UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA CIVIL

AVANCES EN EL DESARROLLO DE UNA PLATAFORMA DE PROCESAMIENTO DE REGISTRO DE VIBRACIONES AMBIENTALES Y SÍSMICAS PARA LA OBTENCIÓN DE FRECUENCIA PREDOMINANTE DE VIBRACIÓN DE UN TRANQUE DE RELAVES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL

JOSÉ ANTONIO ARCE BELTRÁN

PROFESOR GUÍA:
CÉSAR RODRIGO PASTÉN PUCHI

PROFESORA CO-GUÍA:

DIANA PATRICIA COMTE SELMAN

COMISIÓN:

BASTIAN IGNACIO GARRIDO KOGAN

SANTIAGO DE CHILE

2022

RESUMEN DE LA MEMORIA PARA OPTAR AL TÍTULO DE: Ingeniero civil, con mención en Estructuras, Construcción y Geotecnia.
POR: José Antonio Arce Beltrán.
FECHA: 2022
PROF. GUÍA: César Rodrigo Pastén Puchi.

AVANCES EN EL DESARROLLO DE UNA PLATAFORMA DE PROCESAMIENTO DE REGISTROS DE VIBRACIONES AMBIENTALES Y SÍSMICAS PARA LAOBTENCIÓN DE FRECUENCIA PREDOMINATE DE VIBRACIÓN DE UN TRANQUE DE RELAVES

Este trabajo de título tiene como objetivo avanzar en el desarrollo de una plataforma interactiva que permita la obtención de la frecuencia predominante de vibración de un tranque de relaves a través de los métodos de razones espectrales H/V (HVSR) y razones espectrales estándar (SSR). La motivación de este trabajo es realizar el procesamiento de datos de manera más rápida y determinar de forma empírica la frecuencia predominante de vibración de la zona del tranque que se desea estudiar. Para ello, el trabajo se realizó en 2 partes. La primera consiste en desarrollar e implementar el procesamiento de los registros de vibración en el tiempo, obtenidos en una campaña geofísica realizada por el Centro Avanzado de Tecnología para la Minería (AMTC) en el tranque de relaves 'El Torito', ubicado en la Región de Valparaíso, Chile. Estos registros se obtuvieron mediante una red de 28 estaciones, que incorporan un geófono de período corto (4.5 Hz) que registra en 3 componentes (norte-sur, este-oeste y vertical), una antena GPS para la sincronización entre estaciones y una batería para energizar el funcionamiento. Las estaciones fueron distribuidas en la corona, talud y base de la presa. En cada una de las estaciones se obtuvieron registros de ruido ambiental y registros sísmicos para el procesamiento HVSR y SSR, respectivamente. La segunda parte consiste en desarrollar una interfaz gráfica interactiva con el usuario que incorpora el procesamiento de registros obtenidos en la primera parte del trabajo. La interfaz para el procesamiento HVSR permite la selección de la estación, fecha y el intervalo de tiempo en que se quiere aplicar el método; mientras que para el método SSR, la interfaz permite seleccionar los sismos de interés registrados por la red (9 sismos registrados). Los resultados que entrega la interfaz, en el caso de aplicar el método HVSR, son el promedio y desviación estándar de la frecuencia predominante de vibración de la ubicación del geófono en el tranque de relaves. En el caso del método SSR, se muestra el promedio de la frecuencia predominante de vibración obtenida del registro de distintos sismos. Otras funciones que incorpora la plataforma desarrollada es guardar los datos que se muestran como resultados, mostrar las trazas de tiempo que fueron procesadas e identificar la ubicación espacial de las estaciones que fueron seleccionadas dependiendo del método que se aplique. Los resultados obtenidos de los códigos desarrollados fueron comparados con resultados obtenidos a través del software Geopsy. El trabajo se presenta como un manual para facilitar su uso.

El trabajo duro es inútil para aquellos que no creen en sí mismos.

Naruto Uzumaki.

Agradecimientos

En primer lugar, quiero agradecer a mi mamá y hermanas, que sin su amor incondicional no sería quien soy. A mi papá, si bien no me acompaña físicamente, siempre creyó en mí. Sin él no podría haber llegado donde me encuentro ahora, su exigencia y disciplina finalmente dieron sus frutos.

A mis amigos, los cuales afortunadamente he ido sumando a lo largo de todo el camino recorrido hasta ahora. Les quiero agradecer su amistad, sinceramente me llenan de alegría. Cuando uno hace las cosas por que le nacen del corazón se nota, y eso me pasa con ustedes.

A mi profesor guía, César, por su apoyo en el desarrollo del trabajo y por la gran paciencia que tuvo, lo banco profe. También a Bastián que siempre tuvo una buena disposición ayudarme en lo que fuese. Quiero destacar el apoyo de mi amigo, Juan, hemos sido compañero y amigos de aula desde kínder, es una de esas personas que motivan a uno a siempre crecer, sin él este trabajo me hubiese costado mucho más de lo que costó, simplemente un genio.

Finalmente, agradecer a todas las personas que han aportado en la construcción de mi persona, tanto las que me acompañan hoy, como las que por algún motivo u otro ya no están. Eternamente agradecido a todas las personas que en su momento creyeron, y las que actualmente creen en mí.

Tabla de contenido

Índice de Figuras

Índice de Tablas

Capítulo 1
Introducción

Producto de la actividad minera, es necesario desarrollar obras de contención de residuos mineros, debido a su posible impacto al medio ambiente. Estas estructuras reciben el nombre de tranques de relaves, cuando la presa de contención se construye con la fracción gruesa del relave. Debido a la envergadura de los procesos mineros, los tranques almacenan una gran cantidad de relaves, por lo que una falla de estas estructuras puede ser catastrófica. La falla ocurrida en el municipio de Brumadinho, Brasil en enero de 2019 (Morrill et al. 2020) es un claro ejemplo. Dicho esto, nace la necesidad de monitorear estas grandes geo-estructuras.

Junto a la automatización de procesos y, principalmente, producto de los avances científicos, se han desarrollado e implementado con mayor frecuencia métodos no invasivos para caracterizar y obtener propiedades dinámicas de suelos. Este tipo de métodos no invasivos se han vuelto más utilizados en los últimos años debido a su bajo costo (Kafadar, 2020), ya que no requiere mayores gastos por los implementos que se necesitan para levantar este tipo de campañas y caracterización geofísicas. Los ensayos geofísicos invasivos muchas veces requieren de sondajes; ejemplos de estos casos son los ensayos cross-hole y down-hole (Garofalo et al. 2016), en donde se puede obtener velocidad de onda de corte (Vs) la cual puede ser relacionada con la frecuencia predominante de vibración.

A su vez, nuevos métodos empíricos que han innovado en el área han sido formalizados y validados por distintos investigadores. Nakamura (1989) formaliza el método empírico de las razones espectrales H/V (HVSR, por sus siglas en inglés, Horizontal-Vertical Spectral Ratio) para estimar la función de amplificación y de la frecuencia predominante de un depósito de suelos. Lo innovador de este método es que emplea solo registros de vibraciones en la superficie de un depósito de suelos, a diferencia del método de las razones espectrales estándar (SSR, por sus siglas en inglés, Standard Spectral Ratio) planteado por Borchedt (1970), que emplea 2 estaciones, una ubicada en la superficie del sitio de interés y la otra en un sitio de referencia (en general, en roca o sobre un suelo rígido).

Las propiedades dinámicas de suelos anteriormente señaladas son de suma importancia, puesto que pueden ayudar a evaluar los efectos de amplificación que puede generar un sismo en un sitio. Estos efectos varían según condiciones locales de suelo (se ahondará en este punto en el Capítulo 2). Un ejemplo de esto es que, en los lugares de suelo blando los efectos locales tienen más importancia que cualquier efecto relacionado con la fuente sísmica, incluso que en los lugares cercanos a la fuente (Lermo & Chávez-García, 1993).

Para evaluar la amplificación de un depósito de suelos se puede utilizar la teoría de propagación 1D en un medio viscoelástico multicapa, que considera la profundidad de los estratos de suelos del sitio que se desea analizar, el perfil de velocidades de onda de corte, la densidad de los

materiales, entre otras. La definición de todos estos parámetros dificulta el uso de esta metodología.

En casos donde se tienen geometrías más complejas que no se pueden aproximar por métodos unidimensionales, es necesario recurrir a mediciones empíricas de HVSR y SSR, además de simulaciones en 2 y 3 dimensiones para evaluar la amplificación sísmica de las estructuras geológicas o geotécnicas. Por ejemplo, Lovati et al. (2011) comparan los resultados obtenidos de SSR y HVSR calculados a partir de registros en la Colina de Narni (Italia central) con resultados proporcionados por simulaciones numéricas 2D y 3D. Los resultados de la modelización numérica concuerdan a los de datos experimentales en cuanto a las frecuencias en la que se producen los peaks de amplificación, pero en general las amplificaciones de los modelos numéricos se subestiman en un factor que oscila entre 2 y 2,5, respecto a las amplificaciones de los modelos empíricos.

Este trabajo de título busca obtener frecuencias predominantes de vibración de un tranque de relaves, de manera empírica, mediante la automatización de procesos y visualización de resultados, gracias a un código desarrollado en lenguaje Python.

1.1 Objetivo General

El objetivo general de este trabajo es avanzar en el desarrollo de una plataforma interactiva que permita obtener frecuencias de vibración predominante de un tranque de relaves a través de procesamiento de registro de ruido ambiental y registros sísmicos, con los métodos HVSR y SSR.

1.2 Objetivos Específicos

Los objetivos específicos para desarrollar la plataforma son los siguientes:

1. Analizar los registros disponibles en una red temporal de geófonos en el tranque de relaves ’El Torito’.
2. Validar la información de los distintos sismos registrados durante la campaña geofísica.
3. Definir las funcionalidades de la plataforma junto al profesor guía.
4. Desarrollar módulos en lenguaje Python, que agilice procesamiento HVSR y SSR para un procesamientode datos más rápido.
5. Desarrollar una interfaz gráfica en lenguaje Python, que permita al usuario seleccionar una ventana de tiempo específica, en caso de procesar HVSR, o distintos sismos, en caso de querer procesar SSR para así obtener frecuencia predominante de vibración de la zona donde se encuentre(n) la(s) estación(es).
6. Comparar resultados obtenidos con software Geopsy, para su validación.

1.3 Estructura del trabajo

Este trabajo está compuesto de 7 capítulos incluyendo la sección Introducción. El Capítulo 2 corresponde a una revisión bibliográfica, en donde se explica como las condiciones locales de suelo pueden afectar la amplificación o atenuación de un movimiento sísmico. Además, se

introduce a lo que son las características dinámicas de suelos, y frecuencia predominante de vibración de un depósito de suelo (f_0). A su vez, se explican los modelos teóricos mencionados en la sección Introducción y finalmente se explican los modelos empíricos HVSR y SSR. Luego, en el Capítulo 3 se explica la metodología que se llevó a cabo para desarrollar el el trabajo. El Capítulo 4 corresponde a un manual de uso de la interfaz gráfica. Siguiendo con el Capítulo 5, se realiza una validación de resultados comparando el código desarrollado con resultados obtenidos mediante el software Geopsy. El Capítulo 6 corresponde a la discusión de resultados del trabajo y finalmente, el Capítulo 7 abarca comentarios y conclusiones.

Capítulo 2
Marco Teórico

En esta sección se revisará brevemente cómo los suelos pueden afectar la amplificación y atenuación de ondas sísmicas. Luego, se explica en qué consiste la teoría unidimensional de propagación de ondas de corte SH, para así introducir la función de transferencia y después explicar los métodos SSR y HVSR. Finalmente, se explica de qué forma se aplican los 2 métodos señalados a presas y colinas para obtener sus características dinámicas.

2.1 Condiciones Locales de suelo.

Según las leyes de atenuación, la intensidad del movimiento sísmico disminuye con la distancia del epicentro. Sin embargo, la intensidad no es solo función de la magnitud del sismo y la distancia epicentral: el comportamiento del terreno varía significativamente de un sitio a otro y depende de la conformación del subsuelo y de la forma del terreno. En el Terremoto de San Francisco de 1906, se constató por primera vez que la severidad de los daños de las edificaciones (por lo tanto, la magnitud de las aceleraciones locales) aumentaba en rellenos de suelos blandos en relación con suelos firmes (Sauter, 1989).

Un caso emblemático sobre el fenómeno de amplificación de las ondas sísmicas ha sido la Ciudad de México. La ciudad, asentada sobre los sedimentos blandos y saturados del antiguo Lago Texcoco, ha sufrido en varias ocasiones efectos de sismos lejanos. El terremoto del 19 de septiembre de 1985, a pesar de que el epicentro del sismo de gran magnitud (Mw =8.1), se localizó frente a la costa, el impacto principal del evento se experimentó en la Ciudad de México, a 400 [km] del epicentro (Celebi et al. 1987)

Por otra parte, la topografía del terreno influye significativamente en la intensidad del movimiento y puede mostrar un efecto amplificador o atenuador. Se han realizado análisis teóricos y observados casos cuyos resultados indican que ciertas formas topográficas, especialmente montes y valles, muestran varios grados de amplificación. Dependiendo del contenido de frecuencias, de la dirección y el ángulo de incidencia de las ondas sísmicas, la amplitud del movimiento aumenta en los bordes de valles, crestas y en laderas de las colinas. En cambio, en el fondo de un valle el efecto puede ser atenuador (Sauter, 1989).

El terremoto del 3 de marzo de 1985, cuyo epicentro se localizó en la costa central de la Región de Valparaíso, en Canal Beagle, villa la cual se encuentra en la comuna de Viña del Mar, ocasionó daños severos a edificaciones situadas en la cima de los montes, siendo superiores a los daños en edificios de construcción similar localizados en el valle. Posteriormente, se comprobó la influencia de la topografía en la intensidad del movimiento mediante registros simultáneos de varias réplicas; los simogramas de velocidad obtenidos en instrumentos instalados en la cima de las colinas muestran una amplificación del movimiento respecto a sismogramas del mismo evento registrados en estaciones situadas en el valle (Çelebi, 1991).

2.2 Análisis de señales

Para la obtención de HVSR y SSR los registros son sometidos a modificaciones y procesos matemáticos. Las modificaciones constan principalmente de corte del registro en donde se desee aplicar el método, un filtrado del registro, corrección por línea base. Mientras que los procesos son la Transformada de Fourier y suavizado de espectro. A continuación, se detalla las modificaciones y procesos recién señalados, los cuales son obtenidos del trabajo realizado por Pastén, C., (2007).

2.2.1 Transformada de Fourier

La Transformada de Fourier o Espectro de Fourier, S(ω), de una señal, s(t), continua, periódica, estacionaria y de energía finita se define en la Ecuación (1):

$$S(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} s(t) e^{-j\omega t} dt \qquad (1)$$

El espectro de Fourier corresponde a la representación de la señal s(t) en el dominio de las frecuencias.

En general, en los procesos no se pueden obtener muestras de forma continua, por ello se define la Transformada Discreta de Fourier de la señal discreta s, la Transformada Discreta se define en la Ecuación (

$$S_k = \frac{1}{\sqrt{2\pi}} \sum_{i=1}^{n} s_i e^{-j\omega_k (i\Delta t)} \qquad (2)$$

Donde Δt es el intervalo de muestreo. En este caso las frecuencias también son discretas y se definen en las Ecuaciones (3) y (4)

$$\omega_k = k \cdot \Delta\omega \qquad \mathrm{k} = 0 \ldots (\mathrm{n} - 1) \qquad (3)$$

$$\Delta\omega = \frac{2\pi}{T} \qquad \mathrm{T} = (\mathrm{n} - 1)\Delta\mathrm{t} \qquad (4)$$

Con *n* el número total de puntos de la señal *s(t).* y $\Delta\omega$ es el intervalo de frecuencias de muestreo

La implementación computacional de la Transformada Discreta de Fourier es la Fast Fourier Transform, FFT (Press et al., 1992), la cual se emplea en el código desarrollado en el presente trabajo.

