



UNIVERSIDAD NACIONAL DE SAN JUAN
FACULTAD DE CIENCIAS EXACTAS, FÍSICAS Y NATURALES
INSTITUTO GEOFÍSICO SISMOLÓGICO VOLPONI

TESIS DOCTORAL

Metodologías y herramientas computacionales para el procesamiento y modelado de datos gravimétricos

Autor

Lic. Santiago Rubén Soler

Director

Dr. Mario E. Giménez

Codirector

Dr. Leonardo Uieda

2022

Metodologías y herramientas computacionales para el procesamiento y modelado de datos gravimétricos
por Santiago Rubén Soler
doi: [10.6084/m9.figshare.16906909](https://doi.org/10.6084/m9.figshare.16906909)

Versión 1.1.1
Última modificación el 2022-02-08.



Disponible bajo **Licencia Creative Commons Atribución 4.0 Internacional**.
<https://creativecommons.org/licenses/by/4.0/deed.es>

Usted es libre de:

Compartir: Copiar y redistribuir el material en cualquier medio o formato.

Adaptar: Remezclar, transformar y construir a partir del material para cualquier propósito, incluso comercialmente.

Bajo los siguientes términos:

Atribución: Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo del licenciatario.

No hay restricciones adicionales: No puede aplicar términos legales ni medidas tecnológicas que restrinjan legalmente a otras a hacer cualquier uso permitido por la licencia.

Índice general

1. Introducción	1
2. Fundamentos Geofísicos	5
2.1. Potencial gravitatorio	5
2.1.1. Potencial gravitatorio por una distribución de masa	6
2.1.2. Gradiente del potencial gravitatorio	7
2.2. Modelado directo	8
2.2.1. Masas puntuales	9
2.2.2. Prismas rectangulares	12
2.2.3. Teseroides o prismas esféricos	14
2.3. Gravedad terrestre	18
2.3.1. Gravedad normal	19
2.3.2. Disturbio de gravedad	23
2.4. Fuentes equivalentes	24
3. Teseroides de densidad variable	27
3.1. Resumen	27
3.2. Introducción	28
3.3. Metodología	30
3.3.1. Integración por Cuadratura de Gauss-Legendre	30
3.3.2. Discretización adaptativa bidimensional	31
3.3.3. Algoritmo de discretización basado en densidad	33
3.3.4. Resumen del algoritmo	35
3.3.5. Implementación por Software	35
3.4. Determinación de los <i>ratios</i> distancia-tamaño y <i>delta</i>	36
3.4.1. Densidad Lineal	38
3.4.2. Densidad exponencial	38
3.4.3. Densidad sinusoidal	42
3.5. Desempeño del algoritmo	44
3.6. Aplicación a la Cuenca Neuquina	45
3.7. Discusión	49
3.8. Conclusiones	52
3.A. Soluciones analíticas para un cascarón esférico	53
3.A.1. Densidad lineal	54

3.A.2. Densidad exponencial	55
3.A.3. Densidad sinusoidal	55
4. Fuentes equivalentes potenciadas por gradiente	57
4.1. Introducción	57
4.2. Metodología	60
4.2.1. La técnica de fuentes equivalentes	60
4.2.2. Solución por mínimos cuadrados amortiguados	61
4.2.3. Potenciación del gradiente	62
4.2.4. Ubicación de las fuentes	67
4.3. Pruebas sobre datos sintéticos	71
4.3.1. Estrategias de distribución de fuentes	75
4.3.2. Tamaño de las ventanas y cantidad de solapamiento en potenciación del gradiente	76
4.3.3. Interpolación con potenciación del gradiente	78
4.4. Grillado de datos gravimétricos de Australia	80
4.5. Discusión	83
4.5.1. Ubicación de las fuentes	83
4.5.2. Potenciación del gradiente	85
4.5.3. Datos gravimétricos sobre Australia	85
4.6. Conclusiones	86
4.7. Disponibilidad de datos y código	87
4.A. Parámetros para localizar fuentes equivalentes en pruebas con datos sintéticos	88
5. Fatiando a Terra	91
5.1. Introducción	91
5.2. Historia	93
5.3. Paquetes de software	95
5.3.1. Verde	96
5.3.2. Boule	97
5.3.3. Harmonica	98
5.3.4. Pooch	104
5.4. Desarrollo y mejores prácticas	108
5.4.1. Estilo de escritura	109
5.4.2. Documentación	110
5.4.3. Pruebas de software	110
5.4.4. Automatización e integración continua	112
5.5. Comunidad	114
5.6. Adopción en la comunidad científica	115
5.7. Contribuciones del autor al proyecto	117
5.8. Relación con investigaciones científicas	118
5.9. Planes a futuro	118

6. Conclusiones	121
Lista de Abreviaturas	125

Agradecimientos

Esta Tesis Doctoral no hubiese sido posible sin el apoyo económico del Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) y el acompañamiento del Instituto Geofísico Sismológico Volponi de la Universidad Nacional de San Juan, que me brindó la oportunidad, el conocimiento y la libertad para llevar a cabo mis investigaciones.

Agradezco a los tres miembros del Jurado por el tiempo que dedicaron a la lectura de la presente Tesis, cuyas observaciones y sugerencias han mejorado considerablemente su texto y presentación.

Además, deseo agradecer a aquellas personas que de diversas formas fueron partes fundamentales en el trabajo realizado en estos años.

Gracias Agustina por la oportunidad de adentrarme en el mundo de la Geociencia, por tu compañía incondicional, por la disposición a ayudarme con los obstáculos que se presentaban, por tu comprensión y aguante durante los días más difíciles, por ser cómplice de aventuras y principalmente por tu afecto y cariño.

Gracias Leonardo por aceptar trabajar conmigo durante estos años, por permitirme colaborar activamente en tus proyectos, por brindarme excelentes consejos laborales y personales, por estar siempre abierto a nuevas ideas y por el reconocimiento que siempre le diste a mi trabajo.

Gracias Mario por la oportunidad de realizar esta Tesis Doctoral, por alentarme a seguir las líneas de investigación que más me interesaban, por tu apoyo frente a cada idea o proyecto que me surgía, por tus conocimientos en la materia y por el apoyo personal brindado en todos estos años.

Quiero agradecer a mi familia por haberme acompañado en todas las decisiones que he tomado, por siempre estar presentes cuando los necesité y por seguir alentándome en el futuro que me espera. No quiero olvidarme de agradecer a Sol y a todos los Urcullu por adoptarme como un integrante más de su familia.

Por último, deseo mencionar a tres comunidades que me atravesaron durante todo el Doctorado. En primer lugar, me encuentro eternamente agradecido con toda la comunidad de desarrolladores de software de código abierto, sin la cual este trabajo no hubiera sido posible. En segundo lugar, gracias Software Underground por permitirme participar de su comunidad y conocer personas

grandiosas, con quienes espero poder encontrarme personalmente en el futuro. Y por último, pero no menos importante, gracias Geolatinas por dejarme formar parte del grupo hermoso de personas que construyeron; su compañía fue fundamental durante estos dos últimos años.

Capítulo 1

Introducción

La observación y medición del campo gravitatorio terrestre se remonta a los orígenes de la Ciencia moderna. Galileo Galilei realiza las primeras observaciones cuantitativas sobre la caída de los objetos en el Siglo XIV. En 1687 Isaac Newton propone la existencia de la fuerza gravitatoria como interacción entre toda la materia, describiendo una Ley Universal que la gobierna en su famoso tratado *Philosophiae Naturalis Principia Mathematica*.

A lo largo de los siglos subsiguientes muchos científicos y muchas científicas han realizado observaciones e interpretaciones acerca de las variaciones del campo gravitatorio terrestre en distintas localidades. Jean Richer detecta en 1672 que los péndulos oscilaban a frecuencias ligeramente distintas en París y en Guinea Francesa, dando nacimiento a la *gravimetría*. Newton atribuye esta variación a la forma ovalada de la Tierra. Es Pierre Bouguer quien encabeza en 1735 una expedición para realizar cuidadosas mediciones de estas variaciones a diversas latitudes.

Los experimentos llevados a cabo por Henry Cavendish alrededor de 1797 permitieron no solo estimar la densidad media de la Tierra, sino también un valor para la constante de gravitación universal G , medición que se encuentra muy de acuerdo al valor aceptado hoy en día.

Las primeras aplicaciones de mediciones gravimétricas a la geología vienen de la mano de John Pratt y George Airy en la década de 1850, proponiendo dos hipótesis opuestas acerca de cómo las partes rígidas de la corteza y el manto *flotan* sobre un substrato fluido, dando origen al concepto de *isostasia*.

Para finales del Siglo XIX y durante el Siglo XX, los avances tecnológicos permitieron perfeccionar los instrumentos a través de los cuales se pueden realizar observaciones del campo gravitatorio terrestre. Vale destacar los desarrollos de Vening Meinesz, Roland von Eötvös, LaCoste y Romberg ([Blakely, 1995](#)). Estos abrieron las puertas a la aplicación de las observaciones gravimétricas a múltiples problemas geológicos y geofísicos. Integrando estas mediciones con diversas metodologías de las Geociencias, permiten un amplio conocimiento sobre las estructuras y cuerpos debajo de la superficie terrestre.

El arribo del Siglo XXI nos recibe con misiones espaciales diseñadas para

situar satélites especialmente dedicados a la medición del campo gravitatorio terrestre de manera global y a lo largo del tiempo. Este tipo de observaciones pueden integrarse con las obtenidas en la superficie terrestre, sobre la superficie marítima o desde el aire para dar mayor completitud a los resultados e interpretaciones.

Las aplicaciones de los conocimientos que la gravimetría produce se extienden desde la exploración de recursos naturales, mayor comprensión sobre el origen de los fenómenos de nuestro planeta hasta la observación y mitigación de los cambios ambientales, climatológicos y biológicos.

En esta Tesis presentamos los avances realizados por el autor a lo largo de su Doctorado, vinculados al modelado de los campos gravitatorios y al procesamiento de las observaciones gravimétricas. A lo largo del texto desarrollaremos una descripción detallada de dos nuevas metodologías.

La primera consiste en un algoritmo para el cómputo de campos gravitatorios generados por geometrías definidas en coordenadas esféricas conocidas como *teseroides* (o prismas esféricos) cuya densidad varía a lo largo de la dirección radial. El principal interés sobre este nuevo método reside en su potencial aplicación al modelado de estructuras regionales o continentales que presenten variaciones de densidad en su profundidad, como pueden ser grandes cuencas sedimentarias o determinadas regiones del manto litosférico o astenosférico. Dicha metodología y sus resultados han sido publicados en *Geophysical Journal International* bajo el título “Gravitational field calculation in spherical coordinates using variable densities in depth” ([Soler et al., 2019a](#)). Una preimpresión del artículo se encuentra disponible bajo licencia Creative Commons Atribución 4.0 Internacional en [EarthArXiv: https://doi.org/10.31223/osf.io/3548g](#).

La segunda metodología que procederemos a describir consiste en un algoritmo para la interpolación de datos gravimétricos basado en el concepto de fuentes equivalentes que incorpora elementos de aprendizaje automático (*machine learning*). Más precisamente, presentamos las fuentes equivalentes potenciadas por gradiente (*gradient-boosted equivalent sources*), las cuales permiten realizar interpolaciones de grandes cantidades datos de cualquier campo potencial armónico, reduciendo considerablemente los requerimientos computacionales. Esta nueva metodología ha sido publicada en *Geophysical Journal International* bajo el título “Gradient-boosted equivalent sources” ([Soler y Uieda, 2021a](#)), junto con todas las pruebas realizadas sobre datos sintéticos y una aplicación sobre un gran conjunto de datos gravimétricos sobre Australia. Una preimpresión del artículo se encuentra disponible bajo licencia Creative Commons Atribución 4.0 Internacional en [EarthArXiv: https://doi.org/10.31223/X58G7C](#).

En tercer lugar, haremos una descripción de las herramientas disponibles en las librerías de Fatiando a Terra ([Uieda et al., 2013](#)): un proyecto de código abierto para Geofísica, en el cual el autor de esta Tesis ha realizado múltiples contribuciones. Recorreremos los orígenes e historia del proyecto, describiremos las herramientas desarrolladas en él y algunos de los métodos implementados en ellas, mostraremos algunos ejemplos sencillos y pondremos en evidencia el im-

pacto de Fatiando a Terra en la comunidad geocientífica internacional y en cómo este tipo de software resulta indispensable para la reproducibilidad de las investigaciones científicas. Además, expondremos las prácticas utilizadas para el desarrollo de estas herramientas, tales como el manejo de controladores de versiones, el uso de pruebas de software, la utilización de integración continua y la redacción de documentación, entre otras.

Por último finalizaremos con un Capítulo donde se detallan las conclusiones de todo lo desarrollado a lo largo de esta Tesis de manera global, recorriendo cada una de las metodologías aquí introducidas y cómo se vincula el proyecto Fatiando a Terra con las mismas.

Capítulo 2

Fundamentos Geofísicos

2.1. Potencial gravitatorio

Según la Ley de Gravitación Universal de Newton, una masa puntual m_p situada en la posición \mathbf{p} se ve sometida una fuerza gravitatoria \mathbf{F} debido a la presencia de otra masa m situada en la posición \mathbf{q} . Esta fuerza puede expresarse como:

$$\mathbf{F} = -G \frac{m_p m}{\|\mathbf{p} - \mathbf{q}\|^3} (\mathbf{p} - \mathbf{q}), \quad (2.1)$$

donde $G = 6.67430 \times 10^{-11} \text{m}^3 \text{kg}^{-1} \text{s}^{-2}$ es la constante de gravedad universal y $\|\cdot\|$ representa la norma L₂ (Fig. 2.1a). Aplicando la segunda Ley de Newton, se deduce que la masa m_p experimenta una aceleración gravitatoria \mathbf{g} :

$$\mathbf{g} = -\frac{Gm}{\|\mathbf{p} - \mathbf{q}\|^3} (\mathbf{p} - \mathbf{q}). \quad (2.2)$$

Si consideramos que la partícula m_p es una *partícula de prueba*, podemos reescribir la ecuación 2.2, pero ahora resignificándola como la aceleración gravitatoria que sentiría cualquier partícula localizada en un punto \mathbf{p} debido a la presencia de la partícula m :

$$\mathbf{g}(\mathbf{p}) = -\frac{Gm}{\|\mathbf{p} - \mathbf{q}\|^3} (\mathbf{p} - \mathbf{q}). \quad (2.3)$$

Se puede demostrar que la atracción gravitatoria \mathbf{g} define un campo irrotacional, es decir:

$$\nabla \times \mathbf{g} = 0. \quad (2.4)$$

Por ende es posible definir un potencial escalar V al que denominaremos *potencial gravitatorio*:

$$\mathbf{g}(\mathbf{p}) = +\nabla V(\mathbf{p}), \quad (2.5)$$

donde

$$V(\mathbf{p}) = G \frac{m}{\|\mathbf{p} - \mathbf{q}\|}. \quad (2.6)$$

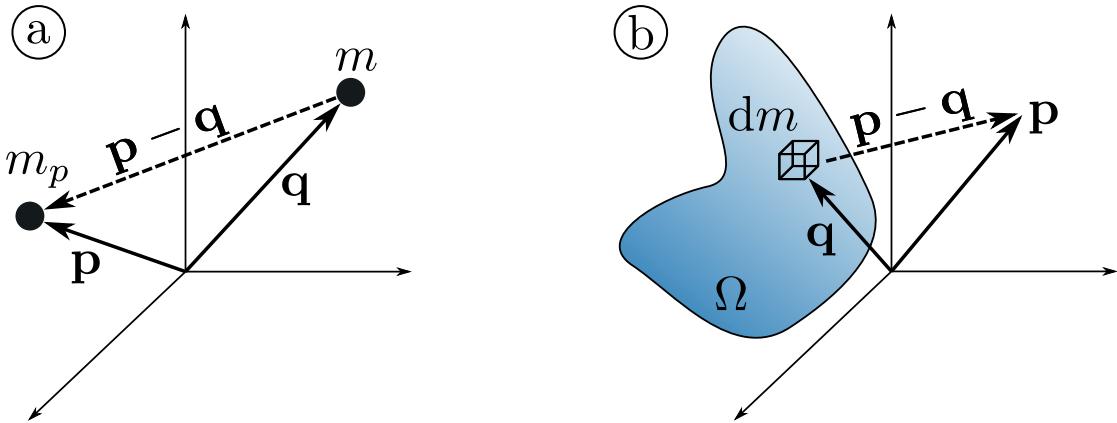


Figura 2.1: (a) Masas puntuales m_p y m localizadas en \mathbf{p} y \mathbf{q} , respectivamente. (b) Distribución de masa Ω , diferencial de masa dm ubicado en el punto \mathbf{q} .

Vale la pena notar que en la ecuación 2.5 se ha aplicado la convención de signo positivo para definir el potencial V , mayormente utilizada en la literatura sobre Geofísica y Geodesia ([Blakely, 1995](#); [Heiskanen y Moritz, 1967](#); [Hinze et al., 2009](#)).

2.1.1. Potencial gravitatorio por una distribución de masa

Partiendo de que un diferencial de masa dm ubicado en \mathbf{q} genera un potencial gravitatorio dV en cualquier punto \mathbf{p} (Fig. 2.1b):

$$dV(\mathbf{p}) = \frac{G}{\|\mathbf{p} - \mathbf{q}\|} dm, \quad (2.7)$$

el potencial gravitatorio generado por una distribución de masa Ω puede calcularse integrando los diferenciales de masa que lo componen:

$$V(\mathbf{p}) = G \int_{\Omega} \frac{dm}{\|\mathbf{p} - \mathbf{q}\|}. \quad (2.8)$$

Si reescribimos los diferenciales de masa como

$$dm = \rho(\mathbf{q}) dv, \quad (2.9)$$

donde $\rho(\mathbf{q})$ es la densidad de masa de la distribución Ω en el punto \mathbf{q} y dv es el diferencial de volumen, el potencial se puede expresar como:

$$V(\mathbf{p}) = G \int_{\Omega} \frac{\rho(\mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|} dv. \quad (2.10)$$

2.1.2. Gradiente del potencial gravitatorio

Según la definición del potencial gravitatorio expuesta en la ecuación 2.5, es posible calcular la aceleración de la gravedad producida por una distribución de masa arbitraria en cualquier punto \mathbf{p} como el gradiente del potencial $V(\mathbf{p})$. Si definimos un sistema de ejes Cartesianos en el cual el punto $\mathbf{p} = (x, y, z)$, las componentes de la aceleración en cada una de las direcciones del sistema quedan expresadas como:

$$g_i(\mathbf{p}) = \frac{\partial V(\mathbf{p})}{\partial i}, \quad \forall i \in \{x, y, z\}. \quad (2.11)$$

El vector $\mathbf{g}(\mathbf{p}) = (g_x(\mathbf{p}), g_y(\mathbf{p}), g_z(\mathbf{p}))$ representa la aceleración gravitatoria en el punto \mathbf{p} generada por la distribución de masa Ω , aunque es común referirse al mismo como el *gradiente gravitatorio*. Reemplazando la ecuación 2.10 en 2.11, las componentes del gradiente gravitatorio pueden expresarse como:

$$g_x(\mathbf{p}) = -G \int_{\Omega} \rho(\mathbf{q}) \frac{(x - x')}{\|\mathbf{p} - \mathbf{q}\|^3} dv, \quad (2.12)$$

$$g_y(\mathbf{p}) = -G \int_{\Omega} \rho(\mathbf{q}) \frac{(y - y')}{\|\mathbf{p} - \mathbf{q}\|^3} dv, \quad (2.13)$$

$$g_z(\mathbf{p}) = -G \int_{\Omega} \rho(\mathbf{q}) \frac{(z - z')}{\|\mathbf{p} - \mathbf{q}\|^3} dv, \quad (2.14)$$

donde $\mathbf{q} = (x', y', z')$.

Las segundas derivadas del potencial gravitatorio definen el *tensor del gradiente gravitatorio*, y sus componentes pueden expresarse como:

$$g_{ij}(\mathbf{p}) = \frac{\partial^2 V(\mathbf{p})}{\partial i \partial j}, \quad \forall i \in \{x, y, z\}, \quad \forall j \in \{x, y, z\}. \quad (2.15)$$

Reemplazando la ecuación 2.10, podemos expresar las componentes diagonales del tensor de la siguiente manera (Grombein et al., 2013):

$$g_{xx}(\mathbf{p}) = G \int_{\Omega} \rho(\mathbf{q}) \left[\frac{3(x - x')^2}{\|\mathbf{p} - \mathbf{q}\|^5} - \frac{1}{\|\mathbf{p} - \mathbf{q}\|^3} \right] dv, \quad (2.16)$$

$$g_{yy}(\mathbf{p}) = G \int_{\Omega} \rho(\mathbf{q}) \left[\frac{3(y - y')^2}{\|\mathbf{p} - \mathbf{q}\|^5} - \frac{1}{\|\mathbf{p} - \mathbf{q}\|^3} \right] dv, \quad (2.17)$$

$$g_{zz}(\mathbf{p}) = G \int_{\Omega} \rho(\mathbf{q}) \left[\frac{3(z - z')^2}{\|\mathbf{p} - \mathbf{q}\|^5} - \frac{1}{\|\mathbf{p} - \mathbf{q}\|^3} \right] dv, \quad (2.18)$$

mientras que las componentes no diagonales pueden expresarse como:

$$g_{xy}(\mathbf{p}) = g_{yx}(\mathbf{p}) = G \int_{\Omega} \rho(\mathbf{q}) \frac{3(x-x')(y-y')}{\|\mathbf{p}-\mathbf{q}\|^5} dv, \quad (2.19)$$

$$g_{xz}(\mathbf{p}) = g_{zx}(\mathbf{p}) = G \int_{\Omega} \rho(\mathbf{q}) \frac{3(x-x')(z-z')}{\|\mathbf{p}-\mathbf{q}\|^5} dv, \quad (2.20)$$

$$g_{yz}(\mathbf{p}) = g_{zy}(\mathbf{p}) = G \int_{\Omega} \rho(\mathbf{q}) \frac{3(y-y')(z-z')}{\|\mathbf{p}-\mathbf{q}\|^5} dv. \quad (2.21)$$

A partir de las ecuaciones 2.16–2.18 podemos calcular el Laplaciano del potencial gravitatorio:

$$\nabla^2 V(\mathbf{p}) = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2}, \quad (2.22)$$

Dado que las tres expresiones se cancelan mutuamente (Blakely, 1995), podemos concluir que cualquier potencial gravitatorio es un *campo armónico*

$$\nabla^2 V(\mathbf{p}) = 0 \quad (2.23)$$

para cualquier punto de observación \mathbf{p} fuera de la distribución de masa Ω .

Vale notar que la ecuación de Laplace 2.23 es válida solo para regiones libres de masas, y representa un caso particular de la ecuación de Poisson (Blakely, 1995):

$$\nabla^2 V(\mathbf{p}) = -4\pi G \rho(\mathbf{p}), \quad (2.24)$$

válida para cualquier punto de observación \mathbf{p} .

2.2. Modelado directo

El cálculo de los efectos gravitatorios de un determinado cuerpo sobre uno o más puntos de observación se suele denominar modelado directo (*forward modelling* en inglés). Aunque las expresiones del potencial gravitatorio (ec. 2.10), las componentes de su gradiente (ecs. 2.12–2.14) y de su tensor (ecs. 2.16–2.18) aparentan ser sencillas, el cómputo de estas magnitudes para una geometría dada no resulta un problema trivial. Solo existen soluciones analíticas a dichas expresiones para determinadas geometrías sencillas o que guardan algún tipo de simetría.

En el desarrollo de esta Tesis nos resultan de principal interés trabajar con tres tipos de geometrías: masas puntuales, prismas rectangulares y prismas esféricos o *teseroides*.

2.2.1. Masas puntuales

La distribución de densidades de una masa puntual puede expresarse sencillamente mediante una Delta de Dirac ([Vladimirov, 1979](#)):

$$\rho(\mathbf{q}) = m \delta(\mathbf{q} - \mathbf{q}'), \quad (2.25)$$

donde \mathbf{q} y m son la posición y la masa de la partícula, respectivamente. Reemplazando esta expresión en la ecuación [2.10](#), obtenemos el potencial gravitatorio generado por una partícula:

$$V(\mathbf{p}) = G \int_{\Omega} m \frac{\delta(\mathbf{q} - \mathbf{q}')}{\|\mathbf{p} - \mathbf{q}'\|} dv = \frac{Gm}{\|\mathbf{p} - \mathbf{q}\|}, \quad (2.26)$$

de acuerdo con la expresión del potencial de la ecuación [2.6](#).

Realizando el mismo reemplazo en las ecuaciones correspondientes a las componentes del gradiente (ecs. [2.12–2.14](#)) arribamos a las siguientes expresiones:

$$g_x(\mathbf{p}) = -Gm \frac{x - x'}{\|\mathbf{p} - \mathbf{q}\|^3}, \quad (2.27)$$

$$g_y(\mathbf{p}) = -Gm \frac{y - y'}{\|\mathbf{p} - \mathbf{q}\|^3}, \quad (2.28)$$

$$g_z(\mathbf{p}) = -Gm \frac{z - z'}{\|\mathbf{p} - \mathbf{q}\|^3}. \quad (2.29)$$

A la hora de realizar el cómputo de los campos gravitatorios de masas puntuales, las posiciones de las partículas como de los puntos de observación pueden venir dados en diferentes sistemas de coordenadas. En el caso de necesitar modelar cualquier conjunto de masas situadas bajo la superficie terrestre, resulta natural utilizar coordenadas esféricas tanto para las posiciones de las partículas como para los puntos de observación. Sin embargo, es muy común trabajar sobre zonas de estudio con dimensiones acotadas, bajo las cuales la aproximación de Tierra plana es suficiente. En estos casos es posible trabajar en coordenadas Cartesianas.

Coordenadas Cartesianas

En el caso de que la posición de la partícula venga dada por el vector $\mathbf{q} = (x', y', z')$ y el punto de observación por el vector $\mathbf{p} = (x, y, z)$ definidos en el mismo sistema de ejes Cartesianos, el módulo de $\mathbf{p} - \mathbf{q}$ se calcula sencillamente bajo la norma L₂:

$$\|\mathbf{p} - \mathbf{q}\| = \sqrt{(x - x')^2 + (y - y')^2 + (z - z')^2}. \quad (2.30)$$

Dado que los numeradores en las ecuaciones [2.27–2.29](#) resultan triviales en coordenadas Cartesianas, el cómputo del potencial y las componentes de su gradiente se puede realizar de manera sencilla.

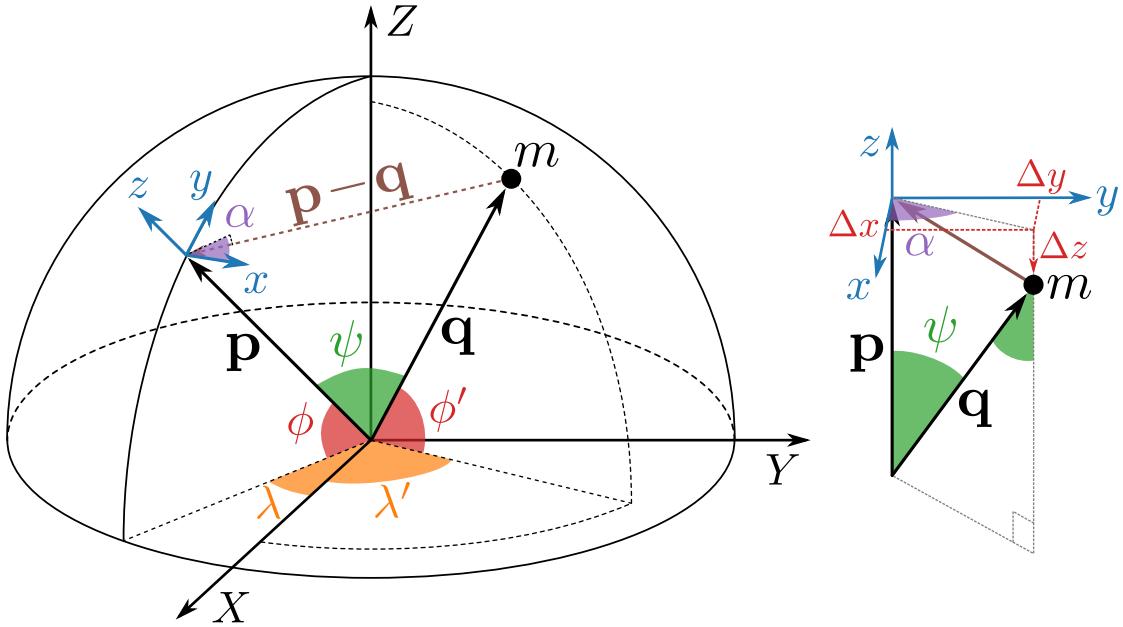


Figura 2.2: Masa puntual m ubicada en la posición \mathbf{q} y punto de observación situado en \mathbf{p} . Se define un sistema de referencia Cartesiano geocéntrico X, Y, Z bajo el cual se considera un sistema de coordenadas esféricas dadas por la distancia radial (r), ángulo longitudinal (λ) y ángulo latitudinal (ϕ). En el punto \mathbf{p} se define un sistema *local* de coordenadas Cartesiano: el eje x apunta hacia el Este, el eje y hacia el Norte geométrico y el eje z en la dirección radial. Se define a ψ como el ángulo descripto por los vectores \mathbf{p} y \mathbf{q} , y α al ángulo azimutal de $\mathbf{q} - \mathbf{p}$ sobre el sistema *local*.

Coordenadas esféricas

Consideremos que las posiciones del punto de observación y de la partícula vienen dados en un sistema de coordenadas esféricas. Para ello vamos a definir un sistema de referencia *geocéntrico* $\{X, Y, Z\}$ y a partir del mismo un sistema de coordenadas esféricas $\{r, \lambda, \phi\}$, donde r es la posición *radial*, λ la posición *longitudinal* y ϕ la posición *latitudinal* (Fig. 2.2). Cualquier punto dado por las coordenadas esféricas (r, λ, ϕ) se puede expresar en coordenadas Cartesianas geocéntricas bajo las siguientes relaciones:

$$\begin{cases} X = r \cos \lambda \cos \phi \\ Y = r \sin \lambda \cos \phi \\ Z = r \sin \phi, \end{cases} \quad (2.31)$$

donde $r \in [0, \infty)$, $\lambda \in (-\pi, \pi]$ y $\phi \in [-\pi/2, \pi/2]$.

El potencial gravitatorio que genera la partícula m sobre el punto de observación \mathbf{p} puede calcularse mediante la ecuación 2.6. Considerando que los vectores \mathbf{p} y \mathbf{q} vienen definidos por las coordenadas esféricas (r, λ, ϕ) y (r', λ', ϕ') , res-

pectivamente, la distancia Euclíadiana entre ellos puede expresarse como (Grombein et al., 2013):

$$\|\mathbf{p} - \mathbf{q}\| = \sqrt{r^2 + r'^2 - 2rr' \cos \psi}, \quad (2.32)$$

donde ψ es el ángulo que describen los vectores \mathbf{p} y \mathbf{q} :

$$\cos \psi = \sin \phi \sin \phi' + \cos \phi \cos \phi' \cos(\lambda - \lambda'). \quad (2.33)$$

La determinación de las componentes del gradiente del potencial gravitatorio hace necesario que definamos las direcciones ortogonales x , y y z a lo largo de las cuales se calculan las derivadas parciales de $V(\mathbf{p})$. La elección más natural es obtener el gradiente con respecto a un sistema de coordenadas *local* $\{x, y, z\}$ (Grombein et al., 2013; Uieda et al., 2016) donde x se orienta en la dirección longitudinal (Este), y en la dirección latitudinal (Norte) y z en la dirección radial.

Los numeradores de las ecuaciones 2.27–2.29 pueden expresarse en términos de las coordenadas esféricas como:

$$\Delta x = -(x - x') = r' \sin \psi \cos \alpha \quad (2.34)$$

$$\Delta y = -(y - y') = r' \sin \psi \sin \alpha \quad (2.35)$$

$$\Delta z = -(z - z') = r' \cos \psi - r, \quad (2.36)$$

donde α es el ángulo azimutal del vector $\mathbf{q} - \mathbf{p}$ sobre el sistema de coordenadas *locales*. Haciendo uso de las siguientes relaciones de trigonometría esférica (Heiskanen y Moritz, 1967, p. 113):

$$\sin \psi \cos \alpha = \cos \phi \sin \phi' - \sin \phi \cos \phi' \cos(\lambda - \lambda') \quad (2.37)$$

$$\sin \psi \sin \alpha = \cos \phi' \sin(\lambda - \lambda'), \quad (2.38)$$

podemos expresar las ecuaciones 2.34, 2.35 y 2.36 como (Grombein et al., 2013):

$$\Delta x = r' \sin \psi [\cos \phi \sin \phi' - \sin \phi \cos \phi' \cos(\lambda - \lambda')] \quad (2.39)$$

$$\Delta y = r' \cos \phi' \sin(\lambda - \lambda') \quad (2.40)$$

$$\Delta z = r' \cos \psi - r. \quad (2.41)$$

Las componentes del gradiente del potencial gravitatorio generado por una masa puntual en coordenadas esféricas quedan determinadas de la siguiente manera:

$$g_x(\mathbf{p}) = Gm \frac{\Delta x}{\|\mathbf{p} - \mathbf{q}\|^3}, \quad (2.42)$$

$$g_y(\mathbf{p}) = Gm \frac{\Delta y}{\|\mathbf{p} - \mathbf{q}\|^3}, \quad (2.43)$$

$$g_z(\mathbf{p}) = Gm \frac{\Delta z}{\|\mathbf{p} - \mathbf{q}\|^3}, \quad (2.44)$$

haciendo uso de las ecuaciones 2.32, 2.33, 2.39, 2.40 y 2.41.

2.2.2. Prismas rectangulares

A la hora de modelar estructuras tridimensionales subyacentes a la superficie terrestre, los prismas rectangulares se presentan como la geometría a elección por parte de muchos autores y muchas autoras. Una de las razones recae en la simpleza de la geometría, que resulta muy útil para modelar estructuras geológicas. Aunque quizás la principal razón es la existencia de soluciones analíticas a las ecuaciones de los campos gravitatorios generados por prismas rectangulares (Nagy et al., 2000, 2002).

Consideremos un prisma rectangular de densidad homogénea ρ cuyos vértices se encuentran determinados por las coordenadas $X_1, X_2, Y_1, Y_2, Z_1, Z_2$ definidas en un sistema de ejes Cartesianos X, Y, Z (Fig. 2.3). El potencial gravitatorio que genera el prisma sobre un punto de observación $\mathbf{p} = (X, Y, Z)$ definido en el sistema X, Y, Z se puede calcular mediante la ecuación 2.10, reemplazando el dominio de integración:

$$V(\mathbf{p}) = G\rho \int_{X_1}^{X_2} \int_{Y_1}^{Y_2} \int_{Z_1}^{Z_2} \frac{dX' dY' dZ'}{\|\mathbf{p} - \mathbf{q}\|}, \quad (2.45)$$

donde

$$\|\mathbf{p} - \mathbf{q}\| = \sqrt{(X - X')^2 + (Y - Y')^2 + (Z - Z')^2}. \quad (2.46)$$

Con el objetivo de simplificar los cálculos, definimos un sistema de coordenadas Cartesianas $\{x, y, z\}$ cuyo origen se encuentra en el punto de observación \mathbf{p} y sus ejes son paralelos a los del sistema $\{X, Y, Z\}$, respectivamente (Fig. 2.3). Los vértices del prisma vendrán dados por las coordenadas $x_1, x_2, y_1, y_2, z_1, z_2$ en el nuevo sistema de referencia, es decir:

$$\begin{aligned} x_1 &= X_1 - X, & x_2 &= X_2 - X, \\ y_1 &= Y_1 - Y, & y_2 &= Y_2 - Y, \\ z_1 &= Z_1 - Z, & z_2 &= Z_2 - Z. \end{aligned} \quad (2.47)$$

Bajo este nuevo sistema de referencia, el potencial gravitatorio se puede expresar como:

$$V(\mathbf{p}) = G\rho \int_{x_1}^{x_2} \int_{y_1}^{y_2} \int_{z_1}^{z_2} \frac{dx' dy' dz'}{\sqrt{x'^2 + y'^2 + z'^2}}. \quad (2.48)$$

El resultado de esta integración posee la siguiente forma (Nagy et al., 2000, 2002):

$$\begin{aligned} V(\mathbf{p}) = G\rho \left[& xy \ln(z + l) + yz \ln(x + l) + zx \ln(y + l) \right. \\ & \left. - \frac{x^2}{2} \arctan \frac{yz}{xl} - \frac{y^2}{2} \arctan \frac{zx}{yl} - \frac{z^2}{2} \arctan \frac{xy}{zl} \right] \Big|_{x_1}^{x_2} \Big|_{y_1}^{y_2} \Big|_{z_1}^{z_2}, \end{aligned} \quad (2.49)$$

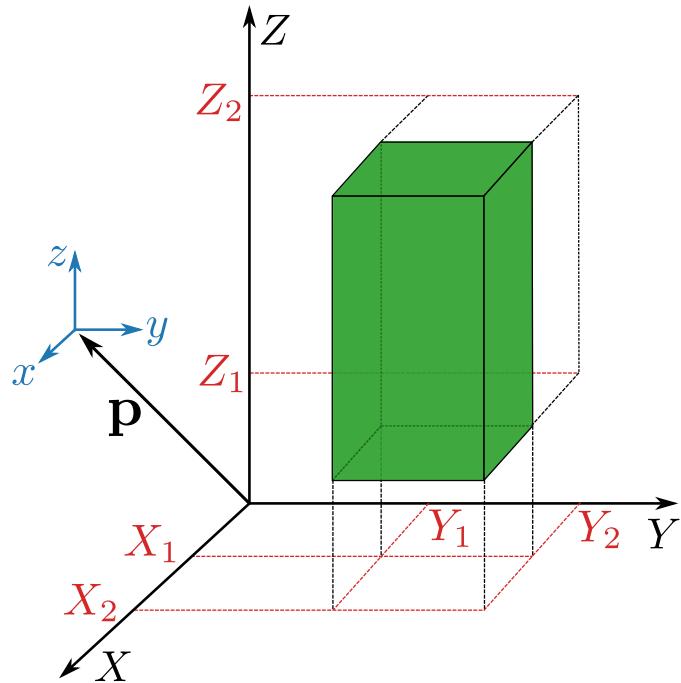


Figura 2.3: Prisma rectangular y punto de observación \mathbf{p} definidos en un sistema de coordenadas Cartesianas $\{X, Y, Z\}$. Se define un sistema de coordenadas Cartesianas $\{x, y, z\}$ con origen en el punto de observación \mathbf{p} y con ejes paralelos a X, Y, Z , respectivamente.

donde

$$l = \sqrt{x^2 + y^2 + z^2}. \quad (2.50)$$

Reemplazando todos los límites del dominio de integración, obtenemos un total de 48 términos. A la hora de evaluarlos numéricamente hay que tener en cuenta que la solución analítica de la ecuación 2.49 no es continua en todo \mathbb{R}^3 , particularmente en los casos en los que el punto de observación se encuentra sobre alguno de los vértices, aristas o caras del prisma ($x_{1,2} = 0, y_{1,2} = 0$ ó $z_{1,2} = 0$). Sin embargo el potencial $V(\mathbf{p})$ sí lo es. Para solucionar esto, es necesario reemplazar aquellos términos que no pueden ser evaluados por sus límites cuando $(x, y, z) \rightarrow (0, 0, 0)$ (Nagy et al., 2000). Por ejemplo:

$$\lim_{(x,y,z) \rightarrow (0,0,0)} xz \ln(z + r) = 0. \quad (2.51)$$

Una solución alternativa es reemplazar la función \ln por una nueva función $\ln2(x)$ definida de la siguiente manera:

$$\ln2(x) = \begin{cases} \ln(x) & x \geq x_{\text{umbral}} \\ 0 & x < x_{\text{umbral}} \end{cases} \quad (2.52)$$

donde x_{umbral} es un valor muy pequeño del argumento de $\ln2$ a partir del cual consideramos que la función debe ser evaluada en su límite en cero. La elección

de este valor dependerá de las dimensiones del problema que estamos resolviendo. En el caso de un modelado directo geofísico, en el cual las dimensiones de los primos vienen dadas por varios metros, podemos asumir un valor de $x_{\text{umbral}} = 10^{-10}\text{m}$ ([Soler et al., 2021b](#)).

Por otro lado, la evaluación de la función arctan requiere ciertos cuidados. Para que el potencial $V(\mathbf{p})$ satisfaga la ecuación de Poisson (ec. 2.24), es necesario utilizar la siguiente forma de la función arctan2(y, x) ([Fukushima, 2020](#)):

$$\arctan2(y, x) = \begin{cases} \arctan(y/x) & x \neq 0 \\ \pi/2 & x = 0, y > 0 \\ -\pi/2 & x = 0, y < 0 \\ 0 & x = 0, y = 0. \end{cases} \quad (2.53)$$

Haciendo uso de las funciones ln2 y arctan2 definidas en las ecuaciones 2.52 y 2.53, respectivamente, el potencial gravitatorio generado por un prisma rectangular en cualquier punto de observación \mathbf{p} puede ser numéricamente calculado mediante:

$$V(\mathbf{p}) = G\rho \left[xy \ln2(z+l) + yz \ln2(x+l) + zx \ln2(y+l) - \frac{x^2}{2} \arctan2(yz, xl) \right. \\ \left. - \frac{y^2}{2} \arctan2(zx, yl) - \frac{z^2}{2} \arctan2(xy, xl) \right] \Big|_{x_1}^{x_2} \Big|_{y_1}^{y_2} \Big|_{z_1}^{z_2}. \quad (2.54)$$

De manera análoga, podemos calcular las componentes del gradiente del potencial gravitatorio generado por un prisma rectangular haciendo uso de las siguientes expresiones ([Nagy et al., 2000, 2002](#)):

$$g_x(\mathbf{p}) = \left[y \ln2(z+l) + z \ln2(y+l) - x \arctan2(yz, xl) \right] \Big|_{x_1}^{x_2} \Big|_{y_1}^{y_2} \Big|_{z_1}^{z_2} \quad (2.55)$$

$$g_y(\mathbf{p}) = \left[z \ln2(x+l) + x \ln2(z+l) - y \arctan2(zx, yl) \right] \Big|_{x_1}^{x_2} \Big|_{y_1}^{y_2} \Big|_{z_1}^{z_2} \quad (2.56)$$

$$g_z(\mathbf{p}) = \left[x \ln2(y+l) + y \ln2(x+l) - z \arctan2(xy, xl) \right] \Big|_{x_1}^{x_2} \Big|_{y_1}^{y_2} \Big|_{z_1}^{z_2}. \quad (2.57)$$

2.2.3. Teseroides o prismas esféricos

En caso de querer realizar modelos directos de estructuras regionales, continentales o globales, el uso de prismas rectangulares en coordenadas Cartesianas no resulta la mejor opción, ya que no tienen en cuenta la curvatura del planeta Tierra. La mayoría de los modelos directos que sí lo hacen se definen en coordenadas esféricas geocéntricas y consisten en discretizar la Tierra en geometrías simples. Este es el caso de los *teseroides* ([Anderson, 1976](#)), también conocidos

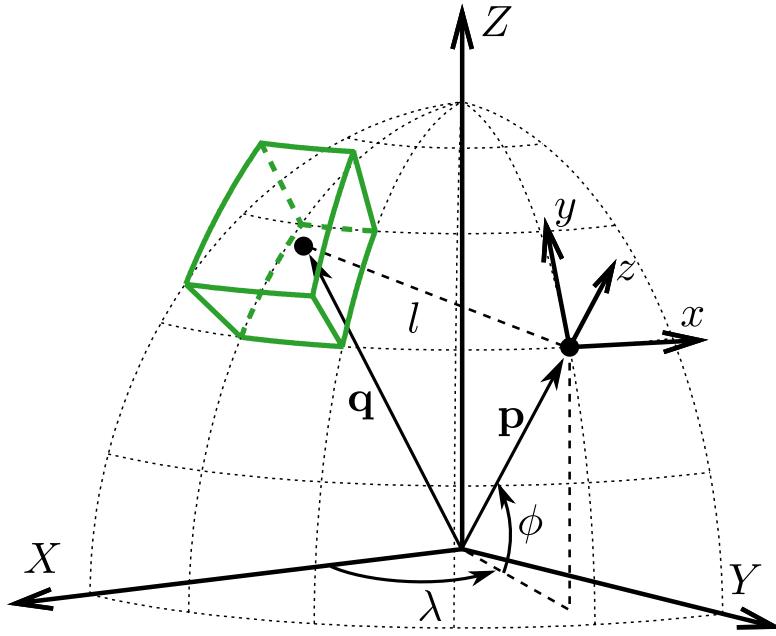


Figura 2.4: Teseroide (prisma esférico) en un sistema de coordenadas esféricas geocéntricas, junto a un punto de cálculo situado en \mathbf{p} y su correspondiente sistema *local* de coordenadas Cartesianas. Obtenida de [Uieda \(2015\)](#).

como prismas esféricos, los cuales se definen como el volumen comprendido entre dos esferas de radios distintos, dos meridianos de longitudes distintas y dos paralelos de distintas latitudes (Fig. 2.4).

El potencial gravitatorio que genera un teseroide de densidad uniforme ρ , definido por las coordenadas $r_1, r_2, \lambda_1, \lambda_2, \phi_1$ y ϕ_2 , sobre un punto de observación \mathbf{p} , dado por las coordenadas esféricas r, λ y ϕ , se puede obtener aplicando estos límites de integración a la ecuación 2.10, quedando expresado como la siguiente integral volumétrica ([Grombein et al., 2013; Uieda et al., 2016](#)):

$$V(\mathbf{p}) = G\rho \int_{r_1}^{r_2} \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \frac{\kappa}{\|\mathbf{p} - \mathbf{q}\|} dr' d\lambda' d\phi', \quad (2.58)$$

donde

$$\kappa = r'^2 \cos \phi', \quad (2.59)$$

la distancia $\|\mathbf{p} - \mathbf{q}\|$ viene dada por la ecuación 2.32 y las coordenadas de integración r' , λ' y ϕ' representan la posición \mathbf{q} de cada volumen infinitesimal del teseroide.

Reemplazando los límites de integración de las ecuaciones 2.12, 2.13 y 2.14 por los del teseroide, podemos obtener las componentes del gradiente del potencial gravitatorio generado por el teseroide en el punto de observación \mathbf{p} con

respecto al sistema de coordenadas *local*:

$$g_x(\mathbf{p}) = G\rho \int_{r_1}^{r_2} \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \frac{\Delta x}{\|\mathbf{p} - \mathbf{q}\|^3} \kappa dr' d\lambda' d\phi', \quad (2.60)$$

$$g_y(\mathbf{p}) = G\rho \int_{r_1}^{r_2} \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \frac{\Delta y}{\|\mathbf{p} - \mathbf{q}\|^3} \kappa dr' d\lambda' d\phi', \quad (2.61)$$

$$g_z(\mathbf{p}) = G\rho \int_{r_1}^{r_2} \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \frac{\Delta z}{\|\mathbf{p} - \mathbf{q}\|^3} \kappa dr' d\lambda' d\phi', \quad (2.62)$$

donde Δx , Δy y Δz vienen dadas por las ecuaciones 2.39, 2.40 y 2.41, respectivamente.

Las ecuaciones 2.58, 2.60, 2.61 y 2.62 contienen integrales elípticas que no poseen solución analítica y deben ser aproximadas numéricamente. Existen dos principales métodos en la literatura para aproximar estas integrales: uno involucra expansiones en serie de Taylor (Grombein et al., 2013; Heck y Seitz, 2006) mientras que los otros hacen uso de la Cuadratura de Gauss-Legendre (GLQ) (Asgharzadeh et al., 2007; Li et al., 2011; Lin y Denker, 2018; Uieda et al., 2016; Wild-Pfeiffer, 2008). En el Capítulo 3 se detallan las razones por las cuales a lo largo de esta Tesis abordaremos la estrategia de la GLQ.

