou seja, a linha vira um comentário no código. O código completo encontra-se no apêndice B.

3.4 SIMULAÇÃO DAS ÓRBITAS PLANETÁRIAS EM VPYTHON

Para simular as órbitas planetárias em Vpython, foi utilizado o GlowScript, uma plataforma online gratuita que permite a criação de animações 3D através da programação em Python. Para ter acesso ao Glowscript é simples, basta fazer uma conta ou logar com a própria conta Google.

Sendo assim, o caráter do código é semelhante ao apresentado no apêndice A e discutido na seção 3.4, a única diferença é que não é necessário importar nenhuma biblioteca. Além disso, não foi utilizado a constante gravitacional G. Aqui utilizou-se a constante gravitacional gaussina k. De acordo com Almeida et al. (2016), pode-se escrever o produto

da constante gravitacional pela massa do Sol como:

$$GM_{Sol} = k^2 \tag{3.1}$$

logo,

$$k = 0.01720209395 AU^{3/2} dia^{-1} M_{\odot}^{1/2}. \tag{3.2}$$

Outras diferenças, por exemplo, é a criação dos planetas. Para posicionar a Terra deve-se criar uma esfera com determinada posição, raio e cor. Isso se repete para cada planeta criado:

```
>>> Earth = sphere(pos=vector(r3*cos(theta3), r3*sin(theta3), 0),
radius=0.05, color=color.blue, make_trail=True)
```

```
>>> Mars = sphere(pos=vector(r4*cos(theta4), r4*sin(theta4), 0),
radius=0.03, color=color.red, make_trail=True)
```

A estrutura dos cálculos é a mesma. O código completo encontra-se no apêndice C, no qual simula as órbitas de todos os planetas do sistema solar. A tabela 3 contém as massas e as excentricidades dos cinco planetas simulados. Observe que o semi-eixo maior dos planetas estão na tabela 1.

Tabela 3 – Massa e excentricidade dos planetas.

Planeta	Massa	Excentricidade
Mercúrio	$3,302.10^{23}$	$0,20563$
Vênus	$4,8685.10^{24}$	$0,00677$
Terra	$5,9722.10^{24}$	$0,0167$
Marte	$6,417110.^{23}$	$0,0934$
Júpiter	$1,8982.10^{27}$	$0,0484$
Saturno	$5,6834.10^{26}$	$0,0565$
Urano	$8,6813.10^{25}$	$0,0472$
Netuno	$1,02413.10^{26}$	$0,0086$

Fonte: Kepler e Saraiva (2004).

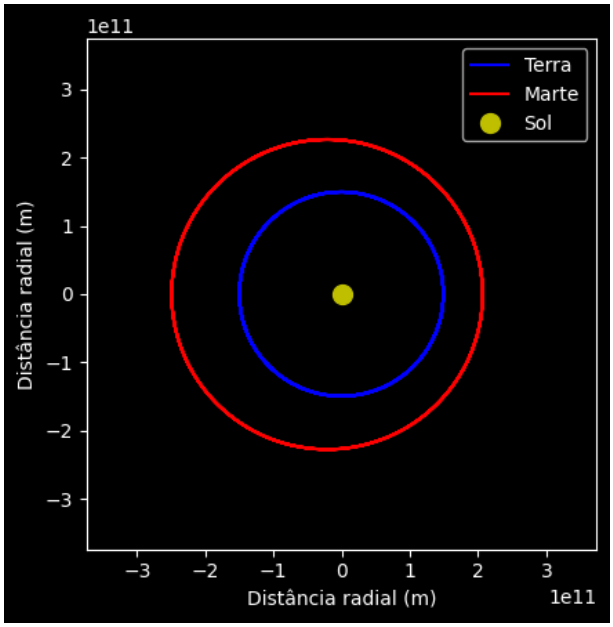
4 RESULTADOS

Apresentam-se aqui os resultados da pesquisa. Primeiramente, mostra-se as órbitas dos planetas a partir da linguagem de programação em Python (via Google Colab) e em Vpython. Em seguida, é apresentado o potencial efetivo da Terra. Ao final, propõe-se um minicurso sobre simulações planetárias.

4.1 ÓRBITAS PLANETÁRIAS EM PYTHON

Como visto na seção 3.3, a montagem do código permitiu simular a órbita da Terra. Assim, utilizando os dados da tabela 3, foi gerado a órbita de Marte (figura 15).

Figura 15 – Simulação da órbita da Terra e Marte.

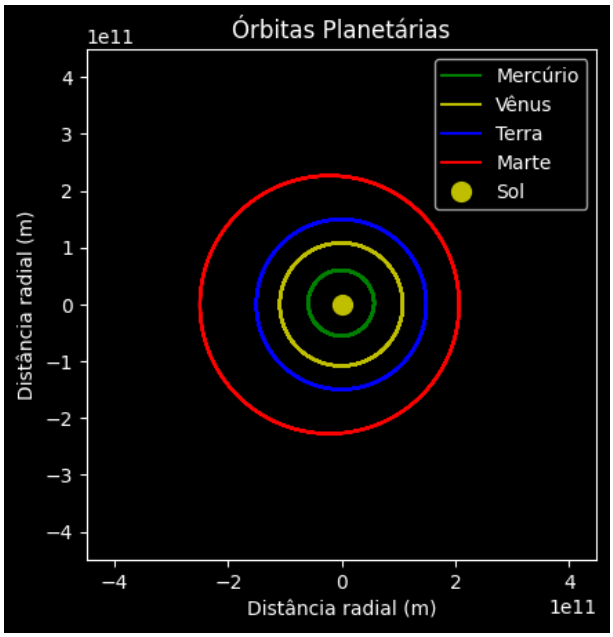


Fonte: Próprio autor.

Ao analisar as órbitas geradas, constata-se que a órbita da Terra se aproxima de um círculo, o que se deve à sua baixa excentricidade. Essa característica pode gerar dúvidas quanto à sua forma real. No entanto, ao comparar a órbita da Terra com a de Marte, cuja excentricidade é mais alta, o leve achatamento da órbita marciana se torna evidente. As órbitas simuladas estão de acordo com o esperado.

De maneira análoga, estendeu-se a simulação, inserindo os planetas Mercúrio e Vênus (figura 16). Conforme a Tabela 1, entre Mercúrio e Marte, o planeta com o maior semieixo maior é Marte. Essa diferença de distância entre os planetas é evidente na Fig.16, onde a órbita de Marte se destaca. Outro ponto importante é a excentricidade orbital de Mercúrio, a maior entre os planetas (figura 17). Essa característica faz com que o Sol, que

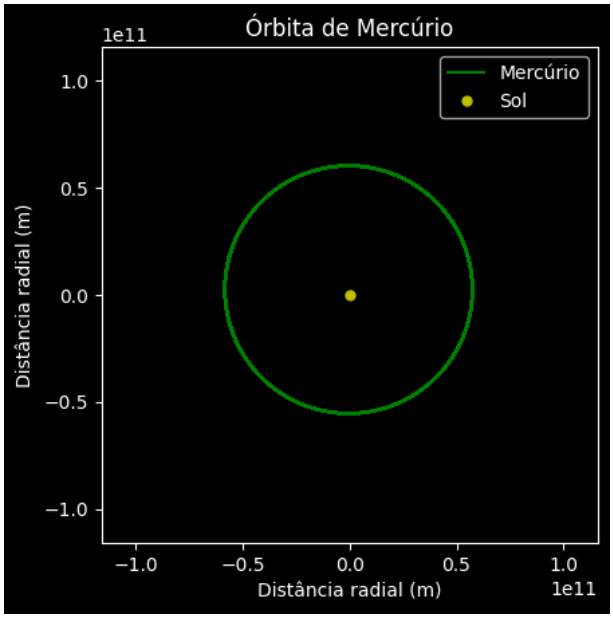
Figura 16 – Simulação das órbitas de Mercúrio, Vênus, Terra e Marte.



Fonte: Próprio autor.

ocupa um dos focos da órbita, esteja relativamente distante do centro, diferentemente do que ocorre na órbita da Terra.

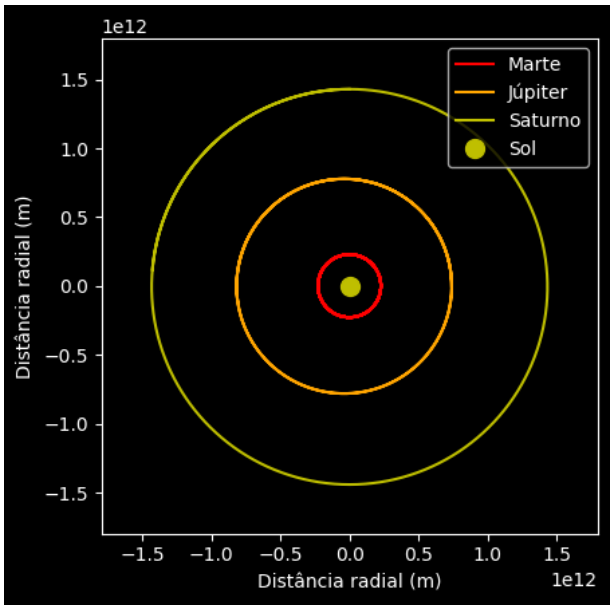
Figura 17 – Simulação da órbita de Mercúrio.