2.2.2 Corrección por línea de base

Si $\bar{s}$ es el promedio de la señal s en un intervalo de tiempo n· Δt, entonces s_r, la señal corregida por línea de base es definida en la Ecuación (5)

$$s_{ri} = s_i - \bar{s} \qquad \text{i} = 1,2,3 \ldots n \tag{5}$$

Donde $\bar{s} = \frac{\sum_{i=1}^{n} s_i}{n}$

La corrección anterior es para evitar que el primer término de la transformada de Fourier, S(0), sea distinta de cero.

2.2.3 Ventana o Taper

La aplicación de una ventana o taper a una señal es la multiplicación en el tiempo de sus elementos, por una función suave y nula en los extremos. En la Ecuación (6), se define matemáticamente a lo que corresponde.

$$s_{wk} = s_k \cdot w_k \qquad k \in (1, \ldots n) \tag{6}$$

Donde s_{wk} es el elemento k de la señal ventaneada, s_k es el elemento k de la señal s y w_k el elemento k de la ventana w.

Al forzar una señal a cero en los extremos previene la aparición de un evento conocido como leakage, que es la aparición de altas frecuencias producidas por el corte repentino de la señal al inicio y al final, considerando que la señal es períodica.

En este trabajo se empleará la ventana de Tukey o ventana de coseno atenuado, que se define en la Ecuación (7).

$$w_k = \begin{cases} \frac{1}{2}\left[1 + cos\left(\frac{2\,\pi(k-1)}{k\,(n-1)} - \pi\right)\right] & si \quad k < \frac{r}{2}(n-1) + 1 \\ 1 & si \quad \frac{r}{2}(n-1) + 1 \le k \le n - \frac{r}{2}(n-1) \\ \frac{1}{2}\left[1 + cos\left(\frac{2\,\pi}{r} - \frac{2\,\pi(k-1)}{r(n-1)} - \pi\right)\right] & \text{si} \quad n - \frac{r}{2}(n-1) < k \end{cases} \tag{7}$$

El valor de n corresponde al número de puntos de la ventana, k corresponde al punto k-ésimo de la ventana n y r al porcentaje de largo de la ventana ($r \le 1$). Un valor de r=0 corresponde a una función cajón y un valor de r=1 corresponde a una ventana de Hanning. En el presente trabajo se empleará un r con un valor de 5%.

Si bien el empleo de ventanas sobre señales tiene las ventajas señaladas, una de sus desventajas es que deforma la amplitud de la señal, sobre todo hacia los extremos. Por lo que la elección de la función debe ser adecuada a las características de la señal. Una señal no estacionaria, como por

ejemplo un pulso de energía, debe ser ventaneada procurando centrar la parte con mayor registro de energía para no perder información en amplitud.

2.2.4 Suavizado de espectros

En general, las representaciones espectrales de una señal tienen una gran dispersión. Así, los espectros se suavizan multiplicándolos por una función que permita mantener las características medias y elimine los saltos abruptos para obtener la tendencia promedio.

La función que se emplea para suavizar los espectros del presente trabajo corresponde a la función definida por Konno y Ohmachi (1998). Esta función viene dada por la Ecuación (8).

$$w_k = \left[\frac{\text{sen}(b \cdot \log_{10}(\frac{f}{fc}))}{b \cdot \log_{10}\left(\frac{f}{fc}\right)} \right]^4 \qquad (8)$$

Donde f_c es la frecuencia central de la ventana, f son las frecuencias donde se evalúa el espectro y b es un coeficiente de ancho de banda.

La Figura 1 muestra la función de suavizado evaluada en una frecuencia de 2[Hz], donde se pueden ver distintos valores de ancho de banda, *b*. Mientras menor es el valor de *b*, más suave es el espectro resultante.

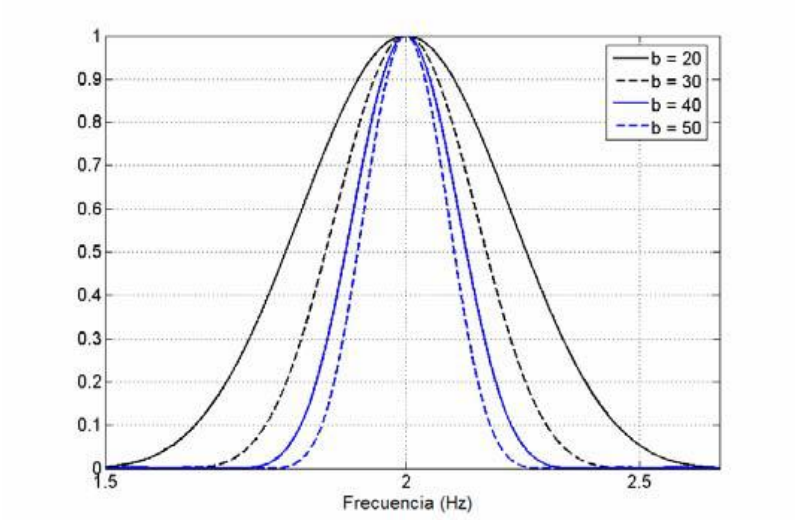


Figura 1: Función de suavizado de Konno y Ohmachi de frecuencia central f_c= 2 [Hz] y distintos valores de ancho de banda *b*. (Pastén, 2007)

Una de las ventajas de utilizar esta función para suavizar espectros es que cuando se utilizan representaciones semi-logarítmicas, como el método HVSR, esta ventana mantiene un ancho de banda gráfico constante.

Según los resultados de Konno y Ohmachi (1998), utilizando esta función para suavizar los espectros, antes de calcular la razón espectral, se pierde menos información en amplitud espectral que utilizando cualquier otro tipo de ventana.

La Figura 2 presenta la diferencia entre espectros suavizados mediante la función de Konno y Ohmachi, con distintos anchos de banda.

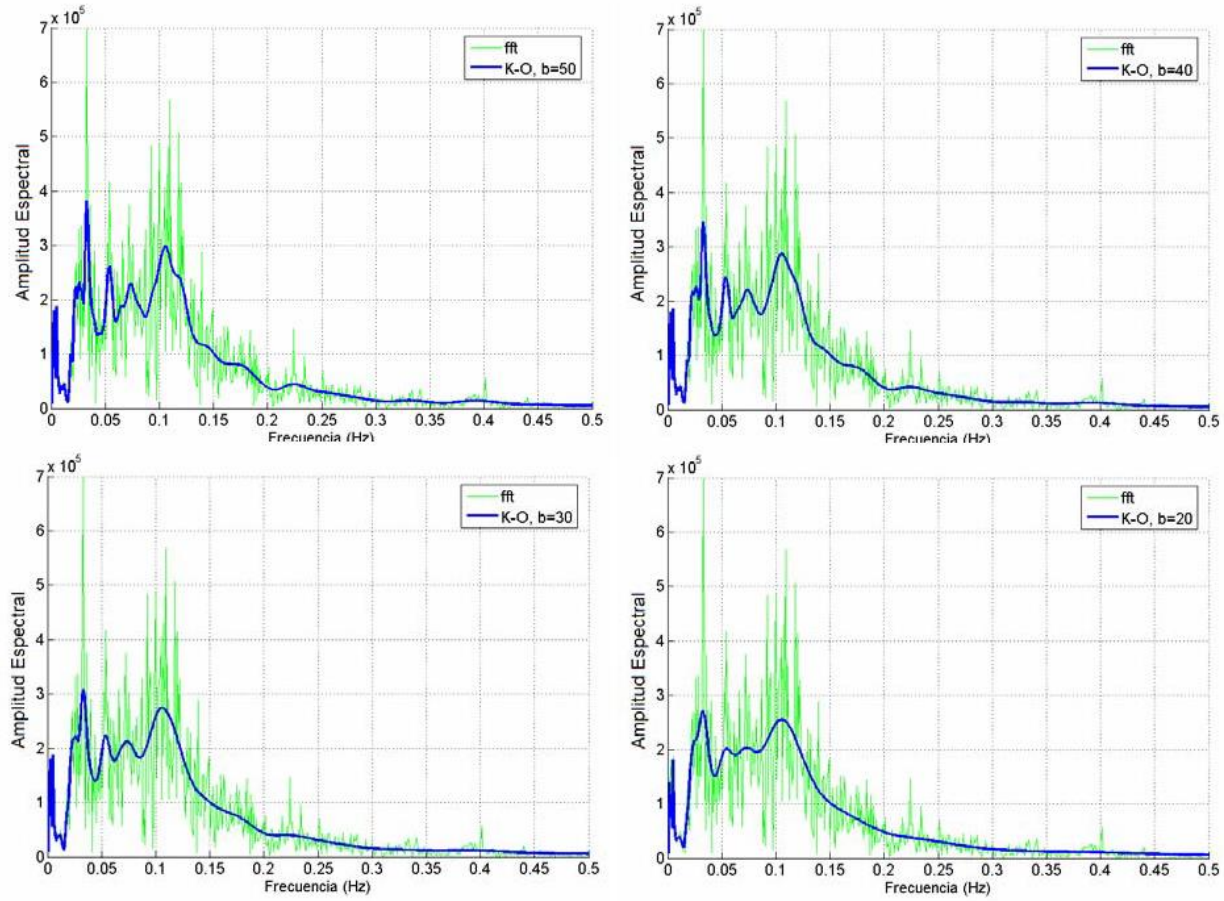


Figura 2: Espectro de Fourier suavizado con funciones de Konno y Ohhmachi de distinto ancho de banda. (Pastén, 2007)

2.3 Respuesta sísmica drenada de depósito de suelos

2.3.1 Teoría unidimensional de propagación de ondas de corte SH

Este método simplificado de respuesta sísmica asume que sólo existe una propagación vertical de ondas de corte y se utiliza para evaluar la respuesta de depósitos de suelo no saturados donde no hay posibilidad de generación de presión de poros.

El modelo unidimensional de propagación de ondas de corte SH se estudia mediante la función de transferencia de este tipo de ondas, la cual se obtiene al realizar un equilibrio dinámico en un elemento diferencial de suelo, considerando además la compatibilidad de desplazamientos entre los distintos estratos de suelo (González, 2018).

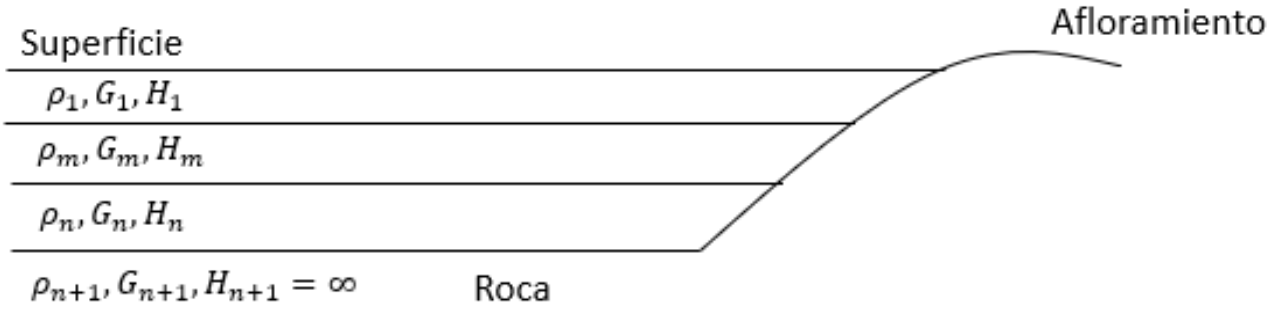


Figura 3: Modelo multicapas de un depósito de suelo.

Considerando un depósito de suelos multicapas como se muestra en la Figura 3, la función de transferencia $FT_{i,j}$ se define en la Ecuación (9).

$$FT_{ij}(\omega) = \frac{E_i(\omega) + F_i(\omega)}{E_j(\omega) + F_j(\omega)} \tag{9}$$

Donde $E_j(\omega)$, $F_j(\omega)$ están dadas por las Ecuaciones (10) y (11).

$$E_{j+1} = \frac{1}{2}\left(E_j(\omega)\left(1+\Delta_j\right)e^{ik_j{}^*(\omega)\cdot H_j} + F_j(\omega)\left(1-\Delta_j\right)e^{-ik_j{}^*(\omega)\cdot H_j}\right) \tag{10}$$

$$F_{j+1} = \frac{1}{2}\left(E_j(\omega)\left(1-\Delta_j\right)e^{ik_j{}^*(\omega)\cdot H_j} + F_j(\omega)\left(1+\Delta_j\right)e^{-ik_j{}^*(\omega)\cdot H_j}\right) \tag{11}$$

$$k_j{}^*(\omega) = \frac{\omega}{V_{sj}\sqrt{1+2iD_j}} \tag{12}$$

Donde k_j corresponde al número de onda complejo.

El contraste de impedancia Δ_j entre los estratos j y (j+1) se calcula a partir de la Ecuación (13).

$$\Delta_j = \frac{\rho_j \cdot V_{sj}}{\rho_{j+1} \cdot V_{s(j+1)}} \sqrt{\frac{1 + 2iD_j}{1 + 2iD_{j+1}}} \quad (13)$$

De las expresiones anteriores, H_j es el espesor del estrato j, ρ_j es la densidad del estrato j, G_j es su módulo de corte y D_j es su amortiguamiento.

Al graficar la función de transferencia FT entre la superficie y el basamento rocoso en función de la frecuencia (Ecuación (9)), suponiendo un depósito de suelo de 50[m] de profundidad, con una velocidad de onda de corte de 550[m/s], una densidad de 2000[kg/m3] y un amortiguamiento del 2%, se obtiene un gráfico como el de la Figura 4, donde la frecuencia asociada al primer y más alto peak se conoce como frecuencia predominante de vibración del depósito de suelo f_0 (González, 2018).

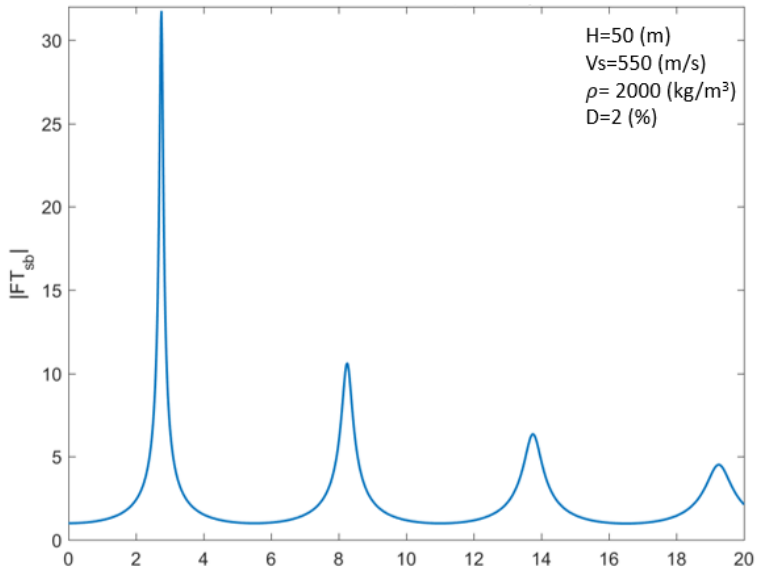


Figura 4: Función de transferencia de un depósito de suelo,fuente: (González, 2018).

2.4 Métodos numéricos de evaluación de respuesta en superficie

2.4.1 Métodos analíticos

Hay numerosas soluciones analíticas propuestas para estudiar el efecto de sitio en configuraciones geométricas simples, entre las cuales se encuentran: capas horizontales, cuñas de suelo, valles semicirculares, triangulares y rectangulares, etc.

La configuración de capas planas horizontales que es incidido por un frente de ondas planas y elásticas es una de las más empleadas y se puede analizar a través del Método de Thomson-Haskell.

La solución exacta de una configuración de suelo tipo cuña que es excitado por ondas SH, predice que la amplitud del desplazamiento del vértice de la cuña es independiente del ángulo de incidencia del frente de ondas para una configuración fija, y que por otra parte, es inversamente proporcional al ángulo interno de la cuña. Esto se puede utilizar para analizar efectos topográficos en depósitos de suelos de geometría regular.