La GLQ consiste en la aproximación de una integral por una suma ponderada del integrando evaluado en determinadas abscisas (Hildebrand, 1987, p. 390):

$$\int_{-1}^1 f(x) dx \approx \sum_{i=1}^N W_i f(x_i), \quad (2.63)$$

donde N es el orden de la cuadratura, x_i corresponden a las raíces del polinomio de Legendre $P_N(x)$ de grado N , y W_i son los pesos ponderados de cada término:

$$W_i = \frac{2}{(1 - x_i^2) [P'_N(x_i)]^2}. \quad (2.64)$$

El polinomio de Legendre $P_N(x)$ se puede obtener mediante relaciones de recurrencia (Hildebrand, 1987, p. 330):

$$P_0(x) = 1 \quad (2.65)$$

$$P_1(x) = x \quad (2.66)$$

$$P_{N+1}(x) = \frac{2N+1}{N+1} x P_N(x) - \frac{N}{N+1} P_{N-1}(x). \quad (2.67)$$

Tabla 2.1: Raíces de algunos polinomios de Legendre junto con los correspondientes pesos ponderados a ser utilizados en la GLQ ([Hildebrand, 1987](#), p. 392).

Orden (N)	Raíces (x_i)	Pesos ponderados (W_i)
1	0	2
2	$\pm \frac{1}{\sqrt{3}}$	1
3	0	$\frac{8}{9}$
	$\pm \sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	$\pm \sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$
	$\pm \sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36}$

Mientras que los valores que asume su primera derivada $P'_N(x)$ en las raíces x_i se pueden obtener gracias a la siguiente relación ([Hildebrand, 1987](#), p. 391):

$$P'_N(x_i) = \frac{N}{(1-x_i^2)} P_{N-1}(x_i). \quad (2.68)$$

La Tabla 2.1 muestra las raíces de los primeros polinomios de Legendre y sus respectivos pesos ponderados.

En el caso de que la integral posea límites de integración distintos a $(-1, 1)$, podemos escalar los nodos y aplicar la GLQ de la siguiente forma:

$$\int_a^b f(x) dx \approx \frac{b-a}{2} \sum_{i=1}^N W_i f(x_i^s), \quad (2.69)$$

donde

$$x_i^s = \frac{b-a}{2} x_i + \frac{b+a}{2}. \quad (2.70)$$

Haciendo uso de la GLQ, podemos aproximar las integrales volumétricas de las ecuaciones 2.58, 2.60, 2.61, 2.62 de la siguiente manera ([Asgharzadeh et al., 2007](#); [Uieda et al., 2016](#)):

$$\int_{r_1}^{r_2} \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} f(r', \lambda', \phi') dr' d\lambda' d\phi' = A \sum_{i=1}^{N_r} \sum_{j=1}^{N_\lambda} \sum_{k=1}^{N_\phi} W_i^r W_j^\lambda W_k^\phi f(r_i, \lambda_j, \phi_k), \quad (2.71)$$

donde

$$A = \frac{(r_2 - r_1)(\lambda_2 - \lambda_1)(\phi_2 - \phi_1)}{8}, \quad (2.72)$$

N_r , N_λ y N_ϕ son los órdenes de la cuadratura en cada dirección; r_i , λ_j y ϕ_k representan los nodos escalados de las cuadraturas para cada dirección; y W_i^r , W_j^λ , W_k^ϕ sus respectivos pesos ponderados.

Como se puede observar en la ecuación 2.71, la GLQ aproxima el campo gravitatorio de un teseroide por el que genera un conjunto de masas puntuales situadas en los nodos escalados de los polinomios de Legendre. La precisión de la aproximación dependerá entonces de la cantidad de nodos utilizados en las sumatorias, es decir, de los órdenes de las cuadraturas. Es posible entonces aumentar la precisión de la GLQ incrementando los órdenes de las cuadraturas. Sin embargo, el aumento de los órdenes de la cuadratura incrementa la cantidad de nodos según $O(n^3)$, haciendo de esta estrategia un método muy ineficiente.

Ku (1977) demostró que la precisión de la aproximación también depende del cociente entre la distancia al punto de observación y la distancia entre nodos adyacentes. Haciendo uso de este conocimiento, Li et al. (2011) han propuesto un método más eficiente para aumentar la precisión de la aproximación manteniendo el orden de la GLQ constante: la *discretización adaptativa*. Esta consiste en ir dividiendo el teseroide en otros teseroides de menor tamaño según la distancia entre ellos y el punto de observación. De esta forma, se incrementa la cantidad de nodos en las regiones más próximas al punto de cómputo, es decir, donde más afecta a la precisión. Al compararlo con un aumento del orden de la cuadratura, la *discretización adaptativa* permite hacer uso de una cantidad considerablemente menor de nodos, pero alcanzando el mismo nivel de precisión. En el Capítulo 3 describiremos en detalle este método.

2.3. Gravedad terrestre

Un cuerpo en reposo situado sobre la superficie de la Tierra experimenta una fuerza de atracción gravitatoria debida a la masa del planeta. Las observaciones que se realizan sobre dicho cuerpo suelen situarse sobre un sistema de referencia solidario a la superficie terrestre y rotando junto con ella. Este sistema se encuentra acelerado y por lo tanto constituye un sistema de referencia no inercial, es decir, un sistema sobre el cual no se cumplen las Leyes del Movimiento de Newton. Esto hace necesario que debamos introducir una aceleración centrífuga sobre dicho cuerpo, la cual no es producida por ningún tipo de interacción, sino que surge de utilizar este sistema de referencia no inercial para describir su movimiento. Llamaremos *gravedad* al vector \mathbf{g} definido como la suma entre la aceleración provocada por la atracción gravitatoria y la aceleración centrífuga. A partir de ella podemos definir un *potencial de gravedad* W :

$$\nabla W(\mathbf{p}) = +\mathbf{g}(\mathbf{p}). \quad (2.73)$$

El *potencial de gravedad* W puede ser expresado como:

$$W(\mathbf{p}) = V(\mathbf{p}) + \Phi(\mathbf{p}), \quad (2.74)$$

donde $V(\mathbf{p})$ es el *potencial gravitatorio* que genera la Tierra sobre el punto \mathbf{p} (ecuación 2.10), y $\Phi(\mathbf{p})$ el *potencial centrífugo* al cual se ve sometido el cuerpo situado en \mathbf{p} debido a la rotación del planeta. Este último puede expresarse como:

$$\Phi(\mathbf{p}) = \frac{1}{2}\omega^2(X^2 + Y^2), \quad (2.75)$$

donde ω es el módulo de la velocidad angular de rotación de la Tierra y X e Y son las coordenadas horizontales del punto \mathbf{p} en un sistema de referencia Cartesiano geocéntrico (Fig. 2.2).

Vale notar que el potencial W no es un campo armónico, ya que no satisface la ecuación de Laplace (ec. 2.23). En cambio, se rige acorde a una *ecuación de Poisson generalizada*:

$$\nabla^2 W(\mathbf{p}) = -4\pi G\rho(\mathbf{p}) + 2\omega^2. \quad (2.76)$$

Las superficies

$$W(\mathbf{p}) = \text{const}, \quad (2.77)$$

se definen como *superficies equipotenciales*.

A partir de la definición del vector de *gravedad* (ec. 2.73), es posible deducir que $\mathbf{g}(\mathbf{p})$ es un vector normal a la superficie equipotencial que pasa por el mismo punto \mathbf{p} . Cada una de las curvas que cortan normalmente a las superficies equipotenciales se definen como *líneas de fuerza* o *líneas de campo* (*plumb lines* en inglés).

La superficie equipotencial que coincide con el nivel medio del mar se define como el *geoide*:

$$W(\mathbf{p}) = W_0. \quad (2.78)$$

El geoide nos permite definir la *altitud ortométrica* H de cualquier punto \mathbf{p} como la distancia entre la superficie del geoide y \mathbf{p} medida a lo largo de la línea de fuerza que pasa por dicho punto (Fig. 2.6).

2.3.1. Gravedad normal

Las mediciones del campo gravitatorio terrestre nos permiten obtener información acerca de las masas que componen nuestro planeta. Los valores de este campo presentan ligeras variaciones a lo largo de la Tierra, originadas por la diferentes densidades de las masas del interior del planeta. Debido a que estas variaciones suelen ser lo suficientemente pequeñas, es posible asociarlas a la presencia de *cuerpos anómalos* bajo la superficie terrestre. Para poder identificarlos es necesario definir primero el concepto de *Tierra normal*.

En una primera aproximación podemos considerar la geometría de la Tierra como un elipsoide de revolución. Si bien esta aproximación no es lo suficientemente apropiada para muchas aplicaciones, nos provee de una descripción matemática sencilla, cuyas derivaciones poseen soluciones analíticas. De esta manera definiremos a la *Tierra normal* como un *elipsoide de referencia*, es decir, un

elipsoide de revolución que representa una superficie equipotencial del potencial de gravedad que este genera. Llamaremos $U(\mathbf{p})$ al *potencial de gravedad normal*, es decir, el potencial de gravedad generado por el elipsoide de referencia y definimos al elipsoide de referencia como la siguiente superficie equipotencial:

$$U(\mathbf{p}) = U_0 = \text{cte.} \quad (2.79)$$

El potencial de gravedad normal puede ser escrito como

$$U(\mathbf{p}) = V_{\text{el}}(\mathbf{p}) + \Phi(\mathbf{p}), \quad (2.80)$$

donde $\Phi(\mathbf{p})$ es el potencial centrífugo debido a la rotación del elipsoide de revolución (ec. 2.75) y $V_{\text{el}}(\mathbf{p})$ es el potencial gravitatorio que produce el mismo elipsoide.

El elipsoide de referencia vendrá definido mediante los siguientes parámetros:

1. la forma del elipsoide, dada por los dos semiejes a y b ,
2. la masa total M del elipsoide, y
3. la velocidad angular ω .

A partir de los dos semiejes podemos describir la superficie del elipsoide como:

$$\frac{X^2}{a^2} + \frac{Y^2}{b^2} + \frac{Z^2}{b^2} = 1 \quad (2.81)$$

donde X , Y y Z son coordenadas Cartesianas geocéntricas.

Tanto la masa del elipsoide como su velocidad angular resultarán útiles para definir cantidades como la gravedad normal. Vale notar que no resulta indispensable definir la distribución de densidad del elipsoide ([Heiskanen y Moritz, 1967](#), p. 64).

El elipsoide de referencia no solo nos permite tener un modelo de *Tierra normal*, sino que además nos permite definir un nuevo sistema de coordenadas: las *coordenadas geodésicas* o *geográficas*. Dado un punto arbitrario \mathbf{p} podemos especificar su ubicación mediante su *longitud* λ , su *latitud geodésica*¹ φ y su *altitud geométrica* h . El ángulo longitudinal λ coincide con la longitud del punto \mathbf{p} desde un sistema de coordenadas esféricas. La altura geométrica h corresponde a la distancia entre el punto \mathbf{p} y la superficie del elipsoide a lo largo de línea de fuerza del potencial U que pasa por el punto \mathbf{p} (Fig. 2.5). Debido a las propiedades del potencial U sus líneas de fuerza todas rectas que cortan normalmente a cada una de sus superficies equipotenciales. Por ende, la altitud geométrica h equivale a la longitud del segmento de recta que une el punto \mathbf{p} con la superficie del elipsoide a lo largo de la dirección normal al mismo. La latitud geodésica corresponde al ángulo comprendido entre el plano ecuatorial y la prolongación de una recta normal al la superficie del elipsoide que pasa por el punto \mathbf{p} (Fig. 2.5).

¹Se notará a la *latitud geodésica* con φ y a la *latitud esférica* con ϕ .

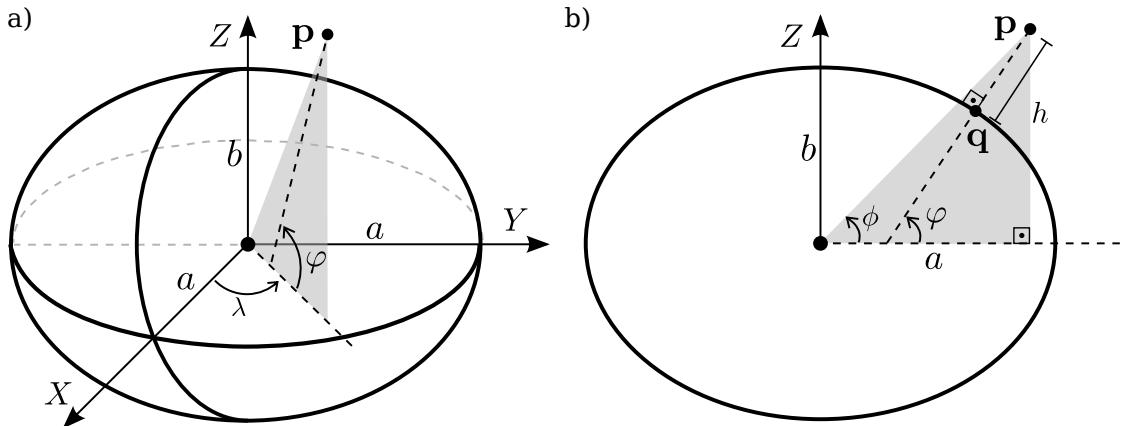


Figura 2.5: Elipsoide de referencia y coordenadas geodésicas del punto \mathbf{p} . (a) Elipsoide de referencia definido por los semiejes a y b , junto con un sistema Cartesiano geocéntrico $\{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$. Se observan los ángulos longitudinal λ y latitudinal φ del punto \mathbf{p} . (b) Sección del elipsoide por el plano que contiene al punto \mathbf{p} . Se destaca el punto \mathbf{q} correspondiente a la intersección de la superficie del elipsoide con la línea de fuerza que pasa por el punto \mathbf{p} . Se observan la latitud geodésica φ , la latitud esférica ϕ y la altitud geométrica h del punto \mathbf{p} . Esta figura es una versión modificada de Oliveira Jr et al. (2021b).

Definiremos el *vector gravedad normal* γ como el gradiente del potencial de gravedad normal:

$$\gamma(\mathbf{p}) = \nabla U(\mathbf{p}). \quad (2.82)$$

Vale notar que el gradiente de un campo en un punto \mathbf{p} consiste en un vector normal a la superficie equipotencial a la cual pertenece el punto \mathbf{p} . De esta manera, al calcular el gradiente del potencial de gravedad normal desde el sistema de coordenadas geodésicas podemos deducir que solo la componente normal es no nula (Heiskanen y Moritz, 1967, p. 68), es decir:

$$\gamma_\lambda = \frac{\partial U}{\partial \lambda} = 0, \quad (2.83)$$

$$\gamma_\varphi = \frac{\partial U}{\partial \varphi} = 0. \quad (2.84)$$

Definimos la *gravedad normal* como el módulo del vector de gravedad normal:

$$\gamma(\mathbf{p}) = |\gamma_h(\mathbf{p})| = \frac{\partial U(\mathbf{p})}{\partial h}. \quad (2.85)$$

Dada la simetría de rotación del elipsoide y el hecho que el potencial centrífugo no depende de la posición longitudinal, la gravedad normal depende únicamente de la altitud geométrica y la latitud geodésica del punto \mathbf{p} .

Fórmula cerrada

Existe una expresión analítica para la gravedad normal en cualquier punto externo al elipsoide de referencia dada por la *fórmula cerrada* (del inglés *closed-form formula*) de [Li y Götze \(2001a\)](#):

$$\gamma(\varphi, h) = \frac{1}{\Omega} \left\{ \frac{GM}{b'^2 + E^2} + \frac{\omega^2 a^2 Eq'}{(b'^2 + E^2)q_0} \left(\frac{1}{2} \sin^2 \beta' - \frac{1}{6} \right) - \omega^2 b' \cos^2 \beta' \right\}, \quad (2.86)$$

donde

$$E = \sqrt{a^2 - b^2} \quad \text{es la excentricidad lineal,} \quad (2.87)$$

$$\Omega = \sqrt{\frac{b'^2 + E^2 \sin^2 \beta'}{b'^2 + E^2}}, \quad (2.88)$$

$$q' = 3 \left(1 + \frac{b'^2}{E^2} \right) \left(1 - \frac{b'}{E} \arctan \frac{E}{b'} \right) - 1, \quad (2.89)$$

$$q_0 = \frac{1}{2} \left[\left(1 + \frac{3b^2}{E^2} \right) \arctan \frac{E}{b} - \frac{3b}{E} \right], \quad (2.90)$$

$$b' = \sqrt{r''^2 - E^2 \cos^2 \beta'}, \quad (2.91)$$

$$\cos \beta' = \sqrt{\frac{1}{2} + \frac{R}{2} - \sqrt{\frac{1}{4} + \frac{R^2}{4} - \frac{D}{2}}}, \quad (2.92)$$

y

$$R = \frac{r''^2}{E^2}, \quad (2.93)$$

$$D = \frac{d''^2}{E^2}, \quad (2.94)$$

$$r''^2 = r'^2 + z'^2, \quad (2.95)$$

$$d''^2 = r'^2 - z'^2, \quad (2.96)$$

$$r' = a \cos \beta + h \cos \varphi, \quad (2.97)$$

$$z' = b \sin \beta + h \sin \varphi, \quad (2.98)$$

$$\tan \beta = \frac{b}{a} \tan \varphi. \quad (2.99)$$

La *fórmula cerrada* para la gravedad normal de la ecuación 2.86 nos permite calcular analíticamente la gravedad normal generada por el elipsoide de revolución sobre cualquier punto exterior al mismo.

Ecuación de Somigliana

Como caso particular, la gravedad normal en cualquier punto situado sobre la superficie del elipsoide de referencia ($h = 0$) se puede calcular de manera sencilla a través de la ecuación de Somigliana (Heiskanen y Moritz, 1967):

$$\gamma(\varphi, h = 0) = \gamma_e \frac{1 + k \sin^2 \varphi}{\sqrt{1 - e^2 \sin^2 \varphi}}; \quad (2.100)$$

donde

$$k = \frac{b}{a} \frac{\gamma_p}{\gamma_e} - 1; \quad (2.101)$$

e es la *primera excentricidad*

$$e = \sqrt{\frac{a^2 - b^2}{a^2}}; \quad (2.102)$$

y γ_e y γ_p son los valores de la gravedad normal en cualquier punto del ecuador y en los polos, respectivamente (ver Heiskanen y Moritz, 1967, p. 68–69, para más detalles).

2.3.2. Disturbio de gravedad

Con el objeto de poder identificar y caracterizar la distribución de densidades anómalas en el interior de la Tierra, es necesario remover la *gravedad normal* de la gravedad terrestre. Definiremos entonces al *vector del disturbio de gravedad* $\delta\mathbf{g}$ (*gravity disturbance vector* en inglés) como la diferencia entre el vector *gravedad* y el vector *gravedad normal* calculados en el punto \mathbf{p} (Fig. 2.6):

$$\delta\mathbf{g}(\mathbf{p}) = \mathbf{g}(\mathbf{p}) - \gamma(\mathbf{p}). \quad (2.103)$$

La mayoría de las metodologías que nos permiten realizar mediciones del campo gravitatorio terrestre consiguen estimar el valor de la componente del vector \mathbf{g} a lo largo de la línea de fuerza del potencial de gravedad W , la cual coincide con el módulo de \mathbf{g} . Medir la dirección del vector \mathbf{g} resulta un problema de mayor complejidad, dificultando el cálculo del *vector del disturbio de gravedad*. Es por ello que introducimos el concepto del *disturbio de gravedad* δg (*gravity disturbance* en inglés), definido como la diferencia entre el módulo del vector *gravedad* y el módulo del vector *gravedad normal* en el mismo punto \mathbf{p} (Heiskanen y Moritz, 1967; Hofmann-Wellenhof y Moritz, 2005; Oliveira Jr et al., 2018):

$$\delta g(\mathbf{p}) = g(\mathbf{p}) - \gamma(\mathbf{p}). \quad (2.104)$$

Vale notar que el *disturbio de gravedad* δg no coincide con el módulo del *vector disturbio de gravedad* $\delta\mathbf{g}$ (Barthelmes, 2013; Oliveira Jr et al., 2018; Sansò y Sideris, 2013). Sin embargo, se puede asumir que el *disturbio de gravedad* se aproxima lo suficiente al módulo del *vector del disturbio de gravedad* si consideramos que la desviación del vector $\mathbf{g}(\mathbf{p})$ de la recta normal al elipsoide en el punto \mathbf{p} es lo suficientemente pequeña.

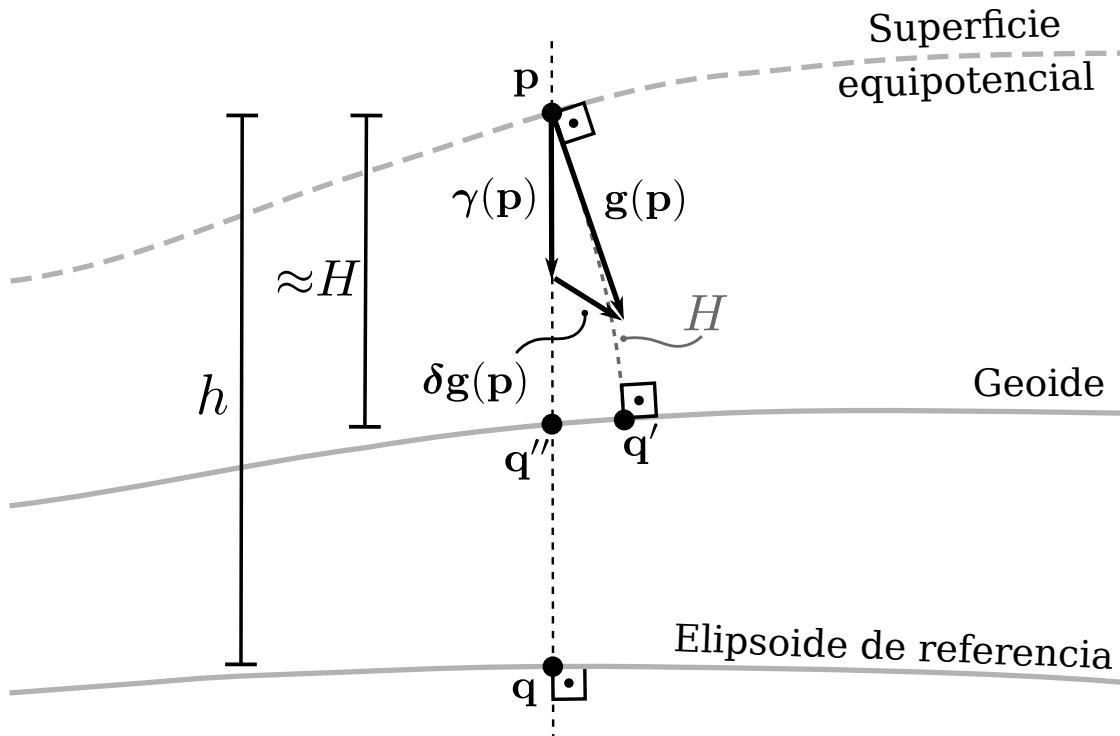


Figura 2.6: Elipsoide de referencia, geoide y vector del disturbio de gravedad. Se identifica un punto p situado a una altitud geométrica h indicada por el segmento de recta que une el punto q sobre el elipsoide de referencia. Se observa el vector de gravedad $\mathbf{g}(p)$ sobre el punto p normal a la superficie equipotencial del potencial de gravedad W que pasa por el mismo punto p . Se identifica además el vector de gravedad normal $\gamma(p)$ y el vector de disturbio de gravedad $\delta\mathbf{g}(p)$. La altitud ortométrica H se determina por la longitud del segmento de la línea de fuerza del potencial W que une el punto q' sobre el geoide con el punto p , aunque puede aproximarse por el segmento de recta que une el punto q'' y el punto p . Las curvaturas de las superficies equipotenciales han sido exageradas para mejorar la visualización. Esta figura es una versión modificada de Oliveira Jr et al. (2021a).

2.4. Fuentes equivalentes

Consideremos una distribución de masa arbitraria limitada a una región Ω con una densidad ρ y un punto de observación p exterior a la región Ω (Fig. 2.7). El potencial gravitatorio V que genera la distribución de masa en el punto p puede calcularse a través de la ecuación 2.10.

Consideremos además una superficie equipotencial arbitraria ∂S de forma tal que el punto p se encuentra fuera de la región S interior a ∂S (Fig. 2.7). Vamos a asumir que el potencial V toma un valor constante V_S en ∂S . Aplicando la

segunda identidad de Green ([Blakely, 1995](#), p. 23) podemos obtener:

$$\int_S \left[V(\mathbf{q}) \nabla^2 \left(\frac{1}{\|\mathbf{p} - \mathbf{q}\|} \right) - \frac{1}{\|\mathbf{p} - \mathbf{q}\|} \nabla^2 V(\mathbf{q}) \right] dv = \oint_{\partial S} \left[V(\mathbf{q}) \frac{\partial}{\partial n} \left(\frac{1}{\|\mathbf{p} - \mathbf{q}\|} \right) - \frac{1}{\|\mathbf{p} - \mathbf{q}\|} \frac{\partial V}{\partial n} \right] ds, \quad (2.105)$$

donde \mathbf{q} es la variable de integración en ambos miembros y n indica la dirección normal a la superficie ∂S .

El primer término de la integral volumétrica es nulo, ya que la inversa de $\|\mathbf{p} - \mathbf{q}\|$ es un campo armónico. Por otro lado, el valor de $V(\mathbf{q})$ en el primer término de la integral de superficie es constante, ya que el dominio de integración ∂S es una superficie equipotencial de V . Por lo tanto, la ecuación anterior puede expresarse de forma más sencilla como:

$$-\int_S \frac{\nabla^2 V(\mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|} dv = V_S \oint_{\partial S} \frac{\partial}{\partial n} \left(\frac{1}{\|\mathbf{p} - \mathbf{q}\|} \right) ds - \oint_{\partial S} \frac{1}{\|\mathbf{p} - \mathbf{q}\|} \frac{\partial V}{\partial n} ds. \quad (2.106)$$

Uno de los corolarios de la primera identidad de Green establece que la integral de la derivada normal de una función armónica sobre una superficie cerrada es nula ([Blakely, 1995](#), p. 20). Por ende podemos anular la primera integral de superficie de la ecuación anterior:

$$-\int_S \frac{\nabla^2 V(\mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|} dv = -\oint_{\partial S} \frac{1}{\|\mathbf{p} - \mathbf{q}\|} \frac{\partial V}{\partial n} ds, \quad (2.107)$$

Si aplicamos la ecuación de Poisson (ec. 2.24) y definimos una densidad superficial σ :

$$\sigma(\mathbf{q}) = \frac{-1}{4\pi G} \frac{\partial V}{\partial n}, \quad (2.108)$$

podemos expresar la ecuación anterior como:

$$G \int_S \frac{\rho(\mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|} dv = G \oint_{\partial S} \frac{\sigma(\mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|} ds. \quad (2.109)$$

Considerando que la densidad ρ es nula fuera de la región Ω , podemos simplificar el dominio de integración de la integral volumétrica. De esta manera, ambos términos equivalen al valor que asume el potencial V en el punto \mathbf{p} (ec. 2.10):

$$V(\mathbf{p}) = G \int_{\Omega} \frac{\rho(\mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|} dv = G \oint_{\partial S} \frac{\sigma(\mathbf{q})}{\|\mathbf{p} - \mathbf{q}\|} ds. \quad (2.110)$$

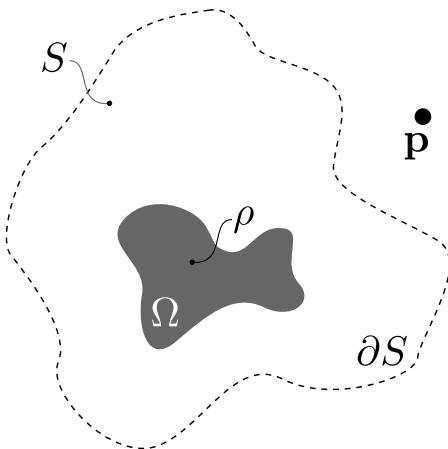


Figura 2.7: Distribución de masa tridimensional con densidad ρ dentro de una región Ω , junto con superficie equipotencial ∂S del potencial gravitatorio generado por la misma y un punto de observación p exterior a la región S interior a ∂S .

Esta relación se conoce como *fuentes equivalentes de Green* (*Green's equivalent layer* en inglés) y de la cual se deduce que el potencial que genera una distribución de densidad tridimensional es indistinguible del que genera una masa superficial (o una capa delgada) distribuida sobre una superficie equipotencial (Blakely, 1995, p. 62).

Esta propiedad de los campos gravitatorios nos permite desarrollar metodologías de interpolación de campos armónicos basados en el modelado de distribuciones de masas arbitrarias, sin necesidad de que guarden relación con las geometrías y densidades de los cuerpos que generan los campos observados. Además, las fuentes equivalentes de Green dejan en evidencia la no unicidad de las fuentes de los campos gravitatorios: existe una infinita cantidad de distribuciones de masas que pueden generar el mismo campo gravitatorio, o una aproximación suficiente para resultar indistinguible a efectos prácticos.

Capítulo 3

Teseroides de densidad variable

3.1. Resumen

Presentamos una nueva metodología para calcular los campos gravitatorios generados por teseroides (prismas esféricos) cuya densidad varía en profundidad según una función continua arbitraria. Esta metodología aproxima los campos gravitatorios mediante una [GLQ](#) junto con la aplicación de dos algoritmos de discretización que controlan automáticamente la precisión de la aproximación dividiendo adaptativamente el teseroide original en otros más pequeños. El primero es un algoritmo preexistente de discretización adaptativa bidimensional que reduce el error debido a la distancia entre el teseroide y el punto de cómputo. El segundo es un nuevo algoritmo de discretización basado en la densidad que reduce los errores debido a la variación de la densidad con la profundidad. La cantidad de subdivisiones que realiza cada algoritmo es indirectamente controlada por dos parámetros: el *ratio distancia-tamaño* y el *ratio delta*. Hemos obtenido soluciones analíticas de los campos gravitatorios generados por un cascarón esférico con densidades variables a lo largo de la dirección radial y los hemos comparado con los resultados del modelo numérico para densidades lineales, exponenciales y sinusoidales. Las densidades oscilantes fueron utilizadas con la única intención de someter al algoritmo a sus límites y no para emular un escenario real. Estas comparaciones nos han permitido obtener valores óptimos para los *ratio distancia-tamaño* y *delta* que garantizan una precisión de 0.1 % en relación con las soluciones analíticas. Los valores óptimos del *ratio distancia-tamaño* para el potencial gravitatorio y su gradiente son 1 y 2.5, respectivamente. La discretización basada en la densidad no es necesaria en los casos de densidad lineal, pero se requiere de un *ratio delta* de 0.1 para densidades exponenciales y la mayoría de las sinusoidales. Estos valores pueden ser extrapolados para cubrir los usos más comunes, siempre y cuando los perfiles de densidad sean más simples que funciones sinusoidales. Los *ratios* densidad-tamaño y *delta* pueden ser configurados por los usuarios o las usuarias para aumentar la precisión de los resultados a expensas de un aumento

en el tiempo de cómputo. Hemos aplicado esta nueva metodología, utilizando una densidad exponencial, para modelar la Cuenca Neuquina, una cuenca de antepaís en Argentina, con una profundidad máxima de 5000 m.

3.2. Introducción

La variación de la densidad de la litósfera con respecto a la profundidad ha sido estudiada por casi un siglo. A lo largo de este tiempo, varias relaciones entre la densidad y la profundidad han sido propuestas para diferentes tipos de rocas (por ejemplo [Maxant, 1980](#); [Rao et al., 1993, 1994](#); [Rao, 1986](#)). Además, densidades que varían con la profundidad han sido utilizadas en el modelado directo e inverso de datos gravitatorios, principalmente aplicados a cuencas sedimentarias ([Cordell, 1973](#); [Cowie y Karner, 1990](#); [Rao et al., 1993, 1994](#); [Rao, 1986](#); [Welford et al., 2010](#); [Zhang et al., 2001](#)). Estos modelos directos han sido desarrollados en coordenadas Cartesianas para cuerpos bidimensionales o tridimensionales, lo que limita su aplicación a escalas locales. La llegada de la gravimetría satelital ha proveído mediciones del campo gravitatorio terrestre con cobertura global, permitiendo el modelado e interpretación en escalas regionales o globales. Por esta razón, diseñar métodos de modelado directo que reproducen las anomalías de gravedad para esas escalas resulta muy importante.

Con el objetivo de tomar en consideración la curvatura de la Tierra, muchos modelos directos globales se definen en coordenadas esféricas geocéntricas (ver Capítulo 2). Un abordaje común es discretizar la Tierra en teseroídes (prismas esféricos) ([Anderson, 1976](#)), los cuales son definidos por el volumen que delimitan pares de latitud, longitud y radios (Fig. 2.4). Los campos gravitatorios generados por un teseroide en cualquier punto exterior vienen dados por integrales de volumen que deben ser aproximadas numéricamente. La literatura ofrece dos abordajes principales: uno involucra expansiones en series de Taylor ([Grombein et al., 2013](#); [Heck y Seitz, 2006](#)), mientras que el otro hace uso de la [GLQ](#). La expansión en series de Taylor no resulta adecuada para desarrollar un algoritmo que soporte densidades variables según funciones arbitrarias: los diferentes términos de las series deberían obtenerse para cada una de las posibles funciones de densidad. Por el contrario, una función de densidad arbitraria puede incluirse dentro de la [GLQ](#) sin necesidad de modificar el método de integración. Es por esta razón que haremos foco en los métodos basados en la [GLQ](#) de aquí en adelante.

La principal dificultad que presenta la integración por [GLQ](#) es la pérdida de precisión que ocurre cuando el punto de cómputo se acerca al teseroide ([Ku, 1977](#)). [Uieda et al. \(2016\)](#) desarrollaron un método a partir del algoritmo de discretización adaptativa tridimensional de [Li et al. \(2011\)](#) para obtener de manera automática campos gravitatorios de teseroídes con densidad uniforme con una precisión de 0.1 %. El algoritmo divide recursivamente un teseroide en otros

más pequeños cuando un determinado umbral es excedido, esto es cuando la distancia normalizada al punto de cómputo es mayor que un parámetro denominado *ratio distancia-tamaño* (D). [Uieda et al. \(2016\)](#) obtuvieron además valores estándar de D para el cómputo del potencial gravitatorio, las componentes de su gradiente y del tensor del gradiente comparando los resultados de la integración numérica con los campos generados por un cascarón esférico.

Dos publicaciones recientes presentan abordajes alternativos para calcular los campos gravitatorios de teseroides homogéneos e incorporan metodologías para el caso de teseroides con densidades variables en profundidad. [Fukushima \(2018\)](#) ha resuelto analíticamente la integral correspondiente al potencial gravitatorio en la dirección radial, obteniendo una integral de superficie, la cual es numéricamente resuelta dividiendo condicionalmente el teseroide y aplicando la cuadratura exponencial doble. El gradiente del potencial y las componentes del tensor del gradiente son calculadas posteriormente por diferencias finitas. [Fukushima \(2018\)](#) además generaliza el método para teseroides con una densidad que varía con el radio según una función polinomial de grado arbitrario. [Lin y Denker \(2019\)](#) han comparado las diferentes metodologías de integración y discretización para teseroides homogéneos. A partir de este análisis han desarrollado un método combinado: para puntos de cómputo cercanos al teseroide, utilizan una integración [GLQ](#) junto con una discretización adaptativa basada en [Uieda et al. \(2016\)](#), pero solo aplicada a las dimensiones horizontales. Si el punto de cómputo se encuentra más allá de una cierta distancia de truncamiento, aplican una aproximación en serie de Taylor de segundo orden, junto con la subdivisión desarrollada por [Grombein et al. \(2013\)](#). [Lin y Denker \(2019\)](#) además presentan una variación de su método combinado que permite calcular los campos gravitatorios generados por teseroides con densidades lineales en la dirección radial.

Los desarrollos de [Lin y Denker \(2019\)](#) y [Fukushima \(2018\)](#) se limitan a funciones de densidad polinomiales. Si bien la mayoría de las funciones suaves pueden aproximarse por funciones lineales de a pasos, la elección del intervalo de discretización no es ni directa ni automática para el caso general. Aunque existen muchos algoritmos de aproximación lineal de a pasos que automatizan este proceso ([Ahmadi et al., 2013; Imamoto y Tang, 2008; Ketkov, 1969; Vandeewalle, 1975](#)), estos requieren un número fijo de intervalos de discretización o bien están diseñados para ser utilizados solo en funciones convexas. El uso de estos algoritmos limitaría el dominio de funciones de densidades que pueden ser asignadas a teseroides y no garantizarían un proceso completamente automático. Además, es bien conocido que el uso de polinomios de alto grado para aproximar una función altamente variable produce resultados inestables cuando se extrae más allá del dominio de datos. Estos obstáculos podrían hacer que aproximar densidades mediante funciones lineales de a pasos o por polinomios de alto grado no resulte adecuado para inversiones de gravedad no lineales (e.g. [Uieda y Barbosa, 2017](#)), especialmente si la función densidad es altamente variable en profundidad.

Presentamos un nuevo algoritmo para el cálculo del potencial gravitatorio y su gradiente generado por un teseroide con una función continua de densidad que varía según la coordenada radial. Esta metodología está basada en una integración [GLQ](#) tridimensional, una versión bidimensional del algoritmo de discretización adaptativa de [Uieda et al. \(2016\)](#) (de acuerdo con [Lin y Denker, 2019](#)), y un nuevo algoritmo de discretización basado en la densidad. Para garantizar la precisión de la aproximación numérica, hemos determinado empíricamente valores óptimos para los parámetros que controlan las discretizaciones, comparando los resultados numéricos con las soluciones analíticas de cascarones esféricos. Finalmente, hemos aplicado la nueva metodología para modelar la Cuenca Neuquina (Argentina) utilizando teseroides con densidades lineales y exponenciales en profundidad.

3.3. Metodología

Consideremos un teseroide en un sistema de coordenadas esféricas geocéntricas definido por pares de latitudes (ϕ_1, ϕ_2) , longitudes (λ_1, λ_2) y radios (r_1, r_2) . Definimos un punto de computo externo \mathbf{p} localizado en un radio r , una latitud ϕ y una longitud λ . [Grombein et al. \(2013\)](#) proveen formulaciones eficientes para las integrales de volumen del potencial gravitatorio generado por un teseroide de densidad homogénea, junto con las componentes de su gradiente (ecs. [2.58](#), [2.60](#), [2.61](#), [2.62](#)). Aquí vamos a considerar los casos en los cuales el teseroide posee una densidad variable con respecto a la coordenada radial r según una función continua arbitraria $\rho(r)$. Por lo tanto, las integrales del potencial gravitatorio y las componentes de su gradiente se ven ligeramente modificadas:

$$V(r, \phi, \lambda) = G \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \frac{\rho(r')}{\ell} \kappa dr' d\phi' d\lambda', \quad (3.1)$$

$$g_\alpha(r, \phi, \lambda) = G \int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \rho(r') \frac{\Delta\alpha}{\ell^3} \kappa dr' d\phi' d\lambda', \quad (3.2)$$

donde $\alpha \in \{x, y, z\}$, y $\Delta\alpha$ vienen dados por las ecuaciones [2.39](#), [2.40](#) y [2.41](#).

3.3.1. Integración por Cuadratura de Gauss-Legendre

Aplicando una [GLQ](#) de orden N podemos aproximar a cada integral definida por las ecuaciones [3.1](#) y [3.2](#) por una suma ponderada de los núcleos de integración evaluados en las raíces de un polinomio de orden N ([Hildebrand, 1987](#), p. 390), conocidos como los nodos de la cuadratura. A diferencia de los teseroides con densidad constante, la función de densidad $\rho(r)$ debe ser incluida

dentro del integrando y evaluada en los nodos de la cuadratura:

$$\int_{\lambda_1}^{\lambda_2} \int_{\phi_1}^{\phi_2} \int_{r_1}^{r_2} \rho(r') f(r', \phi', \lambda') dr' d\phi' d\lambda' \approx A \sum_{i=1}^{N^r} \sum_{j=1}^{N^\phi} \sum_{k=1}^{N^\lambda} W_i^r W_j^\phi W_k^\lambda \rho(r_i) f(r_i, \phi_j, \lambda_k), \quad (3.3)$$

donde A es una constante definida en la ecuación 2.72, $f(r', \phi', \lambda')$ es el núcleo correspondiente a un teseroide con densidad homogénea (Grombein et al., 2013), (r_i, ϕ_j, λ_k) son las coordenadas de los nodos de la cuadratura, N^r, N^ϕ, N^λ son los órdenes de la cuadratura y $W_i^r, W_j^\phi, W_k^\lambda$ son los pesos ponderados en la dirección radial, latitudinal y longitudinal, respectivamente. Vale la pena notar que aplicar la GLQ es equivalente a aproximar el teseroide por $N^r \times N^\phi \times N^\lambda$ masas puntuales localizadas en los nodos de la cuadratura (Asgharzadeh et al., 2007; Ku, 1977).

3.3.2. Discretización adaptativa bidimensional

Ku (1977) dio cuenta de que la integración por GLQ se vuelve menos precisa a medida que el punto de cómputo se acerca al teseroide. Una manera de mitigar este comportamiento sería aumentar el orden de la cuadratura. Al hacer esto, incrementaríamos uniformemente la cantidad de masas puntuales utilizadas para aproximar el teseroide. Sin embargo, solo es necesario incrementar la concentración de masas puntuales en cercanías del punto de cómputo (Uieda et al., 2016). Alternativamente, Li et al. (2011) han propuesto un algoritmo de discretización adaptativa que mantiene fijo el orden de la cuadratura y divide el teseroide según una relación entre la distancia al punto de cómputo y las dimensiones del mismo. Este algoritmo involucra un cómputo más eficiente, ya que produce un aumento en la concentración de masas puntuales solo en las regiones donde es necesario. Uieda et al. (2016) han desarrollado una versión modificada de este algoritmo junto con una implementación computacional eficiente. Ambas versiones del algoritmo subdividen el teseroide en las direcciones latitudinal, longitudinal y radial, por ende podemos definirlos como *algoritmos de discretización adaptativa tridimensionales*. Por otro lado, Lin y Denker (2019) han propuesto un *algoritmo de discretización adaptativa bidimensional* que subdivide solo en las direcciones latitudinal y longitudinal. Remover una dimensión de la discretización produce un cómputo más eficiente ya que reduce la cantidad de subdivisiones en el modelo, aunque manteniendo una precisión aceptable (Lin y Denker, 2019).

A lo largo de este Capítulo nos guiaremos por los resultados de Lin y Denker (2019) y haremos uso de una versión bidimensional del algoritmo de discretización adaptativa de Uieda et al. (2016). Lo que sigue a continuación es un resumen del algoritmo. El lector o la lectora puede referirse a Uieda et al. (2016) para una descripción más detallada.

Paso 1: Corroboramos que el teseroide satisface la desigualdad

$$\frac{d}{L_i} \geq D, \quad (3.4)$$

para sus dimensiones longitudinales y latitudinales L_i ($i \in \{\lambda, \phi\}$), donde D es un escalar positivo denominado *ratio de distancia-tamaño*, d es la distancia entre el punto de cómputo y el centro geométrico del teseroide:

$$d = \left[r^2 + r_t^2 - 2rr_t \cos \psi_t \right]^{\frac{1}{2}}, \quad (3.5)$$

$$\cos \psi_t = \sin \phi \sin \phi_t + \cos \phi \cos \phi_t \cos(\lambda - \lambda_t), \quad (3.6)$$

$$r_t = \frac{r_2 + r_1}{2}, \quad \phi_t = \frac{\phi_2 + \phi_1}{2}, \quad \lambda_t = \frac{\lambda_2 + \lambda_1}{2}, \quad (3.7)$$

y las dimensiones del teseroide se definen como:

$$L_\lambda = r_2 \arccos(\sin^2 \phi_t + \cos^2 \phi_t \cos(\lambda_2 - \lambda_1)), \quad (3.8)$$

y

$$L_\phi = r_2 \arccos(\sin \phi_2 \sin \phi_1 + \cos \phi_2 \cos \phi_1). \quad (3.9)$$

Paso 2: Si todas de las dimensiones del teseroide satisfacen la desigualdad 3.4, entonces calculamos su efecto gravitatorio usando una GLQ de segundo orden (ec. 3.3). Agregamos el efecto a un total acumulado.

Paso 3: Si la desigualdad 3.4 no es satisfecha por una o ambas dimensiones (longitudinal o latitudinal), dividimos el teseroide por la mitad a lo largo de esa dimensión o dimensiones. Repetimos los pasos 1 a 3 para todos los teseroides más pequeños hasta que ninguno de ellos viole la desigualdad 3.4.

Último paso: Al finalizar el algoritmo, una GLQ de segundo orden habrá sido aplicada a todos los teseroides pequeños que satisfacen la desigualdad 3.4, y por lo tanto el total acumulado del paso 2 será equivalente al efecto gravitatorio del teseroide original.

El *ratio* densidad-tamaño D determina cuántas veces el teseroide será dividido. Por lo tanto, su valor regula tanto la precisión del algoritmo como su tiempo de cómputo. Un valor óptimo de D no puede ser calculado directamente a partir del nivel de precisión deseado. En cambio, es posible determinarlo empíricamente comparando los resultados numéricos con la solución analítica para un cascarón esférico. Uieda et al. (2016) han utilizado un cascarón de densidad homogénea para determinar los valores óptimos de D para su algoritmo aplicado a teseroides de densidad constante. Aquí repetiremos dicho experimento numérico utilizando las expresiones analíticas de los campos gravitatorios generados por cascarones con densidad variable según funciones continuas de r . Estos experimentos también corroborarán si los mismos valores de D determinados por Uieda et al. (2016) pueden ser utilizados en conjunto con un algoritmo de discretización adaptativa bidimensional.

3.3.3. Algoritmo de discretización basado en densidad

La integración numérica considerando una función continua de densidad introduce un nuevo tipo de problema: el error de integración que surge de utilizar solo unos pocos nodos de cuadratura para evaluar la variación de la densidad del teseroide. El algoritmo de discretización adaptativa tridimensional puede ayudar a reducir este tipo de error agregando más masas puntuales en la dirección radial. Sin embargo, este algoritmo no considera la función de densidad en sí misma a la hora de decidir cómo subdividir al teseroide, y por lo tanto no es el método más adecuado para realizar esta tarea.

En este trabajo hemos desarrollado un algoritmo de discretización complementario que subdivide al teseroide en la dirección radial tomando en consideración las variaciones de la función densidad. Este *algoritmo de discretización basado en la densidad* se aplica antes del *algoritmo de discretización adaptativa bidimensional* descripto en la Sección anterior. En resumen, el algoritmo divide al teseroide a lo largo de la dirección radial en las profundidades a las cuales ocurre la *máxima variación de densidad*.

Consideremos un teseroide, que llamaremos *original*, con una función de densidad dada por $\rho(r')$. Antes de comenzar a aplicar el algoritmo de discretización basado en la densidad, normalizamos la función $\rho(r')$ al intervalo $[0, 1]$ de la siguiente manera:

$$\rho_n(r') = \frac{\rho(r') - \rho_{\min}}{\rho_{\max} - \rho_{\min}}, \quad (3.10)$$

donde ρ_{\min} y ρ_{\max} son el mínimo y el máximo valor de densidad dentro de los límites de teseroide. Hacemos hincapié en que esta función densidad normalizada no se verá modificada a lo largo del algoritmo. En caso de que la función densidad sea constante, los mínimos y máximos serán iguales y por ende el algoritmo de discretización basado en densidad no será aplicado.

