TRABAJO ESPECIAL DE GRADO

DESARROLLO DE UNA HERRAMIENTA COMPUTACIONAL PARA LA CARACTERIZACIÓN DE YACIMIENTOS EMPLEANDO SÍSMICA PRE-APILADA

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Rivera C., Luisdaniel A.
Para optar al título De Ingeniero Geofísico

Ciudad Universitaria de Caracas. Noviembre de 2019

TRABAJO ESPECIAL DE GRADO

DESARROLLO DE UNA HERRAMIENTA COMPUTACIONAL PARA LA CARACTERIZACIÓN DE YACIMIENTOS EMPLEANDO SÍSMICA PRE-APILADA

TUTOR ACADÉMICO: Prof. Mariano Arnaiz Rodríguez
TUTOR INDUSTRIAL: MSc. Rafael Pinto

Presentado ante la Ilustre
Universidad Central de Venezuela
Por el Br. Rivera C., Luisdaniel A.
Para optar al título De Ingeniero Geofísico

Ciudad Universitaria de Caracas. Noviembre de 2019

Caracas, 04 de noviembre de 2019

Los abajo firmantes, miembros del jurado designado por el Consejo de Escuela de Geología, Minas y Geofísica, de la Facultad de Ingeniería de la Universidad Central de Venezuela, para evaluar el Trabajo Especial de Grado presentado por el Br. **Luisdaniel Antonio Rivera Curbelo,** titulado:

Desarrollo de una herramienta computacional para la caracterización de yacimientos empleando sísmica pre-apilada

Consideran que el mismo cumple con los requisitos exigidos por el plan de estudios conducente al título de Ingeniero Geofísico, y sin que ello signifique que se hacen solidarios con las ideas del autor, lo declaran **APROBADO**.

Prof. Janckarlos Reyes
Jurado

Prof. Vincenzo De Lisa
Jurado

Prof. José Cavada
(Tutor académico)

DEDICATORIA

A mis Padres que me han dado todo en esta vida,

Este es el primer, pero no el último reconocimiento que les voy a dedicar.

AGRADECIMIENTOS

En primer lugar a Dios que me ha dado la salud, el rumbo, cantidad de oportunidades de compartir con personas maravillosas y finalmente la voluntad y perseverancia para estudiar y ser mejor cada día.

A mi familia por el apoyo incondicional que me han suministrado a lo largo de este recorrido. En especial a mi Madre, María Magdalena por ser una mujer llena de amor, ternura, justicia, sabiduría y por todos los sacrificios que hace por mantener mi sonrisa. A mi Padre, Daniel Rivera por su perseverancia y personalidad inquebrantable, que siempre me ha dado fuerzas cuando más lo he necesitado para que no pierda el rumbo… aunque lo trate mal. A mi hermano, Daniel Rivera por ser el hombre al que admiro y que ha servido como motivación, inspiración y ejemplo de excelencia. Sin ustedes, este proyecto no hubiese sido posible.

A los Colegas, por haber estado presentes en esta etapa de mi vida y por haber formado parte de tantas risas y experiencias que me gustaría contarles a mis nietos.

A Ofimanía por ser la patrocinante oficial de mi formación personal y profesional a lo largo de mi corta vida. Empresa que me ha visto crecer y de la cual estoy orgulloso de considerar como una gran familia.

A la Universidad Central de Venezuela por haberme brindado la oportunidad de ser UCVista, de conocer personas valiosas y por mostrarme el significado tras “La casa que vence las sombras”.

A mis profesores, en particular a Janckarlos Reyes, Ricardo Alezones, Orlando Méndez, Inirida Rodriguez, Yaraixa Pérez y José Cavada, que son y seguirán siendo los pilares fundamentales de mi desarrollo profesional por su apoyo ininterrumpido e inigualable sabiduría. Especiales agradecimientos al profesor Orlando Méndez, que me transmitió su deseo por la excelencia y su pasión por la geología y la enseñanza; a Janckarlos Reyes que nos acompañó en la etapa mas difícil de "Geofísica de Parto" y finalmente Rosa Jimenez y a Inirida Rodriguez que no nos abandonaron en los Llanos Venezolanos.

A todos mis amigos UCVistas, que hicieron de mi vida como estudiante una actividad placentera. Especiales agradecimientos a Noel Crasto y Nicole Oliveira por ser compañeros extremadamente valiosos e inigualables que me han acompañado desde los inicios de la carrera; a todos los estudiantes del Departamento de Geofísica, por ser la mejor de las comunidades al disponer de personas con una inmensa calidad humana que permiten considerarlos como una familia y, a nuestros espacios, como un segundo hogar.

A mis tutores, Mariano Arnaiz Rodríguez por exigirme excelencia e inculcarme el criterio de un investigador. A Rafael Pinto, por todo su apoyo incondicional, por creer y depositar su fe en mí y por todo el tiempo que me dedicó a lo largo de esta investigación y finalmente, a Vincenzo de Lisa por ayudarme a culminar la redacción de este trabajo. Sin ustedes, este proyecto no hubiese sido posible.

A la comunidad de Slack, en especial a Jørgen Kvalsvik, Matt Hall y James Bednar por facilitarme los conocimientos y tips útiles para el desarrollo de este trabajo de investigación.

Y Finalmente, a mi novia María Guerrero, mi *Teacher* favorita, por quererme, consentirme y no menos importante, aguantarme estos últimos meses. Espero que aprendas a nadar rápido para que me acompañes… por que aún nos queda muchísimo por patalear.

Rivera C. Luisdaniel A.

DESARROLLO DE UNA HERRAMIENTA COMPUTACIONAL PARA LA CARACTERIZACIÓN DE YACIMIENTOS EMPLEANDO SÍSMICA PRE-APILADA

Tutor académico: Prof. Mariano Arnaiz Rodríguez,
Tutor industrial: MSc. Rafael Pinto
TEG, Caracas, U.C.V. Facultad de Ingeniería. Escuela de Geología, Minas y Geofísica. Departamento de Geofísica. Año 2019, 126 p.

Palabras claves: sísmica preapilada, AVO, caracterización de yacimientos, herramienta computacional, *Python.*

RESUMEN

El departamento de Geofísica de la Universidad Central de Venezuela no dispone de licencias vigentes ni de aplicaciones completamente libres que permitan realizar prácticas o investigaciones a nivel de sísmica preapilada. En tal sentido, se planteó desarrollar una herramienta computacional para el análisis de amplitudes sísmicas pre-apiladas en la caracterización de yacimientos, utilizando sistemas operativos, lenguajes, herramientas, bibliotecas, módulos y datos de libre acceso. La herramienta se desarrolló en *Ubuntu* versión 16.04 (*Xenial Xerus*), utilizando *Python* (v3.7) como lenguaje de programación, *Jupyter notebook* como intérprete y diversas utilidades completando 4 módulos de programación orientados a la organización y visualización de las trazas sísmicas preapiladas, además, del cálculo y despliegue de atributos AVO. Lo anterior se logró validar utilizando un subvolumen de 11x11 trazas sísmicas correspondientes al Proyecto *NW Shelf Australia Poseidon 3D.* La investigación demostró que es posible crear un entorno informático efectivo en la inspección,

visualización y caracterización por AVO mediante el uso de las bibliotecas: *SegyIO* (v1.6), *Pandas* (v0.24.2), *Scipy* (v1.2.1), *Holoviews* (v1.12.1), *Panel* (v0.5.1) y *Multiproccesing* (v16.6). Los flujos de trabajo generados a partir de los módulos de programación no son excluyentes de equipos de alto rendimiento, logrando verificar la existencia de arenas gasíferas, interpretadas en la literatura, a una profundidad en tiempo de 3250 ms (5000 m) sobre las coordenadas del pozo Kronos 1 en un equipo de 32 bits y 2Gb de RAM.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

INTRODUCCIÓN

1.1. Planteamiento del problema

La sísmica de reflexión es la herramienta más utilizada para la evaluación de yacimientos en la exploración y producción de hidrocarburos (Ikelle y Amundsen, 2005). En particular, el análisis de datos sísmicos pre-apilados como la variación de la amplitud con la distancia, conocido como AVO (por sus siglas en inglés, *Amplitude variation with Offset*), provee la técnica más robusta para la caracterización de yacimientos en ambientes clásticos sedimentarios (Avseth, Mukerji y Mavko, 2005; Ikelle y Amundsen, 2005; Russell, 2014). Para el análisis AVO, existen aplicaciones comerciales como *Petrel, Jason, y HampsonRussell* que cuentan con flujos de trabajo especializados en la interpretación cuantitativa destinada a la caracterización de yacimientos. Por su naturaleza comercial, estas aplicaciones no son de código abierto, por lo que resulta imposible conocer con exactitud los detalles en la implementación de la teoría disponible en la literatura.

La alta competitividad dentro de la industria petrolera además de estimular la constante innovación de las herramientas comerciales y la aplicación de la técnica AVO (Almutlaq y Margrave, 2010), ha impulsado el desarrollo tecnológico de toda comunidad geocientífica, incluyendo la creación de diversas corrientes de pensamiento como la Iniciativa para el Código Abierto. El Departamento de Geofísica de la Universidad Central de Venezuela no cuenta con licencias vigentes ni con un programa completamente libre y de fácil acceso que permita la ejecución de los flujos de trabajo necesarios para el análisis de AVO en dos y tres dimensiones, lo cual limita las investigaciones asociadas al campo de la interpretación sísmica. En

virtud de lo anterior, la presente investigación plantea desarrollar una herramienta de acceso libre para el análisis de amplitudes pre-apiladas siguiendo un patrón de desarrollo orientado a la reproducibilidad, con flujos de trabajo documentados, programados e interactivos.

La herramienta desarrollada en este trabajo de investigación constituye un aporte a nivel metodológico ya que cuenta con diferentes algoritmos basados en técnicas, procedimientos y flujos de trabajo utilizados tanto a nivel industrial como los que se enfatizan en la literatura. Así mismo, disponer de una herramienta gratuita, reproducible y de fácil acceso le concederá al Departamento de Geofísica la posibilidad de reducir su dependencia a aplicaciones comerciales. Se espera que, dada la reproducibilidad de la aplicación desarrollada, se formen las bases para la enseñanza del método sísmico orientado al uso de la técnica AVO y que fomente futuras investigaciones y colaboraciones a nivel académico e industrial.

1.2. Objetivos

1.2.1. Objetivo general

Desarrollar una herramienta computacional para el análisis de amplitudes sísmicas pre-apiladas en la caracterización de yacimientos.

1.2.2. Objetivos específicos

- Identificar herramientas que permitan la lectura e inspección del dato de cubos sísmicos.
- Diseñar una rutina de programación que simplifique el análisis de amplitudes pre-apiladas.
- Plantear ejemplos de flujos de trabajo que permitan estandarizar la investigación de las amplitudes de los datos pre-apilados.
- Comprobar el funcionamiento de la herramienta computacional utilizando datos de campo.

- Contrastar la reproducibilidad de los resultados de campo con aplicaciones comerciales.

1.3. Antecedentes

La Iniciativa de código abierto (OSI, por sus siglas en inglés *Open Source Initiative*) forma parte de una corriente de pensamiento que se fundamenta en la colaboración abierta al desarrollar programas y aplicaciones computacionales bajo la premisa de permitir que el conocimiento esté al alcance de todos, sin ningún tipo de restricción, para aumentar la velocidad de desarrollo de tecnologías. Dada la densidad de la comunidad científica que hace vida en la geociencia y al auge de la Ciencia de los Datos (*DataScience* por su nombre en inglés) se han codificado de diversas aplicaciones orientadas al desarrollo de las diferentes especialidades de la Geofísica (Cavada, 2018; *Free Software Foundation*, 2019 y *Open Source Iniciative*, 2019).

En el campo de la exploración e investigación sísmica, Bianco *et al.* (2013) codificaron y compilaron en *Python* el paquete *BRUGES* que contiene diversas ecuaciones geofísicas para aplicación de filtros, petrofísica, física de rocas, transformadas, resolución de la ecuación Zoeppritz y sus aproximaciones lineales, etc. Kvalsvik (2014) desarrolló la biblioteca *SegyIO* para los lenguajes *Python* y *Matlab* que permite la manipulación de datos sísmicos en formato SEG-Y para 2 y 3 dimensiones. Bianco y Hall (2015) utilizaron *SegyIO* para implementar *Seisplot*, una aplicación diseñada en *Python* para desplegar líneas sísmicas 2D. Pese a que estas son libres y de fácil acceso, no contienen un código amigable que permita su reproducibilidad sin conocimientos previos de programación avanzada además de que no son interactivos. Aunque existan este tipo de bibliotecas de caracterización de AVO, no han sido implementadas aplicaciones de libre acceso que combinen elementos de visualización con elementos de interpretación interactiva fundamentadas en la caracterización de yacimientos por AVO.

OpendTect, perteneciente a la compañía dGB, es una aplicación cuya licencia libre limita su uso a visualizaciones y análisis básico de datos apilados en 3D sin poder acceder a funciones especializadas como AVO (dGB *Earth Science*, 2019) en su versión *Free*. Además, el desarrollo de esta herramienta no es distribuido ni tampoco accesible a través de los medios actuales de desarrollo (*Github*), lo cual desincentiva la colaboración y el entendimiento del funcionamiento de la implementación.

1.4. Localización del dato de prueba

El proyecto *NW Shelf Australia Poseidon 3D* está localizado dentro la cuenca Browse ubicada en la región noroccidental de Australia, alrededor de las coordenadas geográficas (N 13°14'00", E 122°12'00") referenciadas al datum EPSG: 28351. La Figura 1.1 muestra la localización del levantamiento sísmico.

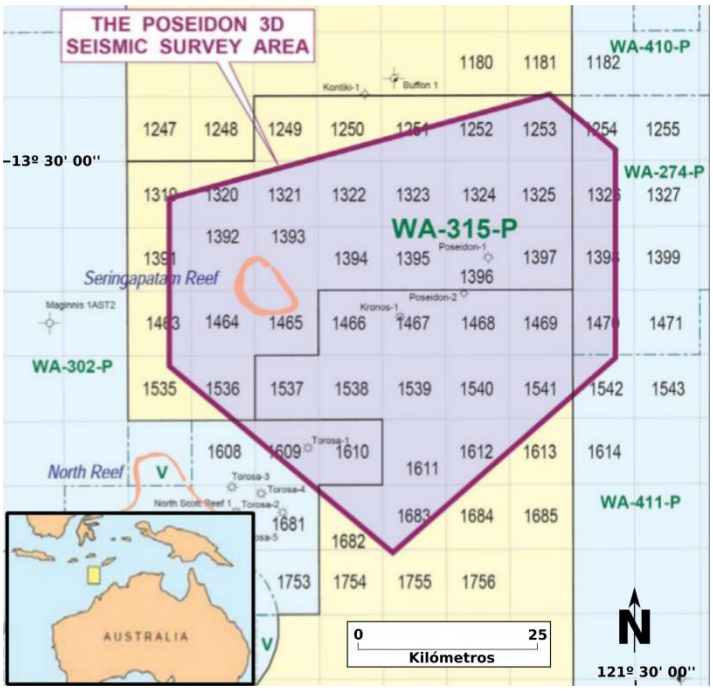

Figura 1.1. Ubicación del proyecto *NW Shelf Australia Poseidon 3D*. Se muestra el área de estudio delimitada por la poligonal púrpura (Modificado de ConocoPhillips, 2012).

CAPÍTULO II

MARCO GEOLÓGICO

La cuenca Browse (Figura 2.1) es una depresión topográfica y estructural de margen pasivo que se ubica en la región noroccidental de Australia. Esta cubre aproximadamente 140.000 km^2 y exhibe una orientación N45E. La cuenca yace entre la Meseta Scott y la Llanura Abisal Argo en el NO y el Bloque Kimberly al SE, limita al SO con la subcuenca Rowley perteneciente a la cuenca Roebuck y al NE con la subcuenca Vulcan de la cuenca Bonaparte. Dentro de esta cuenca se distinguen las subcuencas Caswell, Barcoo y Seringapatam, donde las dos primeras son catalogadas como principales y presentan depocentros de hasta 15 km (ConocoPhillips, 2012).

Se han identificado un total de 22 secuencias estratigráficas entre los períodos Devónico y Neógeno (Figura 2.2) a partir de interpretaciones de registros de pozos y levantamientos sísmicos realizados por diferentes investigadores, entre ellos AGSO *Browse Basin Proyect Team* (1997), ConocoPhillips (2011), ConocoPhillips (2012), *Geoscience Australia* (2014). Dado que la cuenca es una provincia probada de hidrocarburos con importantes campos de gas, condensado y pequeños descubrimientos de petróleo (AGSO, 1997) se hará especial énfasis en el intervalo BB5-BB7 que corresponde a la Formación Plover, intervalo que, de acuerdo con ConocoPhillips (2012), es uno de los objetivos primordiales en exploración además de ser una de las principales fuentes de gas de la cuenca.

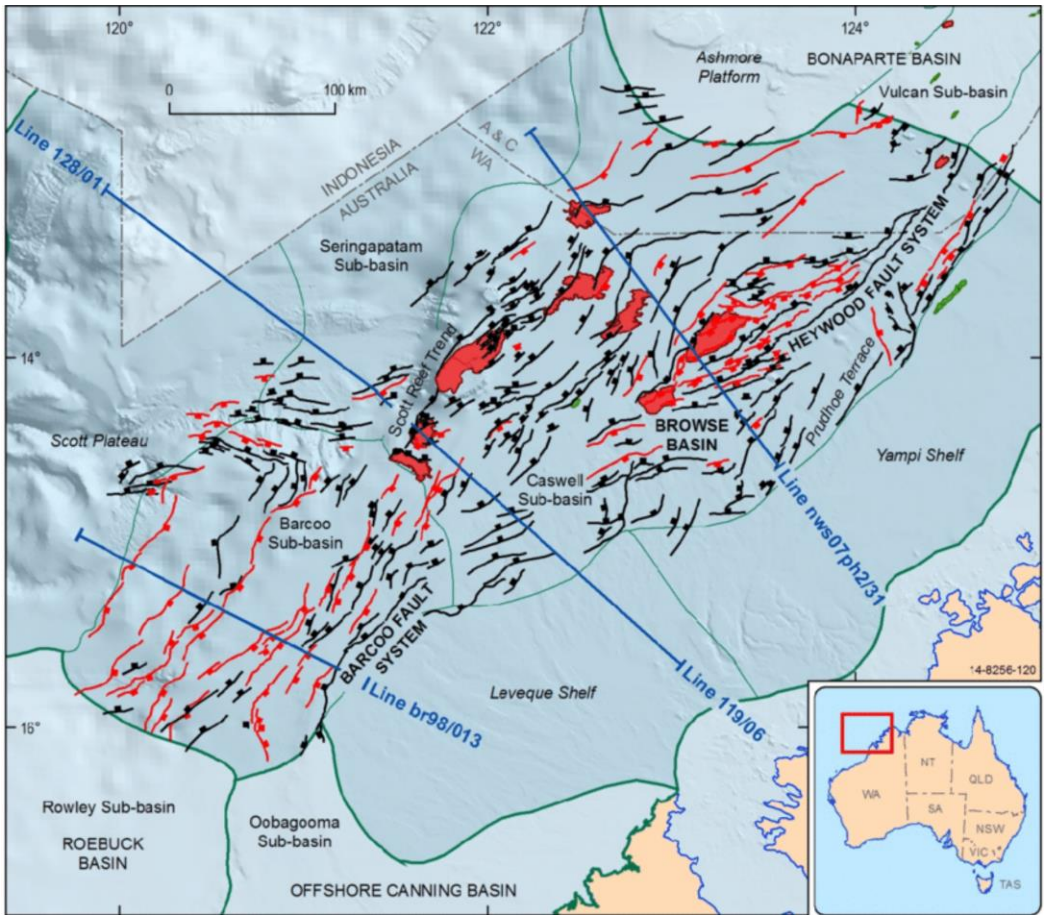

Figura 2.1. Mapa estructural de la Cuenca Browse. (Modificado de Abbott *et. al.*, 2016).

2.1. Evolución tectónica de la Cuenca Browse

Diversos autores (AGSO, 1997; Blevin y Baxter *et al*., 1998, *Geoscience Australia*, 2014; entre otros) coinciden en que la cuenca Browse ha atravesado seis fases tectónicas a lo largo del tiempo geológico:

- Entre el Carbonífero Tardío y Pérmico Temprano (322 ~ 279 M.A) ocurre la primera fase extensiva como consecuencia de la separación del Bloque Sibumasu del noroeste de Australia, que dio como resultado la formación de grábenes intracratónicos actualmente conocidos como las subcuencas Caswell y Barcoo.

- Del Pérmico al Triásico Medio (279 ~ 220 M.A) sucede la primera subsidencia térmica durante la fase post-rift que termina por la reactivación por compresión entre el Triásico Medio y el Jurásico Temprano (220 ~ 195 M.A). Este último período se caracteriza por el incremento de la actividad tectónica junto con la separación entre Argoland y Australia. El sistema de fallas asociadas a los bloques indujeron la tendencia estructural dominante (N45E) y generaron diversos elementos estructurales característicos de la cuenca como los anticlinales Buffon, Scott Reef – brecknock a consecuencia de la inversión de las subcuencas del Carbonífero.
- Entre el Jurásico Medio al Jurásico Tardío (195 ~ 161 M.A) ocurre un segundo pulso extensional marcado por fallamientos normales a pequeña escala en la región noroeste de la subcuenca Caswell, además del colapso de los anticlinales formados en el Triásico permitiendo el desarrollo del potencial hidrocarburífero en la presente subcuenca a consecuencia de la depositación de la Formación Plover (BB5-BB7).
- Entre el Jurásico Tardío al Cenozoico (161 ~ 12 M.A) ocurre una segunda subsidencia térmica caracterizada por reactivaciones menores dada la poca actividad tectónica y cambios eustáticos que permitieron la depositación de las secuencias finales del Jurásico y Cretácico hasta el Mioceno Tardío (BB8 a BB20 en Figura 2.2). Este período finaliza con el segundo pulso compresional que ocurre en el Mioceno Tardío (12 M.A ~ actualidad), marcado por la convergencia entre las placas Eurasia y Austroindia.

2.2. Consideraciones estratigráficas de las secuencias de interés

Se considerarán como formaciones de interés todas aquellas secuencias que han sido identificadas, estudiadas y relacionadas con los sistemas petrolíferos mostrados en la Figura 2.2 (W1, W2 y W3). La Tabla 2.1 muestra un resumen de las litologías y observaciones generales en cada una de las formaciones entre las secuencias 6 y 20).

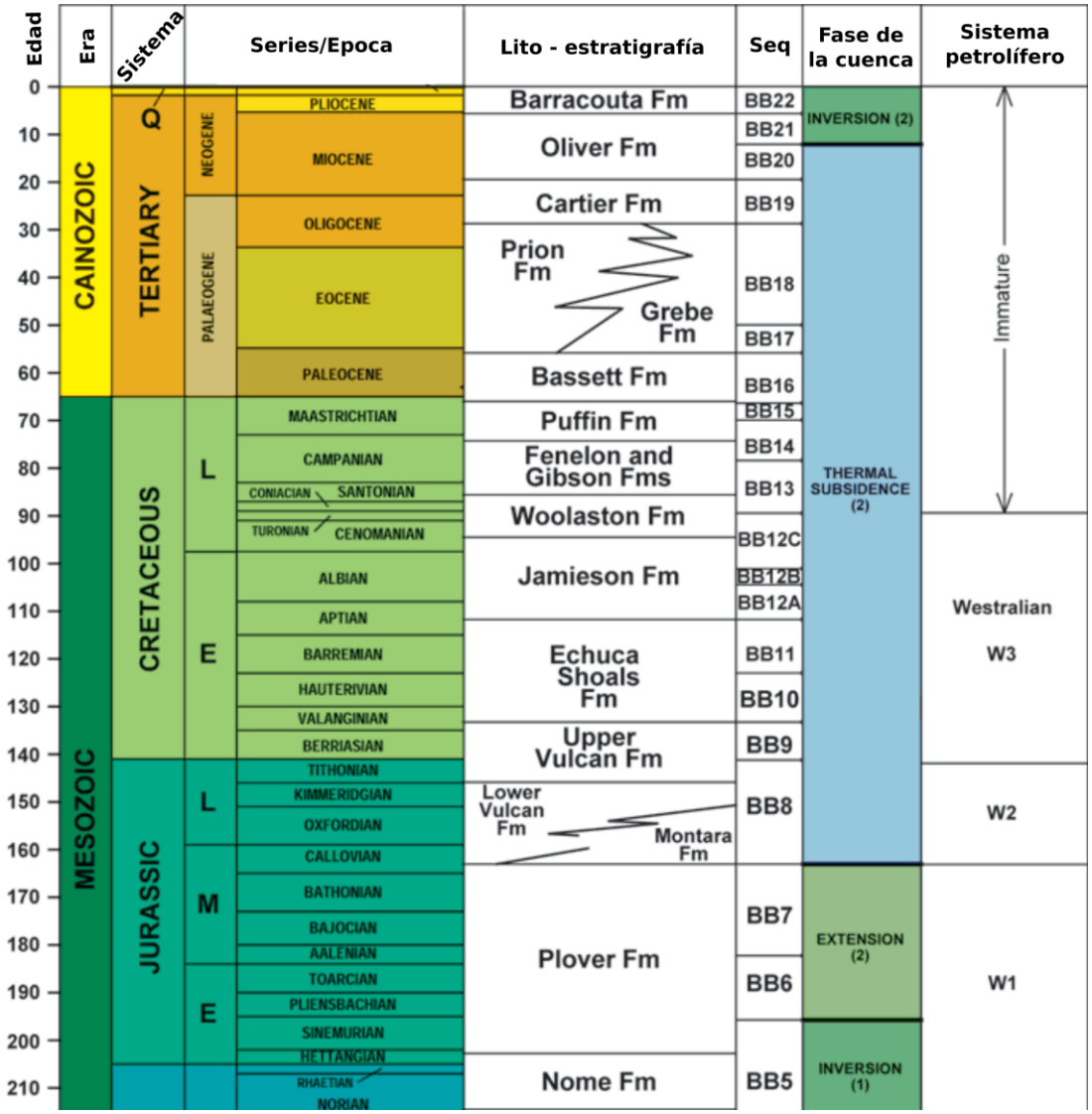

Figura 2.2. Secuencia estratigráfica generalizada de la cuenca Browse (Modificado de *Geoscience Australia*, 2019)

Tabla 2.1. Observaciones generales de las formaciones de interés. Compilado de ConocoPhillips (2011), ConocoPhillips (2012) y Liu (2018.)

Formación	Estudiada por	Litologías	Discordancias y horizontes interpretados
Oliver	ConocoPhillips (2011) y Liu (2018)	Carbonatos con arenas y lutitas calcáreas hacia la base.	Horizontes sísmicos Temio y Tmmio (Tope Mioceno Temprano y Tardío) en la base y entre la Formación respectivamente
Prion		Carbonatos, arenas y limolitas con intercalaciones de carbón hacia la base	Horizonte sísmico Tolig (Tope del Oligoceno) por encima del tope.
Grebe		Carbonatos y lutitas calcareas con intercalaciones de chert y margas en la base. Incluye los Miembros Heywood y Baudin (ambos carbonatos)	-
Johnson		Secuencia de carbonatos con pequeñas cantidades de margas, limolitas y lutitas calcareas de tope a base	Horizonte sísmico Tpal (Tope del Paleógeno) en el tope
Prudhoe		Secuencias de lutitas calcareas, no calcareas y arenas	Horizonte sísmico Tbase (base Paleógeno) en el tope
Fenalon			Horizonte sísmico Kecamp (Campaniense Temprano) identificado entre formaciones.
Gibson			
Woolaston	ConocoPhillips (2011) y ConocoPhillips (2012)	Gradación de limolita a carbonato de tope a base	-
Jamieson		Gradación de lutitas a limolitas de tope a base.	-
Montara		Arenas deltáicas a marinas someras de grano fino	Delimita en la base por la discordancia del Calloviense.
Plover		Secuencia de arenas fluviodeltáicas a arenas marinas de aguas someras, lutitas, limolitas, menores cantidades de carbonatos y sedimentos de origen volcánico en el tope	Delimitada por la discordancia del Triásico y la discordancia del Calloviense.

CAPÍTULO III

MARCO TEÓRICO

3.1. Método de sísmica de reflexión

El método sísmico consiste en la deducción tanto de la naturaleza como de las propiedades elásticas de las rocas en el subsuelo por medio del análisis e interpretación de la propagación de un tren de ondas sísmicas (Telford *et al*., 1990). Cavada (2000) asegura que el método involucra un elemento generador de ondas sísmicas denominado fuente, un medio de propagación (rocas, aire, agua) y un elemento detector-registrador de las ondas denominado receptor. La sísmica de reflexión (convencional) se fundamenta en el cartografiado de la estructura interna de la Tierra (Figura 3.1) a través de la medición del tiempo requerido por un tren de ondas sísmicas para regresar a la superficie luego de haber sido reflejado en la interfaz existente entre dos formaciones con diferentes propiedades elásticas. Hoy en día existen diversas técnicas dentro de los distintos niveles de procesamiento, que permiten interpretaciones más complejas tanto a nivel geológico como en la producción de hidrocarburos (Telford *et al*., 1990; Avseth, Mukerji y Mavko, 2005; Chopra y Castagna, 2014).

Los datos pre-apilados proporcionan dos factores claves que a menudo son irreconocibles cuando se investiga a la misma área geográfica utilizando datos sísmicos post-apilados (Halliburton, 2007). El primero cumple como control de calidad del dato, mientras que el segundo involucra la predicción directa de la litología y los efectos de los fluidos en las amplitudes de las trazas. En este sentido, la interpretación de la sísmica pre-apilada permite detectar gas en capas delgadas,

yacimientos fracturados y en general, la disminución del margen de incertidumbre en la toma de decisiones.

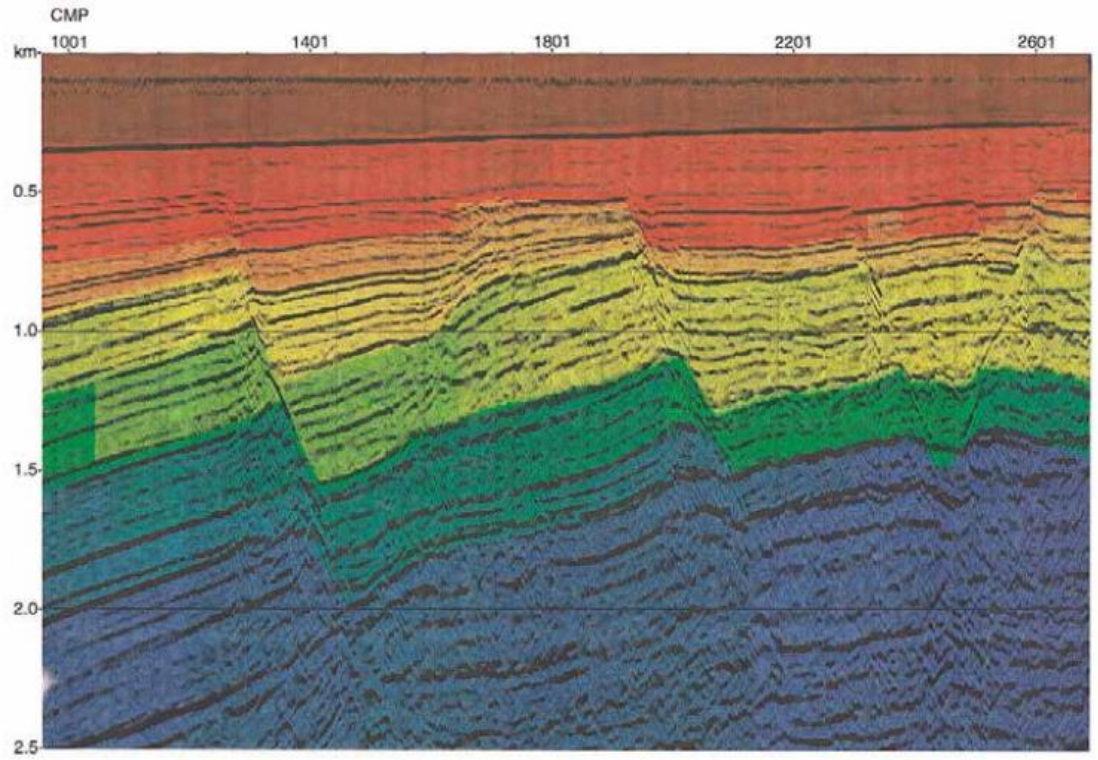

Figura 3.1. Objetivo final de la sísmica de reflexión convencional (Modificado de Yilmaz, 2001).

3.1.1 Apilamiento

Schlumberger (2019) define el proceso de apilamiento o *stacking*, como la suma de las trazas con un mismo punto de reflexión para aumentar la relación señal–ruido y la calidad de la señal al resaltar la respuesta común y desechar lo anómalo. La Figura 3.2 muestra un esquema generalizado de apilamiento sísmico por Punto Medio en Común (CMP, por sus siglas en inglés *Common Mid Point*), el cual consiste en agrupar todas las respuestas sísmicas cuya distancia fuente-receptor coincide en una misma coordenada o punto geométrico (Yilmaz, 2001). En la práctica, se apilan todas aquellas trazas que muestreen un mismo punto a profundidad (CDP, por sus siglas

Common Depth Point) para optener una traza única, durante la fase de procesamiento sísmico, cuya distancia fuente – receptor y ángulo de incidencia son 0, lo cual fundamenta el método sísmico de reflexión convencional.

3.2. Ondas Sísmicas

La exploración de hidrocarburos por medio del método sísmico de reflexión utiliza principalmente las ondas corpóreas que, de acuerdo a diversos autores (entre ellos Cavada, 2000), son ondas que se propagan a través de los materiales del subsuelo y conforme a su naturaleza, pueden dividirse en:

- Ondas Primarias, también conocidas como ondas P u ondas compresionales son las ondas más veloces cuya propagación puede darse a través de cualquier material donde el movimiento de sus partículas tiene el mismo sentido de propagación de la onda.
- Ondas secundarias, también conocidas como ondas S u ondas de transversales son ondas menos veloces que las ondas P. Su propagación se da únicamente a través de sólidos donde el movimiento de las partículas es perpendicular a la dirección de propagación.

La velocidad de propagación de las ondas sísmicas es función de la elasticidad y densidad del medio de propagación por lo general no uniforme. Sin embargo, la velocidad también depende de otras propiedades físicas del medio como litología, mineralogía, porosidad, permeabilidad y fluidos. Rocas con mayor resistencia a la deformación tienden a exhibir mayores velocidades. Las Ondas P involucran deformaciones a nivel volumétrico y de forma, mientras que las S, solo de forma (Chopra y Castagna, 2014).