Las soluciones analíticas para la respuesta de valles aluvionales y cañones bajo la incidencia de ondas SH, muestran una fuerte amplificación dada por la geometría de la configuración, pero también dada por las diferencias entre las propiedades de los sedimentos y el semi-espacio, y el ángulo de incidencia del frente de ondas.

2.5 Métodos experimentales de evaluación de respuesta en superficie

2.5.1 Método de las razones espectrales estándares SSR

El método de las razones espectrales estándares fue introducido por Borcherdt (1970) siendo el más utilizado y el que entrega resultados más confiables para determinar la respuesta de sitio. Emplea una estación como referencia en la base del depósito a analizar o directamente sobre un macizo rocoso. Se basa en obtener los factores de amplificación gracias a la razón espectral de un registro de movimiento en la superficie de un depósito con respecto a la estación de referencia.

Al comparar los espectros de Fourier de la misma componente de las 2 estaciones se puede observar de manera fácil cuanto amplifica la señal el cuerpo en estudio y a que frecuencia lo hace.

El factor de amplificación se determina como:

$$SSR = \frac{H_{est}}{H_{ref}} \tag{14}$$

Donde H_{est} es la amplitud espectral del movimiento horizontal en la parte que se desea estudiar y H_{ref} corresponde a la amplitud espectral del movimiento horizontal en la estación de referencia.

En la Figura 5, se puede ver una representación esquemática de la amplificación de una señal sísmica, en donde en la base del sistema en estudio se encuentra la estación de referencia y en su punto más alto la estación de estudio para estudiar la amplificación sísmica producto de las condiciones locales.

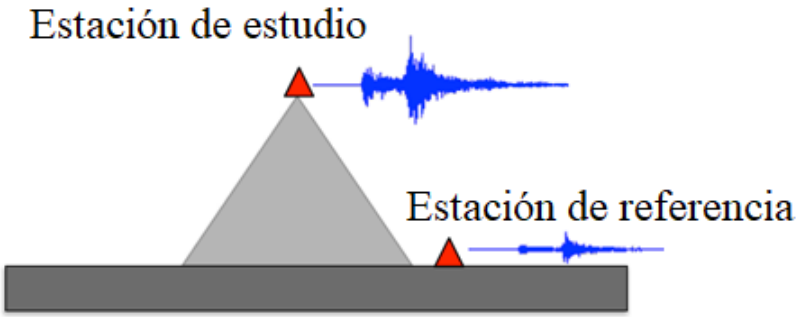


Figura 5: Representación esquemática de amplificación de señal sísmica.

2.5.2 Método HVSR

Este método, también conocido como método de Nakamura (1989), propone estimar el factor de amplificación o la función de transferencia de un depósito de suelos superficial, solicitado por un frente de onda SH que propaga verticalmente desde la base hasta la superficie. Lo anterior, a partir de mediciones de vibraciones ambientales y sísmicas de las componentes verticales y horizontales.

Típicamente, esto es obtenido en una ubicación en superficie de un registro de las 3 componentes (N-S, E-O, Z). Este método fue inicialmente desarrollado en Japón por Nogoshi e Igarashi (1971), basado en los estudios de Kanaj y Tanaka (1961), quienes caracterizaron la respuesta de un sitio a través de registros de ruido sísmico. Por otra parte, el método H/V obtenido a partir de vibraciones ambientales es popularizado por Nakamura (1989) y revisado por Bonneyfoy-Claude et al. (2006).

2.5.2.1 Método HVSR de vibraciones ambientales

Los depósitos de suelos generalmente están expuestos a vibraciones generadas por fuerzas naturales, como mareas y vientos. Por otra parte, están expuestos a vibraciones artificiales producidas por actividades industriales, vehículos, peatones, etc. La suma de ambos efectos es lo que se denomina vibraciones ambientales y es lo que se mide en superficie utilizando el método HVSR.

Al registrar vibraciones ambientales por un largo período de tiempo con una serie de sensores, se logra identificar un conjunto de ondas coherentes que viajan en varias direcciones lo largo de un intervalo que suele incluir frecuencias entre 0,5 y 20 [Hz] (Molnar et al., 2018)

Las fuentes generadoras de vibraciones naturales, por ser de mayor envergadura, en suma, producen una solicitación dinámica aleatoria, la cual permite que un depósito de suelos tienda a vibrar preponderantemente de acuerdo a su frecuencia predominante, considerando como la frecuencia predominante aquella en que la razón espectral es máxima (peak de la razón espectral) (Pastén, 2007). Solicitaciones de menor energía como las que generan vibraciones artificiales tienden a excitar capas de suelo más superficiales, que responden a frecuencias relativamente más altas. por lo tanto es recomendado aplicar el método en ventanas de tiempo en donde no exista ruido

artificial provocado por la actividad humana (Nakamura, 1989). Además, las vibraciones artificiales sostenidas afectarán los registros adversamente en caso de encontrarse dentro del mismo ancho de banda de frecuencia que la frecuencia natural de la zona en estudio; las técnicas de filtrado no pueden eliminar estos ruidos artificiales en los registros. Por lo tanto, de ser el caso, es necesario repetir el ensayo cuando la maquinaria no esté funcionando (Molnar et al., 2018).

Tomando en cuenta que mediciones de vibraciones ambientales en afloramientos rocosos no presentan una dirección predominante de movimiento, de existir alguna amplificación del movimiento en la superficie es producto del efecto de las capas de suelos depositadas sobre la roca basal o basamento rocoso.

La función de transferencia, F_T, de un depósito de suelos que se asocia a la propagación de una onda de corte SH se define en la Ecuación (15).

$$F_t = \frac{S_{HS}}{S_{HB}} \tag{15}$$

Donde S_{HS} corresponde a la amplitud del espectro de Fourier de la componente horizontal en superficie y S_{HB} corresponde a la amplitud del espectro de Fourier de la componente horizontal de la base del depósito de suelos en estudio.

Nakamura (1989) suponiendo que la componente vertical del movimiento no es amplificada por depósitos de suelos se tiene la Ecuación (16).

$$1 = \frac{S_{VS}}{S_{VB}} \tag{16}$$

Además, propone el siguiente término:

$$E_s = \frac{S_{VS}}{S_{VB}} \tag{17}$$

Que representa el efecto de la onda de Rayleigh en el movimiento vertical. Si no hay ondas de Rayleigh la expresión anterior se puede aproximar a la unidad. Luego propone el siguiente término:

$$S_{TT} = \frac{S_T}{E_S} \tag{18}$$

En donde S_{TT} es considerado como la función de transferencia que ha logrado eliminar el efecto de las ondas de Rayleigh.

Incorporando el supuesto que en la base del depósito de suelos el movimiento es igual en todas las direcciones, se tiene la Ecuación (19).

$$\frac{S_{HB}}{S_{VB}} \approx 1 \tag{19}$$

Donde S_{HB} corresponde al espectro de amplitud de Fourier de la componente horizontal del movimiento en la base y S_{VB} corresponde al espectro de amplitud de la componente vertical en el mismo lugar.

Luego:

$$S_{TT} = \frac{\left(\frac{S_{HS}}{S_{HB}}\right)}{\left(\frac{S_{VS}}{S_{VB}}\right)} \tag{20}$$

Desarrollando la ecuación (20), se tiene que:

$$S_{TT} = \left(\frac{S_{HS}}{S_{VS}}\right) \cdot \left(\frac{S_{HB}}{S_{VB}}\right) \tag{21}$$

En donde el segundo término donde se relacionan los espectros de la base es aproximadamente la unidad, por lo que se tiene:

$$HVSR = \frac{S_{HS}}{S_{VS}} \tag{22}$$

Lo anterior descrito plantea que la función de transferencia asociada a la propagación vertical de ondas de corte de un depósito de suelos puede ser estimada simplemente a partir de movimientos medidos en superficie.

En la Figura 6, se puede observar un registro diario de una estación del tranque de relaves “El Torito”, en donde se destaca el registro de ruido ambiental y el registro de ruido artificial proveniente de la actividad humana generada en el tranque de relaves.

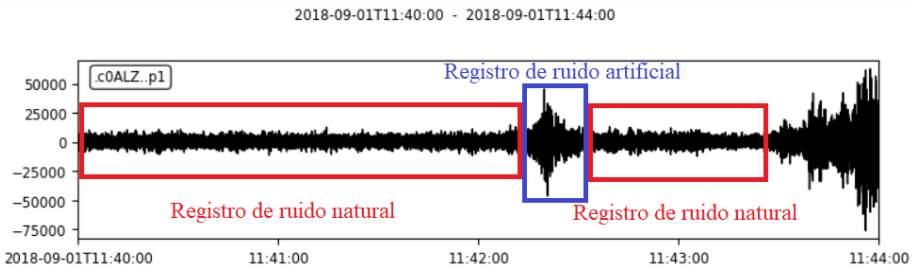


Figura 6: Registro de ruido ambiental.

Por otra parte, en cuanto a la duración del registro de ruido ambiental, esta debe ser proporcional a la frecuencia predominante del sitio en estudio; un sitio con baja frecuencia predominante requiere una duración de registro más larga para proporcionar suficientes ventanas de tiempo para establecer estadísticamente la estabilidad del método HVSR (Molnar et al, 2018)).

2.5.2.2 Método HVSR de registros sísmicos

Corresponde a la razón espectral de la componente horizontal y vertical, pero de ventanas de tiempo de registros sísmicos. Lermo y Chavez (1993) emplearon este método de forma pionera con registros sísmicos de 3 ciudades de México. En el trabajo analizaron una ventana temporal de diez segundos, que es el tiempo en donde se registró la parte fuerte de los sismos, correspondiente a las aceleraciones máximas. Los resultados que obtuvieron fueron que las razones espectrales H/V aplicada a registros sísmicos entrega valores de frecuencias predominantees y amplificaciones de los suelos que son similares con los obtenidos mediante el método de razones espectrales estándar. Además, el método se puede aplicar para identificar efectos producidos por depósitos de suelos y efectos topográficos.

2.5.3 Limitaciones de métodos SSR y HVSR

Las principales limitaciones del método SSR es la instalación de instrumentos en la base del depósito, que es muy costosa, en este caso, en la base de un tranque de relaves que es el sistema de estudio de este trabajo. En cuanto a la instalación de una estación en afloramiento rocoso, este debe estar lo suficientemente cerca de la estación que se desea estudiar, para asegurar que la diferencia de los registros sísmicos sea producida por las condiciones locales del terreno. Además, la estación que se encuentre en el afloramiento rocoso debe estar libre de efectos de sitio y topográficos.

Por otra parte, las principales limitaciones para el uso de HVSR son que la principal utilidad de este método está en la determinación de la frecuencia predominante de un depósito de suelos blandos. Por otra parte, para el caso sísmico, el método HVSR solo sirve para sismos de baja magnitud, ya que para magnitudes mayores se cree una pérdida de linealidad (Lermo & Chávez, 1993).

2.6 Métodos para aproximar o determinar características dinámicas de colinas y presas

Lovati et al. (2011) estudiaron las amplificaciones producto de las condiciones topográficas de una colina. Este estudio consistió en comparar amplificaciones sísmicas debido a la topografía de la colina de Narni, en Italia central, obtenidas experimentalmente y a partir de análisis numéricos

2D y 3D. La cresta de una colina corresponde a la línea máxima en un relieve de montaña. Para esto, se instalaron distintas estaciones sísmicas en la zona ubicadas en la base, talud y cresta de la colina. Al comparar registros de las estaciones de la base con los de la cresta, se observó que las amplificaciones máximas se obtienen en la dirección perpendicular a la elongación principal de la cresta.

Gazetas (1987) plantea que para hacer una predicción realista de la respuesta de una presa de material particulado hay que tener en cuenta los siguientes fenómenos y factores:

a) Comportamiento no lineal-inelástico del suelo:
b) Dependencia de la rigidez del suelo con respecto a la presión de confinamiento
c) Geometría.
d) Interacción presa-relaves.

Dependiendo de la situación, uno de estos fenómenos puede tener mayor influencia sobre la geo-estructura que otro. Por lo tanto, la situación en cuestión dictará el método de análisis adecuado.

Es importante tener en cuenta cual factor de los anteriores es necesario modelar de manera más sofisticada. Por ejemplo, en el caso de las presas modernas rígidas sometidas a aceleraciones máximas de 0.2[g] o menores es necesario modelar adecuadamente los factores (b), (c) y (d). Por otra parte, los efectos de (b) y (c) puede ser contrarrestados por el efecto del factor (d). en efecto, a medida que el grado de heterogeneidad de la presa aumenta (debido a que el confinamiento disminuye a medida que aumenta la cota) y a medida que se hace más estrecha la presa tiende a producirse un efecto “latigazo”. Como resultado de lo anterior, se desarrollan altas aceleraciones absolutas y altas deformaciones en el cuarto superior de la presa. Lo anterior guarda estrecha relación con lo propuesto por Lovati (2011).

Por otra parte, la interacción de la presa con relaves relativamente blandos tiende a aumentar el amortiguamiento efectivo debido a la radiación de la energía de las ondas y a filtrar algunos componentes de alta frecuencia de la excitación que tiende a producir altas aceleraciones en la cresa, por lo que puede cambiar las características de los modos de vibración de la estructura, contrarrestando los cambios respectivos causados por el factor (b) y (c) (Gazetas, 1987).

2.6.1 Comparación de teoría analítica con métodos empíricos para obtención de frecuencia predominante de vibración.

Distintos autores plantean una expresión simplificada para obtener la frecuencia predominante de vibración de una geo-estructura con forma de triángulo, la geometría de la geo-estructura se muestra en la Figura 7.

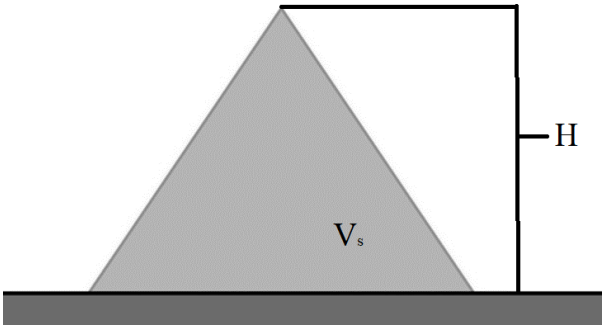


Figura 7: Geometría simplificada de una geo-estructura con forma de triángulo.

Calderón & Cordone (2019) realizaron estudios empíricos en el embalse "Potrerillos", ubicada en la Provincia de Mendoza. Para ello, se realizó un estudio de la frecuencia predominante de la presa y su variación, a través de funciones de transferencia obtenidas de los eventos registrados mediante cuatro acelerómetros, como se observa en la Figura 8. Las estaciones se ubicaron a distintas alturas aguas abajo de la presa y una ubicada en un afloramiento rocoso cercano. Para el estudio se empleó el método SSR.

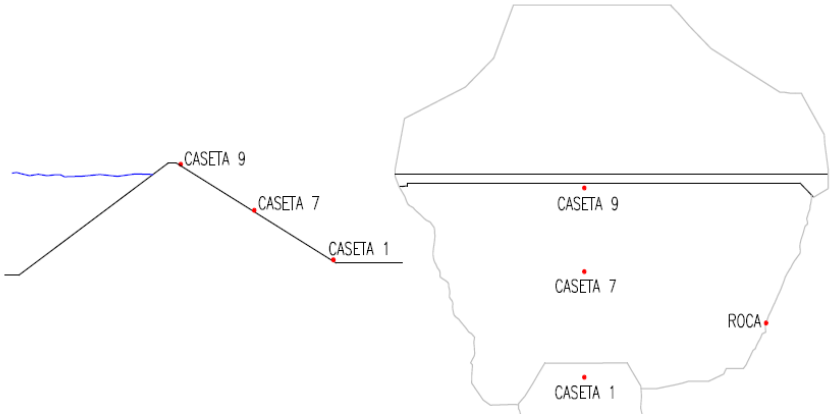


Figura 8: a) corte transversal de la presa, b) planta de la presa, los puntos rojos indican la posición de los acelerógrafos. (Calderon & Cordone, 2019).