El algoritmo puede comprenderse a través de los siguientes pasos (Fig. 3.1):

Paso 1: Definimos una *línea recta* $\rho_l(r')$ que toma los mismos valores que la función densidad normalizada $\rho_n(r')$ en los extremos del teseroide (r_1 y r_2):

$$\rho_l(r') = \frac{\rho_n(r_2) - \rho_n(r_1)}{r_2 - r_1}(r' - r_1) + \rho_n(r_1). \quad (3.11)$$

Paso 2: Evaluamos la función de densidad normalizada y la *línea recta* en un rango de N radios entre r_1 y r_2 . Hemos optado por un $N = 101$, pero el valor específico de N no es crítico para el funcionamiento del algoritmo.

Paso 3: Calculamos la diferencia absoluta entre los valores de la *línea recta* y la función densidad normalizada:

$$\Delta\rho(r') = |\rho_n(r') - \rho_l(r')|. \quad (3.12)$$

Paso 4: Si la siguiente desigualdad se satisface, el teseroide no será subdividido:

$$\max\{\Delta\rho(r')\} \frac{L_r}{L_r^{\text{orig}}} \leq \delta, \quad (3.13)$$

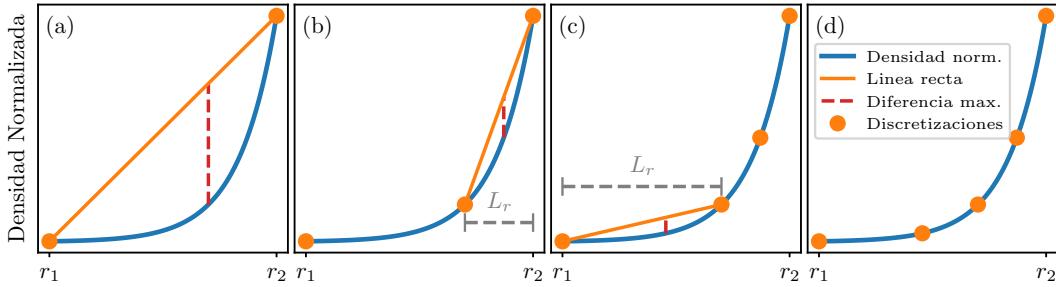


Figura 3.1: Ejemplo de la aplicación del algoritmo de discretización basado en densidad a una función densidad no lineal. (a) La función densidad normalizada $\rho_n(r')$ (azul), los límites actuales del teseroide (puntos naranja), y la *línea recta* $\rho_l(r')$ (línea naranja). Las líneas de a trazos rojas representan la máxima diferencia de densidad $\Delta\rho(r')$ a la cual el teseroide será subdividido (asumiendo que la desigualdad 3.13 no se satisface). (b) Segunda iteración del algoritmo con una nueva *línea recta* y máxima diferencia de densidad. El teseroide será dividido a la profundidad indicada por la linea roja de a trazos. (c) Tercera iteración del algoritmo. (d) Salida final del algoritmo de discretización basado en densidad, asumiendo que los cuatro teseroides nuevos satisfacen la desigualdad 3.13.

donde L_r es la dimensión radial del teseroide en consideración (no confundir con el teseroide *original*),

$$L_r = r_2 - r_1, \quad (3.14)$$

L_r^{orig} es la dimensión radial del teseroide *original*, y δ es una constante positiva que denominaremos *ratio delta*.

Paso 5: Si la desigualdad 3.13 no se satisface, entonces dividimos el teseroide en dos a una profundidad dada por el radio r_{\max} a la cual la máxima diferencia absoluta (ec. 3.12) tiene lugar. Repetimos los pasos 1 a 5 para cada teseroide pequeño producido por este paso.

Paso final: Una vez que todos los pequeños teseroides satisfacen la desigualdad 3.13, cada uno es sometido al algoritmo de discretización adaptativa bidimensional con el objeto de calcular sus efectos gravitatorios.

En la primer iteración, la relación L_r/L_r^{orig} es igual a 1, ya que el teseroide que se considera para subdividir corresponde al *original*. En las subsiguientes iteraciones, esta relación será progresivamente menor que 1 a medida que los teseroides sean cada vez más pequeños. La intención de incluir este factor en la desigualdad 3.13 es limitar la cantidad de divisiones a un número que reduzca significativamente el error numérico: dividir un teseroide grande con un $\max\{\Delta\rho(r')\}$ bajo mejoraría la precisión de la integración en mayor medida que dividiendo un teseroide pequeño con un $\max\{\Delta\rho(r')\}$ mayor.

A mayor valor de δ , se llevarán a cabo menor cantidad de divisiones, y viceversa. Por ende, el valor de δ controla cuántas veces los teseroides serán divi-

didos basados en la función densidad, e indirectamente, determina la precisión y el tiempo de cómputo de la integración numérica. Surge entonces la necesidad de determinar un valor máximo de δ que garantice una precisión aceptable mientras minimiza el tiempo de cómputo.

3.3.4. Resumen del algoritmo

En resumen, dado un teseroide con densidad variable en profundidad según una función continua arbitraria, proponemos seguir los siguientes pasos para calcular numéricamente sus campos gravitatorios en cualquier punto exterior:

Paso 1: Aplicar el *algoritmo de discretización basado en densidad* para subdividir al teseroide a lo largo de la dirección radial, produciendo un conjunto de teseroideos con las mismas dimensiones longitudinales y latitudinales que el original, pero con diferentes límites radiales.

Paso 2: Aplicar el *algoritmo de discretización adaptativa bidimensional* a cada teseroide obtenido en el paso anterior. Si es necesario, el algoritmo subdividirá cada teseroide en las direcciones longitudinal y latitudinal, generando un conjunto de teseroideos más pequeños.

Paso 3: Aplicar una **GLQ** de segundo orden para calcular numéricamente los campos gravitatorios (ec. 3.3) generados por cada teseroide obtenido en el paso anterior. La integración numérica incluye la función densidad y puede ser aplicada sin modificaciones a cualquier función continua. La suma de todos estos resultados corresponde al campo gravitatorio generado por el teseroide original.

3.3.5. Implementación por Software

Hemos implementado los algoritmos descriptos en las secciones anteriores mediante el uso del lenguaje de programación Python. El software está basado en la implementación de [Uieda et al. \(2016\)](#) incluida en la librería Fatiando a Terra v0.5 ([Uieda et al., 2013](#)). Los pasos que involucran un mayor tiempo de cómputo han sido escritos en lenguaje Cython para alcanzar un mejor desempeño. Aprovechamos la naturaleza dinámica del lenguaje Python para admitir como parámetro de entrada funciones de densidades definidas por el usuario o la usuaria. Por ende, nuestro código puede ser utilizado con funciones lineales, exponenciales, polinomiales, sinusoidales, splines cúbicas o cualquier otra función de densidad continua sin necesidad de realizar modificaciones a su programación. Esta implementación se encuentra disponible libremente bajo la licencia de código abierto BSD 3-clause. Todo el código fuente, los scripts de Python, los datos y resultados se encuentran disponibles a través del siguiente repositorio doi.org/10.6084/m9.figshare.8239622 ([Soler et al., 2019b](#)) o github.com/pinga-lab/tesseroid-variable-density. El repositorio también contiene instrucciones para replicar los resultados presentados en este Capítulo.

3.4. Determinación de los *ratios distancia-tamaño* y *delta*

El *ratio distancia-tamaño* D de la discretización adaptativa y el *ratio delta* δ de la discretización basada en la densidad determinan cuántas veces cada teseroide será dividido y por ende controlan indirectamente el error numérico de las integraciones. Debemos determinar valores óptimos para D y δ si deseamos asegurar tanto una precisión numérica aceptable así como una eficiencia computacional de los algoritmos.

[Uieda et al. \(2016\)](#) compararon los resultados de las integraciones numéricas de teseroides homogéneos con las soluciones analíticas de un cascarón esférico ([Grombein et al., 2013; Mikuška et al., 2006](#)) con el objetivo de obtener valores predeterminados para el *ratio distancia-tamaño* D . Seguiremos esta idea, pero en nuestro caso el cascarón esférico debe tener el mismo perfil de densidad que nuestro modelo de teseroides. [Lin y Denker \(2019\)](#) obtuvieron la solución analítica del potencial gravitatorio generado por un cascarón esférico con densidad lineal en la coordenada radial. Aplicando el Teorema del Cascarón de Newton ([Binney y Tremaine, 2008; Chandrasekhar, 1995](#)), podemos derivar expresiones para el potencial gravitatorio de un cascarón esférico con densidades lineales, exponenciales o sinusoidales (ver Apéndice 3.A).

Con el objetivo de comparar los resultados numéricos con las soluciones analíticas debemos construir modelos de un cascarón esférico a partir de teseroides. Dividimos entonces el cascarón esférico a lo largo de las direcciones longitudinales y latitudinales obteniendo un modelo de cascarón conformado por $6 \times 12 = 72$ teseroides de un tamaño de $30^\circ \times 30^\circ$. Hemos definido varios modelos de cascarones con diferentes espesores (Tabla 3.1) para poder evaluar la discretización basada en densidad en diferentes escenarios. Los valores de espesor fueron elegidos para cubrir un amplio rango de posibles aplicaciones: desde modelos topográficos a escalas litosféricas. Dado que la cantidad de subdivisiones en la discretización adaptativa será proporcional al tamaño de los teseroides (ec. 3.4), algunas de estas configuraciones representan los peores casos. La mayoría de las aplicaciones prácticas usarían teseroides más pequeños que $30^\circ \times 30^\circ \times 1000$ km.

Las diferencias entre la solución analítica y los resultados numéricos pueden ser calculados en un único punto de cómputo debido a la simetría rotacional del cascarón esférico. Sin embargo, los resultados numéricos dependen de la posición relativa entre el punto de cómputo y el teseroide ([Asgharzadeh et al., 2007; Ku, 1977; Uieda et al., 2016](#)). Tendremos en cuenta este fenómeno calculando dichas diferencias sobre grillas regulares y almacenando únicamente la diferencia máxima absoluta. Estos cálculos serán realizados sobre cuatro grillas diferentes (Tabla 3.2): una grilla local sobre el polo, una grilla local en el ecuador, una grilla global a altitud cero, y una grilla global a una altitud de 260km sobre la superficie del cascarón (representando la altitud nominal del satélite GOCE).

Espesor	Tamaño de cada teseroide	Cantidad de teseroides
0.1 km	$30^\circ \times 30^\circ$	$6 \times 12 = 72$
1 km	$30^\circ \times 30^\circ$	$6 \times 12 = 72$
10 km	$30^\circ \times 30^\circ$	$6 \times 12 = 72$
100 km	$30^\circ \times 30^\circ$	$6 \times 12 = 72$
1000 km	$30^\circ \times 30^\circ$	$6 \times 12 = 72$

Tabla 3.1: Descripción de los modelos de teseroides utilizados para construir cascarones esféricos y caracterizar la precisión de las integraciones numéricas. El radio exterior (R_2) de cada modelo es igual al radio medio de la Tierra (6378.137 km), mientras que el radio interior (R_1) es determinado por el espesor del cascarón. La cantidad total y las dimensiones horizontales de los teseroides en cada modelo de cascarón están detalladas según dimensiones latitudinales y longitudinales, respectivamente.

Nombre	Espaciado de la grilla	Región (grados)	Altitud (km)
Polo	0.1°	0E / 1E / 89N / 90N	0
Ecuador	0.1°	0E / 1E / 0N / 1N	0
Global	10°	180W / 180E / 90S / 90N	0
Satélite	10°	180W / 180E / 90S / 90N	260

Tabla 3.2: Descripción de las grillas de putos de cómputo utilizadas para caracterizar la precisión de las integraciones numéricas. La altitud de las grillas es definida por encima del radio medio de la Tierra.

Estas grillas cubren un amplio espectro de escenarios y asegurarán una precisión aceptable para cada uno de ellos.

Las comparaciones entre las soluciones analíticas y los resultados numéricos serán llevadas a cabo utilizando funciones de densidades lineales, exponenciales y sinusoidales. La densidad sinusoidal es incluida con el objetivo de evaluar la aproximación numérica en sus límites de desempeño. Repetimos estas comparaciones para cada combinación de modelos de teseroides en la Tabla 3.1 y grillas de la Tabla 3.2. A partir de estos resultados generalizaremos valores óptimos para D y δ que garantizan errores numéricos menores a 0.1 % en comparación con las soluciones analíticas del cascarón esférico en la mayoría de los casos.

3.4.1. Densidad Lineal

El potencial gravitatorio y la componente radial del gradiente que genera un cascarón esférico con una función de densidad lineal dada por

$$\rho(r') = ar' + b, \quad (3.15)$$

posee soluciones analíticas dadas por las ecuaciones del apéndice 3.A: 3.27 y 3.25. Los valores de los coeficientes a y b pueden ser elegidos de forma tal que la densidad asuma valores iguales a $\rho_{\text{in}} = 3300 \text{ kg/m}^3$ y $\rho_{\text{out}} = 2670 \text{ kg/m}^3$ en los radios interior (R_{in}) y exterior (R_{out}) del cascarón, respectivamente:

$$a = \frac{\rho_{\text{out}} - \rho_{\text{in}}}{R_{\text{out}} - R_{\text{in}}}, \quad (3.16)$$

$$b = \rho_{\text{out}} - \frac{\rho_{\text{out}} - \rho_{\text{in}}}{R_{\text{out}} - R_{\text{in}}} R_{\text{out}}. \quad (3.17)$$

Las diferencias absolutas definidas en la ecuación 3.12 son siempre cero para cualquier tipo de densidad lineal. Como resultado, la desigualdad 3.13 será siempre satisfecha y no será necesario subdividir el teseroide en la dirección radial durante la discretización basada en densidad. Por lo tanto, solo la discretización adaptativa bidimensional es el único mecanismo que controla la precisión de la integración numérica para el caso de densidad lineal. Por esta razón ignoraremos los valores de δ y solo determinaremos el mínimo valor de D necesario para garantizar una precisión aceptable.

Calculamos el potencial gravitatorio (V) y su derivada vertical (g_z) para cada modelo de cascarón esférico definido en la Tabla 3.1 en cada grilla de cómputo de la Tabla 3.2. Las derivadas horizontales del potencial son iguales a cero fuera del cascarón debido a la simetría rotacional, y por ende son omitidos del análisis. Estos cálculos son repetidos para valores de *ratio* distancia-tamaño D desde 0.5 a 5 con incrementos de 0.5. Luego calculamos la diferencia absoluta entre los resultados numéricos y las soluciones analíticas para el cascarón. La Figura 3.2 muestra la máxima diferencia absoluta para cada modelo de cascarón y grilla de cómputo como función de D . Las diferencias se muestran relativas a la solución analítica para cada cascarón. Finalmente, seleccionamos el valor óptimo de D como el menor valor para el cual la diferencia es menor a 0.1 %.

A partir de la Figura 3.2 podemos observar que los errores relativos para el potencial V y para su derivada vertical g_z caen por debajo del umbral de 0.1 % para valores de $D = 1$ y $D = 2.5$, respectivamente. Notablemente, un valor de $D = 2$ sería suficiente para g_z en el caso de una grilla a altitud satelital. Para el resto de las configuraciones, dichos valores son consistentes e independientes del espesor de cascarón o de la ubicación geográfica.

3.4.2. Densidad exponencial

Para una función de densidad exponencial, la discretización basada en densidad será aplicada con anterioridad al algoritmo de discretización adaptativa.

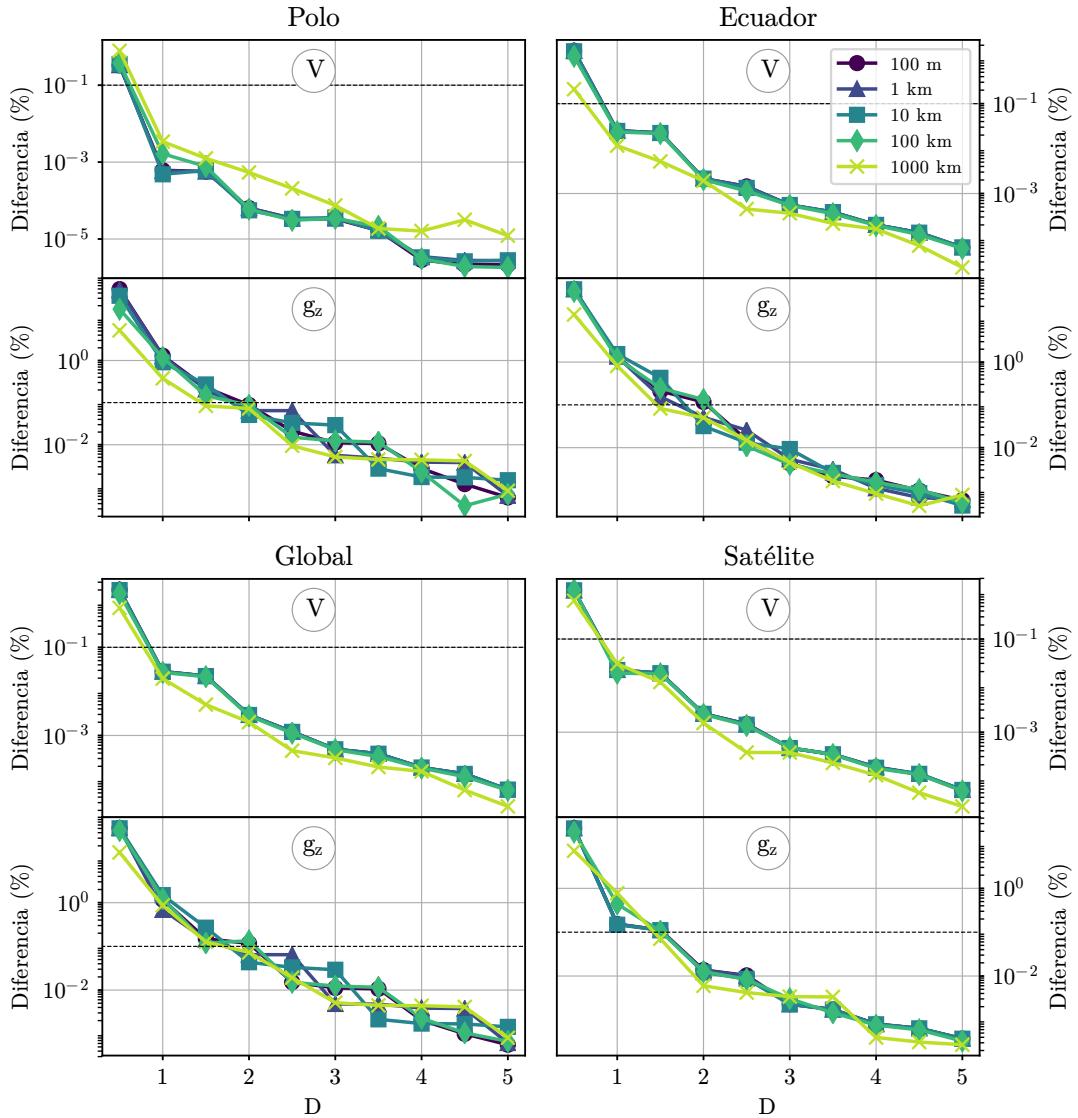


Figura 3.2: Diferencias entre los campos gravitatorios generados por cada modelo de cascarón hecho con teseroides y sus respectivas soluciones analíticas en función del *ratio* distancia-tamaño D . Cada modelo posee una densidad lineal (ec. 3.15). Los cálculos fueron realizados sobre las cuatro grillas descriptas en la Tabla 3.2. Cada curva representa la diferencia máxima absoluta entre los resultados numéricos y las solución analítica para un dado modelo de cascarón. Debido a la linealidad de la función densidad, el algoritmo de discretización basado en densidad no es aplicado. Las diferencias están reportadas como un porcentaje de las soluciones analíticas. Las líneas horizontales de a trazo y color negro representan el umbral de precisión de 0.1 %.

Esto significa que los valores óptimos para el *ratio* distancia-tamaño D y el *ratio delta* δ (ec. 3.13) deben ser determinados simultáneamente. Para ello llevamos a cabo un análisis de error similar al que aplicamos en el caso de densidad lineal. Ahora el cascarón esférico poseerá una densidad según una función exponencial que asume los valores de $\rho_{\text{out}} = 2670 \text{ kg/m}^3$ y $\rho_{\text{in}} = 3300 \text{ kg/m}^3$ en las superficies externas e internas, respectivamente, y que se define de la siguiente manera:

$$\rho(r') = Ae^{-b\frac{r'-R_1}{R_2-R_1}} + C, \quad (3.18)$$

donde

$$A = \frac{\rho_{\text{in}} - \rho_{\text{out}}}{1 - e^{-b}}, \quad (3.19)$$

$$C = \rho_{\text{in}} - A, \quad (3.20)$$

y b es una constante adimensional que determina el grado de variabilidad de la función. Un valor mayor de b aumenta la máxima pendiente de la función densidad (Fig. 3.3).

Exploración del espacio $D-\delta$

Nuestro objetivo es hallar una combinación de valores para D y δ que produzcan un error numérico menor al umbral de 0.1 % y simultáneamente minimicen el tiempo de cómputo. Para ello hacemos uso del método de búsqueda en grilla (*grid search* en inglés) y calculamos el error numérico para cada par (D, δ) perteneciente a una grilla en el espacio $D-\delta$ (Fig. 3.4). Con el objeto de obtener un óptimo desempeño del algoritmo, buscamos el par (D, δ) que minimice el número de subdivisiones de teseroides, manteniendo el error numérico por debajo del umbral de 0.1 %. Este requisito se traduce en minimizar el valor de D y maximizar el valor de δ .

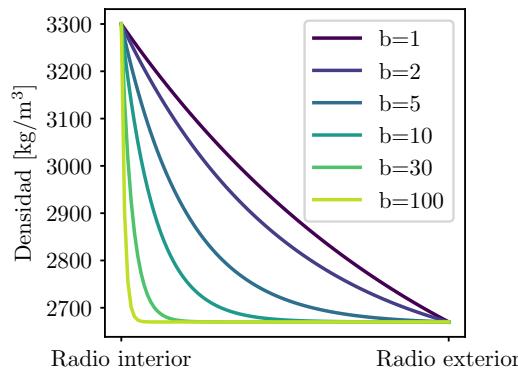


Figura 3.3: Funciones de densidad exponencial asignadas a los modelos de cascarón esférico para la determinación del *ratio* δ . Cada función densidad corresponde a diferentes valores de b en la ecuación 3.18.

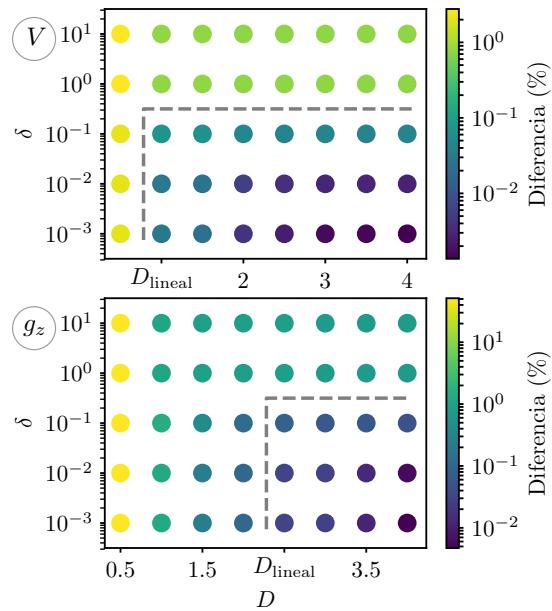


Figura 3.4: Exploración del error numérico en el espacio D - δ . Los valores porcentuales de diferencia fueron obtenidos a partir de la comparación entre las soluciones analíticas y las aproximaciones numéricas de los campos gravitatorios (V y g_z) generados por un cascarón esférico con una densidad exponencial (ec. 3.18) con $b = 30$. Estas comparaciones fueron realizadas sobre la grilla de cómputo *Global* (Tabla 3.2), con los modelos de cascarón esféricos detallados en la Tabla 3.1. Los valores de diferencias porcentuales fueron obtenidos como la máxima diferencia de entre todos los modelos de cascarón. Los puntos interiores a la región delimitada por las líneas de a trazos son los que presentan un error por debajo del umbral de 0.1 %. Los valores de D obtenidos para el caso de densidad lineal para cada campo gravitatorio se encuentran identificados con D_{lineal} .

Dado que la búsqueda en grilla es un proceso computacionalmente costoso, limitamos el análisis a un valor alto de $b = 30$ (Fig.3.3) y la grilla de cómputo *Global* definida en la Tabla 3.2. Calculamos la diferencia relativa entre el resultado numérico y la solución analítica del potencial gravitatorio (V) y su derivada vertical (g_z) para cada modelo de cascarón definido en la Tabla 3.1. La Figura 3.4 muestra la máxima diferencia obtenida para todos los modelos de cascarón. Las líneas de a trazos representan un contorno correspondiente al 0.1 % de error relativo, por ende los puntos interiores a ella son los que caen debajo de dicho umbral. Además, la figura destaca los valores de D obtenidos en la Sección anterior para el caso de densidad lineal (D_{lineal}).

Los valores más pequeños de D que se encuentran debajo del umbral de 0.1 % coinciden con los valores D_{lineal} tanto para V como para g_z . Estos resultados indican que los valores de D_{lineal} obtenidos pueden ser extrapolados a

los casos de densidades no lineales. Esto no es una sorpresa, ya que la discretización adaptativa bidimensional y la discretización basada en densidad son independientes una de la otra: la primera divide al teseroide en las direcciones horizontales, mientras que la otra lo hace solo en la dirección radial. Por ende, es de esperar que el valor óptimo de δ sea independiente del valor óptimo de D . Dado que la búsqueda en grilla fue limitada a una grilla de cómputo específica y a un único valor de b , en los próximos párrafos realizaremos un análisis más detallado para determinar un valor óptimo de δ para el caso de densidad exponencial.

Determinación del *ratio delta*

Habiendo fijado valores de D iguales a los obtenidos para el caso de densidad lineal, nos encontramos en condiciones de explorar los errores de integración como función de δ en mayor detalle. Calcularemos la diferencia entre el error numérico y las soluciones analíticas para todas las combinaciones de grillas de cómputo (Tabla 3.2) y modelos de cascarón esférico (Tabla 3.1), variando los valores de δ entre 10^{-3} y 10^1 . Los cálculos son repetidos para cada valor de $b \in \{1, 2, 5, 10, 30, 100\}$ (Fig. 3.3) con el objeto de analizar la precisión del método para diversos tipos de funciones exponenciales. Dado que valores mayores de δ producen menores subdivisiones, nuestra intención es hallar los valores máximos de δ que producen una diferencia relativa por debajo del umbral de 0.1 %. La Figura 3.5 muestra las diferencias relativas resultantes para V y g_z como función de δ . Por brevedad, cada curva corresponde a la diferencia máxima entre todos los modelos de cascarón esférico. Las curvas correspondientes a $b = 1$ y $b = 2$ se encuentran por debajo del umbral 0.1 % para todos los valores de δ y no se ven modificadas para valores de $\delta > 0.8$, indicando que esas funciones de densidad son lo suficientemente suaves como para poder prescindir de discretizaciones en la dirección radial. Para todos los otros valores de b , las diferencias caen debajo del umbral si $\delta = 0.1$. Estos resultados indican que no hay una relación significativa entre la suavidad de la densidad exponencial y el error numérico.

3.4.3. Densidad sinusoidal

Hasta ahora hemos analizado el comportamiento de la discretización basada en densidad contra funciones de densidades lineales y exponenciales. Sin embargo, este nuevo algoritmo es adecuado para ser utilizado con funciones continuas más complejas, por ejemplo funciones no monotónicas o que presenten múltiples puntos de inflexión. Aunque este tipo de funciones de densidad son cuantitativamente raras de encontrar en seteos geológicos, queremos someter nuestro algoritmo a estos casos con el objetivo de mostrar su capacidad para resolver situaciones más complejas que las abordadas en las secciones anteriores.

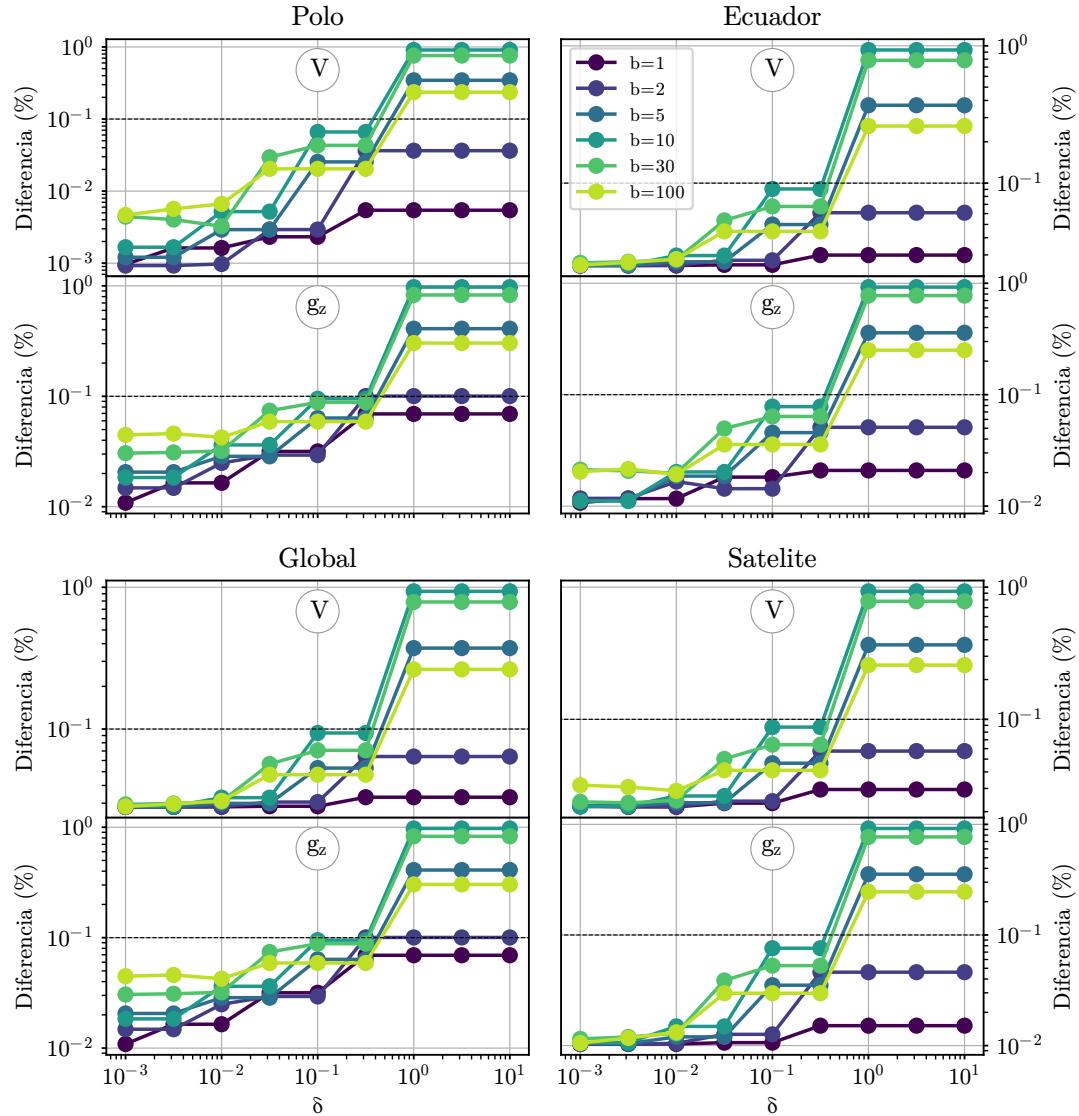


Figura 3.5: Diferencias entre el error numérico y las soluciones analíticas como función del *ratio* δ para varias funciones de densidad exponenciales. Estas comparaciones fueron llevadas a cabo para cada combinación entre los modelos de cascarón (Tabla 3.1) y las grillas de cómputo (Tabla 3.2) haciendo uso de un valor fijo de D . Cada curva corresponde a la diferencia máxima obtenida para cada modelo de cascarón para un valor particular de b (ec. 3.18). Las diferencias están reportadas como un porcentaje de las soluciones analíticas. Las líneas horizontales de a trazos y color negro representan el umbral de precisión de 0.1 %.

Además, obtener un valor óptimo del *ratio* δ para casos de funciones de densidades irreales nos permitiría extrapolarlo a casos más simples y realistas. Es por esta razón que los análisis que se describirán a continuación están destinados a someter el algoritmo a sus extremos y no para emular un escenario real.

Consideremos modelos de cascarones específicos con una función de densidad sinusoidal que se define de la siguiente manera:

$$\rho(r') = A \sin\left(2\pi b \frac{r' - R}{R_2 - R_1}\right) + A, \quad (3.21)$$

donde A es una constante que controla la amplitud y la traslación vertical de la función seno, R es el radio medio de la Tierra y b es una constante adimensional que regula cuántos períodos de la función trigonométrica serán incluidos dentro de los radios interior y exterior. La solución analítica para el potencial V y su gradiente vertical g_z que genera un cascarón esférico con una densidad sinusoidal puede hallarse en el Apéndice 3.A.

Calculamos la diferencia relativa entre los resultados numéricos y la solución analítica para V y g_z para cada combinación de modelos de cascarones (Tabla 3.1) y grillas de cómputo (Tabla 3.2). Hemos fijado el *ratio* distancia-tamaño D a los valores obtenidos para el caso de densidad lineal y exploramos valores de δ entre 10^{-4} y 1. Los cálculos fueron repetidos para valores de b iguales a 1, 2, 5 y 10. (Fig. 3.6). En todos los casos el valor de A fue fijado a 1650 kg/m^3 .

La Figura 3.7 muestra las diferencias relativas entre los resultados numéricos y las soluciones analíticas para casos de densidades sinusoidales. Nuevamente, cada curva corresponde a la diferencia máxima para cada modelo de cascarón. Para todos los valores de b , a excepción de $b = 10$, las diferencias caen debajo del umbral 0.1 % para $\delta = 0.1$. En el caso de $b = 10$, un menor valor de $\delta = 0.01$ es necesario para alcanzar dicho umbral. Vale notar que incluso para el caso de $b = 10$, las diferencias obtenidas utilizando $\delta = 0.1$ se encuentran debajo del 1 %.

3.5. Desempeño del algoritmo

Dado que el algoritmo de discretización basada en densidad introduce subdivisiones a lo largo de la dirección radial, es razonable suponer que el tiempo de cómputo para el caso de densidades variables será más alto que en el caso de densidades homogéneas. Comparar directamente el tiempo de cómputo en ambos casos no podría ser realizado de forma significativa, ya que dependería fuertemente en la implementación del algoritmo, la elección del lenguaje de programación y la función densidad en particular. Una forma indirecta de estimar cuánto aumenta el costo computacional es obtener la cantidad de subdivisiones realizadas sobre el teseroide durante la discretización basada en densidad.

Analizamos densidades exponenciales (ec. 3.18) con valores de b iguales a 1, 2, 5, 10, 30 y 100, y densidades sinusoidales (ec. 3.21) con valores de b iguales a

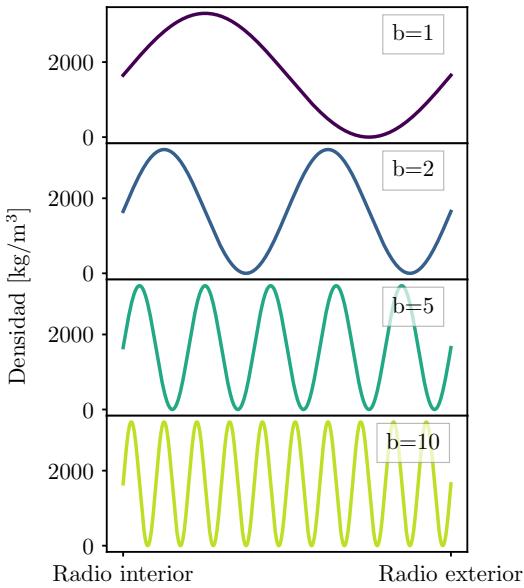


Figura 3.6: Densidades sinusoidales asignadas a los cascarones esféricos durante la determinación del *ratio* δ . Cada función de densidad corresponde a un valor distinto de b en la ecuación 3.21.

1, 2, 5, y 10. Aplicamos el algoritmo de discretización basado en densidad sobre cada una de esas funciones y registramos la cantidad de divisiones que produce en cada caso. Las Figuras 3.8a y 3.8b muestran las densidades exponenciales y sinusoidales junto con sus correspondientes puntos de discretización como círculos naranja.

En el caso de densidad exponencial, el algoritmo realiza una única subdivisión independientemente del valor de b . Por lo tanto, el tiempo de cómputo podría estimarse como el doble del caso de una densidad homogénea (asumiendo implementaciones idénticas). Por otro lado, la Figura 3.8c muestra una relación casi lineal entre el número de divisiones para el caso de densidad sinusoidal y los valores de b . Por lo tanto, el tiempo de cómputo en estos casos parece depender de la cantidad de períodos de la función sinusoidal contenidos dentro de los límites del teseroide.

3.6. Aplicación a la Cuenca Neuquina

Hemos aplicado los nuevos algoritmos y los valores óptimos para D y δ determinados previamente para calcular los efectos gravitatorios de la Cuenca Neuquina, una cuenca sedimentaria localizada al este de los Andes, entre las latitudes 32°S y 40°S (Fig. 3.9a). La cuenca incluye siliciclastos marinos y sedimentarios, carbonatos y evaporitas acumuladas durante el Jurásico y el Cretáci-

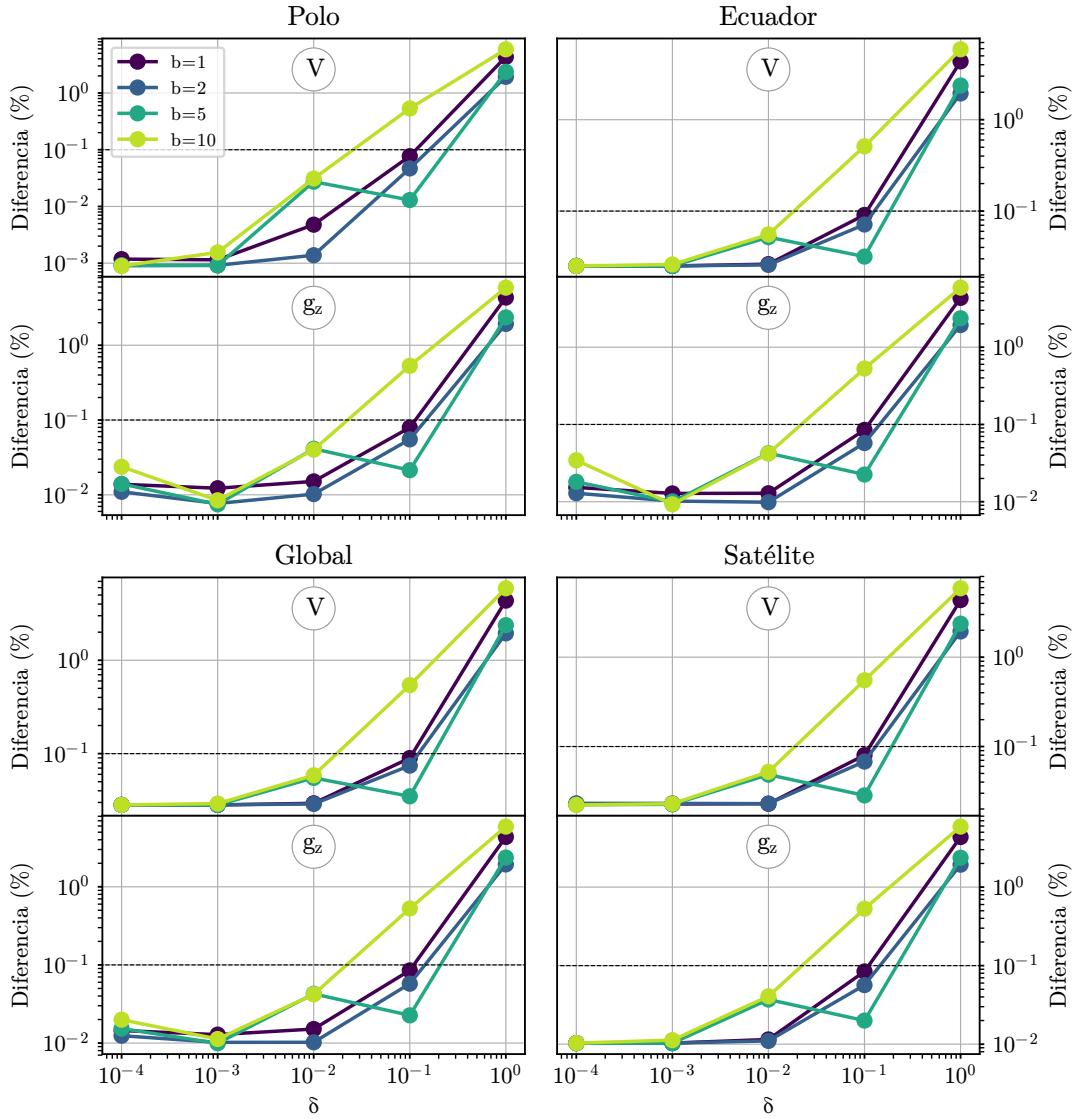


Figura 3.7: Diferencias entre los resultados numéricos y las soluciones analíticas como función de δ para diferentes funciones de densidad sinusoidales. Las comparaciones fueron llevadas a cabo para cada modelo de cascarón (Tabla 3.1) y cada grilla de cómputo (Tabla 3.2) haciendo uso de un valor fijo de D . Cada curva corresponde a la diferencia máxima obtenida para cada modelo de cascarón para un valor particular de b (ec. 3.21). Las diferencias están reportadas como un porcentaje de las soluciones analíticas. Las líneas horizontales de a trazos y color negro representan el umbral de precisión de 0.1 %.

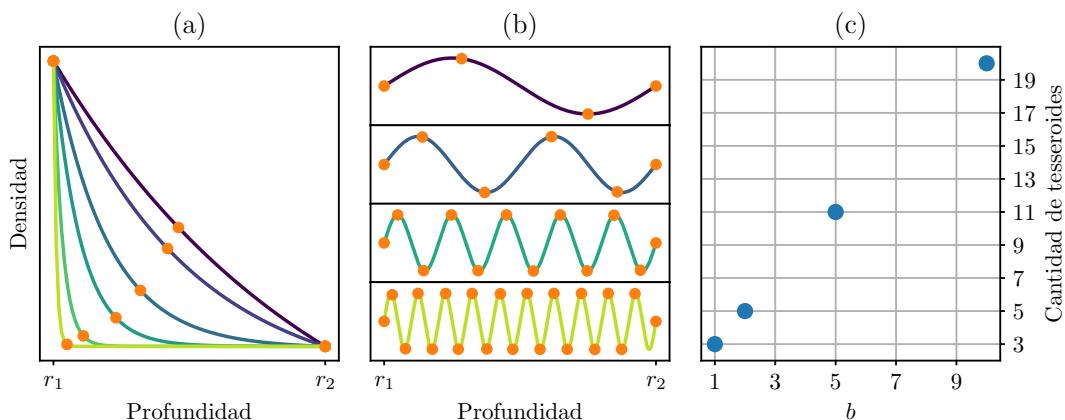


Figura 3.8: Cantidad de subdivisiones realizadas por el algoritmo de discretización basada en densidad (con $\delta = 0.1$) en caso de (a) funciones de densidad exponencial con los mismos valores de b presentes en la Fig. 3.3, (b) funciones de densidad sinusoidal con los mismos valores de b presentes en la Fig. 3.6. En ambas figuras, las ubicaciones de las discretizaciones están marcadas con círculos naranja. (c) Cantidad de tesseroides obtenidos en el caso de densidad sinusoidal en función de b .

to constituyendo un registro estratigráfico hasta 5000m de profundidad ([Howell et al., 2005](#)).

El espesor del paquete sedimentario fue digitalizado a partir de [Heine \(2007\)](#) sobre una grilla regular con una resolución de 0.05° en las direcciones longitudinal y latitudinal (Fig. 3.9b). Creamos un modelo del paquete sedimentario a partir de tesseroides de $0.05^\circ \times 0.05^\circ$ ubicados sobre cada nodo de la grilla. El límite superior de cada tesserode fue fijado a la altitud media de la topografía de la cuenca (845 m sobre el radio medio de la Tierra) y el límite inferior es seleccionado de forma tal que cada tesserode posea la misma dimensión que el espesor de la cuenca sobre el correspondiente punto de la grilla.

Además debemos definir una función densidad para el modelo de tesseroides. [Sigismundi \(2012\)](#) determinó valores mínimos y máximos para el contraste de densidad de la Cuenca Neuquina de -412 kg/m^3 y -275 kg/m^3 , respectivamente. Hemos elegido una variación de densidad exponencial (ec. 3.18) que asume el valor mínimo en la superficie superior y un valor máximo a una profundidad de 5014 m (la región más profunda de la cuenca), con un valor de b igual a 3. Esta variación de densidad se encuentra dentro de los órdenes de magnitud utilizados por [Cowie y Karner \(1990\)](#) y [Cordell \(1973\)](#). La función de densidad se muestra en la Figura 3.10.

Finalmente, hemos calculado el potencial gravitatorio V y la componente vertical de su gradiente (g_z) sobre una grilla de cómputo compuesta por 159×163 nodos (un espaciado de 0.05° en las direcciones longitudinal y latitudinal)

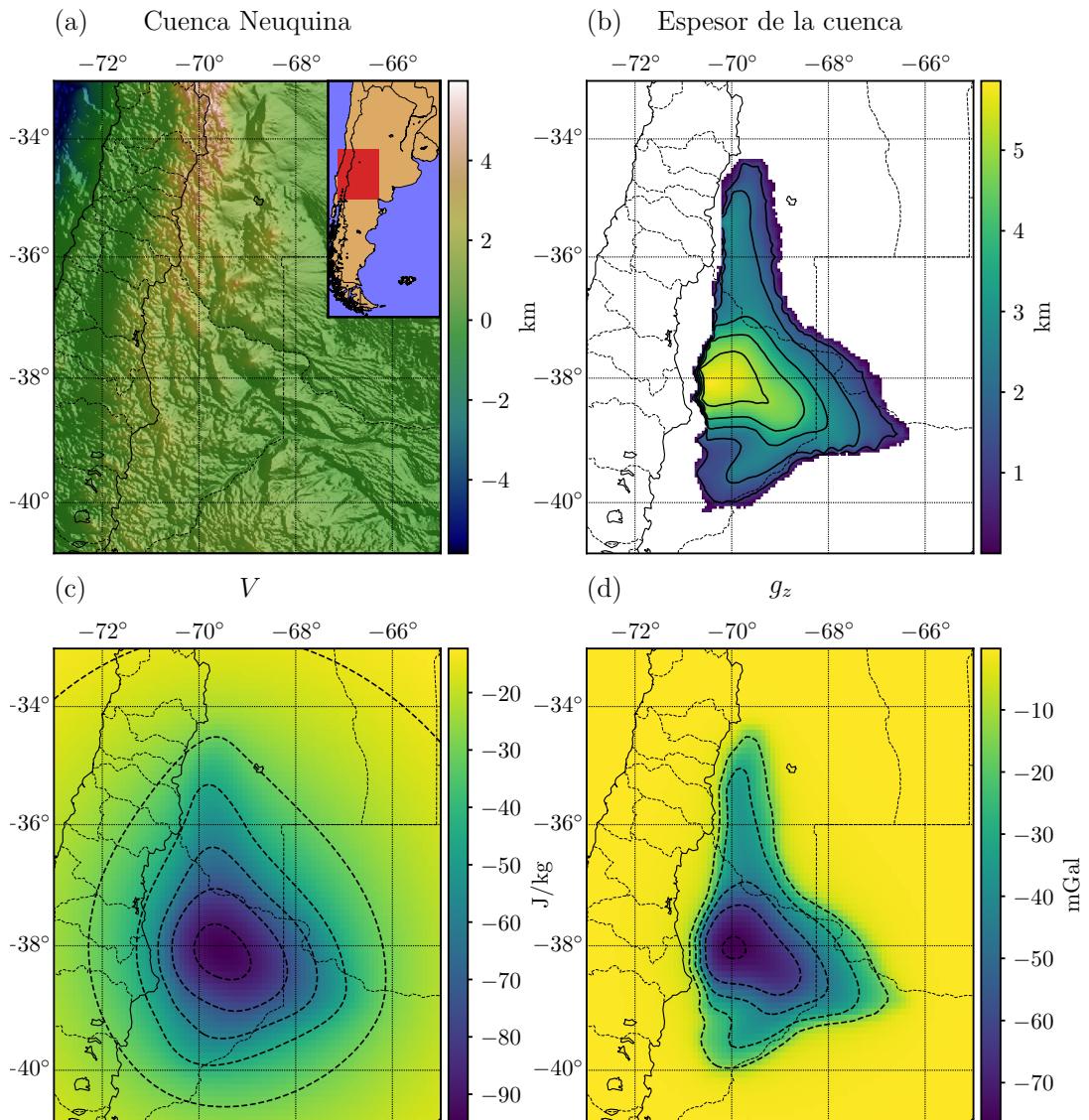


Figura 3.9: Efectos gravitatorios de la Cuenca Neuquina modelada utilizando teseroides con una densidad exponencial como función de la profundidad. (a) Topografía de la Cuenca Neuquina (en km) y su ubicación geográfica en Sudamérica, (b) espesor de la cuenca sedimentaria (en metros; [Heine, 2007](#)) (c) potencial gravitatorio resultante V , (d) componente vertical del gradiente (g_z), calculadas a 10km de altitud sobre el radio medio de la Tierra.

