Edson Alonso Fallaluza

Exemplos de anomalias gravimétricas produzidas por fontes de massa simples: Implementação em linguagem Python

Projeto de pesquisa para a disciplina Estágio Curricular 1

Universidade Federal Fluminense - UFF

Departamento de Geologia e Geofísica

Graduação em Geofísica

Orientador: Rodrigo Bijani

Brasil

28 de maio de 2019

1 Introdução

A Geofísica é a ciência que estuda os fenômenos físicos relacionados aos planetas, principalmente à Terra. Os métodos geofísicos são aplicações diretas da teoria geofísica, e são utilizados por várias outras áreas, tendo um papel essencial na exploração de recursos minerais, àgua, óleo e gás, entre outros (**??**). Para cada objeto de pesquisa existe um método geofísico cuja resposta será mais relevante ao problema inicial, porém o conhecimento de todos os métodos é de suma importância para os geofísicos e os que se utilizam dela, sendo recomendado a integração de vários deles, sempre que possível (**??**).

Uma das áreas da geofísica de destacada importancia é a modelagem númerica de dados potenciais, baseada na teoria matemática da **Análise Vetorial** e suas aplicações à campos físicos existêntes e mensuráveis (**??**). Derivada dessa área, surge o método gravimétrico. Como o próprio nome sugere, este método utiliza medidas do campo gravitacional da Terra para localizar variações horizontais e verticais na densidade das rochas em subsuperfície (**??**). Veja como exemplo o caso dos domos de sal em subsuperfície no mar de *Barents* na Noruega.

Figura 1 – Mapa do Mar de Barents retirado do site <worldatlas.com>

O sal geralmente tem uma densidade muito menor do que as rochas à sua volta, causando, para os entendedores do assunto, uma anomalia gravimétrica negativa. Lembre-se que domos de sal são ótimas trapas para óleo e gás.
Na linguagem da modelagem, os dados medidos em campo chamam-se **Dados Observados**, os quais, claro, dependem de alguns parâmetros desconhecidos em geral (**??**). Nosso objetivo é um pouco ambicioso: Dado um conjunto de Dados Observados, conseguir obter

um conjunto de parâmetros que produzam os dados mais semelhantes aos observados. Estes novos dados chamam-se **Dados Preditos**. Ora, se os Dados Preditos estão próximos o suficiente dos Dados Observados, é razoável acreditar que os nossos parâmetros representam com fidelidade os parâmetros da natureza. É claro que, matemáticamente, isto é verdade. Porém, por motivos matemáticos não importantes para este documento, vale a pena comentar que isso nem sempre é verdade (**??**). Assim, é importante lembrar, novamente, a importância da integração de vários dados geofísicos para aumentar a confiancia nos nossos parâmetros.

Ao tentar procurar por implementações em linguagens *open source* de modelos gravimétricos bidimensionais, muitas dificuldades foram encontradas, tais como:

- Fórmulas equivocadas e de difícil análise;
- Pouco detalhamento sobre os parâmetros utilizados para a modelagem;
- Falta de documentos e artigos que discutam o tema, dentre outros.

Neste trabalho apresentamos a abordagem teórica necessária para o entendimento das expressões que calculam os dados preditos por diferentes formas geométricas simples. Adicionalmente, deixaremos implementações, em liguagem Python, de diversos modelos gravimétricos 2D de formas geométricas simplificadas, como por exemplo:

- Fontes esféricas de massa;
- Fontes retangulares, simulando dikes e soleiras;
- Folhas e hastes de massa que simulam falhas e contatos geológicos.

As rotinas computacionais são disponbilizadas por meio de um *link* da plataforma digital GITHUB () a fim de fornecer todos os elementos necessários para a modelagem.

2 Aspectos teóricos: Atração Gravitacional

Durante séculos, astrônomos passaram suas vidas catalogando a posição de estrelas e planetas, afim de entender o nosso universo. Em meados do século $XVII$, o matemático e físico alemão **Johannes Kepler** trabalhava na geometria do movimento desses astros. Através de observações, Kepler foi capaz de calcular que o planeta hoje conhecido como Marte orbitava ao redor do sol de forma eliptica. Surgem assim, as 3 leis do movimento planetário de Kepler (**??**).

A lei de Kepler explica como os planetas se movem ao redor do sol, porém não explica o porquê. Em 5 de Julho de 1687, no livro *The Principia*, Sir. **Isaac Newton** publica pela primeira vez, a conhecida hoje em dia como Lei da Gravitação Universal. O enunciado desta lei afirma que cada ponto de massa atrai todos os outros pontos de massa por uma força que atua ao longo da linha que intercepta os dois pontos(**??**). A força é proporcional ao produto das suas massas, e inversamente proporcional ao quadrado da distância entre elas (ver Figura 2):

$$\overrightarrow{\mathbf{F}} = \mathbf{G}\frac{m_1 m_2}{r^2}\,\hat{r}, \tag{2.1}$$

em que:

- $\overrightarrow{\mathbf{F}}$ é a força gravitacional entre as massas, de caráter atrativo;
- $\mathbf{G}$ é a constante gravitacional, cujo valor no Sistema Internacional é $6.674 \times 10^{-11}\mathbf{N}(m/kg)^2$;
- m_1 é a primeira massa, em kg;
- m_2 é a segunda massa, em kg;
- r é a distância entre os centros de massa dos objetos;
- $\hat{r}$ é o vetor unitário na direção dos centros de massa das fontes e aponta do centro de massa m_1 para o centro de massa m_2.