Fonte: Próprio autor.

Com a adição de Júpiter e Saturno ao conjunto de planetas, observou-se que seus semi-eixos maiores se distinguem consideravelmente dos demais, o que dificultaria visualizar os primeiros planetas. Dessa forma, na fig.18. optou-se por direcionar a análise comparativa para as órbitas de Marte, Júpiter e Saturno, buscando identificar suas diferenças.

Figura 18 – Simulação das órbitas de Marte, Júpiter e Saturno.



Fonte: Próprio autor.

Na eq.(2.100), nota-se que a energia da órbita é inversamente proprocional ao semi-eixo maior e diretamente proprocional a massa do planeta. Sendo assim, observamos diferentes ordens de grandezas para as energias. Além disso, pelo que foi visto sobre energia na seção 2.7, para estados ligados, a energia das órbitas devem ser negativas. A tabela 4 mostra as energias dos planetas simulados.

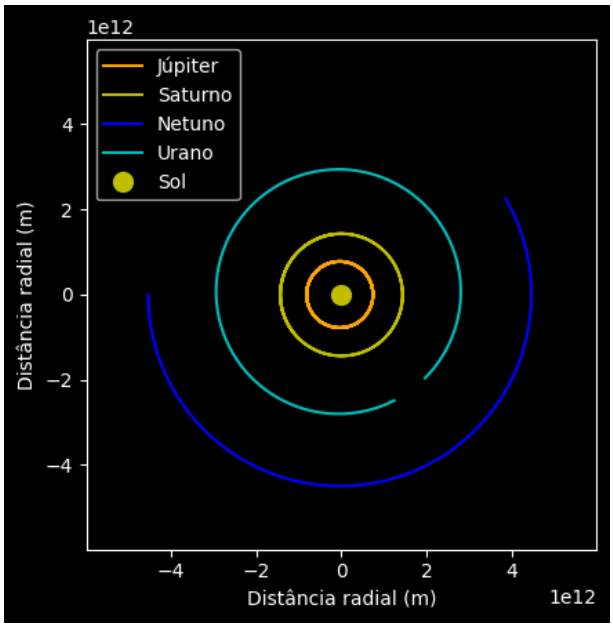
Tabela 4 – Energias obitidas para os planetas simulados.

Planeta	Energia (J)
Mercúrio	$-4,2.10^{32}$
Vênus	$-3,0072.10^{33}$
Terra	$-2,63.10^{33}$
Marte	$-2,404.10^{32}$
Júpiter	$-1,616.10^{35}$
Saturno	$-1,87.10^{34}$
Netuno	$-2,998.10^{33}$
Urano	$-4,156.10^{33}$

Fonte: Próprio autor.

Para finalizar as análises das órbitas planetárias, foi simulado as órbitas de Urano e Netuno. Na fig.(19) é possível observar as distâncias das órbitas de Netuno e Urano em relação a Júpiter. As órbitas ficaram incompletas devido ao tempo de execução muito longo no Google Colab.

Figura 19 – Simulação das órbitas de Júpiter, Saturno, Urano e Netuno.

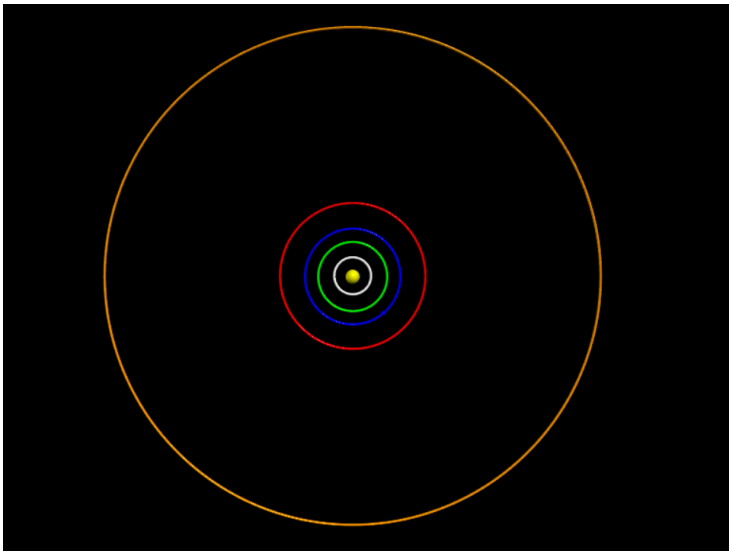


Fonte: Próprio autor.

4.2 ÓRBITAS PLANETÁRIAS EM VPYTHON

As simulações em Vpython são imediatas, ou seja, o tempo de simulação é curto em relação ao Python via Google Colab. Além disso, é possível ajustar o zoom para observar todos os planetas. Dessa forma, foi simulado todos os planetas do sistema solar.

Figura 20 – Simulação do sistema Solar em Vpython. Branco - Mercúrio, verde - Vênus, azul - Terra, vermelho - Marte e laranja - Júpiter.



Fonte: Prórpio autor.

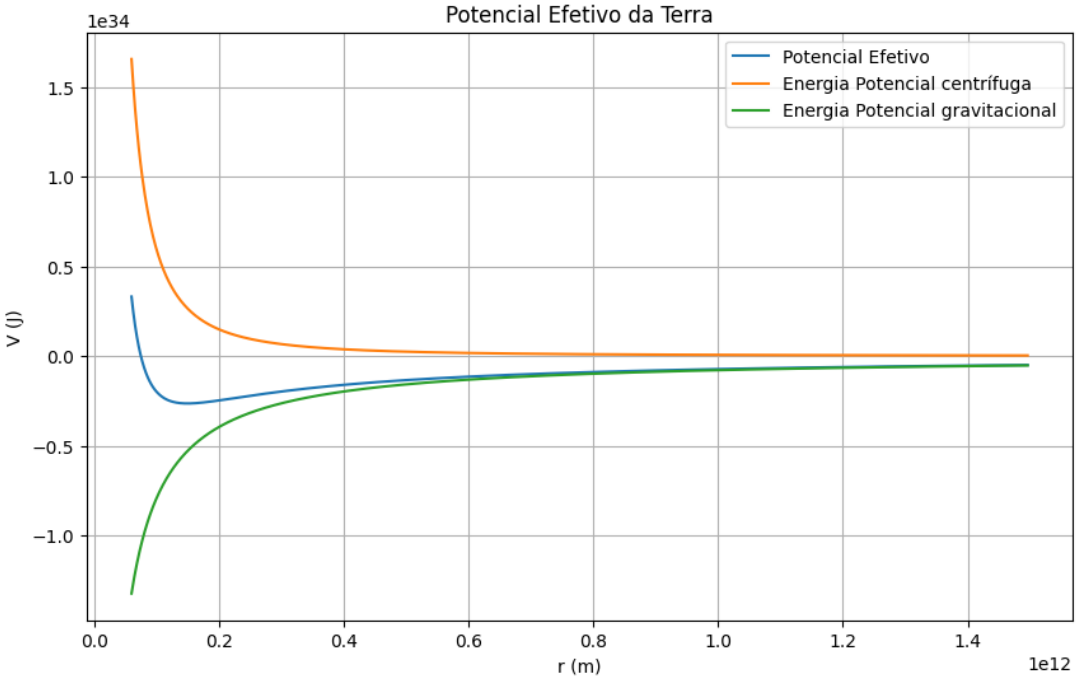
A fig.(20) apresenta uma representação dos cinco primeiros planetas do sistema solar. Através da simulação, é possível explorar os demais planetas utilizando a funcionalidade de zoom.

Para visualizar a simulação web, clique em: **sistema solar**. Através da simulação em Vpython será possível perceber o motivo da fig.(19) não representar a órbita completa de Netuno e Urano. Não trata-se de uma falha no código, mas torna-se uma simulação pesada. A função *rate()*, que altera a quantidade de vezes que o laço é executado por segundo, foi aleterado para *rate(1000000)* e, mesmo assim, a simulação demora um pouco para ser totalmente executada.

4.3 POTENCIAL EFETIVO DA TERRA

A simulação da órbita da Terra permitiu a análise do seu potencial efetivo. Conforme detalhado na Seção 2.7 e expresso na eq.(2.91), o potencial efetivo é a combinação da energia potencial gravitacional e da energia potencial centrífuga.