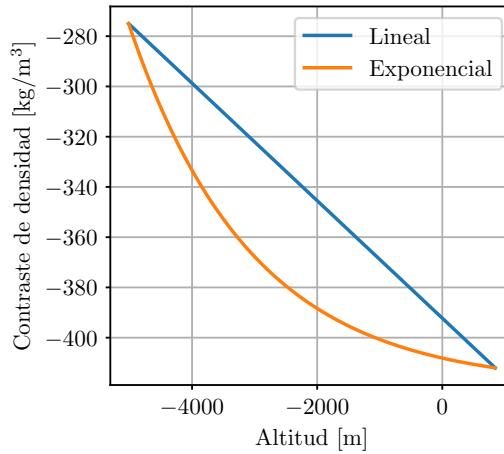


Figura 3.10: Contrastos de densidad lineales y exponenciales utilizados para el cálculo de los campos gravitatorios generados por un modelo de la Cuenca Neuquina compuesto por teseroides. La altitud se define por encima del radio medio de la Tierra, y sus ejes se extiende entre el puntos más profundo hasta el punto más alto de la cuenca.

a una altitud de 10 km sobre el radio medio de la Tierra. Los campos resultantes pueden apreciarse en las Figuras 3.9c-d.

Calculamos las diferencias entre los resultados para la densidad exponencial y para aquellos generados por el mismo modelo pero con una densidad constante dada por el valor medio entre -412 kg/m^3 y -275 kg/m^3 . Además calculamos las diferencias entre el modelo de densidad exponencial y uno con densidad lineal que asume estos dos valores en los puntos más altos y más profundos de la cuenca (Fig. 3.10). Las Figuras 3.11a-b y 3.11c-d muestran las diferencias con la densidad constante y con la densidad lineal, respectivamente. La diferencia máxima absoluta en los campos g_z es aproximadamente de 8 mGal para el caso de densidad homogénea y aproximadamente de 6 mGal para el caso lineal. Estas discrepancias se encuentran bien por encima de los errores de la mayoría de los relevamientos gravimétricos.

3.7. Discusión

Al incluir la función densidad dentro de la GLQ, el método aquí descripto puede ser aplicado sin ninguna modificación a cualquier teseroide cuya densidad puede representarse por una función continua en la dirección radial. El algoritmo de discretización basado en la densidad divide el teseroide a lo largo de la dirección radial con el objetivo de garantizar una integración precisa. Este algoritmo es independiente de la integración GLQ y puede potencialmente ser aplicado para determinar una discretización óptima a la hora de aproximar una

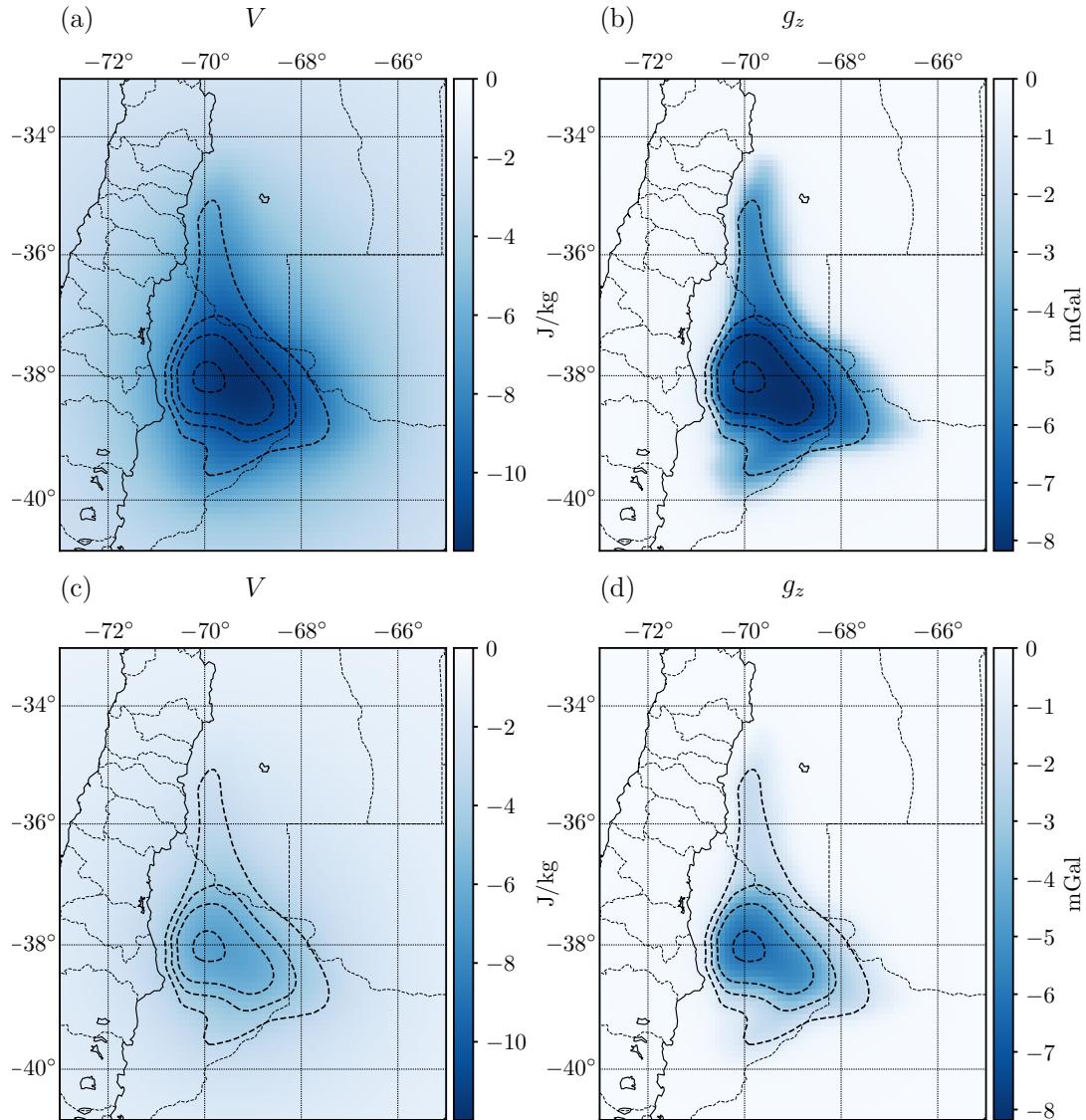


Figura 3.11: Diferencias entre los campos gravitatorios generados por el modelo de teseroides de la Cuenca Neuquina con una densidad exponencial y con el mismo modelo pero con densidades homogéneas y lineales. (a)-(b) Diferencias de V y g_z entre el modelo con densidad exponencial y el modelo con densidad homogénea, (c)-(d) diferencias de V y g_z entre el modelo con densidad exponencial y el modelo con densidad lineal, calculadas a 10km de altitud sobre el radio medio de la Tierra.

función mediante funciones lineales de a pasos ([Lin y Denker, 2019](#)) o funciones polinomiales de a pasos ([Fukushima, 2018](#)).

Los resultados experimentales muestran que la discretización adaptativa bidimensional es suficiente para alcanzar errores debajo del 0.1 % en conjunto con una **GLQ** de segundo orden en caso de densidades lineales (Fig. 3.2). Los valores del *ratio* distancia-tamaño determinados para el potencial gravitatorio ($D = 1$) y para su derivada vertical ($D = 2.5$) son compatibles por los valores determinados por [Uieda et al. \(2016\)](#). Estos resultados muestran además que no existe una relación significativa entre la precisión del método y el espesor del modelo de teseroides.

La exploración del espacio $D-\delta$ para el caso de una densidad exponencial (Fig. 3.4) muestra que los valores de D determinados para el caso lineal son igualmente válidos para ser aplicados al caso exponencial. Además, las diferencias con respecto a las soluciones analíticas caen por debajo del umbral de 0.1 % solo para valores de δ menores a 0.1. Por lo tanto, la aplicación de la discretización basada en la densidad es necesaria para alcanzar el nivel de precisión deseada en casos de densidades no lineales.

Un análisis de errores más detallados muestra que el valor de $\delta = 0.1$ es suficiente para garantizar errores por debajo del 0.1 % para cualquiera de las funciones exponenciales (Fig. 3.5) y para la mayoría de las funciones sinusoidales (Fig. 3.7) que aquí se han probado. La excepción se presenta en el caso de una función sinusoidal con $b = 10$ (ec. 3.21), para la cual un valor de $\delta = 0.01$ es necesario para alcanzar el umbral 0.1 %. Sin embargo, utilizar un valor de $\delta = 0.1$ en este caso produce resultados con errores por debajo del 1 %.

Aunque los algoritmos aquí propuestos pueden ser aplicados a cualquier función de densidad continua, los valores óptimos de D y δ fueron empíricamente determinados solo para funciones lineales, exponenciales y sinusoidales. Por lo tanto, solo podemos afirmar con certeza que estos valores producen resultados con errores por debajo de 0.1 % para los tipos de funciones nombradas. Sin embargo, todas las experiencias numéricas incluyen los escenarios menos favorables (puntos de cómputo sobre la superficie de los teseroides, teseroides de gran tamaño, funciones densidad altamente variables, etc.). Por esta razón, es plausible extrapolar los valores óptimos de D y δ a cualquier densidad continua que represente variaciones realistas de estructuras geológicas. A pesar de esto, alentamos a los usuarios y las usuarias de estos algoritmos a llevar a cabo experimentaciones similares con el objetivo de evaluar su precisión en caso de utilizar funciones de densidades más complejas que las aquí probadas.

El análisis del desempeño del algoritmo muestra que el tiempo de cómputo en los casos de densidades no lineales es como mínimo el doble del necesario para el caso de densidad homogénea o lineal. Esta proporcionalidad puede aumentar con la cantidad de puntos de inflexión de la función densidad. Como en la mayoría de los métodos numéricos, existe un compromiso entre tiempo de cómputo y precisión. Sin embargo, los perfiles de densidad suelen tener pocos puntos de inflexión en la mayoría de las aplicaciones geofísicas. Por lo tanto, el

tiempo de cómputo se encontrará en el mismo orden de magnitud que el caso de densidad homogénea para la mayoría de las aplicaciones en el mundo real.

3.8. Conclusiones

Hemos desarrollado una nueva metodología para calcular el potencial gravitatorio y su gradiente generados por un teseroide con una densidad dada por una función continua de la coordenada radial. Este método resuelve numéricamente integrales volumétricas mediante la [GLQ](#) incluyendo la función densidad dentro del integrando. Mediante una implementación del mismo en lenguaje Python, los usuarios y las usuarias pueden definir su propia función densidad y pasarlala como entrada al algoritmo. Esto permite el uso de funciones continuas arbitrarias sin necesidad de realizar modificaciones sobre el método o el software. La precisión de la integración numérica es controlada automáticamente por un algoritmo de discretización adaptativa bidimensional y un nuevo algoritmo de discretización basado en densidad. El primero subdivide iterativamente al teseroide en caso de que las relaciones entre la distancia al punto de cómputo y las dimensiones longitudinal y latitudinal del teseroide sean menores que un *ratio* distancia-tamaño D predefinido. Este algoritmo minimiza los errores de integración a cuando el punto de cómputo se encuentra cerca del teseroide. Sin embargo, la discretización adaptativa por sí sola no es suficiente para garantizar una precisión aceptable en caso de teseroides con densidades no lineales. Para solucionar esto, el algoritmo de discretización basada en densidad divide al teseroide a lo largo de la dirección radial en los lugares donde la *máxima variación* de la función densidad ocurre. La cantidad de subdivisiones a lo largo del radio, y por ende la precisión del cómputo, están controladas por el parámetro *delta* δ . Este nuevo algoritmo está diseñado para minimizar el error debido a la incapacidad de la [GLQ](#) de producir aproximaciones precisas en caso de funciones de densidad con variaciones pronunciadas.

Hemos determinado de manera empírica valores óptimos para los parámetros D y δ comparando los resultados numéricos con soluciones analíticas para cascarones esféricos. Nuestro análisis incluye un rango de modelos de teseroides y grillas de cómputo así como funciones de densidades de las más utilizadas en problemas reales. Estos valores minimizan el tiempo de cómputo mientras mantienen el error numérico por debajo de un umbral de 0.1 %. Las funciones de densidad utilizadas para establecer estos valores óptimos fueron lineales, exponenciales y sinusoidales. La función lineal representa la variación más sencilla de densidad y no requiere la aplicación de una discretización basada en densidad. Mediante este análisis hemos obtenido valores óptimos para el *ratio* distancia-tamaño D de 1 y 2.5 para el potencial gravitatorio y para su gradiente, respectivamente. Con el objetivo de analizar la precisión de la discretización basada en densidad, hemos calculado el error numérico para casos de densidades exponenciales, desde funciones suaves hasta funciones con fuertes pendientes,

e incluso para funciones sinusoidales de diferentes longitudes de onda. Vale notar que la densidad sinusoidal fue tomada en cuenta con la intención de someter al algoritmo a sus extremos y no para emular escenarios de la vida real. Valores de δ iguales a 0.1 son suficientes para garantizar errores numéricos por debajo del umbral de 0.1 % para la mayoría de los casos examinados. Estos resultados podrían ser extrapolados a otras funciones continuas de densidad que sean lo suficientemente suaves. Tal y como es el caso de la mayoría de las aplicaciones geofísicas. Menores valores de δ pueden ser utilizados para densidades altamente variables con el objetivo de incrementar la precisión de la integración.

Una parte del desempeño y eficiencia computacional es sacrificada con el objeto de obtener un método preciso y de propósito general. La discretización basada en densidad aumenta el tiempo de cómputo al incrementar el número de integraciones según la GLQ. De la misma manera, permitiendo a los usuarios y las usuarias elegir sus propias funciones de densidad incurrimos en un aumento del costo computacional debido a las subsecuentes evaluaciones de dichas funciones. Además, resulta imposible optimizar el código fuente y la formulación matemática para el caso general donde no conocemos las especificidades de la función densidad. Sin embargo, la discretización basada en densidad es independiente de la discretización adaptativa y de la GLQ. Es decir, podemos tomar a este algoritmo como un preprocesado que luego puede ser combinado con métodos de integración más específicos.

La aplicación de este algoritmo al modelado de la Cuenca Neuquina (Argentina) demuestra que los efectos de la compactación sedimentaria no deben ser ignorados. Los resultados producidos por una densidad exponencial muestran un error de 8 mGal con respecto al caso de una densidad homogénea, y un error de 6 mGal con respecto a una función lineal. Los modelos directos robustos son una componente clave para cualquier método de inversión de datos gravimétricos. Por ejemplo, errores de esta magnitud pueden resultar en una significativa sobreestimación del espesor del paquete sedimentario.

3.A. Soluciones analíticas para un cascarón esférico

Consideremos un cascarón esférico con radio interior R_1 y radio exterior R_2 , cuya densidad $\rho(r')$ es función de la coordenada radial. Deseamos obtener una expresión analítica para los campos gravitatorios que genera en cualquier punto externo ubicado a una distancia r del centro del cascarón ($r \geq R_2$).

Según el Teorema del Cascarón de Newton (*Newton's Shell Theorem, Theorem XXXI*) (Binney y Tremaine, 2008; Chandrasekhar, 1995), el potencial gravitatorio generado por el cascarón esférico en cualquier punto externo es equivalente al que generaría si toda su masa estuviera concentrada en un punto localizado en su centro:

$$V_{\text{sh}}(\phi, \lambda, r) = \frac{GM}{r}, \quad (3.22)$$

donde M es la masa total del cascarón, la cual puede ser fácilmente calculada como:

$$M = \iiint_{\Omega} \rho(r') dV = 4\pi \int_{R_1}^{R_2} \rho(r') r'^2 dr', \quad (3.23)$$

donde Ω simboliza el volumen del cascarón.

Combinando las dos ecuaciones anteriores, obtenemos la siguiente expresión para el potencial:

$$V_{\text{sh}}(r) = \frac{4\pi G}{r} \int_{R_1}^{R_2} r'^2 \rho(r') dr', \quad (3.24)$$

la cual es equivalente a la obtenida por [Binney y Tremaine \(2008, p.62\)](#).

El gradiente de aquellos potenciales que dependen solo de r poseen solo una componente no nula: la componente vertical (g_z). Según [Grombein et al. \(2013\)](#), esta puede calcularse como:

$$g_z(r) = \frac{V_{\text{sh}}(r)}{r}. \quad (3.25)$$

A partir de la ecuación 3.24 podemos obtener expresiones del potencial gravitatorio para diferentes funciones de densidad. La integración de las siguientes funciones de densidad fueron llevadas a cabo mediante el uso de SymPy ([Meurer et al., 2017](#)), una librería de Python para matemática simbólica.

3.A.1. Densidad lineal

Para una densidad lineal

$$\rho(r') = ar' + b, \quad (3.26)$$

el potencial gravitatorio en cualquier punto externo es

$$V_{\text{sh}}^{\text{lin}}(r) = \pi G \left[a \frac{R_2^4 - R_1^4}{r} + b \frac{4}{3} \frac{R_2^3 - R_1^3}{r} \right]. \quad (3.27)$$

El primer término de la ecuación reproduce el potencial generado por un cascarón esférico con densidad variable $\rho(r') = ar'$, mientras que el segundo término es equivalente al potencial generado por el mismo cascarón con densidad homogénea $\rho = b$ ([Grombein et al., 2013; Mikuška et al., 2006](#)). La ecuación 3.27 coincide con la obtenida por [Lin y Denker \(2019\)](#).

3.A.2. Densidad exponencial

Para una densidad exponencial

$$\rho(r') = Ae^{-k(r'-R)}, \quad (3.28)$$

donde A , k y R son constantes, el potencial gravitatorio en cualquier punto exterior es

$$V_{\text{exp}}(r) = \frac{4\pi G}{r} \frac{A}{k^3} \left[\left(R_1^2 k^2 + 2R_1 k + 2 \right) e^{-k(R_1 - R)} - \left(R_2^2 k^2 + 2R_2 k + 2 \right) e^{-k(R_2 - R)} \right]. \quad (3.29)$$

3.A.3. Densidad sinusoidal

Para una función de densidad sinusoidal

$$\rho(r') = A \sin(k(r' - R)), \quad (3.30)$$

donde A , k y R son constantes, el potencial gravitatorio en cualquier punto exterior es

$$V_{\text{sine}}(r) = \frac{4\pi G}{r} \frac{A}{k^3} \left[(2 - k^2 R_2^2) \cos(k(R_2 - R)) + 2kR_2 \sin(k(R_2 - R)) - (2 - k^2 R_1^2) \cos(k(R_1 - R)) - 2kR_1 \sin(k(R_1 - R)) \right]. \quad (3.31)$$

Capítulo 4

Fuentes equivalentes potenciadas por gradiente

4.1. Introducción

Las mediciones de las anomalías en los campos potenciales, como el disturbio de gravedad o anomalías magnéticas, son ampliamente utilizadas en la exploración geofísica debido a su bajo costo de adquisición. Estos datos pueden ser recolectados utilizando sistemas terrestres, aéreos, marítimos o satelitales. Durante las mediciones terrestres, los datos son recabados siguiendo trayectorias irregulares a lo largo de la superficie del terreno, produciendo altas variaciones en la altitud de los puntos de observación en zonas montañosas. Las metodologías aéreas y satelitales obtienen los datos a lo largo de líneas de vuelo, produciendo mediciones muy próximas espacialmente a lo largo de trayectorias casi rectas u orbitales, respectivamente, pero con espaciados mucho mayores entre líneas adyacentes. La altitud de observación también puede variar en estos casos debido al movimiento vertical de la aeronave. El procesado de estos datos suele incluir interpolaciones sobre grillas regulares a altitud constante, tanto para mejorar la visualización con el objetivo de realizar interpretaciones, así como para preparar los datos para posteriores procesamientos y modelados (por ejemplo, reducción al polo, cálculo de derivadas, continuación ascendente, deconvolución de Euler).

Existen muchos métodos en la literatura que permiten realizar interpolaciones bidimensionales, por ejemplo *splines* de curvatura continua en tensión ([Smith y Wessel, 1990](#)), *splines* biarmónicas ([Sandwell, 1987](#)), y kriging ([Hansen, 1993](#)). Estos métodos de propósito general poseen limitaciones a la hora de interpolar datos provenientes de campos potenciales: (i) no son capaces de tomar en cuenta las altitudes de los puntos de observación, (ii) las funciones interpoladoras no son necesariamente armónicas, siendo esta la principal presuposición de muchas técnicas de procesado (por ejemplo, continuación ascendente y derivadas verticales).

Un método ampliamente utilizado para interpolar datos gravitatorios y magnéticos es la técnica de fuentes equivalentes (también conocida como capa equivalente –*equivalent layer* en inglés–, funciones de bases radiales –*radial basis functions* en inglés–, o interpolaciones con funciones de Green). Inicialmente introducido por [Dampney \(1969\)](#), el método consiste en ajustar un conjunto finito de fuentes elementales a los datos observados, y luego usar ese modelo para predecir nuevos valores. Además de las interpolaciones, las fuentes equivalentes han sido utilizadas para realizar reducciones al polo de datos magnéticos ([Guspí y Novara, 2009](#); [Nakatsuka y Okuma, 2006](#); [Silva, 1986](#)), continuación ascendente ([Emilia, 1973](#); [Li y Oldenburg, 2010](#)), procesado conjunto de datos del gradiente gravimétrico [Barnes y Lumley \(2011\)](#), modelado del campo magnético litosférico ([Kother et al., 2015](#)), recuperación del vector de inducción magnética a partir de anomalías magnéticas de campo total ([Li et al., 2020](#)), entre otras.

Vale mencionar el método de colocación por mínimos cuadrados (*least-squares collocation method* en inglés), ampliamente utilizado en Geodesia. ([Tscherning, 2015](#), y referencias allí citadas). Esta metodología se suele aplicar para combinar e interpolar diferentes funciones lineales de anomalías derivadas del potencial gravitatorio (anomalías de gravedad, disturbio de gravedad, deflexiones de la vertical, altitud del geoide, etc.). Al igual que las fuentes equivalentes, el método de colocación requiere resolver un sistema de ecuaciones lineales de gran tamaño, cuyo orden equivale a la cantidad de datos observados. De esta manera, la aplicación práctica de ambos métodos sufren de los mismos desafíos computacionales.

Muchas variantes de la técnica de fuentes equivalentes han sido propuestas, usualmente intentando obtener soluciones más precisas o en menor tiempo. Los factores claves que varían entre ellos son: (i) el tipo de fuente, (ii) la ubicación de las fuentes, y (iii) la estrategia de solución.

Los tipos de fuentes más utilizados son masas puntuales para datos gravitatorios o dipolos para datos magnéticos (e.g., [Mendonça y Silva, 1994](#); [Silva, 1986](#); [Siqueira et al., 2017](#); [von Frese et al., 1981](#)). Sin embargo, también se han utilizado exitosamente prismas rectangulares (por ej., [Barnes y Lumley, 2011](#); [Jirigalatu y Ebbing, 2019](#); [Li et al., 2020](#)); teseroides ([Bouman et al., 2016](#)); y hasta fuentes puntuales junto con funciones simples tales como la inversa de la distancia ([Cordell, 1992](#)), en vez de los campos gravitatorios o magnéticos que ellas generan.

La ubicación de las fuentes suele elegirse de acuerdo a alguno de las siguientes estrategias. El método más común es distribuir las fuentes en una grilla regular a una profundidad constante (por ej., [Barnes y Lumley, 2011](#); [Leão y Silva, 1989](#); [Oliveira et al., 2013](#)). Alternativamente, podemos ubicar una fuente debajo de cada punto de observación (por ej., [Cordell, 1992](#); [Siqueira et al., 2017](#)). Algunos trabajos recientes de [Li et al. \(2020\)](#) ubican las fuentes en dos capas superpuestas a profundidades diferentes.

Los coeficientes de las fuentes equivalentes suelen ser estimados mediante mínimos cuadrados ponderados. Esto implica una carga computacional alta

cuando la cantidad de datos es grande (por ej. muestras aéreas o satelitales).

Para reducir esta carga computacional, Mendonça y Silva (1994) construyen la solución iterativamente al tratar un solo dato al mismo tiempo utilizando el concepto de *dato equivalente*. Leão y Silva (1989) procesan los datos de entrada mediante una ventana móvil, solo ajustando los datos dentro de la ventana y prediciendo las observaciones en su centro. Li y Oldenburg (2010) y Barnes y Lumley (2011) aplican diferentes operaciones para generar una representación dispersa de la matriz de sensibilidades (*wavelet compression* y *quadtree discretization*, respectivamente), lo cual mejora significativamente la velocidad de ejecución del algoritmo de mínimos cuadrados. Oliveira et al. (2013) parametrizan las fuentes equivalentes como una función polinomial bivariada a trazos, reduciendo el número de parámetros en la solución. Siqueira et al. (2017) desarrollaron una solución iterativa en la cual la matriz de sensibilidades es transformada en una matriz diagonal con términos constantes a través del concepto de *exceso de masa*. Jirigalatu y Ebbing (2019) aplican el método Gauss-FFT para acelerar las operaciones vinculadas al modelado directo y resuelven el problema de mínimos cuadrados utilizando el descenso del gradiente con el objetivo de evitar calcular las matrices Hessianas y resolver sistemas lineales.

Muchos de los métodos existentes resuelven problemas subdeterminados, requiriendo un número mayor de fuentes equivalentes que de cantidad de datos. Algunos de ellos alcanzan mayores eficiencias al restringir su aplicación a tipos de datos específicos (Siqueira et al., 2017), realizar las interpolaciones sobre grillas regulares (Leão y Silva, 1989), o requerir datos ya grillados (Takahashi et al., 2020), por nombrar algunos. Además, muchas de las optimizaciones propuestas son complejas de implementar en un programa de computación de uso genérico, limitando que sean ampliamente utilizados.

En el presente trabajo, proponemos dos estrategias para reducir el costo computacional de la técnica de fuentes equivalentes:

1. Reducir el número de fuentes equivalentes para conjuntos de datos con sobremuestreo mediante una estrategia de *promedio en bloques* (*block-averaging* en inglés), manteniendo la calidad de la solución.
2. Ajustar el modelo de fuentes equivalentes iterativamente sobre ventanas solapadas utilizando un algoritmo de *potenciación del gradiente* (*gradient-boosting* en inglés) (Friedman, 2001).

La primera estrategia consiste en dividir el área de estudio en bloques y asignar una única fuente a cada bloque, localizada en la ubicación media de los puntos de datos. Para muestras aéreas, navales y satelitales, las cuales presentan sobremuestreo a lo largo de la trayectoria del vehículo, esto puede reducir considerablemente el tamaño del problema inverso manteniendo a su vez la misma calidad en la interpolación.

Por otro lado, el algoritmo de *potenciación del gradiente* permite ajustar el modelo de fuentes equivalentes de manera iterativa, operando de manera individual sobre cada una de las ventanas con solapamiento. Como resultado, nuestro

método resuelve muchos problemas de mínimos cuadrados de menor tamaño en vez de un único gran problema. Este posee algunas semejanzas con la estrategia utilizada por [Leão y Silva \(1989\)](#), pero sin el requerimiento de que tanto las fuentes como los puntos de interpolación se encuentren en grillas regulares.

Mediantes pruebas sobre datos sintéticos, mostramos que: (i) las fuentes *promediadas en bloque* son capaces de alcanzar el mismo nivel de precisión que otras disposiciones de fuentes equivalentes más tradicionales, utilizando una fracción del número de fuentes; y (ii) el algoritmo *potenciación del gradiente* reduce significativamente la memoria necesaria para ajustar grandes cantidades de datos, sin sacrificar precisión en las predicciones. Finalmente, una combinación de ambas estrategias es utilizada para procesar una colección de aproximadamente 1.7 millones de datos de gravedad tomados sobre la superficie de Australia.

4.2. Metodología

4.2.1. La técnica de fuentes equivalentes

De aquí en más seguiremos la propuesta de *fuentes equivalentes generalizadas* de [Cordell \(1992\)](#) y asumiremos que cualquier función armónica $d(\mathbf{p})$ puede ser aproximada por la suma de los efectos de M fuentes puntuales

$$d(\mathbf{p}) = \sum_{j=1}^M \frac{c_j}{\|\mathbf{p} - \mathbf{q}_j\|}, \quad (4.1)$$

donde \mathbf{p} y \mathbf{q}_j son, respectivamente, los vectores posición de datos y fuentes en un espacio Cartesiano tridimensional, y c_j es un coeficiente escalar correspondiente a la masa puntual localizada en \mathbf{q}_j . En la Sección 4.2.4 realizamos una discusión acerca de la distribución horizontal y vertical de las fuentes.

En caso de que poseamos mediciones de nuestra función armónica en N puntos de observación $\{\mathbf{p}_1 \mathbf{p}_2 \dots \mathbf{p}_N\}$, podemos escribir un sistema de N ecuaciones de la forma:

$$d_i = \sum_{j=1}^M \frac{c_j}{\|\mathbf{p}_i - \mathbf{q}_j\|} \quad \forall i = 1, 2, \dots, N, \quad (4.2)$$

donde d_i es el efecto de las fuentes sobre el punto \mathbf{p}_i . Estas ecuaciones pueden ser expresadas de forma matricial como:

$$\mathbf{d} = \mathbf{A}\mathbf{c}, \quad (4.3)$$

donde \mathbf{d} es un vector columna que contiene los valores medidos sobre cada uno de los N puntos de observación, \mathbf{c} es un vector columna formado por los M coeficientes c_j y \mathbf{A} es la matriz Jacobiana de $N \times M$ elementos, los cuales se definen como:

$$a_{ij} = \frac{1}{\|\mathbf{p}_i - \mathbf{q}_j\|} \quad (4.4)$$

Para un conjunto de N observaciones \mathbf{d}^o , podemos hallar la solución de la ecuación 4.3 mediante mínimos cuadrados y de esa manera obtener los valores de \mathbf{c} que mejor ajustan a las observaciones. Estos coeficientes pueden ser, a su vez, utilizados para predecir los valores de la función armónica en cualquier otro punto externo a las fuentes, evaluando la ecuación 4.1. El grillado o la continuación ascendente pueden entonces ser realizados mediante predicciones sobre puntos de una grilla regular o sobre puntos a diferentes altitudes, respectivamente.

4.2.2. Solución por mínimos cuadrados amortiguados

Es posible obtener los valores de los coeficientes \mathbf{c} que mejor ajustan a los valores observados \mathbf{d}^o minimizando la función objetivo

$$\phi(\mathbf{c}) = [\mathbf{d}^o - \mathbf{A}\mathbf{c}]^T \mathbf{W} [\mathbf{d}^o - \mathbf{A}\mathbf{c}] + \lambda_d \mathbf{c}^T \mathbf{c}, \quad (4.5)$$

donde \mathbf{W} es una matriz diagonal de $N \times N$ elementos que contiene los pesos ponderados de los datos observados, y λ_d es el parámetro de *amortiguamiento* (*damping* en inglés), el cual es un valor positivo y posee las mismas unidades que los elementos de la matriz Jacobiana.

El segundo término del miembro derecho de la ecuación 4.5 consiste en una regularización Tikhonov de orden cero (Tikhonov, 1977), también conocida como regularización de amortiguamiento (*damping regularization*), que se utiliza para estabilizar la solución.

El parámetro de amortiguamiento controla cuánta regularización será aplicada. Un valor muy grande generaría soluciones muy suaves que fallarían en reproducir las componentes de altas frecuencias presentes en los datos; mientras que un valor muy pequeño resultaría en un sobreajuste, produciendo resultados de interpolación irreales (Martinez y Li, 2016). El rango de valores aceptables para el parámetro de amortiguamiento λ_d dependerá de los valores de la matriz Jacobiana \mathbf{A} y de los coeficientes. Por lo tanto, este rango puede variar (considerablemente en muchas ocasiones) entre diferentes conjuntos de datos, haciendo difícil una apropiada elección en la práctica.

Para resolver este problema, primero escalaremos la matriz Jacobiana de forma tal que sus elementos sean adimensionales y que cada columna posea varianza unitaria. Definimos la matriz diagonal \mathbf{S} como:

$$\mathbf{S} = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_M \end{bmatrix}_{M \times M}, \quad (4.6)$$

donde σ_j es la desviación estándar de la columna j -ésima de la matriz \mathbf{A} . Luego reescribimos el problema directo de la ecuación 4.3 como:

$$\mathbf{d} = \mathbf{AS}^{-1}\mathbf{Sc} = [\mathbf{AS}^{-1}] [\mathbf{Sc}] = \mathbf{Bm} \quad (4.7)$$

donde $\mathbf{B} = \mathbf{AS}^{-1}$ es la matriz Jacobiana escalada y adimensional, y $\mathbf{m} = \mathbf{Sc}$ es el vector que contiene los coeficientes escalados cuyas unidades son las mismas que la de los datos.

La función objetivo definida en la ecuación 4.5 puede ser reescrita como:

$$\phi(\mathbf{m}) = [\mathbf{d}^o - \mathbf{Bm}]^T \mathbf{W} [\mathbf{d}^o - \mathbf{Bm}] + \lambda \mathbf{m}^T \mathbf{m}, \quad (4.8)$$

donde λ es un parámetro de *amortiguamiento adimensional* y la regularización es aplicada a los coeficientes escalados \mathbf{m} en vez de al vector \mathbf{c} . Utilizando un parámetro de amortiguamiento adimensional limitamos el rango de valores de λ que generan las predicciones más precisas, independientemente del conjunto de datos y sus unidades. A partir de nuestra experiencia, recomendamos buscar valores de λ entre 10^{-6} y 10^4 variando por orden de magnitud. La elección del amortiguamiento y de otros hiperparámetros, como la profundidad de las fuentes, puede realizarse a través de métodos estadísticos conocidos, como la validación cruzada.

El vector de coeficientes escalados $\hat{\mathbf{m}}$ que minimiza la función objetivo puede hallarse resolviendo el *sistema de ecuaciones normales* (Menke, 1989):

$$[\mathbf{B}^T \mathbf{WB} + \lambda \mathbf{I}] \hat{\mathbf{m}} = \mathbf{B}^T \mathbf{Wd}^o. \quad (4.9)$$

Una vez que los coeficientes escalados son obtenidos, los coeficientes sin escalar $\hat{\mathbf{c}}$ pueden ser calculados quitando el factor de escalado:

$$\hat{\mathbf{c}} = \mathbf{S}^{-1} \hat{\mathbf{m}}. \quad (4.10)$$

Las operaciones relacionadas al modelado directo que se utilizan para realizar predicciones (por ejemplo, interpolaciones seguidas de continuaciones ascendentes) permanecen sin variaciones al utilizar el vector $\hat{\mathbf{c}}$ en vez de $\hat{\mathbf{m}}$.

4.2.3. Potenciación del gradiente

La potenciación del gradiente fue inicialmente introducida por Friedman (2001, 2002) como un método para ajustar modelos paramétricos de manera aditiva de la siguiente forma:

$$d = \sum_{k=1}^K \alpha_k f(\mathbf{c}_k), \quad (4.11)$$

donde α_k es un coeficiente llamado *tamaño de paso* y f es una función del vector de parámetros \mathbf{c}_k . Para problemas lineales, estos modelos aditivos pueden expresarse según la siguiente ecuación matricial:

$$\mathbf{d} = \sum_{k=1}^K \mathbf{A}_k \mathbf{c}_k . \quad (4.12)$$

Dada la linealidad de las funciones $f(\mathbf{c}_k)$, los parámetros de tamaño de paso α_k pueden ser incluidos en el vector de parámetros \mathbf{c}_k .

Aplicando esta técnica al problema de fuentes equivalentes, podemos transformar la ecuación 4.3 en un modelo aditivo siguiendo estos pasos:

1. Definir un conjunto de M fuentes equivalentes distribuidas a lo largo de la zona de estudio (ver Sección 4.2.4 para más detalles).
2. Definir un conjunto de K ventanas solapadas de igual tamaño que cubren toda la zona de estudio.
3. Crear K conjuntos distintos de fuentes equivalentes, uno por cada ventana. Cada conjunto estará conformado por la porción de las M fuentes originales que caen dentro de la respectiva ventana. Dado que las ventanas se solapan, el número total de fuentes considerando cada uno de los conjuntos será mayor que M .
4. Definir el vector \mathbf{c}_k como los M_k coeficientes correspondientes a las fuentes equivalentes de la ventana k -ésima.
5. Definir la matriz \mathbf{A}_k como la matriz Jacobiana de $N \times M_k$ elementos entre las fuentes de la ventana k -ésima y todos los N puntos de datos de la muestra.
6. Modelar los datos predichos por las fuentes como una superposición de los efectos de los K conjuntos de fuentes equivalentes (ec. 4.12).

El algoritmo de potenciación del gradiente funciona ajustando cada componente del modelo aditivo, una a la vez, a los residuos de la componente anterior. Friedman (2001) demuestra que esto corresponde a una optimización por descenso del gradiente en el *espacio funcional*. El Algoritmo 1 presenta nuestra adaptación del algoritmo de potenciación del gradiente para hallar las soluciones de los K vectores de parámetros \mathbf{c}_k de la ec. 4.12 mediante mínimos cuadrados amortiguados.

Luego de que todos los vectores de coeficientes \mathbf{c}_k fueron estimados, podemos predecir el efecto del modelo aditivo de fuentes equivalentes mediante la suma

$$d(\mathbf{p}) = \sum_{k=1}^K \sum_{j=1}^{M_k} \frac{c_{kj}}{\|\mathbf{p} - \mathbf{q}_{kj}\|} , \quad (4.13)$$

```

1 Definir el vector residual  $\mathbf{r}_0 = \mathbf{d}^o$ 
2 para  $k = 1$  a  $K$  hacer
3   Calcular la matriz Jacobiana  $\mathbf{A}_k$  de  $N \times M_k$  elementos
4    $\mathbf{B}_k = \mathbf{A}_k \mathbf{S}_k^{-1}$ 
5    $\hat{\mathbf{m}}_k = [\mathbf{B}_k^T \mathbf{W}_k \mathbf{B}_k + \lambda \mathbf{I}]^{-1} \mathbf{B}_k^T \mathbf{W}_k \mathbf{r}_{k-1}$ 
6    $\hat{\mathbf{c}}_k = \mathbf{S}_k^{-1} \hat{\mathbf{m}}_k$ 
7    $\mathbf{d}_k = \mathbf{A}_k \hat{\mathbf{c}}_k$ 
8    $\mathbf{r}_k = \mathbf{r}_{k-1} - \mathbf{d}_k$ 
9 fin para

```

Algoritmo 1: Solución mediante potenciación del gradiente de una regresión por mínimos cuadrados amortiguados.

donde c_{kj} es el j -ésimo elemento del vector \mathbf{c}_k y \mathbf{q}_{kj} es el vector posición de la j -ésima fuente de la k -ésima ventana.

Para mejorar la convergencia del algoritmo, Friedman (2002) sugiere introducir aleatoriedad en el proceso de ajuste. En nuestra adaptación logramos esto aleatorizando el orden en el cual las K ventanas son utilizadas durante el algoritmo de potenciación de gradiente. La Sección 4.3.3 explora los efectos de la aleatorización sobre la velocidad de convergencia del algoritmo y sobre la precisión de la interpolación.

Las matrices \mathbf{A}_k poseen solo $N \times M_k$ elementos (donde M_k es la cantidad de fuentes dentro de la ventana k -ésima), las cuales pueden ser considerablemente más pequeñas que la matriz \mathbf{A} con $N \times M$ elementos. Por lo tanto, el algoritmo de potenciación del gradiente permite ajustar modelos de fuentes equivalentes cuyas matrices Jacobianas fueran más grandes que la memoria computacional disponible. Además, podemos incrementar o disminuir el tamaño de las ventanas según sea necesario, dependiendo de la cantidad de fuentes en el modelo y de la memoria disponible.

Podemos mejorar la eficiencia del algoritmo aún más mediante:

1. La utilización de solo los N_k puntos de datos que caen dentro de la ventana k -ésima a la hora de ajustar los coeficientes de las fuentes (pasos 4 y 5 del Algoritmo 1). De esta manera, podemos reemplazar las matrices Jacobianas \mathbf{A}_k de $N \times M_k$ elementos por matrices $\tilde{\mathbf{A}}_k$ más pequeñas de $N_k \times M_k$ elementos. Seguiremos utilizando todos los N puntos de datos cuando calculamos los valores predichos y los residuos (pasos 7 y 8 del Algoritmo 1).
2. La operación de modelado directo que se realiza en el paso 7 puede ser

llevada a cabo mediante una suma (ec. 4.2), en vez de un producto matricial, lo cual permite evitar almacenar en memoria la matriz \mathbf{A}_k de $N \times M_k$ elementos.

El Algoritmo 2 sintetiza el definitivo *algoritmo de fuentes equivalentes potenciadas por gradiente*, incorporando estas últimas modificaciones. La Figura 4.1 muestra un bosquejo de los pasos del algoritmo aplicado a un conjunto de puntos de observación que simulan una muestra sobre el terreno y ubicando una fuente debajo de cada uno de ellos.

- 1 Definir el vector residual $\mathbf{r}_0 = \mathbf{d}^0$
- 2 **para** $k = 1$ a K **hacer**
- 3 Selecionar los pesos ponderados $\tilde{\mathbf{W}}_k$ y residuos $\tilde{\mathbf{r}}_{k-1}$
correspondientes a los puntos de datos que caen dentro de la
 k -ésima ventana
- 4 Calcular la matriz Jacobiana $\tilde{\mathbf{A}}_k$ considerando puntos de datos y
fuentes dentro de la k -ésima ventana
- 5 $\mathbf{B}_k = \tilde{\mathbf{A}}_k \mathbf{S}_k^{-1}$
- 6 $\hat{\mathbf{m}}_k = [\mathbf{B}_k^T \tilde{\mathbf{W}}_k \mathbf{B}_k + \lambda \mathbf{I}]^{-1} \mathbf{B}_k^T \tilde{\mathbf{W}}_k \tilde{\mathbf{r}}_{k-1}$
- 7 $\hat{\mathbf{c}}_k = \mathbf{S}_k^{-1} \hat{\mathbf{m}}_k$
- 8 Calcular \mathbf{d}_k , donde $d_{ki} = \sum_{j=1}^{M_k} \frac{c_{kj}}{\|\mathbf{p}_i - \mathbf{q}_{kj}\|} \quad \forall i = 1 \text{ to } N$
- 9 $\mathbf{r}_k = \mathbf{r}_{k-1} - \mathbf{d}_k$
- 10 **fin para**

Algoritmo 2: Algoritmo de fuentes equivalentes potenciadas por gradiente.

Vale la pena notar que los dos conjuntos de fuentes equivalentes correspondientes a dos ventanas solapadas contiguas poseen una porción de sus fuentes en las mismas ubicaciones, específicamente aquellas que caen en la intersección de ambas ventanas. Esto nos permite interpretar que el algoritmo de potenciación de gradiente ajusta los coeficientes de las fuentes múltiples veces: una vez por cada ventana que cubre cada fuente. Este hecho puede ser explotado con el objetivo de ahorrar memoria: en vez de almacenar todos los vectores \mathbf{c}_k (ec. 4.12), podemos inicializar un único vector \mathbf{c} con ceros, donde cada elemento representa el coeficiente de cada una de las M fuentes originales. Luego de

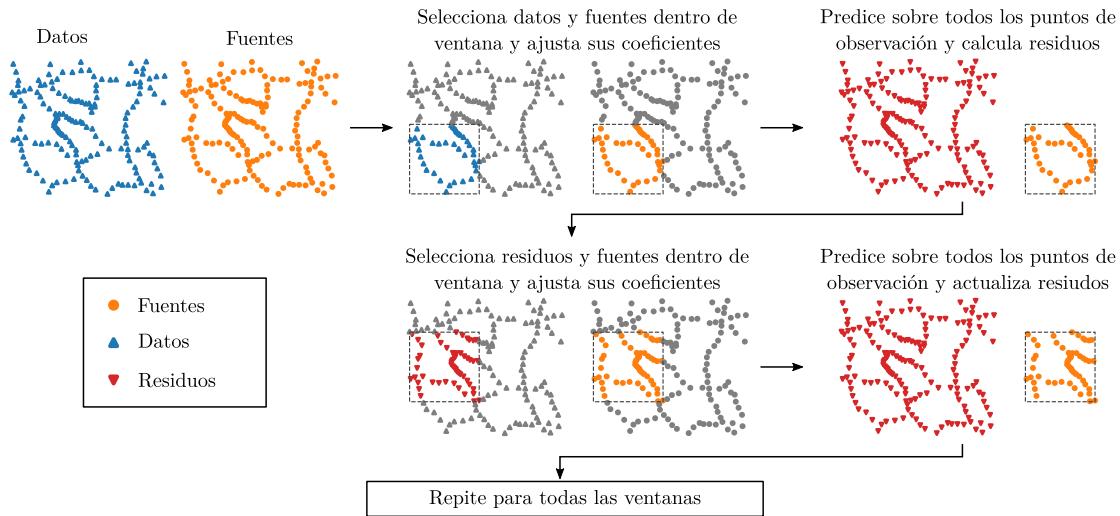


Figura 4.1: Bosquejo del algoritmo de fuentes equivalentes potenciadas por el gradiente. Los puntos de datos están representados por triángulos azules (orientados hacia arriba), las fuentes equivalentes por puntos naranja, los residuos por triángulos rojos (orientados hacia abajo), y los límites de la ventana actual por líneas negras de trazo. El algoritmo comienza seleccionando los datos y las fuentes que caen dentro de la primer ventana y continua estimando los coeficientes de estas fuentes utilizando los puntos de datos seleccionados. Luego, se calcula el efecto de las fuentes ajustadas sobre todos los puntos de observación y usamos los datos para calcular los residuos. Se utiliza otra ventana para seleccionar otro conjunto de residuos y fuentes, se estiman sus coeficientes utilizando los residuos seleccionados en vez de los datos originales. Nuevamente, el efecto de las fuentes estimadas es calculado sobre todos los puntos de observación y se actualizan los residuos. Estos pasos son repetidos para cada ventana en orden aleatorio.

cada iteración del algoritmo de potenciación por gradiente, añadimos los coeficientes estimados \hat{c}_k a sus correspondientes elementos del vector \mathbf{c} . Dado que la función de modelado directo es lineal, podemos calcular el campo resultante mediante la ec. 4.1 en vez de utilizar la ec. 4.13. De esta forma, la cantidad de memoria necesaria para almacenar el conjunto completo de coeficientes se limita a un único vector de M elementos.