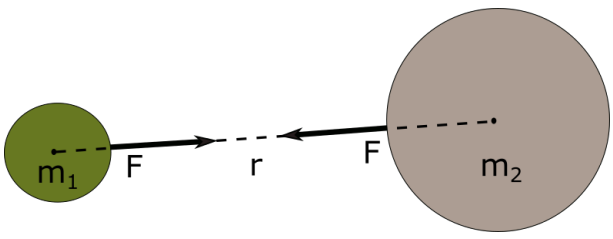

Figura 2 – Figura ilustrativa referente à Lei da Atração gravitacional.

Ora, da segunda Lei de movimento proposta por Newton, sabemos que a resultante de forças atuantes sobre um corpo é proporcional à aceleração assumida pelo corpo:

$$\overrightarrow{\mathbf{F}} = m\overrightarrow{\mathbf{a}}, \tag{2.2}$$

em que $\overrightarrow{\mathbf{F}}$ é a força resultante num objeto e m é a massa de prova, que representa a constante de proporcionalidade na equação 2.2. Para garantir a generalidade do apresentado até aqui, m_2 é tratado como m. Uma vez que as duas formas de calcular $\overrightarrow{\mathbf{F}}$ são corretas, ambas tem de ser iguais, portanto:

$$m\overrightarrow{\mathbf{a}} = \mathbf{G}\frac{m_1 m}{r^2}\ \hat{r}, \tag{2.3}$$

Executando os algebrismos adequados e tratando a aceleração $\overrightarrow{\mathbf{a}}$ como sendo a aceleração gravitacional $\overrightarrow{\mathbf{g}}(\mathbf{r})$, chega-se à seguinte expressão:

$$\overrightarrow{\mathbf{g}}(\mathbf{r}) = -\mathbf{G}\frac{m_1}{r^2}\ \hat{r}, \tag{2.4}$$

em que $\hat{r}$ é o vetor unitário que representa a direção de atuação da aceleração gravitacional $\overrightarrow{\mathbf{g}}(\mathbf{r})$ (**??**). O campo gravitacional é um campo vetorial que descreve a região de influência da força gravitacional que é aplicada a um objeto dotado de massa, em um ponto qualquer do espaço. O sinal de menos indica que a força gravitacional é atrativa.

Neste trabalho, iremos focar apenas na componente vertical, também chamada componente **z**, do campo $\overrightarrow{\mathbf{g}}$, por razões físicas as quais não são o foco deste trabalho. Para isso, tomaremos emprestado um conceito muito importante da Álgebra Linear.
Dado um conjunto ortonormal de vetores $\{\mathbf{v_i}\}$, os quais geram um espaço vetorial, a projeção de um vetor $\mathbf{v}$ qualquer deste espaço sobre um dos vetores da base, digamos $\mathbf{v_k}$ é, simplesmente, o produto interno de $\mathbf{v}$ com o vetor da base $\mathbf{v_k}$ (**??**).
Assim, a componente vertical da atração gravitacional($\mathbf{g_z}$) é simplesmente a projeção ortogonal de $\overrightarrow{\mathbf{g}}$ no terceiro vetor da base, isto é:

$$\mathbf{g_z} = \langle\overrightarrow{\mathbf{g}(\mathbf{r})}, \hat{z}\rangle \tag{2.5}$$

$$\mathbf{g_z} = \langle -\mathbf{G}\frac{m_1}{r^2}\ \hat{r}, \hat{z}\rangle \tag{2.6}$$

Como $\hat{r} = \frac{\overrightarrow{r}}{\|r\|}$, $\overrightarrow{r} = x\hat{x} + y\hat{y} + z\hat{z}$ e $\|r\| = \sqrt{\langle r, r\rangle} = r$ temos que:

$$\mathbf{g_z} = \frac{1}{r}\langle -\mathbf{G}\frac{m_1}{r^2}x\hat{x} + -\mathbf{G}\frac{m_1}{r^2}y\hat{y} + -\mathbf{G}\frac{m_1}{r^2}z\hat{z}, \hat{z}\rangle \tag{2.7}$$