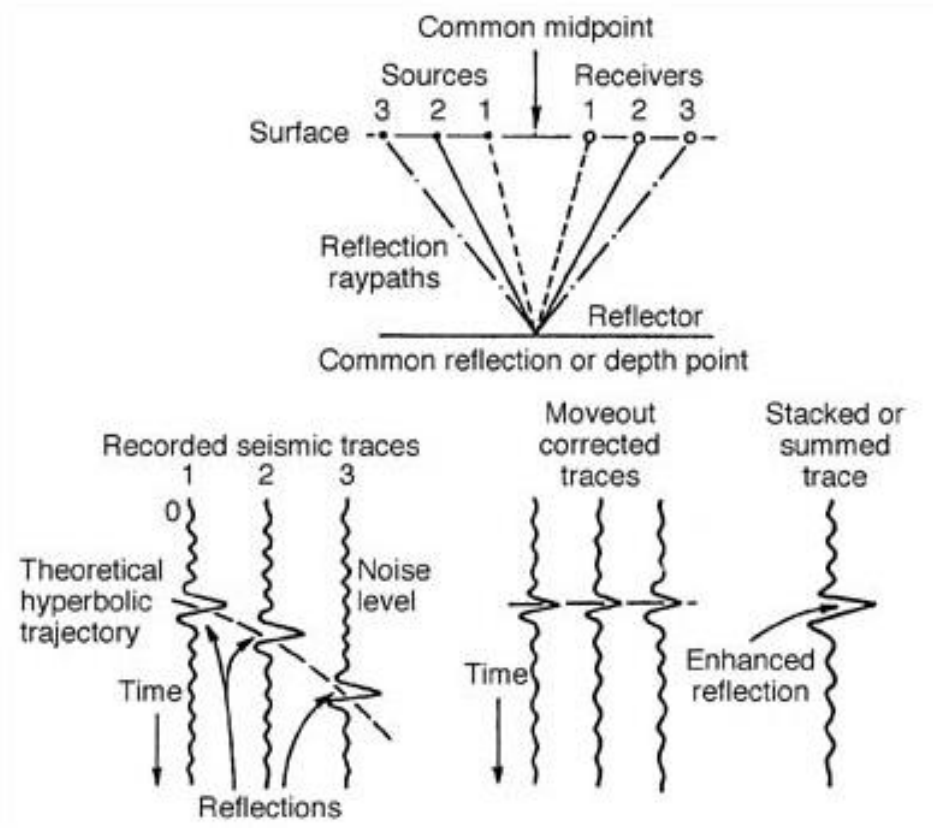

Figura 3.2. Apilamiento por punto medio en común o CMP. (Schlumberger, 2019).

3.3. Partición de energía y dependencia angular

Cuando un tren de ondas sísmicas se genera en un punto dado, este se propaga lejos del punto inicial en forma de frentes de ondas expansivas. Un frente de onda se refiere a todo conjunto de partículas de un medio que experimentan movimientos similares en un instante de tiempo dado. En la práctica, se utilizan rayos sísmicos para idealizar la propagación de las ondas sísmicas (Figura 3.3), es decir, se utiliza un vector perpendicular al tren de onda cuya dirección y sentido coincide con este último (Chopra y Castagna, 2014). Backus y Castagna (1993) afirman que para comprender la propagación de las ondas sísmicas en estos medios anisotrópicos, no uniformes, se emplea el modelo elástico que considera que parte de la energía de la onda P incidente será reflejada y transmitida en forma de onda P. El resto de la energía experimenta un proceso de conversión de onda P a S, si el ángulo de incidencia es diferente de 0º (Figura 3.3). Las Figuras 3.4 y 3.5 ilustran como pueden variar los coeficientes de reflexión y transmisión en función de la elasticidad de la roca in situ y en presencia de un ángulo de incidencia crítico, ángulo a partir del cual, cesa la transmisión de energía puesto que la onda transmitida se comporta como una onda refractada (Chopra y Castagna, 2014).

Diversos autores (Backus y Castagna, 1993; Chopra y Castagna, 2014 y Russell, 2014) afirman que el proceso de partición de energía de un tren de ondas sísmicas está controlado por la Ley de Snell (Ecuación 3.1) y definen a la proporción de energía que se refleja como coeficiente de reflexión y, en correspondencia, al coeficiente de transmisión como al porcentaje de la energía que se transmite de un medio a otro. Zoeppritz (1919) dedujo a partir de la Ley de la conservación de la energía y a la dependencia de estos coeficientes al ángulo de incidencia (Ley de Snell) un modelo matricial avanzado de cuatro ecuaciones con cuatro incógnitas (coeficientes) que puede ser resuelto utilizando inversión de matrices (Ecuación 3.2). Para ángulos de incidencias normales las ecuaciones de Zoeppritz se simplifican a dos ecuaciones que fundamentan el cálculo de amplitudes en la sísmica post-apilada (Ecuaciones 3.3 y 3.4).

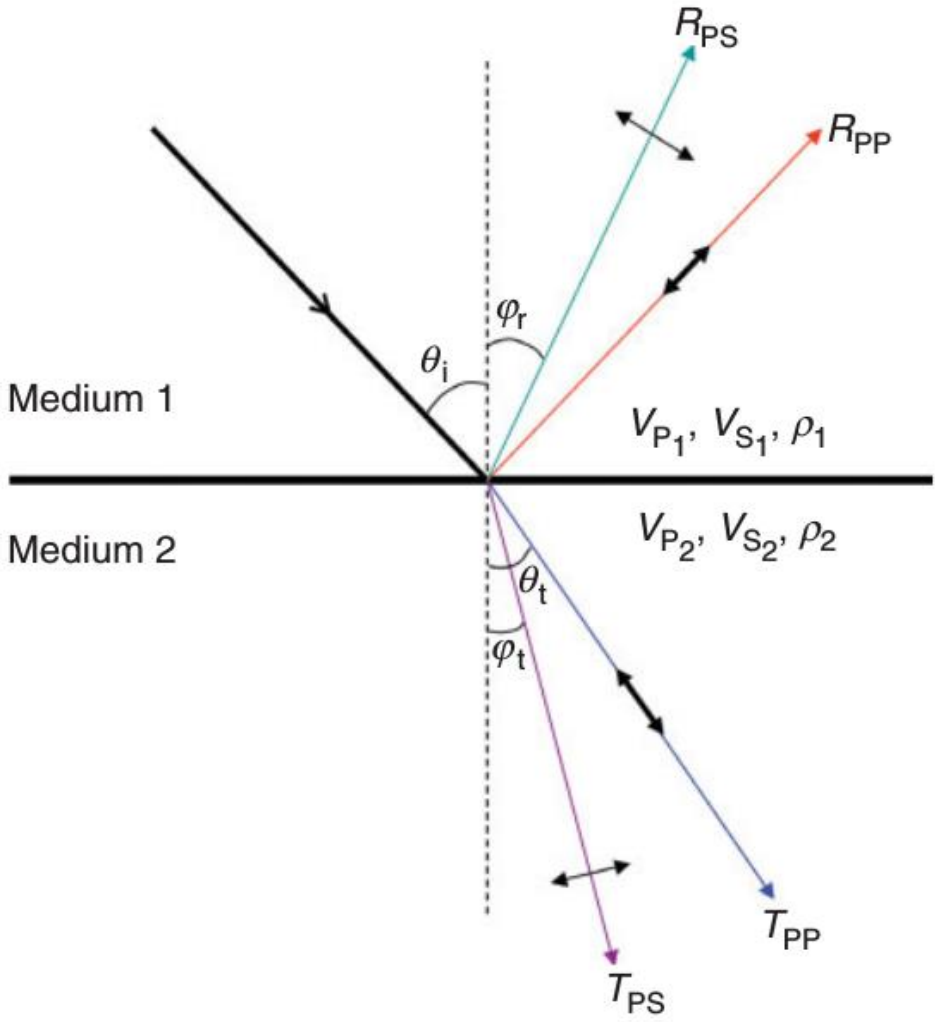

Figura 3.3. Reflexión, transmisión y conversión de energía un tren de ondas P incidente sobre un reflector sísmico. Donde R_{PS} es el coeficiente de reflexión de la onda S convertida, R_{PP}el coeficiente de reflexión de la onda P; θ_i el ángulo de incidencia del pulso original y de R_{PP}, T_{PS} el coeficiente de transmisión de energía de la onda S convertida, T_{PP}el coeficiente de transmisión de la onda P; φ_r, φ_ty θ_t los ángulos de reflexión y transmisión para la onda S convertida y el ángulo de transmisión de la onda P respectivamente. Los vectores negros indican el movimiento de las partículas de cada onda (Chopra y Castagna, 2014).

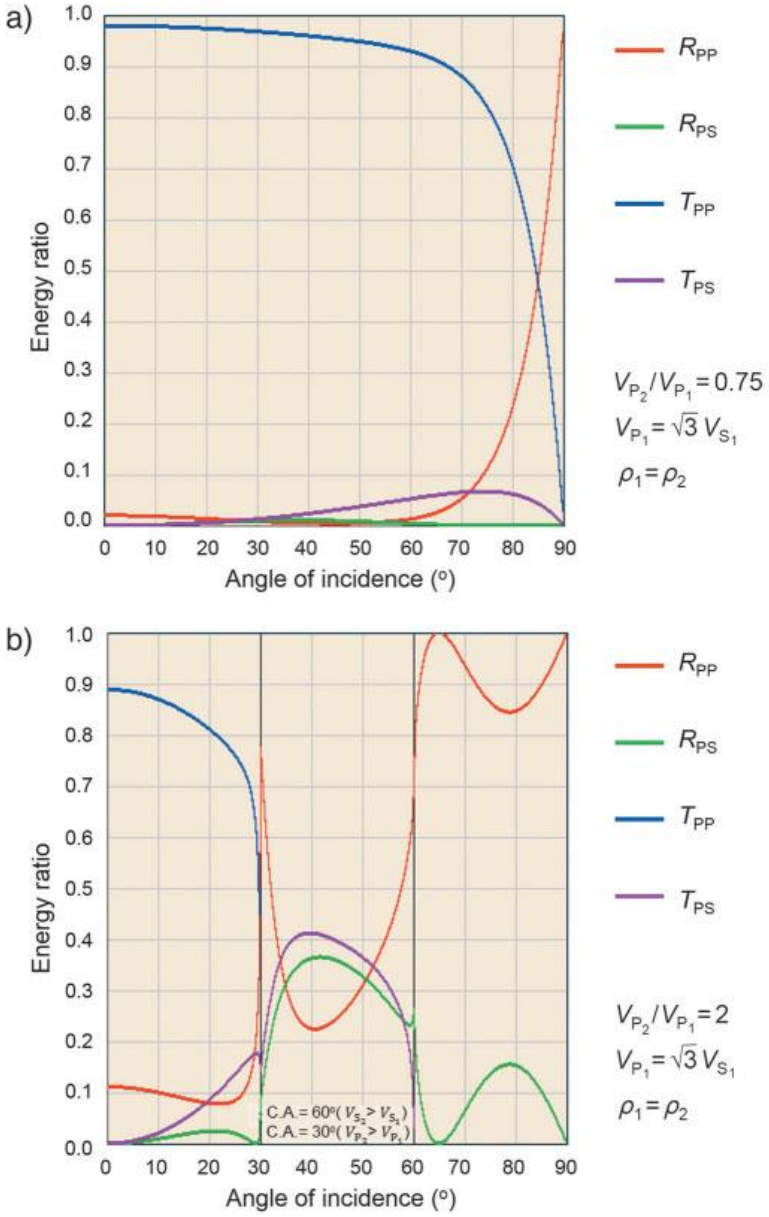

Figura 3.4. Comportamiento de los coeficientes de reflexión y transmisión en función del ángulo de incidencia, donde las velocidades de ambos medios son diferentes mientras que densidades son similares. Se observa que no existe conversión de ondas para ángulos menores a 10°. (a) Exhibe un ángulo crítico para la onda P en 90°. (b) Exhibe un ángulo crítico para la onda P en 30° y para la onda S en 60° (Chopra y Castagna, 2014).

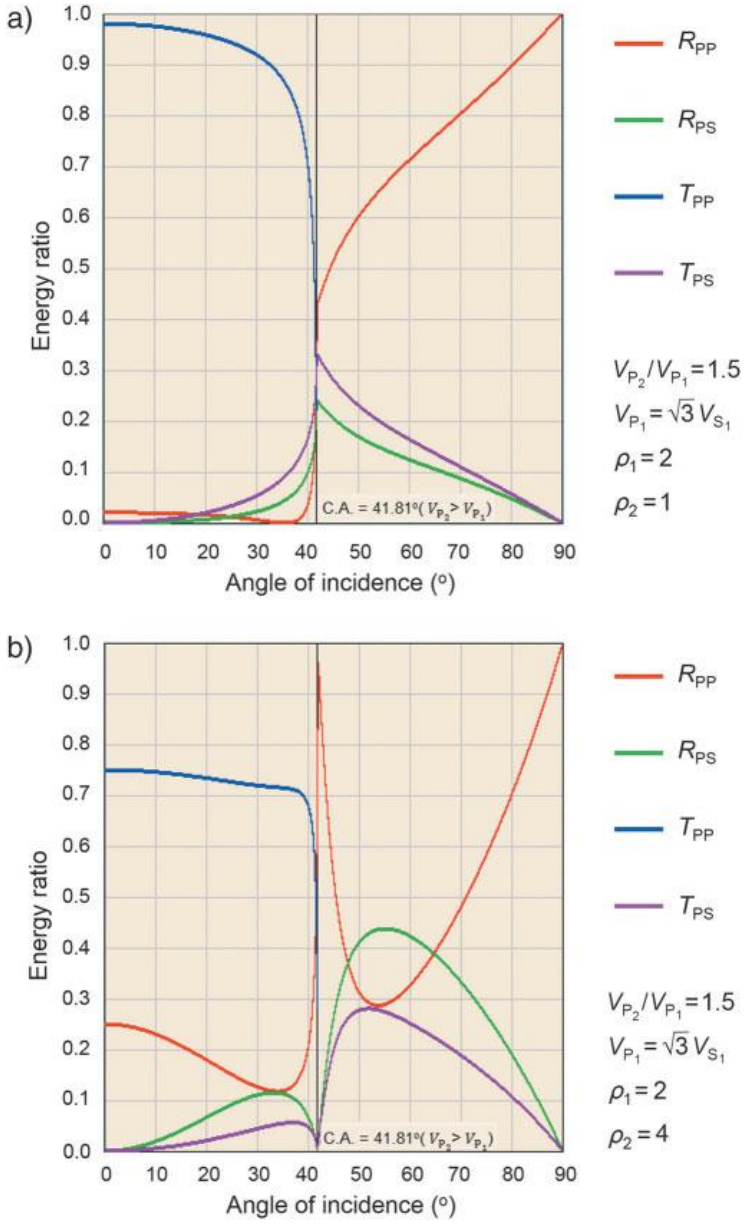

Figura 3.5. Comportamiento de los coeficientes de reflexión y transmisión en función del ángulo de incidencia, donde las velocidades y las densidades de ambos medios son diferentes. Se observa que no existe conversión de ondas para ángulos menores a 10 y 8 º en (a) y (b) respectivamente. Ambas figuras exhiben un ángulo crítico en 40 º (Chopra y Castagna, 2014).

$$\frac{\sin\theta_i}{V_{P_1}} = \frac{\sin\theta_t}{V_{P_2}} = \frac{\sin\varphi_r}{V_{S_1}} = \frac{\sin\varphi_t}{V_{S_2}} \qquad \text{(Ecuación 3.1)}$$

Donde:

θ_i y θ_t: Los ángulos con los que se refleja y transmite el pulso inicial.

φ_ry φ_t: Los ángulos con el que se refleja y transmite la onda S convertida.

V_{P_1}y V_{P_2}: Las velocidades de la onda P en el medio 1 y 2.

V_{S_1} y V_{S_2}: Las velocidades de la onda S en el medio 1 y 2.

$$\begin{pmatrix} R_{PP}(\theta_1) \\ R_{PS}(\theta_1) \\ T_{PP}(\theta_1) \\ T_{PS}(\theta_1) \end{pmatrix} =$$

$$\begin{pmatrix} -\sin\theta_1 & -\cos\varphi_1 & \sin\theta_2 & \cos\varphi_2 \\ \cos\theta_1 & -\sin\varphi_1 & \cos\theta_2 & -\sin\varphi_2 \\ \sin 2\theta_1 & \frac{V_{P_1}}{V_{S_1}}\cos 2\varphi_1 & \frac{\rho_2 {V_{S_2}}^2 V_{P_1}}{\rho_1 {V_{S_1}}^2 V_{P_2}}\sin 2\theta_2 & \frac{\rho_2 V_{S_2} V_{P_1}}{\rho_1 {V_{S_1}}^2}\cos 2\varphi_2 \\ -\cos 2\varphi_1 & \frac{V_{S_1}}{V_{P_1}}\sin 2\varphi_1 & \frac{\rho_2 V_{P_2}}{\rho_1 V_{P_1}}\cos 2\varphi_2 & \frac{\rho_2 V_{P_2}}{\rho_1 V_{P_1}}\sin 2\varphi_2 \end{pmatrix}^{-1} \begin{pmatrix} \sin\theta_1 \\ \cos\theta_1 \\ \sin 2\theta_1 \\ \cos 2\varphi_1 \end{pmatrix}$$

(Ecuación 3.2)

Donde:

$R_{PP}(\theta_1)$ y $R_{PS}(\theta_1)$: Los coeficientes de reflexión para las ondas P y S.

$T_{PP}(\theta_1)$ y $T_{PS}(\theta_1)$: Los coeficientes de transmisión para las ondas P y S.

θ_i y θ_t: Los ángulos con los que se refleja y transmite el pulso inicial.

φ_ry φ_t: Los ángulos con el que se refleja y transmite la onda S convertida.

ρ_1y ρ_2: Las densidades de los medios 1 y 2.

$V_{P_1}y\ V_{P_2}$: Las velocidades de la onda P en el medio 1 y 2.

V_{S_1} y V_{S_2}: Las velocidades de la onda S en el medio 1 y 2.

$$R_{PP}(0°) = \frac{\rho_2 V_{P_2} - \rho_1 V_{P_1}}{\rho_2 V_{P_2} + \rho_1 V_{P_1}} = \frac{Z_2 - Z_1}{Z_2 + Z_1} \quad \text{(Ecuación 3.3)}$$

$$T_{PP}(0°) = \frac{2\rho_1 V_{P_1}}{\rho_2 V_{P_2} + \rho_1 V_{P_1}} = \frac{2Z_1}{Z_2 + Z_1} \quad \text{(Ecuación 3.4)}$$

Donde:

$R_{PP}(0)$ y $T_{PP}(0)$: Los coeficientes de reflexión y transmisión para la onda P en 0°.

ρ_1 y ρ_2: Las densidades de los medios 1 y 2.

V_{P_1} y V_{P_2}: Las velocidades de la onda P en el medio 1 y 2.

V_{S_1} y V_{S_2}: Las velocidades de la onda S en el medio 1 y 2.

Z_n: La impedancia acústica del medio.

La impedancia acústica en cada interfase involucrada en las Ecuaciones 3.3 y 3.4 implican la resistencia a la transmisión de energía (Chopra y Castagna, 2014). El sistema de ecuaciones asociado a la Ecuación 3.2 permite el cálculo de los valores de amplitud para cualquier ángulo, sin embargo, estas son complejas y de soluciones laboriosas que no ofrecen una forma intuitiva de cuantificar el impacto de las propiedades de las rocas sobre los cambios de amplitudes. Por lo anterior, se han diseñado diversas aproximaciones lineales para simplificar el cálculo de estas amplitudes, específicamente de aquellas asociadas a las reflexiones de la onda P (Bacon y Simm, 2014; Chopra y Castagna, 2014; Russell, 2014).

3.3.1. Características de las ecuaciones de Zoeppritz

Chopra y Castagna (2014) especifican que para el entendimiento total de la Ecuación 3.2 es necesario comprender el alcance de la misma:

- Se describen los coeficientes de reflexión en la dirección de propagación del frente de ondas por lo que, para geófonos verticales, la componente vertical corresponde a la reflexión de interés.

- Se asume la solución de ondas planas en las que los coeficientes de reflexión y transmisión son independientes de la frecuencia, por tanto, los coeficientes de reflexión se vuelven complejos en ángulos mayores al ángulo crítico y se introduce un cambio de fase. Cuando realmente, las ondas sísmicas son esféricas y dependientes de la frecuencia, por lo que el máximo coeficiente de reflexión ocurre más allá del ángulo crítico.
- Se describe la reflexión para una interface que separa dos espacios, sin embargo, no se incluyen los efectos de interferencia que pueden ser causados por las propias capas.
- Las amplitudes son equivalentes a los coeficientes de reflexión en ausencia de los efectos que incluyen perdidas por transmisión, atenuación, divergencia, conversión, etc. En la práctica, las amplitudes sísmicas son raramente directamente proporcionales a los coeficientes de reflexión. Descuidar este efecto puede ocasionar interpretaciones erróneas.
- Se asume la asimetría en los cálculos de los coeficientes, cuando en la realidad, el comportamiento de estos es asimétrico ya que la amplitud para una onda descendiente e incidente en una interface desde un medio 1 no es igual al de una onda ascendente e incidente en la misma interface desde un medio 2. Esta discrepancia generalmente se vuelve cada vez más significativa a medida que en el ángulo de incidencia se incrementa.

3.4. AVO

El análisis de AVO consiste en el estudio de la variación de amplitud de las ondas corporeas a medida que estas se propagan por un medio con un ángulo distinto al de incidencia normal (Chopra y Castagna, 2014). Ostrander (1982) demostró que bajo ciertas condiciones geológicas las arenas hidrocarburíferas muestran un claro aumento en amplitud conforme aumenta la distancia fuente-receptor o ángulo de las reflexiones. Por lo anterior, diversos autores (entre ellos Avseth, Mukerji y Mavko, 2005; Chopra y Castagna, 2014; Russell, 2014) describen al AVO como una técnica

efectiva en la prospección de hidrocarburos ya que se fundamenta en la relación entre los coeficientes de reflexión, la velocidad de onda compresional (Vp), la velocidad de onda de corte (Vs) y los módulos elásticos. Estos últimos, muy particulares en las rocas asociadas al sistema petrolífero.

Cantidad de autores especializados en el AVO como técnica prospectiva (Almutlaq y Margrave (2010); Russell (2014); Bacon y Simm, 2014) aseguran que el auge de la técnica tuvo lugar posterior al entendimiento de como afectaban las propiedades elásticas a las amplitudes sísmicas conforme se desarrollaban diferentes acercamientos y simplificaciones a la Ecuación 3.2. Aki y Richards (1980) diseñaron una aproximación lineal de la ecuación Zoeppritz (1919) en función del seno cuadrado del ángulo de incidencia, las velocidades y las densidades de los medios de propagación (Ecuación 3.5), más tarde reformulada por Shuey (1985), cuya investigación sustenta las bases de este trabajo.

$$R(\theta) = A + B\,sin^2\theta + C\,sin^2\theta tan^2\theta \qquad \text{(Ecuación 3.5)}$$

Con:

$$A = \frac{1}{2}\left(\frac{\Delta V_P}{V_P} + \frac{\Delta\rho}{\rho}\right), \qquad B = \frac{\Delta V_P}{2V_P} - 4\left(\frac{V_S}{V_P}\right)^2\left(\frac{\Delta V_S}{V_S}\right) - 2\left(\frac{V_S}{V_P}\right)^2\left(\frac{\Delta\rho}{\rho}\right), \qquad C = \frac{1}{2}\frac{\Delta V_P}{V_P}$$

$$V_P = \frac{V_{P_1} + V_{P_2}}{2}, \qquad V_S = \frac{V_{S_1} + V_{S_2}}{2}, \qquad V_P = \frac{\rho_1 + \rho_2}{2}$$

$$\Delta V_P = V_{P_2} - V_{P_1}, \qquad \Delta V_S = V_{S_2} - V_{S_1}, \qquad \Delta\rho = \rho_2 - \rho_1$$

Donde:

A: El coeficiente de reflexión en 0º relacionado con el contraste de impedancias.

B: El efecto de la velocidad de corte para ángulos diferentes a 0°.

C: La curvatura para la respuesta de amplitudes cercanas al ángulo crítico.

ρ_1 y ρ_2: Las densidades de los medios 1 y 2.

$\Delta\rho$: La diferencia entre las densidades de los medios 1 y 2.

V_{P_1} y V_{P_2}: Las velocidades de la onda P en el medio 1 y 2.

ΔV_P: La diferencia entre las velocidades de onda P en los medios 1 y 2.

V_{S_1} y V_{S_2}: Las velocidades de la onda S para el medio 1 y 2.

ΔV_S: La diferencia entre las velocidades de onda S en los medios 1 y 2.

La aproximación mostrada en la Ecuación 3.5 introduce los atributos sísmicos de intercepto (A) y gradiente (B) que pueden observarse en la Figura 3.6 (Bacon y Simm, 2014). La linearización aplicada al evaluar las amplitudes respecto al seno cuadrado del ángulo de incidencia permite relacionar al gradiente como la pendiente del comportamiento lineal de las amplitudes y al intercepto como la intersección entre el eje de amplitud y esta tendencia lineal. El término C de la Ecuación 3.5 se asocia con la curvatura cuando la evaluación en el seno y la tangente cuadrada del ángulo de incidencia ya no son despreciables.

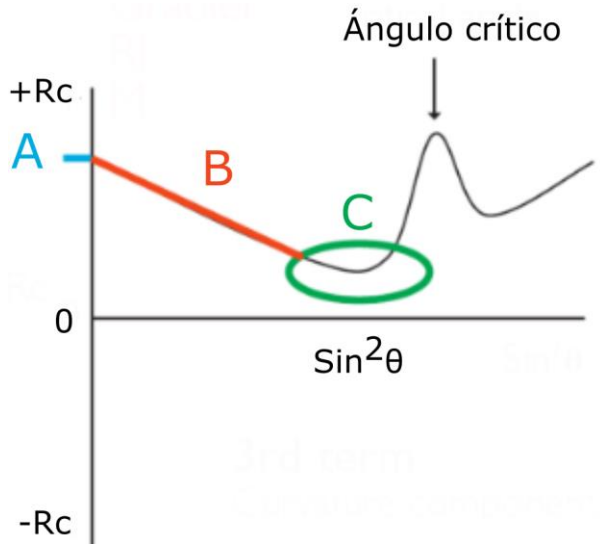

Figura 3.6. Representación gráfica de las tres componentes de la aproximación de las ecuaciones de Zoeppritz (1919) diseñada por Aki y Richards (1980). Se resaltan los atributos de *A*, *B* y *C* (Modificado de Bacon y Simms, 2014).

Shuey (1985) formuló una regresión lineal a partir de la aproximación de Aki-Richards (1980) considerando únicamente sus dos primeros términos (Ecuación 3.6). Una de las asunciones de Shuey (1985) es que para ángulos mayores a 30° el resultado de la aproximación de dos términos diverge totalmente de la solución de la ecuación de Zoeppritz (Bacon y Simms, 2014). La Figura 3.7 muestra el comportamiento de las aproximaciones de dos y tres términos en comparación con el resultado obtenido a partir de las ecuaciones de Zoeppritz.

$$R(\theta) = A + B\, sin^2\theta \qquad \text{(Ecuación 3.6)}$$

Con:

$$A = \frac{1}{2}\left(\frac{\Delta V_P}{V_P} + \frac{\Delta\rho}{\rho}\right), \qquad B = \frac{\Delta V_P}{2V_P} - 4\left(\frac{V_S}{V_P}\right)^2\left(\frac{\Delta V_S}{V_S}\right) - 2\left(\frac{V_S}{V_P}\right)^2\left(\frac{\Delta\rho}{\rho}\right)$$

$$V_P = \frac{V_{P_1} + V_{P_2}}{2}, \qquad V_S = \frac{V_{S_1} + V_{S_2}}{2}, \qquad V_P = \frac{\rho_1 + \rho_2}{2}$$

$$\Delta V_P = V_{P_2} - V_{P_1}, \qquad \Delta V_S = V_{S_2} - V_{S_1}, \qquad \Delta\rho = \rho_2 - \rho_1$$

Donde:

A y B: Los atributos sísmicos de Intercepto y Gradiente.

ρ_1 y ρ_2: Las densidades de los medios 1 y 2.

$\Delta\rho$: La diferencia entre las densidades de los medios 1 y 2.

V_{P_1} y V_{P_2}: Las velocidades de la onda P en el medio 1 y 2.

ΔV_P: La diferencia entre las velocidades de onda P en los medios 1 y 2.

V_{S_1} y V_{S_2}: Las velocidades de la onda S para el medio 1 y 2.

ΔV_S: La diferencia entre las velocidades de onda S en los medios 1 y 2.

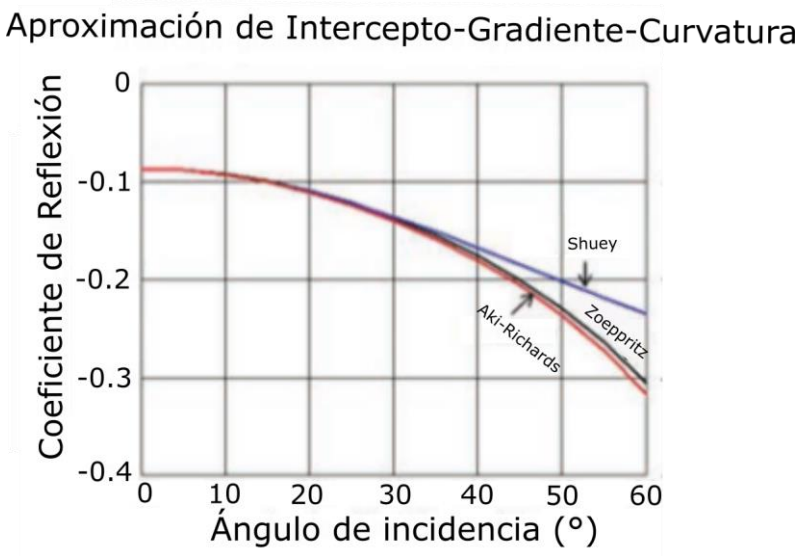

Figura 3.7. Comparación entre las aproximaciones de Aki-Richards (1980) y Shuey (1985) con el resultado de las ecuaciones de Zoeppritz al evaluar una arena gasífera. Nótese la divergencia alrededor de 40º en la aproximación con dos términos (Russell, 2014).

Por su simplicidad, la aproximación de Shuey (1985) es eficaz para el análisis de la respuesta AVO en relación a los efectos de la presencia de fluidos y cambios litológicos ante la ausencia de un volumen denso de datos y para un rango de ángulos menores a 30 °. Esta puede utilizarse para calcular los atributos de intercepto y gradiente directamente del dato sísmico, al aplicarle una regresión lineal a las amplitudes sísmicas en función del seno cuadrado del ángulo de incidencia (Shuey, 1985; Russell, 2010; Russell, 2014; Bacon y Simm, 2014; SEG, 2019). La Figura 3.8 ilustra el cálculo de los atributos AVO a partir de un volumen sísmico.

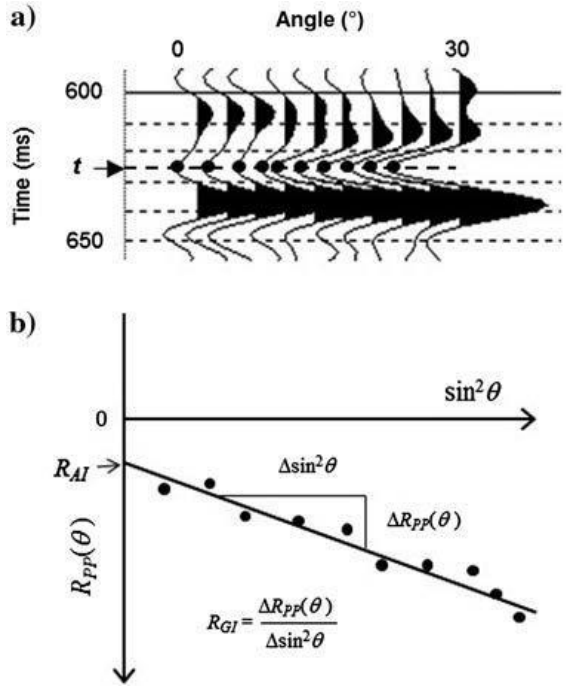

Figura 3.8. Cálculo de los atributos de AVO a partir del dato sísmico. (a) Muestra la selección de amplitudes en un *Gather*. (b) Ilustra el cálculo de los atributos AVO a partir de la regresión lineal de la selección en (a), al evaluar estos en el seno cuadrado del ángulo de incidencia (Russell, 2014).

3.4.2. Descripción de la respuesta AVO

Se entiende por respuesta AVO positiva a cualquier incremento en amplitud ya sea positivo o negativo y no al signo de los atributos, en contraste, una respuesta de AVO negativa implica la disminución de la amplitud respecto al ángulo de incidencia. En tal sentido, la respuesta AVO es una convención descrita en términos de clases (Tabla 3.1) por Rutherford y Williams (1989) al clasificar la respuesta de un contacto lutita–arena en tres tipos, más tarde modificada por Ross y Kinman (1995) y Castagna y Swan (1997). Las Clases de AVO se utilizan para describir el comportamiento particular de las amplitudes. Para evaluar una multitud de respuestas se emplean diagramas cruzados de AVO que permiten realizar discriminaciones litológicas y de fluidos (Bacon y Simm, 2014; Russell, 2014). La Figura 3.9 ilustra las Clases de AVO presentes en la Tabla 3.1.

Tabla 3.1. Clasificación de AVO. (Modificado de Bacon y Simm, 2014).

Clase	G	I	Amp. absolutas	Resp. AVO	Notas
I	-	+	R(30) < R(0)	-	Puede haber cambio de fase
IIp	-	+	R(30) > R(0)	+	Ross y Kinman (1995). Hay cambio de fase
II	-	-	R(30) > R(0)	+	Bajas amplitudes en R(0)
III	-	-	R(30) > R(0)	+	Altas amplitudes en R(0)
IV	+	-	R(30) < R(0)	-	Castagna y Swan (1997). Altas amplitudes en R(0)

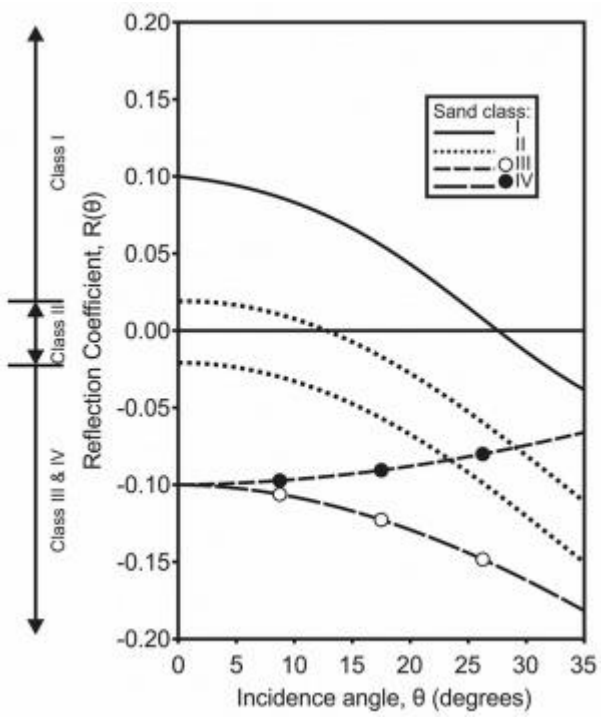

Figura 3.9. Clasificación de AVO por Rutherford y Williams (1989), Ross y Kinman (1995) y Castagna y Swan (1997). Se muestra la leyenda en la parte superior de la figura (SEG *wiki*, 2019).

Los diagramas cruzados de los atributos AVO, específicamente intercepto vs gradiente, proveen un método para diferenciar las arenas gasíferas de arenas altamente porosas con contenido de agua (Bacon y Simm, 2014). En polaridad normal, las arenas con contenido de agua exhiben una tendencia inclinada de izquierda a derecha, dentro de lo que corresponde a los cuadrantes II y IV del diagrama cruzado, mientras que las arenas gasíferas divergen de esta tendencia hacia abajo y a la izquierda (tendencia denominada "*Down to the Leftness*"). Estos puntos anómalos constituyen una respuesta típica en relación al tope de una arena gasífera y es el principal efecto explotado en el análisis de AVO. La Figura 3.10 muestra la diferenciación entre las tendencias de las arenas mencionadas anteriormente en polaridad normal.