Nuestro algoritmo de potenciación del gradiente es similar a la *bootstrap inversion* utilizada por [von Frese et al. \(1988\)](#), la cual también ajusta iterativamente porciones del modelo de fuentes equivalentes a datos residuales. Las diferencias claves entre este y nuestro método son: (i) las fuentes en las porciones de solapamiento de las ventanas son ajustadas más de una vez, permitiendo al algoritmo autocorregirse en caso de obtener soluciones de baja calidad sobre alguna de las ventanas; (ii) utilizamos solo los datos dentro de las ventanas durante el ajuste, lo cual permite la aplicación del algoritmo a conjuntos de datos de mayor

tamaño.

4.2.4. Ubicación de las fuentes

La cantidad ideal de fuentes y sus ubicaciones, tanto horizontales como verticales, han sido debatidas desde la concepción de la técnica de fuentes equivalentes de la mano de [Dampney \(1969\)](#). Las elecciones que se realizan con respecto a estos parámetros juegan un rol importante en la precisión de las predicciones y sobre los recursos computacionales necesarios para estimar los coeficientes de las fuentes. Una distribución ideal de fuentes debería ser simultáneamente capaz de reproducir los datos observados en los puntos de muestreo, realizar predicciones precisas en locaciones no muestreadas y minimizar el costo computacional necesario.

Un gran número de fuentes homogéneamente distribuidas a lo largo y ancho de la zona de estudio son capaces de reproducir los datos observados. Sin embargo, los costos computacionales pueden ser prohibitivos y los problemas subdeterminados que generan son susceptibles a sobreajustar a los datos, resultando en interpolaciones y extrapolaciones de baja calidad. En el otro extremo, la utilización de pocas fuentes reduciría el costo computacional, pero el modelo podría ser incapaz de reproducir el espectro completo de los datos observados.

Las características particulares de cada muestra juegan también un rol importante en la elección de la distribución de las fuentes equivalentes. En el caso de muestras tomadas sobre la superficie terrestre, las observaciones suelen localizarse a lo largo de trayectorias irregulares junto con algunos puntos dispersos. La cobertura de la región de muestreo no es usualmente homogénea, dejando grandes áreas sin ninguna medición. Por otro lado, las observaciones realizadas mediante vehículos aéreos suelen estar localizadas a lo largo de líneas de vuelo casi rectas y con poca separación. Las mediciones se suelen tomar a intervalos muy cortos, produciendo una gran densidad de datos a lo largo de las líneas de vuelo. Esto genera un sesgo en el muestreo, lo cual puede producir artefactos de *aliasing* en las grillas posteriores.

Distribuciones horizontales de las fuentes

Las distribuciones más utilizadas para localizar las fuentes equivalentes horizontalmente son:

1. *Fuentes debajo de datos*: una única fuente equivalente es ubicada en la misma localización horizontal de cada dato (Fig. 4.2b), pero a diferente altitud. Por lo tanto, la cantidad de fuentes es igual a la cantidad de observaciones ($M = N$).
2. *Grilla regular*: una distribución homogénea de fuentes debajo de la zona de estudio (Fig. 4.2c). En la práctica, esto suele generar problemas subdeter-

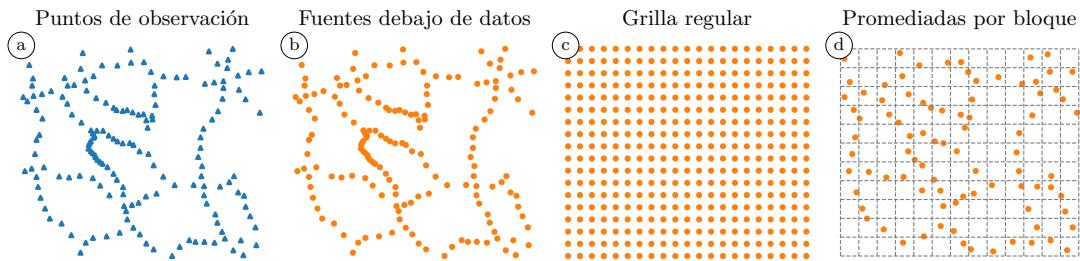


Figura 4.2: Bosquejo de las diferentes distribuciones horizontales para los modelos de fuentes equivalentes. Los puntos azules representan las ubicaciones de las observaciones, y los puntos naranja, las ubicaciones de las fuentes equivalentes según las distintas estrategias de distribución. (a) Conjunto de 166 puntos de observación que emulan una muestra terrestre. (b) Ubicación de 166 fuentes obtenidas mediante la distribución *fuentes debajo de datos*. (c) Ubicación de 378 fuentes obtenidas mediante la distribución *grilla regular*. (d) Ubicación de 87 fuentes obtenidas mediante la distribución *fuentes promediadas por bloque*. Las líneas de a trazos grises representan los bloques espaciales dentro de los cuales la localización media de los datos es calculada.

minados, ya que se requieren mayores cantidades de fuentes que de datos ($M > N$).

En el caso de muestras terrestres, la distribución de *grilla regular* necesita un espaciado entre los nodos lo suficientemente pequeño para poder ser capaz de ajustar a los datos observados. Esto genera una cantidad de fuentes innecesariamente grande en regiones donde no se han realizado observaciones. En contraste, la distribución *fuentes debajo de datos* es más propensa a ajustar de manera precisa a los datos observados haciendo uso de una cantidad menor de fuentes, y por ende reduciendo el costo computacional. Sin embargo, cuando esta distribución es aplicada a muestras aéreas, la distribución *fuentes debajo de datos* ubicará una cantidad indeseablemente grande de fuentes debajo de las líneas de vuelo. Esto puede conllevar la generación de efectos de *aliasing* sobre los valores predichos, como un patrón de rayas paralelas a las líneas de vuelo que suelen observarse en grillas obtenidas de vuelos aeromagnéticos. La distribución de *grilla regular* puede evitar estos efectos al distribuir homogéneamente las fuentes y utilizando una capa de fuentes de densidades continuas (por ejemplo, prismas rectangulares o teseroides).

Aquí proponemos un nuevo tipo de distribución horizontal de fuentes equivalentes que puede simultáneamente reducir el costo computacional y mitigar algunas de las desventajas de las distribuciones existentes. En la distribución de *fuentes promediadas por bloque* (*block-averaged sources* en inglés), cada fuente es ubicada en la posición media de los puntos de datos que caen dentro de un determinado bloque espacial (Fig. 4.2d). Esto se realiza de la siguiente forma:

1. Dividir la región de estudio en bloques rectangulares de mismo tamaño.

2. Calcular la media de las posiciones horizontales de los puntos de observación que caen dentro de cada bloque. Los bloques sin ningún punto de observación son omitidos.
3. Asignar una fuente puntual a cada posición horizontal media calculada en el paso 2.

La cantidad total de fuentes creadas por esta nueva distribución será menor que el número total de observaciones si el tamaño de los bloques es elegido apropiadamente (asegurándose de que los bloques sean lo suficientemente grandes como para contener más de un único punto de observación). El problema sobredeterminado que surge de esta distribución posee un costo computacional inferior y es menos propenso a sobreajustar los datos, ya que la complejidad del modelo es menor. Además, el proceso de *promediado por bloques* puede balancear el espaciado entre fuentes a lo largo de líneas de vuelo y entre líneas contiguas, ayudando a reducir los efectos de *aliasing* en las grillas que se generan. En la Sección 4.3.1, demostramos mediante pruebas sobre datos sintéticos que las *fuentes promediadas por bloque* son capaces de realizar interpolaciones con una precisión comparable a la obtenida por las otras dos distribuciones, pero utilizando una fracción de fuentes equivalentes.

Profundidad de las fuentes

Es ampliamente conocido de la teoría de potencial que la profundidad de una fuente puntual influencia la longitud de onda del campo observado en la superficie. Esto hace que la profundidad de la fuente sea un parámetro clave que afecta el producto de la interpolación y otras operaciones realizadas mediante fuentes equivalentes. En la literatura podemos encontrar múltiples estrategias diferentes para asignar profundidad de las fuentes. Aquí destacaremos las siguientes (Fig. 4.3):

1. *Profundidad constante*: La opción más simple es ubicar todas las fuentes a la misma profundidad (Fig. 4.3a). Si las mediciones fueron obtenidas a altitudes significativamente diversas, algunas mediciones se encontrarán más distantes a las fuentes que otras, lo que podría generar problemas a la hora de reproducir las longitudes de onda más cortas en los puntos de grandes altitudes.
2. *Profundidad relativa*: Las profundidades de las fuentes se determinan al desplazar la coordenada vertical de los puntos de datos hacia abajo por una cantidad constante (Fig. 4.3b). Las fuentes no tendrán todas las mismas coordenadas verticales, pero se encontrarán todas a la misma distancia vertical con respecto a los puntos de observación.
3. *Profundidad variable*: Las profundidades de las fuentes son proporcionales a la distancia horizontal entre primeros vecinos de datos o fuentes

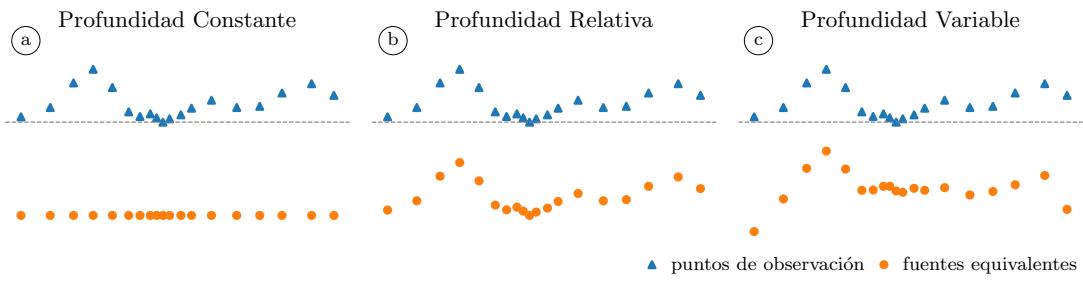


Figura 4.3: Ejemplos de diferentes estrategias para asignar profundidades a las fuentes equivalentes. Asignaremos una única fuente por cada punto de observación, localizándolas en la mismas coordenadas horizontales que los puntos de datos. Las profundidades de las fuentes son (a) una *profundidad constante* a una determinada coordenada vertical, (b) una *profundidad relativa* determinada al desplazar uniformemente la coordenada vertical de los puntos de datos hacia abajo, y (c) una *profundidad variable* determinada por el desplazamiento de la coordenada vertical de los puntos de observación por una cantidad proporcional a la distancia media entre fuentes vecinas. La distancia entre puntos de datos y sus respectivas fuentes (a) depende de la altitud de las observaciones, (b) es constante, y (c) es proporcional a la distancia horizontal de las fuentes. Vale notar cómo las fuentes agrupadas en el centro del perfil (c) se encuentran más someras que sus contrapartes en (b).

(Fig. 4.3c). Diferentes variaciones de esta estrategia han sido propuestas anteriormente, por ejemplo [Cordell \(1992\)](#), [Guspí et al. \(2004\)](#), y [Guspí y Novara \(2009\)](#). Esta estrategia posee la capacidad de preservar el contenido frecuencial de los datos: en caso de que la muestra posea puntos muy agrupados en algunas áreas (formando *clusters* o cúmulos), al ubicar las fuentes debajo de ellas a menores profundidades podremos recuperar las longitudes de onda más cortas presente en esos datos.

Nuestro enfoque sobre la estrategia de *profundidad variable* será:

$$z = z_{obs} + \Delta z + \alpha h, \quad (4.14)$$

donde z es la coordenada vertical (positiva hacia abajo) de una fuente equivalente, Δz es un desplazamiento relativo en profundidad que se aplica de manera uniforme a todas las fuentes, α es un *factor de profundidad* adimensional, h es la distancia horizontal media a las primeras k fuentes vecinas, y z_{obs} es una coordenada vertical relacionada a los puntos de observación que dependerá de la estrategia horizontal utilizada. Para *fuentes debajo de datos*, será la coordenada vertical del punto de observación correspondiente a la fuente. Para una *grilla regular*, puede ser interpolada a partir de la coordenada vertical de todos los puntos de datos. Finalmente, para *fuentes promediadas por bloque* será la coordenada vertical media de todos los puntos de datos que caen dentro del bloque correspondiente a la fuente.

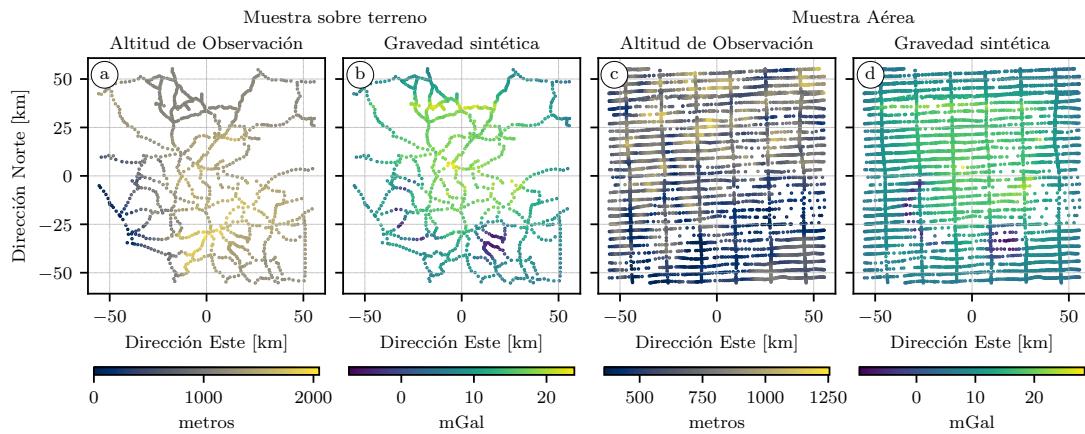


Figura 4.4: Altitudes de los puntos de observación y valores de gravedad para las muestras sintéticas sobre terreno (a-b) y aéreas (c-d). Las altitudes están dadas en metros sobre el plano de altitud cero. Los datos sintéticos de gravedad están contaminados con ruido Gaussiano pseudo-aleatorio con media cero y desviación estándar de 1 mGal.

En la Sección 4.3.1 probamos la efectividad de estas estrategias sobre datos sintéticos.

4.3. Pruebas sobre datos sintéticos

Hemos utilizado un conjunto de datos sintéticos de gravedad para evaluar la precisión de la interpolación entre las diferentes estrategias de distribución horizontal y vertical de fuentes equivalentes, así como también el desempeño del algoritmo de fuentes equivalentes potenciadas por gradiente. Para generar los datos sintéticos hemos creado un modelo de 64 prismas rectangulares, distribuidos a lo largo y ancho de un área de $111319 \text{ m} \times 111319 \text{ m}$ con profundidades variables entre 10000 m y cero. El contraste de densidad de los prismas van desde -900 kg m^{-3} a 500 kg m^{-3} . El modelo incluye prismas de diferentes formas, tamaños y profundidades para crear disturbio de gravedad con una variedad de longitudes de onda.

Hemos creado dos conjuntos de datos sintéticos a partir del mismo modelo, uno que simula un muestreo sobre el terreno y otro una medición aérea (Fig. 4.4). Para crear la muestra sintética sobre el terreno hemos elegido una porción de las observaciones presentes en un conjunto de datos de dominio público sobre Sudáfrica, disponibles a través de NOAA National Centers for Environmental Information (NCEI). Para la muestra aérea, utilizamos una porción de la Great Britain Aeromagnetic Survey adquirida por Hunting Geology and Geophysics Ltd y Canadian Aeroservices Ltd, entre 1955 y 1965, y puesta a disposición pública por el British Geological Survey (BGS). En ambos casos, re-

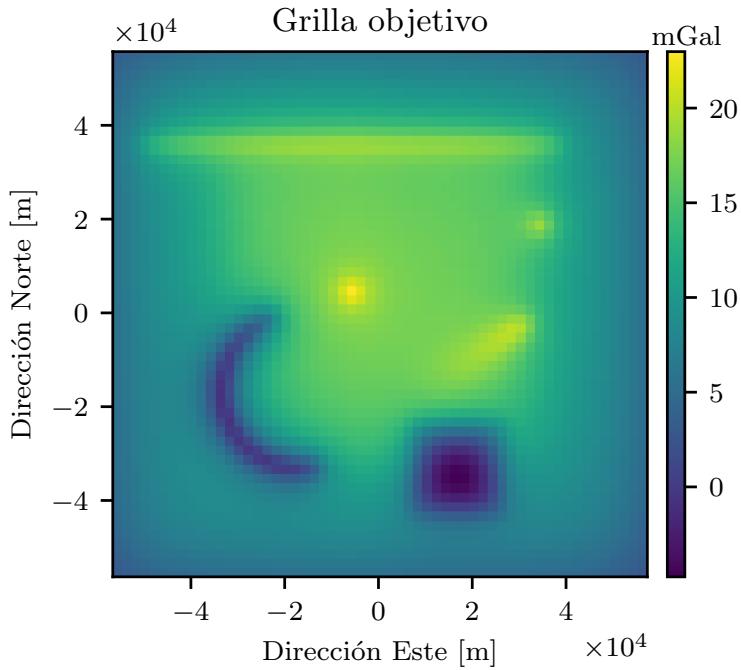


Figura 4.5: Gráfico de la *grilla objetivo* de datos sintéticos de gravedad. La grilla está compuesta por 57×56 puntos con un espaciado de 2 km. Todos los puntos de la grilla se encuentran a una altitud de 2000 m sobre el plano de altitud cero.

escalamos las coordenadas horizontales de cada porción de muestra para cubrir un área de $111319\text{ m} \times 110576\text{ m}$, coincidentes con las dimensiones del modelo sintético. La muestra sintética sobre el terreno contiene 963 observaciones distribuidas en altitudes que van desde 0 m a 2052.2 m (Fig. 4.4a). La muestra aérea posee 5673 observaciones a altitudes entre 359 m y 1255 m (Fig. 4.4c).

Calculamos la componente vertical de la aceleración gravitatoria generada por el modelo sintético utilizando el método de [Nagy et al. \(2000, 2002\)](#) con recientes modificaciones hechas por [Fukushima \(2020\)](#), como se ha implementado en el software de código abierto Harmonica ([Soler et al., 2021b](#)). Hemos generado una *grilla objetivo* de 57×56 puntos con un espaciado de 2 km y localizados a una altitud de 2000 m sobre el plano de altitud cero (Fig. 4.5) que nos servirá de referencia para estimar los errores de las interpolaciones. Luego generamos los datos sintéticos de aceleración de la gravedad sobre la muestra sobre el terreno (Fig. 4.4b) y sobre la muestra aérea (Fig. 4.4d). Además hemos contaminado estos datos sintéticos con ruido Gaussiano pseudo-aleatorio con media cero y con desviación estándar de 1 mGal.

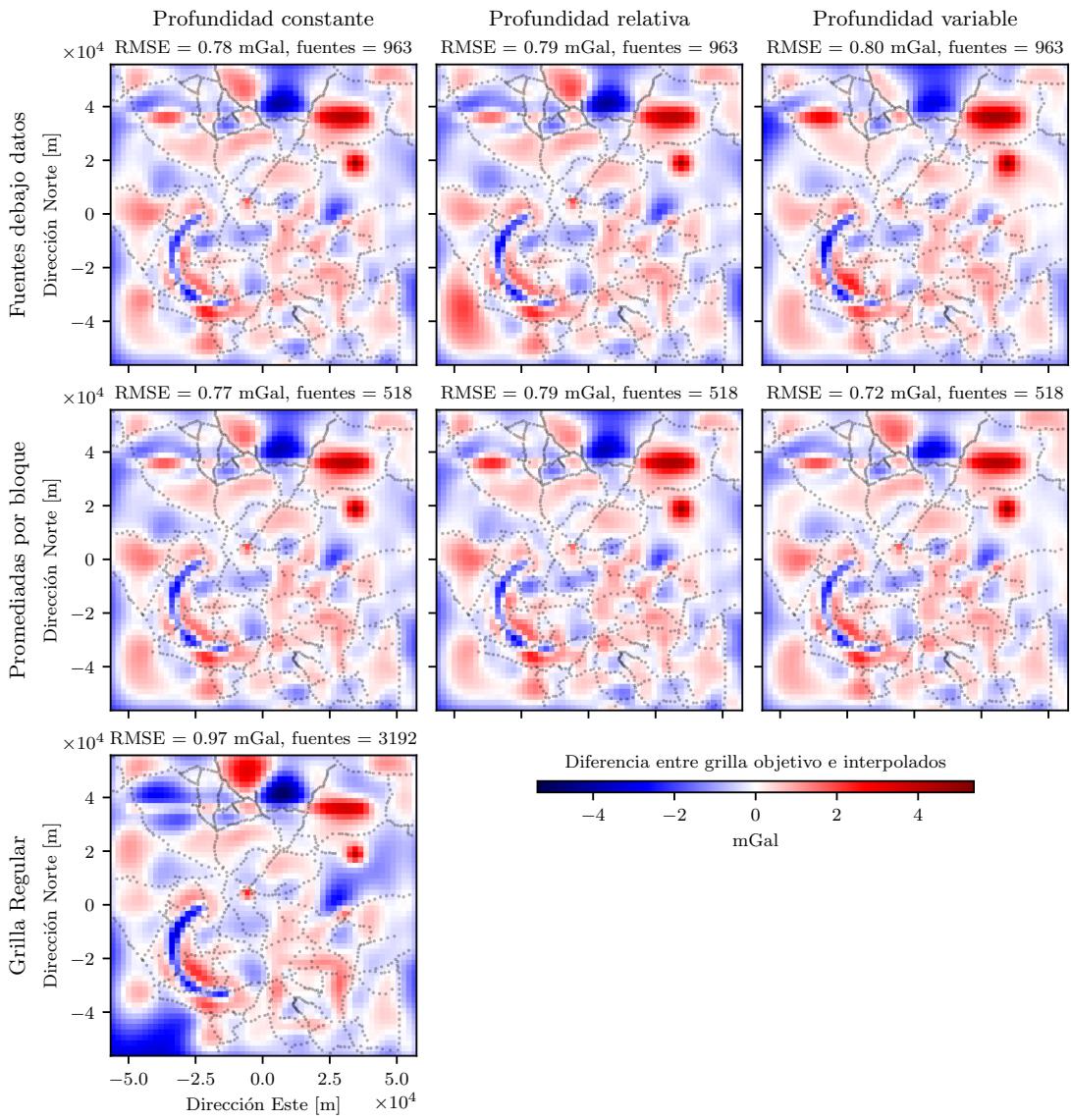


Figura 4.6: Gráficos con las diferencias entre la *grilla objetivo* y los valores interpolados para la muestra sobre el terreno, producidas por cada una de las estrategias de distribución de fuentes. Los puntos negros representan la ubicación horizontal de los puntos correspondientes a los datos sintéticos. La RMSE y el número total de fuentes equivalentes para cada distribución se encuentran reportadas por encima del respectivo gráfico.

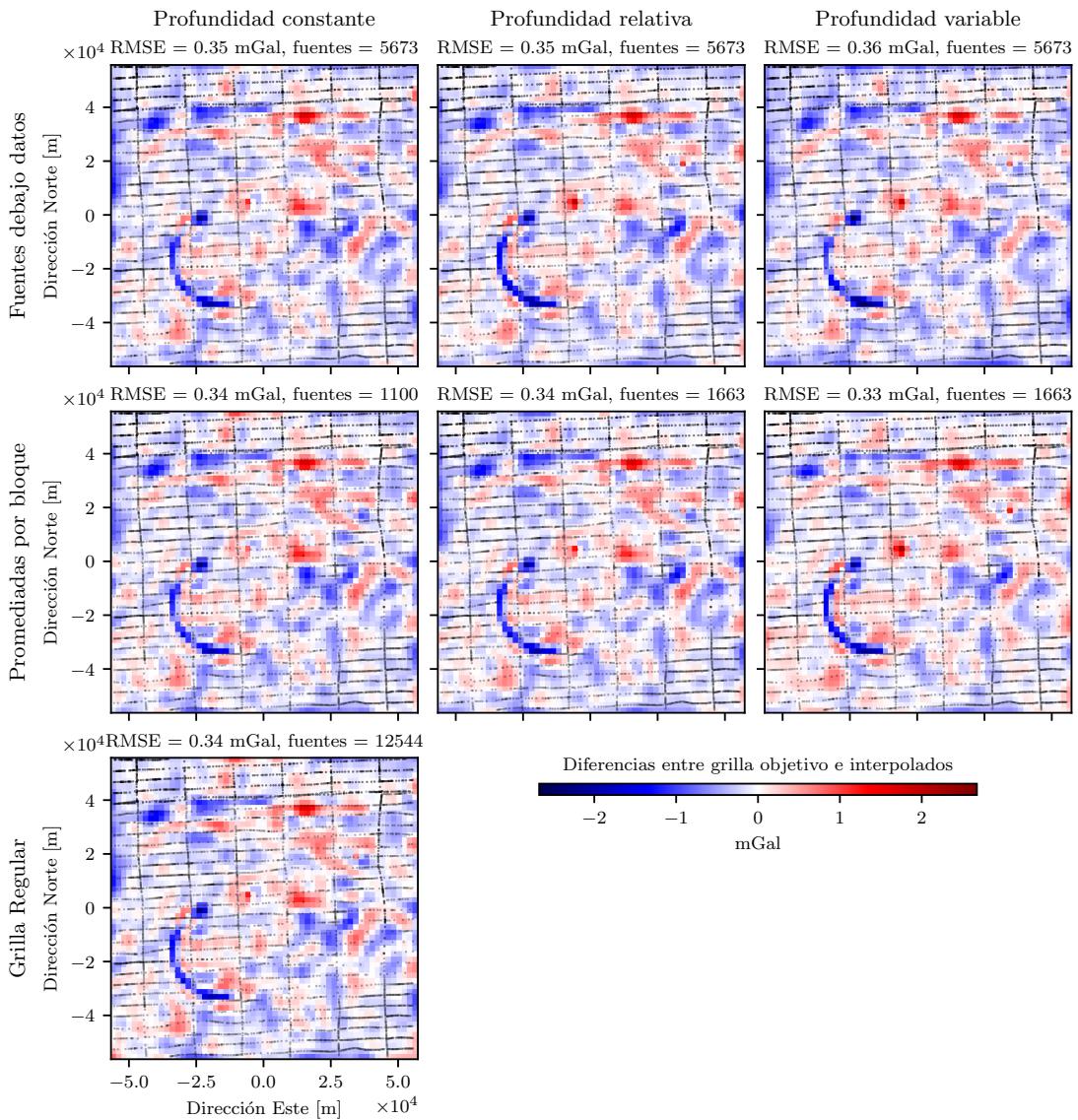


Figura 4.7: Gráficos con las diferencias entre la *grilla objetivo* y los valores interpolados para la muestra aérea, producidas por cada una de las estrategias de distribución de fuentes. Los puntos negros representan la ubicación horizontal de los puntos correspondientes a los datos sintéticos. La **RMSE** y el número total de fuentes equivalentes para cada distribución se encuentran reportadas por encima del respectivo gráfico.

4.3.1. Estrategias de distribución de fuentes

Hemos investigado el efecto sobre la precisión de las interpolaciones de diferentes estrategias para distribuir las fuentes equivalentes, tanto horizontal como verticalmente. Para hacer esto, hemos utilizado las soluciones por mínimos cuadrados amortiguados descriptas en la Sección 4.2.2 (sin potenciación del gradiente) para interpolar los datos sintéticos (Fig. 4.4) y comparamos los resultados con los valores de la *grilla objetivo* (Fig. 4.5). Este proceso es repetido para cada una de las combinaciones de distribución horizontal (*fuentes debajo de datos y fuentes promediadas por bloque*) y tipo de profundidad (*constante, relativa, y variable*), y para fuentes distribuidas en una *grilla regular* con profundidad constante. En total se generan 7 diferentes tipos de combinaciones.

Cada estrategia de distribución de fuentes requiere seleccionar valores para determinados hiperparámetros necesarios para construir, en cada caso, el conjunto de fuentes equivalentes. Por ejemplo, al utilizar una profundidad constante necesitamos definir la profundidad de las fuentes, y si la combinamos con *fuentes promediadas por bloques* se requiere el tamaño de los bloques. Las capacidades predictivas de las fuentes equivalentes depende de la elección de estos hiperparámetros. Para garantizar que nuestras comparaciones sean justas, llevamos a cabo exhaustivas búsquedas sobre las combinaciones de los hiperparámetros (incluyendo el parámetro de *amortiguamiento* de la ec. 4.8) con el objetivo de obtener las mejores predicciones que puedan ser alcanzadas por cada estrategia de distribución de fuentes. La mejor predicción se define como aquella que minimiza la raíz del error cuadrático medio (RMSE) entre los valores interpolados y los valores de la *grilla objetivo* (Fig. 4.5). Los valores para cada uno de los parámetros utilizados en estas búsquedas y aquellos que producen la menor RMSE se exponen en las Tablas 4.1 y 4.2.

Las Figuras 4.6 y 4.7 muestran las diferencias entre la *grilla objetivo* y las mejores predicciones alcanzadas por cada estrategia de distribución de fuentes, tanto para la muestra sobre el terreno como para la muestra aérea, respectivamente. En el caso de la muestra sintética sobre el terreno, las distribuciones horizontales produjeron valores similares de RMSE de aproximadamente 0.8 mGal, independientemente del tipo de profundidad que se utilice; a excepción de las fuentes distribuidas sobre una *grilla regular*, la cual produjo una RMSE mayor de 0.97 mGal. Las diferencias entre la *grilla objetivo* y los valores interpolados son mayores en regiones con una pobre cobertura de datos. Los efectos de borde están presentes en todas las estrategias, pero son notablemente menores para la combinación de *fuentes promediadas por bloque* con profundidad variable. En el caso de la muestra aérea, todas las estrategias (incluyendo la grilla regular) produce valores similares de RMSE de aproximadamente 0.3 mGal. Los gráficos de las diferencias entre la *grilla objetivo* y los valores interpolados son visualmente indistinguibles entre ellos. Vale destacar que para ambos tipos de muestras, cada estrategia produce una cantidad considerablemente diferente de fuentes equivalentes: la *grilla regular* es la que mayor cantidad genera, seguida de las

fuentes *debajo de datos* (produciendo la misma cantidad de fuentes que puntos de observación) y dejando a las *fuentes promediadas por bloque* como la estrategia que genera la menor cantidad de fuentes.

4.3.2. Tamaño de las ventanas y cantidad de solapamiento en potenciación del gradiente

Hemos analizado la relación entre la precisión de las interpolaciones y el tiempo de cómputo del algoritmo de fuentes equivalentes potenciadas por el gradiente en función de dos factores claves: el tamaño de las ventanas y la cantidad de solapamiento entre ventanas contiguas. Las comparaciones fueron realizadas contra la solución obtenida por mínimos cuadrados amortiguados (ec. 4.9, sin potenciación del gradiente) utilizando los datos provistos por la muestra aérea sintética (Figs. 4.4c-d). Para evitar que existan sesgos en los resultados, utilizamos las mismas ubicaciones para las fuentes equivalentes tanto como para las interpolaciones con y sin potenciación del gradiente. Más precisamente, elegimos fuentes promediadas por bloques con un tamaño de bloque de 2000 m × 2000 m y una profundidad relativa de 9000 m.

Tamaño de las ventanas

Las dimensiones de las ventanas controlan el tamaño de las matrices Jacobianas $\tilde{\mathbf{A}}_k$ al limitar la cantidad de puntos de datos y fuentes equivalentes que se utilizan en cada paso del algoritmo de potenciación del gradiente (Alg. 2). Ventanas pequeñas reducen la cantidad total de memoria necesaria para estimar los coeficientes de las fuentes. Sin embargo, estas pueden producir interpolaciones menos precisas al no ser capaces de alcanzar el mínimo global de la función objetivo en la ec. 4.8. El tamaño de la ventana también impacta en el tiempo de cómputo de formas poco intuitivas, ya que ventanas pequeñas generan problemas de mínimos cuadrados más simples, pero también requieren mayor cantidad de iteraciones del algoritmo de potenciación del gradiente.

Hemos calculado la RMSE entre las interpolaciones y la *grilla objetivo* (Fig. 4.5) y registramos los tiempos de cómputo utilizando un valor fijo de solapamiento entre ventanas de 50 % y varios tamaños de ventanas diferentes. Para evitar sesgos introducidos por un determinado ordenamiento aleatorio particular de las ventanas, los cálculos fueron repetidos utilizando diferentes *semillas* para el generador de números pseudo-aleatorios. La Figura 4.8a muestra la RMSE, y la Figura 4.8c el tiempo de cómputo requerido para estimar los coeficientes de las fuentes, ambos en función del tamaño de las ventanas.

Estos resultados muestran que los errores de interpolación obtenidos con potenciación del gradiente son generalmente mayores que el error obtenido por las fuentes equivalentes regulares. Los errores decrecen asintóticamente hasta el ~ 40 % correspondiente para las fuentes equivalentes regulares para ventanas con un área mayor que el ~ 10 % de la región de estudio. Los tiempos de

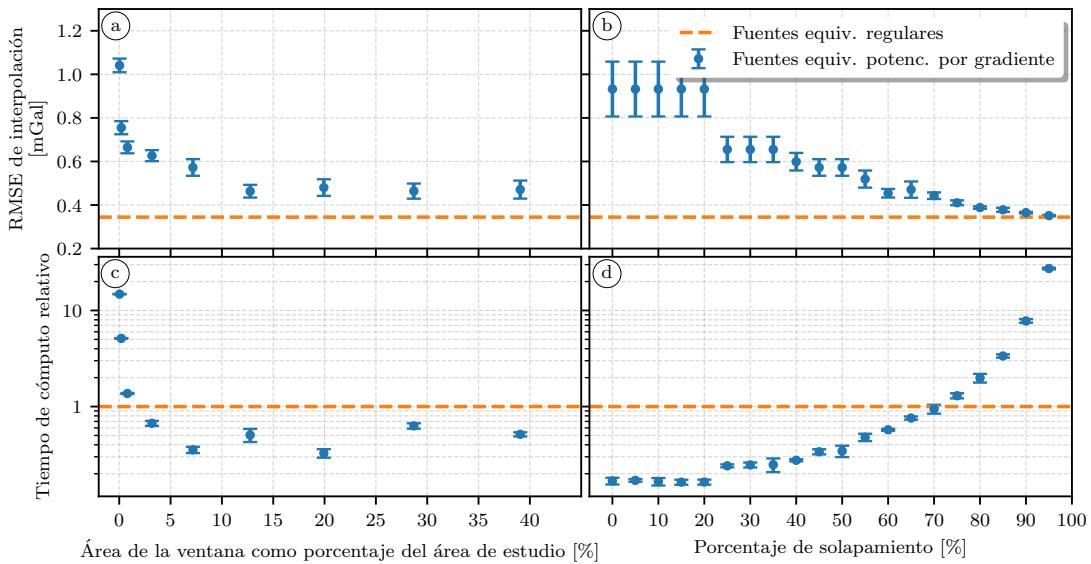


Figura 4.8: RMSE de las interpolaciones (a-b) y los tiempos de cómputos relativos (c-d) para fuentes equivalentes regulares (líneas naranja de a trazos) y para fuentes equivalentes potenciadas por gradiente (puntos azules con barras de error). El solapamiento de las ventanas se muestra como un porcentaje del tamaño de la ventana (un solapamiento del 50 % significa que dos ventanas contiguas comparten un área igual a la mitad del tamaño de la ventana completa). Para las fuentes equivalentes potenciadas por gradiente, la RMSE y los tiempos de cómputo son valores medios (las barras de error corresponden a una desviación estándar) de los resultados obtenidos utilizando diferentes semillas para el generador de números pseudo-aleatorios. Los tiempos de cómputo se muestran como la relación entre el tiempo necesario para estimar los coeficientes de las fuentes equivalentes potenciadas por el gradiente y el necesario para estimar los coeficientes de las fuentes equivalentes regulares.

cómputo decrecen similarmente con el tamaño de la ventana, siendo las fuentes potenciadas por gradiente usualmente más rápidas que las fuentes regulares para ventanas con áreas mayores que el $\sim 5\%$ de la región de estudio. A medida que el tamaño de la ventana aumenta, tanto la RMSE y el tiempo de cómputo parecen estabilizarse cerca de niveles constantes.

Solapamiento de las ventanas

El tamaño del solapamiento entre ventanas contiguas juega un rol importante en el desempeño de las fuentes potenciadas por gradiente. Controla la cantidad de iteraciones y cuántas veces una fuente particular será utilizada en el proceso de ajuste por mínimos cuadrados. Los experimentos realizados en la Sección anterior muestran que un solapamiento de 50 % es suficiente para alcanzar una precisión aceptable en las interpolaciones. Sin embargo, hemos

estudiado de forma independiente cómo la cantidad de solapamiento impacta tanto en la precisión como en el tiempo de cómputo.

Llevamos a cabo un experimento similar al realizado en la Sección 4.3.2, pero esta vez mantuvimos el tamaño de la ventana fijo a 30000 m y variamos la cantidad de solapamiento entre 0 % y 95 % con un paso de 5 %. Todos los otros procedimientos experimentales se mantuvieron intactos. La Figura 4.8b muestra la RMSE y la Figura 4.8d el tiempo de cómputo requerido para estimar los coeficientes de las fuentes, ambos en función de la cantidad de solapamiento entre las ventanas.

Nuestros resultados muestran que la RMSE de la interpolación decrece con la cantidad de solapamiento, alcanzando la misma precisión que las fuentes equivalentes regulares a un solapamiento de aproximadamente 90 %. Por otro lado, el tiempo de cómputo aumenta con la cantidad de solapamiento, haciéndose mayor que el necesario por fuentes equivalentes regulares para solapamientos mayores al 70 %. Este comportamiento era de esperarse, ya que un aumento en el solapamiento aumenta la cantidad de iteraciones en el proceso de potenciación del gradiente, sin reducir el tamaño de los problemas de mínimos cuadrados.

4.3.3. Interpolación con potenciación del gradiente

Finalmente, hemos aplicado las fuentes equivalentes potenciadas por gradiente para interpolar la muestra sintética aérea (Fig. 4.4). Tal y como hemos realizado anteriormente, utilizamos las fuentes promediadas por bloque con un tamaño de bloque de 2 km × 2 km. Basado en los resultados de la Sección 4.3.2, adoptamos un solapamiento entre ventanas contiguas de 50 % y un tamaño de ventanas de 20 km.

Hemos estimado la profundidad relativa de las fuentes y el parámetro de *amortiguamiento* comparando las predicciones con los valores de la *grilla objetivo*. Durante la búsqueda, exploramos valores de profundidad relativa entre 1000 m y 19000 m y de valores de *amortiguamiento* entre 10^{-6} y 10 de a pasos de un orden magnitud. Las predicciones más precisas alcanzaron una RMSE de 0.38 mGal con una profundidad relativa de 3000 m y un *amortiguamiento* de 0.1. Vale la pena notar que la RMSE alcanzada por las fuentes potenciadas por gradiente es comparable con los valores obtenidos por las fuentes equivalentes regulares en la Sección 4.3.1. Para resaltar la importancia de aleatorizar el orden en el que se recorren las ventanas durante la potenciación del gradiente, llevamos a cabo la misma interpolación una vez más haciendo uso de los mismos valores para la profundidad relativa y el *amortiguamiento*, pero esta vez iteramos sobre las ventanas en orden secuencial (de Sur a Norte, de Oeste a Este).

Las Figuras 4.9a-b muestran las diferencias entre la *grilla objetivo* y los resultados de la interpolación obtenida con ventanas en orden aleatorio y en orden secuencial, respectivamente. Las diferencias para las ventanas en orden aleatorio se asemejan a las obtenidas por las fuentes equivalentes regulares que vimos

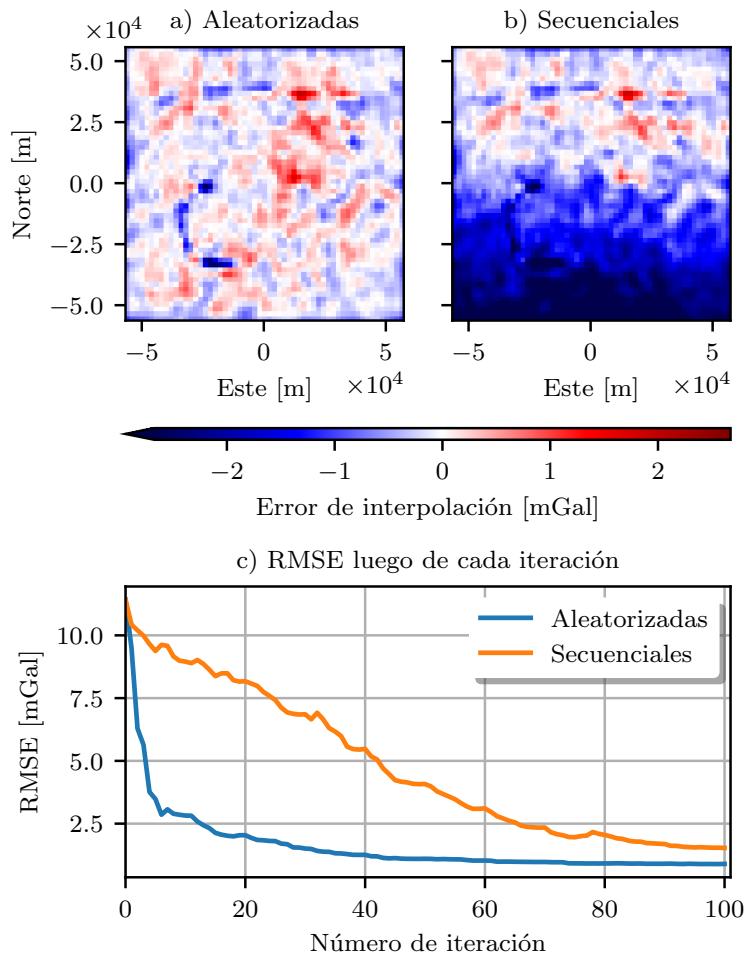


Figura 4.9: Error de la interpolación obtenido a partir de fuentes equivalentes potenciadas por gradiente utilizando un (a) orden aleatorio y un (b) orden secuencial para recorrer las ventanas. (a y b) Mapas de las diferencias entre la *grilla objetivo* y los resultados de la interpolación de la muestra sintética aérea. La escala de colores fue truncada al mismo rango de valores que la de la Figura 4.7. (c) raíz del error cuadrático medio luego de cada iteración del algoritmo de potenciación del gradiente.

en las Figuras 4.6 y 4.7. Por otro lado, las diferencias para las ventanas en orden secuencial muestran una clara tendencia de grandes diferencias negativas hacia el Sur, decreciendo hacia el Norte. Esta tendencia se correlaciona con el orden en el que las ventanas son ejecutadas, con diferencias que van decreciendo en valor absoluto hacia el final del algoritmo. La Figura 4.9c muestra la RMSE del proceso de ajuste luego de cada iteración, tanto para ventanas ejecutadas en orden aleatorio como secuencial, indicando claramente que las ventanas aleatorias generan una convergencia más rápida del algoritmo.

4.4. Grillado de datos gravimétricos de Australia

En esta Sección demostraremos cómo podemos utilizar las fuentes equivalentes potenciadas por gradiente para interpolar grandes cantidades de datos sobre una grilla regular a una altitud uniforme. Para esto, hemos seleccionado una compilación de acceso abierto de muestras de datos de aceleración de gravedad sobre Australia realizada por [Wynne \(2018\)](#) y filtradas y referenciadas al elipsoide WGS84 por [Uieda \(2021\)](#). Contiene 1.7 millones de datos y cubre la mayor parte del territorio australiano. Nuestro objetivo es crear una grilla regular de disturbios de gravedad a una altitud geométrica constante de 2127.58 m (la altitud de observación más alta presente en los datos).

Hemos calculado el disturbio de gravedad quitando la gravedad normal generada por el elipsoide WGS84 sobre los datos de gravedad observados (Fig. 4.10). Los valores de gravedad normal fueron calculados en cada punto de observación mediante la fórmula cerrada de [Li y Götze \(2001b\)](#) (ver Sección 2.3.1) implementada en la librería Boule ([Uieda y Soler, 2020b](#)). Finalmente, transformamos las coordenadas de los puntos de observación a un sistema de coordenadas planas aplicando una proyección de Mercator.

Comenzamos el proceso de interpolación definiendo un conjunto de *fuentes promediadas por bloque* haciendo uso de bloques de 1.8 km de lado, obteniendo un total de 796744 fuentes puntuales. El tamaño de los bloques fue elegido para que concuerde con la resolución deseada para la grilla final (1 minuto de arco equivale aproximadamente a 1.8 km en el ecuador). Basado en los resultados obtenidos en la Sección 4.3.1, elegimos localizar las fuentes siguiendo la estrategia de *profundidad relativa*. El solapamiento de las ventanas fue fijado nuevamente a 50 %. Para determinar el tamaño de las ventanas, calculamos la cantidad de memoria necesaria para almacenar la matriz Jacobiana más grande que genera el algoritmo para diferentes valores de tamaño de ventana (Fig. 4.11a). Hemos seleccionado un tamaño de ventana de 225 km con el objetivo de limitar la cantidad de memoria por debajo de los 16 Gigabytes.