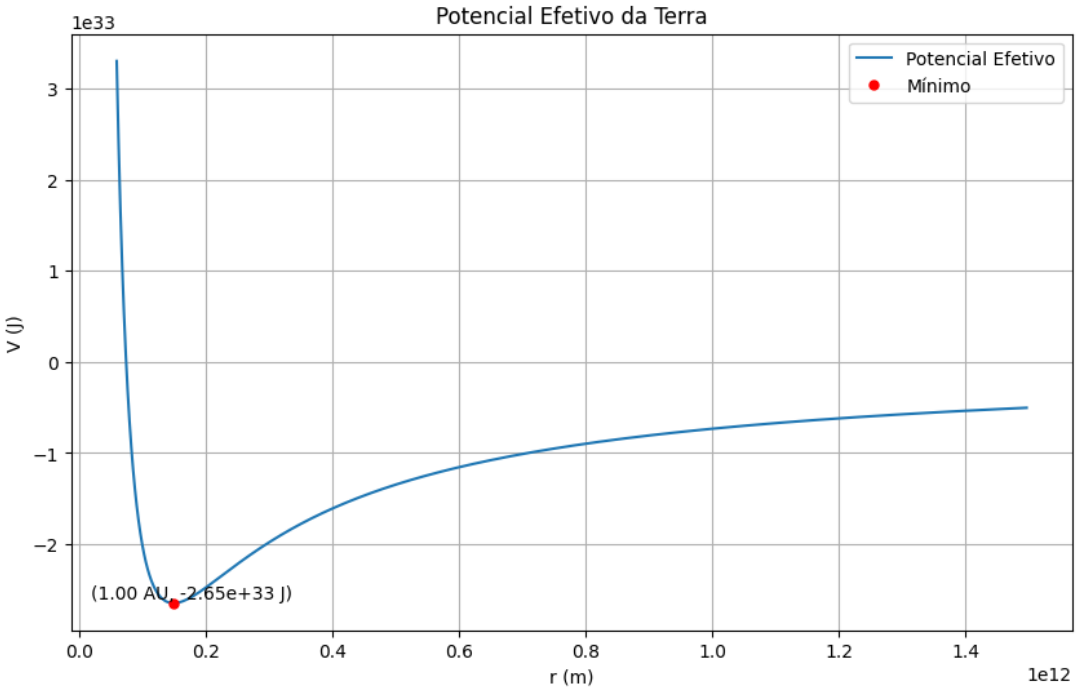
Figura 21 – Potencial efetivo da Terra simulado em Python.



Fonte: Próprio autor.

Note que é possível comparar o resultado da fig.(21) com o comportamento esperado da fig.(14). Ou seja, a fig.(21) está de acordo com o que se espera de um movimento sob força central. Na fig.(22), é possível visualizar o potencial efetivo e identificar o ponto de mínimo, que representa o raio da órbita circular da Terra, bem como o nível de energia associado a essa órbita. Como foi visto na seção 2.8, o r_{min} representa o movimento de uma partícula em movimento circular. Como a órbita da Terra é aproximadamente circular, o resultado obtido para a energia está em concordância com a simulação da órbita terrestre descrita na seção 4.1. O valor calculado é equivalente aos apresentados na tabela 4. O código completo para destacar o ponto mínimo encontra-se no apêndice D.

Figura 22 – Potencial efetivo da Terra isolado, destacando o ponto minímo.

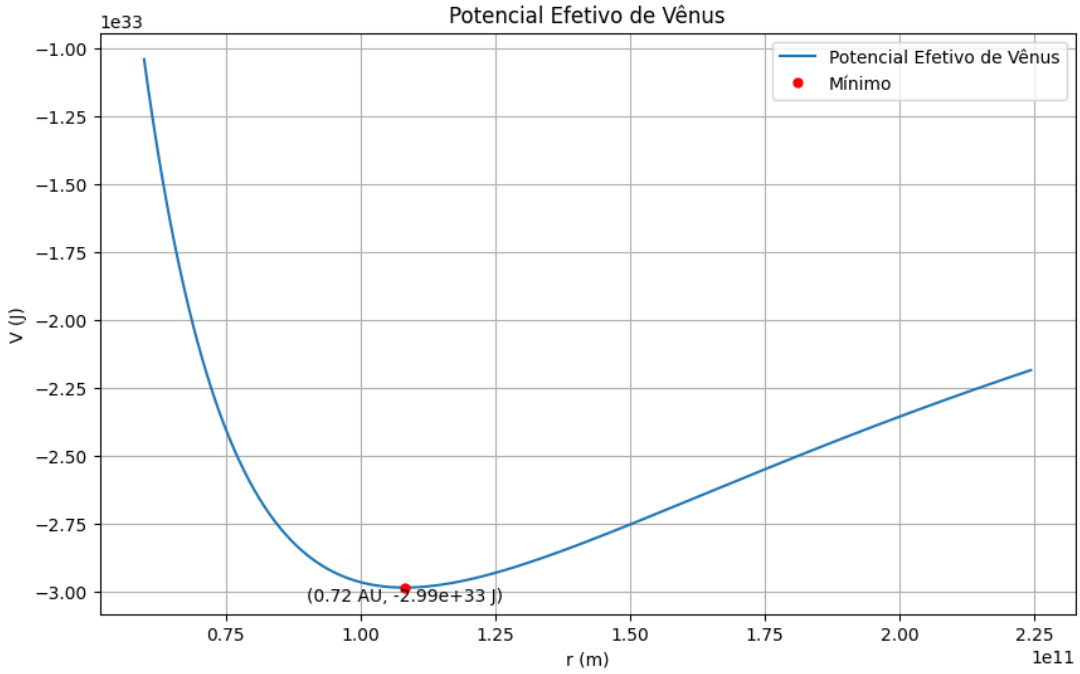


Fonte: Próprio autor.

Com isso, foi simulado támbem o potencial efetivo de Vênus (figura 23), para verificar se teríamos o mesmo comportamento da fig(21) e fig.(14). Tecnicamente é o mesmo código apresentado no apêndice D, porém com os dados de Vênus. Note que o comportamento é similar ao da Terra, assim, todos os planetas do sistema solar terão o potencial efetivo com essas características. Observe na fig.(23) os valores de r_{min} com a energia de Vênus coincidem com o esperado.

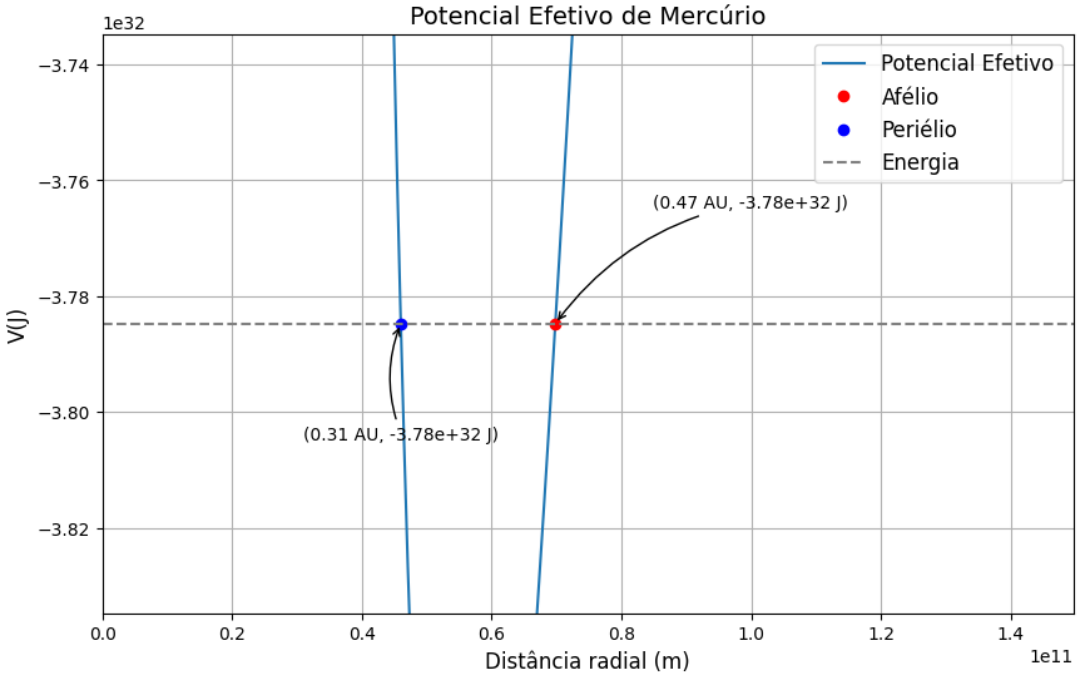
Como a excentricidade de Mércurio é a maior entre os cinco primeiros planetas, foi calculado os pontos r_p e r_a da órbita de mercúrio. Na fig.(22) é possível observar a energia da órbita, que é aproximadamente o que encontramos na tab.(4). O código para o potencial efetivo de mercúrio encontra-se no apêndice E.

Figura 23 – Potencial efetivo de Vênus isolado, destacando o ponto minímo.



Fonte: Próprio autor.

Figura 24 – Representação de r_p, r_a e E no potencial efetivo de mercúrio.



4.4 PLANO DE AULA - MINICURSO

Nesta seção estão apresentados os planos de aula referentes ao minicurso de órbitas planetárias. O minicurso terá dois momentos, o primeiro dia consiste de uma explanação da teoria por trás do estudo de órbitas planetárias, como as leis de Kepler, momento

angular e conservação de energia. O segundo momento será dedicado ao desenvolvimento dos códigos em Python e Vpython. Cada aula terá duração de 2h, totalizando 4h de minicurso.

4.4.1 Plano de aula 1

Tema: Simulação de órbitas planetárias com linguagem de programção em Python.
Duração: 2h
Objetivos:

- Compreender das Leis de Kepler;
- Entender da mecânica celeste;

Conteúdos:

- Evolução dos modelos de sistema solar;
- Leis de Kepler
- Lei da Gravitação Universal, equação da órbita, momento angular, conservação de energia e equação velocidade.