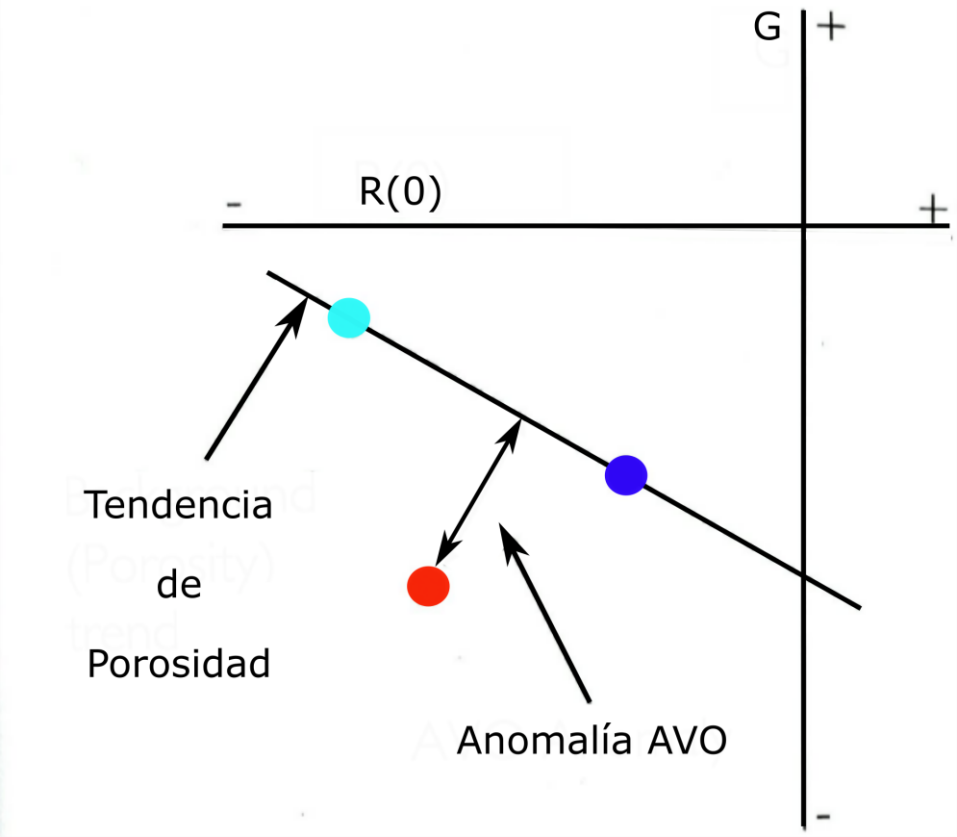

Figura 3.10. Anomalía de AVO definida en un gráfico cruzado. El punto rojo ilustra la tendencia "*Down to the leftness*" (Bacon y Simm, 2014).

3.5. Efecto de los fluidos intersticiales

En términos generales, la presencia de fluidos intersticiales disminuye la densidad total de la roca, la velocidad de la onda P, los parámetros elásticos asociados a esta y, en consecuencia, las impedancias acústicas. En relación a lo anterior, la presencia de gas, al ser el fluido con menor densidad, causa el mayor efecto de reducción en las propiedades mencionadas y cuya representación gráfica corresponde a la tendencia *down to the leftness*. Otra de las características explotadas en el análisis de AVO es la perdida de la amplitud de las arenas altamente porosas respecto a los ángulos de incidencia, mientras que de forma contrastante, las arenas gasíferas adquieren amplitud (Bacon y Simm, 2014).

CAPÍTULO IV

MARCO METODOLÓGICO

Para el cumplimiento de los objetivos planteados se diseñó el flujo de trabajo que se muestra en al Figura 4.1. La primera etapa, consistió en la búsqueda del volumen de datos sísmicos pre-apilados (3D) libremente disponibles y las investigaciones más relevantes asociados a este. La segunda, se enfocó en la creación del ambiente de programación que permitiese la lectura e inspección de datos en formato SEG-Y. La tercera, en la codificación de algoritmos para caracterizar amplitudes sísmicas. La cuarta, en la comparación de los resultados con trabajos previos como método de validación y la última, en garantizar el libre acceso y la reproducibilidad de la herramienta.

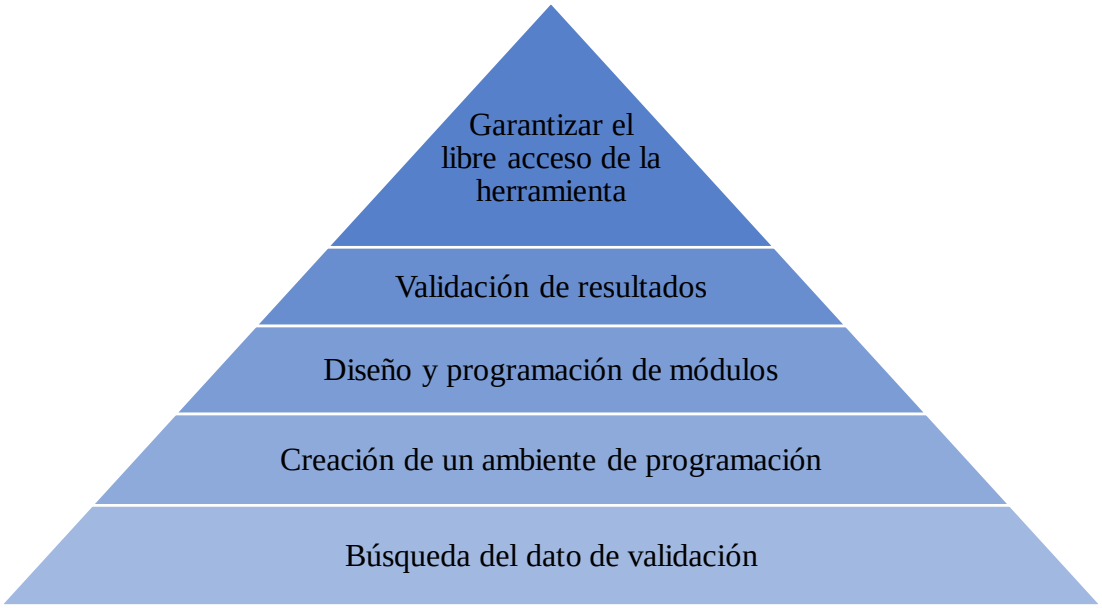

Figura 4.1 Flujo de trabajo para el cumplimiento de los objetivos planteados.

4.1. Revisión bibliográfica y selección del dato de validación

Los datos para validar la herramienta pertenecen al Proyecto "*NW Shelf Australia - Poseidon 3D*" de la petrolera ConocoPhillips y a Geoscience Australia. Este se encuentra disponible en la sección de datos libres del portal digital TerraNubis (2019) de la compañía dGB bajo la licencia gratuita *Creative Commons By Attribution* (CC-BY) e incluye:

- Un levantamiento sísmico 3D en formato CBVS (*OpendTect*) de 84.2 Gb que abarca un área de 2828 km^2 en una provincia probada de hidrocarburos (Cuenca Browse, región noroccidental de Australia).
- Reportes de procesamiento e interpretación.
- Tres archivos apilados a diferentes ángulos de 101 Gb cada uno: 6-18º (*Near-Angle Stack*), 18-30º (*Mid-Angle Stack*) y 30-42º (*Far-Angle Stack*).
- Registros de pozos y tiros de verificación para los pozos: Kronos 1, Pharos 1, Poseidon 1, Poseidon 2, Poseidon North 1, Proteus 1, Proteus 1ST1, Proteus 1ST2 y Torosa 1.

Para agilizar el desarrollo y validación de la herramienta computacional se empleó un subvolumen de 11x11 trazas (121 trazas en total) con amplitudes de 8 bits alrededor del pozo Kronos 1. La Tabla 4.1 resume las dimensiones del Proyecto Poseidon 3D, el subvolumen generado alrededor del pozo y su respectivo tamaño en memoria. En cuanto a trabajos previos en el área, se hizo especial énfasis en la interpretación de la compañía petrolera sobre el pozo Kronos 1 (ConocoPhillips, 2011), el reporte de interpretación sísmica (ConocoPhillips, 2012) y finalmente Liu (2018).

Tabla 4.1. Coordenadas del subvolumen de prueba alrededor del pozo Kronos 1.

Dato	Inline (min)	Inline (max)	Crossline (min)	Crossline (max)	Uso en memoria
Kronos 1	2405	2415	2664	2674	48 Mb
Poseidon	983	4419	504	5556	300 Gb

Los archivos parcialmente apilados a diferentes ángulos, también denominados *angle stacks*, representan un compromiso en la manipulación de los datos sísmicos. La herramienta desarrollada cuenta con funciones que permiten agrupar este tipo de archivos para generar un archivo pseudo-preapilado y, en consecuencia, falicitar la obtención de resultados.

4.2. Creación del ambiente de programación

Se decidió utilizar el sistema operativo *Ubuntu* (v16.04) debido su código abierto, pocas restricciones y disponibilidad de herramientas de programación; *Python* (v3.7) como lenguaje dada su popularidad y lista de bibliotecas para el desarrollo de la Ingeniería y *Jupyter Notebook* como intérprete por su capacidad de permitir la documentación de códigos. Se utilizó la distribución *Anaconda* (vAnaconda3-2019.07-32 bits), compatible con las especificaciones del equipo (Apéndice A), para instalar *Python*, *Jupyter Notebook*, las dependencias básicas y las bibliotecas listadas en el Apéndice B.

4.3. Programación de módulos y funciones

La herramienta fue organizada en cuatro módulos. El primero, consta de diversas funciones de utilidad que permiten la validación, depuración y almacenamiento de la información contenida en los archivos sísmicos. El segundo, contiene funciones de visualización de la geometría basado en sus coordenadas. El tercero, dispone de isntrucciones para el despliegue de las trazas sísmicas. Y el último, abarca funciones de cálculo, almacenamiento y visualización de los atributos AVO. Los tres últimos módulos incluyen elementos que aumentan la interactividad entre el usuario y los resultados, con la finalidad de enriquecer la experiencia, evitar soluciones estáticas y facilitar la exploración de los datos. La Figura 4.2 ilustra el flujo de trabajo asociados a la programación de los módulos.

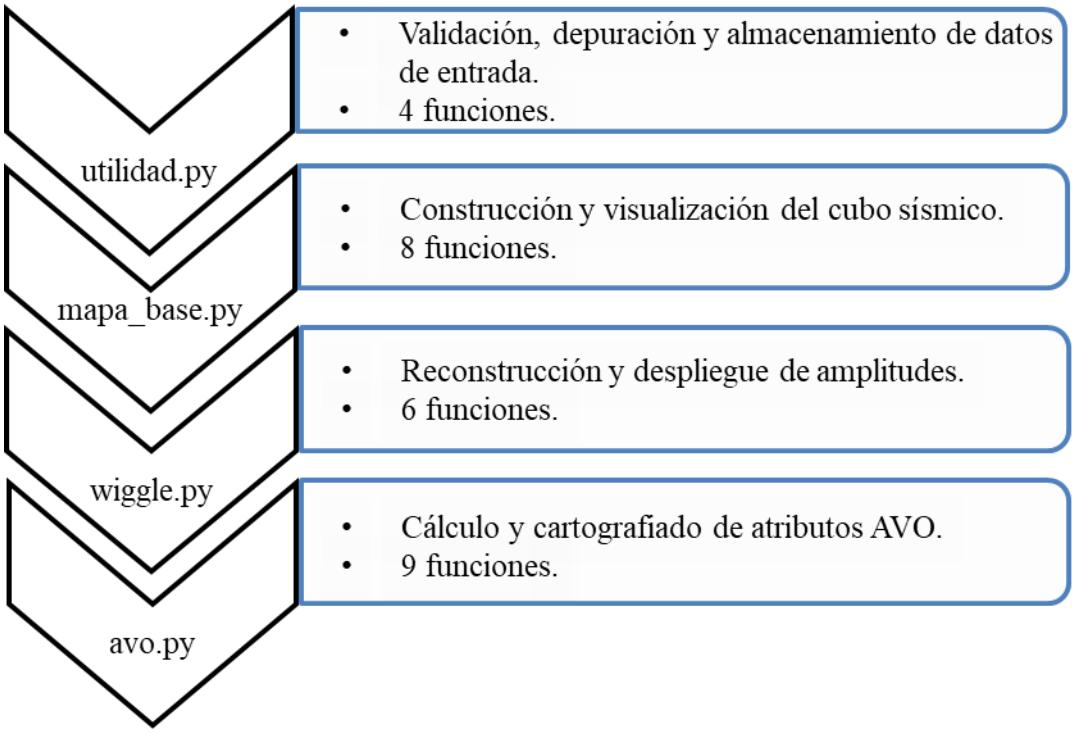

Figura 4.2. Flujo generalizado de los módulos de programación.

4.3.1. Primer módulo: inspección y organización del dato sísmico (utilidad.py)

La etapa inicial del desarrollo de la aplicación se enfocó en la inspección, extracción y almacenamiento en memoria de la información contenida en los archivos en formato SEG-Y (SEG *Technical Standards Committee*, 2017). En la inspección y la extracción se utilizó la biblioteca *SegyIO* (Kvalsvik, 2014), mientras que para el almacenamiento se utilizó la biblioteca *Pandas*. El flujo de trabajo del módulo se ilustra en la Figura 4.3. Este módulo dispone de un total de 4 funciones cuya explicación y diagrama de flujo se muestra en el Apéndice D.1.

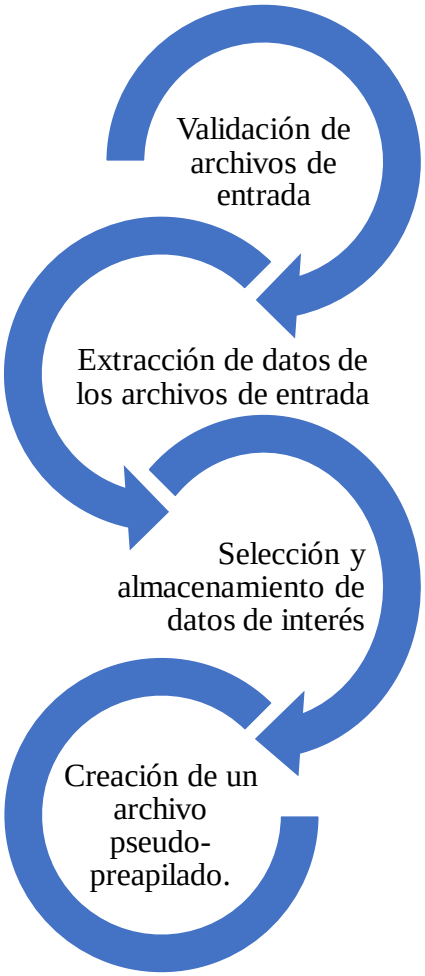

Figura 4.3. Flujo de trabajo del módulo utilidad.py

La primera función del módulo utilidad.py, *validar_archivos*, verifica la existencia y el formato de los archivos de entrada, que son utilizados posteriormente por las funciones *organizar_pozos* y *organizar_cubo*. La función *organizar_cubo* utiliza archivos sísmicos con extensiones .SGY o .SEGY, mientras que la función *organizar_pozos* utiliza un archivo de texto (.TXT). Si los archivos suministrados no existen o no cumplen con los formatos, la función *validar_archivos* interrumpe la ejecución de la aplicación.

Luego se codificó la instrucción *organizar_cubo*, que por medio del método *segyio.attributes(segyio.TraceField.byte)*, extrae del encabezado de las trazas (*Trace Header*) del archivo SEG-Y la información correspondiente a las coordenadas (x, y) y su respectivas líneas sísmicas (*inline* y *crossline*). Estos valores se encuentran en los bytes 181-184, 185-188, 189-192 y 193-196 respectivamente (Apéndice C). Luego utiliza los métodos *arg.min* y *arg.max* del paquete *Numpy* para identificar los puntos que se encuentran en las esquinas del polígono asociado al levantamiento sísmico. Finalmente, emplea el método *pandas.DataFrame* para almacenar estos puntos en memoria RAM dentro de una estructura tabular denominada "*dataframe*" (Figura 4.4).

	iline	xline	utmx	utmy
0	1189	2518	399329.9	8466735.8
1	1199	2508	399546.1	8466799.1
2	1189	2508	399425.6	8466655.5
3	1199	2518	399450.4	8466879.5

Figura 4.4. *Dataframe* generado por la función *organizar_cubo*. Se resumen las coordenadas de las trazas que definen al polígono asociado al levantamiento sísmico.

Posteriormente, se diseñó la función *organizar_pozos*, que a partir de un archivo de texto (Figura 4.5) genera un *dataframe* (Figura 4.6) con la información de aquellos pozos que estén dentro de los límites de coordenadas del cubo sísmico. El archivo de texto debe incluir: nombre del pozo, líneas sísmicas *inline* y *crossline*, coordenadas y la profundidad alcanzada. Cada ítem listado debe estar separado por un espacio.

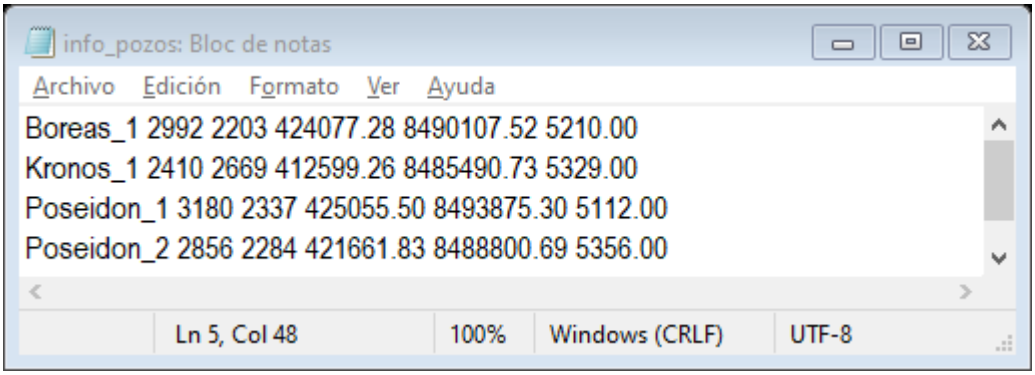

Figura 4.5. Estructura del archivo de texto para ser usado por la función *organizar_pozos*. Los datos están organizados por filas e incluye: nombre del pozo, coordenadas sísmicas inline y crossline, coordenadas utmx y utmx y finalmente la profundidad alcanzada.

nombre	cdp_iline	cdp_xline	utmx	utmy	profundidad
Boreas_1	2992	2203	424077.28	8490107.52	5210.00
Kronos_1	2410	2669	412599.26	8485490.73	5329.00
Pharos_1	3255	1392	435010.57	8487361.39	5220.31
Poseidon_1	3180	2337	425055.50	8493875.30	5112.00
Poseidon_2	2856	2284	421661.83	8488800.69	5356.00
Poseidon_North_1	3525	2400	428611.91	8499338.27	5287.52
Proteus_1ST2	2766	1493	428152.83	8481148.02	5249.71
Torosa_1	1194	2513	399434.90	8466761.60	4671.85

Figura 4.6. *Dataframe* generado por la función *organizar_pozos*. Se resumen los datos de los pozos suministrados dentro del archivo de texto.

Por último, se programó la función *compilar_archivos* que agrupa los archivos apilados a diferentes ángulos en un solo, denominado volumen pseudo-preapilado (en caso de disponer de este tipo de datos iniciales). La función primero extrae la información del encabezado binario (*Binary Header*) de los datos utilizando el método *segyio.spec* y posteriormente, inicializa el nuevo volumen con *segyio.create*. Luego, empleando los métodos *segyio.header* y *segyio.trace* organiza las trazas sísmicas de los archivos apilados a diferentes ángulos en este nuevo volumen. Para garantizar la posición correcta de las trazas sísmicas en este nuevo archivo, se le asignó al encabezado de *offset* de cada traza (byte 37-40) un valor igual a la media aritmética del rango de angulos correspondiente al archivo de procedencia. La Figura 4.7 ilustra el proceso de generación del volumen pseudo-preapilado, el cual facilita la manipulación de las trazas por otras funciones que utilizan el método *segyio.gather(inline, crossline, offset)*.

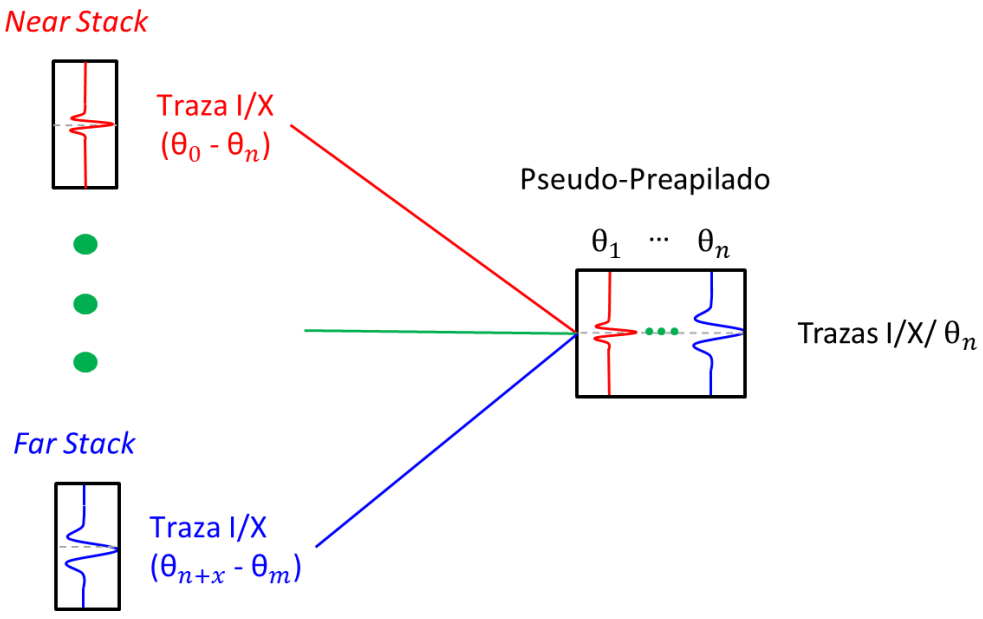

Figura 4.7. Ejemplo ilustrativo de la función *compilar_archivos*, que agrupa trazas de archivos apilados a diferentes ángulos en un solo volumen. Donde, $(\theta_0 - \theta_n)$ es el rango angular en el apilamiento del primer archivo, $(\theta_{n+x} - \theta_m)$ el rango angular del archivo *N*, θ_1 y θ_n las medias aritmédicas asociadas a los rangos angulares utilizados en el apilamiento de los archivos manipulados.

4.3.2. Segundo módulo: visualización 2D del dato sísmico (mapa_base.py)

El segundo módulo se centró en el uso de la biblioteca *Holoviews* para cartografiar el mapa de los elementos que componen el cubo sísmico: polígono del levantamiento sísmico, pozos, líneas sísmicas y trazas de intersección. Se incluyeron métodos de la biblioteca *Panel* para aumentar la interactividad de los resultados. El flujo de trabajo del módulo se resume en la Figura 4.8 y dispone de un total de 10 funciones cuya explicación y diagrama de flujo se describen en el Apéndice D.2.

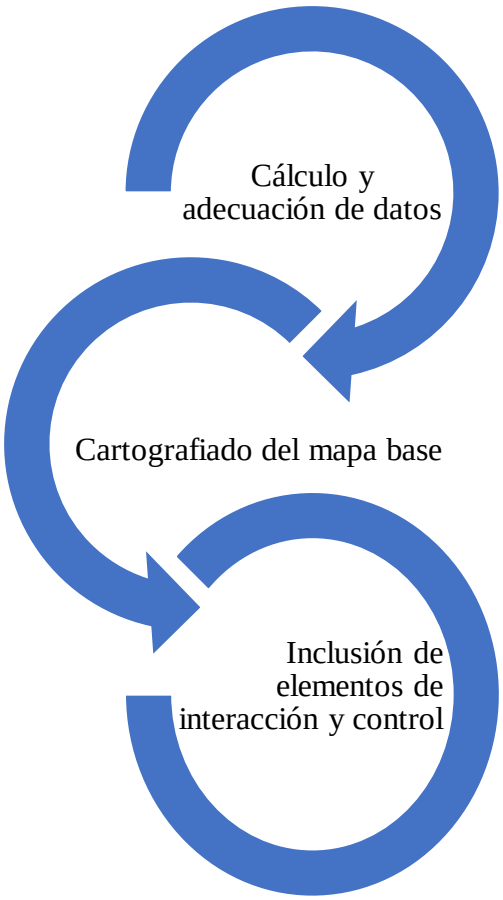

Figura 4.8. Flujo de trabajo del módulo mapa_base.py

Para cartografiar el levantamiento sísmico (Figura 4.9) se codificó la función *plot_poligono* que utiliza el *dataframe* generado por la función *organizar_cubo y* el método *holoviews.Curve* de la biblioteca *Holoviews.* Luego, se programó la función *plot_pozos* que emplea el *dataframe* generado por la función *organizar_pozos* y el método *Holoviews.Scatter* (biblioteca *Holoviews*) con la finalidad de cartografiar los pozos dentro de este (Figura 4.10).

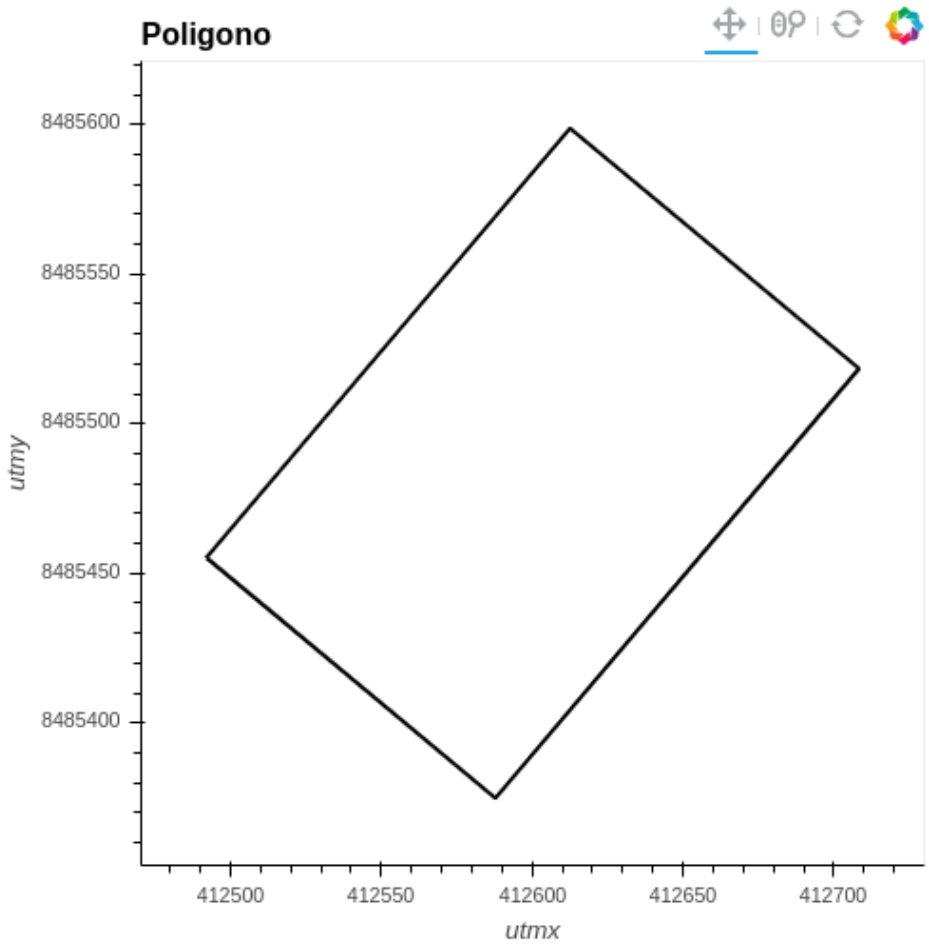

Figura 4.9. Vista en planta del polígono en representación al levantamiento sísmico (función *plot_poligono*). Observese en la parte superior los elementos de interacción.

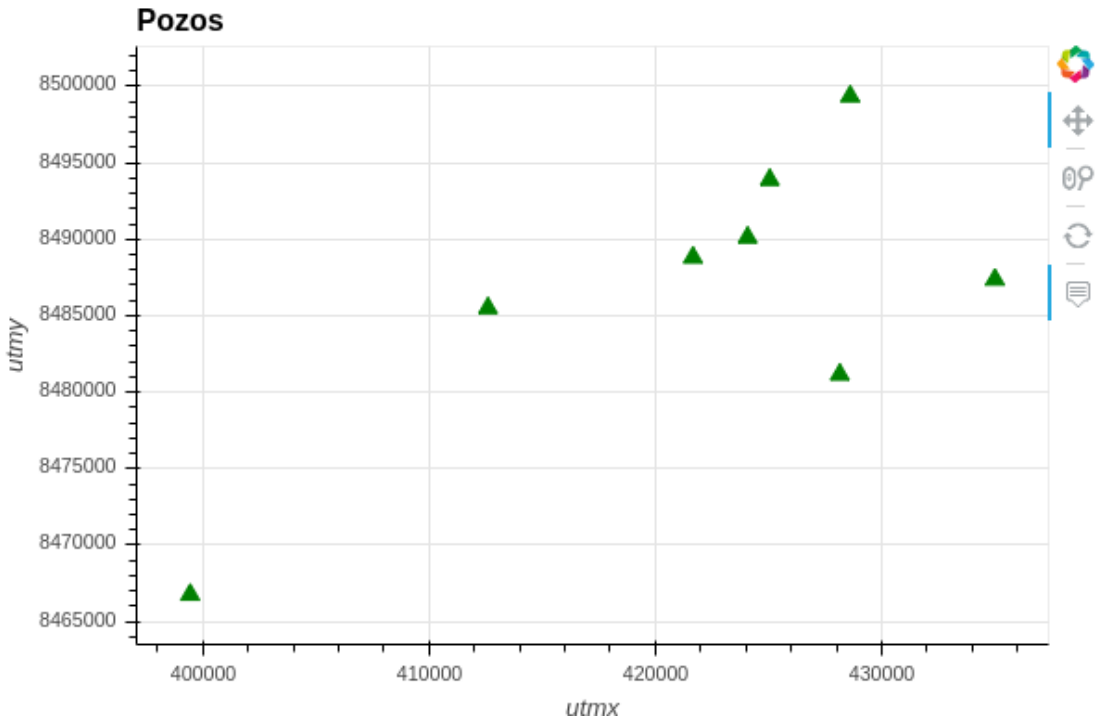

Figura 4.10. Vista en planta de los pozos ubicados dentro del levantamiento sísmico (función. *plot_pozos).* Obsérvese en la parte lateral izquierda los elementos de interacción.

Posteriormente, se codificó la función *dataframe_lineas_sismicas*, que utilizando la Ecuación 4.1, calcula la cantidad de trazas en las dos primeras líneas sísmicas identificadas en la Figura 4.11 como $Inline_i$ y $Crossline_i$. Luego, dada la simetría del cubo y la cantidad de trazas por muestra se utilizó el método *numpy.linspace* del paquete *Numpy* para calcular las coordenadas de las trazas. Seguidamente, se empleó el método *pandas.DataFrame* para almacenar la información calculada en un *dataframe*. Subsiguientemente, se programó la función *plot_lineas_sismicas* que utiliza este último *dataframe* para calcular mediante diferencias vectoriales (Ecuación 4.2) los puntos pertenecientes a cualquiera de las líneas sísmicas diferentes a las iniciales ($Inline_i$ y $Crossline_i$) para ser próximamente cartografiadas por el método *holoviews.Curve*. La Figura 4.11 ilustra el cálculo y cartografiado de una línea arbitraria conformada por los puntos CD, utilizando el procedimiento anterior.

$$NTrazas_{DS} = |DS_{Menor} - DS_{Mayor}| + 1 \quad \text{(Ecuación 4.1)}$$

Donde:

$NTrazas_{DS}$: Número de trazas en una dirección sísmica (*inline* o *crossline*).

DS_{Menor}: Número de la menor línea respecto a una dirección sísmica

DS_{Mayor} : Número de la mayor línea respecto a una dirección sísmica

+1: Condición de numeración de trazas puesto que esta comienza en 1 y no en 0.

$$Mxline_{Inline_n} = Mxline_{Inline_0} - mxline_{Inline_0} + mxline_{Inline_n} \text{ (Ecuación 4.2)}$$

Donde:

$Mcrossline_{inline_n}$: Intersección entre la máxima *crossline* y la *inline* buscada.

$Mcrossline_{inline_0}$: Intesección entre la máxima *crossline* e *inline* inicial.

$mcrossline_{inline_0}$: Intersección entre mínima *crossline* e *inline* inicial.

$mcrossline_{inline_n}$: Intersección entre mínima *crossline* e *inline* buscada.

La Ecuación 4.2 permite el cálculo de las coordenadas y las líneas sísmicas en dirección *inline* con un espaciamiento entre líneas igual uno. La ecuación también es válida para cálculos en dirección *crossline* si se invierten las direcciones asociadas a los índices y subíndices de cada factor en la ecuación.

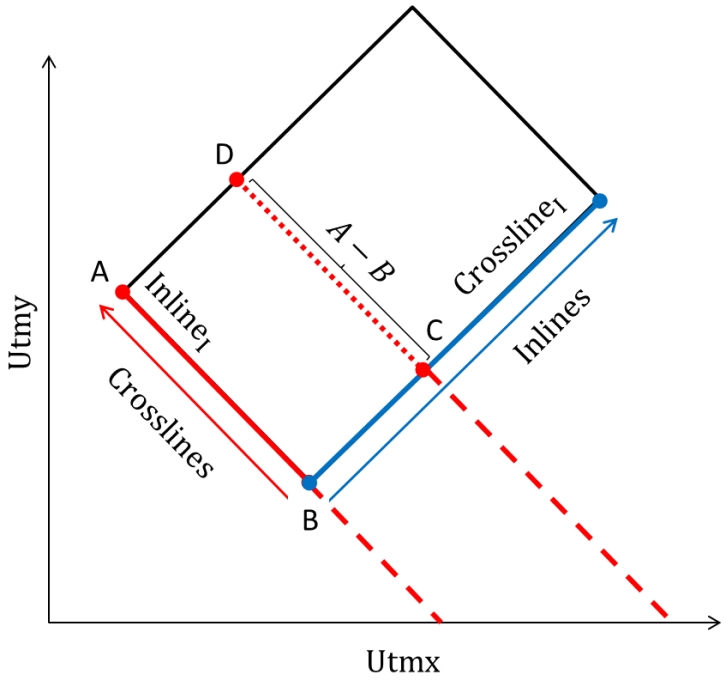

Figura 4.11. Razonamiento aplicado para el cálculo del punto D de una línea X en dirección *inline*, dados los puntos A, B y C obtenidos a partir de la función *Linspace* de *Numpy* (Función *dataframe_lineas_sismicas*). Nótese el incremento de las inlines hacia el NE (flecha azul) mientras que las crosslines aumentan hacia el NO (flecha roja).

Se creó la función *intersección_sismica*, que utiliza las Ecuaciones 4.2 y 4.3 para calcular las coordenadas y la secuencia de numeración de la traza (encabezado *tracf*, byte 5-8) en la intersección de las líneas sísmicas. Luego se almacenan estos datos en memoria empleando un objeto tipo lista que posteriormente son cartografiados por la función *plot_intersesccion_sismica* utilizando el método *holoviews.Scatter*. La Figura 4.12 ilustra la lógica tras la Ecuación 4.3.

$$tracf = \Delta Line_{inline} * NTrazas_{crossline} + \Delta Line_{crossline} + 1 \quad \text{(Ecuación 4.3)}$$

Donde:

$\Delta Line_{Inline}$: Diferencia entre la línea buscada en dirección *inline* y la *inline* inicial.

$\Delta Line_{Crossline}$: Diferencia entre la línea buscada en dirección *crossline* y la *crossline* inicial.

$NTrazas_{Crossline}$: Cantidad de trazas sísmicas en dirección *crossline*.

+1: Condición de numeración de *tracf* puesto que este comienza en 1 y no en 0.