Hemos determinado la profundidad relativa de las fuentes y el parámetro de *amortiguamiento* aplicando una validación cruzada de K iteraciones (*K-Fold cross-validation* en inglés) mediante la librería scikit-learn ([Pedregosa et al., 2011](#)). Este método divide aleatoriamente los datos originales en k conjuntos (denominados *folds*), ajusta el modelo de fuentes equivalentes utilizando solo los datos de $k - 1$ conjuntos y valida el modelo comparando sus predicciones sobre los puntos pertenecientes al conjunto restante. Este proceso se lleva a cabo una vez por cada uno de los k conjuntos, obteniendo una media de las **RMSE** para dicho modelo. Con el objetivo de acelerar el cómputo, realizamos la validación cruzada sobre un subconjunto de los datos originales correspondientes a un área de $300 \text{ km} \times 300 \text{ km}$ que contiene 14934 puntos. Llevamos a cabo la validación cruzada repetidamente para combinaciones de la profundidad relativa con valores entre 1000 m y 15000 m, y el parámetro de *amortiguamiento* con valores entre 0.01 y 10000 con pasos de un orden de magnitud. La Figura 4.11c mues-

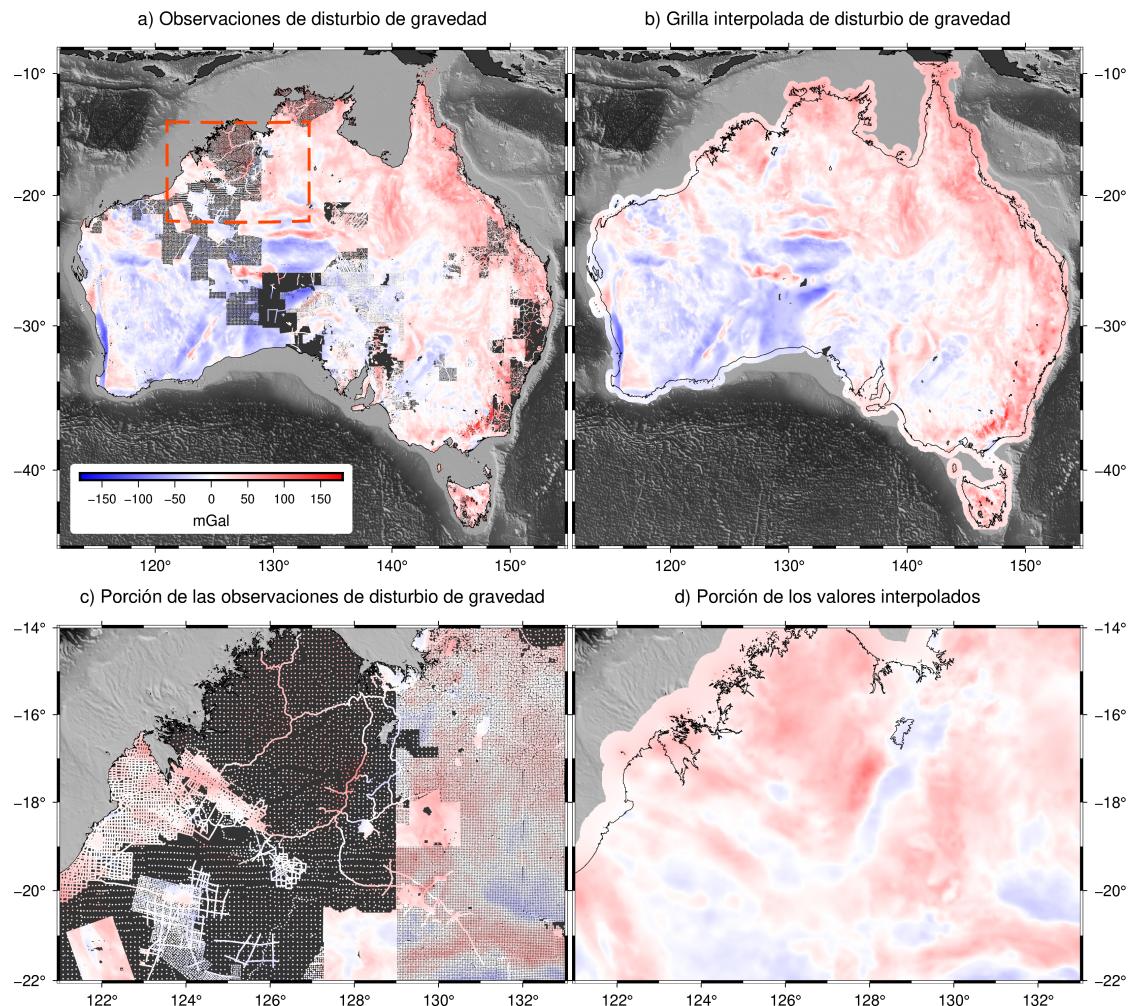


Figura 4.10: Mapas de los valores observados (a y c) y valores interpolados (b y d) de disturbio de gravedad sobre Australia. Los valores observados en a y c se muestran como círculos coloreados. El rectángulo rojo con línea de trazos señala la zona representada en los mapas c y d. Las observaciones son una porción de una compilación de [Wynne \(2018\)](#) compuesta por 1.7 millones de mediciones de aceleración de la gravedad sobre el terreno. Los valores interpolados fueron obtenidos mediante la aplicación de fuentes equivalentes potenciadas por gradiente y calculados sobre una grilla regular a una altitud de 2127.58 m sobre la superficie del elipsoide WGS84.

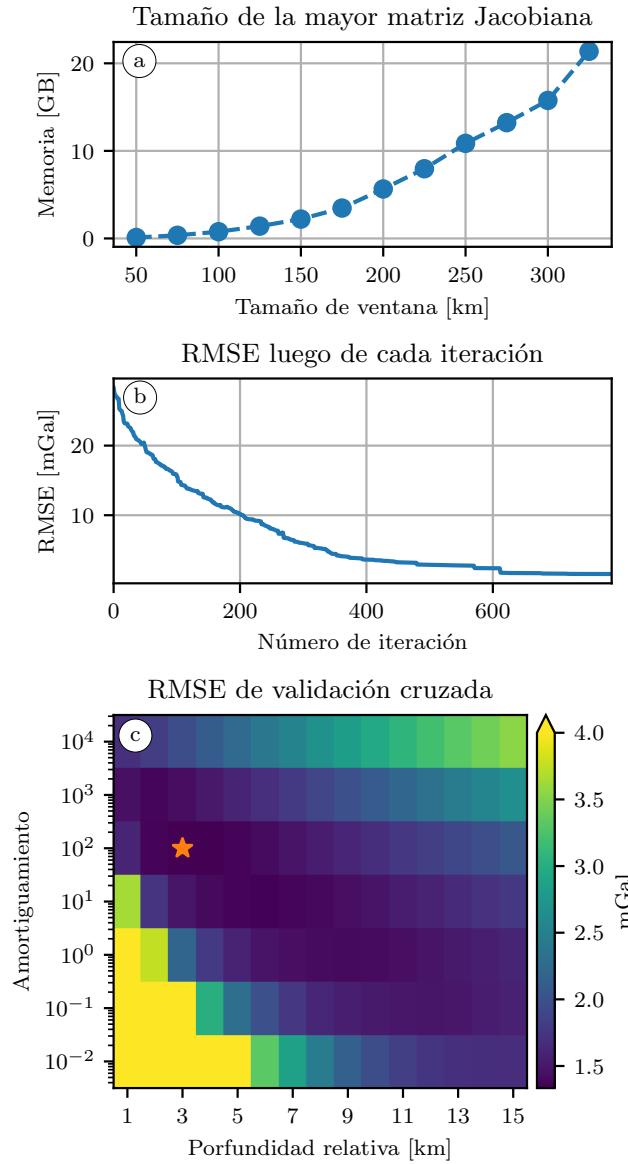


Figura 4.11: (a) Cantidad de memoria computacional necesaria para almacenar la mayor matriz Jacobiana en función del tamaño de las ventanas. Nuestra implementación de algoritmo almacena los elementos de la matriz como reales de doble precisión (64 bits). (b) RMSE entre los valores observados y los valores predichos sobre cada punto de observación luego de cada iteración del algoritmo de potenciación del gradiente. (c) RMSE obtenidas de la validación cruzada de K iteraciones para pares de valores de *amortiguamiento* y profundidad relativa de las fuentes. La estrella naranja señala la RMSE mínima.

tra los valores de **RMSE** obtenidas a partir de la validación cruzada y señala su mínimo valor alcanzado de 1.33 mGal, el cual corresponde a una profundidad relativa de 3000 m y un *amortiguamiento* de 100.

Finalmente, procedemos a estimar los coeficientes de las fuentes utilizando el conjunto completo de datos y los parámetros determinados con anterioridad. Los coeficientes de las fuentes fueron utilizados luego para predecir valores del disturbio de gravedad sobre una grilla regular de 2442×2085 puntos a una altitud de 2127.58 m por encima del elipsoide. En una computadora de modestas especificaciones con 16 núcleos y 16 Gigabytes de RAM, la estimación de los coeficientes de las 796744 fuentes mediante el algoritmo de potenciación del gradiente tomó aproximadamente 1.3 horas, y la predicción sobre los puntos de la grilla, 18 minutos.

La Figura 4.10 muestra la distribución de datos originales y la grilla de valores interpolados. Los puntos de la grilla que se encuentran más allá de 50 km del punto de datos más cercano fueron enmascarados para evitar extrapolaciones irreales. La Figura 4.11b muestra la **RMSE** entre los datos observados y las predicciones del modelo luego de cada iteración del algoritmo. La Figura 4.12 muestra un mapa de los residuos, es decir, la diferencia entre los valores observados y predichos del disturbio de gravedad. La pequeña gráfica insertada muestra un histograma de los residuos, los cuales se encuentran normalmente distribuidos alrededor de cero.

4.5. Discusión

4.5.1. Ubicación de las fuentes

Los resultados de nuestras pruebas sobre datos sintéticos (Figs. 4.6 y 4.7) muestran que no hay diferencias significativas sobre la precisión de la interpolación entre las diferentes estrategias de distribución de las fuentes, tanto en términos de la **RMSE** así como también a partir de inspecciones visuales de los mapas de diferencias. Por lo tanto podemos concluir que todas las estrategias de distribución de las fuentes son capaces de producir interpolaciones de calidad comparable. Sin embargo, las *fuentes promediadas por bloque* hacen uso de menor cantidad de fuentes si las comparamos con las otras estrategias, lo que reduce el costo computacional involucrado en la estimación de sus parámetros y en el modelado directo. Para garantizar que las interpolaciones sean capaces de reproducir las altas frecuencias presentes en los datos, el tamaño de los bloques utilizados durante el promediado debe ser elegido de forma tal que se corresponda con la resolución deseada en la grilla resultante.

La elección de la profundidad de las fuentes no aparenta tener un impacto significativo sobre el error de interpolación. En el caso particular de una muestra dispersa sobre terreno con *fuentes promediadas por bloque*, el uso de profundidades variables reduce visiblemente los efectos de borde y la presencia de

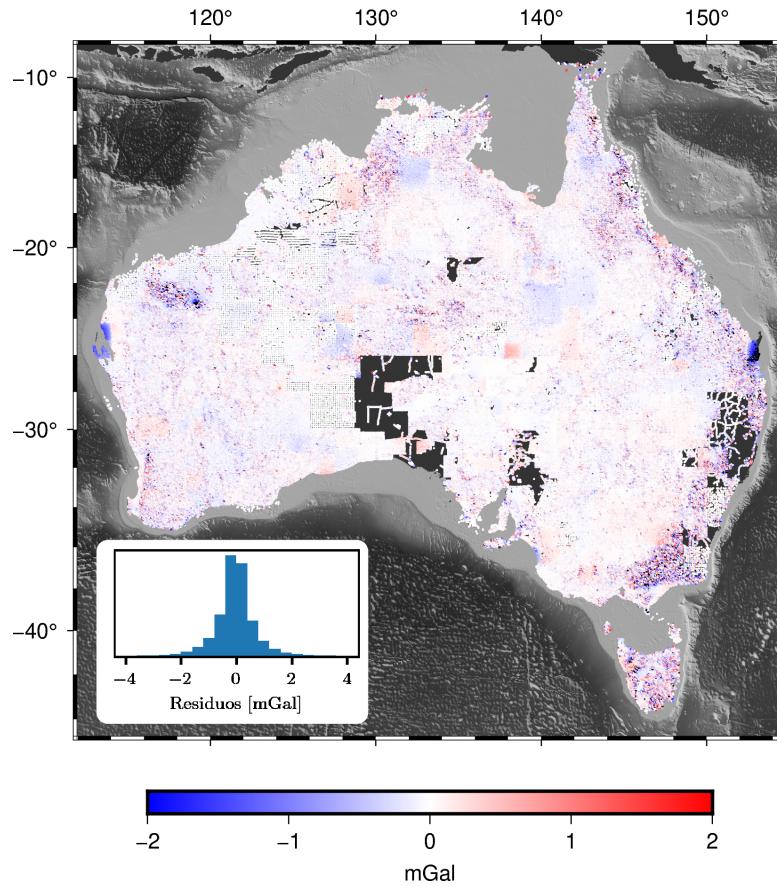


Figura 4.12: Residuos. Diferencias entre el disturbio de gravedad observado sobre Australia y los valores predichos por las fuentes equivalentes estimadas sobre los mismos puntos de observación. La escala de colores ha sido truncada para mejorar la visualización alrededor de la porción mayoritaria de los valores de residuos. El gráfico insertado muestra un histograma de los residuos.

artefactos en áreas con poca cobertura. A primera vista, la elección de una estrategia particular para elegir la profundidad de las fuentes no parecería tener impacto sobre el tiempo de cómputo. Sin embargo, la búsqueda del conjunto de hiperparámetros que produce la interpolación más precisa (por ejemplo, mediante validación cruzada), uno debe resolver el problema inverso una vez por cada combinación posible de los parámetros. Una estrategia como la de *profundidad variable* requiere una mayor cantidad de parámetros a ser determinados (el desplazamiento relativo Δz , el factor de profundidad α , y la cantidad de vecinos cercanos k presentes en la ec. 4.14) en comparación con las otras que solo requieren uno solo. La presencia de un mayor número de parámetros significa incrementar las dimensiones del espacio de parámetros y por ende, aumentar la cantidad de posibles combinaciones. Por lo tanto, recomendamos utilizar una *profundidad constante* o una *profundidad relativa* al procesar grandes cantidades de datos con el objetivo de minimizar el tiempo de cómputo.

4.5.2. Potenciación del gradiente

A partir de las Figuras 4.8a y 4.8c, podemos ver que las fuentes equivalentes potenciadas por gradiente producen interpolaciones ligeramente menos precisas que las fuentes equivalentes regulares, pero son capaces de alcanzar menores tiempos de cómputo. La reducción en la precisión puede ser debida a que el algoritmo de potenciación del gradiente no consigue converger al mínimo global de la función objetivo. A medida que las ventanas aumentan en tamaño, los errores de interpolación decrecen debido a que mayor cantidad de puntos son incluidos dentro del ajuste por mínimos cuadrados. Al mismo tiempo, el proceso de ajuste se vuelve más rápido debido a la reducción en la cantidad de iteraciones. Nuestros resultados indican que es deseable maximizar el tamaño de las ventanas, lo cual puede hacerse mientras las matrices Jacobianas involucradas puedan ser almacenadas en la memoria disponible.

Los resultados mostrados en las Figuras 4.8b y 4.8d indican que valores de solapamiento entre 40 % y 70 % logran un balance entre la precisión de la interpolación y el tiempo de cómputo. Esto corrobora nuestra elección inicial de solapamiento de 50 %, la cual es suficientemente bueno para producir interpolaciones precisas en tiempos razonables.

Finalmente, los resultados expuestos en la Figura 4.9 señalan la importancia de aleatorizar el orden en el que las ventanas son iteradas. Corriendo el algoritmo de potenciación del gradiente secuencialmente se producen predicciones significativamente menos precisas y se alcanza una menor velocidad de convergencia del método.

4.5.3. Datos gravimétricos sobre Australia

La aplicación de las fuentes equivalentes potenciadas por gradiente a los datos de aceleración de gravedad sobre Australia demuestra que el método es capaz de interpolar y realizar continuaciones ascendentes sobre grandes cantidades de datos en tiempos razonables utilizando recursos computacionales modestos. La grilla resultante (Fig. 4.10) preserva la alta resolución de los datos originales mientras evita artefactos de *aliasing* debido al promediado de las ubicaciones de las fuentes. Algunas porciones de la grilla se observan más suaves y con menores amplitudes que los datos originales (por ejemplo, porciones ubicadas al sudoeste), las cuales son esperables luego de aplicar continuaciones ascendentes. A partir del análisis de validación cruzada sobre una porción de los datos, estimamos que el error de las interpolaciones es de aproximadamente 1.33 mGal.

Los residuos más altos que se aprecian en la Figura 4.12 están ubicados en regiones en las cuales los datos presentan señales de alta amplitud y longitud de onda corta. Este comportamiento es esperable, ya que el método involucra algún grado de suavizado debido al uso del término de *amortiguamiento* y de la elección de la profundidad de las fuentes. Existen además residuos cuyas seña-

les poseen bajas amplitudes y largas longitudes de onda que parecen coincidir con las áreas ocupadas por algunas de las ventanas del algoritmo de potenciación del gradiente. Una posible causa de estos artefactos es la inhabilidad de las fuentes equivalentes dentro de ventanas particulares de ajustar adecuadamente las componentes de largas longitudes de onda. Vale notar, sin embargo, que estos residuos de larga longitud de onda poseen amplitudes menores a 1 mGal y no representan una fuente de error significativa.

Los valles alargados alrededor del valor mínimo de la RMSE obtenida durante la validación cruzada (Fig. 4.11c) muestran que existe cierta ambigüedad en la elección del parámetro de *amortiguamiento* y la profundidad de las fuentes. Uno podría elegir un valor alto de *amortiguamiento* junto a fuentes someras o bien valores pequeños de *amortiguamiento* junto a fuentes profundas y alcanzar aproximadamente los mismos resultados. Esto es esperable, ya que ambos parámetros controlan la suavidad de la interpolación.

4.6. Conclusiones

La técnica de fuentes equivalentes ha probado ser altamente adecuada para interpolar disturbios de gravedad y anomalías magnéticas. Las dos razones principales que la hacen sobresalir por encima de otros interpoladores bidimensionales es el hecho que las fuentes equivalentes toman en cuenta la altitud de las observaciones y que los valores interpolados representan siempre funciones armónicas. El principal desafío a la hora de utilizar fuentes equivalentes en la práctica es el alto costo computacional que surge al estimar los coeficientes de las fuentes, especialmente los requerimientos en memoria para almacenar la matriz Jacobiana.

Presentamos dos estrategias que pueden ser simultáneamente aplicadas a la interpolación de conjuntos de datos con millones de puntos en cualquier hardware modesto: las *fuentes promediadas por bloque*, que reducen el número de fuentes necesarias para la interpolación, y las *fuentes equivalentes potenciadas por gradiente*, que dividen el problema inverso en conjuntos más pequeños definidos por ventanas solapadas. Ambos métodos fueron probados sobre datos sintéticos con el objetivo de comparar su precisión y cómo se desenvuelven en términos de eficiencia computacional.

Nuestros resultados muestran que las *fuentes promediadas por bloque* reducen el costo computacional necesario para estimar los coeficientes de las fuentes en comparación con dos estrategias clásicas (ubicar fuentes debajo de cada punto de observación o en grillas regulares). Mostramos además que esta reducción del número de fuentes no afecta la precisión de las predicciones. El uso de *fuentes promediadas por bloque* podría también ayudar a prevenir efectos de *aliasing* en los valores interpolados, especialmente cuando las observaciones se encuentran anisotrópicamente distribuidas (por ejemplo en muestreos aéreos o navales). Se debe prestar especial atención durante la elección del tamaño de los bloques.

A modo de *regla del pulgar*, recomendamos elegir un tamaño aproximadamente igual al de la resolución deseada para grilla regular sobre la cual los valores serán interpolados.

Las pruebas realizadas para comparar las estrategias de selección de profundidades de las fuentes muestran que cualquiera de las aquí sugeridas (*profundidad constante*, *profundidad relativa* y *profundidad variable*) producen interpolaciones con precisiones comparables. Sin embargo, nos vemos inclinados a recomendar el uso de *profundidad constante* o *profundidad relativa* para la mayoría de las aplicaciones, ya que involucran menor cantidad de hiperparámetros que serán necesarios configurar antes de realizar la interpolación.

Las fuentes equivalentes potenciadas por gradiente reducen significativamente la memoria necesaria para estimar los coeficientes de las fuentes, permitiendo interpolar grandes conjuntos de datos con millones de puntos, que de otra forma producirían matrices Jacobianas imposibles de almacenar en las cantidades de memoria disponibles en computadoras personales modernas. Las interpolaciones obtenidas mediante este nuevo método alcanzan aproximadamente la misma precisión que las fuentes equivalentes regulares, mientras reducen el tiempo de cómputo por aproximadamente un factor de 3. Mostramos además que un solapamiento de 50 % entre ventanas contiguas alcanza un buen compromiso entre precisión y tiempo de cómputo. El tamaño de las ventanas debe ser elegido como el máximo valor posible que cree matrices Jacobianas lo suficientemente pequeñas que puedan ser almacenadas en la memoria disponible. Además, aleatorizar el orden en el que las ventanas son iteradas aumenta la velocidad de convergencia del algoritmo y resulta esencial para producir predicciones precisas.

El método de potenciación del gradiente puede ser utilizado en conjunción con cualquier distribución horizontal de las fuentes, estrategia para elegir sus profundidades o tipo de fuente (por ejemplo, fuentes puntuales, prismas, tesseloides, etc), ya que no depende de ninguna suposición acerca de las fuentes. Futuros trabajos podrían dedicarse a investigar la aplicación del algoritmo de potenciación del gradiente a otros métodos de fuentes equivalentes.

4.7. Disponibilidad de datos y código

El código fuente en Python utilizado para producir todos los resultados y figuras presentados en este trabajo se encuentra disponible en [Soler y Uieda \(2021b\)](#) y <https://github.com/compgeolab/eql-gradient-boosted> bajo la licencia de código abierto BSD 3-clause.

La implementación de las fuentes equivalentes potenciadas por gradiente se encuentra basada en la implementación de fuentes equivalentes presente en la librería Harmonica ([Soler et al., 2021b](#)). Entre otros software utilizados en este estudio se incluyen: Pooch ([Uieda et al., 2020a](#)) para descargar y almacenar conjuntos de datos, Verde ([Uieda, 2018](#)) para reducciones por bloques y manipula-

ción de coordenadas, Boule ([Uieda y Soler, 2020b](#)) para el cálculo de gravedad normal, Xarray ([Hoyer y Hamman, 2017](#)) y NumPy ([Harris et al., 2020](#)) para el manejo de arreglos multidimensionales y para cálculos numéricos, Numba ([Lam et al., 2015](#)) para compilaciones en tiempo de ejecución y paralelizaciones, scikit-learn ([Pedregosa et al., 2011](#)) para validaciones cruzadas, Matplotlib ([Hunter, 2007](#)) y PyGMT ([Uieda et al., 2020b](#)) para la generación de figuras y mapas, y el ambiente de programación Jupyter ([Kluyver et al., 2016](#)). Harmonica, Boule, Pooch, y Verde son partes del proyecto Fatiando a Terra ([Uieda et al., 2013](#)).

Todos los datos utilizados son de acceso abierto y disponibles públicamente. Las muestras sintéticas fueron generadas utilizando un conjunto de datos de gravedad sobre Sudáfrica de dominio público, distribuidos por el NOAA NCEI (<https://www.ngdc.noaa.gov/mgg/gravity/gravity.html>), y la Great Britain Aeromagnetic Survey distribuida por la British Geological Survey (BGS) bajo una Open Government License (<https://www.bgs.ac.uk/products/geophysics/aeromagneticRegional.html>). El relieve sombreado presente en la Figura 4.10 corresponde al SRTM15+ de [Tozer et al. \(2019\)](#). Los datos de gravedad sobre Australia están basados en una compilación distribuida por Geoscience Australia bajo una Creative Commons Attribution 4.0 International Licence ([Wynne, 2018](#)), los cuales fueron filtrados y referenciados al elipsode WGS84 por [Uieda \(2021\)](#), que a su vez se distribuye bajo la misma licencia (<https://doi.org/10.6084/m9.figshare.13643837>).

4.A. Parámetros para localizar fuentes equivalentes en pruebas con datos sintéticos

Las Tablas 4.1 y 4.2 muestran valores de parámetros utilizados para analizar estrategias de distribución de fuentes equivalentes con datos sintéticos (Sección 4.3.1), junto con los valores óptimos correspondientes a cada estrategia. Estos valores óptimos fueron utilizados para producir los resultados de las Figuras 4.6 y 4.7.

Tabla 4.1: Parámetros utilizados para producir cada distribución de fuentes durante la interpolación de datos sintéticos sobre terreno. Contiene además el conjunto de parámetros que genera el menor RMSE para cada estrategia de distribución de fuentes y su correspondiente valor.

Distribución	Profundidad	Parámetros	Valores	Óptimo	RMSE
Debajo de datos	Constante	Profundidad (m)	1000 a 17000, con paso de 2000	7000	0.78
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-1}	
	Relativa	Profundidad (m)	1000 a 17000, con paso de 2000	9000	0.79
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-1}	
	Variable	Profundidad (m)	0 a 1400, con paso de 200	1000	0.80
		Factor de profundidad	0.1, 0.5, 1, 2, 3, 4, 5 y 6	1	
		k vecinos cercanos	1, 5, 10 y 15	15	
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	1	
Prom. por bloque	Constante	Profundidad (m)	1000 a 17000, con paso de 2000	7000	0.77
		Tamaño de bloque (m)	1000, 2000, 3000 y 4000	3000	
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-1}	
	Relativa	Profundidad (m)	1000 a 17000, con paso de 2000	7000	0.79
		Tamaño de bloque (m)	1000, 2000, 3000 y 4000	3000	
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-1}	
	Variable	Profundidad (m)	0 a 1400, con paso de 200	600	0.72
		Factor de profundidad	0.1, 0.5, 1, 2, 3, 4, 5 y 6	1	
		k vecinos cercanos	1, 5, 10 y 15	15	
		Tamaño de bloque (m)	1000, 2000, 3000 y 4000	3000	
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-1}	
Grilla regular	Constante	Profundidad (m)	1000 a 9000, con paso de 2000	3000	0.97
		Espaciado de grilla (m)	1000, 2000, 3000 y 4000	2000	
		Amortiguamiento	$10^1, 10^2, 10^3$ y 10^4	10^2	

Tabla 4.2: Parámetros utilizados para producir cada distribución de fuentes durante la interpolación de datos sintéticos aéreos. Contiene además el conjunto de parámetros que genera el menor RMSE para cada estrategia de distribución de fuentes y su correspondiente valor.

Distribución	Profundidad	Parámetros	Valores	Óptimo	RMSE
Debajo de datos	Constante	Profundidad (m)	1000 a 17000, con paso de 2000	7000	0.35
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-2}	
	Relativa	Profundidad (m)	1000 a 17000, con paso de 2000	9000	0.35
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-2}	
	Variable	Profundidad (m)	50 a 1450, con paso de 200	1450	0.36
		Factor de profundidad	1 a 6, con paso de 1	1	
		k vecinos cercanos	1, 5, 10 y 15	15	
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	1	
Prom. por bloque	Constante	Profundidad (m)	1000 a 17000, con paso de 2000	9000	0.34
		Tamaño de bloque (m)	1000, 2000, 3000 y 4000	3000	
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-4}	
	Relativa	Profundidad (m)	1000 a 17000, con paso de 2000	9000	0.34
		Tamaño de bloque (m)	1000, 2000, 3000 y 4000	2000	
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-3}	
	Variable	Profundidad (m)	50 a 1450, con paso de 200	50	0.33
		Factor de profundidad	1 a 6, con paso de 1	2	
		k vecinos cercanos	1, 5, 10 y 15	15	
		Tamaño de bloque (m)	1000, 2000, 3000 y 4000	2000	
		Amortiguamiento	$10^{-4}, 10^{-3}, \dots, 10^2$	10^{-2}	
Grilla regular	Constante	Profundidad (m)	1000 a 9000, con paso de 2000	7000	0.34
		Espaciado de grilla (m)	1000, 2000 y 3000	1000	
		Amortiguamiento	$10^{-3}, 10^{-2}, \dots, 10^2$	10^{-1}	

Capítulo 5

Fatiando a Terra

“El cómputo en la Ciencia ha evolucionado no solo porque el software lo ha hecho, sino porque nosotros, como científicos, estamos haciendo mucho más que solo aritmética de punto flotante.”

Fernando Pérez, PyCon Canada 2012

5.1. Introducción

Desde su invención, las computadoras han sido puestas a disposición de la comunidad científica con el objetivo de resolver problemas que resultaban inalcanzables. Esta interacción entre una tecnología de vanguardia y el ambiente científico generaba no solo beneficios para esta última parte, sino también una gran retroalimentación. Se desarrollaron lenguajes de programación especialmente diseñados para resolver problemas numéricos junto con interfaces que facilitaran la visualización y manipulación de datos científicos. La relación entre Ciencia y las herramientas computacionales se desarrolló tan rápido que fue necesario crear el término *computación científica* para diferenciarla de los otros usos que se estaban gestando para las computadoras (telecomunicaciones, fines comerciales, sistemas estatales de datos, etc.). Hoy en día es imposible imaginar una Ciencia que no necesite de las herramientas computacionales para su avance y la resolución de los problemas que enfrenta en la actualidad.

A medida que los problemas científicos se vuelven cada vez más complejos de resolver, también lo hacen las tareas necesarias para hacerlo. Mantenerse actualizado en los últimos conocimientos en la materia, adquirir nuevos datos, desarrollar el software necesario para procesarlos y finalmente generar un nuevo conocimiento se presenta como un desafío titánico para ser desempeñado por una persona o por apenas un puñado de investigadores. La complejidad actual de la Ciencia requiere que el flujo de trabajo científico se distribuya a lo ancho de la comunidad, ofreciendo productos o soluciones para cada una de sus etapas, que puedan ser utilizados libremente por otros investigadores y otras investigadoras, que a su vez puedan modificarlos y volver a distribuirlos

en caso de desearlo. En resumen, los problemas científicos actuales requieren soluciones comunitarias y colaborativas, tanto para dar respuestas a las preguntas fundamentales, así como para desarrollar herramientas que faciliten la resolución de estos problemas.

Los movimientos de Software Libre y de código abierto que comenzaron a proliferar durante las décadas de los 80 y 90 tuvieron una fuerte raíz en el ámbito científico. Muchos científicos y científicas comenzaron a desarrollar herramientas computacionales a partir de necesidades propias de sus investigaciones que luego las distribuyeron bajo licencias de Software Libre y de código abierto. Dentro de las Ciencias de la Tierra podemos hallar Seismic Unix ([Stockwell, 1999](#)), cuyo desarrollo comenzó a finales de la década del 70 y está disponible bajo una licencia BSD Clause-3; y GMT (Generic Mapping Tool) ([Wessel et al., 2019; Wessel y Smith, 1991](#)), creado en 1988 por Paul Wessel y Walter H.F. Smith y lanzado bajo la GNU Lesser General Public License¹ en 1991.

A inicios de la década del 2000 se comienzan a desarrollar herramientas especialmente orientadas a aplicaciones científicas y de cálculo numérico que hacían uso del lenguaje de programación Python, creado por Guido van Rossum una década atrás. En 2001 se lanza IPython ([Perez y Granger, 2007](#)), un intérprete de comandos interactivo, cuyo objetivo es facilitar la forma en la cual los usuarios y las usuarias operan con sus datos. En el mismo año ve la luz la primera versión de SciPy ([Virtanen et al., 2020](#)), una librería con módulos de optimización, álgebra lineal, integración, interpolación, resolución de ecuaciones diferenciales, aplicación de Transformada de Fourier, entre otros. En 2003 aparece la primera versión de Matplotlib ([Hunter, 2007](#)), una librería para realización de gráficos y visualización de datos con una interfaz de programación de aplicaciones (API) orientada a objetos. En 2006, se bautiza la librería NumPy ([Harris et al., 2020](#)), heredera de Numeric, dando soporte a operaciones con arreglos al lenguaje Python, para lo cual no había sido diseñado.

Estas herramientas comenzaron a ser utilizadas dentro de diferentes ambientes científicos y los entornos que ofrecía Python comenzaron a perfilarse como un excelente ecosistema para la computación científica ([Millman y Avanzis, 2011; Oliphant, 2007; Perez et al., 2011](#)).

La segunda década del nuevo milenio comenzó a vislumbrar una explosión en la cantidad de aplicaciones y nuevos desarrollos de software alrededor de estas nuevas herramientas. Se destacan proyectos como Jupyter ([Kluyver et al., 2016](#)), lanzado en 2016, que logró introducir un nuevo tipo de forma de interactuar con código de investigación: los *Jupyter Notebooks*, que facilitan no solo la experimentación, sino la comunicación de los procesos y resultados científicos junto con la reproducibilidad de los mismos. Librerías como Pandas ([Wes McKinney, 2010](#)), que ofrece herramientas para la manipulación y análisis de datos; scikit-learn ([Pedregosa et al., 2011](#)), especialmente orientada a la aplicación de algoritmos de aprendizaje automático; SymPy ([Meurer et al., 2017](#)) para cálculo

¹https://en.wikipedia.org/wiki/GNU_Lesser_General_Public_License

simbólico; entre otras. Surge también Anaconda², una distribución de Python orientada específicamente a la Ciencia de Datos, aprendizaje automático e investigaciones científicas; facilitando la instalación del intérprete y las múltiples librerías en diferentes plataformas y sistemas operativos.

En este Capítulo expondremos las características del proyecto Fatiando a Terra, del cual el autor de esta Tesis forma parte del grupo de desarrolladores. Recorremos su historia y cómo, dentro de su marco, se construyen herramientas computacionales con aplicaciones directas en Geofísica, particularmente sobre temáticas relacionadas a métodos potenciales. Enumeraremos las prácticas de desarrollo de software aprendidas y aplicadas en la construcción del proyecto, y qué beneficio han traído a la generación de investigaciones científicas reproducibles. También exploraremos de qué manera existe una retroalimentación entre los proyectos de software científico y las investigaciones que estos permiten. Finalmente bosquejaremos un conjunto de ideas a implementar a futuro, definiendo los nuevos caminos que el proyecto debe tomar en función del ambiente científico actual.

5.2. Historia

Los orígenes del proyecto se remontan a finales de la década del 2000, cuando Leonardo Uieda y Vanderlei Oliveira Jr. junto a otros estudiantes se encontraban cursando los últimos años del Bachillerato en Geofísica (*Bacharelado em Geofísica*) en la Universidade de São Paulo. En este contexto surge la idea de implementar una alternativa propia a los software de modelado gravimétrico 2D comerciales. Tras asistir a cursos dictados por Software Carpentry³ en Canadá, Leonardo Uieda adquiere mayores conocimientos en el uso de sistema de control de versiones junto a otras mejores prácticas para el desarrollo de software y regresa con nuevas ideas para el proyecto. De vuelta en São Paulo, se gesta un diagrama de un posible diseño para este proyecto. La idea finalmente se desarrolla mediante una implementación del método de [Talwani et al. \(1959\)](#) para modelado directo de polígonos 2D, y la posterior construcción de una interfaz gráfica escrita en C que más tarde se reimplementaría en Python.

En paralelo, y como proyecto final de su Bachillerato, Leonardo Uieda realiza una implementación en Python del algoritmo para el cálculo de campos gravitatorios de teseroides mediante la [GLQ](#). Luego de ser reescrito en C, este proyecto deriva en el lanzamiento del software Tesseroids ([Uieda et al., 2016](#)) que ha sido ampliamente utilizado por la comunidad geocientista.

El éxito de estos proyectos llevó a estos estudiantes a aspirar a una idea mucho más ambiciosa: desarrollar un software de código abierto para modelar el planeta Tierra de forma completa. En ese entonces surge un nombre para el proyecto: *Fatiando a Terra*, que puede traducirse como *rebanando la Tierra*.

²<https://www.anaconda.com>

³<https://software-carpentry.org>

Durante su Master en Geofísica en el Observatório Nacional, Rio de Janeiro, Leonardo comienza el desarrollo de Fatiando a Terra, transformándolo en el hogar de las implementaciones que realiza a lo largo de sus investigaciones, todas mediante el lenguaje Python. Entre ellas podemos encontrar: modelados directos con diferentes geometrías, continuación ascendente, deconvolución de Euler, fuentes equivalentes, hasta un incipiente *framework* de inversión y algunas implementaciones de tomografías sísmicas simples.

En los años posteriores, cuando la mayoría de los creadores del proyecto se encontraban cursando sus Doctorados, Fatiando a Terra comienza a cobrar mayor forma. Leonardo y Vanderlei deciden utilizar el *framework* de inversión para dar un curso sobre inversiones geofísicas en la Universidade de São Paulo⁴. Fatiando adquiere su propio dominio (fatiando.org) y su primera página web, alojada en un servidor casero por José Caparica Jr⁵. Además se elige distribuirlo bajo la licencia [BSD 3-clause](#). En 2013, el proyecto es presentado en una charla en SciPy 2013 ([Uieda et al., 2013](#)).

En los años posteriores Fatiando a Terra comienza a cobrar mayor reconocimiento. Empieza a ser utilizado en publicaciones científicas ([Carlos et al., 2014, 2016; Hidalgo-Gato y Barbosa, 2015, 2017; Oliveira et al., 2015; Reis et al., 2016; Siqueira et al., 2017; Uieda y Barbosa, 2017, 2012a](#), entre otros), dictado de clases (Tópicos de inversão em Geofísica⁶, [Uieda et al. \(2014\)](#), Geofísica 1: Gravimetria e magnetometria⁷, Geofísica 2: Sismología e sísmica⁸) y en trabajos finales de grado y posgrado ([Carlos, 2013; Ferreira Melo, 2020; Sales, 2014; Soler, 2015; Uieda, 2016](#)). Además comienza a atraer la atención de la comunidad internacional, recibiendo colaboraciones de investigadores y desarrolladores de diferentes partes del mundo. La utilización de la librería fatiando por otros investigadores e investigadoras deriva a su vez en un ciclo de retroalimentación: quienes comienzan siendo usuarios o usuarias, terminan contribuyendo al proyecto. De esta forma fatiando comienza a alojar implementaciones de métodos novedosos recientemente publicados ([Oliveira et al., 2013; Uieda y Barbosa, 2012b](#)).

Mi primera contribución al proyecto consistió en una implementación del promedio radial del espectro de frecuencias de grillas de gravedad o magnetismo⁹. Desde entonces comencé a involucrarme cada vez más, lo que me permitió adquirir mayores conocimientos en el uso de controladores de versiones, flujos de trabajo para el desarrollo colaborativo, creación de funciones de prueba, buenas prácticas para el diseño de algoritmos y redacción de documentación.

La última versión del paquete fatiando es la [v0.5](#), lanzada en Noviembre de

⁴<https://github.com/pinga-lab/inversao-iag-2012>

⁵Más adelante se utilizarán los servicios de [Read The Docs](#) para alojar el sitio web, que luego se reemplazarán por GitHub Pages.

⁶<https://www.leouieda.com/teaching/inversao-iag-2012.html> y <https://www.leouieda.com/teaching/inversao-unb-2014.html>

⁷<https://www.leouieda.com/teaching/geofisica1.html>

⁸<https://www.leouieda.com/teaching/geofisica2.html>

⁹<https://github.com/fatiando/fatiando/pull/303>

2016. Si bien ese paquete en particular se encuentra obsoleto y no recibe mayor mantenimiento, esto no significa que la vida de el proyecto haya finalizado en ese entonces¹⁰.

A partir de 2018 el proyecto tomó una nueva dirección. El panorama de software de código abierto para Geofísica había cambiado mucho desde los inicios de Fatiando a Terra: la cantidad de nuevos paquetes diseñados para atacar diversos problemas de las Geociencias había aumentado considerablemente ([Cockett et al., 2015](#); [de la Varga et al., 2019](#); [Rücker et al., 2017](#); [The ObsPy Development Team, 2019](#)). Dentro de este nuevo ecosistema, fatiando no poseía un objetivo claro. Esto no solo hacía difícil que potenciales usuarios y usuarias identifiquen el propósito del proyecto, sino que también constituía una base de código difícil de mantener. Por otro lado, para ese entonces fatiando había sido el hogar de implementaciones de métodos clásicos de la Geofísica, de métodos muy novedosos (orientados principalmente a la investigación científica), así como de código *juguete* diseñado para ser utilizado en clases con fines didácticos pero sin las capacidades para atacar problemas reales. Sumado a esto, la versión de Python 2.7 llegaba pronto a su final de vida, lo que hacía necesario adaptar fatiando al nuevo Python 3.

Estas razones ponían en evidencia la necesidad de establecer objetivos claros para el proyecto, así como también repensar su diseño y sustentabilidad a futuro. Por esto se tomó la decisión de dividir el proyecto en varios paquetes que posean objetivos claros y concisos. Esto permitiría no solo una fácil adopción por parte de usuarios y usuarias, sino también que otros proyectos los utilicen como dependencias en caso de desearlo. Además, manteniendo los campos de acción de cada paquete aislados del resto, se facilitaría el desarrollo a futuro: los colaboradores no necesitan familiarizarse con el proyecto completo, sino solo con algunas de sus partes. Por otro lado, la decisión de reescribir gran parte del código se presentó como una oportunidad para pensar mejores diseños del software ya existente y de implementar mejores prácticas para el desarrollo de software, estableciendo estándares de calidad más altos.

5.3. Paquetes de software

Actualmente Fatiando a Terra está compuesto por cuatro paquetes de software destinados a ofrecer soluciones a diversas problemáticas en Geociencias, así como también a otras áreas de la Ciencia:

Verde Procesado y grillado de datos espaciales

Boule Elipsoides de referencia para aplicaciones geodésicas y geofísicas

¹⁰La documentación correspondiente a la versión 0.5 de fatiando puede hallarse en <https://legacy.fatiando.org>.

Harmonica Procesado, modelado directo e inverso de datos gravimétricos y magnéticos

Pooch Descargar y almacenar datos científicos de la web de forma sencilla

5.3.1. Verde

Verde ofrece herramientas para procesar datos espaciales (tales como muestras geofísicas obtenidas en el campo, mediciones de batimetría, etc) e interpolarlos sobre grillas regulares.

La mayoría de los muestreos de datos espaciales se componen por datos irregularmente distribuidos sobre la zona de estudio: puntos dispersos, trayectos irregulares o líneas casi rectas. Sin embargo, muchas metodologías de procesado e interpretación requieren que los datos se sitúen sobre puntos pertenecientes a una grilla regular. Este problema suele resolverse mediante la interpolación de los datos originales sobre una grilla regular, proceso que se conoce como *grillado*.

Dentro de los algoritmos de grillado existe una categoría que se conoce como *interpolaciones con funciones de base radial* (*radial basis function interpolation* en inglés) (Franke, 1982). Un ejemplo de un algoritmo que entra en esta categoría son las *splines* biarmónicas (Sandwell, 1987). Estos métodos asumen que podemos representar los datos observados por una combinación lineal de funciones de Green, es decir:

$$d_i = \sum_{j=1}^M c_j G(\mathbf{p}_i, \mathbf{q}_j), \quad (5.1)$$

donde d_i es el dato i -ésimo, \mathbf{p}_i es la ubicación de ese mismo punto, c_j es un coeficiente escalar, G es una función de Green y \mathbf{q}_j es la ubicación del punto que define la función de Green en el término j -ésimo. El proceso de interpolación consiste en ajustar los valores de los coeficientes c_j mediante mínimos cuadrados de forma tal que se recuperen los datos observados, y luego utilizarlos para predecir valores sobre los puntos de la grilla:

$$d(\mathbf{p}) = \sum_{j=1}^M c_j G(\mathbf{p}, \mathbf{q}_j), \quad (5.2)$$

donde \mathbf{p} es la ubicación de cualquier punto de la grilla.

Dentro de Verde hallamos implementaciones de algoritmos de interpolación de este tipo siguiendo un diseño similar a los modelos de aprendizaje automático de scikit-learn (Pedregosa et al., 2011). Esto permite que quienes estén familiarizados con la aplicación de métodos de aprendizaje automático en Python puedan fácilmente entender cómo utilizar la API de Verde. Entre los algoritmos de interpolación que se encuentran implementados están las *splines* biarmónicas, así como métodos más complejos como la interpolación de vectores 2D con funciones de Green (Sandwell y Wessel, 2016).

Las clases de interpoladores en Verde poseen un método `fit()` que se usa para ajustar los parámetros c_j mediante mínimos cuadrados amortiguados partiendo de un conjunto de datos y sus ubicaciones. Luego podemos hacer uso del método `predict()` para predecir valores en cualquier conjunto de puntos, o bien usando el método `grid()` podemos predecir sobre una grilla regular.

Además, Verde cuenta con utilidades para realizar reducciones por bloque, clases para determinación de tendencias polinomiales, funciones para el manejo de coordenadas, y herramientas para realizar validaciones cruzadas y selección de modelos.

El diseño de Verde permite su extensibilidad: todos los interpoladores heredan su estructura a partir de una clase base denominada `BaseGridder`, lo cual permite construir nuevos tipos de interpoladores de manera sencilla, restando que definir únicamente el tipo de función de Green a utilizar. Además, debido a la compatibilidad con las librerías científicas de Python, es posible integrarlo a flujos de trabajos establecidos con gran facilidad.

La versión v1.0.0 de Verde ha sido publicada en el Journal of Open Source Software luego de un proceso abierto de revisión por pares ([Uieda, 2018](#)). Podemos acceder a la documentación correspondiente a la última versión estable de la librería en [fatiando.org/verde](#).

5.3.2. Boule

Boule es una librería que nos permite representar elipsoides de referencia mediante objetos. Además nos ofrece herramientas para calcular los campos gravitatorios que estos elipsoides generan y realizar conversiones de coordenadas.

Como hemos visto en la Sección 2.3, un elipsoide de referencia puede definirse mediante los semiejes mayor a y menor b , su masa total M y su velocidad angular ω . De forma alternativa pero equivalente, es posible definirlo mediante los siguientes parámetros:

- el semieje mayor a ,
- el achatamiento $f = (a - b)/a$,
- la constante gravitacional geocéntrica GM (donde M es la masa del elipsoide y G la constante de gravitación universal),
- y la velocidad angular ω .

Las funcionalidades principales de Boule vienen dadas dentro de las clases `Ellipsoid` y `Sphere`, las cuales nos permiten definir cualquier elipsoide referencia a partir de los parámetros mencionados. La clase `Sphere` está orientada a representar esferas, las cuales poseen achatamiento nulo y en vez de especificar su semieje mayor, se define su radio. Sin embargo, Boule ya ofrece algunos

de los elipsoides de referencia más ampliamente usados en Geodesia y Geofísica, tales como el WGS84 (`boule.WGS84`), el GRS80: Geodetic Reference System (`boule.GRS80`) e incluso un elipsoide de referencia para el planeta Marte definido según los parámetros disponibles en [Ardalan et al. \(2009\)](#) (`boule.MARS`). Además podemos hallar elipsoides esféricos para la Luna (`boule.MOON`), Venus (`boule.VENUS`) y Mercurio (`boule.MERCURY`) definidos según [Wieczorek \(2015\)](#).

A su vez, la clase `Ellipsoid` posee una serie de propiedades que se calculan a partir de los parámetros que definen cada elipsoide, entre ellos: el semieje menor b , la primera y segunda excentricidad, la excentricidad lineal, el radio medio del elipsoide (medido desde un sistema geocéntrico), el módulo del vector aceleración de la gravedad sobre cualquier punto en el ecuador o en sus polos. Además, la misma clase ofrece un conjunto de métodos útiles para realizar cálculos acerca de los campos gravitatorios que genera el elipsoide y conversión de unidades. El método `geocentric_radius()` nos permite calcular la distancia entre el centro del elipsoide y cualquier punto de su superficie a partir de su latitud geodésica o geocéntrica. Los métodos `geodetic_to_spherical()` y `spherical_to_geodetic()` nos permiten convertir coordenadas geodésicas en esféricas geocéntricas o viceversa siguiendo las ecuaciones de [Vermeille \(2002\)](#). El método `normal_gravity()` implementa la *fórmula cerrada* de [Li y Götze \(2001a\)](#) para calcular el módulo de la aceleración de la gravedad generada por el elipsoide en cualquier punto dado por sus coordenadas geodésicas. En el caso de la clase `Sphere`, su método `normal_gravity()` hace uso de una ecuación más sencilla ([Heiskanen y Moritz, 1967](#)):

$$\gamma(\phi, h) = \sqrt{\left(\frac{GM}{(R+h)^2}\right)^2 + \left(\omega^2(R+h) - 2\frac{GM}{(R+h)^2}\right)\omega^2(R+h)\cos^2\phi}, \quad (5.3)$$

donde R es el radio de la esfera, ϕ la coordenada latitudinal y h la altitud sobre la superficie de la esfera correspondientes al punto de observación.

En el Código 5.1 se ejemplifica cómo se puede utilizar BOule para instanciar el elipsoide WGS84, generar una grilla regular de puntos en coordenadas geodésicas elevadas por sobre la superficie del mismo y realizar tareas como el cálculo de gravedad normal o de conversión de unidades.

Si bien Boule es una librería que se encuentra en un estado funcional y puede ser utilizada en investigaciones científicas al día de la fecha, aún no ha alcanzado su primera versión estable. Es posible que aún sea necesario realizar modificaciones sobre su diseño en los planes de incorporar otros tipos de elipsoides, como elipsoides triaxiales. Podemos acceder a la documentación correspondiente a la última versión en [fatiando.org/boule](#).