Por otra parte, Verdugo (2017) analiza los registros de aceleración obtenidos durante el terremoto de Illapel, M_w=8.3, en dos tranques de relaves instrumentadas, “El Torito” y “Tórtolas-Oeste”. Se analizan los períodos naturales empleando el método SSR entre una estación en la cresta del tranque y una estación en un afloramiento rocoso cercano.

En los trabajos recién mencionados, se concluye que los valores resultantes de la frecuencia predominante de los distintos tranques analizados empíricamente concuerdan con la expresión simplificada de un triángulo elástico unidimensional excitado mediante ondas de corte ascendente. En donde la expresión viene dada por la Ecuación (23) y permite obtener la frecuencia predominante de vibración

$$f_0 = \frac{V_s}{2.56 \cdot H} \tag{23}$$

Finalmente, Pastén (2022) a través de los métodos SSR y HVSR, estudia el comportamiento dinámico del tranque de relaves “El Torito”, mediante una red de geófonos de 28 estaciones que se muestra en la Figura 9. Cabe señalar que los registros en el tiempo empleados son los mismos que se procesan en el presente trabajo.

Figura 9: Red de geófonos ubicados en tranque de relaves ’El Torito’. (Pastén et al., 2019)

En dicho trabajo, Pastén et al. plantean que la expresión para determinar la frecuencia predominante de vibración para un tranque de relaves no es la de un triángulo elástico, sino que es la de un depósito de suelo horizontal, como se muestra en la Figura 3,pero haciendo referencia a una sola capa, no a estratos de multicapa.

La expresión para determinar la frecuencia predominante de vibración para un estrato de suelo horizontal viene dada por la Ecuación (24).

$$f_0 = \frac{V_s}{4 \cdot H} \tag{24}$$

2.6.2 Uso de SSR y HVSR en tranques de relaves.

El uso de los métodos SSR y HVSR para obtener frecuencias de vibración predominante en presas ha sido empleado por Pastén (2019, 2022), en el tranque de relaves “El Torito”. En ambos trabajos se busca obtener frecuencias predominantes de vibración para observar el comportamiento dinámico de la presa.

Del 29 de agosto hasta el 26 de septiembre del 2018, tiempo en el cual fue instalada la red de geófonos, se registraron 11 sismos entre 3.9 y 5.8 Mw. Por lo que se empleó el método SSR para observar el comportamiento de la presa bajo cargas sísmicas y luego comparar resultados obtenidos mediante el método HVSR, en donde fue aplicado para ventanas de ruido ambiental y para sismos.

Pastén et al. (2019) emplea la sección transversal A-A’, que se muestra en la Figura 10. En donde la estación de referencia corresponde a la “T06” y la estación de estudio a la “T08”. Mientras que para el análisis HVSR se incluye la estación “T07”, que se encuentra en el talud del tranque.

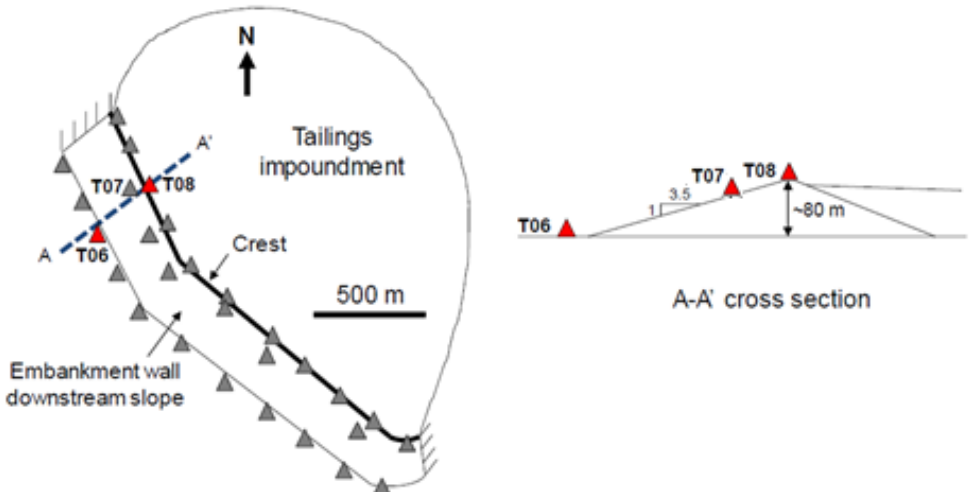


Figura 10: Vista en planta de tranque de relaves “El Torito” y sección transversal A-A’. (Pastén et al., 2019)

Los resultados para el método SSR se muestran en la Figura 11. Mientras que los resultados para el método HVSR aplicado a ruido ambiente son mostrados en la Figura 12.

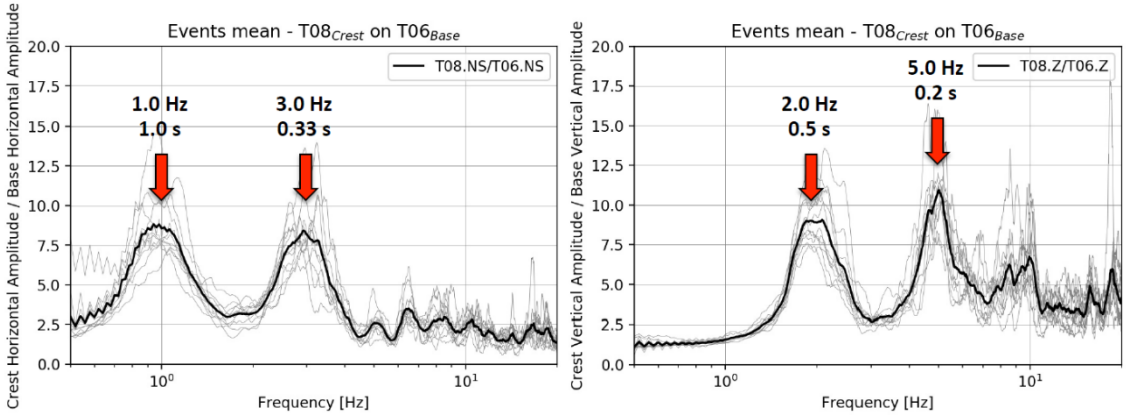


Figura 11: Resultados de método SSR, sección transversal A-A'.

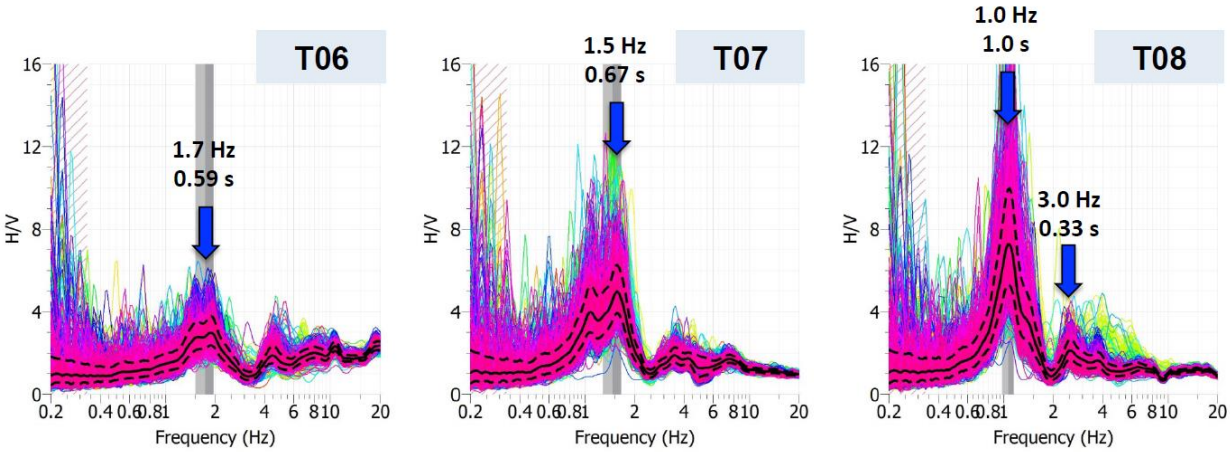


Figura 12: Resultados HVSR sección A-A' (Pastén et al., 2019)

Finalmente, los resultados del método HVSR aplicado a sismos, se muestra en la Figura 13.

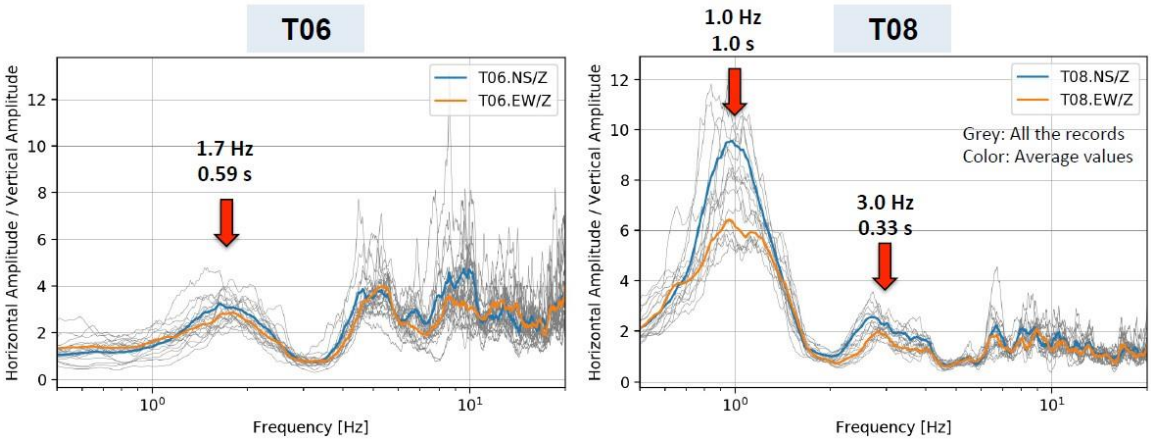


Figura 13: Resultados HVSR sísmico sección transversal A-A'(Pastén et al., 2019).

Los resultados que se obtuvieron fueron los siguientes:

a) El método HVSR y el SSR son consistentes en cuanto a la obtención de frecuencia predominante de vibración y los factores de amplificación.
b) La frecuencia de vibración cambia a lo largo de la sección transversal aguas abajo.
c) La frecuencia predominante puede utilizarse como índice para calibrar los modelos numéricos que a menudo se desarrollan para determinar respuesta sísmica de la geo-estructura.
d) La frecuencia predominante de vibración del tranque es de 1[Hz] aproximadamente.
e) El suelo de fundación es más rígido que el material que se emplea para la construcción de la geo-estructura.
f) El método HVSR identifica solo el primer modo de vibración de la estructura, mientras que el SSR identifica 2.
g) Se detectan ligeras no linealidades en el empleo de HVSR en sismos.

A su vez, Pastén et al. (2022), estudiaron la variación de la frecuencia predominante de vibración a través del eje longitudinal de la presa. Para ello, empleó el método HVSR para distintas estaciones, usando así registros de las estaciones "T04", "T05", "T07" y "T14". Uno de los resultados de ese estudio se puede observar en la Figura 14.

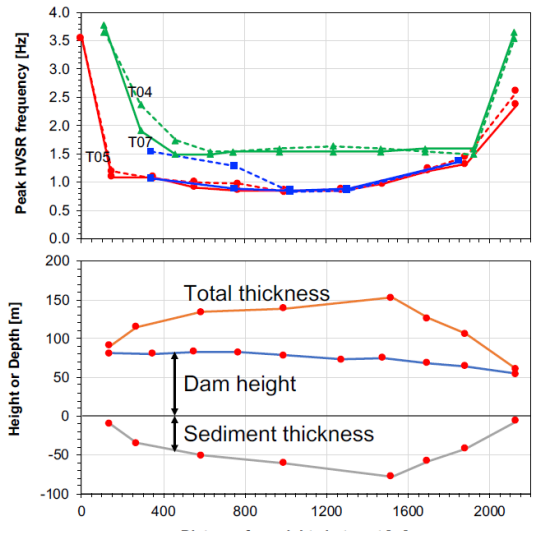


Figura 14: Variación de frecuencia predominante de vibración en relación con la distancia con el estribo derecho del tranque. (Pastén et al., 2020)

Lo que se expone en la Figura 14 es que la frecuencia predominante de vibración cambia con la altura del tranque, además la frecuencia aumenta hacia los apoyos donde el espesor del relleno es menor.

2.7 Módulos de funciones empleados para funcionamiento de interfaz gráfica

Para el desarrollo del trabajo, se implementó el lenguaje Python, en el entorno de programación Jupyter. El trabajo corresponde a una interfaz gráfica, en la cual se realizan múltiples procesamientos, por lo que es necesaria la implementación de distintos módulos, los cuales poseen funciones ya programadas (como la transformada rápida de Fourier, por ejemplo). Dichos módulos se explican en la siguiente subsección.

- Modulo Os: Este módulo provee una manera versátil de usar funcionalidades dependientes del Sistema Operativo como crear una carpeta, listar contenidos de una carpeta, conocer acerca de un proceso, finalizar un proceso, etc.
- Módulo NumPy: Es el paquete fundamental para la computación científica en Python. Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados y un surtido de rutinas para realizar operaciones rápidas con matrices, incluyendo operaciones matemáticas, lógicas, de manipulación de formas, de ordenación, de selección, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria, entre otras cosas.
- Módulo Obspy: Código abierto dedicado a proporcionar un marco de trabajo en Python para el procesamiento de datos sismológicos. Proporciona analizadores

para formatos de archivo comunes y rutinas de procesamiento de señales sismológicas que permiten la manipulación de series temporales sismológicas, tiene como objetivo facilitar el desarrollo rápido de aplicaciones para la sismología.

- Módulo Scipy: Proporciona algoritmos para la optimización, la integración, la interpolación, los problemas de valores propios, las ecuaciones algebraicas, las ecuaciones diferenciales, la estadística y entre otras cosas. Extiende NumPy proporcionando herramientas adicionales para el cálculo de matrices y proporciona estructuras de datos especializadas, como matrices dispersas y árboles k-dimensionales.
- Módulo Matplotlib: Matplotlib es una completa biblioteca para crear visualizaciones estáticas, animadas e interactivas en Python.
- Módulo Tkinter: Proporciona un conjunto de herramientas robustas e independientesde la plataforma para administrar ventanas. Tkinter es un conjunto de envoltorios que implementan los widgets de Tk como clases de Python. Las principales virtudes de tkinter son que es rápido, y que normalmente viene incluido con Python. Aunque su documentación estándar es débil, hay buen material disponible, que incluye: referencias,tutoriales, un libro y otros.
- Módulo PIL: Es una librería gratuita que permite la edición de imágenes directamente desde Python. Soporta una variedad de formatos, incluidos los más utilizados como GIF, JPEG y PNG. Una gran parte del código está escrito en C, por cuestiones de rendimiento.

Capítulo 3
Metodología

3.1 Tarea I: Revisión de antecedentes

La Tarea I consistió en la revisión de antecedentes y archivos entregados por parte del profesor guía. Dentro de estos antecedentes se pueden encontrar scripts desarrollados por alumnos para el procesamiento de data para trabajos previos, un archivo Excel y las distintas trazas de las estaciones.

Los scripts desarrollados contienen en parte el código empleado para el procesamiento de señales para los métodos HVSR y SSR, pero debido al objetivo final estos debieron ser adaptados a las necesidades del presente trabajo.

Por otra parte, se revisó un documento Excel en donde se encuentran 9 sismos previamente identificados por las 28 estaciones (ver Figura 9) durante la campaña geofísica realizada el año 2018. Un sismo previamente identificado, para efectos del presente trabajo, es aquel que es identificado después de la ocurrencia de este y no inmediatamente. Además, sus características, como fecha, hora, coordenadas, distancia epicentral, etc. ya son conocidas y están guardadas en un archivo Excel.

El archivo anterior fue modificado acorde a las necesidades del presente trabajo, y se dio origen a otro archivo con las características que se muestran en la Figura 15.