Metodologia:

- Incialmente, será aplicado um questionário para, ao final do minicurso, observar se houve melhora no desenvolvimento do aluno;
- Será questionado: "Como Kepler chegou a conclusão de as órbitas dos planetas do sistema solar são elipses e não círculos?";
- Serão discutidas as histórias, deduções das leis e equações gerais do movimento de corpos sob força central.

Avaliação do processo de ensino e aprendizagem:

- Serão avaliadas as discussões durante a apresentação teórica;
- Avaliação do questionário inicial.

Recursos:

- Quadro branco;
- Pincel;

- projetor;
- Notebook.

Questionário:

1. O que diz a primeira lei de Kepler?
2. O que a segunda lei de Kepler nos diz sobre a velocidade de um planeta em sua órbita?
3. Quais os tipos de órbitas possíveis?
4. Qual o planeta do Sistema Solar tem a órbita com maior excentricidade?
5. Existe alguma relação do momento angular com a segunda lei de Kepler?

4.4.2 Plano de aula 2

Tema: Simulação de órbitas planetárias com linguagem de programção em Python.
Duração: 2h
Objetivos:

- Compreender o básico da linguagem de programação em Python;
- Transpor as equações do movimento planetário para o Python;
- Criar simulações em Python e Vpython.

Conteúdos:

- Introdução a linguagem de programação em Python;
- Aplicação das bibliotecas Matplotlib, Numpy e Vpython;
- Como criar animações no Google Colab;
- Simulando órbitas planetárias em Python e Vpython.

Metodologia:

- Incialmente, será apresentado os primeiros passos para instalação do Google Colab e primeiros passos para programar;
- Serão apresentadas as bibliotecas e recursos necessários para começar a simular as órbitas;
- Serão transpostas as equações deduzidas e discutidas na aula 1 para a o programa;

- Serão simuladas ao menos duas órbitas em Python e Vpython;
- Por fim, serão simulados o potencial efetivo de dois planetas do sistema solar.

Avaliação do processo de ensino e aprendizagem:

- Serão avaliadas as plotagem das órbitas de Marte e Saturno, tanto em Python quanto em Vpython.

Recursos:

- Quadro branco;
- pincel;
- computador;
- data-show;
- internet.

Questionário:

1. O que significa afirmar que o momento angular da órbita é constante?
2. Como a energia da órbita está associada a primeira lei de Kepler? Classifique os tipos de órbitas em relação à energia total do corpo em órbita.
3. Se um planeta tem uma órbita com alta excentricidade, como isso afeta sua velocidade em diferentes pontos da órbita?
4. Qual o significado da energia mecânica de uma órbita ser negativa?
5. Fisicamente, o que corresponde o ponto mínimo da curva do potencial efetivo de uma órbita?

5 CONCLUSÃO

A utilização de simulações em linguagem Python no ensino de Física revela-se uma ferramenta útil na promoção de aulas mais interativas e engajadoras. O desenvolvimento de simulações de órbitas planetárias, por meio da transposição das leis e equações fundamentais do movimento planetário, proporciona aos alunos uma compreensão mais madura e realista do fenômeno. Essa abordagem permite visualizar o movimento dos planetas com base em dados reais, o que contribui para uma experiência de aprendizado mais significativa.

Os resultados da pesquisa demonstraram a eficácia do modelo proposto, confirmando as previsões teóricas tanto para as órbitas planetárias e suas energias quanto, para o potencial efetivo. Desta forma, este trabalho contribui para a formação de professores, capacitando-os a desenvolver seus próprios simuladores educacionais. Adicionalmente, disponibiliza-se na web um simulador do sistema solar, acessível a alunos e professores do ensino básico, fomentando a exploração interativa dos conceitos de astronomia.

Neste estudo, a apresentação de conceitos, equações e códigos relacionados à simulação de órbitas planetárias foi realizada de forma didática e detalhada. Buscou-se, especificamente, tornar transparente o processo de desenvolvimento das ideias físicas que fundamentam as equações e a construção dos códigos de simulação, visando contribuir significativamente para o aprendizado do leitor.

Como perspectivas, as simulações desenvolvidas ainda podem ser aprimoradas para incorporar aspectos adicionais, como a incrementar a precessão do periélio da órbita de Mercúrio, as inclinações das órbitas planetárias em relação à eclíptica e a modelagem de cometas e asteroides. Observa-se que as órbitas planetárias possuem baixa excentricidade, resultando em trajetórias quase circulares. No caso da Terra, a pequena variação entre periélio e afélio não causa variações significativas de temperatura no planeta. O principal fator associado à temperatura terrestre é a inclinação da Terra em relação à eclíptica.

Para incentivar os alunos de licenciatura em Física aos conceitos de mecânica celeste e linguagem de programação em Python, pretende-se aplicar uma oficina de simulação de órbitas planetárias que foi proposta neste trabalho.

É importante ressaltar que devido ao Python ser uma linguagem de código aberto, torna-se uma ferramenta de baixo custo para as escolas, viabilizando a simulação sem custos. Além disso, o uso de simulações em Python alinha-se com competências e habilidades da BNCC, que destaca o uso de tecnologias digitais para a aprendizagem e o desenvolvimento do pensamento computacional.

REFERÊNCIAS

ALMEIDA, L. N. d. et al. Estudo de órbitas planetárias utilizando simulações numéricas com python. Universidade Federal de Mato Grosso, 2016.

ARAÚJO, I. M. d.; FIGUEIRA, M. V. **As leis de Kepler: da revolução científica a uma simulação para o ensino básico**. Dissertação (B.S. thesis) — Brasil, 2022.

AUSUBEL, D. P. A aprendizagem significativa. **São Paulo**, 1982.

AVILA, G. Kepler e a órbita elíptica. **Revista do professor de Matemática**, n. 15, p. 7, 1989.

BALDOW, R.; SILVA, A. P. T. B. Galileu, kepler e suas descobertas: análise de uma peça teatral vivenciada com estudantes do ensino fundamental e médio. **Experiências em Ensino de Ciências**, v. 9, n. 2, p. 45–68, 2014.

BERNARDES, T. d. O.; IACHEL, G.; SCALVI, R. M. F. Metodologias para o ensino de astronomia e física através da construção de telescópios. **Caderno Brasileiro de Ensino de Física**, Universidade Federal de Santa Catarina (UFSC), v. 25, n. 1, p. 103–117, 2008.

BEZERRA, D. et al. A evolução do ensino da física–perspectiva docente. **Scientia Plena**, v. 5, n. 9, 2009.

CARROLL, B. W.; OSTLIE, D. A. **An introduction to modern astrophysics**. [S.l.]: Cambridge University Press, 2017.

CONCEIÇÃO, C. P. D. da; ADMIRAL, T. D. Uso de simulação em python para ensinar força de arrasto a uma turma de ensino médio. **Brazilian Journal of Science**, v. 1, n. 10, p. 64–73, 2022.

CONTADOR, P. R. M. **Kepler o legislador dos céus**. [S.l.]: Livraria da Física, 2022.

CORCETTI, N. T.; VERASZTO, E. V. Um estudo da contribuição de ptolomeu para a evolução do modelo geocêntrico a partir de uma perspectiva histórica. **Anais do XXII Simpósio Nacional de Ensino de Física. São Carlo, SP: USP–SNEF**, 2017.

DEBALD, F. R. B. Tics e prática pedagógica universitária. **Revista Pleiade**, v. 1, n. 1, p. 83–94, 2007.

FARIA, E. T. O professor e as novas tecnologias. **Ser professor**, v. 4, p. 57–72, 2004.

FRANCO, A. F.; FILHO, J. d. O. C. As metodologias ativas como instrumento para se atingir uma aprendizagem significativa, reflexiva e interdisciplinar no ensino jurídico. In: **Colloquium Socialis, Presidente Prudente**. [S.l.: s.n.], 2017. v. 1, p. 510–516.

HEWITT, P. G. **Fundamentos de física conceitual**. [S.l.]: Bookman, 2000.

JUNIOR, A. G. B.; ALVES, F. M. Aprendendo o movimento circular uniforme utilizando o ambiente virtual vpython. **Revista do Professor de Física**, v. 3, n. Especial, p. 47–48, 2019.

KEPLER, S.; SARAIVA, M. d. F. O. Astronomia e astrofísica. **Porto Alegre: Editora da**, 2004.

LOBO, A. S. M.; MAIA, L. C. G. O uso das tics como ferramenta de ensino-aprendizagem no ensino superior. **Caderno de Geografia**, Pontifícia Universidade Católica de Minas Gerais, v. 25, n. 44, p. 16–26, 2015.

MEDEIROS, A.; MEDEIROS, C. F. d. Possibilidades e limitações das simulações computacionais no ensino da física. **Revista Brasileira de Ensino de Física**, SciELO Brasil, v. 24, p. 77–86, 2002.

MOREIRA, M. A. Linguagem e aprendizagem significativa. In: **Conferência de encerramento do IV Encontro Internacional sobre Aprendizagem Significativa, Maragogi, AL, Brasil**. [S.l.: s.n.], 2003. v. 8.