Para a nossa sorte, o conjunto $\{\hat{x}, \hat{y}, \hat{z}\}$ é um conjunto ortonormal, assim:

$$\mathbf{g_z} = -\mathbf{G}\frac{m_1 z}{r^2 r} \tag{2.8}$$

O leitor pode verificar que:

$$\mathbf{g_x} = -\mathbf{G}\frac{m_1 x}{r^2 r} \tag{2.9}$$

$$\mathbf{g_y} = -\mathbf{G}\frac{m_1 y}{r^2 r} \tag{2.10}$$

Como $m = \rho v$, onde ρ é a densidade do objeto e v é o seu volume, temos:

$$\mathbf{g_z} = -\mathbf{G}\frac{\rho v z}{r^3} \tag{2.11}$$

Ora, para um corpo qualquer, podemos tomar um elemento infinitesimal de volume dv. Para este elemento, a densidade correspondente é $\rho(v)$. Assim, os autores mais conhecidos na área como **??**) definem a **componente vertical da atração gravitacional** como:

$$\mathbf{g_z} = \int_V -\mathbf{G}\frac{\rho(v) z}{r^3} dv \tag{2.12}$$

2.1 Modelos gravimétricos 2D

Nesta seção do documento, modelos gravimétricos bidimensionais são apresentados a partir de um conjunto de observações, cujas coordenadas Cartesianas são representadas por $(x_1, z_1), (x_2, z_2), \ldots, (x_N, z_N)$. A título de simplificação, todos os exemplos a seguir possuem a mesma distribuição de observações, conforme apresentado na Figura esquemática abaixo. De fato, como esta é uma implementação em Python, deixaremos especificado que os gravímetros estão todos no mesmo eixo $\mathbf{z} = 0$ e as coordenadas $\mathbf{x}$ foram criadas usando a função, da biblioteca **Numpy**, **numpy.linspace**$(-\mathbf{10}, \mathbf{10}, \mathbf{300})$.

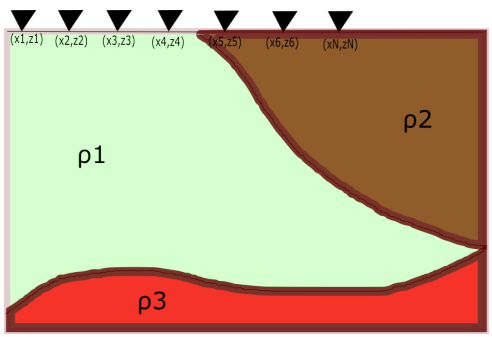

Figura 3 – Gravímetros.

2.1.1 Esfera de massa

Decorre da equação 2.12 que, a componente vertical da atração gravitacional para o caso de uma esfera de massa m é:

$$g_z = -\mathbf{G}\frac{m(z_p - z_c)}{[(x_p - x_c)^2 + (z_p - z_c)^2]^{3/2}} \tag{2.13}$$

Onde:

. m é a massa da esfera;

. (x_c, z_c) são as coordenadas do centro de massa da esfera;

. (x_p, z_p) são as coordenadas do centro de massa do referencial

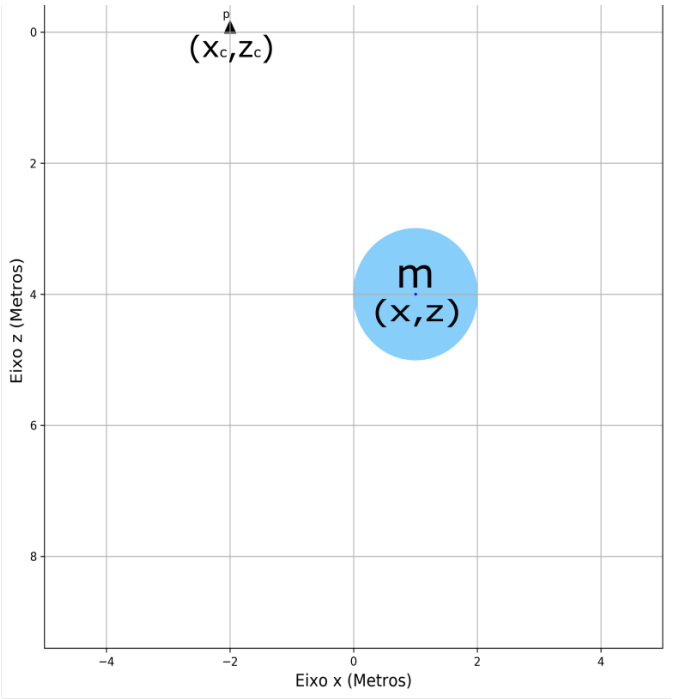

Figura 4 – Esfera em subsuperfície

Vale a pena observar que, neste caso, a esfera é de tamanho infinitesimal, isto é, o raio não é um parâmetro a ser tomado em conta, apenas a posição do seu centro de massa. Na implementação em Python, os parâmetros da esfera serão passados em formato de lista, da seguinte forma: **esfera** = [x_c, z_c, massa da esfera]. Para o gráfico abaixo, foram utilizados os seguintes parâmetros:

- Para a curva azul:

  $esfera_1$ = [-7.5, 4.0, 2E6]

$esfera_2 = [2.5, 4.0, 1E6]$

- Para a curva vermelha:

  $esfera_1 = [-7.5, 4.0, 1E6]$

  $esfera_2 = [2.5, 4.0, 1E6]$

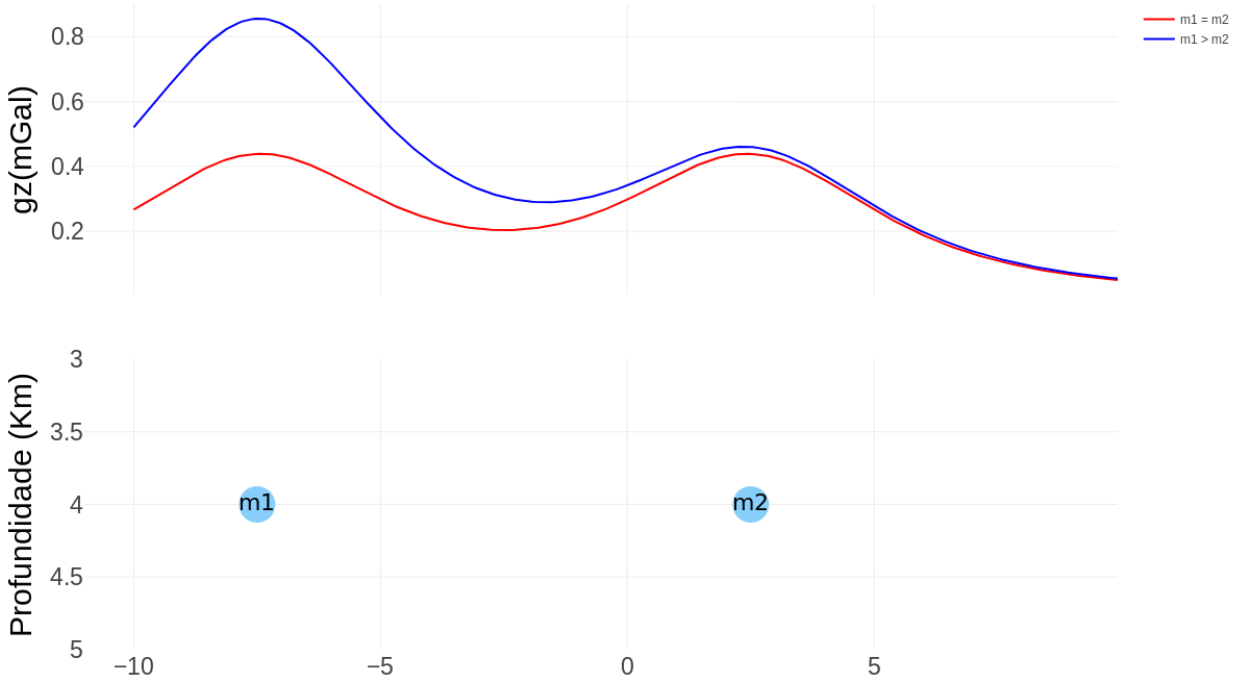

Figura 5 – Esferas de massa m_1 e m_2

2.1.2 Disco de Massa

Decorre da equação 2.12 que, a componente vertical da atração gravitacional para o caso de um disco de massa m é:

$$g_z = -2\mathbf{G}\frac{m(z_p - z_c)}{(x_p - x_c)^2 + (z_p - z_c)^2} \tag{2.14}$$

Onde:

. m é a massa do disco;

. (x_c, z_c) são as coordenadas do centro de massa do disco;

. (x_p, z_p) são as coordenadas do centro de massa do referencial

Na implementação em Python, os parâmetros do disco serão passados em formato de lista, da seguinte forma: **disco** = [x_c, z_c, massa do disco]. Para o gráfico abaixo, foram utilizados os seguintes parâmetros:

- $disco$ = [0.0, 5.0, 2.5E5]