Event	Time	Latitude [°]	Longitude [°]	Depth [km]	Magnitude type	Magnitude	Epicentral dist [km]	Hipocentral dist[km]	Duration [s]	Time to stations [s]	Arrival time to stations [s]
1	2018-08-31T10:26:44	-31.807	-70.7508	106.6	mb	4.3	74.5821	130.1002	120	26.02	10:27:00
2	2018-08-31T13:25:01	-32.2062	-70.4787	96.26	Mww	5	71.2527	119.762	105	23.95	13:25:18
3	2018-09-07T02:39:17	-28.892	-69.9535	92.4	Mww	5.8	407.7849	418.1224	180	83.62	2:40:12
4	2018-09-07T23:12:47	-30.5238	-69.7354	106.14	Mwr	4.9	255.7664	276.9155	156	55.38	23:13:24
5	2018-09-10T08:24:11	-28.7518	-71.8113	10	Mww	5.1	410.8944	411.0161	160	82.2	8:25:10
6	2018-09-12T19:31:14	-31.529	-72.008	34.56	ML	4	138.7729	143.0115	80	28.6	19:31:34
7	2018-09-14T18:15:11	-32.5454	-71.6043	32.47	Mwr	4.5	59.251	67.5647	42	13.51	18:15:18
8	2018-09-14T18:46:51	-32.5629	-71.607	21.25	mb	4.1	60.1508	63.7941	47	12.76	18:46:58
9	2018-09-18T23:50:10	-30.8741	-71.435	48.93	Mwr	3.9	171.749	178.583	80	35.72	23:50:40

Figura 15: Tabla de sismos modificada.

Es de suma importancia tener en cuenta que la última columna fue creada en función de los registros de la estación T14, en donde se observó el tiempo en el cual esta estación comienza a registrar el sismo. El tiempo de inicio aplica para todas las estaciones. Se emplea esta estación como referencia puesto que es la que se encuentra en la cresta de la mitad del tranque de relaves en su componente longitudinal, por lo que es más conservador considerar este tiempo de llegada del sismo.

Luego, se procedió a revisar las trazas diarias de cada estación. Estas corresponden a archivos en formato *.mseed,* Un archivo en formato *.mseed* es un subconjunto del formato *.seed* standard que es usado para datos de series de tiempo. Se incluye metadata muy limitada para la serie temporal en miniSEED más allá de la identificación de la serie temporal e indicadores simples de estado. En particular, no se incluyen las coordenadas geográficas, la información de respuesta/escala y otra información necesaria para interpretar los valores de los datos.

Las series de tiempo se almacenan como registros de datos de longitud fija generalmente independientes, cada uno de los cuales contiene un pequeño segmento de valores de series contiguas. Se requiere un lector de miniSEED para reconstruir series de tiempo contiguas más largas a partir de los segmentos del registro de datos. Las longitudes de registro comunes son 512 bytes (para transmisiones en tiempo real) y 4096 bytes (para archivo); otras longitudes de registro se utilizan para escenarios especiales.

Un archivo de miniSEED es una concatenación de registros de datos. Dependiendo de las capacidades del lector previsto, los registros de datos para múltiples canales de datos pueden multiplexarse juntos.

Dicho esto, los archivos con los que trabaja el código corresponden a trazas diarias en 3 componentes. Cada componente es un archivo independiente, esto se diferencia en los últimos 4 caracteres del archivo. Si el archivo termina en *.pri0*, corresponde al registro de la componente Z, en caso de terminar en *.pri1*, corresponde a la componente Norte-Sur. Finalmente, en caso de terminar en *.pri2*, corresponde a la componente Este-Oeste.

A modo ejemplo en la Figura 16, se muestran la información de la estación “T06”, con fecha de registro 01/09/2018, componente Z.

```
        network:
        station: c0ALU
       location:
        channel: p0
      starttime: 2018-09-01T00:00:00.000000Z
        endtime: 2018-09-01T23:59:59.995000Z
  sampling_rate: 200.0
          delta: 0.005
           npts: 17280000
          calib: 1.0
        _format: MSEED
          mseed: AttribDict({'dataquality': 'D', 'number_of_records': 9032, 'encoding': 'STEIM1', 'byteorder': '<', 'record_le
ngth': 4096, 'filesize': 36995072})
```

Figura 16: Información del archivo *’c0alu180901000000.pri0’*

En la Figura 16, se puede ver que la estación corresponde a la C0ALU, gracias a eso, el nombre del archivo comienza con estos 5 caracteres. Luego, en el parámetro *starttime*, se tiene la fecha 01/08/2018 a las 00:00:00 hrs, que corresponde al inicio de la toma de datos, luego del nombre de la estaciónen el nombre del archivo se tiene *'180901000000'*. El número 18 corresponde al año,09 al mes y el 01 al día. Los siguientes 6 caracteres corresponden a la hora de inicio. Por lotanto, es muy importante notar que la configuración del nombre del archivo corresponde a una concatenación del nombre de la estación con la fecha en formato UTC. Finalmente, se puede ver que el archivo analizado termina con *.pri0*, por lo que corresponde a la traza diaria registrada en el eje vertical.

Es muy importante tener claro todo lo anterior, ya que la selección de archivos a procesar en el código para la implementación de HVSR y SSR, es gracias al formato que se tienen nombradas las trazas en el tiempo en sus respectivas carpetas.

Por otra parte, siguiendo con la descripción de la Figura 16, el parámetro *sampling_rate* corresponde a los datos por segundo que se registran. El dato *delta* corresponde a la diferencia de tiempo entre un dato y otro. Y el dato *npts* es el número de datos que tiene la traza.

Las trazas diarias deben ser guardadas en carpetas con el nombre de la estación para la ejecución del código. Es de suma importancia que el usuario tenga los registros que quiera analizar para un correcto funcionamiento de la interfaz.

Finalmente, se generó un archivo Excel que contiene una tabla con los pixeles de las coordenadas de cada estación de una imagen con vista en planta del tranque para poder marcar la estación que se procesó (los pixeles son obtenidos de manera manual) y además el ángulo de proyección de cada estación. Este ángulo se obtiene mediante la proyección de la dirección del tranque de relaves con el Norte y el Oeste. Se utiliza para poder realizar un análisis separado de las componentes transversales y longitudinales, en caso de que se desee. Lo anterior, se puede ver esquemáticamente en la Figura 17.

Figura 17: Ángulo proyectado para descomponer la data procesada en sección transversal y longitudinal del tranque de relaves.

El archivo Excel que se generó, se muestra en laFigura 18.

Estacion	Pixel x	Pixel Y	Angulo
T01	440	244	56.49
T02	0	0	0
T03	511	183	56.49
T04	463	290	56.49
T05	530	220	56.49
T06	485	334	56.49
T07	527	275	56.49
T08	553	270	56.49
T09	512	386	56.49
T10	553	334	56.49
T11	577	322	56.49
T12	541	437	56.49
T13	578	384	56.49
T14	608	374	56.49
T15	598	480	34.19
T16	652	430	34.19
T17	655	418	34.19
T18	651	529	34.19
T19	709	493	34.19
T20	716	469	34.19
T21	707	566	34.19
T22	0	0	0
T23	758	505	34.19
T24	757	603	34.19
T25	805	547	34.19
T26	809	645	34.19
T27	826	593	34.19
T28	847	580	34.19
T29	858	667	164.38
T30	0	0	0
T31	891	610	164.38

Figura 18: Tabla para obtener pixel x e y en imagen y ángulo para proyección de componentes.

3.2 Tarea II: Desarrollo de código, procesamiento de señales e interfaz gráfica

Dentro de esta tarea, lo primero que se llevó a cabo fue el desarrollo del código de procesamiento de señales para los métodos SSR y HVSR y luego, se generó la interfaz gráfica interactiva con el usuario. Para ambos métodos el procesamiento de señales es similar, sin embargo, serán explicados de manera independiente.

Para el método SSR, se seleccionan las 2 estaciones y la ventana de tiempo que se desea trabajar. Luego, se cortan las trazas sísmicas a través de la función *.trim* de Python dejando una ventana de 180 [seg] de análisis, esto se decidió ya que se analizaron las trazas de los registros de los distintos sismos para la estación “T14” y se concluyó que la duración del sismo más largo correspondía a ese valor. Se procede a corregir la línea base de la señal a través de la función

.detrend, luego se filtran las trazas a través de un filtro pasa-banda, con una frecuencia mínima= 0.05[Hz] y una frecuencia máxima= 25[Hz]. El filtro es de orden 4. Luego, se multiplican las ventanas a trabajar por una ventana tukey del 5% y se procede a descomponer la traza en función del ángulo proyectado que tenga la estación, obtenido de la tabla que se muestra en la Figura 18. Se aplica la transformada rápida de Fourier de la traza descompuesta y luego se suaviza mediante el suavizado de Konno y Ohmachi con un factor b=40. Finalmente, en caso de que se quiera obtener el espectro de las componentes combinadas se emplea el método de la media cuadrática. En caso de que se quieran analizar independientemente no se modifican. Así con las señales procesadas de ambas estaciones se procede a aplicar el método SSR, en donde se divide el espectro que se desea estudiar por el espectro de la estación de referencia, como lo indica la Fórmula (14).

En el caso que se deseen analizar distintos sismos mediante el mismo método se promedian los resultados SSR de los sismos.

Por otra parte, la metodología que se aplicó para el procesamiento HVSR es similar en cuanto a procesamiento de señales. La diferencia radica en que las ventanas de tiempo de procesamiento HVSR son de 60 [seg], es decir, si se desea procesar 600 [seg] se tendrán 10 ventanas. Luego, cada ventana es filtrada con un filtro pasa-banda, con frecuencia mínima= 0.05[Hz] y una frecuencia máxima= 25[Hz]. El filtro es de orden 4. Así, se multiplican las ventanas a trabajar por una ventana tukey del 5% y se procede a descomponer la traza en función del ángulo proyectado que tenga la estación, obtenido de la tabla que se muestra en la Figura 18. Se aplica la transformada rápida de Fourier de la traza descompuesta y luego se suaviza mediante el suavizado de Konno y Ohmachi con un factor b=40.

Siguiendo, se procede a realizar un análisis estadístico de las distintas ventanas, este análisis estadístico aplicado al análisis HVSR no es directo, es decir, no se obtiene la desviación estándar de las ventanas directamente, ya que se aplica la función logaritmo natural y luego exponencial para que no se tengan resultados menores que 0. Esto se realiza para garantizar la física del problema, y tener resultados más conservadores, puesto que la distribución normal es la que más incertidumbre entrega. Dicho esto, la metodología que se siguió para el análisis estadístico fue la siguiente, considerando σ como desviación estándar y μ como el promedio:

1) Se obtiene HVSR de cada ventana.
2) Se aplica la función Ln() a cada una de las ventanas.
3) Se obtiene σ del Ln(H/V) (es decir, de las ventanas procesadas).
4) Se obtiene μ de las ventanas Ln(H/V).
5) Se suma y resta σ a μ.
6) Se aplica la función exp() a la suma y resta de los valores.

En la Figura 19 se muestra esquemáticamente el proceso recién señalado.

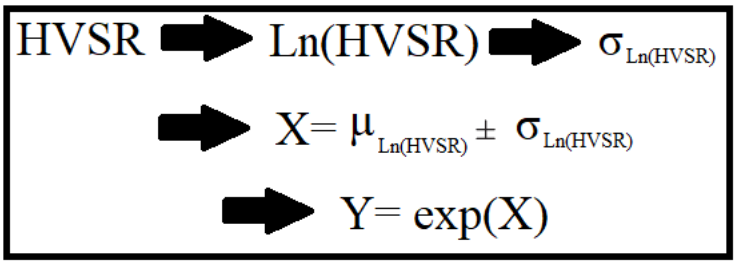


Figura 19: Esquema de análisis estadístico HVSR.

Por otra parte, la segunda parte de la Tarea II consistió en el desarrollo de la interfaz gráfica. Para esto lo primero que se hizo fue establecer la funcionalidad de esta junto con el profesor guía. Luego, se procedió a trabajar en ella consultando bibliotecas, foros, videos, etc. que se encuentran en distintos sitios de internet.

La primera interfaz que se generó fue la del método HVSR, luego en base a los resultados obtenidos en cuanto a programación de la plataforma, se desarrolló la interfaz del método SSR. Cabe señalar que a medida que se avanzaba en la funcionalidad de esta se iba retroalimentando por parte del profesor guía.

3.3 Tarea III: Validación de resultados

Para la validación de resultados se procedió a comparar los gráficos que se obtuvieron mediante el código desarrollado y los gráficos obtenidos mediante el software Geopsy, el cual desarrolla, distribuye y mantiene software de código abierto para la investigación y las aplicaciones geofísicas.

Para la validación de resultados se compararon los resultados obtenidos mediante el software Geopsy y los resultados obtenidos mediante la interfaz desarrollada. De esta forma, para la validación de los resultados del método HVSR y SSR se seleccionaron 6 estaciones a lo largo del tranque, 3 ubicadas aguas abajo y 3 en el coronamiento. Para el análisis HVSR se seleccionaron trazas de tiempo aleatorias en donde se procese ruido ambiental y para el análisis SSR se generaron 3 perfiles transversales en donde se procesaron 2 sismos. El detalle de esto será explicado en el Capítulo 6.

Capítulo 4
Manual de uso

A continuación, se detallan los pasos a seguir para el correcto funcionamiento y uso de la interfaz gráfica desarrollada.

4.1 Instalación de software y módulos

La instalación de software y módulos son los primeros pasos para poder usar la interfaz gráfica. Dicho esto, se debe seguir el siguiente listado:

1) Descargar e instalar el programa Anaconda, a través de la consola de este programa se procederá a instalar los distintos módulos para el procesamiento de señales y el desarrollo de la interfaz gráfica.
2) Descargar e instalar Jupyter Notebook, el cual será el entorno de programación en el cual se ejecuta el código.
3) Instalar los módulos mencionados en la sección 2.7.

La instalación de estos módulos se debe llevar a cabo en la consola de Anaconda, para ello, una vez instalado el software, se debe abrir "Anaconda Prompt", que se encuentra en la barra de inicio, como se ve en la Figura 20.

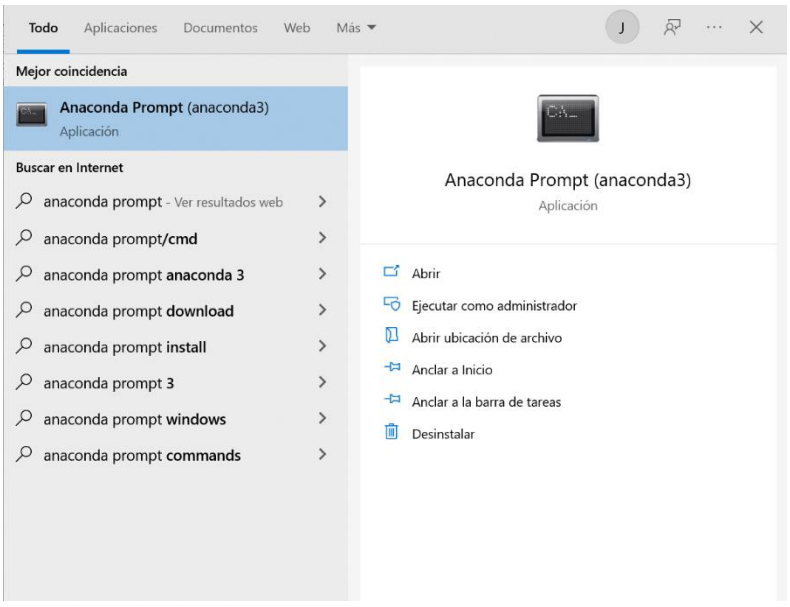


Figura 20: Navegador para abrir consola de Anaconda

Una vez abierta la consola, se desplegará una ventana como la que se observa en la Figura 21.