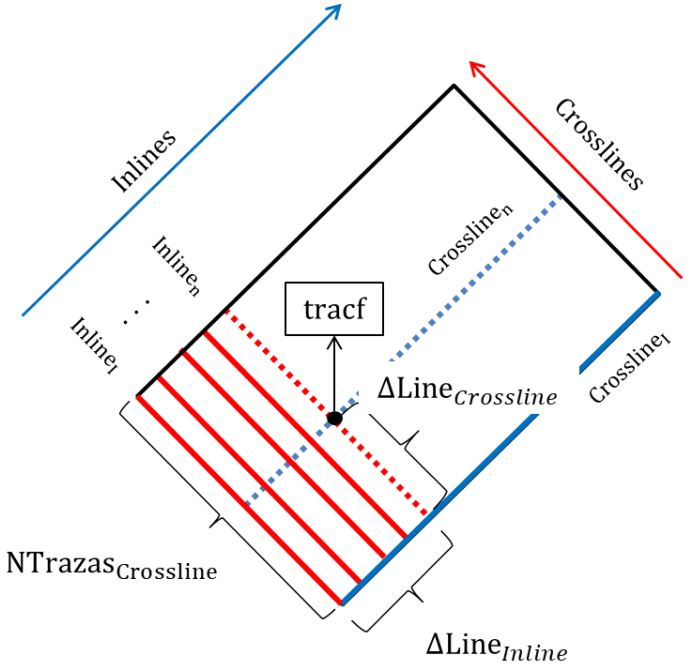

Figura 4.12 . Razonamiento aplicado para la formulación de la Ecuación 4.3 en el cálculo del encabezado *tracf*. Nótese el incremento de las *inlines* hacia el NE (flecha azul) mientras que las *crosslines* aumentan hacia el NO (flecha roja).

Se codificó la función *gather_box* que utiliza las Ecuaciones 4.4, 4.5 y la lista generada por la función *intersección_sismica* para crear un *dataframe* con todos los puntos dentro de un área de selección alrededor de la intersección de las líneas sísmicas, con la finalidad de facilitar la búsqueda y el despliegue de las trazas en otros módulos. Posteriormente, se programó la función *plot_gather_box* que utiliza este último *dataframe* y los métodos *holoviews.Scatter* y *holoviews.Curve* para cartografiar esta subregión de estudio mostrada en la Figura 4.13 en la intersección entre las líneas sísmicas iniciales. Se agregó una rutina para solucionar problemas de borde asociados a intersecciones en el borde del levantamiento sísmico.

$$Lm_{DS} = LS_{tracf} - (nd//2 * Itrazas_{DS}) \qquad \text{(Ecuación 4.4)}$$

$$LM_{DS} = LS_{tracf} + (nd//2 * Itrazas_{DS}) \qquad \text{(Ecuación 4.5)}$$

Donde:

Lm_{DS}: Límite inferior de la ventana en una dirección sísmica (*inline* o *crossline*).

LM_{DS}: Límite superior de la ventana respecto a una dirección sísmica.

$nd//2$: Parte entera de la mitad de nd (cantidad de trazas alrededor de la intersección).

$Itrazas_{DS}$: Intervalo entre líneas respecto a una dirección sísmica.

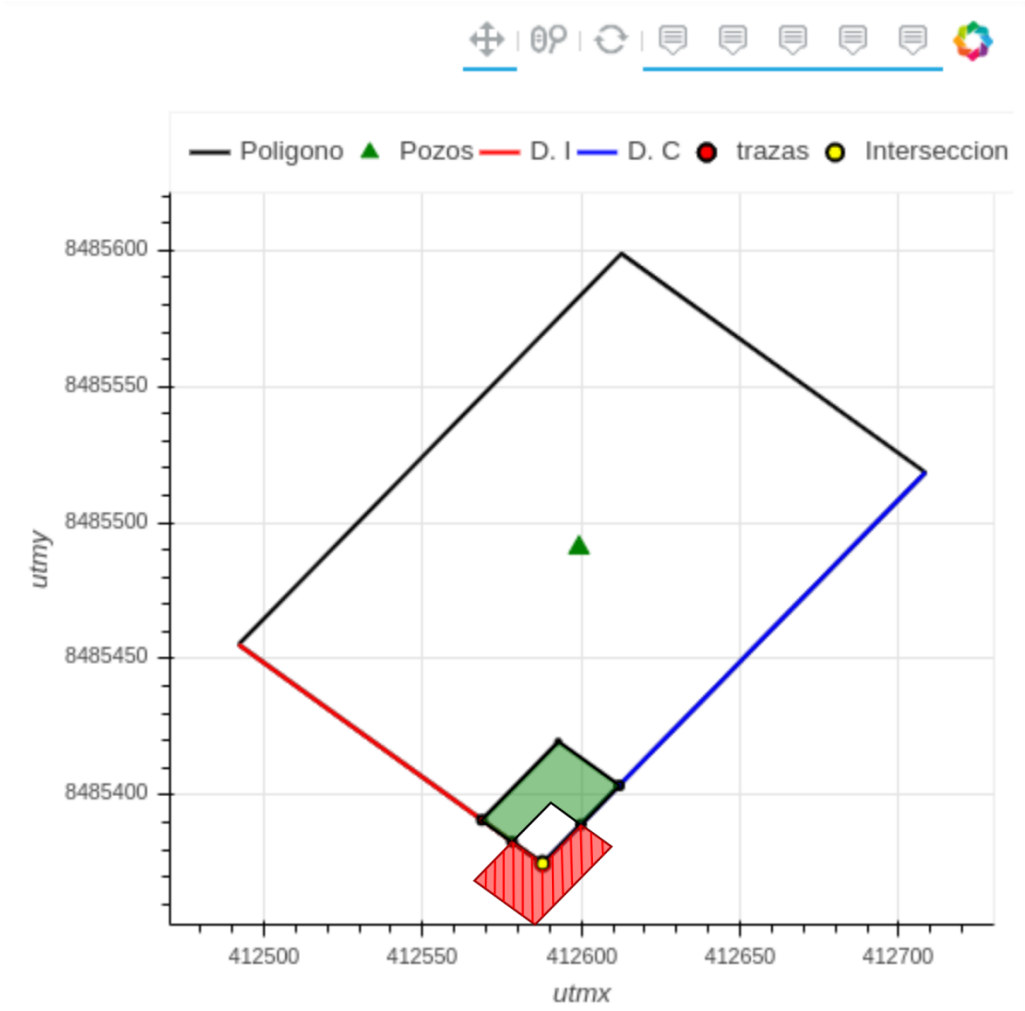

Figura 4.13. Problema de borde en el cálculo del área de selección de trazas cuando la intersección de las líneas sísmicas coincide con uno de los bordes del levantamiento sísmico. Se marca en rojo el área en conflicto, mientras que en verde, el área adicionada para solucionar los problemas de borde.

Finalmente, se programó la función *mapa_base* para generar el mapa base del levantamiento sísmico, empleando el método *holoviews.Overlay* de la biblioteca *Holoviews* y los objetos generados por las funciones *plot_poligono*, *plot_pozos*, *plot_lineas_sismicas*, *plot_interseccion_sismica*, *plot_gather_box*. Se incluyó en la función *mapa_base* tres nuevos elementos: un selector (Figura 4.14a) a modo de leyenda que permite atenuar elementos del gráfico, una herramienta de superposición (*Hover tool;* Figura 4.14b) que visualiza la información al superponer el cursor sobre los elementos del gráfico y finalmente, un panel de control (Figura 4.15) que permite manipular el mapa al involucrar los siguientes métodos de la biblioteca *Panel*:

- *Widgets.IntSlider* a modo de controles deslizantes para facilitar la selección de las líneas sísmicas y la cantidad de trazas alrededor de la intersección (Figura 4.15a).
- *Widgets.Select* que permite seleccionar los pozos en el gráfico (Figura 4.15b).

4.3.3. Tercer módulo: despliegue de trazas sísmicas (wiggle.py)

El tercer módulo despliega las trazas que están dentro del área de selección generada por la función *gather_box*. Para la manipulación y despligue de las trazas sísmicas se utilizaron las bibliotecas *SegyIO* (Kvalsvik, 2014) y *Holoviews* respectivamente. Se incluyeron métodos de la biblioteca *Panel* para aumentar la interactividad en el gráfico resultante. El flujo de trabajo del módulo se resume en la Figura 4.16 y dispone de un total de 6 funciones cuya explicación y diagrama de flujo se detallan en el Apéndice D.3.

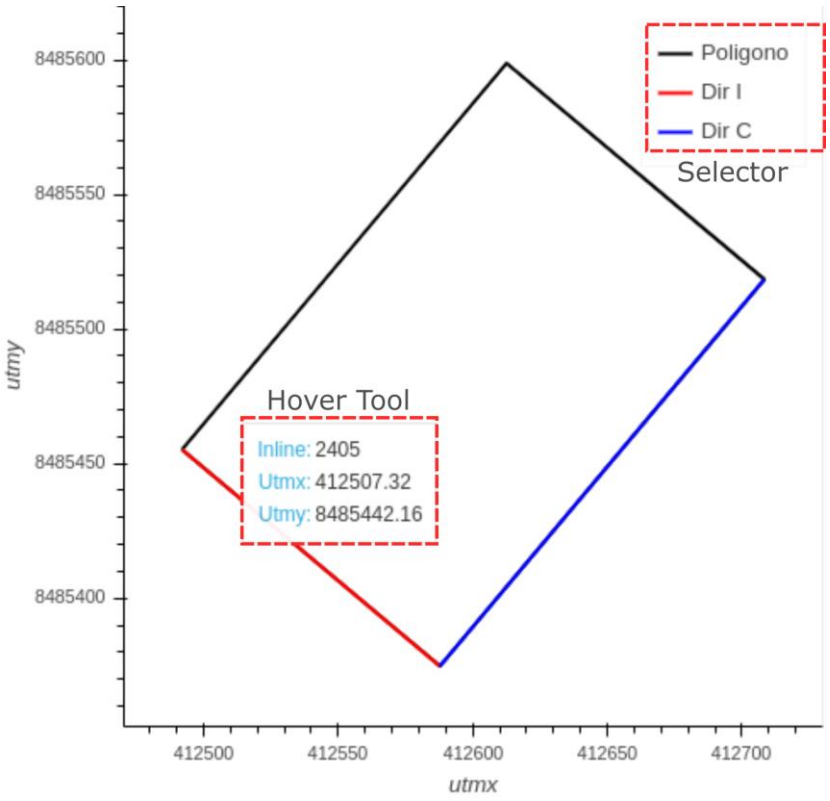

Figura 4.14. Despliegue de la herramienta de superposición (*Hover tool*) al posicionar el cursor sobre los elementos del gráfico (líneas sísmicas). Se muestra en la parte superior derecha el elemento selector.

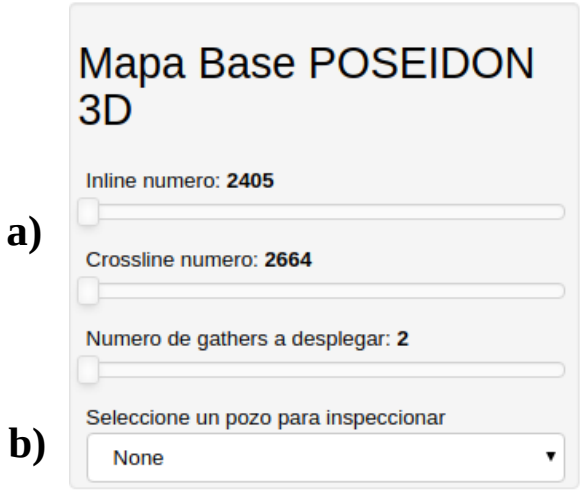

Figura 4.15. Elementos de la biblioteca *Panel* para aumentar la interactividad del mapa base. (a) Señala las barras deslizantes para el control de la ventana de estudio. (b) Marca el panel de selección para la inspección rápida de pozos.

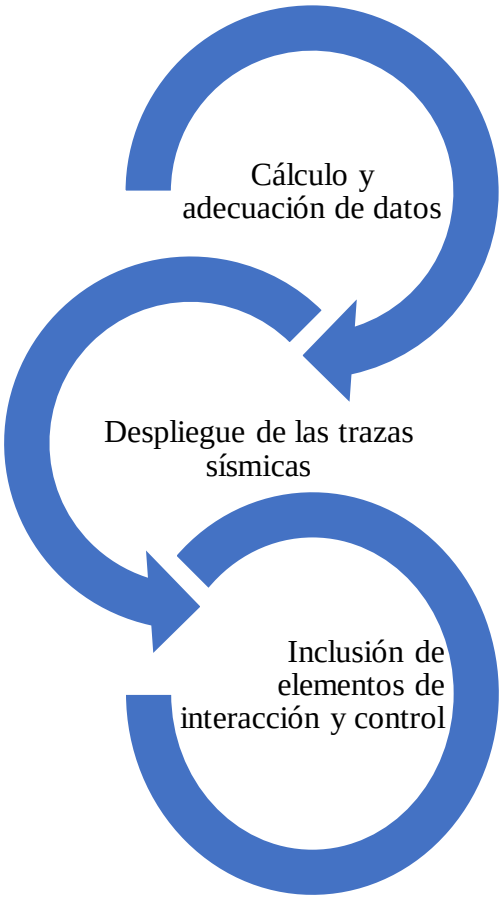

Figura 4.16. Flujo de trabajo del módulo wiggle.py.

La primera función del tercer módulo, *eje_tiempo*, extrae del archivo pseudo-preapilado el intervalo de muestreo y la cantidad de muestras por traza (bytes 115-116 y 117-118 respectivamente) utilizando el método *segyio.attributes(segyio.TraceField.byte)*. Luego construye un vector en representación al eje vertical o eje de tiempo por medio del método *numpy.arange* del paquete *Numpy*.

Se programó la función *min_max_lineas_sismicas* para extraer del área de selección, generada por la función *gather_box* del módulo *mapa_base*, el rango de las líneas sísmicas en una dirección preferencial seleccionada por el usuario (*inline* o *crossline*). Seguidamente, se codificó la función *interSpline_guardar* que primero extrae el contenido de amplitud de las trazas utilizando el método *segyio.gather(inline, crossline, offset*), luego interpola estas conveniencia mediante el método *inter1d* del paquete *Scipy* y por último, almacena estos valores en un *dataframe* (Figura 4.17) discriminando entre amplitudes positivas y negativas.

	tiempo	linea_sismica	iline	xline	amplitud_12	amplitud_positiva_12	amplitud_negativa_12
375	1500.0	Inline	2405	2664	45.0	45.0	0.0
376	1504.0	Inline	2405	2664	-1.0	0.0	-1.0
377	1508.0	Inline	2405	2664	-26.0	0.0	-26.0
378	1512.0	Inline	2405	2664	-10.0	0.0	-10.0
379	1516.0	Inline	2405	2664	-12.0	0.0	-12.0

Figura 4.17. Sección del *dataframe* resultante de la función *interSpline_guardar*. Nótese las columnas en representación a la separación de polaridad realizada por ángulo de incidencia para cada muestra a lo largo del eje del tiempo (eje Z).

Posteriormente, se programó la función *plot_gather* que, a partir del *dataframe* generado por la función *interSpline_guardar* y los métodos *holoviews.Curve* y *holoviews.Area* cartografía las trazas sísmicas con su respectiva polaridad. Se le asignó el color rojo a las amplitudes negativas y azul a las amplitudes positivas. Finalmente, se utilizó el método *holoviews.Overlay* y la función *factor_escalado* para agrupar y escalar las trazas en un solo objeto (*gather*) que se muestra en la Figura 4.18.

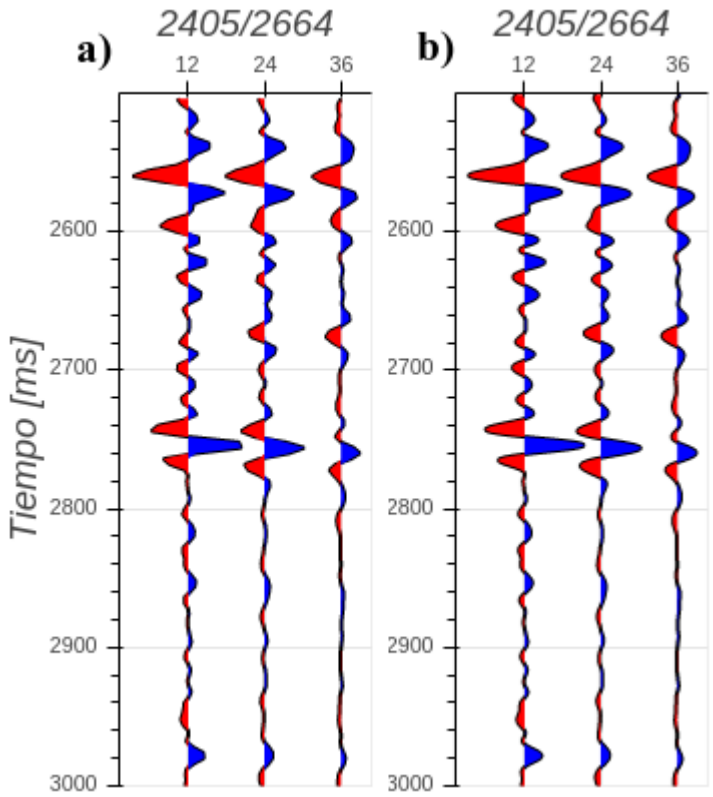

Figura 4.18. Trazas sísmicas desplegadas (*gather*) por la función *plot_gather*. (a) Muestra el despliegue de las ondículas sísmicas utilizando los parámetros nativos del dato sísmico. (b) Muestra las ondículas posterior al proceso de interpolación para mejorar el trazado de las mismas.

La última función del módulo, *get_wiggle*, combina el objeto resultante de la función *plot_gather* con un elemento selector que atenua las amplitudes a conveniencia y la herramienta de superposición (*Hover tool*) para desplegar la información de las amplitudes en un instante de tiempo específico. Se incluyó el método *holoviews.NdLayout* para agrupar en un solo objeto los *gathers* a mostrar en función del área de selección (mapa base) y finalmente, un panel de control que involucra los siguientes métodos de la biblioteca *Panel*:

- *Widgets.RadioButtonGroup* para controlar la dirección de despliegue de las amplitudes seleccionadas: *inline* o *crossline* (Figura 4.19a).
- *Widgets.IntRangeSlider* a modo de control deslizante para seleccionar el tope y la base de las amplitudes a mostrar. Lo anterior permite disminuir la demanda en memoria RAM en caso de no disponer de suficientes recursos para cargar todo el contenido de las trazas (Figura 4.19b).
- *Widgets.IntSlider* para controlar el intervalo de muestreo en el proceso de interpolación. Por defecto, utiliza el intervalo nativo del archivo SEG-Y. Disminuir este parámetro permite mejorar la presentación de las ondículas (Figura 4.19c).

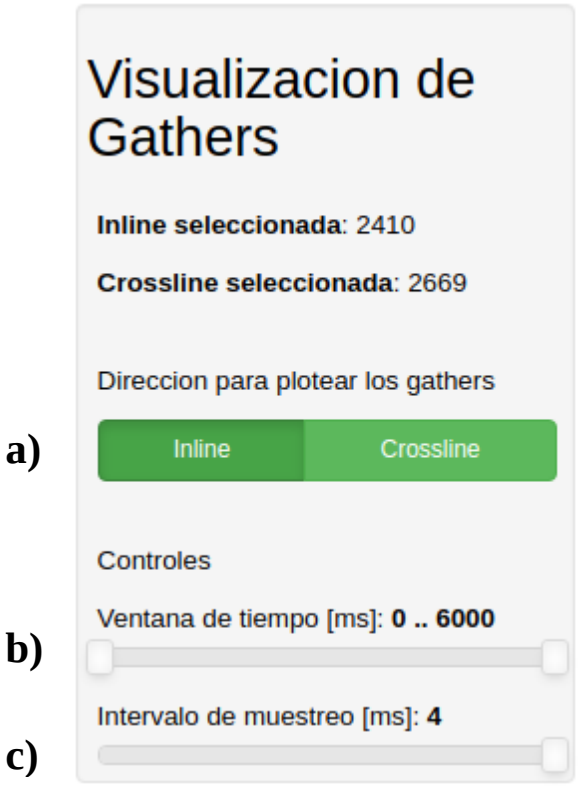

Figura 4.19. Elementos de la biblioteca *Panel* para aumentar la interactividad en el despliegue de las ondículas sísmicas (*gathers).* (a) Señala los botones para la selección de las direcciones sísmicas. (b) y (c) muestran las barras deslizantes para controlar el tamaño de la ventana de tiempo y el intervalo de muestreo respectivamente.

4.3.4. Cuarto módulo: cálculo, almacenamiento y cartografiado de AVO (avo.py)

El cuarto Módulo se diseñó para calcular, almacenar y visualizar los atributos de AVO (intercepto y gradiente) y su respectiva localización en un mapa base. Este módulo utiliza los métodos del paquete *Scipy* y las bibliotecas *Pandas*, *SegyIO* y *Holoviews* respectivamente. Se incluyeron métodos de la biblioteca *Panel* para aumentar la interactividad de los resultados. El flujo de trabajo del módulo se resume en la Figura 4.20 y dispone de un total de 9 funciones cuya explicación y diagrama de flujo se describen en el Apéndice D.4.

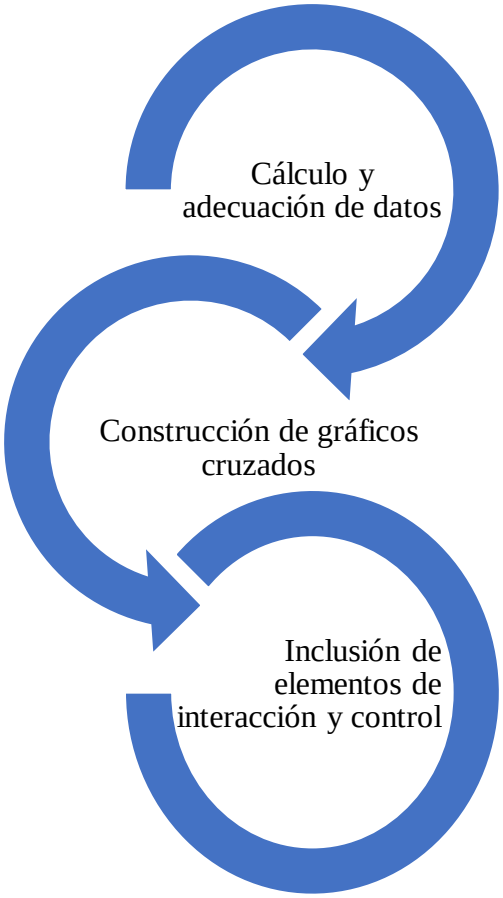

Figura 4.20. Flujo de trabajo del módulo avo.py

Se seleccionó como métodología de cálculo de atributos AVO la regresión lineal de Shuey (1985) por la pequeña cantidad de datos en los volúmenes de entrada. Por ello, se programó la función *calculo_guardado_atributos_AVO* que utiliza el método *stats.lineregress* de la biblioteca *Scipy* para resolver la formulación matricial de Shuey (Ecuación 4.6) que calcula: intercepto, gradiente y el coeficiente de correlación, valor P y error estándar de la regresión lineal. Estos valores son almacenados en archivos SEG-Y previamente creados por la función *archivos_avo* que genera copias de los volúmenes de entrada iniciales por medio de los métodos *segyio.spec* y *segyio.create*. La ecuación 4.6 corresponde a una ecuación matricial de la forma $b = Ac$, que también puede ser resuelta por mínimos cuadrados aplicando: $c = (A^T A)^{-1}(A^T b)$ (Avseth, Mukerji y Mavko, 2005).

$$\begin{pmatrix} R_{(x_0)} \\ R_{(x_1)} \\ \vdots \\ R_{(x_n)} \end{pmatrix} = \begin{pmatrix} 1 & sin^2\theta_{(t,x_0)} \\ 1 & sin^2\theta_{(t,x_1)} \\ \vdots & \vdots \\ 1 & sin^2\theta_{(t,x_0)} \end{pmatrix} \begin{pmatrix} R_{(t,0)} \\ G_{(t)} \end{pmatrix} \qquad \text{(Ecuación 4.6)}$$

Donde:

$R_{(x_n)}$: Coeficiente de reflexión de la traza *N* contenida en el SEG-Y.

$\theta_{(t,x_n)}$: El ángulo de incidencia para cada muestra de la traza.

$R_{(t,0)}$ y $G_{(t)}$: Los atributos intercepto y gradiente.

Para acelerar el cálculo de los parámetros mencionados anteriormente, se codificó la función *avo_multiproceso* que, en primera instancia emplea un objeto generador de *Python* (*Python Generator*), producido por la función *generador_argumentos*, para suministrar en tiempo real y sin consumo de memoria, los argumentos necesarios para el método *multiprocessing.Pool.map* de la biblioteca *Multiprocessing*. Este último añade un administrador de procesos el cual ejecuta en paralelo la función *calculo_guardado_atributos_AVO*. En este tipo de ejecución se realizan cálculos simultaneos en cada uno de los procesadores disponibles en el equipo. Varias pruebas con *avo_multiproceso*, determinaron que al asignar los parámetros *maxtasksperchild* =1 y *chunksize* = 1 del método *multiproccessing.Pool.map*, se logra aumentar el rendimiento a largo plazo.

Concluidos los cálculos, la función *organizacion_atributos* se encarga de organizar en un *dataframe* (Figura 4.21) los atributos de acuerdo a una ventana seleccionada por el usuario. Luego, se codificó la función *grafico_cruzado* que emplea el *dataframe* anterior, el método *holoviews.Points* y la herramienta *Box_select* para generar un gráfico cruzado interactivo, de ejes y escala de colores variables y seleccionables. Por defecto, se grafica intercepto vs gradiente con una escala de colores controlada por el error estándar (Figura 4.22). A partir del gráfico cruzado el usuario puede seleccionar las respuestas favorables utilizando la herramienta de selección (*Box_Select*) que subsecuentemente, son graficadas en un mapa base dinámico a partir del método *holoviews.DynamicMap* (Figura 4.23).

	inline	crossline	utmx	utmy	tiempo	gradiente	intercepto	cCorr	error	desvE
0	2410	2669	412600.2	8485486.8	3000.0	-13.073828	1.414968	-0.993951	0.070060	1.444621
1	2410	2669	412600.2	8485486.8	3004.0	7.370966	-2.028215	0.733716	0.475568	6.825867
2	2410	2669	412600.2	8485486.8	3008.0	36.995590	-9.167079	0.989274	0.093325	5.462513
3	2410	2669	412600.2	8485486.8	3012.0	52.712326	-14.070242	0.999278	0.024196	2.004396
4	2410	2669	412600.2	8485486.8	3016.0	52.295303	-15.659876	0.993951	0.070060	5.778486

Figura 4.21. *Dataframe* generado por la función *organizacion_atributos*. Se resumen los atributos AVO y parámetros estadísticos para las 5 primeras trazas del dato.

Se programó la función *visualizacion_avo* que combina los objetos resultantes de la función *grafico_cruzado* con la función *visualizacion_de_linea* que despliega una línea sísmica auxiliar en función de la selección del usuario. Finalmente, se le añadió a la función *visualizacion_avo* un panel de selección para controlar los gráficos anteriormente mencionados, empleando los siguientes métodos de la biblioteca *Panel*:

- *Widgets.IntRangeSlider* a modo de controles deslizantes para manejar las dimensiones de la ventana a visualizar en el gráfico cruzado (Figura 4.24a).
- *Widgets.Select* para controlar los atributos empleados en la realización del gráfico cruzado. Por defecto, se utilizan intercepto y gradiente (Figura 4.24b).
- *Widgets.Checkbox* para activar o desactivar la visualización de la línea sísmica controlada por la ventana de selección (Figura 4.24c).
- *Widgets.RadioButtonGroup* para controlar la selección de la dirección de la línea sísmica a desplegar (Figura 4.24d).
- *Widgets.Textinput* para elegir el número de la línea sísmica a desplegar (Figura 4.24e).

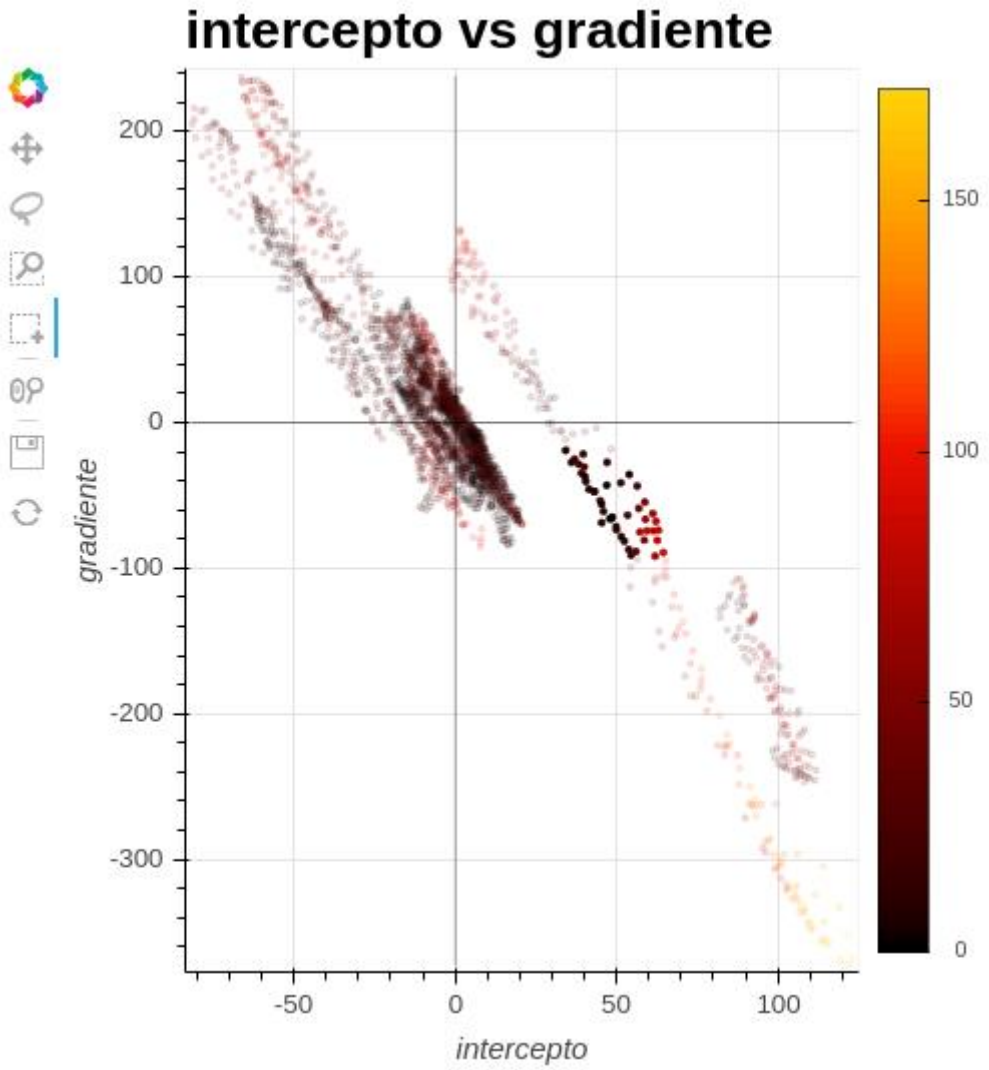

Figura 4.22. Gráfico cruzado de intercepto vs gradiente cuya escala de colores es controlada por el error estándar del calculo. La herrmienta *box_select* se marca en azul y se ubica en la zona lateral izquierda. La opacidad de los puntos no seleccionados disminuye automáticamente.

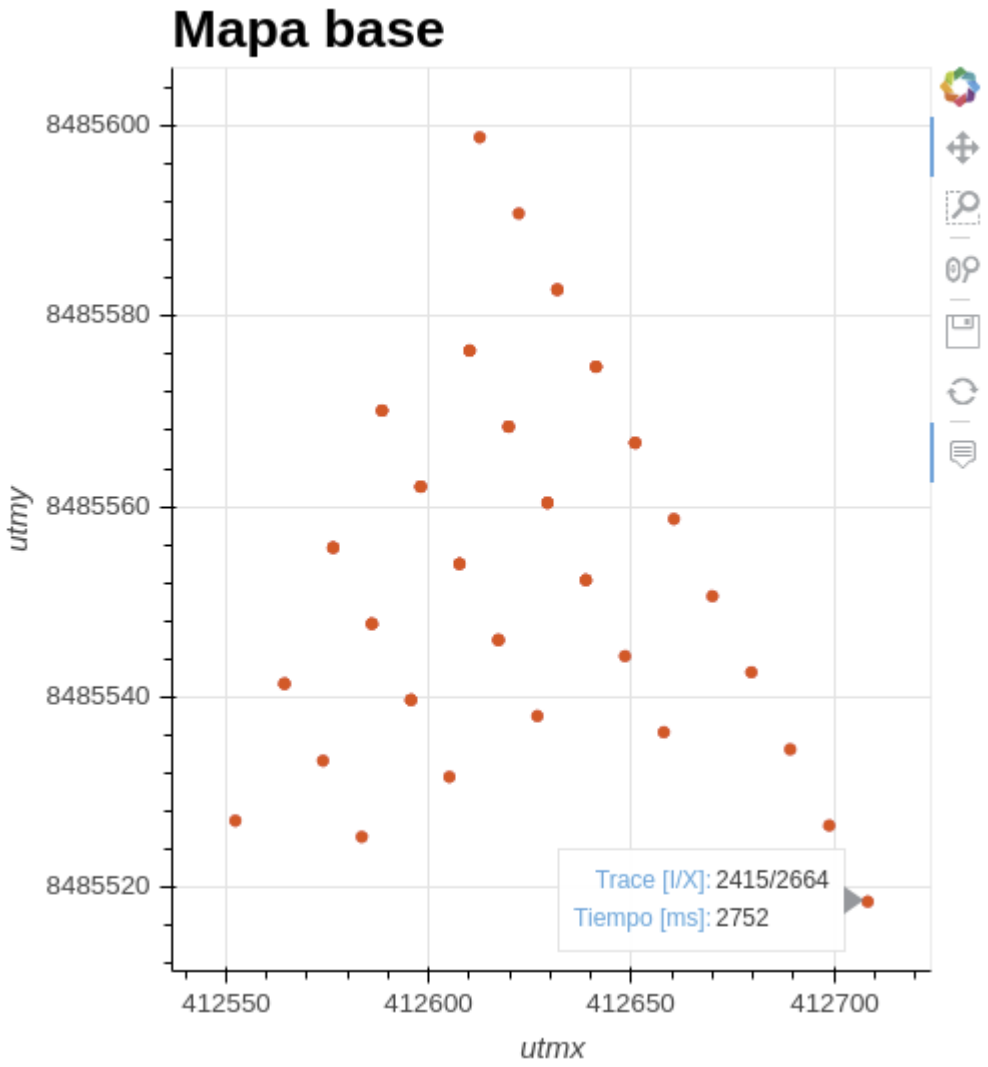

Figura 4.23. Mapa base de los puntos seleccionados en el gráfico cruzado (en Figura 4.18). Se muestra el despliegue de la herramienta de superposición (*Hover Tool*).