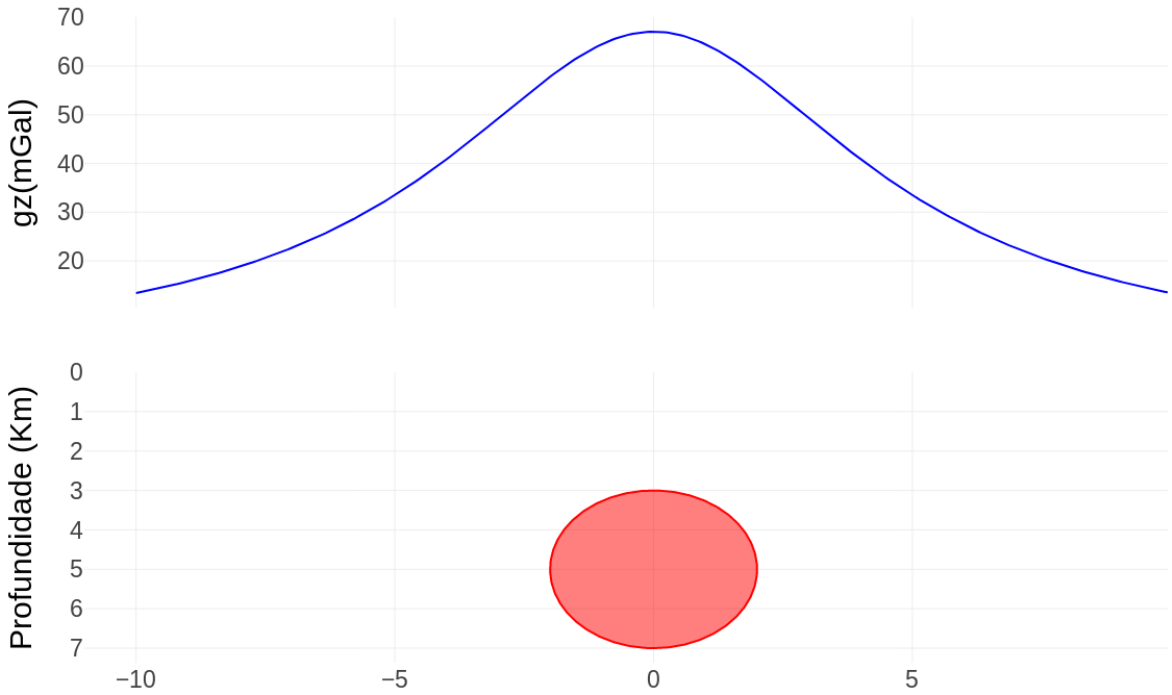

Figura 6 – Disco de massa m

2.1.3 Haste fina sintética

Neste caso, a implementação foi feita utilizando o princípio da superposição. É claro que uma haste unidimensional é apenas a soma infinita (não enumerável) de fontes pontuais. Claro, devido à nossa limitação em conseguir computar infinitas fontes pontuais, criamos uma linha de massas suficientemente densa a fim de replicar uma pequena haste com um ângulo de mergulho. Na implementação em Python, foi utilizada uma função chamada **linemasses**, a qual está bem explicada no código que é disponibilizado. O leitor é convidado a pensar porque os parâmetros passados são os mesmos, apenas tem dimensões maiores: **haste** = [x_c, z_c, massa da haste]. Para calcular os vetores x_c e z_c, a função **linemasses** recebe um ponto inicial e um ponto final. Para o gráfico abaixo, a **massa** é sempre de $1E5Kg$ e foram utilizados os seguintes parâmetros:

- Para a curva azul:

Ponto inicial = $(-1.85857864376269031.6414213562373094)$

Ponto final = $(-6.101219330881975, 5.884062043356595)$

- Para a curva vermelha:

  Ponto inicial = $(-3.9798989873223327, 1.6414213562373094)$

  Ponto final = $(-3.9798989873223327, 5.884062043356595)$

- Para a curva vermelha:

  Ponto inicial = $(-7.0, 3.762741699796952)$

  Ponto final = $(-1.0, 3.762741699796952)$

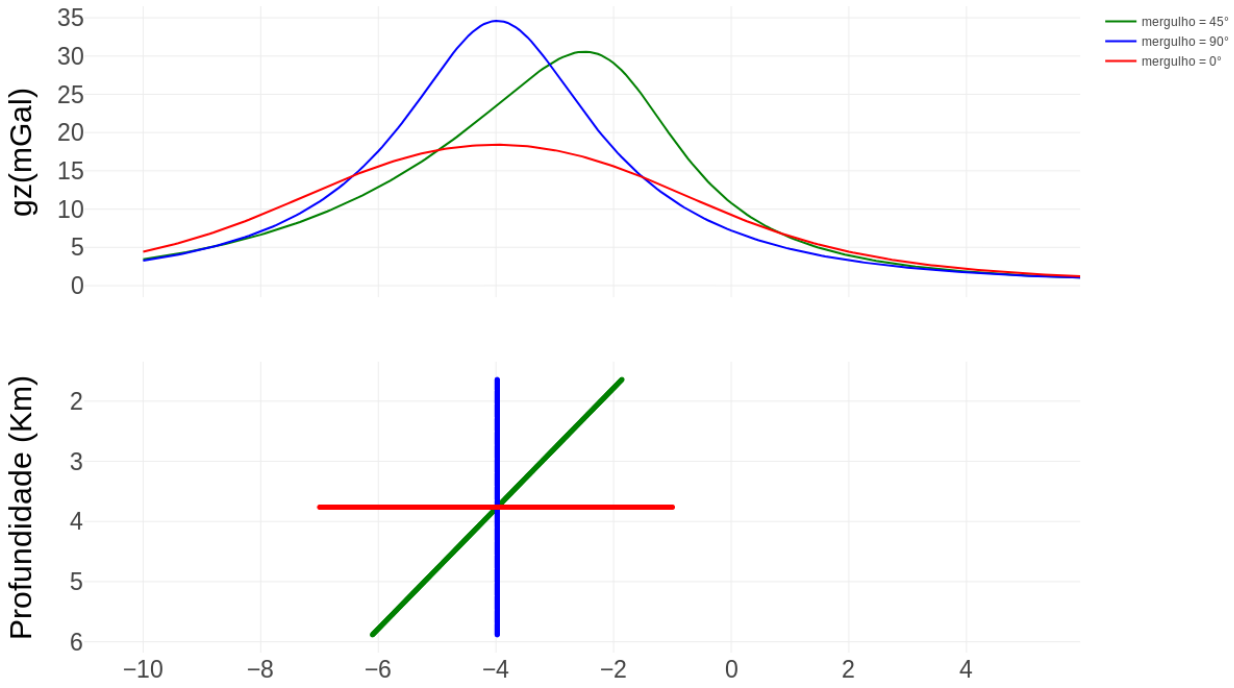

Figura 7 – Haste Sintética

2.1.4 Folha Finita

Decorre da equação 2.12 que, a componente vertical da atração gravitacional para o caso de uma folha finita com mergulho α(90+ ângulo de mergulho) é:

$$g_z = 2\mathbf{G}\rho T[sin(\alpha)ln(\frac{r2}{r1}) - (\theta_2 + \theta_1)cos(\alpha)] \tag{2.15}$$

Onde:

. ρ é a densidade da folha;

. T é a espessura da folha;

. $r1 = \sqrt{(x^2 + h^2)}$

. $r2 = \sqrt{((x + lcos(\alpha))^2 + (h + lsin(\alpha))^2)}$

O leitor pode verificar que θ_1 e θ_2 podem ser calculados utilizando a lei dos cossenos.