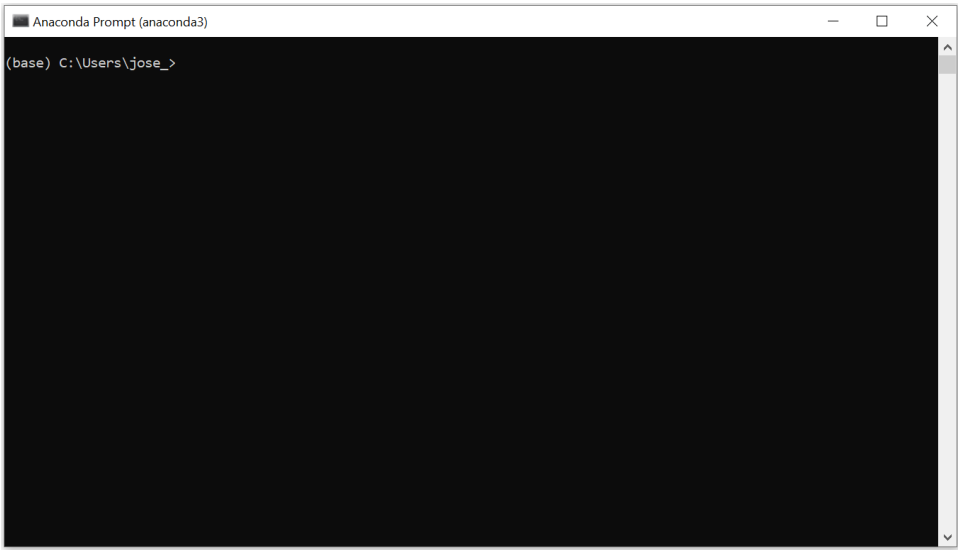


Figura 21: Consola de Anaconda.

Luego, como se mencionó, se deben instalar los módulos detallados en la sección 2.7, para ello se debe escribir los siguientes comandos, acorde al módulo que se desea instalar:

- Para el módulo “Os”, se debe escribir: “conda install -c jmcmurray os”.
- Para el módulo “NumPy”, “conda install numpy”.
- Para el módulo “Obspy”, “conda install -c conda-forge obspy”.
- Para el módulo “Scipy”, “conda install -c anaconda scipy”.
- Para el módulo Matplotlib, “conda install -c conda-forge matplotlib”.
- Para el módulo “Tkinter”, “conda install -c anaconda tk”.
- Para el módulo PIL, “conda install -c anaconda pillow”.

A modo ejemplo, en la Figura 22, se muestra la instalación del módulo “Os”, en donde para finalizar el proceso se debe aceptar la instalación ingresando el carácter “y”.

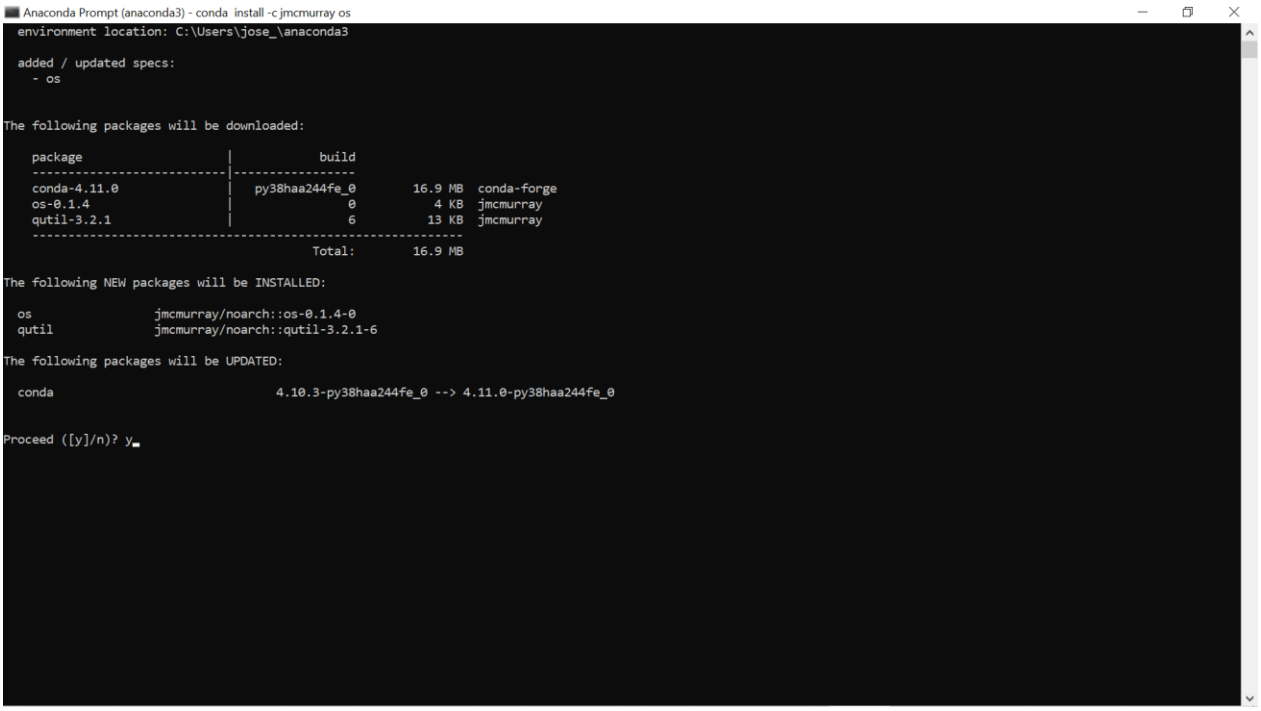


Figura 22: Instalación de módulo “OS”.

4.2 Archivos con los que trabaja el código

Los archivos con los que trabaja el código corresponden a: trazas diarias, 2 archivos Excel, imágenes de ornamento y finalmente una imagen para ubicar la o las estaciones que fueron procesadas en una vista en planta del tranque de relaves “El Torito”.

Las estaciones con sus respectivas trazas diarias, correspondiente a los archivos en formato *.mseed* como se señaló anteriormente, deben estar organizadas en carpetas individuales, es decir, para cada estación debe existir una carpeta con su nombre “TXX” (con XX el número de la estación), y en ella encontrarse los registros en el tiempo. De esta forma, la organización de la data se muestra en la Figura 23 y Figura 24.

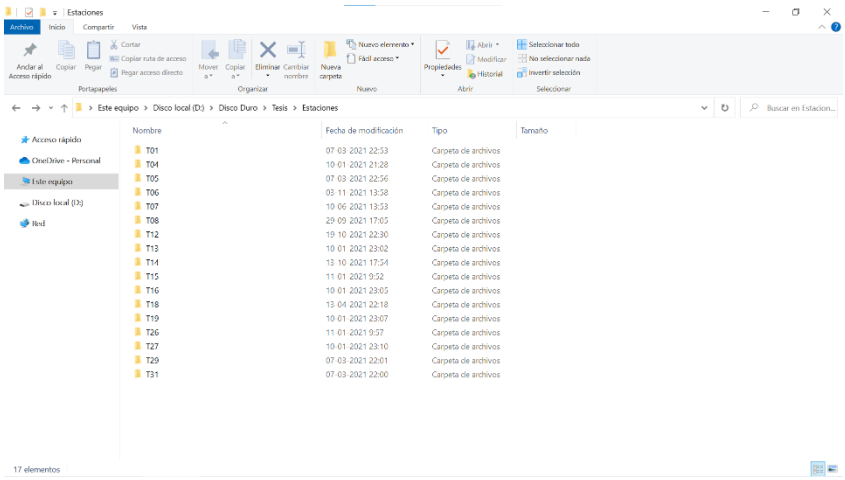


Figura 23: Carpeta con registros de distintas estaciones.

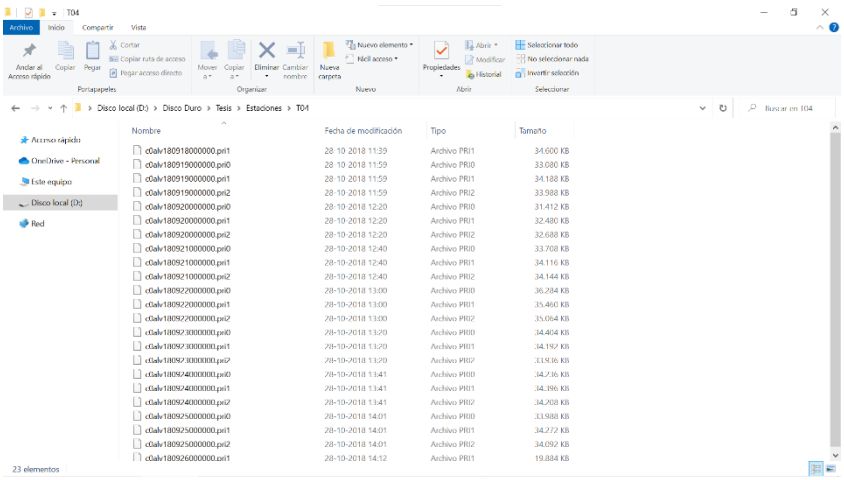


Figura 24: Carpeta de estación “T04” con distintos registros.

Cabe señalar que por capacidad de espacio en el equipo en donde se desarrolló el presente trabajo no se cuenta con la totalidad de las estaciones y trazas obtenidas en la campaña geofísica.

Los 2 archivos Excel con los que trabaja el código son los que se muestran en la Figura 15 y Figura 18. Que llevan por nombre “Sismos_importantes_final_modificado.csv” y “Coordenadas Estaciones en foto empleada.csv”, respectivamente.

Las imágenes de ornamento no serán detalladas ya que no tienen mayor relevancia para el trabajo desarrollado. Finalmente, dependiendo del método que se desea emplear (SSR o HVSR), se emplea una imagen para identificar la(s) estación(es) de las cuales se obtuvieron los registros procesados.

4.3 Importar módulos, definición de variables globales y definición de ruta de archivos.

Los módulos que fueron señalados e instalados en la sección 4.1, deben ser importados para que así el código pueda emplear las funciones predeterminadas que vienen con estos. De esta forma la importación de módulos en el código se observa en la Figura 25.

```
#Importación de módulos.
%matplotlib notebook
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
#Importar modulo os:
import os
#Importar modulo numpy como np:
import numpy as np
#Importar read de modulo obspy:
from obspy import read
#Importar signal de modulo obspy:
from obspy import signal
#Importar signal de modulo scipy:
from scipy import signal
#Importar modulo matplotlib.pyplot como plt:
import matplotlib.pyplot as plt
#Importar modulo tkinter:
from tkinter import *
#Importar función de suavizado de Konno & Ohmachi:
from obspy.signal.konnoohmachismoothing import konno_ohmachi_smoothing
#Se importa Figure para poder graficar datos:
from matplotlib.figure import Figure
#Se importa Image e ImageTk para poder mostrar figuras cargadas:
from PIL import Image, ImageTk
```

Figura 25: Importación de módulos.

Luego, el código trabaja con variables globales que deben ser definidas para el procesamiento de señales. Estas se muestran en la Figura 26.

```
ventana=60                  #segundos ventanas de analisis HVSR.
freq_last = int(20.0)       #ultima frecuencia a considerar.
tiempo_analisis_SSR= 180    #duración de analisis SSR.
frec_min_filtrado=0.05      #se determina la frecuencia mínima de filtrado[Hz].
frec_max_filtrado=25        #se determina la frecuencia máxima de filtrado[Hz].
ventana_tukey='5%'          # % con el que se trabaja en la ventana tukey.
bandwith_KyO=40             # valor de parámetro b para suaviazado de Konno y Ohmachi.
```

Figura 26: Definición de variables globales

Finalmente, se debe indicar la ruta de los archivos Excel y de la carpeta en donde se encuentren las carpetas de las estaciones con sus respectivos registros sísmicos. Esto se muestra en la Figura 27.

```
#Se procede a definir rutas de archivos externos
archivo_coordenadas='C:/Users/jose_/Script Tesis/Scripts desarrollados/Coordenadas Estaciones en foto empleada.csv'
archivo_eventos='D:/Disco Duro/Tesis/Sismos_Importantes_Finales/Sismos_importantes_final_modificado.csv'
path='D:/Disco Duro/Tesis/Estaciones/'
```

Figura 27: Ruta de archivos Excel y carpeta con archivos de las estaciones.

4.4 Funciones programadas

En la siguiente sección se muestran las funciones programadas para el método HVSR y SSR.

4.4.1 Funciones para HVSR

La Tabla 1 muestra las funciones programadas para calcular HVSR, indicando los parámetros de entrada (inputs) y los outputs de cada función. Dichas fórmulas se adjuntan en el Anexo A.

Tabla 1: Funciones programadas para método HVSR

Función	Inputs	Outputs
H_on_V_ambiental	-Estación. -Fecha de análisis. -Duración de análisis. -Hora de inicio del análisis. -Variable para definir componentes horizontales independientes o combinadas. -Variables globales mostradas en la Figura 26.	Diccionario que contiene: -Espectro con resultado de análisis HVSR. -Matriz con valores del gráfico. -Las 3 trazas que fueron procesadas.
crear_txt_HVSR	-Matriz obtenida de función H_on_V_ambiental.	-Archivo en formato .txt con los valores de la matriz.
H_on_V_aux	-Diccionario de la función H_on_V_ambiental.	-Ventana que muestra: -Espectro con resultados estadísticos de análisis HVSR. -Trazas que fueron procesadas. -Botones que permiten: -Guardar gráfico como archivo en formato png. -Ver ubicación de la estación en el tranque de relaves. -Guardar resultados en formato .txt
ventana_estacion_HV_mapa	-Coordenadas en pixeles de la estación en estudio proveniente de archivo Excel. -Imagen con vista en planta del tranque de relaves.	-Ventana que muestra la ubicación en el mapa de la estación en el tranque de relaves.
InfoProcesamiento_HVSR	-Variables que son mostradas en la Figura 26.	-Ventana que muestra las variables que son señaladas en la Figura 26.

4.4.2 Funciones para el método SSR

Las funciones programadas para el método SSR se muestran en la Tabla 2.

Tabla 2: Funciones programadas para SSR.

Función	Inputs	Outputs
ventana_todos_sismos	-Archivo Excel de la Figura 15.	-Ventana que muestra la información de los sismos en el archivo Excel.
ventana_sismos_filtrados	-Archivo Excel de la Figura 15. -Número correspondiente a M_w mínimo.	-Ventana que muestra los sismos con M_w mayor al indicado como input.
sismos_en_comun	-Nombre de 2 estaciones.	-Ventana que muestra sismos en común que registraron las 2 estaciones.
sismos_en_comun_filtrados	-Nombre de 2 estaciones. -Número correspondiente a M_w mínimo.	-Ventana que muestra sismos en común que registraron las 2 estaciones con M_w mayor al indicado como input. -Checkbuttons que permiten marcar que sismo(s) se desea(n) procesar mediante método SSR.
TransFourier	-Estación. -Tiempo de análisis definido para método SSR, definido en la Figura 26. -Hora y fecha de inicio del sismo, obtenida del archivo Excel mostrado en la Figura 15.	-Estación. -Matriz con resultados de transformada de Fourier de componentes combinadas e independientes. -Las 3 trazas que fueron procesadas.
SSR_sismos	-Estaciones para emplear método SSR. -Fecha y hora de sismo(s) obtenido(s) del archivo Excel mostrado en la Figura 15. -Matriz con resultados de transformada de Fourier de componentes combinadas e independientes de ambas estaciones. -Las 3 trazas que fueron procesadas.	-Ventana que muestra: -Espectro con resultados de análisis SSR de las 3 componentes, en caso de ser más de 1 sismo analizado muestra todos y resalta el promedio. -Botones que permiten: -Guardar gráfico como archivo en formato png. -Ver ubicación de las estaciones en el tranque de relaves. -Guardar resultados en formato .txt -Ver trazas de las 3 componentes que fueron procesadas.
InfoProcesamiento_SSR	-Variables que son mostradas en la, no incluye la duración de sub-ventanas, propio de HVSR.	-Ventana que muestra las variables que son mostradas en la Figura 26.

Tabla 2: Funciones programadas para SSR

Función	Input	Output
ventana_estacion_SSR_mapa	-Coordenadas en pixeles de la estación en estudio proveniente de archivo Excel. -Imagen con vista en planta del tranque de relaves.	-Ventana que muestra la ubicación en el mapa de las estaciones en el tranque de relaves.
crear_txt_SSR	-Matriz de resultados obtenida de función SSR_sismos.	-Guardar resultados en formato .txt
ver_trazas_Z_SSR	-Trazas en dirección Z procesadas mediante SSR de las 2 estaciones.	-Ventana donde se pueden observar las trazas.
ver_trazas_NS_SSR	-Trazas en dirección NS procesadas mediante SSR de las 2 estaciones.	-Ventana donde se pueden observar las trazas.
ver_trazas_EW_SSR	-Trazas en dirección EW procesadas mediante SSR de las 2 estaciones.	-Ventana donde se pueden observar las trazas.