NERLING, M. A. M.; DARROZ, L. M. Tecnologias e aprendizagem significativa. **Cenas Educacionais**, v. 4, p. e10956–e10956, 2021.

NEVES, M. C. D. A terra e sua posiçc ao no universo: Formas, dimens oes e modelos orbitais. **Revista Brasileira de Ensino de Féísica**, v. 22, n. 4, p. 557, 2000.

NUSSENZVEIG, H. M. **Curso de física básica: Mecânica (vol. 1)**. [S.l.]: Editora Blucher, 2013. v. 394.

PELIZZARI, A. et al. Teoria da aprendizagem significativa segundo ausubel. **revista PEC**, v. 2, n. 1, p. 37–42, 2002.

PESENTE, G. M.; MATOS, E. A. S. Á. de; AVELINO, A. O ensino de matemática por meio da linguagem de programação python. **Revista Contemporânea**, v. 3, n. 3, p. 1239–1256, 2023.

PINTO, O. M.; HEREDIA, J. R.; GONÇALVES, R. A. Simulações em python: uma alternativa para o ensino de física. **REVISTA DELOS**, v. 17, n. 60, p. e2349–e2349, 2024.

STEINER, J. E. A origem do universo. **Estudos avançados**, SciELO Brasil, v. 20, p. 231–248, 2006.

TAVARES, R. Aprendizagem significativa. **Revista conceitos**, v. 10, n. 55, p. 55–60, 2004.

THORNTON, S. T.; MARION, J. B. **Dinâmica clássica de partículas e sistemas**. [S.l.]: Cengage Learning, 2011.

VASCONCELOS, C. F. Utilizando a linguagem python para resolução de problemas de física. 2022.

WINTERLE, P.; STEINBRUCH, A. **Geometria analítica**. [S.l.]: Makron Books, São Paulo, 2000.

APÊNDICE A – CÓDIGO DAS ÓRBITAS VIA GOOGLE COLAB

A seguir o passo a passo do código da simulação das órbitas planetárias em Python via Google Colab:

```
!apt-get install libx264-155
!pip install pyffmpeg
import matplotlib.pyplot as plt
import matplotlib.animation as animation
plt.rcParams['animation.ffmpeg_path'] = '/usr/bin/ffmpeg'
plt.rcParams['animation.codec'] = 'h264'
from IPython.display import HTML
import numpy as np


# Constantes

G = 6.67430e-11  # Constante gravitacional
M_sol = 1.98847e30  # Massa do Sol
AU = 1.495978707e11  # Unidade astronômica

# Parâmetro para Mercúrio

M_mercurio = 3.302e23 # massa de Mercúrio
A1 = 0.387*AU # semi-eixo maior
epsilon1 = 0.20563 # excentricidade
theta1 = -np.pi / 2 # ângulo inicial

#Parâmetros Vênus

M_venus = 4.8685e24 # massa de Vênus
A2 = 0.723*AU # semi-eixo maior
epsilon2 = 0.00677 # excentricidade
theta2 = -np.pi / 2 # ângulo inicial

# Parâmetros da Terra

M_terra = 5.9722e24  # massa da Terra
A3 = 1.0 *AU  # semi-eixo maior
epsilon3 = 0.0167 # excentricidade
theta3 = -np.pi / 2  # ângulo inicial

# Parâmetros de Marte

M_marte = 6.4171e23  # massa de marte
```

```
A4 = 1.524 *AU # semi-eixo maior
epsilon4 = 0.0934  # excentricidade
theta4 = 0  # ângulo inicial

# Cálculo das posições iniciais

r1 = A1 * (1 - epsilon1**2) / (1 + epsilon1 * np.cos(theta1))  # Mercúrio
r2 = A2 * (1 - epsilon2**2) / (1 + epsilon2 * np.cos(theta2))  # Vênus
r3 = A3 * (1 - epsilon3**2) / (1 + epsilon3 * np.cos(theta3))  #  Terra
r4 = A4 * (1 - epsilon4**2) / (1 + epsilon4 * np.cos(theta4))  # Marte
x1 = r1 * np.cos(theta1)
y1 = r1 * np.sin(theta1)
x2 = r2 * np.cos(theta2)
y2 = r2 * np.sin(theta2)
x3 = r3 * np.cos(theta3)
y3 = r3 * np.sin(theta3)
x4 = r4 * np.cos(theta4)
y4 = r4 * np.sin(theta4)

# Cálculo das velocidades iniciais

v1 = np.sqrt(G * (M_sol+M_mercurio) * (2 / r1 - 1 / A1))  # Mercúrio
v2 = np.sqrt(G * (M_sol+M_venus) * (2 / r2 - 1 / A2))  # Vênus
v3 = np.sqrt(G * (M_sol+M_terra) * (2 / r3 - 1 / A3))  # Terra
v4 = np.sqrt(G * (M_sol+M_marte) * (2 / r4 - 1 / A4))  # Marte

vx1 = v1 * np.cos(theta1 + np.pi / 2)
vy1 = v1 * np.sin(theta1 + np.pi / 2)
vx2 = v2 * np.cos(theta2 + np.pi / 2)
vy2 = v2 * np.sin(theta2 + np.pi / 2)
vx3 = v3 * np.cos(theta3 + np.pi / 2)
vy3 = v3 * np.sin(theta3 + np.pi / 2)
vx4 = v4 * np.cos(theta4 + np.pi / 2)
vy4 = v4 * np.sin(theta4 + np.pi / 2)

# Simulação

t_end = 10 * 365.25 * 24 * 3600  # Tempo total da simulação em segundos

# Passos de tempo individuais

dt_mercurio = 0.0005 * 365.25 * 24 * 3600  # Menor dt para Mercúrio
dt_venus = 0.0008 * 365.25 * 24 * 3600
dt_terra = 0.001 * 365.25 * 24 * 3600
dt_marte = 0.0012 * 365.25 * 24 * 3600

t = np.arange(0, t_end, dt_mercurio)
```

```
x1_pos = []
y1_pos = []
t = np.arange(0, t_end, dt_venus)
x2_pos = []
y2_pos = []
t = np.arange(0, t_end, dt_terra)
x3_pos = []
y3_pos = []
t = np.arange(0, t_end, dt_marte)
x4_pos = []
y4_pos = []

for _ in t:
    # Calcula a aceleração gravitacional
    r1_vec = np.array([x1, y1])
    r2_vec = np.array([x2, y2])
    a1 = -G * M_sol * r1_vec / np.linalg.norm(r1_vec)**3
    a2 = -G * M_sol * r2_vec / np.linalg.norm(r2_vec)**3
    r3_vec = np.array([x3, y3])
    r4_vec = np.array([x4, y4])
    a3 = -G * M_sol * r3_vec /
    np.linalg.norm(r3_vec)**3
    a4 = -G * M_sol * r4_vec / np.linalg.norm(r4_vec)**3

    # Atualiza a velocidade e a posição

    vx1 += a1[0] * dt_mercurio
    vy1 += a1[1] * dt_mercurio
    vx2 += a2[0] * dt_venus
    vy2 += a2[1] * dt_venus
    x1 += vx1 * dt_mercurio
    y1 += vy1 * dt_mercurio
    x2 += vx2 * dt_venus
    y2 += vy2 * dt_venus
    vx3 += a3[0] * dt_terra
    vy3 += a3[1] * dt_terra
    x3 += vx3 * dt_terra
    y3 += vy3 * dt_terra
    vx4 += a4[0] * dt_marte
    vy4 += a4[1] * dt_marte
    x4 += vx4 * dt_marte
    y4 += vy4 * dt_marte

    x1_pos.append(x1)
    y1_pos.append(y1)
    x2_pos.append(x2)
    y2_pos.append(y2)
```

```
    x3_pos.append(x3)
    y3_pos.append(y3)
    x4_pos.append(x4)
    y4_pos.append(y4)

#Energias

E1 = 0.5*M_mercurio*(vx1**2+vy1**2)-G*M_sol*M_mercurio/r1
E2 = 0.5*M_venus*(vx2**2+vy2**2)-G*M_sol*M_venus/r2
E3 = 0.5*M_terra*(vx3**2+vy3**2)-G*M_sol*M_terra/r3
E4 = 0.5*M_marte*(vx4**2+vy4**2)-G*M_sol*M_marte/r4

print("Energia da órbita de mercúrio:",E1)

print("Energia da órbita de Vênus:",E2)

print("Energia da órbita da Terra:",E3)

print("Energia da órbita de Marte:",E4)


# Plotagem e Animação

%matplotlib inline
plt.style.use('dark_background')   # fundo preto
fig, ax = plt.subplots()  # configuração da figura
ax.set_aspect('equal')  # ajuste na proporção nos eixos x e y
ax.set_xlim([-3.0 * AU, 3.0 * AU]) # limites no eixo x
ax.set_ylim([-3.0 * AU, 3.0 * AU]) # limites no eixo y
ax.set_xlabel('Distância radial (m)')  # definição do eixo x
ax.set_ylabel('Distância radial (m)') # definição do eixo y
plt.title('Úrbitas Planetárias')  # título
plt.savefig('orbitas.png', dpi = 500) # salvar figura em qualidade de 500 dpi
line1, = ax.plot([], [], 'g-', label='Mercúrio') # exibir rastro de Mercúrio
line2, = ax.plot([], [], 'y-', label='Vênus')   # exibir rastro de Vênus
line3, = ax.plot([], [], 'b-', label='Terra')   # exibir rastro da Terra
line4, = ax.plot([], [], 'r-', label='Marte')   # exibir rastro de Marte
sol, = ax.plot([0], [0], 'yo', markersize=10, label='Sol') # adicionando o sol
ax.legend()  # exibir legenda
def animate(i):
    line1.set_data(x1_pos[:i], y1_pos[:i])
    # Mercúrio
    line2.set_data(x2_pos[:i], y2_pos[:i])  # Vênus
    line3.set_data(x3_pos[:i], y3_pos[:i])
    # Terra
    line4.set_data(x4_pos[:i], y4_pos[:i])  # Marte
    return line1, line2, line3, line4, sol # retornando o sol também
```

```
ani = animation.FuncAnimation(fig, animate, frames=len(t), interval=10, blit=True)

# Gerando um vídeo HTML5
html = ani.to_html5_video()

# Exibindo a animação no Colab
HTML(html)

# Salvando a animação como um arquivo de vídeo (MP4)
ani.save('simulacao_orbitas.mp4', writer='ffmpeg', fps=30)

# (Opcional) Exibir um link para download do vídeo no Colab
from google.colab import files
files.download('simulacao_orbitas.mp4')
```