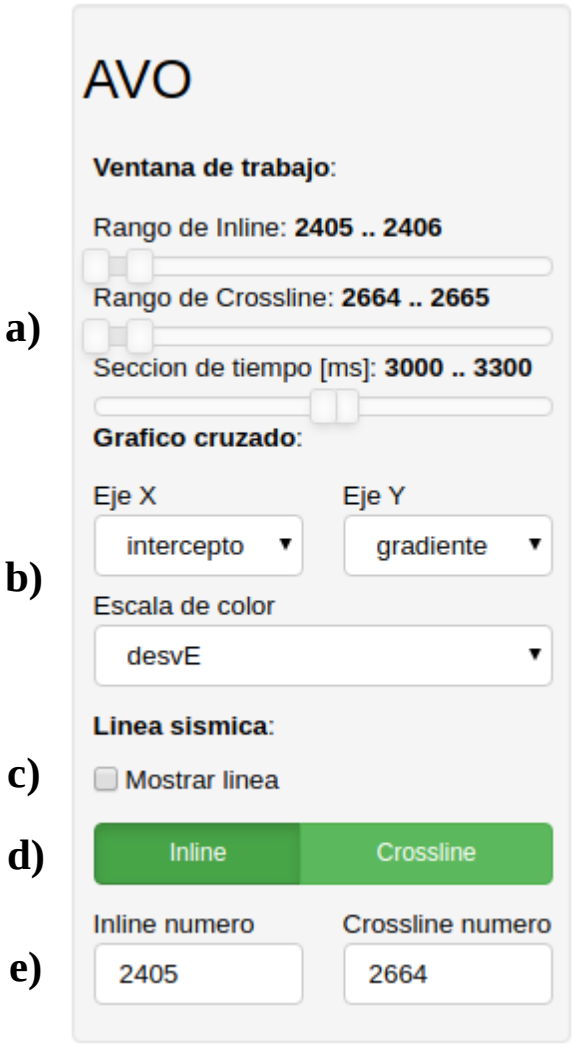

Figura 4.24. Elementos de la biblioteca *Panel* para aumentar la interactividad en el despliegue del gráfico cruzado de los atributos AVO. (a) Muestra las barras deslizantes para controlar las dimensiones de la ventana de visualización. (b) Señala las opciones de configuración para los ejes y escala de color. (c) Destaca la opción para mostrar la línea sísmica. (d) Indica los botones para la selección de las direcciones sísmicas. (e) Marca las opciones de ingreso para desplegar las líneas sísmicas.

4.4. Validación y acceso de la herramienta de caracterización

La etapa de validación se logró al comparar las trazas desplegadas y el gráfico cruzado (funciones *get_wiggle* y *visualización_avo)* con trabajos, investigaciones e interpretaciones previamente realizadas en el proyecto Poseidon 3D. El flujo de trabajo en la validación de resultados se muestra en la Figura 4.25.

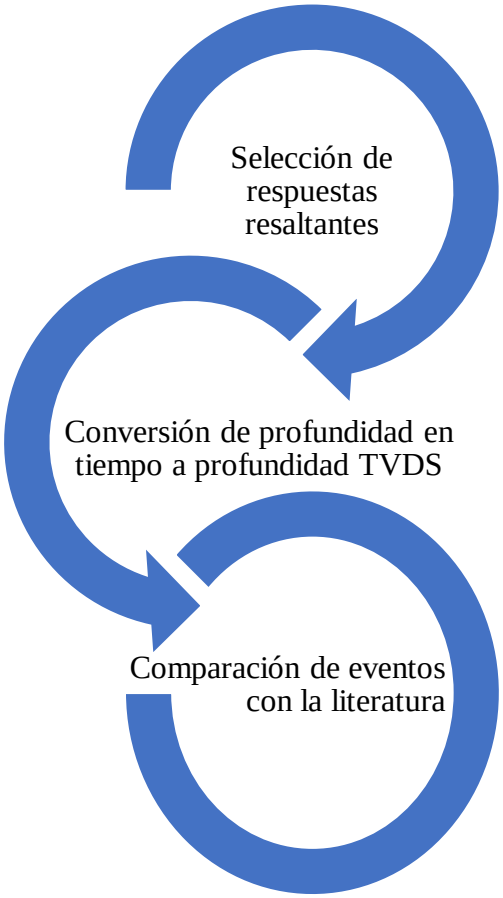

Figura 4.25. Flujo de trabajo aplicado para la validación de resultados.

4.4.1. Validación del módulo wiggle.py

Para la validación del módulo *wiggle.py* (despliegue de trazas) primero se seleccionaron aquellos eventos o reflexiones con amplitudes resaltantes. Luego, la profundidad en tiempo de estas respuestas fueron comparadas con las diferenciaciones litológicas realizadas por ConocoPhillips (2011), ConocoPhillips (2012) y Liu (2018) con la finalidad de identificar las formaciones y los horizontes. Finalmente, se correlacionó la respuesta sísmica con la información geológica disponible en la literatura.

4.4.2. Validación del módulo avo.py

La validación del módulo *avo.py* se inició con la selección de puntos con la tendencia hacia arriba y a la derecha en el gráfico cruzado de intercepto vs gradiente (función *visualizacion_avo*). Se extrapoló la profundidad TVDS (por sus siglas *True Vertical Depth Subsea*) de estos puntos utilizando las tablas de velocidad de Schlumberger (2010) relativas al pozo Kronos 1. Posteriormente, se comparó la profundidad de estos eventos con el tope de la Formación Plover seguido del tope de las arenas gasíferas mostrado en el registro petrofísico de ConocoPhillips (2011) y mencionadas por ConocoPhillips (2012).

4.5. Reproducibilidad y acceso de la herramienta

No fue posible reproducir los resultados obtenidos en aplicaciones comerciales (último objetivo específico). Sin embargo se logró comparar el alcance de la herramienta respecto a las interpretaciones presentes en la literatura. Para garantizar la reproducibilidad y el acceso de la herramienta, en concordancia con las licencias de los elementos empleados, se habilitó el código fuente (Apéndice D) en un repositorio de la plataforma *Github*. Lo anterior fomenta la transparencia y motiva la continuidad de este trabajo de investigación.

CAPÍTULO V

RESULTADOS Y ANÁLISIS

5.1. Mapa base (Módulo basemap.py)

El mapa base, generado con la función *mapa_base*, constituye el primer y único método interactivo que dispone esta herramienta para validar el dato cargado. En este, se valida el formato además de la integridad del dato. La primera validación se logra al verificar automaticamente que los archivos de entrada sean compatibles con el módulo, respetando el estándar establecido por SEG *Technical Standards Committee* (2017). La segunda validación consiste en una inspección visual de los datos cargados usando el mapa base generado.

En la Figura 5.1 se muestra el mapa base correspondiente al dato de prueba y las herramientas que permiten inspeccionar tanto la geometría como la posición geográfica de cada elemento en el cubo. Los elementos interactivos muestran con satisfacción los valores validados. Sin depender enteramente del archivo SEG-Y, se logró cartografiar en 799 ms el mapa base con los diversos elementos mostrados en la Figura 5.1. El panel de control del mapa base se muestra en el Apéndice E.1.

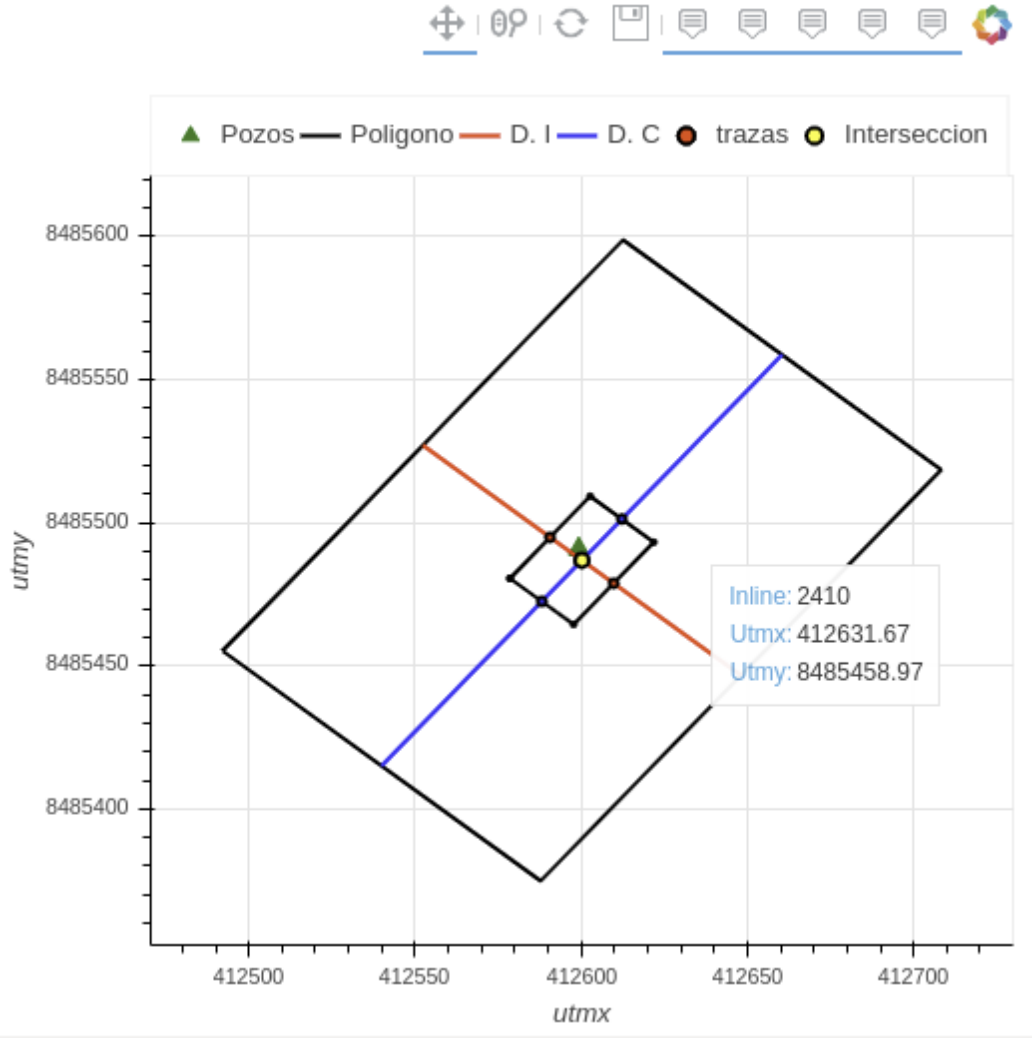

Figura 5.1. Mapa base del dato de prueba. Se muestran las líneas 2410 y 2669 (*inline* y *crossline*) cercanas al pozo Kronos 1, el área de selección para el análisis de trazas en función de las líneas anteriores y la herramienta de superposición que señala las coordenadas de un punto de la línea 2410. Se omite el panel de control a consecuencia de las dimensiones del mismo. El Apéndice E.1 muestra el objeto completo.

5.2. Despliegue de *gather*s (Módulo wiggle.py)

El módulo wiggle.py despliega satisfactoriamente las trazas sísmicas que se muestra en la Figura 5.2. El número de trazas a visualizar, en la dirección *inline* o *crossline*, depende del área de selección generada en el mapa base. Por lo anterior, los elementos interactivos permitieron mostrar el contenido de amplitud de las trazas 2410/2668, 2410/2669 y 2410/2670 en un plazo de 894 ms. No fue posible evitar la repetición de los títulos y leyendas ya que no se encontró una forma directa de relacionar cada *gather* desplegado con una sola herramienta luego de la ejecución del método *holoviews.NdLayout*. Sin embargo, lo anterior no limita el despliegue de las trazas.

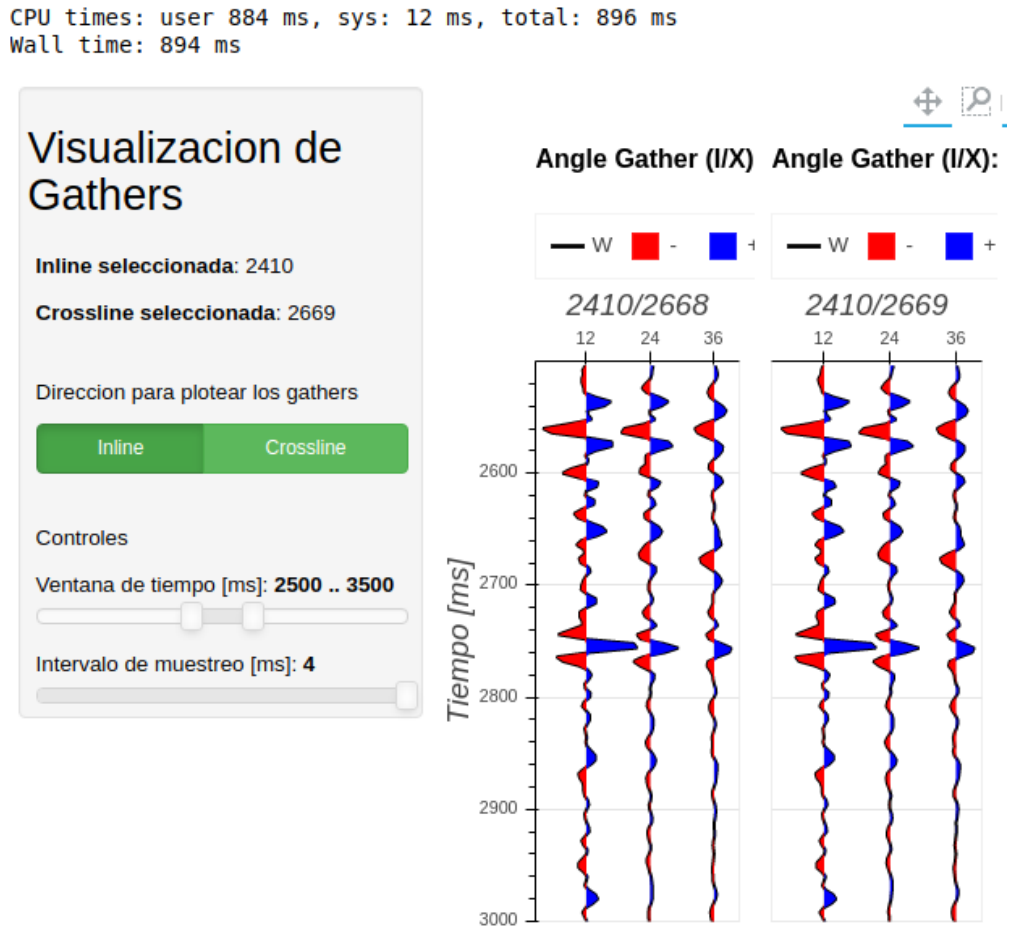

Figura 5.2. Despliegue de los *gathers* alrededor del pozo Kronos 1 (2410/2669) en dirección *Inline*. Se omite de la Figura la traza 2410/2670 debido a las dimensiones del gráfico.

5.2.1 Validación del despliegue de las trazas

La Tabla 5.1 resume la comparación entre los eventos más notables identificados a partir del despliegue de trazas del módulo wiggle.py y las interpretadas en ConocoPhillips (2012) y Liu (2018). A partir de los *gathers* desplegados por el módulo se logró identificar polaridad inversa y ondículas de fase 0, por lo que los eventos de interés se interpretaron en los picos y valles de amplitud de las trazas.

Tabla 5.1. Comparación de profundidad de eventos interpretados en la literatura con los interpretados a partir de la observación de las amplitudes en la herramienta sobre el pozo Kronos 1 (2410/2669).

Evento	P. teórica (ms)	P. Interpretada (ms)	Polaridad teórica	Polaridad interpretada	Amplitud absoluta	Observaciones
Piso oceánico	670	670		-	-97	Figura 5.3a
F. Oliver	1140	1150		+	69	Figura 5.3b
H. Tmmio	1595	1596	+	+	96	Figura 5.4a
H. Temio	1800	1800	+	+	40	Base de la Formación Oliver. Figura 5.4a
H. Tolig	1925	1925	+	+	71	Figura 5.4a
F. Prion	1995	1985		-	-113	Figura 5.4a
F. Grebe	2200-2010	2210		-	-34	Figura 5.4b
F. Johnson	2560	2560	-	-	-82	Coincide con el Horizonte Tpal. Figura 5.5a
H. Tbase	2600	2600	-	-	-44	Tope de la Formación Prudhoe. Figura 5.5a
F. Jamieson	2750	2750		+	100	Figura 5.5b
F. Montara	3050	3050		+	34	Figura 5.5b
F. Plover	3055	3058		+	50	Figura 5.5b

El primer evento de interés (Figura 5.3a) que puede identificarse desde la superficie se encuentra a una profundidad en tiempo de 670 ms. Se caracteriza por ser la tercera respuesta con mayor amplitud absoluta y se asocia con el contraste de impedancia existente entre el agua y el piso oceánico. Este evento representa un incremento en la impedancia acústica (densidad por velocidad) y exhibe valores negativos de amplitud como consecuencia del tipo de polaridad aplicada en los datos (europea o inversa). La Formación Oliver (Figura 5.3b) fue interpretada 10 ms por debajo de la profundidad teórica debido a que, en 1140 ms, se identificó un evento con menor amplitud que supone ser parte de la respuesta de la ondícula de fase 0 en 1150 ms.

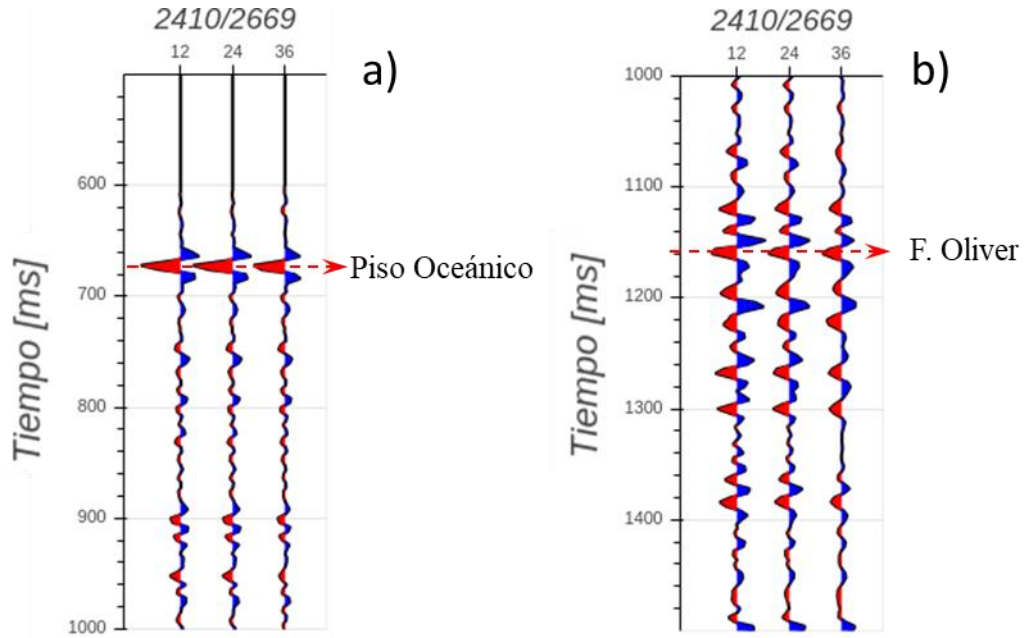

Figura 5.3. Interpretación de formaciones y horizontes en la primera ventana de tiempo comprendida entre 500 y 1500 ms. (a) Muestra la interpretación del Piso Oceánico. (b) Exhibe el tope de la Formación Oliver.

El cuarto evento con mayor amplitud absoluta (Figura 5.4a) fue identificado a una profundidad en tiempo de 1596 ms y se caracteriza por un pico en amplitud que coincide con el Horizonte Tmmio (Mioceno Medio) interpretado por Liu (2018). Dado que la respuesta está asociada a un contraste negativo de impedancia se puede relacionar dicho evento con las arenas y lutitas calcáreas de la propia Formación Oliver, razonamiento que es cónsono con las diferentes respuestas que se pueden observar de 1600 a 1800 ms. En 1800 ms se observa una respuesta tenue caracterizada por un pequeño pico en amplitud que coincide con el Horizonte Temio, interpretado por Liu (2018), el cual supone la base de la Formación Oliver y el tope de la Formación Cartier de acuerdo a la columna tecnoestratigráfica de la Figura 2.2. En 1925 ms se puede observar el sexto evento con mayor amplitud que coincide con el Horizonte Tolig (Liu, 2018) que marca la transición entre las rocas clásticas de la Formación Cartier y los carbonatos de la Formación Prion. En 1985 ms se puede observar el evento con mayor la amplitud absoluta que se caracteriza por un valle de amplitud asociado con un contraste positivo muy marcado de impedancia acústica asociado a un cambio litológico de baja velocidad y densidad a uno de mayores propiedades físicas. Por lo anterior y en concordancia con la geología, se puede asociar tal respuesta con los carbonatos de la Formación Prion, razonamiento que es congruente con interpretaciones realizadas por ConocoPhillips (2011).

ConocoPhillips (2011) sugiere que el tope de la Formación Grebe se encuentra entre 2200 y 2210 ms. Sin embargo, en la Figura 5.4b se observa una reflexión doble (*doublet*) en este intérvalo, lo que sugiere una interferencia entre las reflexiones individuales del tope y la base de posibles capas delgadas. En este caso, se tomó como tope de esta Formación aquella respuesta con la mayor amplitud (2210 ms).

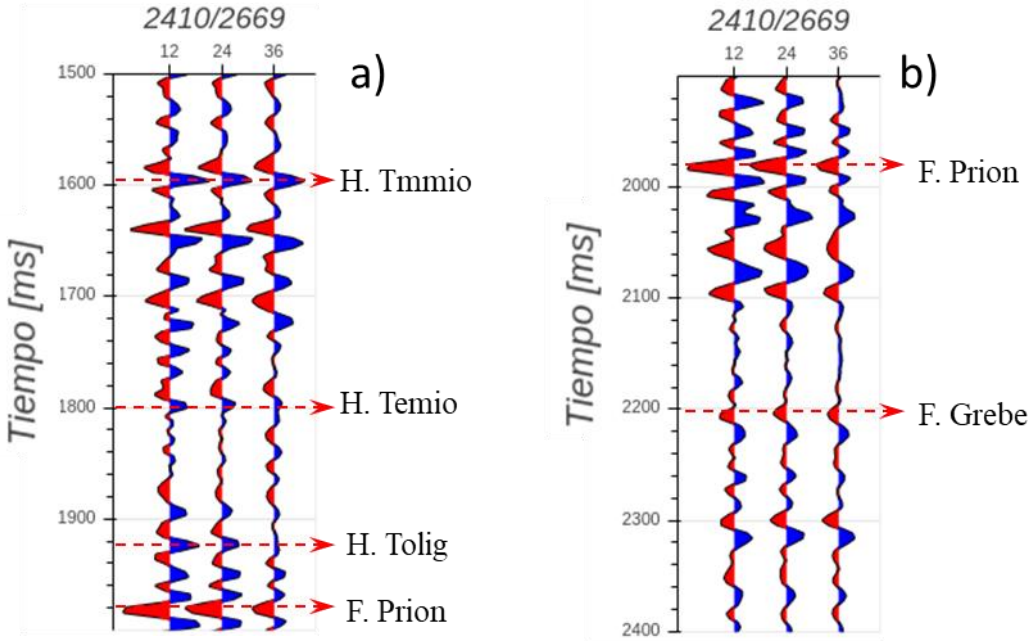

Figura 5.4. Interpretación de formaciones y horizontes en la segunda ventana de tiempo comprendida entre 1500 y 2400 ms. (a) Contiene las interpretaciones de los horizontes Tmmio, Temio, Tolig y la Formación Prion. (b) Muestra el tope de las formaciones Prion y Grebe.

El quinto evento con mayor amplitud se interpretó a una profundidad en tiempo de 2560 ms. Se distingue por un valle después de 160 ms de quietud desde 2400 ms y coincide con el tope Formación Johnson (Figura 5.5a) de acuerdo con ConocoPhillips (2011), ConocoPhillips (2012) y con el Horizonte Tpal según Liu (2018). El notable incremento de amplitud absoluta en 2560 ms se puede asociar con la interfaz entre las margas de la Formación Grebe y las lutitas calcáreas de la Formación Johnson.

A partir de las asunciones de Liu (2018) se identificó el Horizonte Tbase (Figura 5.5a) caracterizado por un valle en amplitud 50 ms por debajo del tope del Horizonte Tpal. La interpretación del Horizonte Tbase es congruente con el tope de la Formación Prudhoe identificado por ConocoPhillips (2011).No se logró identificar el tope de la Formación Woolaston ni el horizonte Kecamp mencionado por Liu (2018) debajo de la Formación Gibson, la cual fue interpretada a partir de la geología regional.

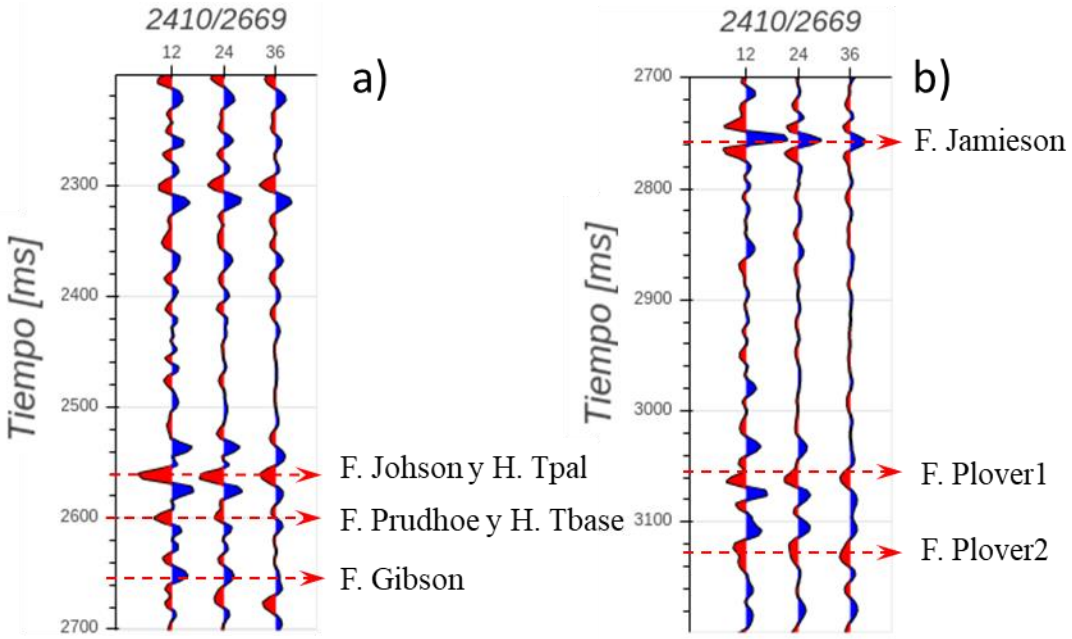

Figura 5.5. Interpretación de formaciones y horizontes en la tercera ventana de tiempo comprendida entre 2300 y 3200 ms. (a) Se muestra la interpretación de los horizontes Tbase, Kecamp y las formaciones Johnson, Prudhoe, Gibson. (b) Exhibe el tope de las formaciones Jamieson y Plover.

La segunda mayor respuesta en amplitud se puede observar a una profundidad de 2750 ms. Se caracteriza por un pico en amplitud que representa un contraste negativo de impedancia acústica e implica un cambio litológico de alta velocidad y/o densidad a uno de menores propiedades físicas. En concordancia con la geología e interpretaciones realizadas por ConocoPhillips (2011) se puede asociar este evento con la interfaz entre los carbonatos de la Formación Woolaston y las lutitas calcáreas de la Formación Jamieson (Figura 5.5b). Se puede observar la poca heterogeneidad de la Formación Jamieson al presentar respuestas de baja amplitud hasta el evento ubicado a 3058 ms atribuible a las rocas volcánicas en el tope de las Formación Plover.

No se logró identificar la Formación Montara ubicada entre las formaciones Jamieson y Plover (3000 y 3100 ms) como consecuencia de su poco espesor (8 metros). Nótese que la respuesta asociada al tope de la Formación Plover no es clara en la traza de 12° lo que sugiere que la Formación Montara interfiere con este. De las resoluciones verticales, mostradas Tabla 5.2, se verifica que la Formación Montara no presenta un espesor suficiente como para ser prospectado por la sísmica. Adicionalmente, se pudieron distinguir las dos secuencias de la Formación Plover (Figura 5.5b) descritas en la literatura como diferentes paquetes de rocas volcánicas, limolitas y areniscas.

Tabla 5.2. Resumen de la resolución vertical en la principal secuencia de interés asociada a la Formación Plover. Los valores fueron calculados a partir de las velocidades Schlumberger (2010) y el período en cada intervalo.

Intervalo (ms)	Velocidad (m/s)	Frecuencia (Hz)	Resolución vertical (m)
3000-3100	3204	22	20.6
3100-3200	3246	25	20.3
3200-3300	3290	25	17.8

En resumen, se identificaron 15 de los 17 eventos presentes en la literatura. Entre estos, la mayor diferencia en tiempo corresponde a 10 ms en los eventos asociados a las formaciones Oliver y Prion de ConocoPhillips (2011). Esta diferencia se puede adjudicar a que el tope litológico (ConocoPhillips, 2011) no coincide con el tope acústico interpretado a partir de las reflexiones.

5.3. Cálculo de los atributos AVO (Módulo avo.py)

Los datos originales del proyecto Poseidon 3D poseen sólo tres volúmenes parcialmente apilados que cubren un rango de 12° a 42°, por lo que el cálculo de los atributos de amplitud están limitados a aproximaciones lineales. Se compararon los tiempos de ejecución del cálculo de los atributos AVO utilizando un solo procesador vs el uso de todos los núcleos disponibles. La Figura 5.6 muestra los tiempos asociados a la ejecución con un solo núcleo (Figura 5.6a) y en paralelo (Figura 5.6b), donde *Wall time* implica el tiempo total en la ejecución de la función y CPU *time* hace referencia al tiempo empleado por el equipo para ejecutar un proceso. Se observa un contraste de 1 min con 01 s respecto al tiempo total de ejecución y una diferencia notable en la duración de los procesos asociados. El primero se relaciona con un aumento en el rendimiento de 66% en la ejecución bajo las mismas condiciones; mientras que el segundo, implica mayor rendimiento para volúmenes con mayor densidad de datos debido a que se crea un proceso por traza (cálculo).

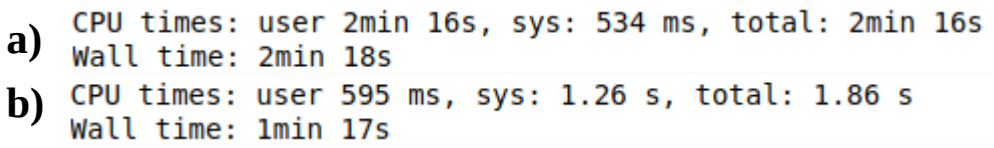

Figura 5.6. Tiempos de ejecución de la función *calculo_guardado_atributos_AVO* empleada de forma serial (a) y en paralelo (b).

5.4. Visualización de atributos AVO

Los elementos interactivos del módulo avo.py permitieron cartografiar satisfactoriamente el gráfico cruzado y el mapa base de los atributos AVO. La validación del módulo se realizó al utilizar una ventana de 300 ms en la Formación Plover (pozo Kronos 1), cuya ejecución mínima se realizó en 2.8 s (Apéndice E.2). La Figura 5.7 muestra sólo el gráfico cruzado de intercepto vs gradiente a lo largo de una ventana de tiempo situada entre 3000 y 3300 ms. Nótese que la mayor parte de los puntos se encuentran alineados y concentrados en dirección NO-SE (cuadrantes IV-II); cuya orientación es compatible con la tendencia de porosidad o línea de lutita. Debido a que los datos poseen polaridad europea, se seleccionaron las respuestas alejadas de esta tendencia hacia arriba y a la derecha (cuadrante I). La escala de color permite identificar, en los puntos seleccionados, un error estándar entre 0 y 10 unidades de amplitud, lo cual se traduce en una estimación de alta confianza de los atributos AVO.

En la Figura 5.8 se muestran, en un mapa base, las respuestas favorables seleccionadas en el gráfico cruzado. La herramienta de superposición permitió la identificación de la profundidad en tiempo de estas respuestas (3248 y 3252 ms). Tras la conversión tiempo-profundidad se logró correlacionar estas soluciones con las arenas gasíferas interpretadas en la literatura (ConocoPhillips, 2011) a una profundidad de 4996.92 m, que subyacen a rocas volcánicas y suprayacen a lutitas.

En la figura 5.9, se correlacionan las respuestas favorables ubicadas en el *gather* entre 3248 y 3252 ms (Figura 5.9a) y las arenas gasíferas en 5000 m (Figura 5.9b). Dado que los espesores de los paquetes de interés son menores a la resolución vertical entre 3200 y 3300 ms, los topes en 3244 y 3258 ms se superponen con la respuesta obtenida por el gráfico cruzado. Empleando los registros petrofísicos, se calcularon las impedancias y los coeficientes de reflexión de los dos primeros paquetes de arenas gasíferas (Tabla 5.3), pudiendo afirmar que la respuesta observada es la más prominente.

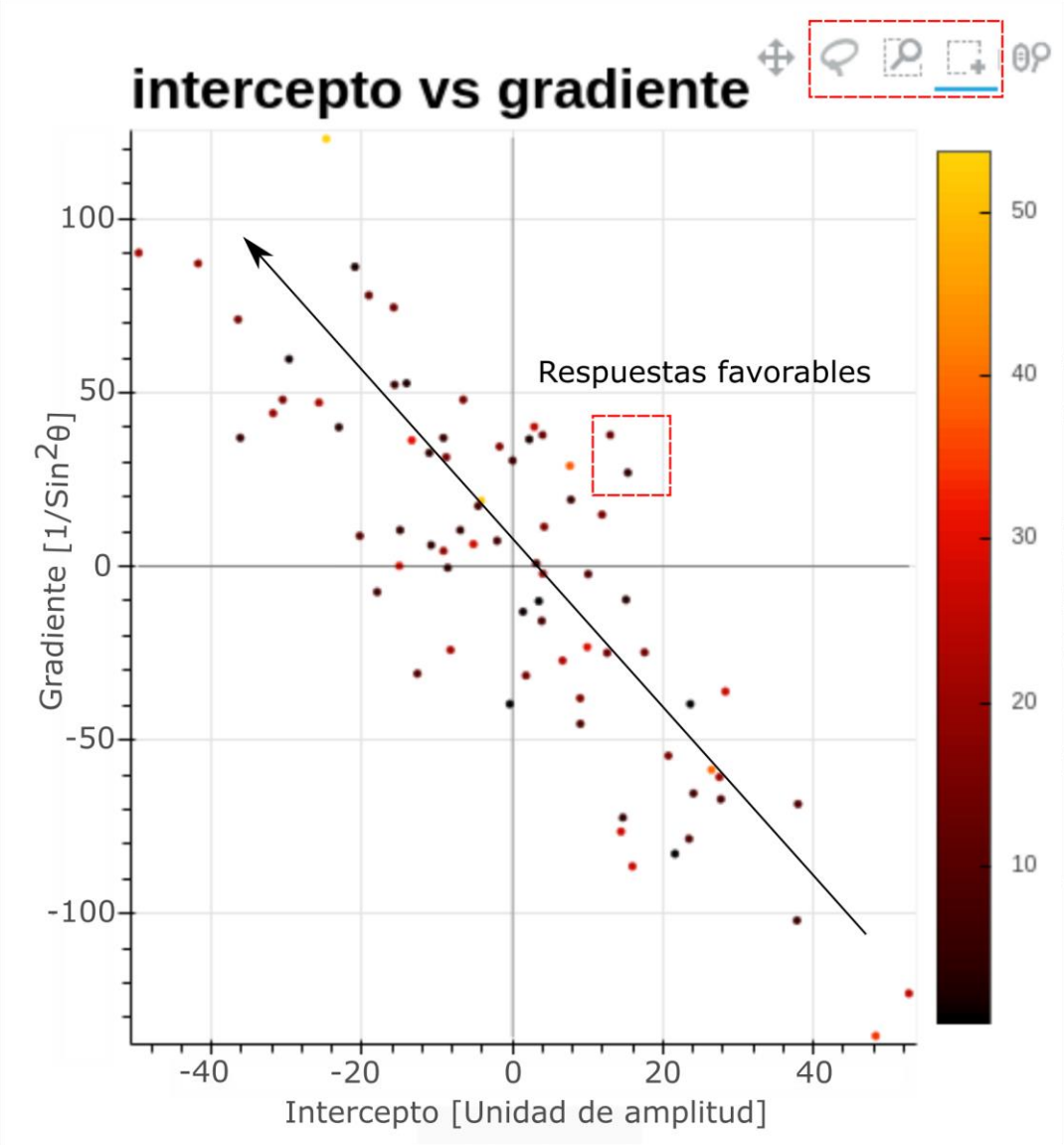

Figura 5.7. Gráfico cruzado de intercepto vs gradiente utilizando el error estándar de la aproximación lineal como escala de color. Se muestran las respuestas entre 3000 y 3300 ms en el pozo Kronos 1. Se resalta la orientación de la línea de lutita o tendencia de porosidad en negro y los puntos anómalos en rojo, que se caracterizan por la tendencia arriba y hacia la derecha como consecuencia de la polaridad inversa del dato. El la parte superior derecha se marcan en rojo las herramientas para la selección. No se muestra en panel de control debido a las dimensiones del dato (Apéndice E.2).