5.3.3. Harmonica

Harmonica fue creada con la intención de ser el nuevo hogar de aquellas implementaciones de métodos de procesamiento y modelado de datos gravimétricos.

Código 5.1: Ejemplo de cálculo de gravedad normal y conversión de unidades con elipsoide de referencia WGS84 mediante Boule ([v0.3.1](#))

```
import boule
import verde as vd

# Obtenemos el elipsoide WGS84
elipsoide = boule.WGS84

# Generamos una grilla de puntos de observación en coordenadas geodésicas
# ubicados a una altitud geométrica de 150 metros
region = (-170, 180, -90, 90)
longitud, latitud, altitud = vd.grid_coordinates(
    region=region, spacing=10, extra_coords=150
)

# Calculamos la gravedad sobre estos puntos
gravedad_normal = elipsoide.normal_gravity(latitud, altitud)

# Convertimos estos puntos a coordenadas esféricas geocéntricas
longitud, latitud_sph, radio = elipsoide.geodetic_to_spherical(
    longitud, latitud, altitud
)
```

cos y magnéticos. Además, surgió como oportunidad de rediseñar algunas de estas implementaciones con el objetivo de facilitar su utilización, extensibilidad y mantenimiento a futuro.

Una de las principales herramientas que ofrece Harmonica son las funciones de modelado directo para distintas geometrías y para diferentes sistemas de coordenadas. La geometría más sencilla que hallamos son masas puntuales en coordenadas esféricas geocéntricas o en coordenadas Cartesianas, implementando las ecuaciones [2.26](#) y [2.29](#). Podemos calcular los campos gravitatorios producidos por masas puntuales mediante la función `point_gravity()`. El Código [5.2](#) ejemplifica cómo podemos utilizar esta función para calcular la aceleración de la gravedad producida por dos masas puntuales dadas en coordenadas Cartesianas.

Harmonica cuenta con una implementación del método de [Nagy et al. \(2000\)](#) para calcular los campos gravitatorios de prismas rectangulares mediante la función `prism_gravtiy()`, haciendo uso de las correcciones de [Fukushima \(2020\)](#) descriptas en la Sección [2.2.2](#).

La librería también cuenta con una implementación en Python de los algoritmos de discretización adaptativa y [GLQ](#) para calcular los campos gravitatorios de teseroides con densidad constante, a los cuales podemos acceder mediante la función `tesseroid_gravity()`.

Código 5.2: Ejemplo de cálculo de aceleración de la gravedad producida por dos masas puntuales dadas en coordenadas Cartesianas mediante Harmonica (v0.4.0).

```
import verde as vd
import harmonica as hm

# Definimos las coordenadas de dos masa puntuales debajo de la superficie
# (en metros)
easting = [500, 1500]
northing = [700, 1300]
upward = [-50, -100]
points = (easting, northing, upward)

# Definimos las masas de cada una de ellas (en unidades SI)
masses = [3e11, -10e11]

# Definimos una grilla regular de puntos de observación (50 metros sobre
# la superficie)
coordinates = vd.grid_coordinates(
    region=(0, 2000, 0, 2000), shape=(100, 100), extra_coords=50
)

# Calculamos la componente vertical (hacia abajo) de la aceleración
# gravitatoria
gravity = hm.point_gravity(
    coordinates, points, masses, field="g_z", coordinate_system="cartesian"
)
```

Además, desde la versión v0.4.0 es posible calcular los campos gravitatorios de teseroides con densidad variable en la dirección radial, aplicando la metodología descripta en el Capítulo 3.

Cada una de estas funciones admiten una cantidad arbitraria de geometrías y puntos de observación que pueden venir dadas por estructuras de datos muy sencillas, tales como listas, tuplas o arreglos de valores numéricos. Sin embargo pueden ser utilizadas para aumentar el grado de abstracción del modelo. Por ejemplo, la función `prism_layer()` permite construir una capa de prismas rectangulares de iguales dimensiones horizontales, distribuidos regularmente de forma tal que las caras de prismas contiguos se encuentran en contacto. La función permite asignar diferentes valores para las caras superiores e inferiores de los prismas, y devuelve una instancia de `xarray.Dataset` que contiene las coordenadas de los centros de los prismas junto con las coordenadas `top` y `bottom` para cada uno de ellos. Harmonica ofrece un *accessor* de Xarray para

Código 5.3: Ejemplo de cálculo de aceleración de la gravedad producida por un teseroide con densidad lineal en la dirección radial utilizando Harmonica (v0.4.0).

```
from numba import njit
import boule as bl
import verde as vd
import harmonica as hm

# Utilizamos Boule para obtener el radio medio del elipsoide WGS84
radio_medio = bl.WGS84.mean_radius

# Definimos un teseroide cuya superficie superior se encuentra en el
# radio medio del elipsoide, posee un espesor de 10km y dimensiones
# latitudinales y longitudinales de 10 grados por 10 grados.
teseroide = [-70, -60, -40, -30, radio_medio - 10e3, radio_medio]

# Definimos una densidad lineal para este teseroide.
@njit
def densidad(radio):
    """Funcion de densidad lineal"""
    radio_superior = radio_medio
    radio_inferior = radio_medio - 10e3
    densidad_superior = 2670
    densidad_inferior = 3000
    slope = (
        (densidad_superior - densidad_inferior)
        / (radio_superior - radio_inferior)
    )
    return slope * (radio - radio_inferior) + densidad_inferior

# Definimos puntos de computo en una grilla regular a 100km por
# encima del radio medio del elipsoide
coordenadas = vd.grid_coordinates(
    region=[-80, -40, -50, -10],
    shape=(80, 80),
    extra_coords=100e3 + radio_medio,
)

# Calculamos la componente radial de la aceleracion gravitoria
# que genera el teseroide
gravity = hm.tesseroid_gravity(
    coordenadas, teseroide, densidad, field="g_z"
)
```

poder calcular los campos gravitatorios que toda la capa genera sobre un conjunto de puntos de observación, la cual hace uso de la función `prism_gravity()` mencionada anteriormente. Este tipo de capas resultan muy útiles para calcular los efectos gravimétricos de la topografía o para realizar inversiones como las de [Uieda y Barbosa \(2017\)](#). El Código 5.4 ejemplifica cómo podemos usar la función `prism_layer()` junto con el método `prism_gravity()` del *accessor* para calcular el efecto gravitatorio de la topografía de Sudáfrica sobre una grilla regular a 4000 m de altitud.

La mayoría de los métodos de modelado directo involucran cálculos moderadamente complejos que suelen requerir un alto costo computacional, especialmente a la hora de modelar grandes cantidades de geometrías sobre muchos puntos de observación. Escribir estas funciones en Python puro resultaría en implementaciones que no harían un uso eficiente de los recursos computacionales debido a la misma naturaleza del intérprete de Python. Las alternativas más viables son recurrir a código precompilado (escribiendo las implementaciones en lenguajes como C o FORTRAN y luego añadir *wrappers* de Python) o bien a compilaciones en tiempo de ejecución. En los orígenes de Harmonica, hemos optado por realizar nuestras implementaciones mediante Numba ([Lam et al., 2015](#)): un compilador de Python de alto desempeño. Numba se encarga de pre-compilar el código que implementa cada uno de estos algoritmos al momento de ejecutar las funciones por primera vez, utilizando en adelante los binarios ya compilados. Mediante este método no solo podemos hacer un uso mucho más eficiente de los recursos computacionales sino que además podemos habilitar sencillamente la parallelización de nuestros algoritmos, distribuyendo la tarea en los núcleos disponibles del procesador donde se está ejecutando. Además, dada la naturaleza de Numba, no es necesario que distribuyamos los binarios de Harmonica para cada plataforma o sistema operativo: Numba se encarga de la precompilación durante el tiempo de ejecución. De esta manera, se simplifica mucho la tarea de distribuir e instalar Harmonica en cualquier tipo de plataforma.

Otra de las capacidades que presenta Harmonica es la de realizar cálculos a partir de los algoritmos de fuentes equivalentes. La clase `EquivalentSources` contiene una implementación del método de fuentes equivalentes que utiliza fuentes puntuales y la inversa de la distancia como función de Green (tal y como se describe en la Sección 4.2.1). El ajuste de los coeficientes de las fuentes se realiza mediante un algoritmo de mínimos cuadrados amortiguados, haciendo uso del parámetro de *amortiguamiento* adimensional introducido en la Sección 4.2.2. `EquivalentSources` permite ajustar fuentes equivalentes dadas en coordenadas Cartesianas, mientras que `EquivalentSourcesSph` nos permite aplicar la misma metodología pero definiendo las fuentes puntuales y los puntos de observación en coordenadas esféricas geocéntricas. Estas clases heredan muchos de los métodos y atributos de la clase `BaseGridder` de Verde, haciendo que su utilización sea a su vez muy similar a la que encontramos en los métodos de aprendizaje automático de scikit-learn ([Pedregosa et al., 2011](#)). Las

Código 5.4: Ejemplo de cálculo de efecto gravitatorio de la topografía de Sudáfrica sobre una grilla regular con Harmonica ([v0.4.0](#)) haciendo uso de `harmonica.prism_gravity()`.

```
import pyproj
import verde as vd
import harmonica as hm

# Cargamos datos de topografia sobre Sudafrica
sudafrica_topo = hm.datasets.fetch_south_africa_topography()

# Proyectamos la grilla en coordenadas planas
projection = pyproj.Proj(
    proj="merc",
    lat_ts=sudafrica_topo.latitude.values.mean(),
)
sudafrica_topo = vd.project_grid(
    sudafrica_topo.topography,
    projection=projection,
)

# Construimos un arreglo 2D de densidades para los prismas.
# Cada valor de topografia positiva le asigna una densidad de 2670 kg/m3
# al prisma correspondiente.
# Cada valor de topografia negativa le asigna un contraste de densidad
# de 1000 kg/m3 - 2900 kg/m3.
densidad = sudafrica_topo.copy()
densidad.values[:] = 2670.0
densidad = densidad.where(sudafrica_topo >= 0, 1000 - 2900)

# Construimos la capa de prismas
prismas = hm.prism_layer(
    (sudafrica_topo.easting, sudafrica_topo.northing),
    surface=sudafrica_topo,
    reference=0,
    properties={"density": densidad},
)

# Construimos una grilla de puntos de observacion a 4000m s/el elipsoide
coordinates = vd.grid_coordinates(
    region=(12, 33, -35, -18), spacing=0.2, extra_coords=4000
)
easting, northing = projection(*coordinates[:, :2])
coordinates_projected = (easting, northing, coordinates[:, -1])

# Calculamos el efecto gravitatorio de los prismas sobre la grilla
g_z = prismas.prism_layer.gravity(coordinates_projected, field="g_z")
```

decisiones de diseño han permitido que los métodos de `EquivalentSources` y `EquivalentSourcesSph` reutilicen muchas de las funciones centrales del modelado directo de masas puntuales, permitiendo no solo un ahorro en cantidad de código, sino también el reciclado de las oportunidades que estas proveen mediante el compilador Numba, como paralelizar la construcción de la matriz Jacobiana.

Desde la versión [v0.4.0](#), la clase `EquivalentSources` cuenta con la opción de utilizar fuentes promediadas por bloques, definiendo el tamaño de cada bloque mediante el parámetro `block_size`. Además, se agregó una nueva clase: `EquivalentSourcesGB` la cual implementa las fuentes equivalentes potenciadas por gradiente. El Código [5.5](#) muestra un ejemplo de uso de esta clase para gridear un conjunto de datos gravimétricos de Sudáfrica haciendo uso de fuentes promediadas por bloque en conjunto con la potenciación del gradiente.

Harmonica ofrece además algunas otras funcionalidades para procesar datos gravimétricos: aplicación de la corrección de Bouguer utilizando una aproximación de placa infinita, cálculo de profundidad de raíces según un modelo isostático de Airy, la capacidad de leer archivos `.gdf` provistos por el servicio ICGEM¹¹, creación de muestras sintéticas aéreas o sobre terreno a partir de muestras reales.

Si bien Harmonica es una librería que se encuentra en un estado funcional y puede ser utilizada en investigaciones científicas al día de la fecha, aún no ha alcanzado su primera versión estable. Podemos acceder a la documentación correspondiente a la última versión en fatiando.org/harmonica.

5.3.4. Pooch

Muchas de las librerías de software científico hacen uso de datos de muestra para ejemplificar su funcionamiento en las secciones de la documentación que se conocen como *Galería de ejemplos*. Usualmente estos datos de muestra se incluyen dentro de los repositorios donde se aloja el mismo código fuente de la librería, sin embargo estos archivos de datos no son empaquetados para su distribución con el objetivo de reducir el tamaño de las futuras instalaciones. En cambio, los datos de muestra suelen ser descargados al ser necesarios mediante código implementado dentro de cada una de estas librerías, haciendo uso de paquetes de Python para descargar archivos mediante protocolo HTTP por ejemplo. Pooch surge como una solución a este problema transversal a muchas librerías del ecosistema científico de Python.

Pooch permite gestionar un *registro* en el cual podemos listar las direcciones de cada uno de los archivos que queremos ofrecer para su descarga, junto con el *hash* de una suma de verificación. Al solicitar algún o algunos de estos archivos, Pooch se encarga de descargarlos de su correspondiente localización, almacenarlos en un *caché* local y realizar una suma de verificación con el objeti-

¹¹<http://icgem.gfz-potsdam.de/home>

Código 5.5: Ejemplo de uso de las fuentes potenciadas por gradiente mediante la clase `EquivalentSourcesGB` de `Harmonica` ([v0.4.0](#)).

```
import pyproj
import boule as bl
import harmonica as hm

# Descargamos datos de muestra: mediciones de aceleracion
# de la gravedad sobre Sudfrica
datos = hm.datasets.fetch_south_africa_gravity()

# Proyectamos los datos en coordenadas planas
proyeccion = pyproj.Proj(proj="merc", lat_ts=datos.latitude.mean())
easting, northing = proyeccion(
    datos.longitude.values, datos.latitude.values
)
coordenadas = (easting, northing, datos.elevation)

# Calculamos el disturbio de gravedad, quitando la gravedad normal
elipsoide = bl.WGS84
disturbio = (
    datos.gravity
    - elipsoide.normal_gravity(datos.latitude, datos.elevation)
)

# Inicializamos las fuentes equivalentes
# Usaremos fuentes promedidas por bloque (con bloques de 2km x 2km),
# y ventanas de 100km x 100km.
fuentes_equiv = hm.EquivalentSourcesGB(
    depth=9e3, # profundidad de las fuentes
    damping=10, # parametro de amortiguamiento
    window_size=100e3,
    block_size=2e3,
    random_state=42,
)

# Ajustamos los coeficientes de las fuentes a los datos mediante
# potenciacion de gradiente
fuentes_equiv.fit(coordenadas, disturbio)

# Interpolamos los datos sobre una grilla regular de 2km de espaciado
# a una altitud de 1km
grilla = fuentes_equiv.grid(
    upward=1000,
    spacing=2e3,
    data_names="disturbio_de_gravedad",
)
```

Código 5.6: Ejemplo de descarga de archivos con Pooch ([v1.5.2](#)) mediante la clase `pooch.Pooch`. Este código es a modo de exemplificación y no está escrito para ser ejecutado.

```
import pooch

# Inicializamos una instancia de la clase pooch.Pooch
odie = pooch.create(
    # Utilizamos el directorio cache por defecto de nuestro sistema
    # operativo
    path=pooch.os_cache("my-project"),
    # Definimos la dirección base donde se encuentran los archivos a
    # descargar
    base_url="https://www.somewebpage.org/science/data/",
    # Especificamos los archivos a descargar en el registro
    registry={
        "temperature.csv": "sha256:19
uheidhlkjdwihcwljchwochhw89dcgw9dcgwc",
        "gravity-disturbance.nc": "sha256:1
upodh2ioduhw9celdjhlfvhksgdwikdgcowjhcwoduchowjg8w",
    },
)
# Solicitamos la descarga de uno de los archivos
file_path = odie.fetch("gravity-disturbance.nc")
```

vo de certificar la integridad de los mismos. La descarga de los archivos se lleva a cabo únicamente si no existe tal archivo en el *caché* local. En el Código 5.6 podemos visualizar cómo es posible utilizar la función `pooch.create()` para inicializar una instancia de la clase `pooch.Pooch`, especificando el directorio donde se descargarán los archivos, la dirección base donde se encuentran alojados los archivos de interés, y el *registro* donde enumeramos el nombre de los archivos y sus *hashes* de las sumas de verificación. Luego podemos solicitar la descarga de uno de los archivos mediante el método `fetch()`, el cual devuelve la ruta al archivo descargado y almacenado localmente.

Pooch no solo nos permite descargar archivos desde el protocolo HTTPS, sino también desde FTP y SFTP con autenticación. Incluso es posible descargar archivos almacenados en repositorios de acceso abierto como Zenodo¹² o figshare¹³ únicamente mediante el identificador de objetos digital (DOI). A su vez, también soporta diferentes tipos de suma de verificación, como SHA256, MD5, XXH128, entre otras. Además ofrece la posibilidad de realizar postpro-

¹²<https://zenodo.org/>

¹³<https://figshare.com>

Código 5.7: Ejemplo de descarga de archivos con Pooch ([v1.5.2](#)) mediante la función `pooch.retrieve()`.

```
import pooch

# Usamos la funcion retrieve() para descargar un unico archivo en
# el directorio de cache por defecto de nuestro sistema operativo
file_path = pooch.retrieve(
    # Especificamos la ubicacion del archivo a descargar
    url="https://github.com/fatiando/pooch/raw/v1.0.0/data/tiny-data.txt",
    # Y su suma de verificacion
    known_hash="md5:70e2af3fd7e336ae478b1e740a5f08e",
)
```

cesados luego de la descarga, como por ejemplo descomprimir archivos `.zip`, `.tar`, `.gzip`, `.xz`, etc.

El paquete presenta ciertas cualidades que lo hacen un excelente candidato para ser utilizado por librerías de terceros. En primer lugar, es posible configurarlo para que se adapte a diferentes necesidades en pocas líneas con una sintaxis sencilla. En segundo lugar, es un paquete puramente escrito en Python con dependencias mínimas. Y por último, presenta un alto grado de extensibilidad: es posible definir funciones propias para otro tipos de descargas (mediante otros protocolos, por ejemplo) o bien para realizar postprocesados. Estas funciones personalizadas puede acoplarse fácilmente al *framework* de Pooch, permitiendo utilizar su misma sintaxis y sus capacidades de descarga, almacenamiento y comprobación de sumas de verificación, incluso en aplicaciones particulares.

La versión v0.7.1 de Pooch fue publicada en el Journal of Open Source Software luego de una revisión abierta por pares ([Uieda et al., 2020a](#)).

Hasta entonces la principal audiencia de Pooch consistía en otros y otras desarrolladores de software que deseaban una forma sencilla de distribuir datos de muestra a sus usuarios y usuarias. Sin embargo, en los meses subsiguientes se presentó la oportunidad de incluir otro tipo de audiencia al paquete: usuarios y usuarias finales que deseaban descargar pocos archivos por única vez. Esto se materializó con la creación de la función `pooch.retrieve()` que permite acceder a las mismas funcionalidades de Pooch, pero sin la necesidad de predefinir una instancia de `pooch.Pooch` y con una interfaz más sencilla. El Código 5.7 ejemplifica cómo podemos utilizar la función `pooch.retrieve()` para descargar un archivo, almacenarlo localmente y comprobar su suma de verificación en una única ejecución. La función devuelve la ruta al archivo descargado y almacenado localmente.

Actualmente Pooch es utilizado por varias librerías del ambiente científico, entre ellas: MetPy ([May et al., 2016](#)), icepack ([Shapero et al., 2020](#)), MOABB ([Jayaram y Barachant, 2018](#)), MNE-Python ([Gramfort et al., 2013](#)), yt ([Turk et al.,](#)

2010) junto a otras librerías de Fatiando como Harmonica y Verde. Además, proyectos con aplicaciones más amplias han incluido a Pooch dentro de sus dependencias, como Xarray (Hoyer y Hamman, 2017), scikit-image (van der Walt et al., 2014) y Project Pythia¹⁴.

La documentación correspondiente a la última versión estable de Pooch puede hallarse en fatiando.org/pooch.

5.4. Desarrollo y mejores prácticas

Desde sus orígenes, Fatiando a Terra ha sido desarrollado siguiendo las mejores prácticas que se encontraban en conocimiento de sus desarrolladores. Es por esta razón que a lo largo de sus años de vida se hayan incorporado cada vez mayor cantidad o mejores herramientas para desempeñar estas mejores prácticas, a medida que los desarrolladores adquirían mayores conocimientos.

Todo desarrollo dentro del proyecto se realiza mediante un software de controlador de versiones, y a excepción de algunos momentos en la concepción del proyecto, utilizamos git¹⁵ como controlador de versiones distribuido y GitHub¹⁶ como servidor de repositorios.

Gracias a git, cada librería posee una *rama* principal de desarrollo donde se provee del código fuente que será utilizado por los usuarios y usuarias finales. El flujo de desarrollo de cada una de las librerías del proyecto se basa en los *Issues* y *Pull Requests* que ofrece GitHub. En cada *Issue* ponemos en evidencia la existencia de un problema en el código de la *rama* principal, exponemos una nueva idea a implementar a futuro o solicitamos una nueva capacidad o comportamiento del código existente. Los *Issues* representan un canal de comunicación entre toda la comunidad, incluyendo usuarios y usuarias, desarrolladores y contribuidores.

Cada vez que alguien desea colaborar con alguna modificación o una nueva implementación, esta persona escribe su código en una nueva *rama* que se desprende de la principal y abre un *Pull Request*, es decir, una solicitud a los mantenedores de incluir ese nuevo código en la rama principal. Dentro de los *Pull Requests* otros colaboradores pueden realizar una revisión del código, sugerir cambios y discutir alternativas, de forma tal que la propuesta inicial alcance el estado necesario para poder ser incorporada al código fuente de la librería. Los requisitos para que un nuevo código sea aceptado no son estrictos, sin embargo se busca alcanzar un determinado estándar de calidad siguiendo un conjunto de mejores prácticas. Las revisiones de código no tienen como objetivo determinar si una cierta contribución debe ser aceptada o rechazada, sino que se toman como punto de partida para aplicar las modificaciones necesarias hasta alcan-

¹⁴<https://projectpythia.org/>

¹⁵<https://git-scm.com>

¹⁶<https://www.github.com>

zar dicho estándar de calidad. La guía para los y las contribuidores¹⁷ brinda instrucciones detalladas de cómo llevar a cabo este proceso.

Muchas de las mejores prácticas que aplicamos a lo largo del proyecto pueden hallarse en [Wilson et al. \(2014, 2017\)](#), sin embargo presentamos a continuación una descripción detallada de las más importantes.

5.4.1. Estilo de escritura

El lenguaje Python admite estilos de escritura muy variados, priorizando que el código sea legible por encima de poseer una sintaxis muy estricta. Sin embargo, a la hora de mantener una base de código entre muchos y muchas desarrolladores, es recomendable establecer un estilo de escritura común. Dentro del proyecto hemos decidido seguir los lineamientos establecidos en PEP8 (siglas en inglés de *Python Enhancement Proposals*: propuestas de mejoras de Python)¹⁸. Dentro de estos lineamientos se establecen recomendaciones acerca del tipo de *indentación* (sangrado) que se debe utilizar; la cantidad máxima de caracteres por línea; la cantidad de líneas vacías antes y después de la definición de funciones, clases y métodos; la forma en que se deben nombrar las variables, las funciones y las clases; entre muchas otras.

Con el objetivo de facilitar la tarea de adecuar cualquier nuevo código a estas reglas, tanto para quien lo escribe como para quien lo revisa, hemos normalizado el uso del auto-formateador Black¹⁹, el cual modifica automáticamente cualquier código sintácticamente correcto para adecuarlo a muchos de los lineamientos establecidos en PEP8. De esta forma, los y las contribuidores pueden utilizar el estilo de escritura que les sea más cómodo y luego correr Black para que adopte el estilo que utilizamos en Fatiando.

Sin embargo, Black no es capaz de aplicar algunas de las recomendaciones establecidas en PEP8, especialmente las que tienen que ver con el nombre de las variables, el modo de uso de operadores lógicos, entre otras. Para este tipo de situaciones hacemos uso de otros software que permiten detectar este tipo de errores, además de muchas otras recomendaciones para mejorar la escritura de nuestro código: `pylint`²⁰ y `flake8`²¹. Estas herramientas no solo identifican violaciones a PEP8, sino que además realizan un análisis más profundo del código: desde detectar variables definidas pero no utilizadas, módulos importados innecesariamente, definiciones de funciones extremadamente largas o con muchos argumentos, entre muchas otras. Dentro del proyecto utilizamos `pylint` y `flake8` para identificar este tipo de errores más difíciles de detectar, sin embargo establecemos un criterio más flexible: permitimos ignorar alguna de sus recomendaciones si lo consideramos conveniente.

¹⁷<https://github.com/fatiando/community/blob/main/CONTRIBUTING.md>

¹⁸<https://www.python.org/dev/peps/pep-0008/>

¹⁹<https://black.readthedocs.io/en/stable/>

²⁰<https://pylint.org/>

²¹<https://flake8.pycqa.org/en/latest/>

5.4.2. Documentación

Cada nueva función, clase o método dentro de las librerías de Fatiando debe estar acompañado de su correspondiente documentación, es decir, un texto que describa su funcionamiento, que enumere los parámetros de entrada que admite y que especifique sus salidas o resultados. Opcionalmente la documentación puede incluir ejemplos sencillos de uso, mostrando cuáles son las salidas esperadas para el mismo, y referencias a artículos científicos si el código en cuestión implementa algún método que podamos hallar en la literatura.

Dentro del proyecto se ha establecido como criterio que la documentación debe seguir un estilo establecido por `numpydoc`²², una extensión para Sphinx²³. Mediante Sphinx podemos generar la documentación de toda la librería como un sitio web estático que puede ser servido y consultado por los usuarios y las usuarias. Como hemos mencionado anteriormente, las documentaciones de todos los paquetes del proyecto se pueden hallar bajo el dominio fatiando.org.

Además de la documentación de cada función, clase o método, Sphinx nos permite agregar otro tipo de contenido: descripción del paquete, instrucciones para la instalación, galería de ejemplos, tutoriales específicos para diferentes tareas, etc. Toda la documentación de cada uno de los proyectos forma parte del mismo repositorio sobre el cual desarrollamos el código fuente, lo que nos permite hacer uso del mismo sistema de control de versiones, *Issues* y *Pull Requests* para su mantenimiento y actualización. Esto no solo simplifica la tarea concentrando todo el flujo de trabajo en un mismo lugar, sino que también permite recibir reportes de mejora de la documentación existente del mismo modo que lo hacemos con el código.

Una de las mejores prácticas que hemos aplicado dentro del proyecto es la manera en la que diseñamos el código para una nueva implementación. En vez de partir por la escritura de la implementación misma, comenzamos por la creación de sus ejemplos, en donde identificamos de qué manera deseamos que esa nueva porción de software sea utilizada. Es decir, partimos nuestro diseño bostezando la interfaz con la cual los usuarios y usuarias interactuarán. Luego procedemos a redactar la documentación de esta nueva implementación, siendo consistentes con el diseño inicial. Una vez que nos encontramos conformes con la propuesta, procedemos a escribir la propia implementación. Este flujo de trabajo ha demostrado ser más eficiente que otros que guardan el diseño de la interfaz como última instancia, lo cual suele requerir múltiples iteraciones entre modificaciones de la implementación y de la interfaz.

5.4.3. Pruebas de software

Una de las mejores prácticas más conocidas para el desarrollo de software consiste en el agregado de funciones de prueba o *testing* para cada nueva im-

²²<https://numpydoc.readthedocs.io/>

²³<https://www.sphinx-doc.org/>

plementación. Las funciones de prueba son porciones de software que tienen como único objetivo corroborar el buen funcionamiento del código que compone un determinado paquete. Estas funciones de prueba no están destinadas a ser utilizadas por usuarios y usuarias finales de la forma en que sí son diseñadas las funciones incluidas en la API del proyecto. Por el contrario, se añaden para ser ejecutadas por desarrolladores o por servicios de integración continua (ver Sección 5.4.4).

En el caso particular del software científico, un comportamiento no deseado de nuestro código puede conllevar la obtención de resultados erróneos, que pueden repercutir en el avance de una determinada disciplina o incluso hasta en la toma de decisiones sobre sociedades y la vida de personas. Si bien estos comportamientos erróneos pueden ser percibidos por los usuarios y usuarias o desarrolladores durante el uso cotidiano del código, muchos otros pueden permanecer inadvertidos durante mucho tiempo. Las pruebas de software son herramientas extremadamente útiles a la hora de detectar estos comportamientos no deseados, especialmente porque permiten adelantarse y reconocerlos antes de que puedan tener impactos negativos. Nos permiten detectar errores mientras escribimos una nueva implementación, al realizar modificaciones sobre el código existente, o al intentar utilizar nuestro proyecto con versiones actualizadas de sus dependencias.

Una de las metodologías más aplicadas para construir funciones de prueba son las *pruebas unitarias* (*unit testing*). Estas consisten en concebir el código fuente de nuestro proyecto como un conjunto de unidades independientes de código. Cada unidad de código puede ser interpretada de forma distinta, pero podemos asumir que cada función o método es una unidad. Las pruebas unitarias consisten en escribir funciones de prueba que corroboren el buen funcionamiento de una unidad a la vez, o al menos un determinado comportamiento de una única unidad. Cada una de estas funciones de prueba debe ser independiente del resto y ser capaz de ser ejecutada de manera automatizada. Además, es deseable que todas las funciones de pruebas formen un conjunto completo de prueba, es decir, que las funciones de prueba en su totalidad corroboren el buen funcionamiento de toda la base de código de la librería.

Dentro de Fatiando a Terra hacemos un uso intensivo de las pruebas de software, siendo un requerimiento indispensable para que un nuevo código pueda formar parte de cada paquete. Para su escritura y ejecución hacemos uso del paquete `pytest`²⁴, del cual se pueden obtener reportes detallados sobre la ejecución de las funciones de prueba, identificando cuál o cuáles fallan y por qué razón. Además, `pytest` ofrece herramientas muy útiles para una escritura más simple de estas funciones de prueba, tales como parametrizaciones mediante decoradores de funciones y la creación de *fixtures* que pueden ser utilizados transversalmente en cada función de prueba, ahorrando la necesidad de repetir código. `pytest` permite además generar un reporte de cobertura, es decir, un

²⁴<https://docs.pytest.org>

porcentaje de qué cantidad de la base del código es sometida a prueba y detalla cuáles son las líneas que no.

La gran mayoría del código fuente que podemos encontrar en las librerías del proyecto es fácilmente probable, especialmente las que tienen que ver con estructuras de datos y comportamientos particulares frente a diferentes parámetros. Sin embargo, en algunas de nuestras librerías incluimos implementaciones de métodos numéricos y soluciones analíticas a problemas geofísicos, los cuales pueden presentar determinada complejidad de corroborar, ya que su implementación surge de la propia necesidad de arribar a los resultados numéricos que ellas generan. A pesar de esto, existen algunos métodos para someterlas a prueba. Tanto las soluciones analíticas como los cálculos numéricos pueden ser corroborados con valores conocidos o con soluciones analíticas a casos particulares. Por ejemplo, la implementación del cálculo de gravedad normal en Boule (ver Sección 5.3.2) es corroborada por comparación con la ecuación de Somigliana (ec. 2.100) sobre puntos en la superficie del elipsoide. Así como la implementación del modelado directo por teseroides en Harmonica (ver Sección 5.3.3) es puesta a prueba comparando sus resultados numéricos con la solución analítica para un cascarón esférico. Otro método consiste en verificar que nuestros resultados numéricos o analíticos satisfacen determinadas propiedades matemáticas. Por ejemplo, la implementación del modelado directo por prismas rectangulares presente en Harmonica (ver Sección 5.3.3) debe verificar una determinada simetría: los valores de los campos a lados opuestos del prisma deben poseer los mismos valores.

Es importante destacar que un conjunto finito de funciones de prueba no garantizan la inexistencia de comportamientos no deseados o errores en la base del código, al igual que una cobertura del 100 % no es un indicativo de la calidad del conjunto de pruebas de software. Es por esto que es necesario considerar la escritura de funciones de prueba tan importante como las implementaciones mismas.

5.4.4. Automatización e integración continua

Muchas de las tareas involucradas en la aplicación de las mejores prácticas mencionadas deben ser realizadas repetitivamente ante una nueva solicitud de incorporación de código o incluso periódicamente tras actualizaciones en las dependencias de cada librería. Realizar estas tareas de forma manual no solo representa una rutina tediosa, sino que además no resulta eficaz. Es por ello que dentro del proyecto recurrimos a los servicios de *integración continua* para automatizar muchas de estas tareas, quitándonos el peso de tener que ejecutarlas manualmente y permitiéndonos ocupar ese tiempo en otras tareas de forma más eficiente.

Cada vez que un contribuidor o contribuidora crea un *Pull Request*, el servicio de integración continua que provee GitHub mediante las GitHub Actions realiza diversas tareas. Haciendo uso de máquinas virtuales, ejecuta compro-

baciones sobre el estilo de escritura del último código publicado en ese *Pull Request*. Para ello utiliza tanto Black como pylint y flake8, devolviendo un reporte de su ejecución donde se pueden identificar las violaciones a las recomendaciones que realizan estas dos utilidades. Además, mediante otro conjunto de máquinas virtuales procede a instalar el nuevo código junto con todas sus dependencias bajo diferentes sistemas operativos y haciendo uso de diferentes versiones de Python 3. Luego ejecuta la *suite* de pruebas de software en cada configuración mediante pytest, devolviendo también su reporte de fallos y de cobertura. Otro servicio de integración continua que se ejecuta es la construcción de la documentación correspondiente a este nuevo código, la cual también devuelve información acerca de posibles fallos que se hayan encontrado.

A la hora de incorporar un código nuevo a la *rama* principal de cualquier librería del proyecto, se requiere que:

- las comprobaciones de estilos escritura pasen correctamente,
- las pruebas de software no presenten errores en ninguna plataforma, y
- la documentación pueda construirse sin errores.

Dentro de la interfaz del *Pull Request* podemos observar si algunas de estas ejecuciones han fallado, evitando que incorporemos ese nuevo código en la *rama* principal. De esta forma se garantiza que la *rama* principal cuente únicamente con código que sigue estos estándares, bajo la idea de que es más sencillo y factible enmendar una contribución antes de ser incorporada que solucionar el problema posteriormente.

Los servicios de integración continua no son utilizados únicamente durante el proceso de *Pull Request*. Las GitHub Actions son usadas además para construir la documentación presente en la *rama* principal cada vez que un cambio se incorpora a ella y publicarla de forma automática para que esté disponible para toda la comunidad.

Otra tarea que se realiza con cierta periodicidad dentro del proyecto es la de generar *lanzamientos* (*releases* en inglés). Cada lanzamiento consiste en asignar un número a una versión particular del código y publicarlo bajo esa denominación en servicios que proveen las fuentes para distintos gestores de paquetes, tales como PyPI²⁵ (*Python package index*) y conda-forge²⁶. Gracias a esto los usuarios y usuarias pueden descargar e instalar nuestras librerías mediante la utilización de gestores de paquetes. Por ejemplo, si se desea instalar Boule con el gestor pip es posible hacerlo ejecutando:

```
pip install boule
```

Y si deseamos instalar Harmonica con el gestor conda podemos hacerlo con:

```
conda install --channel conda-forge harmonica
```

²⁵<https://pypi.org/>

²⁶<https://conda-forge.org/>

Cada vez que se genera un nuevo lanzamiento mediante GitHub, los servicios de integración continua publican esta nueva versión en PyPI mediante su API de forma automática. Afortunadamente, conda-forge realiza una tarea similar pero desde su lado: registra un nuevo lanzamiento en los paquetes que provee y automatiza el proceso para su publicación.

5.5. Comunidad

Una de las partes fundamentales para que el proyecto se haya podido desarrollar de la forma en que lo hizo es la comunidad que se ha construido a su alrededor y las comunidades con las que ésta interactúa. La comunidad de Fatiando a Terra no solo está formada por los mantenedores del código, desarrolladores y contribuidores, sino también por usuarios y usuarias que intercambian ideas y preguntas mediante los canales de comunicación que se han establecido.

Hacia el resto de la comunidad científica Fatiando se presenta mediante su página web, donde se pueden encontrar descripciones de cada una de las librerías, enlaces a la documentación respectiva de cada una de ellas, instrucciones de instalación, videos tutoriales, información sobre nuestros canales de comunicación e instructivos para comenzar a colaborar en el proyecto.

Como hemos mencionado anteriormente, gran parte del intercambio de ideas al respecto del desarrollo de cada librería, reportes de fallos, propuesta de ideas y revisión de código se realiza mediante GitHub. Sin embargo, muchas otras conversaciones suceden dentro de la sala de Slack²⁷, donde se realizan preguntas, se ofrecen soluciones a problemas comunes, se anuncian eventos y más. La sala de Slack constituye un canal de comunicación más informal y más directo entre los integrantes de la comunidad. Además, se organizan dos tipos de videollamadas periódicas: las reuniones de la comunidad y las reuniones de desarrolladores. Las primeras se llevan a cabo un par de veces al año y tienen como objetivo la sociabilización y la discusión de proyectos, sin entrar en muchos detalles técnicos. Las segundas suceden semanalmente y en ellas se discuten problemáticas específicas del desarrollo de las librerías. A pesar de la diferente naturaleza de ambas, cualquier persona es bienvenida a participar, independientemente de su grado de conocimiento o habilidades técnicas.

Dentro de cada uno de los canales de comunicación de la comunidad se espera que sus integrantes respeten el Código de Conducta²⁸ con el objetivo de generar un ambiente ameno y respetuoso.

La comunidad gestada alrededor de Fatiando a Terra interactúa además con otras comunidades del software de código abierto como SimPEG (Cockett et al., 2015), pyGIMLi (Rücker et al., 2017), GemPy (de la Varga et al., 2019), emsig (Werthmüller, 2017; Werthmüller et al., 2019), PyVista (Sullivan y Kaszynski,

²⁷<http://contact.fatiando.org/>

²⁸https://github.com/fatiando/community/blob/main/CODE_OF_CONDUCT.md

2019), subsurface²⁹, pyGMT (Uieda et al., 2020b), SHTOOLS (Wieczorek y Meschke, 2018), Xarray (Hoyer y Hamman, 2017), scikit-image (van der Walt et al., 2014) y MetPy (May et al., 2016), entre otras.

Aquellas que poseen una estrecha vinculación con las Ciencias de la Tierra se relacionan dentro de comunidades como *software underground*³⁰, donde se llevan a cabo reuniones de desarrolladores y eventos como los *Transform*³¹ en los cuales de dictan tutoriales, diversas charlas y *hackathons*.

5.6. Adopción en la comunidad científica

Desde sus comienzos, las herramientas provistas en Fatiando a Terra han sido utilizadas en publicaciones científicas por investigadores de diferentes partes del mundo. Muchas de ellas citan a los artículos Uieda et al. (2013) y Uieda (2018). Además, el proyecto ha sido presentado en congresos científicos y reuniones científicas (Soler et al., 2021a; Uieda et al., 2013; Uieda y Soler, 2020a).

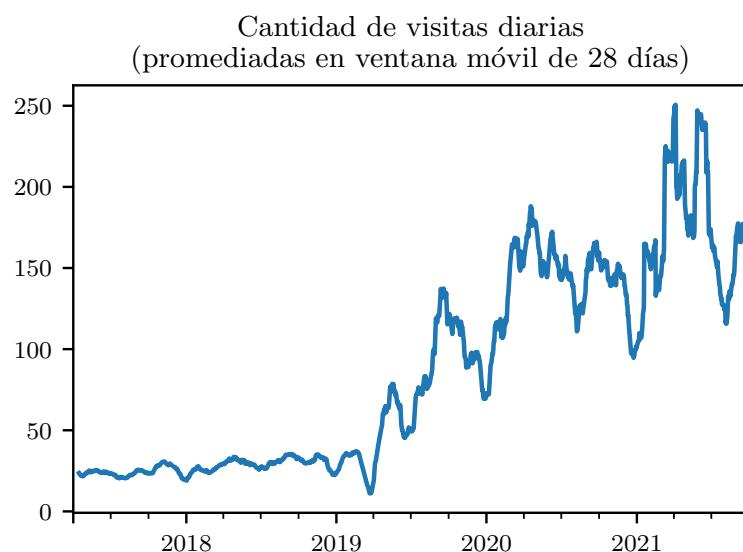


Figura 5.1: Cantidad de visitas a fatiando.org entre Abril de 2017 y Septiembre de 2021 en ventana móvil de 28 días. Cada valor representa la cantidad de visitas diarias promediadas en una ventana de 28 días centrada en su fecha correspondiente.

Estimar la cantidad de usuarios y usuarias que utilizan las herramientas que ofrece el proyecto no es trivial. Es posible conocer la cantidad de descargas de

²⁹<https://softwareunderground.github.io/subsurface/>

³⁰<https://softwareunderground.org/>

³¹<https://softwareunderground.org/transform>

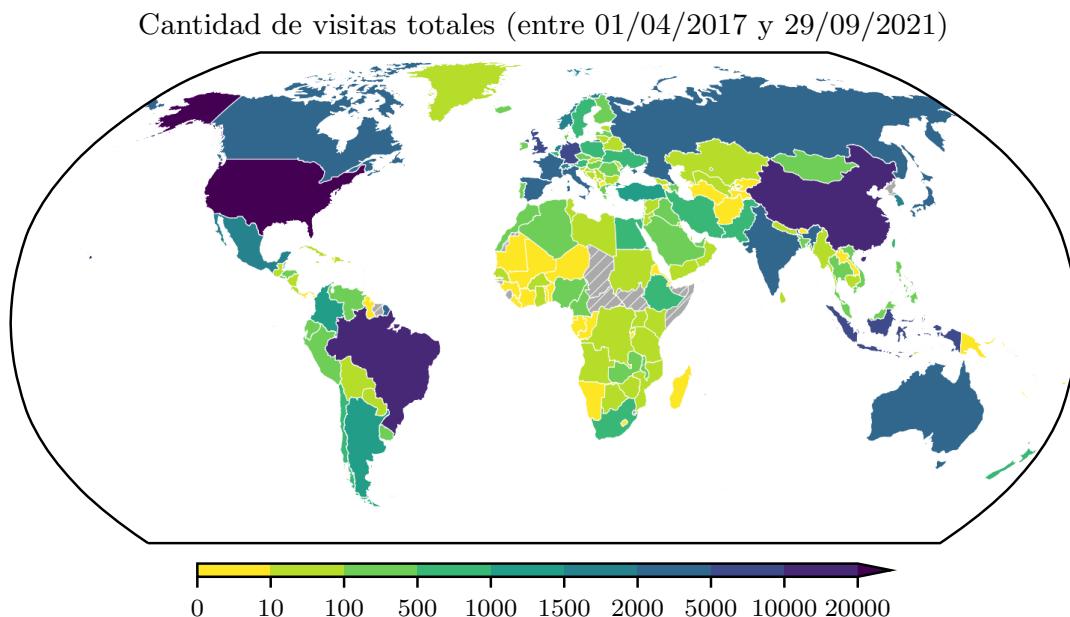


Figura 5.2: Cantidad de visitas a fatiando.org entre Abril de 2017 y Septiembre de 2021 por ubicación geográfica. Los países en color gris con líneas diagonales blancas corresponden a sitios de los cuales no se han recibido visitas en el mismo período. La escala de colores posee pasos no proporcionales para mejorar la visualización del mapa.

los paquetes desde PyPI o `conda-forge`, sin embargo muchas de ellas son realizadas de manera automática por servicios como los de integración continua, introduciendo una sobreestimación de la cifra. Quizás una métrica más representativa de esta variable es la cantidad de visitas al sitio web fatiando.org, incluidas las páginas de documentación. Estos datos han sido recopilados mediante los servicios de Google Analytics de manera anónima (no se recogen las direcciones de IP de quienes visitan el sitio). La Figura 5.1 muestra la cantidad de visitas diarias que recibió el sitio desde Abril del 2017 a Septiembre de 2021 promediadas por ventana móvil de 28 días. Por su parte, la Figura 5.2 expone la cantidad de visitas en el mismo intervalo según su ubicación geográfica.

Estos datos ilustran que la cantidad de visitas experimenta una tendencia ascendente, principalmente desde comienzos del 2019, con algunas variaciones estacionales. Además, podemos ver que países como Estados Unidos de América, Brasil y China acaparan la mayor cantidad de consultas al sitio web del proyecto, aunque el interés sobre el mismo parece estar distribuido entre muchos países del globo.

5.7. Contribuciones del autor al proyecto

Los primeros aportes que he realizado a Fatiando a Terra fueron a la antigua librería fatiando. Entre ellos podemos encontrar: una función para el cálculo del promedio radial del espectro de potencias de una grilla, una función para leer archivos .gdf que provee el servicio de ICGEM³², un agregado a la galería de ejemplos y arreglos a algunos errores de software.

Una vez que el proyecto se volcó en el desarrollo de las librerías descriptas en las secciones anteriores, mi rol en el proyecto cobró cada vez más relevancia.

Dentro de Harmonica he rediseñado los modelos directos de masas puntuales, teseroides y prismas rectangulares, portándolos a Python 3, habilitando su paralelización mediante Numba y aplicando optimizaciones adicionales. He creado las clases `EquivalentSources` y `EquivalentSourcesSph` para aplicar la técnica de fuentes equivalentes, las cuales fueron también paralelizadas mediante Numba. He creado la función `harmonica.prism_layer` para la construcción de capas de prismas, resultando útiles para el modelado directo de topografía. Además, he incorporado las metodologías desarrolladas en esta Tesis a la librería, permitiendo el modelado directo de teseroides con densidad variable y la utilización de fuentes equivalentes promediadas por bloque y fuentes equivalentes potenciadas por gradiente. Actualmente, me encuentro liderando el desarrollo de Harmonica, siendo el principal autor y responsable de toma de decisiones.

Además, he realizado múltiples contribuciones a las otras librerías del proyecto. Dentro de Boule he implementado los métodos para la transformación de coordenadas y para el cálculo de gravedad normal mediante la fórmula cerrada de Li y Götze (2001b); además he agregado varias propiedades a la clase `Ellipsoid` y he escrito la nueva clase `Sphere`. Dentro de Pooch he escrito un método para corroborar la disponibilidad de los archivos remotos y he contribuido a la construcción de la función `pooch.retrieve`. En Verde he mejorado el manejo de proyecciones dentro de las clases de interpoladores y he creado la función `verde.make_xarray_grid` para la construcción de `xarray.Datasets` a partir de las grillas generadas por la librería. En cada una de estas librerías, me posiciono como el segundo autor con mayor cantidad de contribuciones.

Mis contribuciones al proyecto no se limitan únicamente al agregado de nuevas características, sino que una gran parte de ellas involucran tareas de mantenimiento: desde la resolución de errores de software, asegurar soporte para cada nueva versión de las dependencias, adopción de nuevas herramientas que ayudan al desarrollo, agregado de pruebas de software, revisión del código de terceros, hasta la realización de modificaciones al sitio web del proyecto y a las páginas de documentación de cada librería. Tengo además una participación activa en la comunidad, asistiendo a la gran mayoría de las reuniones del proyecto, respondiendo preguntas de los usuarios y las usuarias, resolviendo pro-

³²<http://icgem.gfz-potsdam.de/home>

blemas que les puedan surgir, generando nuevas características frente a nuevas necesidades que ellos y ellas puedan presentar. Incluso he desarrollado y dictado un tutorial sobre el uso de nuestras librerías ([Soler et al., 2021a](#)) y participado en charlas dictadas en reuniones científicas ([Uieda et al., 2021a,b,c](#)).