APÊNDICE B – CÓDIGO DO POTENCIAL EFETIVO

A seguir o passo a passo do código da simulação do potencial efetivo da Terra em Python via Google Colab:

```
import numpy as np
import matplotlib.pyplot as plt

# Constantes
G = 6.67430e-11  # Constante gravitacional
M_sol = 1.98847e30  # Massa do Sol
M_terra = 5.9722e24  # Massa da Terra
AU = 1.495978707e11  # Unidade astronômica

# Parâmetros da órbita da Terra
A1 = 1.0 * AU  # Semi-eixo maior
epsilon1 = 0.0167  # Excentricidade

# Cálculo do momento angular da Terra
L = M_terra * np.sqrt(G * M_sol * A1 * (1 - epsilon1**2))

# Função para calcular o potencial efetivo
def effective_potential(r):
    U_gravitacional = -G * M_sol * M_terra / r
    U_centrifugo = L**2 / (2 * M_terra * r**2)
    return U_gravitacional + U_centrifugo

# Cria um array de valores de r
r_values = np.linspace(0.4* AU, 10 * AU, 1000)

# Calcula os valores do potencial efetivo para cada valor de r
Ueff_values = [effective_potential(r) for r in r_values]

U_centrifugo = [L**2 / (2 * M_terra * r**2) for r in r_values]
U1_values = [-G * M_sol * M_terra / r for r in r_values]



# Plota o gráfico do potencial efetivo
plt.figure(figsize=(10, 6))
plt.plot(r_values, Ueff_values, label='Potencial Efetivo')
plt.plot(r_values, U_centrifugo, label='Energia centrífuga')
plt.plot(r_values, U1_values, label='Energia Potencial gravitacional')
plt.xlabel('r (m)')
plt.ylabel('V (J)')
plt.title('Potencial Efetivo da Terra')
plt.savefig('potencial_terra.png', dpi = 500)
```

```
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(10, 6))
plt.plot(r_values, Ueff_values, label='Potencial Efetivo')
#plt.plot(r_values, U_centrifugo, label='Energia Potencial centrífuga')
#plt.plot(r_values, U1_values, label='Energia Potencial')
plt.xlabel('r (m)')
plt.ylabel('V (J)')
plt.title('Potencial Efetivo da Terra')
plt.savefig('potencial_terra.png', dpi = 500)
plt.legend()
plt.grid(True)
plt.show()

plt.figure(figsize=(10, 6))
#plt.plot(r_values, Ueff_values, label='Potencial Efetivo')
plt.plot(r_values, U_centrifugo, label='Energia centrífuga', c="green")
plt.plot(r_values, U1_values, label='Energia Potencial gravitacional')
plt.xlabel('r (m)')
plt.ylabel('V (J)')
plt.title('Potencial Efetivo da Terra')
plt.savefig('potencial_terra.png', dpi = 500)
plt.legend()
plt.grid(True)
plt.show()
```