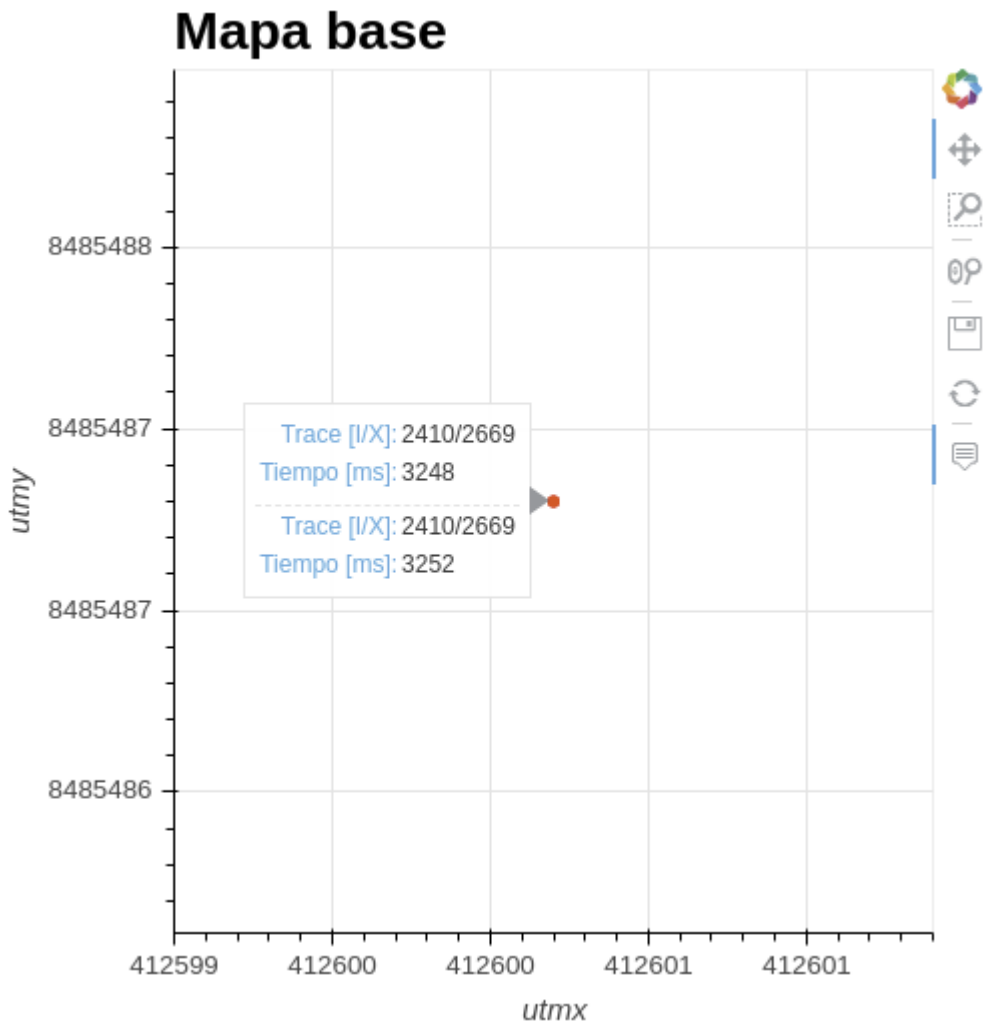

Figura 5.8. Mapa base de los puntos seleccionados en el gráfico cruzado generado por la función *visualizacion_avo* en torno al pozo Kronos 1. Se muestran las coordenadas de los puntos seleccionados utilizando la herramienta de superposición.

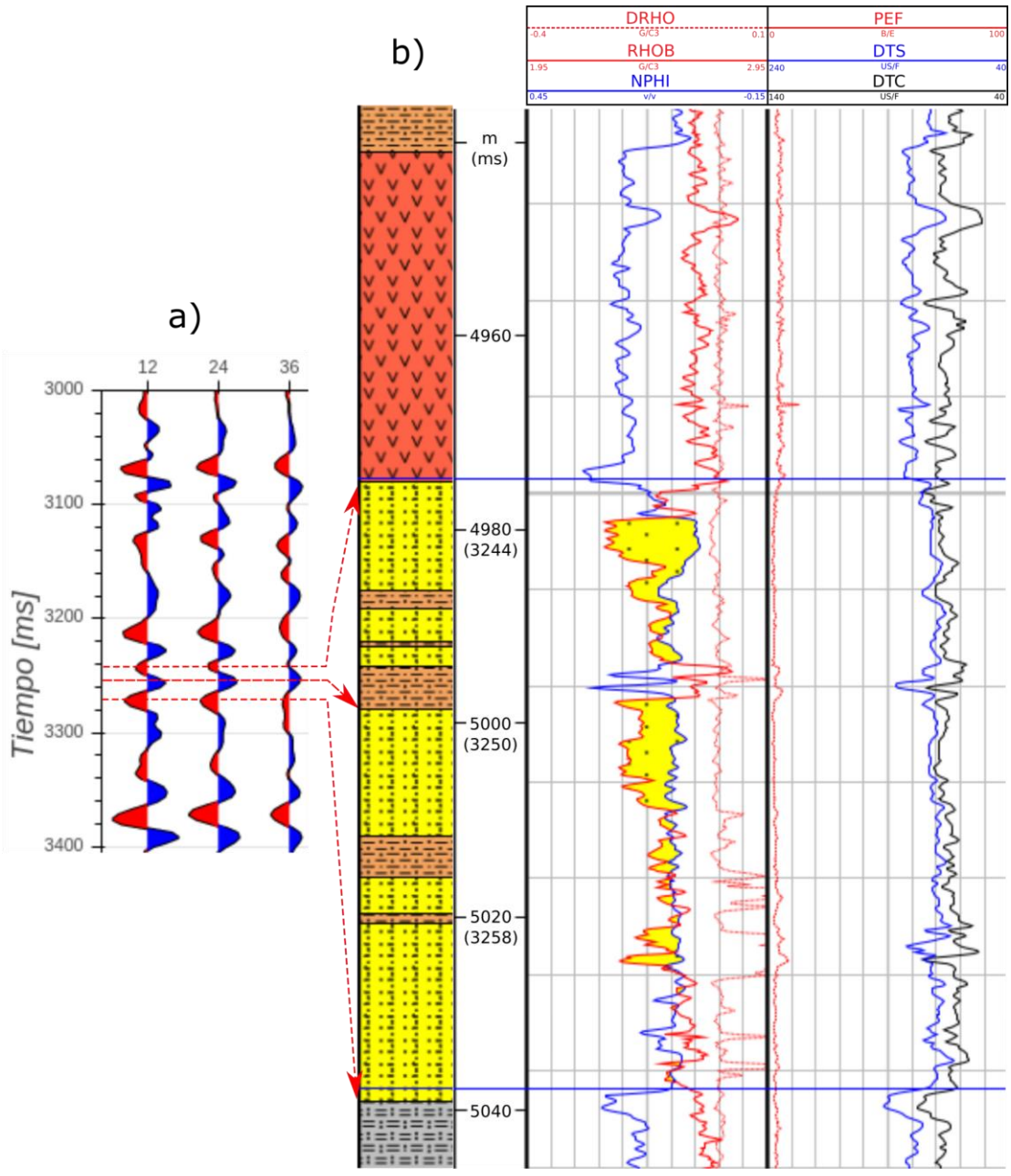

Figura 5.9. Comparación de la respuesta AVO en el *gather* asociado al pozo Kronos 1 y el registro petrofísico del intervalo asociado a la presencia de gas. Se muestra el registro densidad y sónico en la base de la Formación Plover (Modificado de ConocoPhillips, 2011).

Tabla 5.3. Valores de impedancias acústicas correspondientes a las dos primeras arenas gasíferas calculadas empleando la ecuación 3.3.

Tope	Velocidad (m/s)	Densidad (kg/m^3)	Impedancia Z (Pa $\frac{s}{m}$)	C. de Reflexión (R_{pp})
Arena sin porosidad	5047	2650	13374550	-0,112
Arena gasífera 1 (4980 m)	4686	2280	10684080	
Lutita	5468	2650	**14490200**	**-0.147**
Arena gasífera 2 (5000 m)	4686	2300	**10777800**	

5.5. Flujo de trabajo

El flujo de trabajo asociado a esta herramienta se encuentra en la dirección https://github.com/lsdR94/Prestack_characterization_tool, dentro la carpeta *notebooks*, bajo el nombre de Prestack_Characterization_Tool[12-10-2019].ipynb (*Jupyter-notebook*). Este repositorio también contiene los módulos programados, los subvolumenes utilizados del Proyecto Poseidon 3D, guías para instalar el ambiente de programación y los flujos de trabajo asociados a cada módulo.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

En función de los resultados obtenidos, se pudieron esbozar las siguientes conclusiones:

a) Este trabajo comprueba que es posible el diseño e implementación de una herramienta en la inspección, visualización y caracterización de los datos sísmicos por medio de la técnica AVO utilizando *Ubuntu* versión 16.04 (*Xenial Xerus*) como sistema operativo, *Python* (v3.7) como lenguaje de programación, *Jupyter notebook* como intérprete y las bibliotecas *SegyIO* (v1.6), *Pandas* (v0.24.2), *Scipy* (v1.2.1), *Holoviews* (v1.12.1), *Panel* (v0.5.1), *Multiprocesing* (v16.6).

b) La recopilación e implemetación de elementos y herramientas interactivas, permitió evitar soluciones estáticas al generar gráficos interactivos que facilitan la manipulación del dato sísmico y hacen de la interpretación una actividad dinámica.

c) La codificación de los módulos permite estandarizar flujos de trabajo que combinan elementos de programación modernos y facilitan la inspección de la geometría y visualización de amplitudes, además del cálculo y despliegue de los atributos AVO.

d) Los flujos de trabajo estandarizados en esta investigación, no son exclusivos de equipos de alto rendimiento. Estos se lograron completar en un tiempo menor a 2 minutos para un set de datos 11x11 trazas en un equipo de 2 Gb de RAM, 32 bits.

e) La herramienta permitió verificar la interpretación de arenas gasíferas presentes en el pozo Kronos 1, localizadas a una profundidad de 5000 m (3250 ms) mediante el gráfico cruzado de intercepto vs gradiente.

Logrados parte de los objetivos planteados, se recomienda:

a) Comprobar la reproducibilidad de los resultados utilizando aplicaciones comerciales.

b) Incorporar los métodos de la biblioteca de *GeoViews* de *Pyviz* para aumentar la calidad en el posicionamiento geográfico.

c) Evaluar la reestructuración del código utilizando clases y no funciones; en tal sentido, considerar el cartografiado de los *gathers* como un objeto para vincular las herramientas interactivas que ofrece la función.

d) Adicionar una rutina de programación que permita convertir datos en el dominio del *offset* al dominio de ángulos. Así mismo, incluir módulos de cálculo de parámetros elásticos específicos de reservorios y otras técnicas que complementen el análisis de AVO como la sustitución de fluidos.

e) Probar el funcionamiento de la aplicación utilizando el volumen completo de datos pre-apilados.

f) Incluir métodos interactivos y compatibles que solucionen el problema de sobrecartografiado u *overplotting* al evaluar un gran grupo de trazas (millones de muestras) como los que dispone la biblioteca *Datashader* de *Pyviz*.

g) Implementar rutinas que permitan realizar la conversión profundidad- tiempo.

REFERENCIAS

Abott, S., Lech, M., Romeyn, R. y Rollet, N. (2016). A regional assessment of CO2 storage potential in the Browse Basin: Results of a study undertaken as part of the National CO2 Infrastructure Plan. Recuperado de: https://www.researchgate.net/figure/Structural-elements-of-the-Browse-Basin-showing-Palaeozoic-to-Early-Cretaceous-faults-in_fig17_307634080

Aki, K. y Richards, G. (1980). Quantitative Seismology. San Francisco: Freeman.

Almutlaq, M. y Margrave, G. (2010). Tutorial AVO Inversion. Documento en línea. Disponible en: https://www.crewes.org/ForOurSponsors/ResearchReports/2010/CRR201002.pdf . Revisado el 03 de noviembre de 2018.

Amundsen, L. y Ikelle, L. (2005). Introduction to petroleum seismology. Society of Exploration Geophysics, Tulsa, Oklahoma, USA. Investigations in Geophysics (12), 1-26, 111 pp. Revisado el 03 de noviembre de 2018

AGSO Browse Basin Proyect Team. (1997). Browse Basin High Resolution Study, Interpretation Report. Record 1997/38. Australian Geological Survey Organisation. Documento en línea. Disponible en: https://d28rz98at9flks.cloudfront.net/23689/Rec1997_038.pdf. Revisado el 15 de enero de 2019. 1-29 p.

Avseth, P., Mukerji, T. y Mavko, G. (2005). Quantitative Seismic Interpretation: Applying Rock Physics to Reduce Interpretation Risk. Cambridge, New York, Melbourne. 168-203 pp.

Backus, M. y Castagna, J. (1993). Offset-dependent-Reflectivity-Theory and practice of AVO analysis. Investigations in Geophysics Series No. 8 (3). Tulsa, USA. 1-5 pp.

Bacon, M. y Simm, R. (2014). Seismic Amplitude: An Interpreter's Handbook. Cambridge University Press. Reino Unido. 1-97, 116-117 pp.

Bianco, E. y Hall, M. (2015). Seisplot. Github. Disponible en: https://github.com/agile-geoscience/seisplot

Bianco, E., Bougher, B., Hall, Matt., Amato del Monte, A., Hamlyn, W., y Ross-Ross, S. (2013).Geophysics library with various helpful functions. Github. Disponible en: https://github.com/agile-geoscience/bruges

Blevin, J., Boreham, G., Cathro, D., Loutit, T., Romine, K., Sayers, J., Struckmeyer, H. y Totterdell, J. (1998). Tecnostratigraphic Framework and Petroleum Systems of the Browse Basin, North West Shelf. En PURCELL, P.G. & R.R. (Eds), 1998, The Sedimentary Basins of Western Australia 2: Proceedings of Petroleum Exploration Society of Australia Symposium, Perth, WA, 1998.

Blevin, J., Baxter, K., Cathro, D., Loutit, T., Romine, K., Sayers, J., Struckmeyer, H. y Totterdell, J. (1998). Structural Evolution of the Browse Basin, North West Shelf; New Concepts from Deep-seismic Data. En PURCELL, P.G. & R.R. (Eds), 1998, The Sedimentary Basins of Western Australia 2: Proceedings of Petroleum Exploration Society of Australia Symposium, Perth, WA, 1998.

Castagna, J. y Swan, H. (1997). Principles of AVO crossplotting. The Leading Edge No. 16. 337-342 pp.

Cavada, J. (2000). Guía de prospección sísmica por refracción. Documento en línea. Disponible en: http://www.geocities.ws/geofisicaucv/Archivos/refracc4.pdf

Cavada, J. (2018). Python. 2-21pp.

Chopra, S y Castagna, J. (2014). AVO: Investigation in Geophysics, 16. Society of Exploration Geophysicist. Tulsa, Oklahoma. 1-8, 45-60 pp.

DevCode. (2019). Funciones en Python. Recuperado de: https://devcode.la/tutoriales/funciones-en-python/

dGB Earth Science. (2019). Software. Recuperado de: https://dgbes.com/index.php/software#commercial

ConocoPhillips. (2011). Well Completion Report. Volumen 2: Interpretative Data. Documento en línea. Disponible en: https://drive.google.com/drive/folders/0B7brcf-eGK8CRUhfRW9rSG91bW8?usp=sharing

ConocoPhillips. (2012). 2009 Poseidon 3D Marine Surface Seismic Survey Interpretation Report: Browse Basin, Western Australia. Documento en línea. Disponible en: http://static.dgbes.com/images/PDF/Processing_Report_Poseidon3D.pdf. Revisado el 23 de enero de 2019.

Free Software Foundation. (2019). The Free Software Foundation (FSF) is a nonprofit with a worldwide mission to promote computer user freedom. We defend the rights of all software users. Recuperado de: https://www.fsf.org/

Geoscience Australia. (2014). Offshore Petroleum Exploration Acreage Release. Australia 2014. Documento en línea. Disponible en: https://archive-petroleumacreage.industry.slicedtech.com.au/files/files/2014/documents/regional-geology/Regional_Geology-Browse.pdf. Revisado el 28 de enero de 2018.

Halliburton (2007). Efficient Management of Prestack Seismic Data Brings More Successful Exploration Efforts. Landmark Graphics Corporation. Documento en línea. Disponible en: https://www.landmark.solutions/Portals/0/LMSDocs/Whitepapers/2007-08_efficient-management-of-prestack-seismic-data-white-paper.pdf.

Holoviews. (2019). Holoviews. Recuperado de: http://holoviews.org/

Kvalsvik, J. (2014). Segyio. Github. Disponible en: https://github.com/equinor/segyio#tutorial.

Mittag, J. (2010). Difference between a module, library and a framework. Recuperado de: https://stackoverflow.com/questions/4099975/difference-between-a-module-library-and-a-framework

Numpy. (2019). Numpy. Recuperado de: https://numpy.org/

Open Source Iniciative. (2019). The Open Source Definition (Annotated). Recuperado de: https://web.archive.org/web/20060924132143/http://www.opensource.org/licenses/index.php

Ostrander, W. (2006). Memoirs of successful geophysicists: CSEG Recorder, 31, no. 6, 38–41 pp. Documento en línea. Disponible en: http://74.3.176.63/publications/recorder/2006/06jun/jun2006-memoirs.pdf

Python Software Foundation. (2017). Módulos — Tutorial de Python 3.6.3. Recuperado de: http://docs.python.org.ar/tutorial/3/modules.html

Python Software Foundation. (2017). 10. Classes – Tutorial de Python 3.6.3. Recuperado de: http://docs.python.org.ar/tutorial/3/classes.html

Python. (2019). multiprocessing — Process-based “threading” interface. Recuperado de: https://docs.python.org/2/library/multiprocessing.html

Pandas. (2019). Python Data Analysis Library. Recuperado de: https://pandas.pydata.org/

Panel. (2019). A high-level app and dashboarding solution for Python. Recuperado de: https://panel.pyviz.org/index.html#

Ross, C y Kinman, D. (1995). Nonbright-spot AVO: two examples. Geophysics No. 60. 1398-1408 pp.

Russell, B. (2010). Making sense of all that AVO and inversión stuff!. Documento en línea. Disponible en: http://www.agl.uh.edu/pdf/russell-2010.pdf

Russell, B. (2014). Prestack seismic amplitude analysis: An integrated overview. Documento en línea. Disponible en: https://www.cgg.com/technicalDocuments/cggv_0000020953.pdf. Revisado el 04 de noviembre de 2018.

Rutherford, S y Williams, R. (1989). Amplitude versus offset variations in gas sands. Geophysics N0. 54. 680-688 pp.

Schlumberger. (2010). A3 VELOCITY REPORT, Browse Basin, Well: Kronos 1. Documento en línea. Disponible en: https://drive.google.com/drive/folders/1w4yFrgsJxOxAUMLOm9TBWByGgWkiyqtx

Schlumberger Oilfield Glossary. (2019). Crossline. Recuperado de: https://www.glossary.oilfield.slb.com/es/Terms/c/crossline.aspx?p=1

Schlumberger Oilfield Glossary. (2019). Inline. Recuperado de: https://www.glossary.oilfield.slb.com/en/Terms/i/in_line.aspx

Schlumberger Oilfield Glossary. (2019). Stacking. Recuperado de: https://www.glossary.oilfield.slb.com/en/Terms/s/stack.aspx#

SEG Wiki. (2019). Angle stacks. Recuperado de: https://wiki.seg.org/wiki/Angle_stacks

SEG Wiki. (2019). AVO intercept and gradient. Recuperado de: https://wiki.seg.org/wiki/AVO_intercept_and_gradient

SEG Wiki. (2019). Dictionary:Amplitude variation with angle/offset (AVA/AVO). Recuperado de: https://wiki.seg.org/wiki/Dictionary:Amplitude_variation_with_angle/offset_(AVA/AVO)

SEG Wiki. (2019). Direct hydrocarbon indicators. Recuperado de: https://wiki.seg.org/wiki/Direct_hydrocarbon_indicators

SEG Technical Standards Committee. (2017). SEG-Y_r2.0: SEG-Y revision 2.0 Data Exchange format. Documento en línea. Disponible en: https://seg.org/Portals/0/SEG/News%20and%20Resources/Technical%20Standards/seg_y_rev2_0-mar2017.pdf

Shuey, R. (1985). A simplification of the Zoeppritz equations. Geophysics No.50. 609-614 pp.

Scipy. (2019). Scipy. Recuperado de: https://www.scipy.org

SubSurfWiki. (2012). Gather. Recuperado de: http://www.subsurfwiki.org/wiki/Gather

Telford, W., Geldart, L., Sheriff, E., Keys, D. (1990). Applied Geophysics. Cambridge University Press – 2da ed. Reino Unido. 136-137 pp. Documento en línea. Disponible en: https://kobita1234.files.wordpress.com/2016/12/telford-geldart-sheriff-applied-geophysics.pdf

Terranubis (2019). Data Room: Project NW Shelf Australia Poseidon 3D. Datos en línea. Disponible en: https://terranubis.com/datainfo/NW-Shelf-Australia-Poseidon-3D. Revisado el: 29 de diciembre de 2018.

Yilmaz, O. (2001). *Seismic Data Analysis*. Investigation in Geophysics No. 10 (1). Tulsa, USA, 15 p. Documento en línea. Disponible en: http://library.seg.org/

Zoeppritz, K. (1919). Erdbebenwellen VII, VII B, Über Reflexion und Durchgang seismischer Wellen durch Unstetigkeitsflächen [On the reflection and transmission of seismic waves at surfaces of discontinuity]: Nachrichten von der Königlichen Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-physikalische Klasse, 66–84.

APÉNDICE

Apéndice A: Especificaciones del equipo

- Laptop: Vit P2400-01 (P2400-01).
- Arquitectura: 32 bits.
- Procesador: INTEL Core i3 2370M 2.40 GHz (3M Cache).
- Memoria: 2 GB DDR3 de RAM.
- Gráfica: Intel HD GRAPHICS FAMILY 3000.
- Discoduro: 320 GB, 5400 RPM.
- Sistema operativo: Ubuntu 16.04 (Xenial Xerus).

Apéndice B: Bibliotecas y módulos

B.1. Python (3.7)

Cavada (2018), define Python como un lenguaje de programación de alto nivel, orientado a objetos, diseñado para ser fácil de programar y fácil de leer; con un prósito general y no específico como Octave, Matlab, R entre otros.

B.2. SegyIO (1.6)

Kvalsvik (2014), caracteriza SegyIO como una pequeña biblioteca en C, con licencia LGPL, para una fácil interacción con los datos sísmicos en formato SEG-Y y Seismic Unix, con enlaces de lenguaje para Python y Matlab. SegyIO, es un intento de crear una biblioteca fácil de usar, integrable y orientada a la comunidad para aplicaciones sísmicas.

B.3. Pandas (0.24.2)

Es una biblioteca de código abierto bajo la licencia BSD, que proporciona estructuras y herramientas para el análisis de datos en el lenguaje de programación Python (Pandas, 2019)

B.4. Numpy (1.15.4)

Es el paquete fundamental para la computación científica en Python, bajo la licencia BSD. Contiene funciones para el desarrollo de operaciones matriciales, algebra lineal, transformadas y contenedores multidimensionales genéricos (Numpy, 2019).

B.5. Scipy (1.2.1)

Es un ecosistema de software de código abierto basado en Python para el desarrollo y aplicación de matemática, ciencias e ingeniería. Scipy incluye a Pandas y al paquete Numpy (Scipy, 2019).

B.6. Holoviews (1.12.1)

Es una biblioteca de Python a código abierto (licencia BSD), diseñada para facilitar el análisis y visualización de datos empleando los motores y código fuente de las plataformas de visualización *Bokeh*, *Matplotlib y Plotly* (Holoviews, 2019).

B.7. Panel (0.5.1)

Panel es una biblioteca de Python a código abierto (licencia BSD), que permite crear aplicaciones web interactivas personalizadas y paneles de utilidad conectando widgets definidos por el usuario a gráficos, imágenes, tablas o texto (Panel, 2019).

B.8. Multiprocesing (16.6)

Es un paquete que permite la gerencia de procesos y subprocesos relacionados con la ejecución de instrucciones (Python, 2019).

Apéndice C: Encabezados sísmicos

La Tabla C.1 muestra los bytes asociados a cada encabezado presente en un archivo SEG-Y con formato estándar. Para mayor información, consultar SEG *Technical Standards Committee*, (2017).

Tabla C.1. Distribución de encabezados y bytes asociados a las trazas de un archivo SEG-Y con formato estándar.

Bytes	Encabezado
1-4	TRACE_SEQUENCE_LINE
5-8	TRACE_SEQUENCE_FILE
9-12	FieldRecord
13-16	TraceNumber
17-20	EnergySourcePoint
22-25	CDP
26-29	CDP_TRACE
29-30	TraceIdentificationCode
31-32	NSummedTraces
33-24	NStackedTraces
35-36	DataUse
37-40	offset
41-44	ReceiverGroupElevation
45-48	SourceSurfaceElevation
49-52	SourceDepth
53-56	ReceiverDatumElevation
57-60	SourceDatumElevation
61-64	SourceWaterDepth

Bytes	Encabezado
65-68	GroupWaterDepth
69-70	ElevationScalar
71-72	SourceGroupScalar
73-76	SourceX
77-80	SourceY
81-84	GroupX
85-88	GroupY
89-90	CoordinateUnits
91-92	WeatheringVelocity
93-94	SubWeatheringVelocity
95-96	SourceUpholeTime
97-98	GroupUpholeTime
99-100	SourceStaticCorrection
101-102	GroupStaticCorrection
103-104	TotalStaticApplied
105-106	LagTimeA
107-108	LagTimeB
109-110	DelayRecordingTime
111-112	MuteTimeStart
113-114	MuteTimeEND

Bytes	Encabezado
115-116	TRACE_SAMPLE_COUNT
117-118	TRACE_SAMPLE_INTERVAL
119-120	GainType
121-122	InstrumentGainConstant
123-124	InstrumentInitialGain
125-126	Correlated
127-128	SweepFrequencyStart
129-130	SweepFrequencyEnd
131-132	SweepLength
133-134	SweepType
135-136	SweepTraceTaperLengthStart
137-138	SweepTraceTaperLengthEnd
139-140	TaperType
141-142	AliasFilterFrequency
143-144	AliasFilterSlope
145-146	NotchFilterFrequency
147-148	NotchFilterSlope
149-150	LowCutFrequency
151-152	HighCutFrequency
153-154	LowCutSlope

Bytes	Encabezado
155-156	HighCutSlope
157-158	YearDataRecorded
159-160	DayOfYear
161-162	HourOfDay
163-164	MinuteOfHour
165-166	SecondOfMinute
167-168	TimeBaseCode
169-170	TraceWeightingFactor
171-172	GeophoneGroupNumberRoll1
173-174	GeophoneGroupNumberFirstTraceOrigField
175-176	GeophoneGroupNumberLastTraceOrigField
177-178	GapSize
179-180	OverTravel
181-184	CDP_X
185-188	CDP_Y
189-192	INLINE_3D
193-196	CROSSLINE_3D
197-200	ShotPoint
201-202	ShotPointScalar
203-204	TraceValueMeasurementUnit

Bytes	Encabezado
205-208	TransductionConstantMantissa
209-210	TransductionConstantPower
211-212	TransductionUnit
213-214	TraceIdentifier
215-216	ScalarTraceHeader
217-218	SourceType
219-222	SourceEnergyDirectionMantissa
223-224	SourceEnergyDirectionExponent
225-228	SourceMeasurementMantissa
229-230	SourceMeasurementExponent
231-232	SourceMeasurementUnit
233-236	UnassignedInt1
237-240	UnassignedInt2

Apéndice D: Funciones de la herramienta de caracterización

D.1. Módulo de utilidad

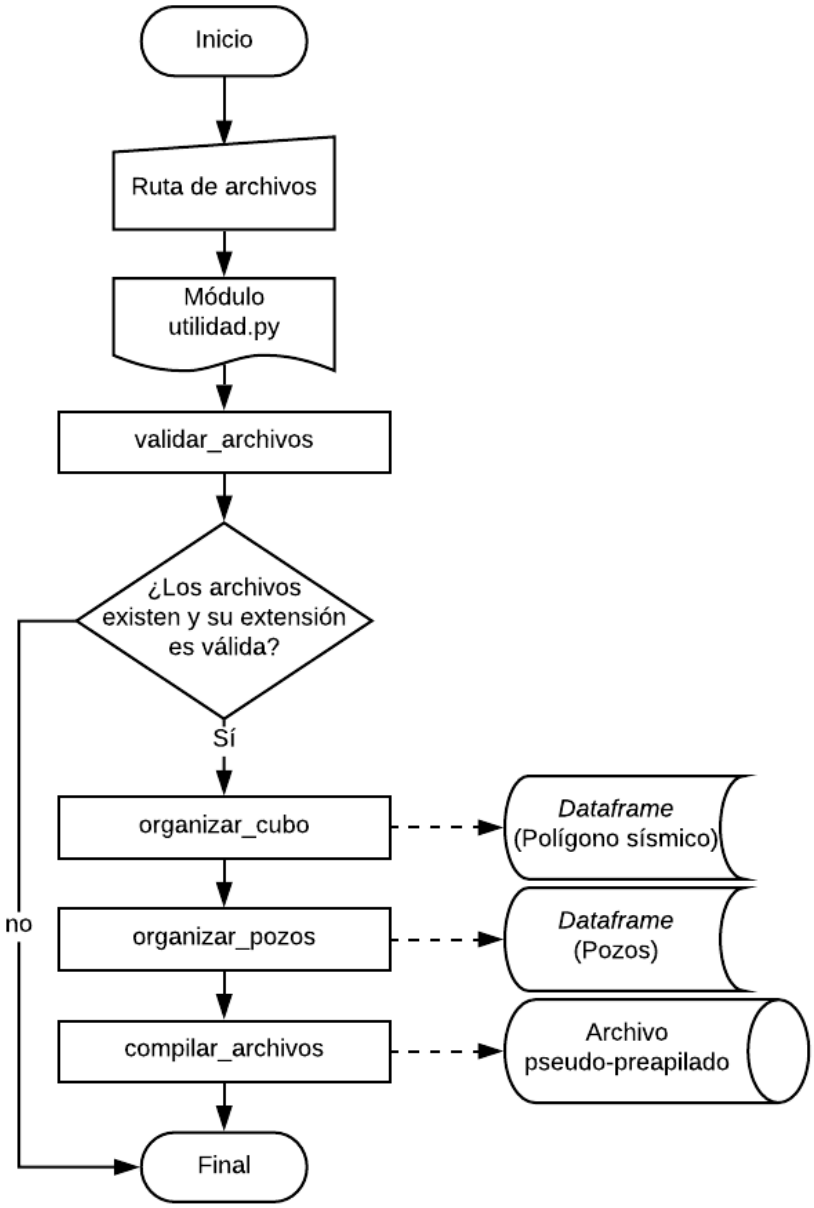

Figura D.1. Diagrama de flujo del módulo utilidad.py

D.1.1. validar_archivos

```
def validar_archivos(ruta_archivo):

    # Primera validación: el archivo existe?
    if os.path.isfile(ruta_archivo):
        file, ext = os.path.splitext(ruta_archivo)

        # Segunda validación: la extensión del archivo es txt, segy o sgy?
        if ext.lower() == (".txt") or ext.lower() == (".sgy") or ext.lower() == (".segy") :

            # Tercera validación: el aarchivo SEG-Y es estandard?
            if ext.lower() == (".sgy") or ext.lower() == (".segy"):
                with segyio.open(ruta_archivo,'r') as segyfile:
                    segyio_header = list(segyio.tracefield.keys.keys())
                    file_header = segyfile.header[0].keys()

                    # Comparando los headers de la traza dentro del archivo con los de segyIO
                    for i in range(0,len(segyfile.header[0].keys()),1):
                        if str(segyio_header[i]) == str(file_header[i]):

                            # Si la lista de headers es la misma, return True
                            return(True)

                        # Si el SEG-Y no es estandar, return texto: SEG-Y no es estandar
                        else:
                            return(f"El SEG-Y suministrado no cumple los requerimientos necesarios")

            # Si las primeras validaciones son verdaderas y no es SEG-y, return true
            else:
                return(True)

        # Caso contrario, return texto: la extensio del archivo no es valida
        else:
            return(f"La extension del archivo no es valida.")

    # Si la primera validacion no es verdadera, regresar: el archivo no existe
    else:
        return(f"El archivo suministrado no existe.")
```

D.1.2. compilar_archivos

```
def compilar_archivos(lista_gathers, ruta_archivo_compilado, lista_de_angulos):

    # Validar los archivos
    for file in lista_gathers:
        if validar_archivos(file) == True:
            print(f"{file} ha sido validado")
        else:
            return(f"Validacion fallida: {validar_archivos(file)}")

    # Crear el arreglo de offset
    offsts = np.array(lista_de_angulos)

    # Inicializar los archivos
    with segyio.open(lista_gathers[0]) as f:

        # Funcion Spect para construir en nuevo SEG-Y
        spec = segyio.spec()
        spec.sorting = f.sorting
        spec.format = 1
        spec.samples = f.samples
        spec.ilines = f.ilines
        spec.xlines = f.xlines
        spec.offsets = offsts

        # Inicializar el nuevo archivo
        with segyio.create(ruta_archivo_compilado, spec) as s:

            # Indice para compilar trazas y encabezados
            merge_index = 0
            # Indice para los encabezados y trazas de los archivos originales
            stack_index = 0
            # Ciclo for para establecer los parametros necesarios para indexar las trazas
            for il in spec.ilines:
                for xl in spec.xlines:
                    for offset in spec.offsets:
                        # Asignacion de headers
                        s.header[merge_index] = {segyio.su.tracl  : f.header[stack_index][1],
                                                 segyio.su.tracr  : f.header[stack_index][5],
                                                 segyio.su.fldr   : f.header[stack_index][9],
                                                 segyio.su.cdp    : f.header[stack_index][21],
                                                 segyio.su.cdpt   : f.header[stack_index][25],
                                                 segyio.su.offset : offset, # 37
                                                 segyio.su.scalco : f.header[stack_index][71],
                                                 segyio.su.ns     : f.header[stack_index][115],
                                                 segyio.su.dt     : f.header[stack_index][117],
                                                 segyio.su.cdpx   : f.header[stack_index][181],
                                                 segyio.su.cdpy   : f.header[stack_index][185],
                                                 segyio.su.iline  : il,  # 189
                                                 segyio.su.xline  : xl}  # 193

                        # Extraer el indice de angulo para corresponder el indice del archivo en el ciclo
                        file_index = lista_de_angulos.index(offset)
```

```
                # Abrir el archivo asociado al angulo en el ciclo for ()offset
                with segyio.open(lista_gathers[file_index], "r") as stack:

                    # Copiar las amplitudes de cada archivo al nuevo SEG-Y
                    s.trace[merge_index] = stack.trace[stack_index]

                merge_index += 1
            stack_index += 1

    return (f"Compilado exitoso. Ruta del nuevo SEG-Y: {ruta_archivo_compilado}")
```