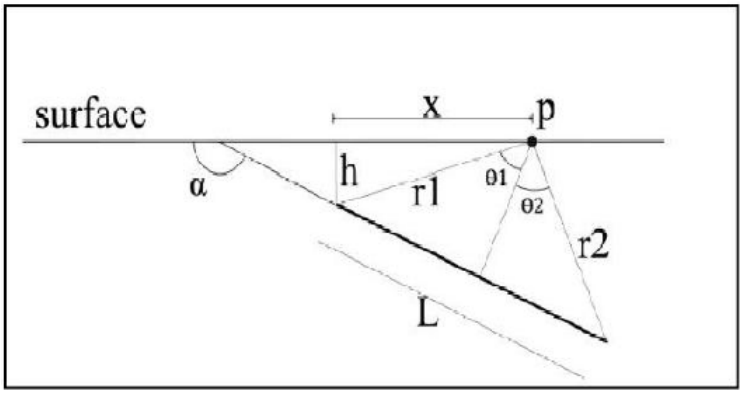

Figura 8 – Folha Finita

Na implementação em Python, os parâmetros da folha serão passados em formato de lista, da seguinte forma: **folha** = [x_c, z_c, ρ, mergulho, comprimento, espessura]. Ainda em Python, vale a pena dizer que este modelo tem umas limitações:

- O ângulo de mergulho α tem que estar no intervalo $[0, 90]$ e em graus;
- A profundidade do topo não deve ser maior do que $7Km$;
- A espessura da Folha não deve ser maior do que 1(Lembre-se que a ideia é que a folha seja fina).

Para o gráfico abaixo, foram utilizados os seguintes parâmetros:

- $folha$ = [2.5, 2.0, 2.6E5, 30.0, 6.0, 0.2]

2.1.5 Folha Semi-Finita

Decorre da equação 2.12 que, a componente vertical da atração gravitacional para o caso de uma folha semi-finita é:

$$g_z = 2\mathbf{G}\rho T[\frac{\pi}{2} - tan^{-1}(\frac{x}{z})] \tag{2.16}$$

Onde:

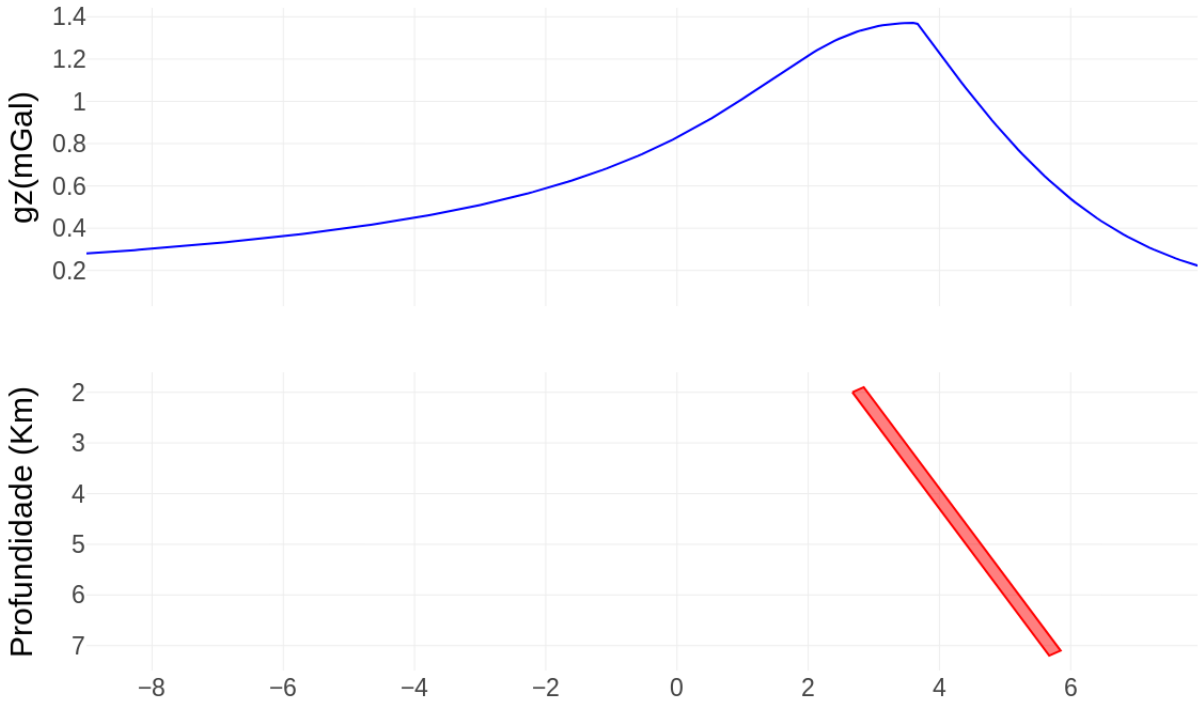

Figura 9 – Folha Finita

. ρ é a densidade da folha;