APÊNDICE C – CÓDIGO DAS ÓRBITAS VIA VPYTHON

A seguir o passo a passo do código da Simulação das órbitas planetárias em Vpython via Glowscript:

```
# Constantes físicas utilizadas na simulação

k = 0.01720209395  # Define a constante gravitacional Gaussiana
GM = k**2     # Calcula o produto da constante gravitacional (G) pela massa do Sol (M_sol)
dt = 0.001    # Define o passo de tempo da simulação

# Configuração da cena do VPython para visualização da simulação

scene = canvas(background=color.black)  # Cria uma janela de visualização com fundo preto
scene = canvas(title='Simulação do Sistema Solar', width=800, height=600, center=vector(0,0,0))

# Parâmetros orbitais e físicos de Mercúrio

M_mercurio = 3.302e23  # Define a massa de Mercúrio em kg
A1 = 0.387            # Define o semi-eixo maior da órbita de Mercúrio em UA
epsilon1 = 0.20563      # Define a excentricidade da órbita de Mercúrio
theta1 = -pi / 2       # Define o ângulo inicial de Mercúrio em sua órbita (em radianos)

# Parâmetros orbitais e físicos de Vênus

M_venus = 4.8685e24   # Define a massa de Vênus em kg
A2 = 0.723            # Define o semi-eixo maior da órbita de Vênus em UA
epsilon2 = 0.00677      # Define a excentricidade da órbita de Vênus
theta2 = -pi / 2       # Define o ângulo inicial de Vênus em sua órbita (em radianos)

# Parâmetros orbitais e físicos da Terra

M_terra = 5.9722e24   # Define a massa da Terra em kg
A3 = 1.0              # Define o semi-eixo maior da órbita da Terra em UA
epsilon3 = 0.0167       # Define a excentricidade da órbita da Terra
theta3 = -pi / 2       # Define o ângulo inicial da Terra em sua órbita (em radianos)

# Parâmetros orbitais e físicos de Marte

M_marte = 6.4171e23   # Define a massa de Marte em kg
A4 = 1.524            # Define o semi-eixo maior da órbita de Marte em UA
epsilon4 = 0.0934       # Define a excentricidade da órbita de Marte
theta4 = -pi / 2       # Define o ângulo inicial de Marte em sua órbita (em radianos)

# Parâmetros orbitais e físicos de Júpiter

M_J = 1.8982e27      # Define a massa de Júpiter em kg
```

```
A5 = 5.2028           # Define o semi-eixo maior da órbita de Júpiter em UA
epsilon5 = 0.0484       # Define a excentricidade da órbita de Júpiter
theta5 = -pi/2         # Define o ângulo inicial de Júpiter em sua órbita (em radianos)

# Parâmetros orbitais e físicos de Saturno

M_S = 5.6834e26      # Define a massa de Saturno em kg
A6 = 9.5826           # Define o semi-eixo maior da órbita de Saturno em UA
epsilon6 = 0.0565       # Define a excentricidade da órbita de Saturno
theta6 = pi/2          # Define o ângulo inicial de Saturno em sua órbita (em radianos)

# Parâmetros orbitais e físicos de Netuno

M_N = 1.02413e26    # Define a massa de Netuno em kg
A8 = 30.047          # Define o semi-eixo maior da órbita de Netuno em UA
epsilon8 = 0.0086     # Define a excentricidade da órbita de Netuno
theta7 = pi           # Define o ângulo inicial de Netuno em sua órbita (em radianos)

# Parâmetros orbitais e físicos de Urano

M_U = 8.6813e25      # Define a massa de Urano em kg
A7 = 19.189          # Define o semi-eixo maior da órbita de Urano em UA
epsilon7 = 0.0472     # Define a excentricidade da órbita de Urano
theta8 = -pi/4        # Define o ângulo inicial de Urano em sua órbita (em radianos)

# Informações de cores dos planetas para exibição no console

print('Mercúrio - Branco')
print('Vênus - verde')
print('Terra - azul')
print('Marte - vermelho')
print('Júpiter - laranja')
print('Saturno - yellow')
print('Urano - ciano')
print('Netuno - Azul')

# Cálculo das posições iniciais dos planetas usando coordenadas polares para cartesianas

r1 = A1 * (1 - epsilon1**2) / (1 + epsilon1 * cos(theta1))
r2 = A2 * (1 - epsilon2**2) / (1 + epsilon2 * cos(theta2))
r3 = A3 * (1 - epsilon3**2) / (1 + epsilon3 * cos(theta3))
r4 = A4 * (1 - epsilon4**2) / (1 + epsilon4 * cos(theta4))
r5 = A5 * (1 - epsilon5**2) / (1 + epsilon5 *cos(theta5))
r6 = A6 * (1 - epsilon6**2) / (1 + epsilon6 * cos(theta6))
r7 = A7 * (1 - epsilon7**2) / (1 + epsilon7 * cos(theta7))
r8 = A8 * (1 - epsilon8**2) / (1 + epsilon8 * cos(theta8))
```

```
x1 = r1 * cos(theta1)  # Calcula a coordenada x inicial de Mercúrio
y1 = r1 * sin(theta1)  # Calcula a coordenada y inicial de Mercúrio
x2 = r2 * cos(theta2)  # Calcula a coordenada x inicial de Vênus
y2 = r2 * sin(theta2)  # Calcula a coordenada y inicial de Vênus
x3 = r3 * cos(theta3)  # Calcula a coordenada x inicial da Terra
y3 = r3 * sin(theta3)  # Calcula a coordenada y inicial da Terra
x4 = r4 * cos(theta4)  # Calcula a coordenada x inicial de Marte
y4 = r4 * sin(theta4)  # Calcula a coordenada y inicial de Marte
x5 = r5 * cos(theta5)  # Calcula a coordenada x inicial de Júpiter
y5 = r5 * sin(theta5)  # Calcula a coordenada y inicial de Júpiter
x6 = r6 * cos(theta6)  # Calcula a coordenada x inicial de Saturno
y6 = r6 * sin(theta6)  # Calcula a coordenada y inicial de Saturno
x7 = r7 * cos(theta7)  # Calcula a coordenada x inicial de Urano
y7 = r7 * sin(theta7)  # Calcula a coordenada y inicial de Urano
x8 = r8 * cos(theta8)  # Calcula a coordenada x inicial de Netuno
y8 = r8 * sin(theta8)  # Calcula a coordenada y inicial de Netuno

# Cálculo das velocidades iniciais dos planetas usando a conservação de energia

v1 = sqrt(GM * (2 / r1 - 1 / A1))  # Calcula a velocidade inicial de Mercúrio
v2 = sqrt(GM * (2 / r2 - 1 / A2))  # Calcula a velocidade inicial de Vênus
v3 = sqrt(GM * (2 / r3 - 1 / A3))  # Calcula a velocidade inicial da Terra
v4 = sqrt(GM * (2 / r4 - 1 / A4))  # Calcula a velocidade inicial de Marte
v5 = sqrt(GM * (2 / r5 - 1 / A5))  # Calcula a velocidade inicial de Júpiter
v6 = sqrt(GM * (2 / r6 - 1 / A6))  # Calcula a velocidade inicial de Saturno
v7 = sqrt(GM * (2 / r7 - 1 / A7))  # Calcula a velocidade inicial de Urano
v8 = sqrt(GM * (2 / r8 - 1 / A8))  # Calcula a velocidade inicial de Netuno

vx1 = v1 * cos(theta1 + pi / 2)  # Calcula a componente x da velocidade inicial de Mercúrio
vy1 = v1 * sin(theta1 + pi / 2)  # Calcula a componente y da velocidade inicial de Mercúrio
vx2 = v2 * cos(theta2 + pi / 2)  # Calcula a componente x da velocidade inicial de Vênus
vy2 = v2 * sin(theta2 + pi / 2)  # Calcula a componente y da velocidade inicial de Vênus
vx3 = v3 * cos(theta3 + pi / 2)  # Calcula a componente x da velocidade inicial da Terra
vy3 = v3 * sin(theta3 + pi / 2)  # Calcula a componente y da velocidade inicial da Terra
vx4 = v4 * cos(theta4 + pi / 2)  # Calcula a componente x da velocidade inicial de Marte
vy4 = v4 * sin(theta4 + pi / 2)  # Calcula a componente y da velocidade inicial de Marte
vx5 = v5 * cos(theta5 + pi / 2)  # Calcula a componente x da velocidade inicial de Júpiter
vy5 = v5 * sin(theta5 + pi / 2)  # Calcula a componente y da velocidade inicial de Júpiter
vx6 = v6 * cos(theta6 + pi / 2)  # Calcula a componente x da velocidade inicial de Saturno
vy6 = v6 * sin(theta6 + pi / 2)  # Calcula a componente y da velocidade inicial de Saturno
vx7 = v7 * cos(theta7 + pi / 2)  # Calcula a componente x da velocidade inicial de Urano
vy7 = v7 * sin(theta7 + pi / 2)  # Calcula a componente y da velocidade inicial de Urano
vx8 = v8 * cos(theta8 + pi / 2)  # Calcula a componente x da velocidade inicial de Netuno
vy8 = v8 * sin(theta8 + pi / 2)  # Calcula a componente y da velocidade inicial de Netuno

# Cálculo das energias totais dos planetas (cinética + potencial)
```

```
E1 = 0.5*M_mercurio*(vx1**2+vy1**2)-GM*M_mercurio/r1  # Calcula a energia total de Mercúrio
E2 = 0.5*M_venus*(vx2**2+vy2**2)-GM*M_venus/r2      # Calcula a energia total de Vênus
E3 = 0.5*M_terra*(vx3**2+vy3**2)-GM*M_terra/r3      # Calcula a energia total da Terra
E4 = 0.5*M_marte*(vx4**2+vy4**2)-GM*M_marte/r4      # Calcula a energia total de Marte
E5 = 0.5*M_J*(vx5**2+vy5**2)-GM*M_J/r5              # Calcula a energia total de Júpiter
E7 = 0.5 * (vx7**2 + vy7**2)-GM*M_N/r7              # Calcula a energia total de Netuno
E8 = 0.5 * (vx8**2 + vy8**2)-GM*M_U/r8              # Calcula a energia total de Urano

# Exibe as energias totais dos planetas no console

print('Energia de Mercúrio:',E1)
print('Energia de Vênus:',E2)
print('Energia de Terra:',E3)
print('Energia de Marte:',E4)
print('Energia de Júpiter:',E5)

# Cria o objeto visual para o Sol

sol = sphere(pos=vector(0, 0, 0), radius=0.15, color=color.yellow)  # Cria uma esfera amarela

# Cria os objetos visuais para os planetas e define suas velocidades iniciais

Me = sphere(pos=vector(r1*cos(theta1), r1*sin(theta1), 0), radius=0.02,
color=color.white, make_trail=True)  # Cria a esfera para Mercúrio
Me.velocidade = vector(v1*cos(theta1 + pi/2), v1*sin(theta1 + pi/2), 0)

V = sphere(pos=vector(r2*cos(theta2), r2*sin(theta2), 0), radius=0.03,
color=color.green, make_trail=True)  # Cria a esfera para Vênus
V.velocidade = vector(v2*cos(theta2 + pi/2), v2*sin(theta2 + pi/2), 0)

Earth = sphere(pos=vector(r3*cos(theta3), r3*sin(theta3), 0), radius=0.05,
color=color.blue, make_trail=True)  # Cria a esfera para a Terra
Earth.velocidade = vector(v3*cos(theta3 + pi/2), v3*sin(theta3 + pi/2), 0)

Mars = sphere(pos=vector(r4*cos(theta4), r4*sin(theta4), 0), radius=0.03,
color=color.red, make_trail=True)  # Cria a esfera para Marte
Mars.velocidade = vector(v4*cos(theta4 + pi/2), v4*sin(theta4 + pi/2), 0)

J = sphere(pos=vector(r5*cos(theta5), r5*sin(theta5), 0), radius=0.03,
color=color.orange, make_trail=True)  # Cria a esfera para Júpiter
J.velocidade = vector(v5*cos(theta5 + pi/2), v5*sin(theta5 + pi/2), 0)

S = sphere(pos=vector(r6*cos(theta6), r6*sin(theta6), 0), radius=0.03,
color=color.yellow, make_trail=True)  # Cria a esfera para Saturno
S.velocidade = vector(v6*cos(theta6 + pi/2), v6*sin(theta6 + pi/2), 0)

U = sphere(pos=vector(r7*cos(theta7), r7*sin(theta7), 0), radius=0.03,
```

```
color=color.cyan, make_trail=True)  # Cria a esfera para Urano
U.velocidade = vector(v7*cos(theta7 + pi/2), v7*sin(theta7 + pi/2), 0)

N = sphere(pos=vector(r8*cos(theta8), r8*sin(theta8), 0), radius=0.03,
color=color.blue, make_trail=True)  # Cria a esfera para Netuno
N.velocidade = vector(v8*cos(theta8 + pi/2), v8*sin(theta8 + pi/2), 0)

# Define as cores dos planetas

Earth.color = color.blue
Mars.color = color.red
Me.color = color.white
V.color = color.green
J.color = color.orange
S.color = color.yellow
U.color = color.cyan
N.color = color.blue

# Inicializa o tempo da simulação

t = 0

# Loop principal da simulação que calcula as posições e velocidades

while True:  # Inicia um loop infinito que executa a simulação continuamente
    rate(1000000000)  # Controla a taxa de quadros da animação, ajustando a velocidade da simulação

    # Cálculo da aceleração gravitacional de Mercúrio e atualização de sua velocidade e posição

    r_vec1 = Me.pos - sol.pos  # Calcula o vetor da posição relativa entre Mercúrio e o Sol
    a1 = -GM * r_vec1 / mag(r_vec1)**3  # Calcula a aceleração gravitacional de Mercúrio
    Me.velocidade = Me.velocidade + a1 * dt  # Atualiza a velocidade de Mercúrio
    Me.pos = Me.pos + Me.velocidade * dt  # Atualiza a posição de Mercúrio

    # Cálculo da aceleração gravitacional de Vênus e atualização de sua velocidade e posição

    r_vec2 = V.pos - sol.pos  # Calcula o vetor da posição relativa entre Vênus e o Sol
    a2 = -GM * r_vec2 / mag(r_vec2)**3  # Calcula a aceleração gravitacional de Vênus
    V.velocidade = V.velocidade + a2 * dt  # Atualiza a velocidade de Vênus
    V.pos = V.pos + V.velocidade * dt  # Atualiza a posição de Vênus

    # Cálculo da aceleração gravitacional da Terra e atualização de sua velocidade e posição

    r_vec3 = Earth.pos - sol.pos  # Calcula o vetor da posição relativa entre a Terra e o Sol
    a3 = -GM * r_vec3 / mag(r_vec3)**3  # Calcula a aceleração gravitacional da Terra
    Earth.velocidade = Earth.velocidade + a3 * dt  # Atualiza a velocidade da Terra
    Earth.pos = Earth.pos + Earth.velocidade * dt  # Atualiza a posição da Terra
```

```
# Cálculo da aceleração gravitacional de Marte e atualização de sua velocidade e posição

r_vec4 = Mars.pos - sol.pos  # Calcula o vetor da posição relativa entre Marte e o Sol
a4 = -GM * r_vec4 / mag(r_vec4)**3  # Calcula a aceleração gravitacional de Marte
Mars.velocidade = Mars.velocidade + a4 * dt  # Atualiza a velocidade de Marte
Mars.pos = Mars.pos + Mars.velocidade * dt  # Atualiza a posição de Marte

# Cálculo da aceleração gravitacional de Júpiter e atualização de sua velocidade e posição

r_vec5 = J.pos - sol.pos  # Calcula o vetor da posição relativa entre Júpiter e o Sol
a5 = -GM * r_vec5 / mag(r_vec5)**3  # Calcula a aceleração gravitacional de Júpiter
J.velocidade = J.velocidade + a5 * dt  # Atualiza a velocidade de Júpiter
J.pos = J.pos + J.velocidade * dt  # Atualiza a posição de Júpiter

# Cálculo da aceleração gravitacional de Saturno e atualização de sua velocidade e posição

r_vec6 = S.pos - sol.pos  # Calcula o vetor da posição relativa entre Saturno e o Sol
a6 = -GM * r_vec6 / mag(r_vec6)**3  # Calcula a aceleração gravitacional de Saturno
S.velocidade = S.velocidade + a6 * dt  # Atualiza a velocidade de Saturno
S.pos = S.pos + S.velocidade * dt  # Atualiza a posição de Saturno

# Cálculo da aceleração gravitacional de Urano e atualização de sua velocidade e posição

r_vec7 = U.pos - sol.pos  # Calcula o vetor da posição relativa entre Urano e o Sol
a7 = -GM * r_vec7 / mag(r_vec7)**3  # Calcula a aceleração gravitacional de Urano
U.velocidade = U.velocidade + a7 * dt  # Atualiza a velocidade de Urano
U.pos = U.pos + U.velocidade * dt  # Atualiza a posição de Urano

# Cálculo da aceleração gravitacional de Netuno e atualização de sua velocidade e posição

r_vec8 = N.pos - sol.pos  # Calcula o vetor da posição relativa entre Netuno e o Sol
a8 = -GM * r_vec8 / mag(r_vec8)**3  # Calcula a aceleração gravitacional de Netuno
N.velocidade = N.velocidade + a8 * dt  # Atualiza a velocidade de Netuno
N.pos = N.pos + N.velocidade * dt  # Atualiza a posição de Netuno

t += dt  # Incrementa o tempo da simulação pelo passo de tempo
```