4.4.3 Funciones programadas para métodos HVSR y SSR

A continuación, la Tabla 3 muestra una función programada para ambos métodos.

Tabla 3: Funciones programadas para ambos métodos.

Función	Input	Output
ubicacion_en_el_mapa	-Archivo Excel que se muestra en la Figura 18 -Estación	-Pixel X y pixel Y de la estación en la Figura 9.
AcercaDe	-Ninguno	-Ventana que muestra información sobre el trabajo desarrollado.
AyudaMenu	-Ninguno	-Ventana que muestra contacto de estudiante con el fin de aclarar posibles consultas.

4.4.4 Funciones programadas para interfaz interactiva

Tabla 4: Funciones que corren interfaces gráficas de cada método.

Función	Input	Output
ventana_SSR	-Ninguno	-Ventana interactiva en donde se puede operar método SSR.
Ventana_HV	-Ninguno	-Ventana interactiva en donde se puede operar método HVSR.

4.5 Uso de interfaz gráfica

En la siguiente sección se muestra la interfaz gráfica desarrollada y como usarla, además se muestran esquemas del uso de la interfaz.

Al ejecutar el script que contiene todas las funciones, se abre la ventana que permite la selección entre el procesamiento HVSR y SSR, la ventana se muestra en la Figura 28.

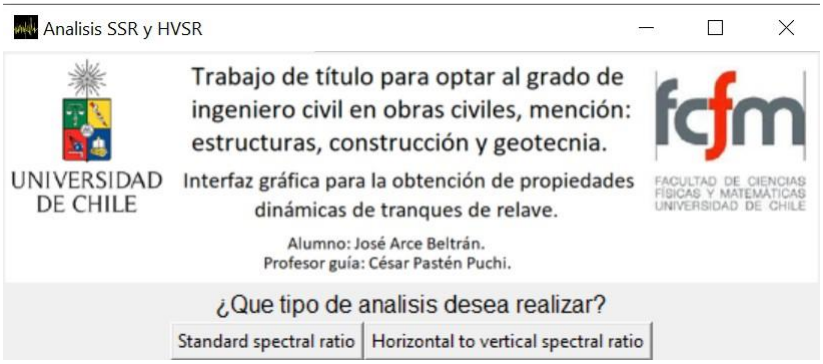


Figura 28: Ventana inicial de la interfaz.

4.5.1 Interfaz de cálculo HVSR

Al clickear el botón “Horizontal to vertical spectral ratio”, se abre la ventana en donde se ingresan los datos sobre el procesamiento que se quiere hacer, específicamente: estación, fecha de análisis, hora de inicio del análisis, duración del análisis, y 2 checkbuttons los cuales permiten elegir entre análisis de las componentes horizontales combinadas o de cada componente independiente (cabe señalar que las componentes corresponden a las transversales y longitudinales), como se muestra en la Figura 17. La ventana emergente que corresponde al procesamiento HVSR se muestra en la Figura 29.

Figura 29: Ventana para procesamiento HVSR.

Al ingresar los datos de la Tabla 5, como ejemplo, se puede observar la Figura 30, cabe señalar que los resultados mostrados en las Figura 31 y Figura 32 no son resultados finales.

Tabla 5: Datos ingresados para procesamiento HVSR

Estación a analizar	T07
Fecha de análisis	08/09/2018
Hora de inicio del análisis	00:00:00
Duración del análisis[seg]	600
Componentes horizontales	Independientes

Figura 30: Ventana para procesamiento HVSR con datos ingresados.

Al ejecutar el botón “Correr”, se abre otra ventana con los resultados del procesamiento HVSR y las trazas sísmicas procesadas de las 3 componentes, como se muestra en la Figura 31.

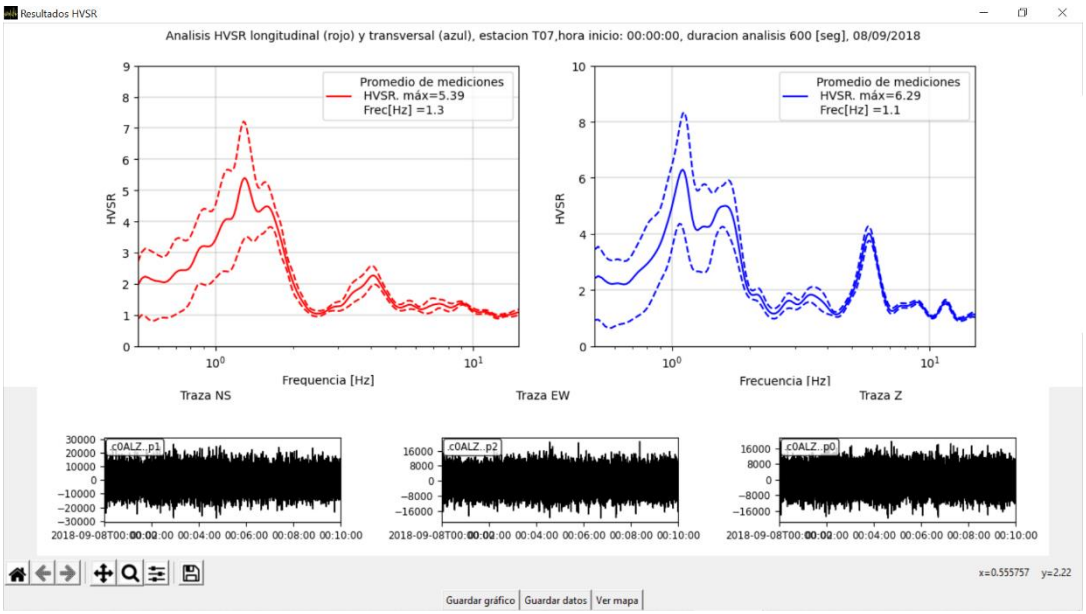


Figura 31: Resultados HVSR con datos de Tabla 5.

En caso de que se quiera realizar un análisis de las componentes combinadas, el resultado de este se muestra en la Figura 32.

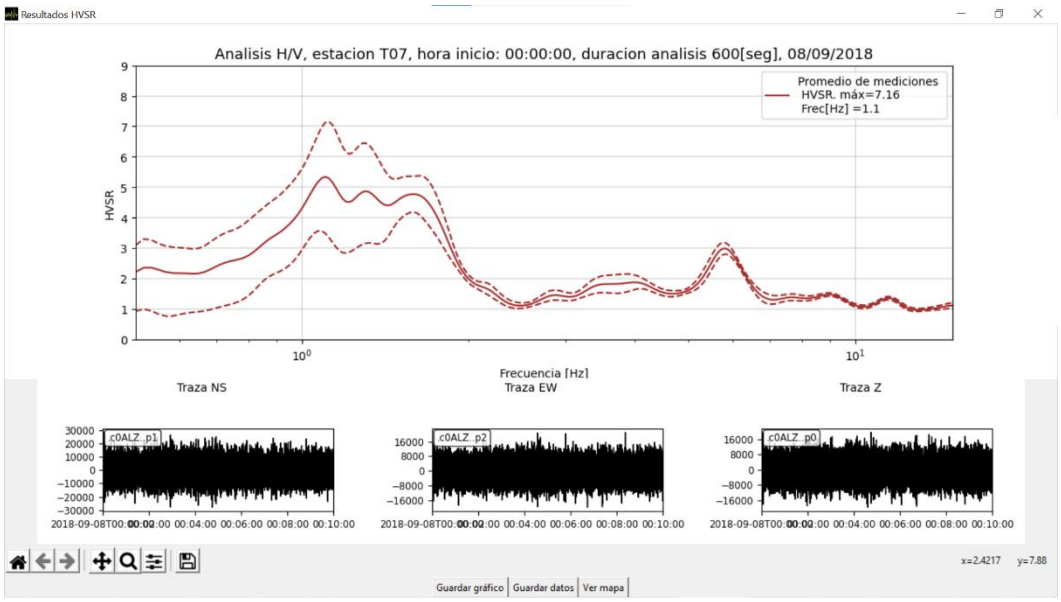


Figura 32: Resultados HVSR con componentes combinadas.

En las Figura 31 y Figura 32, en la parte inferior de la ventana de resultados se pueden observar 3 botones, para efectos prácticos y a modo ejemplo se utilizará la Figura 32. Dicho esto, los 3 botones se destacan en la Figura 33.

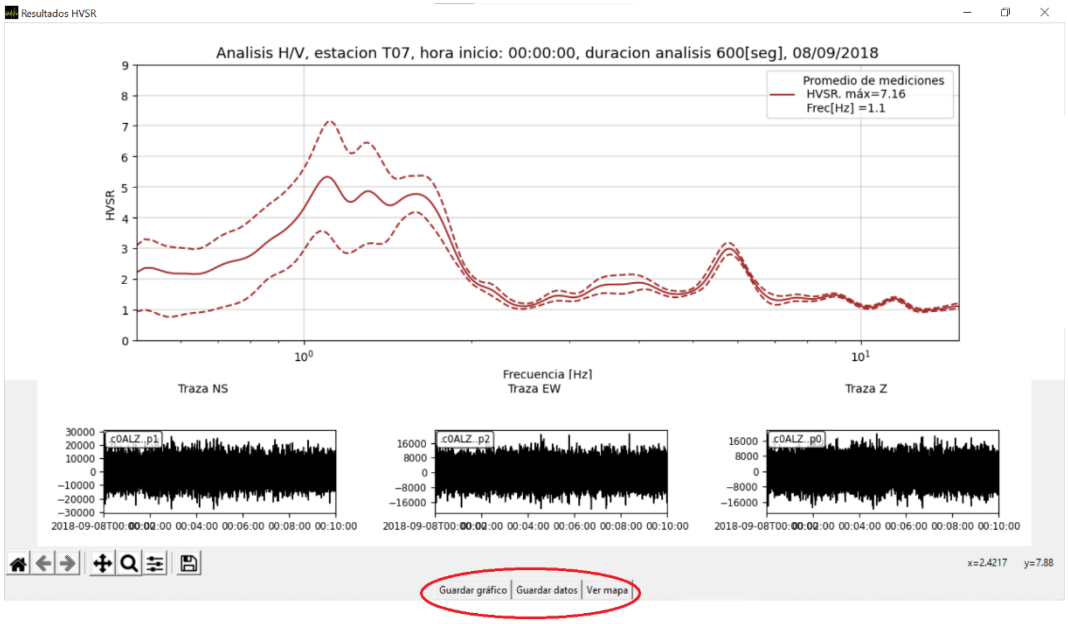


Figura 33: Botones destacados.

De izquierda a derecha, el primer botón correspondiente a “Guardar gráfico”, guarda el (o los) gráficos de resultados en formato *.png*. De esta forma, al presionar el botón se abre la ventana que se muestra en la Figura 34, en donde se puede seleccionar la carpeta en donde se desee guardar la imagen.

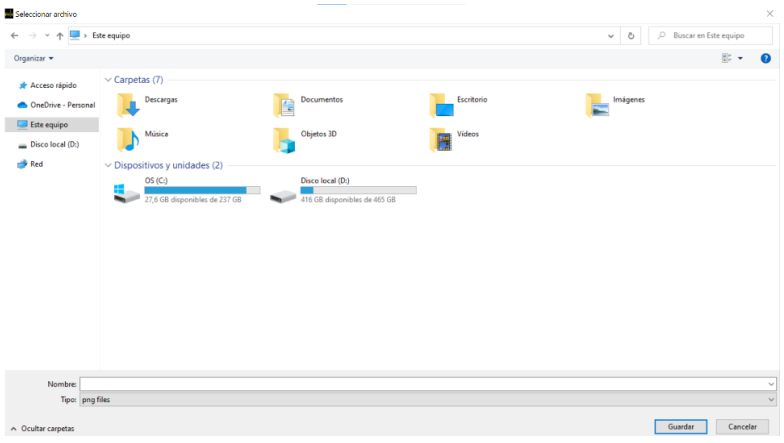


Figura 34: Ventana para seleccionar carpeta donde se desea guardar gráficos de resultados HVSR.

Siguiendo con el segundo botón correspondiente a “Guardar datos”, permite guardar los datos del procesamiento HVSR en un archivo *.txt*, la ventana que se abre para seleccionar la carpeta en donde se desee guardar el archivo la cual es idéntica a la mostrada en la Figura 34, por lo que no se detallará. Por otra parte, el archivo que se guarda corresponde al mostrado en la Figura 35 en donde la primera columna es el vector de frecuencias que se usa para graficar los resultados, la segunda columna es un vector con los resultados de HVSR de las direcciones combinadas, la tercera es la desviación estándar de las direcciones combinadas, correspondiente a la respectiva frecuencia, ya que tal como se señaló en la subsección 3.2 se realiza un análisis estadístico. La cuarta columna corresponde al resultado HVSR de la componente longitudinal del tranque, y la quinta a su análisis estadístico; finalmente la sexta columna responde a los resultados del análisis transversal, mientras que la séptima es la desviación estándar respectiva a la sexta columna.

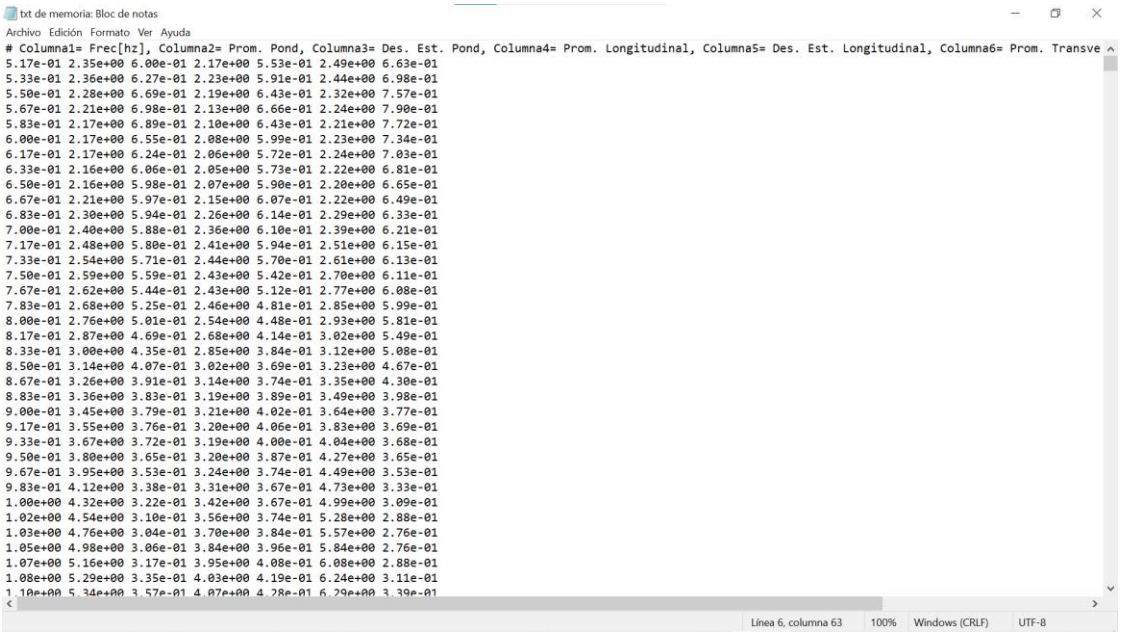


Figura 35: Archivo *.txt* con los resultados del resultado HVSR con datos de la Tabla 5.

Finalmente, el botón que corresponde a “Ver Mapa” muestra la ubicación de la estación de donde se obtuvieron las trazas procesadas. En la Figura 36se puede ver la ubicación en el tranque de relaves de la estación “T07”.