D.1.3. organizar_pozos

```
def organizar_pozos(ruta_pozos, dataframe_sismico):

    if validar_archivos(ruta_pozos) == True:

        dataframe_pozos = pd.read_csv(ruta_pozos,
                    sep=" ",
                    header = None,
                    names= ["name","cdp_iline","cdp_xline","utmx","utmy", "depth"])

        dataframe_pozos["index"] = dataframe_pozos["name"]
        dataframe_pozos.set_index("index", inplace = True)

        # Ajustando el dataframe en funcion del levantamiento sismico
        dataframe_pozos = dataframe_pozos[(dataframe_pozos.cdp_iline >= dataframe_sismico.iline.min
()) &
                                (dataframe_pozos.cdp_iline <= dataframe_sismico.iline.max()) &
                                (dataframe_pozos.cdp_xline >= dataframe_sismico.xline.min()) &
                                (dataframe_pozos.cdp_xline <= dataframe_sismico.xline.max())]
        return(dataframe_pozos)

    else:
        print(validar_archivos(ruta_pozos))
        return()
```

D.1.4. organizar_cubo

```
def organizar_cubo(ruta_archivo):

    if validar_archivos(ruta_archivo) == True:
        df = pd.DataFrame([])
        corners = []
        with segyio.open(ruta_archivo, "r") as segy:
            df = pd.DataFrame([])

            # Arreglo para extraer las coordenadas minimas y maximas
            utmx = segy.attributes(segyio.TraceField.CDP_X)[:]
            utmy = segy.attributes(segyio.TraceField.CDP_Y)[:]

            # Extraer los puntos
            corners += [utmx.argmin(), utmx.argmax(), utmy.argmin(), utmy.argmax()]

            # Construir el dataframe con las coordenadas de las esquinas
            for index in corners:
                series = {"iline": segy.attributes(segyio.TraceField.INLINE_3D)[index],
                      "xline": segy.attributes(segyio.TraceField.CROSSLINE_3D)[index],
                      "utmx": segy.attributes(segyio.TraceField.CDP_X)[index],
                      "utmy": segy.attributes(segyio.TraceField.CDP_Y)[index],
                      "scalar" : segy.attributes(segyio.TraceField.SourceGroupScalar)[index]}
                df = pd.concat([df, pd.DataFrame(series)],
                               ignore_index = True, axis="rows")

            # Adaptar las coordenadas de acuerdo con el byte scalar
            df['utmx'] = np.where(df['scalar'] == 0, df['utmx'],
                        np.where(df['scalar'] > 0, df['utmx'] * df['scalar'],
                                         df['utmx'] / abs(df['scalar'])))
            df['utmy'] = np.where(df['scalar'] == 0, df['utmy'],
                        np.where(df['scalar'] > 0, df['utmy'] * df['scalar'],
                                         df['utmy'] / abs(df['scalar'])))

            # Eliminar la columna de escalar
            df = df.drop(["scalar"], axis = 1)

            # Concatenating the first row once again to close the survey's polygon
            #first_row = pd.DataFrame(df.iloc[0]).transpose()
            #df = pd.concat([df, first_row], ignore_index = True, axis = 0)

            return df

    else:
        print(validar_archivos(ruta_archivo))
        return()
```

D.2. Módulo basemap

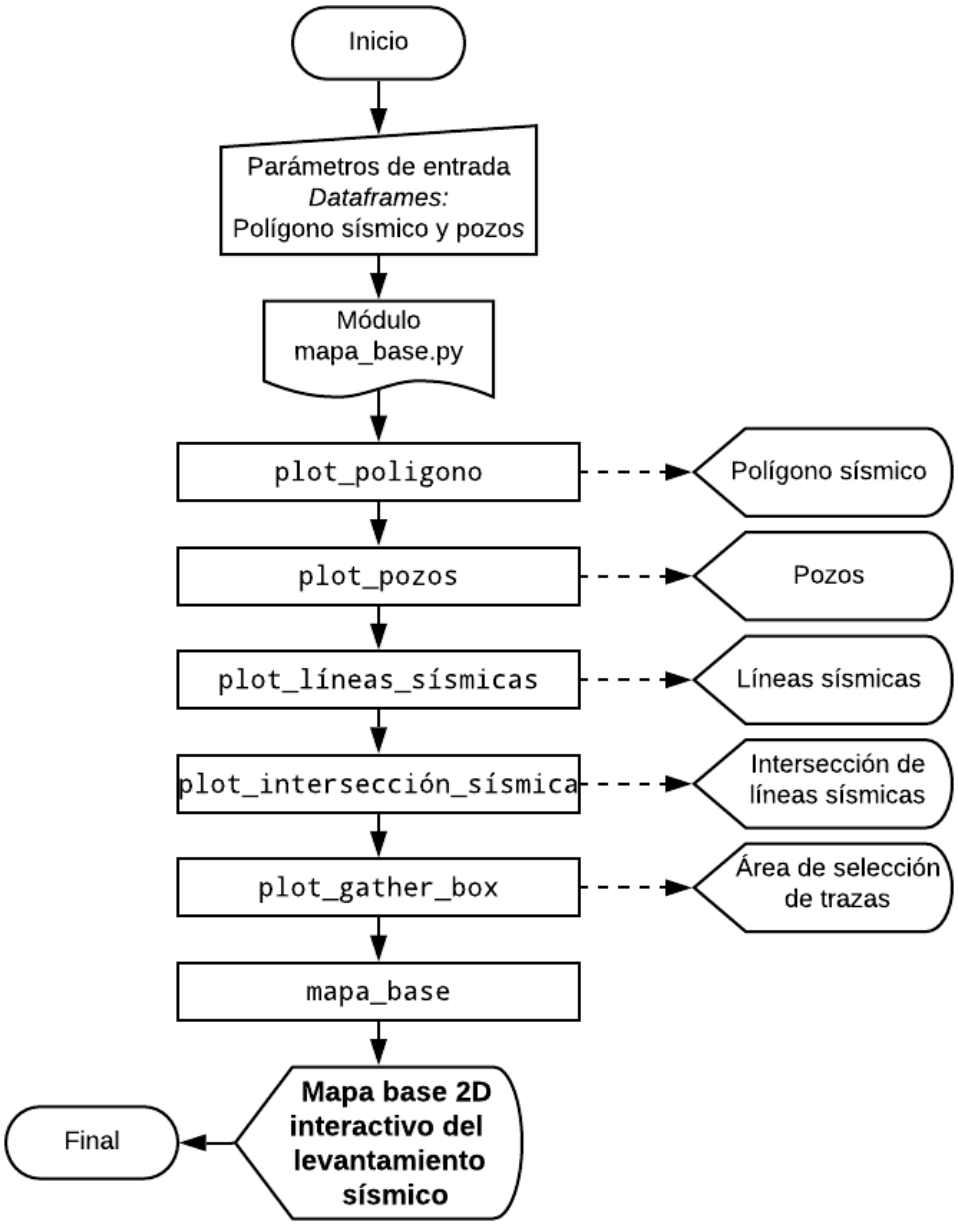

Figura D.2. Flujo de trabajo del módulo mapa_base.py.

D.2.1 validar_puntos

```
def validar_puntos(dataframe_cubo):

    # Codigo en desarrollo: calcular las esquinas del dataframe
    x1, y1 = dataframe_cubo["utmx"].iloc[0], dataframe_cubo["utmy"].iloc[0]
    x2, y2 = dataframe_cubo["utmx"].iloc[1], dataframe_cubo["utmy"].iloc[1]
    x3, y3 = dataframe_cubo["utmx"].iloc[2], dataframe_cubo["utmy"].iloc[2]
    x4, y4 = dataframe_cubo["utmx"].iloc[3], dataframe_cubo["utmy"].iloc[3]

    if dataframe_cubo["utmx"].iloc[0] == 0:
        dataframe_cubo["utmx"].iloc[0], dataframe_cubo["utmy"].iloc[0] = x4 - x3 + x2, y4 - y3 + y2

    if dataframe_cubo["utmx"].iloc[1] == 0:
        dataframe_cubo["utmx"].iloc[1], dataframe_cubo["utmy"].iloc[1] = x1 - x4 - x3, y1 - y4 - y3

    if dataframe_cubo["utmx"].iloc[2] == 0:
        dataframe_cubo["utmx"].iloc[2], dataframe_cubo["utmy"].iloc[2] = x4 - x1 - x2, y4 - y1 - y2

    if dataframe_cubo["utmx"].iloc[3] == 0:
        dataframe_cubo["utmx"].iloc[3], dataframe_cubo["utmy"].iloc[3] = x1 - x2 + x3, y1 - y2 + y3

    # Re-evaluando el dataframe
    utmx_min = dataframe_cubo.iloc[dataframe_cubo["utmx"].idxmin()]
    utmx_max = dataframe_cubo.iloc[dataframe_cubo["utmx"].idxmax()]
    utmy_min = dataframe_cubo.iloc[dataframe_cubo["utmy"].idxmin()]
    utmy_max = dataframe_cubo.iloc[dataframe_cubo["utmy"].idxmax()]

    # Reconstruyendo el dataframe en orden
    dataframe_poligono = pd.concat([utmx_min,utmy_min,utmx_max,utmy_max,utmx_min],
                            ignore_index = True, axis = "columns").transpose()

    dataframe_poligono.iline = dataframe_poligono.iline.astype(int)
    dataframe_poligono.xline = dataframe_poligono.xline.astype(int)

    return(dataframe_poligono)
```

D.2.2. plot_poligono

```
def plot_poligono(dataframe_sismico):

    # Crear el dataframe_poligono
    dataframe_poligono = validar_puntos(dataframe_sismico)

    #Plotting the boundaries of the Seismic Survey. Holoviews Curve element
    pol = hv.Curve(dataframe_poligono,["utmx","utmy"], label = "Poligono")
    pol.opts(line_width=2, color="black", tools = ['pan','wheel_zoom','reset'], default_tools=[],
          xformatter = '%.0f', yformatter = '%.0f', height = 500, width = 500, padding = 0.1,
          toolbar = 'above')

    return [pol, dataframe_poligono]
```

D.2.3. plot_pozos

```
def plot_pozos(wells_dataframe):

    # Declarando el hover tool
    wells_hover = HoverTool(tooltips=[("Utmx", "@utmx{(0.0)}"),
                                      ("Utmy", "@utmy{(0.0)}"),
                                      ("Depth", "@depth{(0)}")])
    # Ploteando pozos. Metodo Holoviews Scatter
    wells = hv.Scatter(wells_dataframe,["utmx","utmy"],
                       ["name","cdp_iline", "cdp_xline", "depth"],
                       label = "Pozos")
    wells.opts(line_width = 1,
               color = "green",size = 10 ,marker = "^",
               padding = 0.1, width=600, height=400, show_grid=True,
               tools = [wells_hover] + ['pan','wheel_zoom','reset'], default_tools=[],
              xformatter = '%.0f', yformatter = '%.0f')

    return (wells)
```

D.2.4. dataframe_lineas_sismicas

```
def dataframe_lineas_sismicas(dataframe_poligono):

    # Para reducir el estres al leer el codigo
    df = dataframe_poligono

    # Computando la diferencia entre el min y max de las lineas sismicas
    dif_ilines = abs(df["iline"].min() - df["iline"].max()) + 1
    dif_xlines = abs(df["xline"].min() - df["xline"].max()) + 1

    # vector de las coordenadas de todos los puntos en la primera inline
    x_utmx = np.linspace(float(df[df["utmy"] == df["utmy"].min()]["utmx"]),
                         df["utmx"].min(),
                         num = dif_xlines, endpoint = True)

    x_utmy = np.linspace(df["utmy"].min(),
                         float(df[df["utmx"] == df["utmx"].min()]["utmy"][0]),
                         num = dif_xlines, endpoint = True)

    # Arreglo de lineas a lo largo de la direccion inline
    xlines = np.arange(df["xline"].min(),
                       df["xline"].max() + 1,
                       1)

    # vector de las coordenadas de todos los puntos en la primera crossline
    i_utmx = np.linspace(float(df[df["utmy"] == df["utmy"].min()]["utmx"]),
                         df["utmx"].max(),
                         num = dif_ilines, endpoint = True)

    i_utmy = np.linspace(df["utmy"].min(),
                         float(df[df["utmx"] == df["utmx"].max()]["utmy"]),
```

```
                num = dif_ilines, endpoint = True)

    # Arreglo de lineas a lo largo de la direccion xline
    ilines = np.arange(df["iline"].min(),
                       df["iline"].max() + 1,
                       1)

    # creando dataframes para facilitar plots posteriores
    dataframe_xline = pd.DataFrame({"iline": df["iline"].min(),
                                    "xline": xlines,
                                    "utmx": x_utmx, "utmy": x_utmy})
    dataframe_iline = pd.DataFrame({"iline": ilines,
                                    "xline": df["xline"].min(),
                                    "utmx": i_utmx, "utmy": i_utmy})

    return([dataframe_iline, dataframe_xline])
```

D.2.5 plot_lineas_simicas

```
def plot_lineas_sismicas(dataframe_poligono, iline_numero, xline_numero):

    # Creando el dataframe de las lineas sismicas
    ilines, xlines = dataframe_lineas_sismicas(dataframe_poligono)

    # Declarando el hover tool
    iline_hover = HoverTool(tooltips=[("Inline", f"{iline_numero}"),
                                      ("Utmx", "$x{(0.00)}"),
                                      ("Utmy", "$y{(0.00)}")])

    xline_hover = HoverTool(tooltips=[("Crossline", f"{xline_numero}"),
                                      ("Utmx", "$x{(0.00)}"),
                                      ("Utmy", "$y{(0.00)}")])

    # cambiando los atributos del hover tool
    iline_hover.show_arrow, xline_hover.show_arrow = False, False
    iline_hover.point_policy, xline_hover.point_policy = "follow_mouse", "follow_mouse"
    iline_hover.anchor, xline_hover.anchor = "bottom_right", "bottom_right"
    iline_hover.attachment, xline_hover.attachment = "right", "right"
    iline_hover.line_policy, xline_hover.line_policy = "interp", "interp"

    # calculando el segundo punto para dibujar las lineas sismicas, utilizando diferencia de vectores
    ## Esto debe ser refactorizado
    iutmx = float(xlines["utmx"].iloc[-1] - xlines["utmx"].iloc[0] + ilines[ilines["iline"] == iline_numer
o]["utmx"])
    iutmy = float(xlines["utmy"].iloc[-1] - xlines["utmy"].iloc[0] + ilines[ilines["iline"] == iline_numer
o]["utmy"])
    xutmx = float(ilines["utmx"].iloc[-1] - ilines["utmx"].iloc[0] + xlines[xlines["xline"] == xline_nume
ro]["utmx"])
    xutmy = float(ilines["utmy"].iloc[-1] - ilines["utmy"].iloc[0] + xlines[xlines["xline"] == xline_nume
ro]["utmy"])

    # Cartografiando las inlines. elemento Holoviews Curve
```

```
    iline = hv.Curve([(float(ilines[ilines["iline"] == iline_numero]["utmx"]),
                       float(ilines[ilines["iline"] == iline_numero]["utmy"])),
                      (iutmx, iutmy)], label = "D. I")

    # Cartografiando las Crosslines. elemento Holoviews Curve
    xline = hv.Curve([(float(xlines[xlines["xline"] == xline_numero]["utmx"]),
                       float(xlines[xlines["xline"] == xline_numero]["utmy"])),
                      (xutmx, xutmy)], label = "D. C")

    # Añadiendo el hover tool al plot de las lineas
    iline.opts(line_width = 2, color = "red",
            tools = [iline_hover] + ['pan','wheel_zoom','reset'], default_tools=[])
    xline.opts(line_width = 2, color = "blue",
            tools = [xline_hover] + ['pan','wheel_zoom','reset'], default_tools=[])

    # Superposicion de las lineas sismicas en un plot
    lineas_sismicas = iline * xline

    return lineas_sismicas
```

D.2.6. intersección_sismica

```
def interseccion_sismica(dataframe_poligono, iline_numero, xline_numero):

    # Asignando una variable para cada dataframe asociado a las lineas sismicas
    ilines, xlines = dataframe_lineas_sismicas(dataframe_poligono)

    # Computando la cantidad de CDP a lo largo de las crosslines
    dif_xlines = abs(dataframe_poligono["xline"].max() - dataframe_poligono["xline"].min()) + 1

    # Calculo de tracf
    tracf = (iline_numero - dataframe_poligono["iline"].min()) * dif_xlines + (xline_numero - dataframe
_poligono["xline"].min()) + 1

    # Calculo de las coordenadas del tracf
    bx = float(xlines[xlines["xline"] == xline_numero]["utmx"])
    by = float(xlines[xlines["xline"] == xline_numero]["utmy"])
    ax = xlines["utmx"].iloc[0]
    ay = xlines["utmy"].iloc[0]
    cx = float(ilines[ilines["iline"] == iline_numero]["utmx"])
    cy = float(ilines[ilines["iline"] == iline_numero]["utmy"])

    tutmx = bx - ax + cx
    tutmy = by - ay + cy

    lista_de_tracf = [int(tracf), tutmx, tutmy]
    return (lista_de_tracf)
```

D.2.7. plot_interseccion_sismica

```
def plot_interseccion_sismica(dataframe_poligono, iline_numero, xline_numero):

    lista_de_tracf = interseccion_sismica(dataframe_poligono,
                                iline_numero, xline_numero)

    # Declarando el hover tool
    tracf_hover = HoverTool(tooltips=[("Tracf", f"{lista_de_tracf[0]}"),
                                ("(I/X)", f"({iline_numero}/{xline_numero})"),
                                ("(Utmx", "$x{(0.00)}"),
                                ("(Utmy", "$y{(0.00)}")])

    # Cambiando los atributos del hover tool
    tracf_hover.show_arrow = False
    tracf_hover.point_policy = "follow_mouse"
    tracf_hover.anchor = "bottom_right"
    tracf_hover.attachment = "right"
    tracf_hover.line_policy = "interp"


    # Ploteo de la interseccion. Elemento Holoviews Scatter.
    tracf_plot = hv.Scatter((lista_de_tracf[1], lista_de_tracf[2]), label = "Interseccion")
    tracf_plot.opts(size = 7, line_color = "black", line_width = 2, color = "yellow",
            tools = [tracf_hover] + ['pan','wheel_zoom','reset'], default_tools=[])

    return(tracf_plot)
```

D.2.8. gather_box

```
def gather_box(dataframe_poligono, iline_numero, xline_numero, cantidad_gathers_a_mostrar,
            inline_step, crossline_step):

    # menos estres al leer
    nd = cantidad_gathers_a_mostrar

    # coordenadas del tracf
    tracf = interseccion_sismica(dataframe_poligono,
                            iline_numero, xline_numero)

    # Puntos de las lineas sismicas
    ilines = dataframe_lineas_sismicas(dataframe_poligono)[0]
    xlines = dataframe_lineas_sismicas(dataframe_poligono)[1]

    tracf_dataframe = pd.DataFrame(columns = {"tracf","utmx","utmy"})
    lista_tracf = []
```

```
# min max de las lineas
iline_min = iline_numero - nd//2 * inline_step
iline_max = iline_numero + nd//2 * inline_step
xline_min = xline_numero - nd//2 * crossline_step
xline_max = xline_numero + nd//2 * crossline_step

# Solving boundary problems
if iline_min < dataframe_poligono["iline"].min():
    iline_max = iline_max + abs(dataframe_poligono["iline"].min() - iline_min)
    iline_min = dataframe_poligono["iline"].min()

if iline_max > dataframe_poligono["iline"].max():
    iline_min = iline_min - abs(dataframe_poligono["iline"].max() - iline_max)
    iline_max = dataframe_poligono["iline"].max()

if xline_min < dataframe_poligono["xline"].min():
    xline_max = xline_max + abs(dataframe_poligono["xline"].min() - xline_min)
    xline_min = dataframe_poligono["xline"].min()

if xline_max > dataframe_poligono["xline"].max():
    xline_min = xline_min - abs(dataframe_poligono["xline"].max() - xline_max)
    xline_max = dataframe_poligono["xline"].max()

# tracfs alrededor de la interseccion
for inline in range(int(iline_min), int(iline_max) + 1, 1):
    for xline in range(int(xline_min), int(xline_max) + 1, 1):
        coordenadas_tracf = interseccion_sismica(dataframe_poligono, inline, xline)

        lista_tracf = lista_tracf + [coordenadas_tracf + [inline, xline]]

df = pd.DataFrame(lista_tracf, columns = ["tracf","utmx","utmy", "iline", "xline"])

# extracting the possible gathers along the seismic lines
iline_gathers = df[(df.iline >= iline_min) &
                   (df.iline <= iline_max) &
                   (df.xline == xline_numero)]

xline_gathers = df[(df.xline >= xline_min) &
                   (df.xline <= xline_max) &
                   (df.iline == iline_numero)]
return([df, iline_gathers, xline_gathers])
```

D.2.9. plot_gather_box

```
def plot_gather_box(lista_gather_box):

    # Repartiendo objetos dentro de la lita_gather_box
    df, iline_gathers, xline_gathers = lista_gather_box

    # Crear un poligono que contenga los gathers
    gathers_poligono = validar_puntos(df)

    # Hover tool para el dato
    hover = HoverTool(tooltips=[("tracf", f"@tracf"),
                                ("I/X", f"@iline" + "/" + f"@xline"),
                                ("Utmx", "@utmx{(0.0)}"),
                                ("Utmy", "@utmy{(0.0)}")])

    # Cambiando los atributos del hover tool
    hover.show_arrow = False
    hover.point_policy = "follow_mouse"
    hover.anchor = "bottom_right"
    hover.attachment = "right"
    hover.line_policy = "interp"

    # Ploteando la posicion de los gathers
    plot_iline_gather = hv.Scatter(iline_gathers, ["utmx", "utmy"], ["tracf", "iline", "xline"],
                                   label = "trazas")
    plot_iline_gather. opts(size = 4, line_color = "black", line_width = 2, color = "blue",
                            tools = [hover] + ['pan','wheel_zoom','reset'], default_tools=[])

    plot_xline_gather = hv.Scatter(xline_gathers, ["utmx", "utmy"], ["tracf", "iline", "xline"],
                                   label = "trazas")
    plot_xline_gather.opts(size = 4, line_color = "black", line_width = 2, color = "red",
                           tools = [hover] + ['pan','wheel_zoom','reset'], default_tools=[])

    # Ploteando la caja de los gathers
    poligono = hv.Curve(gathers_poligono, ["utmx","utmy"]).opts(line_width = 2, line_color = "black")
    poligono.opts(tools = ['pan','wheel_zoom','reset'], default_tools=[])

    puntos = hv.Scatter(gathers_poligono, ["utmx","utmy"]).opts(size = 3, color = "black")
    puntos.opts(tools = ['pan','wheel_zoom','reset'], default_tools=[])

    #Overlay
    overlay = poligono * puntos * plot_iline_gather * plot_xline_gather

    return(overlay)
```

D.2.10. mapa_base

```
def mapa_base(dataframe_sismico, dataframe_pozos, levantamiento_sismico,
        inline_step, crossline_step):

    # Widgets
    iline_number = pn.widgets.IntSlider(name = "Inline numero",
                                start = int(dataframe_sismico["iline"].min()),
                                end = int(dataframe_sismico["iline"].max()),
                                step = 1,
                                value = int(dataframe_sismico["iline"].min()))

    xline_number = pn.widgets.IntSlider(name = "Crossline numero",
                                start = int(dataframe_sismico["xline"].min()),
                                end = int(dataframe_sismico["xline"].max()),
                                step = 1,
                                value = int(dataframe_sismico["xline"].min()))

    gather_display = pn.widgets.IntSlider(name = "Numero de gathers a desplegar",
                                 start = 2,
                                 end = 6,
                                 step = 2,
                                 value = 2)

    select_well = pn.widgets.Select(name = "Seleccione un pozo para inspeccionar",
                            options = ["Ninguno"] + list(dataframe_pozos["name"]),
                            value = "Ninguno")

    @pn.depends(iline_number.param.value, xline_number.param.value,
            gather_display.param.value,
            select_well.param.value)

    def basemap_plot(iline_number, xline_number, gather_display, select_well):

        # Primer elemento
        pol, dataframe_poligono = plot_poligono(dataframe_sismico)

        # Segundo elemento
        pozos = plot_pozos(dataframe_pozos)

        # Tercer elemento
        lineas_sismicas = plot_lineas_sismicas(dataframe_poligono,
                                    iline_number, xline_number)

        # Cuarto elemento
        gather = gather_box(dataframe_poligono,
                    iline_number, xline_number, gather_display, inline_step, crossline_step)

        gathers = plot_gather_box(gather)

        # Quinto elemento
        tracf = plot_interseccion_sismica(dataframe_poligono,
                                iline_number, xline_number)
```

```
    # Supoerposicion final
    mapa_base = pol * pozos * lineas_sismicas * gathers * tracf
    mapa_base.opts(legend_position = 'top')

    return(mapa_base)

def actualizar_barra(event):
    if select_well.value != "None":
        iline_number.value = int(dataframe_pozos["cdp_iline"].loc[str(select_well.value)])
        xline_number.value = int(dataframe_pozos["cdp_xline"].loc[str(select_well.value)])

select_well.param.watch(actualizar_barra, 'value')

widgets = pn.WidgetBox(f"## Mapa Base {levantamiento_sismico}", iline_number, xline_number,
                       gather_display, select_well)

return pn.Row(widgets, basemap_plot).servable()
```

D.3. Módulo wiggle

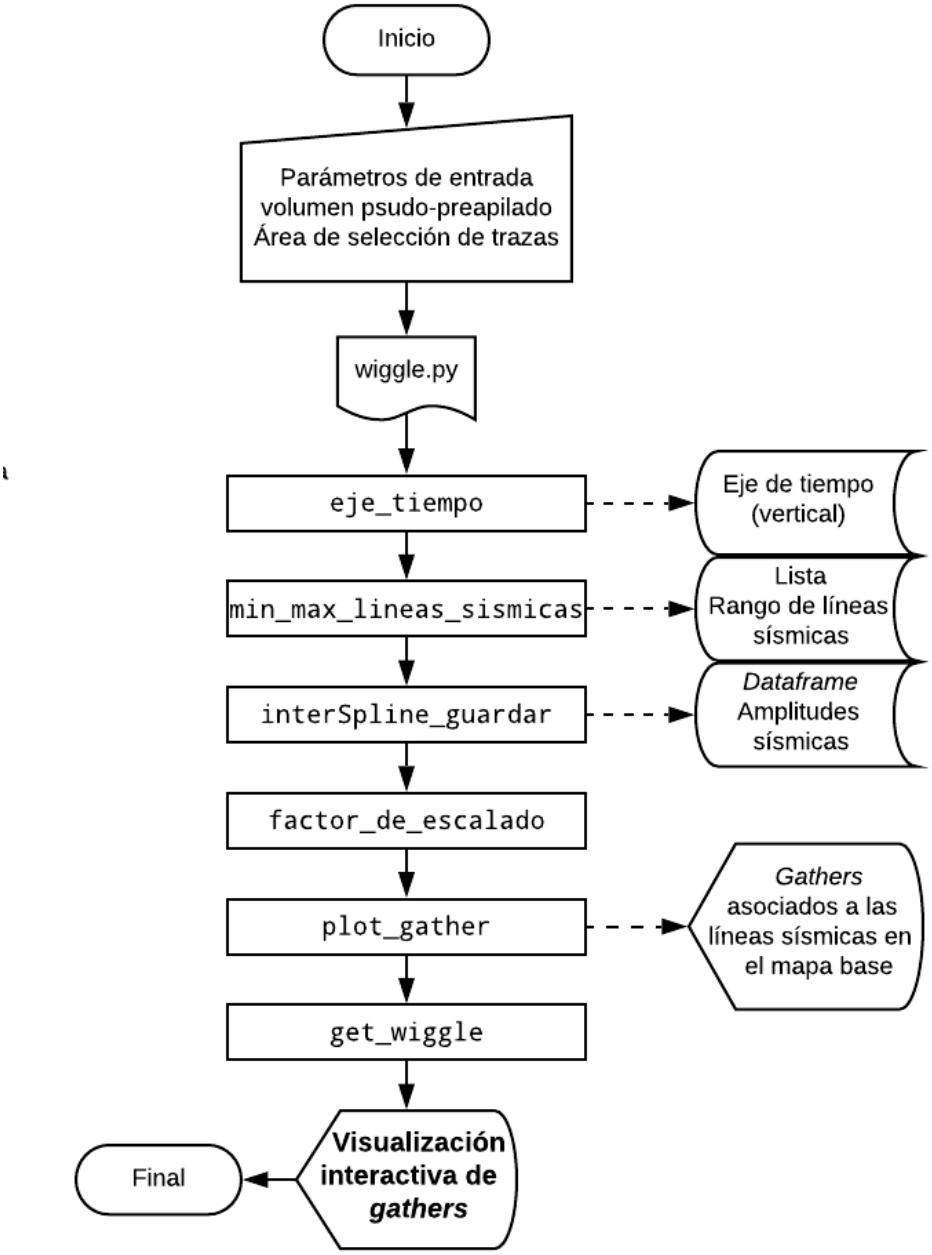

Figura D.3. Flujo de trabajo del módulo wiggle.py.

D.3.1. min_max_lineas_sismicas

```
def min_max_lineas_sismicas(df, boton, iline_numero, xline_numero,
                            cantidad_gathers_a_mostrar, inline_step, crossline_step):

    iline_min, iline_max = df["iline"].min(), df["iline"].max()
    xline_min, xline_max = df["xline"].min(), df["xline"].max()

    if boton == "Inline":
        return[iline_numero, iline_numero, xline_min, xline_max]

    else:
        return[iline_min, iline_max, xline_numero, xline_numero]
```

D.3.2. factor_de_escalado

```
def factor_de_escalado(ruta_stacks_compilados):

    scaling_fac = 0
    with segyio.open(ruta_stacks_compilados,"r") as segy:
        something = abs(segyio.tools.collect(segy.trace[:]))
        if something.max() > scaling_fac:
            scaling_fac = something.max()

    return (float(scaling_fac))
```

D.3.3. eje_tiempo

```
def eje_tiempo(ruta_stacks_compilados):

    # Cada traza tiene el mismo intervalo de muestre al igual que el numero de muestras
    primera_traza = 0
    with segyio.open(ruta_stacks_compilados,'r') as segy:
        # Extraer el intervalo de muestreo y la cantidad de amplitudes del header [us a s]
        intervalo_muestreo = float(segy.attributes(segyio.TraceField.TRACE_SAMPLE_INTERVAL)[pr
imera_traza]/1000)
        muestras_por_traza = int(segy.attributes(segyio.TraceField.TRACE_SAMPLE_COUNT)[primera
_traza])

        arreglo_tiempo = np.arange(primera_traza, intervalo_muestreo * muestras_por_traza, intervalo_m
uestreo)

    return (arreglo_tiempo)
```

D.3.4 interSpline_guardar

```
def interSpline_guardar(iline_numero, xline_numero, ruta_stacks_compilados,
                        time_interval, angulo_entre_gathers,
                        angle_bet_gathers):

    # Inicializando el dataframe de amplitud y los angulos
    dataframe_amp = pd.DataFrame([])
    angulo = angle_bet_gathers

    # Creando el eje del tiempo
    eje_y = eje_tiempo(ruta_stacks_compilados)

    #Interpolar a
    nuevo_tiempo = np.arange(eje_y[0], eje_y[-1] + time_interval, time_interval)

    dataframe_amp["time_axis"] = nuevo_tiempo
    dataframe_amp["iline"] = iline_numero
    dataframe_amp["xline"] = xline_numero

    # Extraccion del arreglo de amplitud del archivo segy
    for angulo in angulo_entre_gathers:
        with segyio.open(ruta_stacks_compilados,"r") as segy:
            amp = segy.gather[iline_numero, xline_numero, angulo]

        # Interpolacion por Spline cubicos
        f_spline = interp1d(eje_y, amp, kind = "cubic")
        new_amp = f_spline(nuevo_tiempo)

        # Creando dos series adicionales: Amplitud negativa y positiva
        dataframe_amp[f"amplitud_{angulo}"] = new_amp
        dataframe_amp[f"amplitud_positiva_{angulo}"] = new_amp
        dataframe_amp[f"amplitud_negativa_{angulo}"] = new_amp

        # Separando los polaridad de las amplitudes para el metodo Area de holoviews
        dataframe_amp.loc[dataframe_amp[f"amplitud_positiva_{angulo}"] < 0 , f"amplitud_positiva_{angulo}"] = 0
        dataframe_amp.loc[dataframe_amp[f"amplitud_negativa_{angulo}"] > 0 , f"amplitud_negativa_{angulo}"] = 0

    return (dataframe_amp.round(4))
```

D.3.5. plot_gather

```
def plot_gather(df_amp, iline_numero, xline_numero, ruta_stacks_compilados, tope_eje_tiempo,
                lista_de_angulos, angulo_entre_gathers):

    # Inicializado el plot
    ondicula = hv.Curve((0,0))

    # Calculando el factor de escalado
    f_escalado = factor_de_escalado(ruta_stacks_compilados)
```

```
    contador_de_escala = 0
    xticks = []

    # Ploteando el dato en funcion del factor de escalado
    for angulo in lista_de_angulos:

        df_amp[f"s_amplitud_{angulo}"] = df_amp[f"amplitud_{angulo}"] + contador_de_escala
        df_amp[f"s_amplitud_positiva_{angulo}"] = df_amp[f"amplitud_positiva_{angulo}"] + contador
_de_escala
        df_amp[f"s_amplitud_negativa_{angulo}"] = df_amp[f"amplitud_negativa_{angulo}"] + contado
r_de_escala

        # Asignacion del hover tool
        hover_w= HoverTool(tooltips=[('Tiempo', '@time_axis'),
                                     ('Amplitud', f"@amplitud_{angulo}"),
                                     ("Angulo",f"{angulo}")])

        # Ploteando ondiculas
        wiggle = hv.Curve(df_amp, ["time_axis", f"s_amplitud_{angulo}"],
                          [f"amplitud_{angulo}"],
                        label = "W")
        wiggle.opts(color = "black", line_width = 2, tools = [hover_w])

        # Facilitando el entendimiento del codigo posterior
        x = df_amp["time_axis"]
        y = contador_de_escala
        y2 = df_amp[f"s_amplitud_negativa_{angulo}"]
        y3 = df_amp[f"s_amplitud_positiva_{angulo}"]

        # Funcion Fill in between: elemento Holoviews Area
        negativo = hv.Area((x, y, y2), vdims=['y', 'y2'],
                           label = "-").opts(color = "red", line_width = 0)
        positivo = hv.Area((x, y, y3), vdims=['y', 'y3'],
                           label = "+").opts(color = "blue", line_width = 0)
        fill_in_between = negativo * positivo

        # Superponer las areas coloreadas con la ondicula de fase 0
        ondicula = ondicula * wiggle * fill_in_between

        # Para la siguiente iteracion
        xticks = xticks + [(contador_de_escala, angulo)]
        contador_de_escala = contador_de_escala + f_escalado

        # Añadiendo los toques finales
        ondicula.opts(height = 500, width = 150, padding = 0.1,
                      show_grid = True, xaxis = "top", invert_axes = True, invert_yaxis=True,
                      fontsize={"title": 10, "labels": 14, "xticks": 8, "yticks": 8},
                      ylabel = f"{iline_numero}/{xline_numero}", xlabel = "Time [ms]",
                      xformatter = "%.0f", yformatter = "%.0f", xticks = xticks,
                      xlim = (tope_eje_tiempo, tope_eje_tiempo + 500),
                      active_tools = ["pan", "wheel_zoom"],
                      legend_position = 'top')

    return(ondicula)
```