APÊNDICE D – POTENCIAL EFETIVO E PONTO MÍNIMO

A seguir o código para situar o ponto mínimo no potenial efetivo:

```
import numpy as np
import matplotlib.pyplot as plt

# Constantes
G = 6.67430e-11  # Constante gravitacional
M_sol = 1.98847e30  # Massa do Sol
M_terra = 5.9722e24  # Massa da Terra
AU = 1.495978707e11  # Unidade astronômica

# Parâmetros da órbita da Terra
A1 = 1.0 * AU  # Semi-eixo maior
epsilon1 = 0.0167  # Excentricidade

# Cálculo do momento angular da Terra
L = M_terra * np.sqrt(G * M_sol * A1 * (1 - epsilon1**2))

# Função para calcular o potencial efetivo
def effective_potential(r):
    U_gravitacional = -G * M_sol * M_terra / r
    U_centrifugo = L**2 / (2 * M_terra * r**2)
    return U_gravitacional + U_centrifugo

# Cria um array de valores de r
r_values = np.linspace(0.4* AU, 10 * AU, 1000)

# Calcula os valores do potencial efetivo para cada valor de r
Ueff_values = [effective_potential(r) for r in r_values]

# Plota o gráfico do potencial efetivo
plt.figure(figsize=(10, 6))
plt.plot(r_values, Ueff_values, label='Potencial Efetivo')

# Encontra o índice do mínimo e do máximo do potencial efetivo
min_index = np.argmin(Ueff_values)
max_index = np.argmax(Ueff_values) # descomente esta linha se quiser marcar o máximo

# Encontra o valor de r e Ueff correspondentes ao mínimo e ao máximo
r_min = r_values[min_index]
Ueff_min = Ueff_values[min_index]
```

```
# Marca o ponto de mínimo no gráfico
plt.plot(r_min, Ueff_min, 'ro', label='Mínimo', markersize=5)

# Adiciona rótulos para o ponto de mínimo
plt.text(r_min + 0.2*AU, Ueff_min + 1e30, f'({r_min/AU:.2f} AU, {Ueff_min:.2e} J)',
         ha='center', va='bottom')


plt.xlabel('r (m)')
plt.ylabel('V (J)')
plt.title('Potencial Efetivo da Terra')
plt.legend()
plt.grid(True)
plt.show()
```

APÊNDICE E – POTENCIAL EFETIVO DE MERCÚRIO

A seguir o código para simular o potencial efetivo da Mercúrio destacando o afélio e periélio:

```
import numpy as np
import matplotlib.pyplot as plt

# Constantes
G = 6.67430e-11  # Constante gravitacional
M_sol = 1.98847e30  # Massa do Sol
M_mercurio = 3.302e23  # Massa de Mercúrio
AU = 1.495978707e11  # Unidade astronômica

# Parâmetros da órbita de Mercúrio
A1 = 0.387 * AU  # Semi-eixo maior
epsilon1 = 0.20563  # Excentricidade

# Cálculo do momento angular de Mercúrio
L = M_mercurio * np.sqrt(G * M_sol * A1 * (1 - epsilon1**2))

# Função para calcular o potencial efetivo
def effective_potential(r):
    U_gravitacional = -G * M_sol * M_mercurio / r
    U_centrifugo = L**2 / (2 * M_mercurio * r**2)
    return U_gravitacional + U_centrifugo

# Cria um array de valores de r
r_values = np.linspace(0.1 * AU, 4 * AU, 1000)  # Ajuste o intervalo se necessário

# Calcula os valores do potencial efetivo para cada valor de r
Ueff_values = [effective_potential(r) for r in r_values]

# Calcula os raios do afélio e periélio
r_afelio = A1 * (1 + epsilon1)
r_perielio = A1 * (1 - epsilon1)

# Calcula as energias correspondentes ao afélio e periélio
E_afelio = effective_potential(r_afelio)
E_perielio = effective_potential(r_perielio)

# Plota o gráfico do potencial efetivo
plt.figure(figsize=(10, 6))
plt.plot(r_values, Ueff_values, label='Potencial Efetivo')

# Destaca os pontos de afélio e periélio com rótulos de coordenadas ajustados e setas
plt.plot(r_afelio, E_afelio, 'ro', label='Afélio')
```

```
plt.annotate(f'({r_afelio/AU:.2f} AU, {E_afelio:.2e} J)',
             xy=(r_afelio, E_afelio), xytext=(r_afelio + 0.1*AU, E_afelio + 2e30),
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=0.2'), fontsize=10)

plt.plot(r_perielio, E_perielio, 'bo', label='Periélio')
plt.annotate(f'({r_perielio/AU:.2f} AU, {E_perielio:.2e} J)',
             xy=(r_perielio, E_perielio), xytext=(r_perielio - 0.1*AU, E_perielio - 2e30),
             arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=-0.2'), fontsize=10)

# Traça a reta que representa a energia
plt.axhline(y=E_afelio, color='gray', linestyle='--', label='Energia')

# Ajusta os limites do gráfico
plt.xlim(0, 1.0 * AU)  # Ajuste o intervalo conforme necessário
plt.ylim(E_perielio - 5e30, E_afelio + 5e30)  # Ajuste o intervalo conforme necessário

# Adiciona rótulos, título e legenda
plt.xlabel('Distância radial (m)', fontsize=12)
plt.ylabel('V(J)', fontsize=12)
plt.title('Potencial Efetivo de Mercúrio', fontsize=14)
plt.legend(fontsize=12)
plt.grid(True)

# Exibe o gráfico
plt.show()
```