. T é a espessura da folha;

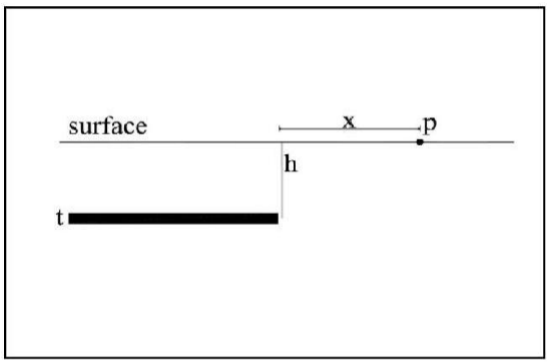

Figura 10 – Folha Semi-Finita

Na implementação em Python, os parâmetros da folha serão passados em formato de lista, da seguinte forma: **folha** = [x_c, z_c, ρ, espessura]. Para o gráfico abaixo, foram utilizados os seguintes parâmetros:

- $folha$ = [-3.0, 2.0, 1E6, 0.4]

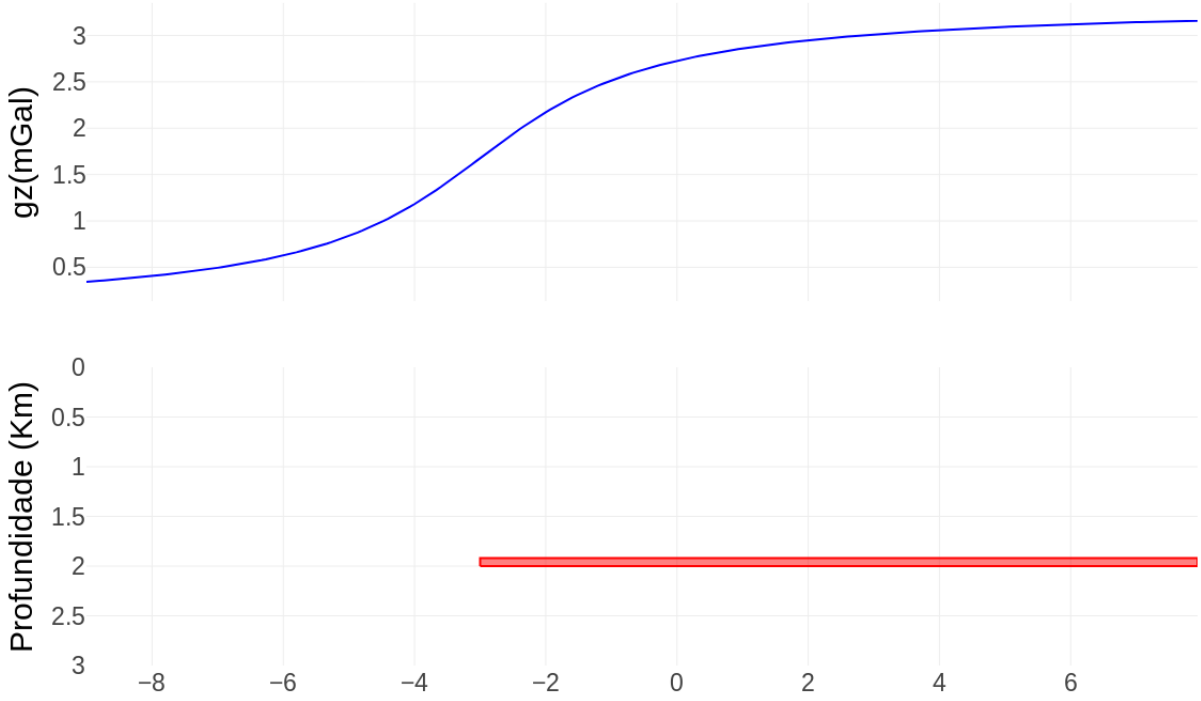

Figura 11 – Folha Semi-Finita

2.1.6 Prisma

Decorre da equação 2.12 que, a componente vertical da atração gravitacional para o caso de um prisma é:

$$g_z = \mathbf{G}\rho A \left[\frac{1}{\sqrt{x^2 + y^2 + {h_1}^2}} - \frac{1}{\sqrt{x^2 + y^2 + {h_2}^2}} \right] \tag{2.17}$$

Onde:

. ρ é a densidade da folha;

. A é cross section, que neste caso é igual a 1, pois nosso problema é bidimensional;

. $h1$ é a distância até o topo o prisma;

. $h2$ é a distância até a base do prisma;