D.3.6 get_wiggle

```
def get_wiggle(df, iline_numero, xline_numero, ruta_stacks_compilados, lista_de_angulos,
          angulo_entre_gathers, gathers_a_desplegar, inline_step, crossline_step):

   # Mostrar
   mostrar_iline_numero = pn.widgets.StaticText(name = 'Inline seleccionada', value = str(iline_numer
o))

   motrar_xline_numero = pn.widgets.StaticText(name = 'Crossline seleccionada', value = str(xline_nu
mero))

   direccion_gathers = pn.widgets.StaticText(value = "Direccion para plotear los gathers")

   controles = pn.widgets.StaticText(value = "Controles")

   # botones
   boton_sismico = pn.widgets.RadioButtonGroup(name='Radio Button Group',
                                    options=['Inline', 'Crossline'], button_type='success')


   # barras
   ventana_de_tiempo = pn.widgets.IntRangeSlider(name = 'Ventana de tiempo [ms]',
                                start = 0,
                                end = 6000,
                                value = (0, 6000),
                                step = 500)

   intervalo_de_muestreo = pn.widgets.IntSlider(name = "Intervalo de muestreo [ms]",
                                start = 1,
                                end = 4,
                                step = 1,
                                value = 4)

   # Codigo escencial para los widgets "Decorator" para modificar el API
   @pn.depends(ventana_de_tiempo.param.value,
          intervalo_de_muestreo.param.value,boton_sismico.param.value)

   def gather_plot(ventana_de_tiempo, intervalo_de_muestreo ,boton_sismico):

      # Inicializando diccionario para el elemento holoviews layout
      dict_gathers = {}

      max_min_gathers = min_max_lineas_sismicas(df, boton_sismico, iline_numero, xline_numero,
                                gathers_a_desplegar, inline_step, crossline_step)

      for inline in range(max_min_gathers[0], max_min_gathers[1] + 1, 1):
         for crossline in range(max_min_gathers[2], max_min_gathers[3] + 1, 1):

      # Creando el dataframe de amplitud
          dataframe_amp = interSpline_guardar(inline, crossline,
                                    ruta_stacks_compilados, intervalo_de_muestreo,
                                    lista_de_angulos, angulo_entre_gathers)
```

```
    # Cortando el dataframe de acuerdo con la ventana de tiempo. en 0, no hay corte
        dataframe_amp = dataframe_amp[(dataframe_amp.time_axis >= ventana_de_tiempo[0]) &
                                      (dataframe_amp.time_axis <= ventana_de_tiempo[1])]

        # Ploteando las ondiculas
        gather = plot_gather(dataframe_amp, inline, crossline, ruta_stacks_compilados,
                             ventana_de_tiempo[0], lista_de_angulos, angulo_entre_gathers)

        # Modificando algunos parametros del plot en ciclo for
        if (inline == max_min_gathers[0]) and (crossline == max_min_gathers[2]):
            gather.opts(xlabel = "Tiempo [ms]", width = 200)
        else:
            gather.opts(xlabel = " ", yaxis = None, width = 150)

        dict_gathers[f"{inline}/{crossline}"] = gather

    # NdLayout para el diccionario
    NdLayout = hv.NdLayout(dict_gathers, kdims = f"Angle Gather (I/X)")
    return(NdLayout)

widgets = pn.WidgetBox(f"## Panel de visualizacion de Gathers", mostrar_iline_numero,
                                                motrar_xline_numero,
                                                pn.Spacer(height = 10),
                                                direccion_gathers,
                                                boton_sismico,
                                                pn.Spacer(height = 10),
                                                controles,
                                                ventana_de_tiempo,
                                                intervalo_de_muestreo)

return((pn.Row(widgets, gather_plot).servable()))
```

D.4. Módulo avo

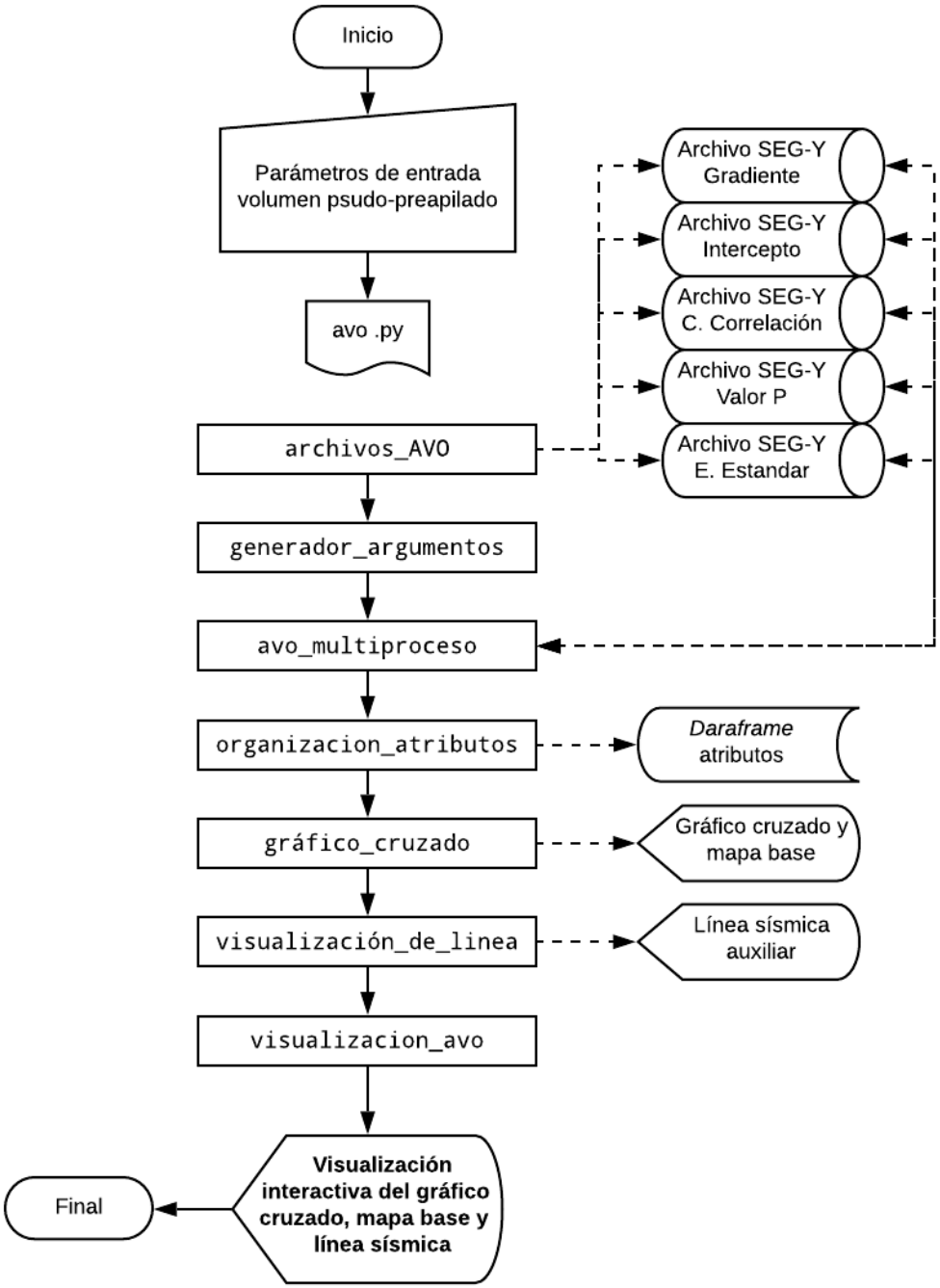

Figura D.4. Flujo de trabajo del módulo avo.py.

D.4.1. archivos_AVO

```
def archivos_AVO(lista_de_angulos, ruta_gradiente, ruta_intercepto, ruta_cCorr,
            ruta_error, ruta_desvE):

    # Inicializando stacks
    with segyio.open(lista_de_angulos[0]) as f:

        # Funcion Spec para construir nuevos segy
        spec = segyio.tools.metadata(f)

        # Inicializando el archivo compilado
        with segyio.create(ruta_gradiente, spec) as gradiente:

            # Asignando los encabezados [bytes]
            gradiente.text[0] = f.text[0]
            gradiente.bin = f.bin
            gradiente.header = f.header
            gradiente.trace = f.trace

        print(f"Creacion exitosa. Ruta de Gradiente: ({ruta_gradiente})")


    # Making copies of the new segy
    copyfile(ruta_gradiente, ruta_intercepto)
    print(f"Creacion exitosa. Ruta de Intercepto: ({ruta_intercepto})")
    copyfile(ruta_gradiente, ruta_cCorr)
    print(f"Creacion exitosa. Ruta Coeficiente de correlacion: ({ruta_cCorr})")
    copyfile(ruta_gradiente, ruta_error)
    print(f"Creacion exitosa. Ruta de error: ({ruta_error})")
    copyfile(ruta_gradiente, ruta_desvE)
    print(f"Creacion exitosa. Ruta de Desviacion estandar: ({ruta_desvE})")

    return ()
```

D.4.2. generador_argumentos

```
def generador_argumentos(ruta_stacks_compilados, list_of_angles, gradient_path, intercept_path, rval
ue_path, pvalue_path, stderr_path):

    index = 0
    with segyio.open(ruta_stacks_compilados, "r") as gather:
        for iline_number in gather.ilines:
            for xline_number in gather.xlines:
                yield [ruta_stacks_compilados, list_of_angles, iline_number, xline_number, index,
                    gradient_path, intercept_path, rvalue_path, pvalue_path, stderr_path]
                index += 1
```

D.4.3. calculo_guardado_atributos_AVO

```
def calculo_guardado_atributos_AVO(generador_de_parametros):

    # Parametros para el calculo y guardado de atributos
    gather = generador_de_parametros[0]
    lista_de_angulos = generador_de_parametros[1]
    iline_numero = generador_de_parametros[2]
    xline_numero = generador_de_parametros[3]
    indice = generador_de_parametros[4]
    gradient_file = generador_de_parametros[5]
    intercept_file = generador_de_parametros[6]
    rvalue_file = generador_de_parametros[7]
    pvalue_file = generador_de_parametros[8]
    stderr_file = generador_de_parametros[9]

    # Inicializando la una lista para el seno
    sins = []

    # Creando matrices de zeros para guardar los atributos
    with segyio.open(gather, "r") as gather:
        gradient = np.zeros([len(gather.samples), ], dtype = "float32")
        intercept = np.zeros([len(gather.samples), ], dtype = "float32")
        rvalue = np.zeros([len(gather.samples), ], dtype = "float32") #Correlation coefficient.
        pvalue = np.zeros([len(gather.samples), ], dtype = "float32")
        stderr = np.zeros([len(gather.samples), ], dtype = "float32") #Standard error of the estimated grad
ient.

        # Ciclo para agregar valores del seno y los valores de las amplitudes en vectores(3,1501)
        for angulo in lista_de_angulos:
            sins += [np.sin(np.radians(angulo)) * np.sin(np.radians(angulo))]

            # Si el angulo es el primero, inicializa en vector
            if angulo == lista_de_angulos[0]:
                amp = gather.gather[iline_numero, xline_numero, angulo].reshape(len(gather.samples), 1)

            # Si no es el primero, concaneta
            else:
                amp = np.concatenate((amp,
                                      gather.gather[iline_numero, xline_numero, angulo].reshape(len(gather.samples
),
                                                                                               1)),axis=1)

        # Ciclo for para seleccionar los parametros a introducir en la funcion de Scipy
        for row in range(amp.shape[0]):
            gradient[row], intercept[row], rvalue[row], pvalue[row], stderr[row] = stats.linregress(sins, am
p[row])

    # Guardando el dato: Abriendo y cerrando los segy's. Deberia incrementar la rendimiento
    with segyio.open(gradient_file, "r+") as gradient_segy:
        gradient_segy.trace[indice] = gradient

    with segyio.open(intercept_file, "r+") as intercept_segy:
```

```
        intercept_segy.trace[indice] = intercept

    with segyio.open(rvalue_file, "r+") as rvalue_segy:
        rvalue_segy.trace[indice] = rvalue

    with segyio.open(pvalue_file, "r+") as pvalue_segy:
        pvalue_segy.trace[indice] = pvalue

    with segyio.open(stderr_file, "r+") as stderr_segy:
            stderr_segy.trace[indice] = stderr

    print(f"Atributos AVO para el tracf {indice}, [{iline_numero},{xline_numero}], han sido caculado
s y guardado exitosamente")
```

D.4.4. avo_multiproceso

```
def avo_multiproceso(ruta_stacks_compilados, lista_de_angulos, gradient_file, intercept_file, rvalue_f
ile, pvalue_file, stderr_file):

    input_args = generador_argumentos(ruta_stacks_compilados, lista_de_angulos,
                            gradient_file, intercept_file, rvalue_file, pvalue_file, stderr_file)

    if __name__ == "__main__":
        p = Pool(maxtasksperchild = 1)
        p.map(calculo_guardado_atributos_AVO, input_args, chunksize=1)
```

D.4.5 organización_atributos

```
def organizacion_atributos(gradient_file, intercept_file, rvalue_file, pvalue_file, stderr_file,
                        rango_inline, rango_crossline, ventana_de_tiempo):

    with segyio.open(gradient_file, "r") as g, segyio.open(intercept_file, "r") as f:
        with segyio.open(rvalue_file, "r") as r, segyio.open(pvalue_file, "r") as p, segyio.open(stderr_file,
"r") as s:
            pando = pd.DataFrame([])
            for i in range(g.tracecount):
                pando = pd.concat([pando, pd.DataFrame({"inline":g.header[i][189],
                                                        "crossline": g.header[i][193],
                                                        "utmx":g.header[i][181], "utmy":g.header[i][185],
                                                        "tiempo":g.samples,
                                                        "gradiente":g.trace[i], "intercepto":f.trace[i],
                                                        "cCorr":r.trace[i], "error":p.trace[i],
                                                        "desvE":s.trace[i],
                                                        "scalar" : g.header[i][71]})],
                                    ignore_index = True, axis="rows")

            pando['utmx'] = np.where(pando['scalar'] == 0, pando['utmx'],
                                    np.where(pando['scalar'] > 0, pando['utmx'] * pando['scalar'],
                                                pando['utmx'] / abs(pando['scalar'])))
            pando['utmy'] = np.where(pando['scalar'] == 0, pando['utmy'],
```

```
                                    np.where(pando['scalar'] > 0, pando['utmy'] * pando['scalar'],
                                                pando['utmy'] / abs(pando['scalar'])))

    pando = pando.drop(["scalar"], axis=1)

    return pando[(pando["tiempo"] >= ventana_de_tiempo[0]) &
                 (pando["tiempo"] <= ventana_de_tiempo[1]) &
                 (pando["inline"] >= rango_inline[0]) &
                 (pando["inline"] <= rango_inline[1]) &
                 (pando["crossline"] >= rango_crossline[0]) &
                 (pando["crossline"] <= rango_crossline[1])].reset_index(drop=True)
```

D.4.6. grafico_cruzado

```
def grafico_cruzado(eje_x, eje_y, dataframe, escala):

    # Herramientas para seleccionar valores en el plot
    opts.defaults(opts.Points(tools=['box_select', 'lasso_select']))

    # Fuentes
    f_size = {'ticks': '10pt', 'title': '20pt', 'ylabel': '15px', 'xlabel': '15px'}

    # Escala para la escala (AHA!)
    levels = np.linspace(dataframe[escala].min(),
                         dataframe[escala].max(), 100,
                         endpoint = True).tolist()

    # Preparando el dato para plotear
    data = hv.Points(dataframe, [eje_x, eje_y], vdims = escala)
    data.opts(size = 3, width = 600, height = 600, padding = 0.01,
              title = f"{eje_x} vs {eje_y}",fontsize = f_size,
              active_tools = ["box_select"],
              toolbar='above',
              framewise = True, show_grid = True,
              color = escala, color_levels = levels, cmap = "fire", colorbar = True)

    # Axis del plot
    x_axis = hv.Curve([(0,dataframe[eje_y].min()), (0,dataframe[eje_y].max())])
    x_axis.opts(color = "black", line_width = 0.5)
    y_axis = hv.Curve([(dataframe[eje_x].min(), 0), (dataframe[eje_x].max(), 0)])
    y_axis.opts(color = "black", line_width = 0.5)

    # Declarar los puntos para el metodo selection stream
    selection = streams.Selection1D(source = data)

    # Funcion para la seleccion de puntos en el crossplot
    def selected_info(index):
        hc = dataframe.iloc[index]
        plot = hv.Scatter(hc, ["utmx","utmy"])
        plot.opts(color = "red", size = 5, width = 600, height = 600, padding = 0.1,
                  title = f"Posicion de las trazas seleccionadas",fontsize = f_size,
                  xformatter = "%.1f", yformatter = "%.1f",
```

```
            toolbar = 'above',
            tools = [HoverTool(tooltips=[('Trace [I/X]', f"@inline/@crossline"),
                        ('Tiempo [ms]', f"@time_slice")])],
            framewise = True, show_grid = True)

    return plot

dynamic_map = hv.DynamicMap(selected_info, streams = [selection])

layout = ((data * x_axis * y_axis) + dynamic_map).opts(merge_tools=False)

return (layout)
```

D.4.7. dataframe_linea_crossplot

```
def dataframe_linea_crossplot(boton, ruta_stacks_compilados, iline_numero, xline_numero, ventana_de_tiempo,
                intervalo_de_tiempo, lista_de_angulos):

    with segyio.open(ruta_stacks_compilados, "r") as f:

        # Inicializando el dataframe
        amp_df = pd.DataFrame([])

        # Creando el eje Y
        eje_t = eje_tiempo(ruta_stacks_compilados)

        # Interpolar a:
        new_time = np.arange(eje_t[0],eje_t[-1] + intervalo_de_tiempo, intervalo_de_tiempo)
        amp_df["tiempo"] = new_time
        amp_df["linea_sismica"] = boton
        amp_df["iline"] = iline_numero
        amp_df["xline"] = xline_numero

        # arreglo de trazas
        for angle in lista_de_angulos:

            # Creando otras dos serias: Amplitudes negativas y positivas
            amp_df[f"amplitud_{angle}"] = f.gather[iline_numero,xline_numero, angle]
            amp_df[f"amplitud_positiva_{angle}"] = f.gather[iline_numero,xline_numero, angle]
            amp_df[f"amplitud_negativa_{angle}"] = f.gather[iline_numero,xline_numero, angle]

            # Separando las amplitudes para la funcion fill in between
            amp_df.loc[amp_df[f"amplitud_positiva_{angle}"] < 0 , f"amplitud_positiva_{angle}"] = 0
            amp_df.loc[amp_df[f"amplitud_negativa_{angle}"] > 0 , f"amplitud_negativa_{angle}"] = 0

    return (amp_df[(amp_df.tiempo >= ventana_de_tiempo[0]) &
                (amp_df.tiempo <= ventana_de_tiempo[1])].round(4))
```

D.4.8. visualización_de_linea

```
def visualizacion_de_linea(boton, ruta_stacks_compilados, iline_number, xline_number, rango_inline, rango_crossline,
                           ventana_de_tiempo,intervalo_de_tiempo, lista_de_angulos):

    with segyio.open(ruta_stacks_compilados, "r") as f:

        if boton == "Inline":
            rango_inline = [iline_number]
            rango_crossline = np.arange(rango_crossline[0], rango_crossline[1]+1)

        else:
            rango_inline = np.arange(rango_inline[0], rango_inline[1]+1)
            rango_crossline = [xline_number]

        # Calculando el factor de escala para el eje x
        f_escalado = factor_de_escalado(ruta_stacks_compilados)
        contador_de_escala = 0
        xticks = []

        for iline in rango_inline:
            for xline in rango_crossline:
                amp_df = dataframe_linea_crossplot(boton, ruta_stacks_compilados, iline, xline, ventana_de_tiempo,
                                                   intervalo_de_tiempo, lista_de_angulos)
                for angle in lista_de_angulos:

                    amp_df[f"s_amplitud_{angle}"] = amp_df[f"amplitud_{angle}"] + contador_de_escala
                    amp_df[f"s_amplitud_negativa_{angle}"] = amp_df[f"amplitud_negativa_{angle}"] + contador_de_escala
                    amp_df[f"s_amplitud_positiva_{angle}"] = amp_df[f"amplitud_positiva_{angle}"] + contador_de_escala

                    # Designacion de hover tool
                    hover_w= HoverTool(tooltips=[('Time', '@tiempo'),
                                                 ('Amplitude', f"@amplitud_{angle}")])
                    # ploteando gathers
                    wiggle = hv.Curve(amp_df, ["tiempo", f"s_amplitud_{angle}"], [f"amplitud_{angle}"],
                                      label = "W")
                    wiggle.opts(color = "black", line_width = 2, tools = [hover_w])

                    # facilitando el codigo posterior
                    x = amp_df["tiempo"]
                    y = contador_de_escala
                    y2 = amp_df[f"s_amplitud_negativa_{angle}"]
                    y3 = amp_df[f"s_amplitud_positiva_{angle}"]

                    # Fill in between: elemento Holoviews Area
                    negative = hv.Area((x, y, y2), vdims=['y', 'y2'],
                                       label = "-").opts(color = "red", line_width = 0)
                    positive = hv.Area((x, y, y3), vdims=['y', 'y3'],
                                       label = "+").opts(color = "blue", line_width = 0)
                    fill_in_between = negative * positive
```

```
            # superposicion de las areas y ondicula de fase 0
            if contador_de_escala == 0:
                wiggle_display = wiggle * fill_in_between
            else:
                wiggle_display = wiggle_display * wiggle * fill_in_between

            # para la siguiente iteracion
            contador_de_escala = contador_de_escala + f_escalado

    # xticks
    x = np.arange(0, contador_de_escala, f_escalado * len(lista_de_angulos))
    if boton == "Inline":
        for line in range(len(rango_crossline)):
            nlines = len(rango_crossline)
            xticks += [(x[line],rango_crossline[line])]
            line = iline_number
    else:
        for line in range(len(rango_inline)):
            nlines = len(rango_inline)
            xticks += [(x[line],rango_inline[line])]
            line = xline_number


    wiggle_display.opts(height = 500, width = 500 + (10*nlines), padding = 0.01,
                        ylabel = f"{boton} {line}", xlabel = "Tiempo [ms]",
                        show_grid = True, xaxis = "top", invert_axes = True, invert_yaxis=True,
                        title = f"Posicion de las trazas seleccionadas",
                        xticks = xticks,
                        fontsize = {'ticks': '10pt', 'title': '20pt', 'ylabel': '15px', 'xlabel': '15px'},
                        xformatter = "%.0f", yformatter = "%.0f",
                        toolbar = 'above', framewise = True,
                        legend_position = 'top')

    return wiggle_display
```

D.4.9. visualizacion_avo

```
def visualizacion_avo(dataframe_sismico, ruta_stacks_compilados,
                      gradient_file, intercept_file, rvalue_file, pvalue_file, stderr_file,
                      lista_de_angulos):

    df = pd.DataFrame(columns = ["inline","crossline","tiempo",
                                 "gradiente","intercepto","cCorr","error","desvE"])
    # Seleccion de ventana
    inst = pn.widgets.StaticText(name = "Ventana de trabajo", value = "")

    rango_inline = pn.widgets.IntRangeSlider(name = 'Rango de Inline',
                                             start = int(dataframe_sismico["iline"].min()),
                                             end = int(dataframe_sismico["iline"].max()),
                                             value = (int(dataframe_sismico["iline"].min()),
                                                      int(dataframe_sismico["iline"].min())+1),
```

```
                                step = 1)

rango_crossline = pn.widgets.IntRangeSlider(name = 'Rango de Crossline',
                                start = int(dataframe_sismico["xline"].min()),
                                end = int(dataframe_sismico["xline"].max()),
                                value = (int(dataframe_sismico["xline"].min()),
                                      int(dataframe_sismico["xline"].min())+1),
                                step = 1)

# Seleccion de ventana de tiempo
barra_tiempo = pn.widgets.IntRangeSlider(name = 'Seccion de tiempo [ms]',
                                 start = 0,
                                 end = 6000,
                                 value = (3000, 3300),
                                 step = 100)

# Parametros del crossplot
eje = pn.widgets.StaticText(name = "Grafico cruzado", value = "")
eje_x = pn.widgets.Select(name = "Eje X",
                    options = list(df.columns),
                    value = "intercepto")
eje_y = pn.widgets.Select(name = "Eje Y",
                    options = list(df.columns),
                    value = "gradiente")

# Seleccion de escala
escala = pn.widgets.Select(name = "Escala de color",
                         options = list(df.columns),
                         value = "desvE")

inst2 = pn.widgets.StaticText(name = "Linea sismica", value = "")

# Botones
boton_linea = pn.widgets.RadioButtonGroup(name='Radio Button Group',
                               options=['Inline', 'Crossline'], button_type='success')

# Entrada de linea
iline_input = pn.widgets.TextInput(name = 'Inline numero',
                           value= str(rango_inline.value[0]))
xline_input = pn.widgets.TextInput(name = 'Crossline numero',
                           value= str(rango_crossline.value[0]))

# Visualizacion de linea
checkbox = pn.widgets.Checkbox(name = "Mostrar linea")


# Nuevamente, el "Decorator"
@pn.depends(rango_inline.param.value, rango_crossline.param.value,
        barra_tiempo.param.value,
        eje_x.param.value, eje_y.param.value,
        escala.param.value,
        boton_linea.param.value,
        iline_input.param.value, xline_input.param.value,
        checkbox.param.value)
```

```
def avo_stuff(rango_inline, rango_crossline, barra_tiempo,
          eje_x, eje_y, escala, boton_linea, iline_input, xline_input,
          checkbox):

   attribute_dataframe = organizacion_atributos(gradient_file, intercept_file,
                                rvalue_file, pvalue_file, stderr_file,
                                rango_inline, rango_crossline,
                                barra_tiempo)
   # Grafico cruzado
   g_cruzado = grafico_cruzado(eje_x, eje_y, attribute_dataframe, escala)

   if checkbox:
      # Line visualization
      g_cruzado += visualizacion_de_linea(boton_linea, ruta_stacks_compilados,
                                int(iline_input), int(xline_input),
                                rango_inline, rango_crossline,barra_tiempo, 4,
                                lista_de_angulos)

   return(g_cruzado).opts(merge_tools=False)


w_width = 250
w_height = 50
row1 = pn.Row(inst, height = 30, width = w_width)
row2 = pn.Row(rango_inline, height = w_height, width = w_width)
row3 = pn.Row(rango_crossline, height = w_height, width = w_width)
row4 = pn.Row(barra_tiempo, height = w_height, width = w_width)
row5 = pn.Row(eje_x, eje_y, height = w_height, width = w_width)
row6 = pn.Column(escala, pn.Spacer(height = 10),inst2, checkbox, boton_linea, width = w_width)
row8 = pn.Row(iline_input, xline_input, height = w_height, width = w_width)

widgets = pn.WidgetBox(f"## Visualizacion de AVO", row1,
                           row2,
                           row3,
                           row4,
                           eje,
                           row5,
                           row6,
                           row8,
                  pn.Spacer(height = 30, width = 250))

return pn.Row(widgets, avo_stuff).servable()
```

Apéndice E: Resultados

E.1. Mapa base

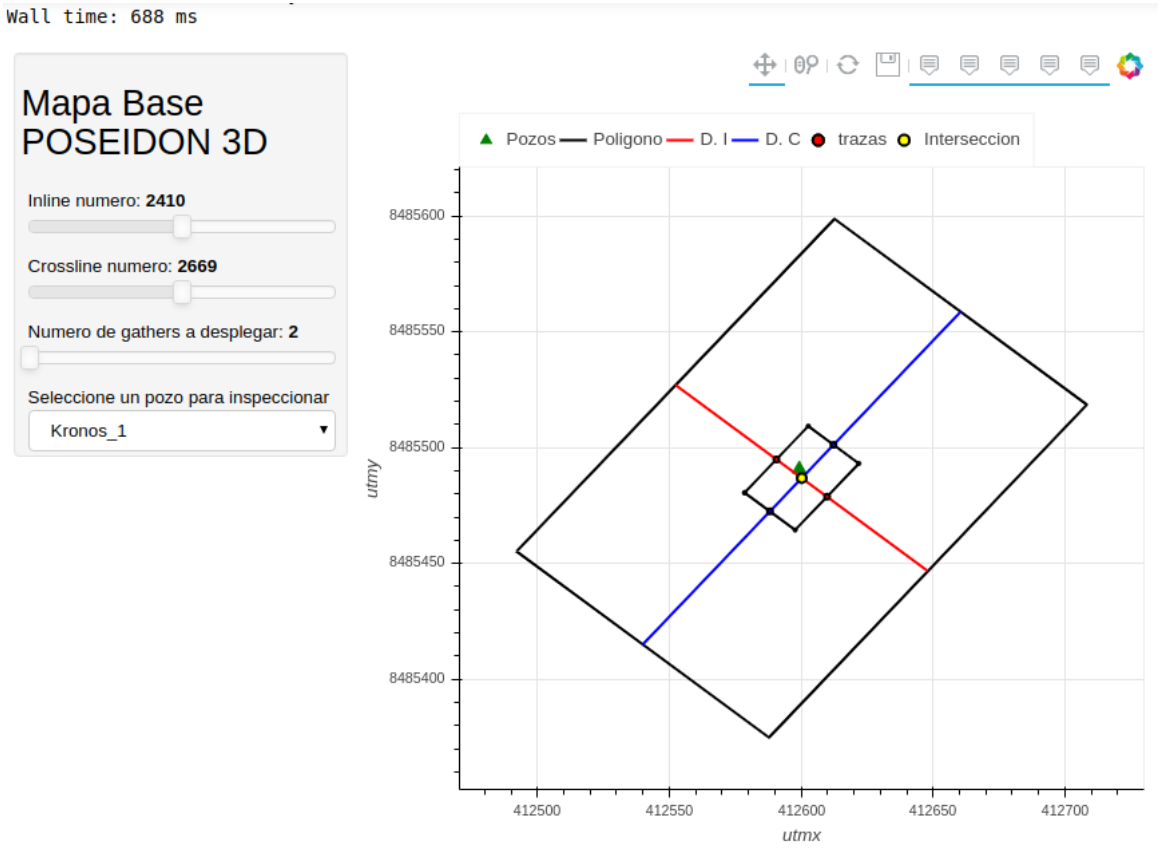

Figura E.1. Mapa base con panel de control.

E.2. Gráfico cruzado

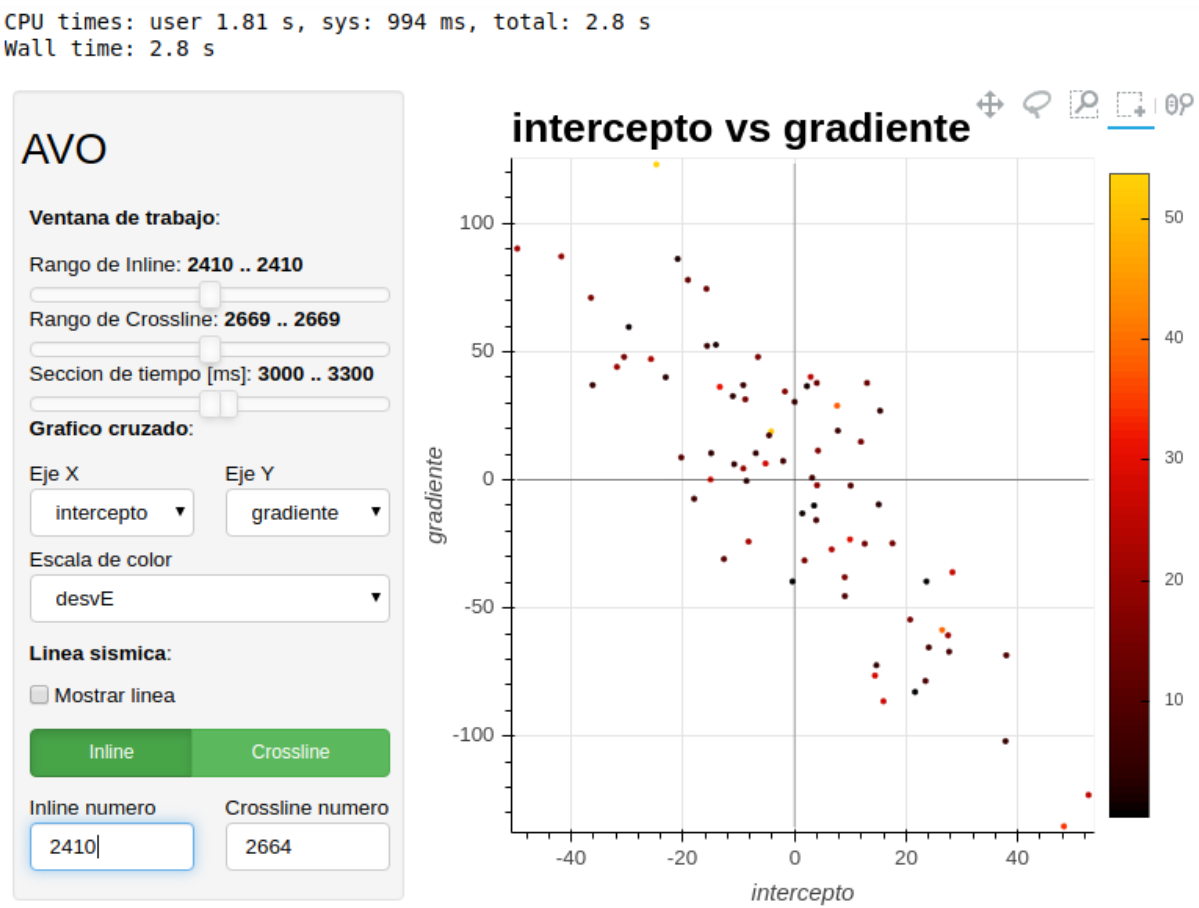

Figura E.2. Gráfico cruzado con panel de control.

Apéndice F: Glosario de términos recurrentes

F.1. Función (programación)

Una función es una porción o bloque de código reutilizable que se encarga de realizar una determinada tarea. Permite organizar el código (DevCode, 2019). Cavada (2018) define como función a todas aquellas instrucciones creadas por el usuario.

F.2. Módulo y biblioteca (programación)

Un módulo es un archivo que contiene una colección de funciones ejecutables en una interfaz que funcionan como una caja negra. Mientras que, una biblioteca es una colección de funciones y/o módulos relacionadas a un fin (Mittag, 2010). Cavada (2018) define asocia un módulo como un equivalente a una biblioteca de funciones.

F.3. Objeto: Clase y método (programación)

Una clase es un tipo especial de objeto que provee una forma de empaquetar datos y funcionalidad juntos. En tal sentido, un método se refiere a cualquier función dentro de una clase (*Python Software Foundation*, 2017).

F.4. Inline

Es la dirección en la que se adquirieron los datos. Cuando se habla de una línea sísmica, hace referencia a la línea N, paralela a la dirección en la que se adquirió el dato (Schlumberger *Oilfield Glossary*, 2019).

F.5. Crossline

Es la dirección perpendicular a la dirección de adquisición. Cuando se habla de una línea sísmica, hace referencia a la línea M, perpendicular a la dirección en la que se adquirió el dato (Schlumberger *Oilfield Glossary*, 2019).

F.6. Gather

Un *gather* se refiere a una colección de trazas sísmicas que comparten algunos atributos geométricos comunes. Estos se organizan a partir los registros de campo para examinar la dependencia de la amplitud, señal-ruido, contenido de frecuencia, fase y otros atributos sísmicos, en desplazamiento, ángulo de incidencia, acimut y otros atributos geométricos que son importantes tanto a nivel de procesamiento como de interpretación (SubSurfWiki, 2012).