5.8. Relación con investigaciones científicas

Como se ha mencionado en los Capítulos [3](#) y [4](#), las herramientas desarrolladas en Fatiando a Terra han resultado indispensables para la culminación de las investigaciones descriptas en esta Tesis. Sentaron bases para la construcción de las implementaciones de estas nuevas metodologías y proveyeron de herramientas útiles para ponerlas a pruebas sobre datos sintéticos o reales. Esta relación entre el proyecto e investigaciones científicas como facilitador de herramientas no solo se ve reflejada en los avances de esta Tesis, sino que se replica en todos los trabajos que han hecho uso de las mismas, particularmente los mencionados en la Sección [5.2](#).

Existe además una relación de retroalimentación entre el proyecto y estas investigaciones científicas: muchos de los métodos publicados en artículos científicos son incorporados al mismo, generando una forma alternativa de acceder a sus implementaciones. Esto facilita mucho su aplicación por parte de otros investigadores y otras investigadoras: no es necesario recurrir a los materiales suplementarios de las publicaciones (en caso de que existan), sino que pueden ser accedidos gracias a paquetes de software propiamente distribuidos que además aplican mejores prácticas para su desarrollo y mantenimiento. Un ejemplo de esto son los algoritmos desarrollados a lo largo de esta Tesis y descriptos en los Capítulos [3](#) y [4](#), los cuales fueron incorporados a Harmonica.

5.9. Planes a futuro

Cada uno de los paquetes del proyecto se encuentran en situaciones diferentes con respecto a su estabilidad, estado de su desarrollo y penetración como dependencia de otros proyectos. En función de estas características, cada uno de los paquetes presenta diversos objetivos a alcanzarse en el corto y mediano plazo.

Por ejemplo, Pooch ha sido adoptado como dependencia por muchos otros proyectos, es por ello que se hace un fuerte énfasis en priorizar la retrocompatibilidad (*backward compatibility*) a la hora de incluir nuevas características con el objetivo.

A largo plazo sería deseable poder lanzar versiones estables de Harmonica y Boule para luego presentarlos para su publicación en el Journal of Open Source Software. Para lograr esto se requiere alcanzar otros objetivos intermedios.

Dentro de Boule existe la intención de incorporar elipsoides triaxiales y unificar la inicialización de cualquier tipo de elipsoide (esfera, elipsoide de revolución o triaxial) combinando todos en una sola clase o proveyendo una función particular que decide qué clase debe utilizar en función de los parámetros que recibe. Además, deseamos incluir cálculos de campos gravitatorios generados por los elipsoides, es decir, aceleración de la gravedad que no contenga las componentes de la aceleración centrífuga.

Por su parte, en Harmonica deseo incorporar las metodologías desarrolladas en esta Tesis: modelado directo de teseroides con densidades variables y las fuentes equivalentes potenciadas por gradiente. Por su parte, en Harmonica deseamos implementar algunas de las metodologías basadas en Transformadas de Fourier, tales como continuación ascendente, derivadas direccionales, reducción al polo de datos magnéticos o deconvolución de Euler. A largo plazo poseemos el objetivo de diseñar un *framework* de inversión similar al que existía en fatiando v0.5, pero haciendo uso a las últimas herramientas disponibles.

De forma más global se encuentra el objetivo de incrementar la interoperabilidad de nuestros paquetes con los desarrollados por las otras comunidades de software de código abierto mencionadas en 5.5. En esta tarea, comunidades como *software underground* juegan un papel fundamental como punto de reunión entre todas las personas involucradas.

Capítulo 6

Conclusiones

Durante el desenvolvimiento de esta Tesis Doctoral se desarrollaron dos nuevas metodologías para el modelado y procesamiento de datos gravimétricos que pueden ser aplicadas a un amplio espectro de problemáticas geofísicas, desde investigaciones científicas hasta exploración de recursos naturales. Cada una de estas metodologías ha sido implementada mediante software a través del lenguaje de programación Python y se encuentra disponible bajo licencias de código abierto.

La primer metodología consiste en un modelo directo que permite calcular los campos gravitatorios generados por teseroides cuya densidad puede expresarse como una función continua dependiente de la coordenada radial. Este nuevo método resuelve numéricamente las integrales involucradas mediante una Cuadratura de Gauss-Legendre de segundo orden aplicada en conjunto con dos algoritmos de discretización:

- un algoritmo de discretización adaptativa bidimensional que divide al teseroide en las direcciones longitudinales y latitudinales en función de la distancia entre el centro del teseroide y el punto de cómputo; y
- un algoritmo de discretización basado en la densidad que lo fracciona en la dirección radial, donde toma lugar la *máxima variación de densidad*.

Estos algoritmos cuentan con dos parámetros que controlan de manera indirecta la cantidad de subdivisiones que se llevan a cabo: el *ratio* distancia-tamaño D y el *ratio delta* δ , respectivamente. Los valores que éstos asumen poseen un impacto sobre el tiempo de cómputo necesario para llevar a cabo los cálculos y sobre la precisión de los resultados.

Con el objetivo de hallar valores óptimos de D y δ que minimicen el tiempo de cómputo mientras alcanzan precisiones aceptables, llevamos a cabo comparaciones empíricas entre los resultados numéricos y las soluciones analíticas para cascarones esféricos con densidades lineales, exponenciales y sinusoidales. Como resultado, hallamos que al utilizar valores de D de 1 y 2.5 para el potencial gravitatorio y para su gradiente, respectivamente, la integración numérica

alcanza errores por debajo del 0.1 % para cascarones de diferentes espesores y sobre puntos de observación situados en diversas localizaciones. De manera análoga, la elección de un *ratio* δ de 0.1 permite alcanzar la misma precisión para diferentes funciones de densidad, a excepción de funciones sinusoidales con altas frecuencias, en cuyo caso resulta en errores por debajo del 1 % con respecto a las soluciones analíticas. Además, concluimos que las elecciones de D y δ no poseen una relación significativa entre sí y por ende la elección de sus valores puede realizarse de forma independiente.

Si bien la incorporación del algoritmo de discretización basado en la densidad introduce un mayor costo computacional, esta metodología presenta un alto grado de flexibilidad: es posible utilizar cualquier tipo de función continua sin necesidad de modificar la implementación del algoritmo. Esta característica hace que esta metodología pueda ser ampliamente utilizada para diferentes fines, sin necesidad de limitarse a casos particulares.

Finalmente, esta nueva metodología fue aplicada para el modelado directo de la Cuenca Neuquina ubicada al Norte de la Patagonia Argentina. Se ha calculado el efecto gravitatorio del paquete sedimentario de la cuenca tomando en consideración la compactación del relleno, demostrando que existe una diferencia apreciable en comparación a asumir valores constantes de densidad.

La segunda metodología consiste en un algoritmo que permite realizar interpolaciones de grandes cantidades de datos gravimétricos o magnéticos mediante la utilización de *fuentes equivalentes potenciadas por gradiente*. Este nuevo método construye conjuntos más pequeños de fuentes equivalentes mediante la utilización de ventanas solapadas, cuyos coeficientes son determinados de forma iterativa mediante un algoritmo de mínimos cuadrados amortiguados. De esta forma, las *fuentes equivalentes potenciadas por gradiente* son capaces de transformar un gran problema con altos requisitos de memoria computacional en pequeños problemas más sencillos de resolver que son acumulados de forma iterativa.

Mediante sucesivas pruebas sobre datos sintéticos, hemos sido capaces de demostrar que esta nueva metodología es capaz de alcanzar precisiones similares a las obtenidas mediante fuentes equivalentes regulares, reduciendo el tiempo de cómputo de forma considerable y permitiendo interpolar grandes cantidades de datos. También hallamos que un solapamiento del 50 % entre ventanas contiguas produce un balance óptimo entre tiempo de cómputo y precisión de las predicciones. Además, hemos demostrado que aleatorizar el orden en el cual se lleva a cabo la iteración de las ventanas es fundamental para obtener resultados más precisos y acelerar la convergencia del algoritmo. Con el objeto de maximizar la precisión del método, recomendamos elegir el máximo tamaño de ventana que genere matrices Jacobianas que puedan almacenarse en memoria.

En adición a esta metodología, presentamos una nueva estrategia para ubicar las fuentes equivalentes, que denominamos *fuentes promediadas por bloques*. Esta consiste en dividir la región de estudio en bloques de igual tamaño y ubicar

una fuente en la posición horizontal media de las observaciones que caen dentro de cada bloque. De esta forma, la cantidad de fuentes equivalentes es mucho menor si comparamos con otras estrategias, como ubicar una fuente debajo de cada observación o en una grilla regular.

Mediante la utilización de datos sintéticos, hemos comparado las predicciones que se obtienen al utilizar cada distribución de fuentes equivalentes, incluyendo también diversas formas de determinar sus profundidades. Nuestros resultados indican que las *fuentes promediadas por bloques* reducen el costo computacional del método, alcanzando el mismo nivel de precisión que al utilizar distribuciones de fuentes clásicas. A partir de nuestra experiencia, recomendamos elegir un tamaño de bloque del mismo orden que la resolución de la grilla sobre la cual se interpolarán los datos.

Hemos observado que las diferentes estrategias para asignar la profundidad de las fuentes no modifican significativamente la precisión de las interpolaciones, a excepción de las *profundidades variables* que producen resultados ligeramente más precisos. Sin embargo, dado que involucran mayor cantidad de hiperparámetros, su uso puede conllevar un aumento en el costo computacional durante la determinación de los mismos mediante métodos como la validación cruzada. Por esta razón, recomendamos la utilización de *profundidades constantes* o *profundidades relativas* en la mayoría de los casos.

Dado que las *fuentes equivalentes potenciadas por gradiente* no realizan ninguna suposición sobre las ubicaciones de las fuentes mismas, es posible utilizar este método junto con *fuentes promediadas por bloques*, reduciendo los requerimientos de memoria computacional por dos medios independientes.

Finalmente, hemos aplicado las *fuentes equivalentes potenciadas por gradiente* junto con *fuentes promediadas por bloques* para interpolar una colección de datos gravimétricos sobre Australia con más de 1.7 millones de puntos. Las predicciones se realizaron sobre una grilla regular a altitud constante con una resolución de 1 minuto de arco, aproximadamente. El proceso fue llevado a cabo en una computadora personal modesta, utilizando menos de 16 Gigabytes de memoria computacional. Esto ha demostrado la capacidad de las *fuentes equivalentes potenciadas por gradiente* a la hora de interpolar grandes cantidades de datos manteniendo todos los beneficios de la técnica de fuentes equivalentes: toma en consideración la altitud de las observaciones, permite realizar predicciones en cualquier punto del espacio, y garantiza que las mismas representen un campo armónico.

En paralelo al desarrollo de estas nuevas metodologías, a lo largo de esta Tesis he realizado múltiples contribuciones a proyectos de software de código abierto, particularmente a las librerías pertenecientes a Fatiando a Terra. Estas contribuciones han favorecido a la creación y desarrollo de cuatro librerías de software escritas en lenguaje Python: Verde (interpolaciones de datos georreferenciados mediante funciones de Green), Boule (elipsoides de referencia y cálculo de gravedad normal), Harmonica (procesamiento y modelado de datos

potenciales) y Pooch (descarga de datos científicos de la web). La mayoría de estas librerías poseen una íntima relación con las áreas de la Geofísica en las que se desarrolla esta Tesis.

Las investigaciones científicas aquí expuestas han tenido una profunda relación de retroalimentación con los proyectos de software nombrados: gran parte del código de estos últimos ha sido desarrollado en función de las necesidades de las investigaciones que se llevaron a cabo, mientras que los proyectos científicos adquirieron una base de herramientas sobre las cuales fue posible experimentar y desarrollar nuevas metodologías. Como resultado de esta relación, los proyectos de software reciben implementaciones de métodos de vanguardia, permitiendo al resto de la comunidad científica poder aplicarlos a sus problemáticas particulares sin necesidad de escribir sus propias implementaciones. La incorporación de los dos métodos desarrollados en esta Tesis a la librería Harmonica son un ejemplo de ello.

De esta manera, el avance científico no se realiza únicamente dentro de los artículos publicados, sino que existe también una colaboración indirecta con el resto de la comunidad, poniendo a disposición estas implementaciones bajo licencias de código abierto.

La oportunidad de desarrollar software científico siguiendo las mejores prácticas ha tenido un impacto en la calidad de las investigaciones expuestas en esta Tesis. El desarrollo de cada una de las metodologías presentadas ha sido realizado utilizando las mismas herramientas que se detallaron en el Capítulo anterior: el uso de controladores de versiones, la aplicación de las recomendaciones acerca de los estilos de escritura, la creación de funciones de prueba para las nuevas implementaciones de los métodos, entre otras. La aplicación de estas prácticas heredadas del desarrollo de software permitieron generar publicaciones científicas reproducibles.

Lista de Abreviaturas

GLQ Cuadratura de Gauss-Legendre

RMSE raíz del error cuadrático medio

API interfaz de programación de aplicaciones

DOI identificador de objetos digital

Bibliografía

- Ahmadi, H., Marti, J. R., y Moshref, A. (2013). Piecewise linear approximation of generators cost functions using max-affine functions. En *2013 IEEE Power & Energy Society General Meeting*. IEEE. doi:[10.1109/pesmg.2013.6672353](https://doi.org/10.1109/pesmg.2013.6672353).
- Anderson, E. G. (1976). The effect of topography on solutions of Stokes' problem. *Kensington, NSW, Australia: School of Surveying, University of New South Wales*.
- Ardalan, A. A., Karimi, R., y Grafarend, E. W. (2009). A new reference equipotential surface, and reference ellipsoid for the planet mars. *Earth, Moon, and Planets*, 106(1):1–13. doi:[10.1007/s11038-009-9342-7](https://doi.org/10.1007/s11038-009-9342-7).
- Asgharzadeh, M., Von Frese, R., Kim, H., Leftwich, T., y Kim, J. (2007). Spherical prism gravity effects by Gauss-Legendre quadrature integration. *Geophysical Journal International*, 169(1):1–11.
- Barnes, G. y Lumley, J. (2011). Processing gravity gradient data. *Geophysics*, 76(2):I33–I47. doi:[10.1190/1.3548548](https://doi.org/10.1190/1.3548548).
- Barthelmes, F. (2013). Definition of functionals of the geopotential and their calculation from spherical harmonic models. *Scientific Technical Report; 09/02; ISSN 1610-0956*. doi:[10.2312/GFZ.B103-0902-26](https://doi.org/10.2312/GFZ.B103-0902-26).
- Binney, J. y Tremaine, S. (2008). *Galactic Dynamics*. Princeton University Press, second edition.
- Blakely, R. J. (1995). *Potential Theory in Gravity and Magnetic Applications*. Cambridge University Press. doi:[10.1017/cbo9780511549816](https://doi.org/10.1017/cbo9780511549816).
- Bouman, J., Ebbing, J., Fuchs, M., Sebera, J., Lieb, V., Szwilus, W., Haagmans, R., y Novak, P. (2016). Satellite gravity gradient grids for geophysics. *Scientific Reports*, 6(1). doi:[10.1038/srep21050](https://doi.org/10.1038/srep21050).
- Carlos, D. U. (2013). *Aplicação de Processamento e Inversão Geofísica 3D de Dados de Aerogradiometria da Gravidade na Estimação da Estrutura do Minério de Ferro no Sinclinal Gandarela - Quadrilátero Ferrífero - Minas Gerais*. Tesis de Doctorado, Observatório Nacional. URL <https://www.pinga-lab.org/thesis/carlos-phd.html>.

- Carlos, D. U., Uieda, L., y Barbosa, V. C. (2014). Imaging iron ore from the quadrilátero ferrífero (brazil) using geophysical inversion and drill hole data. *Ore Geology Reviews*, 61:268–285. doi:[10.1016/j.oregeorev.2014.02.011](https://doi.org/10.1016/j.oregeorev.2014.02.011).
- Carlos, D. U., Uieda, L., y Barbosa, V. C. (2016). How two gravity-gradient inversion methods can be used to reveal different geologic features of ore deposit — a case study from the quadrilátero ferrífero (brazil). *Journal of Applied Geophysics*, 130:153–168. doi:[10.1016/j.jappgeo.2016.04.011](https://doi.org/10.1016/j.jappgeo.2016.04.011).
- Chandrasekhar, S. (1995). *Newton's Principia for the Common Reader*. Oxford University Press.
- Cockett, R., Kang, S., Heagy, L. J., Pidlisecky, A., y Oldenburg, D. W. (2015). SimPEG: An open source framework for simulation and gradient based parameter estimation in geophysical applications. *Computers & Geosciences*, 85:142–154. doi:[10.1016/j.cageo.2015.09.015](https://doi.org/10.1016/j.cageo.2015.09.015).
- Cordell, L. (1973). Gravity analysis using an exponential density-depth function – San Jacinto Graben, California. *Geophysics*, 38(4):684–690.
- Cordell, L. (1992). A scattered equivalent-source method for interpolation and gridding of potential-field data in three dimensions. *Geophysics*, 57(4):629–636. doi:[10.1190/1.1443275](https://doi.org/10.1190/1.1443275).
- Cowie, P. A. y Karner, G. D. (1990). Gravity effect of sediment compaction: examples from the North Sea and the Rhine Graben. *Earth and Planetary Science Letters*, 99(1):141–153.
- Dampney, C. N. G. (1969). The equivalent source technique. *Geophysics*, 34(1):39–53. doi:[10.1190/1.1439996](https://doi.org/10.1190/1.1439996).
- de la Varga, M., Schaaf, A., y Wellmann, F. (2019). GemPy 1.0: open-source stochastic geological modeling and inversion. *Geoscientific Model Development*, 12(1):1–32. doi:[10.5194/gmd-12-1-2019](https://doi.org/10.5194/gmd-12-1-2019).
- Emilia, D. A. (1973). Equivalent Sources Used As An Analytic Base For Processing Total Magnetic Field Profiles. *Geophysics*, 38(2):339–348. doi:[10.1190/1.1440344](https://doi.org/10.1190/1.1440344).
- Ferreira Melo, F. (2020). *Interpretações qualitativa e quantitativa de dados magnéticos*. Tesis de Doctorado, Observatório Nacional. URL <https://www.pinga-lab.org/thesis/melo-phd.html>.
- Franke, R. (1982). Smooth interpolation of scattered data by local thin plate splines. *Computers & Mathematics with Applications*, 8(4):273–281. doi:[10.1016/0898-1221\(82\)90009-8](https://doi.org/10.1016/0898-1221(82)90009-8).

- Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232. doi:[10.1214/aos/1013203451](https://doi.org/10.1214/aos/1013203451).
- Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378. doi:[10.1016/s0167-9473\(01\)00065-2](https://doi.org/10.1016/s0167-9473(01)00065-2).
- Fukushima, T. (2018). Accurate computation of gravitational field of a tesseroid. *Journal of Geodesy*, 92(12):1371. doi:[10.1007/s00190-018-1126-2](https://doi.org/10.1007/s00190-018-1126-2).
- Fukushima, T. (2020). Speed and accuracy improvements in standard algorithm for prismatic gravitational field. *Geophysical Journal International*, 222(3):1898–1908. doi:[10.1093/gji/ggaa240](https://doi.org/10.1093/gji/ggaa240).
- Gramfort, A., Luessi, M., Larson, E., Engemann, D. A., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., y Hämäläinen, M. S. (2013). MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, 7(267):1–13. doi:[10.3389/fnins.2013.00267](https://doi.org/10.3389/fnins.2013.00267).
- Grombein, T., Seitz, K., y Heck, B. (2013). Optimized formulas for the gravitational field of a tesseroid. *Journal of Geodesy*, 87(7):645–660. doi:[10.1007/s00190-013-0636-1](https://doi.org/10.1007/s00190-013-0636-1).
- Guspi, F., Introcaso, A., e Introcaso, B. (2004). Gravity-enhanced representation of measured geoid undulations using equivalent sources. *Geophysical Journal International*, 159(1):1–8. doi:[10.1111/j.1365-246X.2004.02364.x](https://doi.org/10.1111/j.1365-246X.2004.02364.x).
- Guspi, F. y Novara, I. (2009). Reduction to the pole and transformations of scattered magnetic data using Newtonian equivalent sources. *Geophysics*, 74(5):L67–L73. doi:[10.1190/1.3170690](https://doi.org/10.1190/1.3170690).
- Hansen, R. O. (1993). Interpretive gridding by anisotropic kriging. *Geophysics*, 58(10):1491–1497. doi:[10.1190/1.1443363](https://doi.org/10.1190/1.1443363).
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., y Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362. doi:[10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2).
- Heck, B. y Seitz, K. (2006). A comparison of the tesseroid, prism and point-mass approaches for mass reductions in gravity field modelling. *Journal of Geodesy*, 81(2):121–136. doi:[10.1007/s00190-006-0094-0](https://doi.org/10.1007/s00190-006-0094-0).
- Heine, C. (2007). *Formation and evolution of intracontinental basins*. Tesis de Doctorado, School of Geosciences, The University of Sydney, Australia. URL <http://www.earthbyte.org/Resources/ICONS/ch4.html#x1-960004.3.4>.

- Heiskanen, W. A. y Moritz, H. (1967). Physical geodesy. *Bulletin géodésique*, 86(1):491–492. doi:[10.1007/bf02525647](https://doi.org/10.1007/bf02525647).
- Hidalgo-Gato, M. C. y Barbosa, V. C. F. (2015). Edge detection of potential-field sources using scale-space monogenic signal: Fundamental principles. *GEOPHYSICS*, 80(5):J27–J36. doi:[10.1190/geo2015-0025.1](https://doi.org/10.1190/geo2015-0025.1).
- Hidalgo-Gato, M. C. y Barbosa, V. C. F. (2017). The monogenic signal of potential-field data: A Python implementation. *GEOPHYSICS*, 82(3):F9–F14. doi:[10.1190/geo2016-0099.1](https://doi.org/10.1190/geo2016-0099.1).
- Hildebrand, F. B. (1987). *Introduction to numerical analysis*. Courier Corporation.
- Hinze, W. J., von Frese, R. R. B., y Saad, A. H. (2009). *Gravity and Magnetic Exploration*. Cambridge University Press. doi:[10.1017/cbo9780511843129](https://doi.org/10.1017/cbo9780511843129).
- Hofmann-Wellenhof, B. y Moritz, H. (2005). *Physical Geodesy*. SpringerWien-NewYork.
- Howell, J. A., Schwarz, E., Spalletti, L. A., y Veiga, G. D. (2005). The Neuquén basin: an overview. *Geological Society, London, Special Publications*, 252(1):1–14.
- Hoyer, S. y Hamman, J. (2017). xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, 5(1). doi:[10.5334/jors.148](https://doi.org/10.5334/jors.148).
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95. doi:[10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55).
- Imamoto, A. y Tang, B. (2008). A recursive descent algorithm for finding the optimal minimax piecewise linear approximation of convex functions. En *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*. IEEE. doi:[10.1109/wcecs.2008.42](https://doi.org/10.1109/wcecs.2008.42).
- Jayaram, V. y Barachant, A. (2018). MOABB: trustworthy algorithm benchmarking for BCIs. *Journal of Neural Engineering*, 15(6):066011. doi:[10.1088/1741-2552/aadea0](https://doi.org/10.1088/1741-2552/aadea0).
- Jirigalatu, J. y Ebbing (2019). A fast equivalent source method for airborne gravity gradient data. *Geophysics*, 84(5):G75–G82. doi:[10.1190/geo2018-0366.1](https://doi.org/10.1190/geo2018-0366.1).
- Ketkov, Y. L. (1969). Optimal methods of piecewise-linear approximation. *Soviet Radiophysics*, 9(6):692–695. doi:[10.1007/bf01038576](https://doi.org/10.1007/bf01038576).
- Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., y Willing, C. (2016). Jupyter Notebooks – a publishing format for reproducible computational workflows. En Loizides, F. y Schmidt, B., editores, *Positioning*

- and Power in Academic Publishing: Players, Agents and Agendas, páginas 87 – 90. IOS Press.
- Kother, L., Hammer, M. D., Finlay, C. C., y Olsen, N. (2015). An equivalent source method for modelling the global lithospheric magnetic field. *Geophysical Journal International*, 203(1):553–566. doi:[10.1093/gji/ggv317](https://doi.org/10.1093/gji/ggv317).
- Ku, C. C. (1977). A direct computation of gravity and magnetic anomalies caused by 2- and 3-dimensional bodies of arbitrary shape and arbitrary magnetic polarization by equivalent-point method and a simplified cubic spline. *GEOPHYSICS*, 42(3):610–622. doi:[10.1190/1.1440732](https://doi.org/10.1190/1.1440732).
- Lam, S. K., Pitrou, A., y Seibert, S. (2015). Numba. En *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC - LLVM '15*. ACM Press. doi:[10.1145/2833157.2833162](https://doi.org/10.1145/2833157.2833162).
- Leão, J. W. D. y Silva, J. B. C. (1989). Discrete linear transformations of potential field data. *Geophysics*, 54(4):497–507. doi:[10.1190/1.1442676](https://doi.org/10.1190/1.1442676).
- Li, D., Liang, Q., Du, J., Sun, S., Zhang, Y., y Chen, C. (2020). Transforming Total-Field Magnetic Anomalies Into Three Components Using Dual-Layer Equivalent Sources. *Geophysical Research Letters*, 47(3). doi:[10.1029/2019gl084607](https://doi.org/10.1029/2019gl084607).
- Li, X. y Götze, H.-J. (2001a). Ellipsoid, geoid, gravity, geodesy, and geophysics. *GEOPHYSICS*, 66(6):1660–1668. doi:[10.1190/1.1487109](https://doi.org/10.1190/1.1487109).
- Li, X. y Götze, H.-J. (2001b). Ellipsoid, geoid, gravity, geodesy, and geophysics. *Geophysics*, 66(6):1660–1668. doi:[10.1190/1.1487109](https://doi.org/10.1190/1.1487109).
- Li, Y. y Oldenburg, D. W. (2010). Rapid construction of equivalent sources using wavelets. *Geophysics*, 75(3):L51–L59. doi:[10.1190/1.3378764](https://doi.org/10.1190/1.3378764).
- Li, Z., Hao, T., Xu, Y., y Xu, Y. (2011). An efficient and adaptive approach for modeling gravity effects in spherical coordinates. *Journal of Applied Geophysics*, 73(3):221–231. doi:[10.1016/j.jappgeo.2011.01.004](https://doi.org/10.1016/j.jappgeo.2011.01.004).
- Lin, M. y Denker, H. (2018). On the computation of gravitational effects for tesseroids with constant and linearly varying density. *Journal of Geodesy*, 93(5):723–747. doi:[10.1007/s00190-018-1193-4](https://doi.org/10.1007/s00190-018-1193-4).
- Lin, M. y Denker, H. (2019). On the computation of gravitational effects for tesseroids with constant and linearly varying density. *Journal of Geodesy*, 93(5):723–747. doi:[10.1007/s00190-018-1193-4](https://doi.org/10.1007/s00190-018-1193-4).
- Martinez, C. y Li, Y. (2016). Denoising of gravity gradient data using an equivalent source technique. *GEOPHYSICS*, 81(4):G67–G79. doi:[10.1190/geo2015-0379.1](https://doi.org/10.1190/geo2015-0379.1).

- Maxant, J. (1980). Variation of density with rock type, depth, and formation in the Western Canada Basin from density logs. *Geophysics*, 45(6):1061–1076.
- May, R., Arms, S., Marsh, P., Bruning, E., Leeman, J., Bruick, Z., y Camron, M. D. (2016). Metpy. doi:[10.5065/D6WW7G29](https://doi.org/10.5065/D6WW7G29).
- Mendonça, C. A. y Silva, J. B. C. (1994). The equivalent data concept applied to the interpolation of potential field data. *Geophysics*, 59(5):722–732. doi:[10.1190/1.1443630](https://doi.org/10.1190/1.1443630).
- Menke, W. (1989). *Geophysical Data Analysis: Discrete Inverse Theory*. Academic Press. ISBN 9780080507323.
- Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., y Scopatz, A. (2017). SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3:e103. doi:[10.7717/peerj-cs.103](https://doi.org/10.7717/peerj-cs.103).
- Mikuška, J., Paštka, R., y Marušiak, I. (2006). Estimation of distant relief effect in gravimetry. *Geophysics*, 71(6):J59–J69.
- Millman, K. J. y Aivazis, M. (2011). Python for scientists and engineers. 13(2):9–12. doi:[10.1109/mcse.2011.36](https://doi.org/10.1109/mcse.2011.36).
- Nagy, D., Papp, G., y Benedek, J. (2000). The gravitational potential and its derivatives for the prism. *Journal of Geodesy*, 74(7-8):552–560. doi:[10.1007/s001900000116](https://doi.org/10.1007/s001900000116).
- Nagy, D., Papp, G., y Benedek, J. (2002). Corrections to "the gravitational potential and its derivatives for the prism". *Journal of Geodesy*, 76(8):475–475. doi:[10.1007/s00190-002-0264-7](https://doi.org/10.1007/s00190-002-0264-7).
- Nakatsuka, T. y Okuma, S. (2006). Reduction of magnetic anomaly observations from helicopter surveys at varying elevations. *Exploration Geophysics*, 37(1):121–128. doi:[10.1071/EG06121](https://doi.org/10.1071/EG06121).
- Oliphant, T. E. (2007). Python for scientific computing. 9(3):10–20. doi:[10.1109/mcse.2007.58](https://doi.org/10.1109/mcse.2007.58).
- Oliveira, V. C., Sales, D. P., Barbosa, V. C. F., y Uieda, L. (2015). Estimation of the total magnetization direction of approximately spherical bodies. *Nonlinear Processes in Geophysics*, 22(2):215–232. doi:[10.5194/npg-22-215-2015](https://doi.org/10.5194/npg-22-215-2015).
- Oliveira, Jr., V. C., Barbosa, V. C. F., y Uieda, L. (2013). Polynomial equivalent layer. *Geophysics*, 78(1):G1–G13. doi:[10.1190/geo2012-0196.1](https://doi.org/10.1190/geo2012-0196.1).

- Oliveira Jr, V. C., Uieda, L., y Barbosa, V. C. F. (2018). Should geophysicists use the gravity disturbance or the anomaly? doi:[10.5281/zenodo.1255306](https://doi.org/10.5281/zenodo.1255306).
- Oliveira Jr, V. C., Uieda, L., y Barbosa, V. C. F. (2021a). Sketch of the gravity and normal gravity vectors on the earth's surface, the geoid, and the reference ellipsoid. doi:[10.6084/m9.figshare.15044259.v1](https://doi.org/10.6084/m9.figshare.15044259.v1).
- Oliveira Jr, V. C., Uieda, L., y Barbosa, V. C. F. (2021b). Sketch of three coordinate systems: Geocentric Cartesian, Geocentric Geodetic, and Topocentric Cartesian. doi:[10.6084/m9.figshare.15044241.v1](https://doi.org/10.6084/m9.figshare.15044241.v1).
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., y Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- Perez, F. y Granger, B. E. (2007). IPython: A system for interactive scientific computing. 9(3):21–29. doi:[10.1109/mcse.2007.53](https://doi.org/10.1109/mcse.2007.53).
- Perez, F., Granger, B. E., y Hunter, J. D. (2011). Python: An ecosystem for scientific computing. 13(2):13–21. doi:[10.1109/mcse.2010.119](https://doi.org/10.1109/mcse.2010.119).
- Rao, C. V., Chakravarthi, V., y Raju, M. (1993). Parabolic density function in sedimentary basin modelling. *pure and applied geophysics*, 140(3):493–501.
- Rao, C. V., Chakravarthi, V., y Raju, M. (1994). Forward modeling: Gravity anomalies of two-dimensional bodies of arbitrary shape with hyperbolic and parabolic density functions. *Computers & Geosciences*, 20(5):873–880.
- Rao, D. B. (1986). Modelling of sedimentary basins from gravity anomalies with variable density contrast. *Geophysical Journal International*, 84(1):207–212.
- Reis, A. L. A., Oliveira, V. C., Yokoyama, E., Bruno, A. C., y Pereira, J. M. B. (2016). Estimating the magnetization distribution within rectangular rock samples. *Geochemistry, Geophysics, Geosystems*, 17(8):3350–3374. doi:[10.1002/2016gc006329](https://doi.org/10.1002/2016gc006329).
- Rücker, C., Günther, T., y Wagner, F. M. (2017). pyGIMLi: An open-source library for modelling and inversion in geophysics. *Computers and Geosciences*, 109:106–123. doi:[10.1016/j.cageo.2017.07.011](https://doi.org/10.1016/j.cageo.2017.07.011).
- Sales, D. P. (2014). Estimativa do vetor de magnetização total de corpos aproximadamente esféricos. Tesis de Maestría, Observatório Nacional. URL <https://www.pinga-lab.org/thesis/daiana-msc.html>.

- Sandwell, D. T. (1987). Biharmonic spline interpolation of GEOS-3 and SEASAT altimeter data. *Geophysical Research Letters*, 14(2):139–142. doi:[10.1029/GL014i002p00139](https://doi.org/10.1029/GL014i002p00139).
- Sandwell, D. T. y Wessel, P. (2016). Interpolation of 2-d vector data using constraints from elasticity. *Geophysical Research Letters*, 43(20):10,703–10,709. doi:[10.1002/2016gl070340](https://doi.org/10.1002/2016gl070340).
- Sansò, F. y Sideris, M. G., editores (2013). *Geoid Determination*. Springer Berlin Heidelberg. doi:[10.1007/978-3-540-74700-0](https://doi.org/10.1007/978-3-540-74700-0).
- Shapero, D., Badgeley, J., y Hoffman, A. (2020). icepack: glacier flow modeling with the finite element method in Python. doi:[10.5281/zenodo.1205640](https://doi.org/10.5281/zenodo.1205640).
- Sigismondi, M. E. (2012). *Estudio de la deformación litosférica de la cuenca Neuquina: estructura termal, datos de gravedad y sísmica de reflexión*. Tesis de Doctorado, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires. URL http://digital.bl.fcen.uba.ar/Download/Tesis/Tesis_5361_Sigismondi.pdf.
- Silva, J. B. C. (1986). Reduction to the pole as an inverse problem and its application to low-latitude anomalies. *Geophysics*, 51(2):369–382. doi:[10.1190/1.1442096](https://doi.org/10.1190/1.1442096).
- Siqueira, F. C. L., Jr., V. C. O., y Barbosa, V. C. F. (2017). Fast iterative equivalent-layer technique for gravity data processing: A method grounded on excess mass constraint. *Geophysics*, 82(4):G57–G69. doi:[10.1190/geo2016-0332.1](https://doi.org/10.1190/geo2016-0332.1).
- Smith, W. H. F. y Wessel, P. (1990). Gridding with continuous curvature splines in tension. *Geophysics*, 55(3):293–305. doi:[10.1190/1.1442837](https://doi.org/10.1190/1.1442837).
- Soler, S., Balza Morales, A., y Pesce, A. (2021a). Processing gravity and magnetic data with harmonica. URL <https://youtu.be/0bxZcCAr6bw>.
- Soler, S. R. (2015). Métodos espectrales para la determinación de la profundidad del punto de curie y el espesor elástico de la corteza terrestre. doi:[10.6084/m9.figshare.16752466.v1](https://doi.org/10.6084/m9.figshare.16752466.v1).
- Soler, S. R., Pesce, A., Gimenez, M. E., y Uieda, L. (2019a). Gravitational field calculation in spherical coordinates using variable densities in depth. *Geophysical Journal International*, 218(3):2150–2164. doi:[10.1093/gji/ggz277](https://doi.org/10.1093/gji/ggz277).
- Soler, S. R., Pesce, A., Gimenez, M. E., y Uieda, L. (2019b). Source code and data for “Gravitational field calculation in spherical coordinates using variable densities in depth”. doi:[10.6084/m9.figshare.8239622](https://doi.org/10.6084/m9.figshare.8239622).
- Soler, S. R. y Uieda, L. (2021a). Gradient-boosted equivalent sources. *Geophysical Journal International*, 227(3):1768–1783. doi:[10.1093/gji/ggab297](https://doi.org/10.1093/gji/ggab297).

- Soler, S. R. y Uieda, L. (2021b). Supplementary material for "Gradient-boosted equivalent sources". doi:[10.6084/m9.figshare.13604360](https://doi.org/10.6084/m9.figshare.13604360).
- Soler, S. R., Uieda, L., Oliveira Jr, V. C., Shea, N., y Pesce, A. (2021b). Harmonica: Forward modeling, inversion, and processing gravity and magnetic data. doi:[10.5281/zenodo.3628741](https://doi.org/10.5281/zenodo.3628741).
- Stockwell, J. W. (1999). The CWP/SU: Seismic un*x package,. 25(4):415–419. doi:[10.1016/s0098-3004\(98\)00145-9](https://doi.org/10.1016/s0098-3004(98)00145-9).
- Sullivan, C. y Kaszynski, A. (2019). PyVista: 3d plotting and mesh analysis through a streamlined interface for the visualization toolkit (VTK). 4(37):1450. doi:[10.21105/joss.01450](https://doi.org/10.21105/joss.01450).
- Takahashi, D., Jr., V. C. O., y Barbosa, V. C. F. (2020). Convolutional equivalent layer for gravity data processing. *Geophysics*, 85(6):G129–G141. doi:[10.1190/geo2019-0826.1](https://doi.org/10.1190/geo2019-0826.1).
- Talwani, M., Worzel, J. L., y Landisman, M. (1959). Rapid gravity computations for two-dimensional bodies with application to the mendocino submarine fracture zone. *Journal of Geophysical Research*, 64(1):49–59. doi:[10.1029/jz064i001p00049](https://doi.org/10.1029/jz064i001p00049).
- The ObsPy Development Team (2019). Obspy 1.1.1. doi:[10.5281/zenodo.1040770](https://doi.org/10.5281/zenodo.1040770).
- Tikhonov, A. N. (1977). *Solutions of Ill Posed Problems*. Vh Winston. ISBN 0-470-99124-0.
- Tozer, B., Sandwell, D. T., Smith, W. H. F., Olson, C., Beale, J. R., y Wessel, P. (2019). Global Bathymetry and Topography at 15 Arc Sec: SRTM15+. *Earth and Space Science*, 6(10):1847–1864. doi:[10.1029/2019ea000658](https://doi.org/10.1029/2019ea000658).
- Tscherning, C. C. (2015). Least-squares collocation. En *Encyclopedia of Geodesy*, páginas 1–5. Springer International Publishing. doi:[10.1007/978-3-319-02370-0_51-1](https://doi.org/10.1007/978-3-319-02370-0_51-1).
- Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., y Norman, M. L. (2010). yt: A MULTI-CODE ANALYSIS TOOLKIT FOR ASTROPHYSICAL SIMULATION DATA. 192(1):9. doi:[10.1088/0067-0049/192/1/9](https://doi.org/10.1088/0067-0049/192/1/9).
- Uieda, L., Soler, S., y Pesce, A. (2021a). Design useful tools that do one thing well and work together: rediscovering the UNIX philosophy while building the Fatiando a Terra project. URL <https://github.com/fatiando/agu2021>.
- Uieda, L. (2015). A tesseroid (spherical prism) in a geocentric coordinate system with a local-north-oriented coordinate system. doi:[10.6084/m9.figshare.1495525](https://doi.org/10.6084/m9.figshare.1495525).

- Uieda, L. (2016). *Forward modeling and inversion of gravitational fields in spherical coordinates*. Tesis de Doctorado, Observatório Nacional. URL <https://www.leouieda.com/about/phd.html>.
- Uieda, L. (2018). Verde: Processing and gridding spatial data using Green's functions. *Journal of Open Source Software*, 3(29):957. doi:[10.21105/joss.00957](https://doi.org/10.21105/joss.00957).
- Uieda, L. (2021). Ground gravity data compilation for Australia filtered by survey quality and packaged in CF-compliant netCDF. doi:[10.6084/m9.figshare.13643837](https://doi.org/10.6084/m9.figshare.13643837).
- Uieda, L. y Barbosa, V. C. (2017). Fast nonlinear gravity inversion in spherical coordinates with application to the South American Moho. *Geophys J Int*, 208(1):162–176. doi:[10.1093/gji/ggw390](https://doi.org/10.1093/gji/ggw390).
- Uieda, L. y Barbosa, V. C. F. (2012a). Robust 3d gravity gradient inversion by planting anomalous densities. *GEOPHYSICS*, 77(4):G55–G66. doi:[10.1190/geo2011-0388.1](https://doi.org/10.1190/geo2011-0388.1).
- Uieda, L. y Barbosa, V. C. F. (2012b). Supplementary material to robust 3d gravity gradient inversion by planting anomalous densities"by leonardo uieda and valéria c. f. barbosa. doi:[10.6084/m9.figshare.91574](https://doi.org/10.6084/m9.figshare.91574).
- Uieda, L., Barbosa, V. C. F., y Braitenberg, C. (2016). Tesseroids: Forward-modeling gravitational fields in spherical coordinates. *GEOPHYSICS*, 81(5):F41–F48. doi:[10.1190/geo2015-0204.1](https://doi.org/10.1190/geo2015-0204.1).
- Uieda, L., Li, L., Soler, S., y Pesce, A. (2021b). Fatiando a terra: Open-source tools for geophysics. URL <https://github.com/fatiando/2021-gsh>.
- Uieda, L., Oliveira, V., y Barbosa, V. (2013). Modeling the earth with Fatiando a Terra. En van der Walt, S., Millman, J., y Huff, K., editores, *Proceedings of the 12th Python in Science Conference*, páginas 96–103. SciPy. doi:[10.25080/majora-8b375195-010](https://doi.org/10.25080/majora-8b375195-010).
- Uieda, L., Oliveira, V. C., y Barbosa, V. C. F. (2014). Geophysical tutorial: Euler deconvolution of potential-field data. *The Leading Edge*, 33(4):448–450. doi:[10.1190/tle33040448.1](https://doi.org/10.1190/tle33040448.1).
- Uieda, L. y Soler, S. (2020a). From scattered data to gridded products using Verde. URL <https://youtu.be/-xZdNdVzm3E>.
- Uieda, L., Soler, S., Rampin, R., van Kemenade, H., Turk, M., Shapero, D., Banihirwe, A., y Leeman, J. (2020a). Pooch: A friend to fetch your data files. *Journal of Open Source Software*, 5(45):1943. doi:[10.21105/joss.01943](https://doi.org/10.21105/joss.01943).
- Uieda, L. y Soler, S. R. (2020b). Boule v0.2.0: Reference ellipsoids for geodesy, geophysics, and coordinate calculations. doi:[10.5281/zenodo.3939204](https://doi.org/10.5281/zenodo.3939204).

- Uieda, L., Soler, S. R., Pesce, A., Perozzi, L., y Wieczorek, M. A. (2021c). Harmonica and boule: Modern python tools for geophysical gravimetry. doi:[10.5194/egusphere-egu21-8291](https://doi.org/10.5194/egusphere-egu21-8291).
- Uieda, L., Tian, D., Leong, W. J., Toney, L., Newton, T., y Wessel, P. (2020b). PyGMT: A Python interface for the Generic Mapping Tools. doi:[10.5281/zenodo.4253459](https://doi.org/10.5281/zenodo.4253459).
- van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., y Yu, T. (2014). scikit-image: image processing in Python. *PeerJ*, 2:e453. doi:[10.7717/peerj.453](https://doi.org/10.7717/peerj.453).
- Vandewalle, J. (1975). On the calculation of the piecewise linear approximation to a discrete function. *IEEE Transactions on Computers*, C-24(8):843–846. doi:[10.1109/t-c.1975.224320](https://doi.org/10.1109/t-c.1975.224320).
- Vermeille, H. (2002). Direct transformation from geocentric coordinates to geodetic coordinates. *Journal of Geodesy*, 76(8):451–454. doi:[10.1007/s00190-002-0273-6](https://doi.org/10.1007/s00190-002-0273-6).
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., y SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272. doi:[10.1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2).
- Vladimirov, V. S. (1979). *Generalized functions in mathematical physics*. MIR.
- von Frese, R. R., Hinze, W. J., y Braile, L. W. (1981). Spherical earth gravity and magnetic anomaly analysis by equivalent point source inversion. *Earth and Planetary Science Letters*, 53(1):69–83. doi:[10.1016/0012-821x\(81\)90027-3](https://doi.org/10.1016/0012-821x(81)90027-3).
- von Frese, R. R. B., Ravat, D. N., Hinze, W. J., y McGue, C. A. (1988). Improved inversion of geopotential field anomalies for lithospheric investigations. *Geophysics*, 53(3):375–385. doi:[10.1190/1.1442471](https://doi.org/10.1190/1.1442471).
- Welford, J. K., Shannon, P. M., O'Reilly, B. M., y Hall, J. (2010). Lithospheric density variations and Moho structure of the Irish Atlantic continental margin from constrained 3-D gravity inversion. *Geophysical Journal International*, 183(1):79–95.
- Werthmüller, D. (2017). An open-source full 3D electromagnetic modeler for 1D VTI media in Python: empymod. *Geophysics*, 82(6):WB9–WB19. doi:[10.1190/geo2016-0626.1](https://doi.org/10.1190/geo2016-0626.1).

- Werthmüller, D., Mulder, W. A., y Slob, E. C. (2019). emg3d: A multigrid solver for 3d electromagnetic diffusion. *Journal of Open Source Software*, 4(39):1463. doi:[10.21105/joss.01463](https://doi.org/10.21105/joss.01463).
- Wes McKinney (2010). Data Structures for Statistical Computing in Python. En Stéfan van der Walt y Jarrod Millman, editores, *Proceedings of the 9th Python in Science Conference*, páginas 56 – 61. doi:[10.25080/Majora-92bf1922-00a](https://doi.org/10.25080/Majora-92bf1922-00a).
- Wessel, P., Luis, J. F., Uieda, L., Scharroo, R., Wobbe, F., Smith, W. H. F., y Tian, D. (2019). The generic mapping tools version 6. 20(11):5556–5564. doi:[10.1029/2019gc008515](https://doi.org/10.1029/2019gc008515).
- Wessel, P. y Smith, W. H. F. (1991). Free software helps map and display data. 72(41):441–441. doi:[10.1029/90eo00319](https://doi.org/10.1029/90eo00319).
- Wieczorek, M. (2015). Gravity and topography of the terrestrial planets. páginas 153–193. Elsevier. doi:[10.1016/b978-0-444-53802-4.00169-x](https://doi.org/10.1016/b978-0-444-53802-4.00169-x).
- Wieczorek, M. A. y Meschede, M. (2018). SHTools: Tools for working with spherical harmonics. 19(8):2574–2592. doi:[10.1029/2018gc007529](https://doi.org/10.1029/2018gc007529).
- Wild-Pfeiffer, F. (2008). A comparison of different mass elements for use in gravity gradiometry. *Journal of Geodesy*, 82(10):637–653. doi:[10.1007/s00190-008-0219-8](https://doi.org/10.1007/s00190-008-0219-8).
- Wilson, G., Aruliah, D. A., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., Haddock, S. H. D., Huff, K. D., Mitchell, I. M., Plumley, M. D., Waugh, B., White, E. P., y Wilson, P. (2014). Best practices for scientific computing. *PLoS Biology*, 12(1):e1001745. doi:[10.1371/journal.pbio.1001745](https://doi.org/10.1371/journal.pbio.1001745).
- Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., y Teal, T. K. (2017). Good enough practices in scientific computing. *PLOS Computational Biology*, 13(6):e1005510. doi:[10.1371/journal.pcbi.1005510](https://doi.org/10.1371/journal.pcbi.1005510).
- Wynne, P. (2018). NetCDF Ground Gravity Point Surveys Collection. doi:[10.26186/5C1987FA17078](https://doi.org/10.26186/5C1987FA17078).
- Zhang, J., Zhong, B., Zhou, X., y Dai, Y. (2001). Gravity anomalies of 2D bodies with variable density contrast. *Geophysics*, 66(3):809–813.