Veja que, neste caso, estamos no plano $y = 0$, assim nossa equação é mais simples:

$$g_z = \mathbf{G}\rho \left[\frac{1}{\sqrt{x^2 + {h_1}^2}} - \frac{1}{\sqrt{x^2 + {h_2}^2}} \right] \tag{2.18}$$

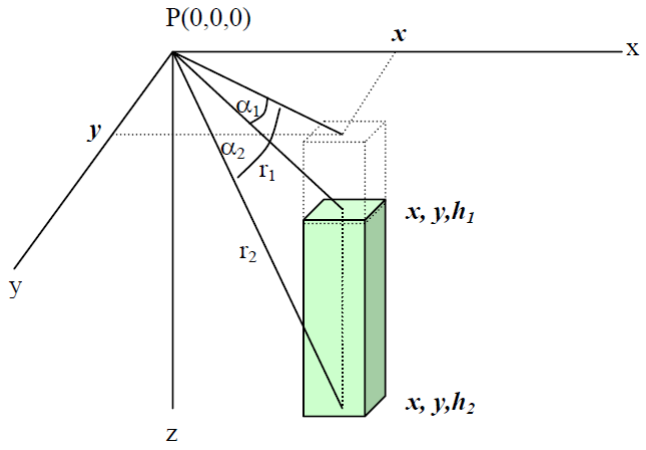

Figura 12 – Prisma

Na implementação em Python, os parâmetros da folha serão passados em formato de lista, da seguinte forma: **prisma** = [x_c, z_{topo}, comprimento vertical, comprimento horizontal, ρ]. Para o gráfico abaixo, foram utilizados os seguintes parâmetros:

- *prisma* = [2.5, 3.0, 5.0, 3E5, 4.0]

2.2 Perspectivas

A ideia é implementar mais codigos de modelagem em linguagem python para outros casos, e tambem considerar a modelagem magnética de fontes simples.

2.3 Conclusões

Fica claro a importância da implementação computacional para a modelagem gravimétrica, uma vez que muitos aspectos físico-interpretativos só são possiveis de serem vistos graças às suas curvas de $\mathbf{g_z}$. O Python é claramente uma ótima ferramenta para isto, uma vez que ele pode performar tanto os cálculos, de forma efetiva, como a visualização, de forma organizada.
A disponibilização deste documento serve como alavanca para o estudo de alunos recém começando seus estudos na gravimetria e, principalmente, na computação científica.

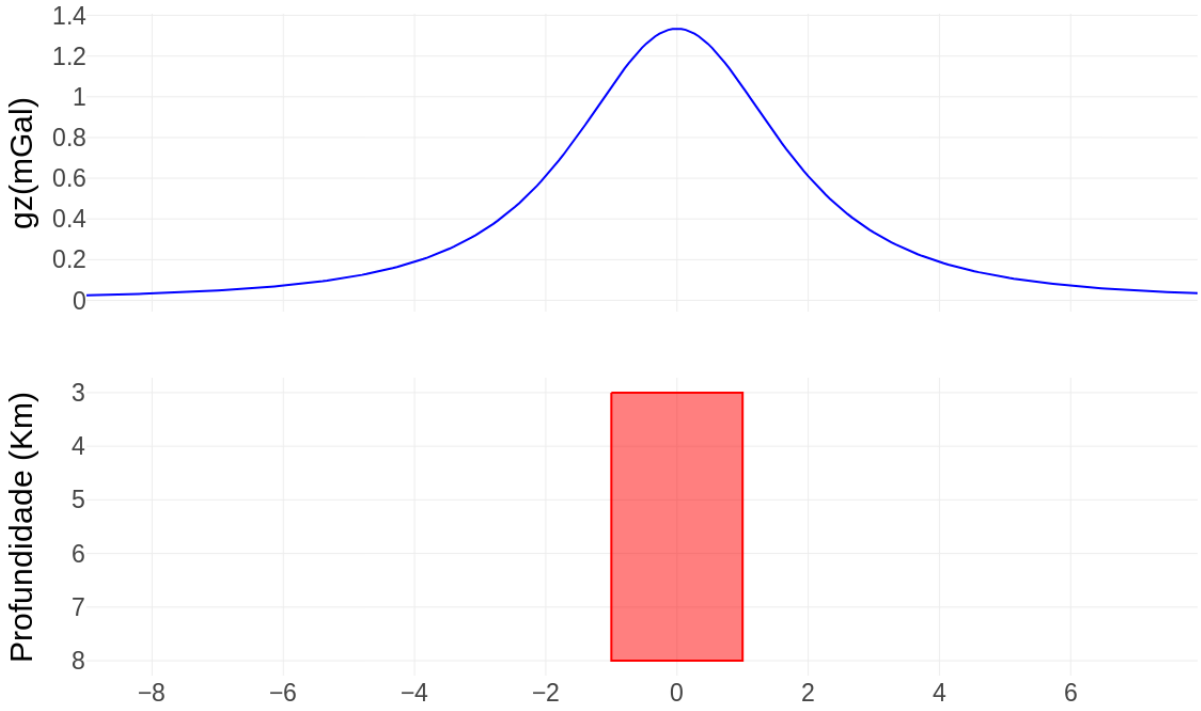

Figura 13 – Prisma

Referências