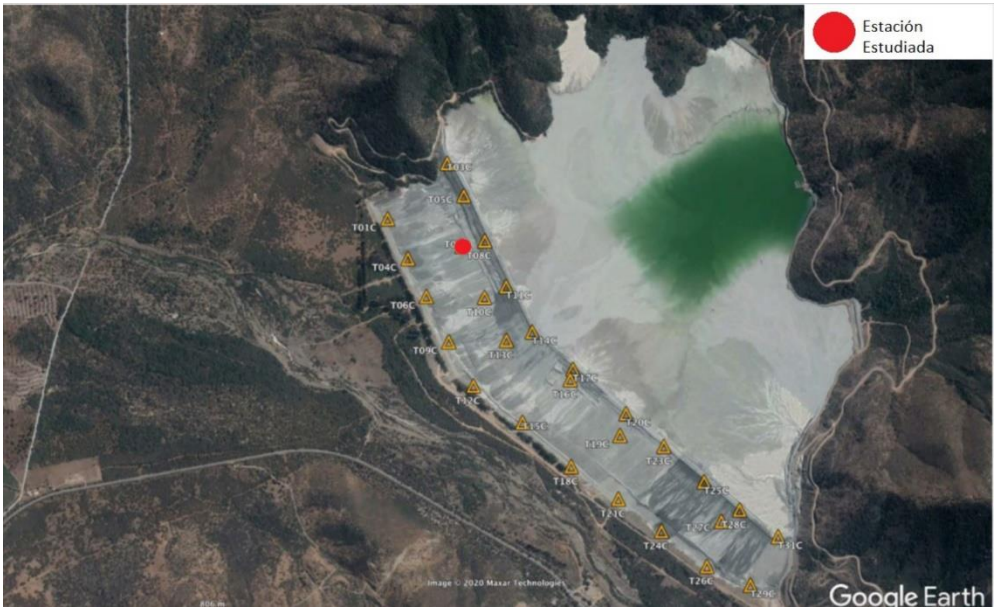


Figura 36: Ubicación de la estación "T07" en el tranque de relaves.

Así, la Figura 37 muestra un esquema del uso de la interfaz de cálculo de HVSR.

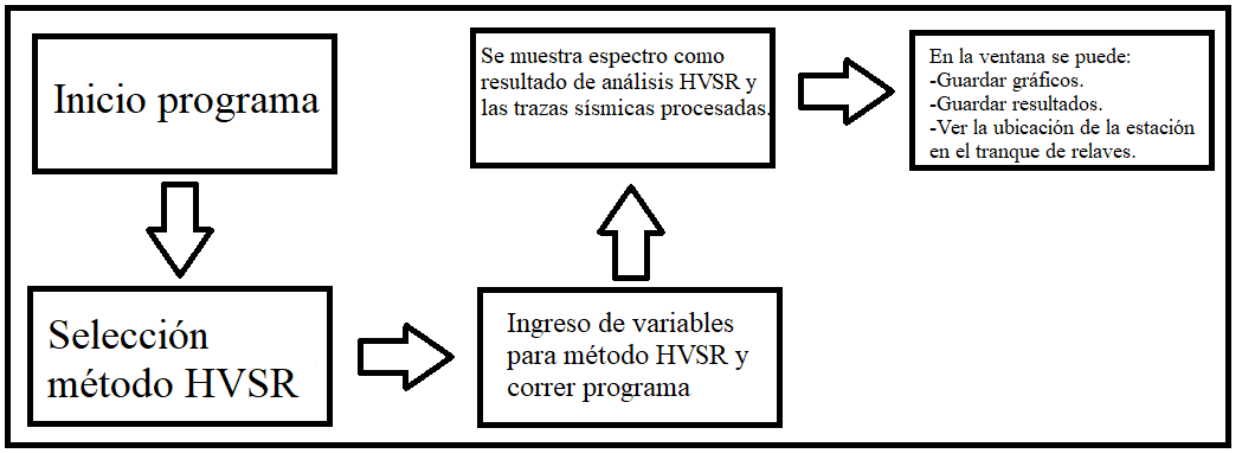


Figura 37: Esquema para utilizar interfaz de cálculo HVSR.

4.5.2 Interfaz de cálculo SSR

Al clickear el botón “Standard spectral ratio”, se abre la ventana en donde se puede ingresar la estación que se desea estudiar, la estación que se desea emplear como referencia y el valor correspondiente a los sismos con M_w mínima que se desean estudiar. Además, en la parte derecha se pueden observar 4 botones, en donde el primero corresponde a “Ver todos los sismos”, el segundo a “Ver sismos con $M_w > M_w$ min”, el tercero a “Ver sismos en común” y el último a Ver sismos en común con $M_w > M_w$min”. Esto se puede ver en la Figura 38.

Figura 38: Interfaz de cálculo SSR.

Al hacer click sobre el botón “Ver todos los sismos”, la interfaz muestra todos los sismos previamente identificados con las características que se muestran en la Figura 15. Así la ventana que se muestra con las características se muestra en la Figura 39.

Eventos registrados

Event	Time	Latitude [°]	Longitude [°]	Depth [km]	Magnitude type	Magnitude	Epicentral dist [km]	Hipocentral dist[km]	Duration [s]	Time to stations [s]
1	2018-08-31T10:26:44	-31.807	-70.7508	106.6	mb	4.3	74.5821	130.1002	120	26.02
2	2018-08-31T13:25:01	-32.2062	-70.4787	96.26	Mww	5	71.2527	119.762	105	23.95
3	2018-09-07T02:39:17	-28.892	-69.9535	92.4	Mww	5.8	407.7849	418.1224	180	83.62
4	2018-09-07T23:12:47	-30.5238	-69.7354	106.14	Mwr	4.9	255.7664	276.9155	156	55.38
5	2018-09-10T08:24:11	-28.7518	-71.8113	10	Mww	5.1	410.8944	411.0161	160	82.2
6	2018-09-12T19:31:14	-31.529	-72.008	34.56	ML	4	138.7729	143.0115	80	28.6
7	2018-09-14T18:15:11	-32.5454	-71.6043	32.47	Mwr	4.5	59.251	67.5647	42	13.51
8	2018-09-14T18:46:51	-32.5629	-71.607	21.25	mb	4.1	60.1508	63.7941	47	12.76
9	2018-09-18T23:50:10	-30.8741	-71.435	48.93	Mwr	3.9	171.749	178.583	80	35.72

Figura 39: Ventana con todos los sismos previamente identificados.

Luego al ingresar un valor de “M_w mínimo” igual a 5 y hacer click en el botón “Ver sismos con M_w > M_w min”, se tiene la Figura 40

Sismos con Mw>5

Event	Time	Latitude [°]	Longitude [°]	Depth [km]	Magnitude type	Magnitude	Epicentral dist [km]	Hipocentral dist[km]	Duration [s]	Time to stations [s]
3	2018-09-07T02:39:17	-28.892	-69.9535	92.4	Mww	5.8	407.7849	418.1224	180	83.62
5	2018-09-10T08:24:11	-28.7518	-71.8113	10	Mww	5.1	410.8944	411.0161	160	82.2

Figura 40: Ventana con sismos con M_w > 5

Empleando el mismo perfil transversal usado en la Figura 10, en donde la estación de referencia

es la “T06” y la estación a estudiar es la “T08”, como se muestra en la Figura 41 y al ejecutar el botón “Ver sismos en común” se muestran los sismos en común registrados entre las carpetas de la estación “T06” y “T08”, como se puede ver en la Figura 42.

Figura 41: Ingreso de estaciones para ver que sismos preidentificados fueron registrados en ambas.

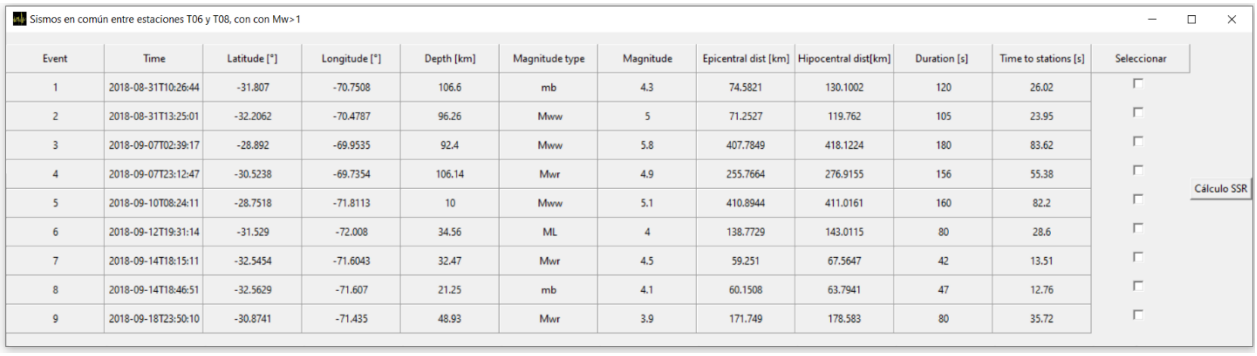
Sismos en común entre estaciones T06 y T08

Event	Time	Latitude [°]	Longitude [°]	Depth [km]	Magnitude type	Magnitude	Epicentral dist [km]	Hipocentral dist[km]	Duration [s]	Time to stations [s]
1	2018-08-31T10:26:44	-31.807	-70.7508	106.6	mb	4.3	74.5821	130.1002	120	26.02
2	2018-08-31T13:25:01	-32.2062	-70.4787	96.26	Mww	5	71.2527	119.762	105	23.95
3	2018-09-07T02:39:17	-28.892	-69.9535	92.4	Mww	5.8	407.7849	418.1224	180	83.62
4	2018-09-07T23:12:47	-30.5238	-69.7354	106.14	Mwr	4.9	255.7664	276.9155	156	55.38
5	2018-09-10T08:24:11	-28.7518	-71.8113	10	Mww	5.1	410.8944	411.0161	160	82.2
6	2018-09-12T19:31:14	-31.529	-72.008	34.56	ML	4	138.7729	143.0115	80	28.6
7	2018-09-14T18:15:11	-32.5454	-71.6043	32.47	Mwr	4.5	59.251	67.5647	42	13.51
8	2018-09-14T18:46:51	-32.5629	-71.607	21.25	mb	4.1	60.1508	63.7941	47	12.76
9	2018-09-18T23:50:10	-30.8741	-71.435	48.93	Mwr	3.9	171.749	178.583	80	35.72

Figura 42: Sismos en común preidentificados entre estaciones "T06" y "T08".

Finalmente, para realizar el procesamiento SSR entre los sismos disponibles, se debe clickear el último botón correspondiente a “Ver sismos en común con M_w >M_w min”, en donde se despliega

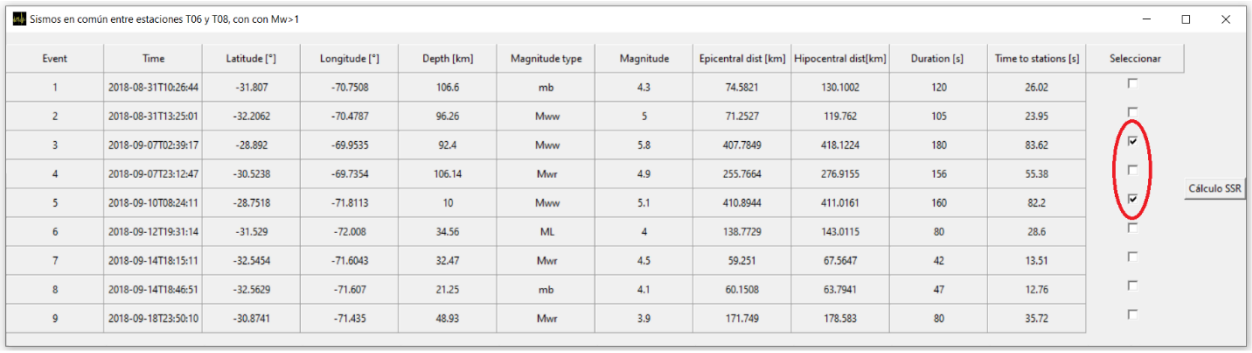
una ventana como la señalada en la Figura 43, en donde se puede seleccionar que sismos se desean procesar mediante un checkbutton en cada evento. La ventana desplegada muestra los sismos que se deseen procesar entre las estaciones “T06” y “T08” con un M_w mínimo igual a 1.



Event	Time	Latitude [°]	Longitude [°]	Depth [km]	Magnitude type	Magnitude	Epicentral dist [km]	Hipocentral dist[km]	Duration [s]	Time to stations [s]	Seleccionar
1	2018-08-31T10:26:44	-31.807	-70.7508	106.6	mb	4.3	74.5821	130.1002	120	26.02	☐
2	2018-08-31T13:25:01	-32.2062	-70.4787	96.26	Mww	5	71.2527	119.762	105	23.95	☐
3	2018-09-07T02:39:17	-28.892	-69.9535	92.4	Mww	5.8	407.7849	418.1224	180	83.62	☐
4	2018-09-07T23:12:47	-30.5238	-69.7354	106.14	Mwr	4.9	255.7664	276.9155	156	55.38	☐
5	2018-09-10T08:24:11	-28.7518	-71.8113	10	Mww	5.1	410.8944	411.0161	160	82.2	☐
6	2018-09-12T19:31:14	-31.529	-72.008	34.56	ML	4	138.7729	143.0115	80	28.6	☐
7	2018-09-14T18:15:11	-32.5454	-71.6043	32.47	Mwr	4.5	59.251	67.5647	42	13.51	☐
8	2018-09-14T18:46:51	-32.5629	-71.607	21.25	mb	4.1	60.1508	63.7941	47	12.76	☐
9	2018-09-18T23:50:10	-30.8741	-71.435	48.93	Mwr	3.9	171.749	178.583	80	35.72	☐

Figura 43: Sismos en común entre la estación "T06" y "T08" con un M_w mínimo > 1

Al seleccionar los sismos “Evento 3”, y “Evento 5”, se tiene la Figura 44, y al clickear el botón “Cálculo SSR” comienza el procesamiento SSR, en la Figura 45Figura 44 se muestran los resultados del procesamiento SSR en una ventana emergente.



Event	Time	Latitude [°]	Longitude [°]	Depth [km]	Magnitude type	Magnitude	Epicentral dist [km]	Hipocentral dist[km]	Duration [s]	Time to stations [s]	Seleccionar
1	2018-08-31T10:26:44	-31.807	-70.7508	106.6	mb	4.3	74.5821	130.1002	120	26.02	☐
2	2018-08-31T13:25:01	-32.2062	-70.4787	96.26	Mww	5	71.2527	119.762	105	23.95	☐
3	2018-09-07T02:39:17	-28.892	-69.9535	92.4	Mww	5.8	407.7849	418.1224	180	83.62	☑
4	2018-09-07T23:12:47	-30.5238	-69.7354	106.14	Mwr	4.9	255.7664	276.9155	156	55.38	☐
5	2018-09-10T08:24:11	-28.7518	-71.8113	10	Mww	5.1	410.8944	411.0161	160	82.2	☑
6	2018-09-12T19:31:14	-31.529	-72.008	34.56	ML	4	138.7729	143.0115	80	28.6	☐
7	2018-09-14T18:15:11	-32.5454	-71.6043	32.47	Mwr	4.5	59.251	67.5647	42	13.51	☐
8	2018-09-14T18:46:51	-32.5629	-71.607	21.25	mb	4.1	60.1508	63.7941	47	12.76	☐
9	2018-09-18T23:50:10	-30.8741	-71.435	48.93	Mwr	3.9	171.749	178.583	80	35.72	☐

Figura 44: Selección de sismos a procesar.

En la ventana de resultados se muestran los gráficos del método SSR de las 3 componentes, Transversal, Longitudinal y Vertical (Z). En donde cada uno de los sismos se muestra en el gráfico mediante una línea gris y sus promedios se muestran en línea roja. En esta ventana de resultados, que corresponde a la Figura 45, se muestran 6 botones, en donde 3 corresponden a los mostrados en el método HVSR, por lo que no serán detallados. Luego, los otros 3 botones corresponden a la opción de ver los registros sísmicos procesados, se tomó la decisión de crear una ventana para cada sismo ya que de esta forma será más difícil confundir las trazas y, el usuario al querer procesar una mayor cantidad de sismos.

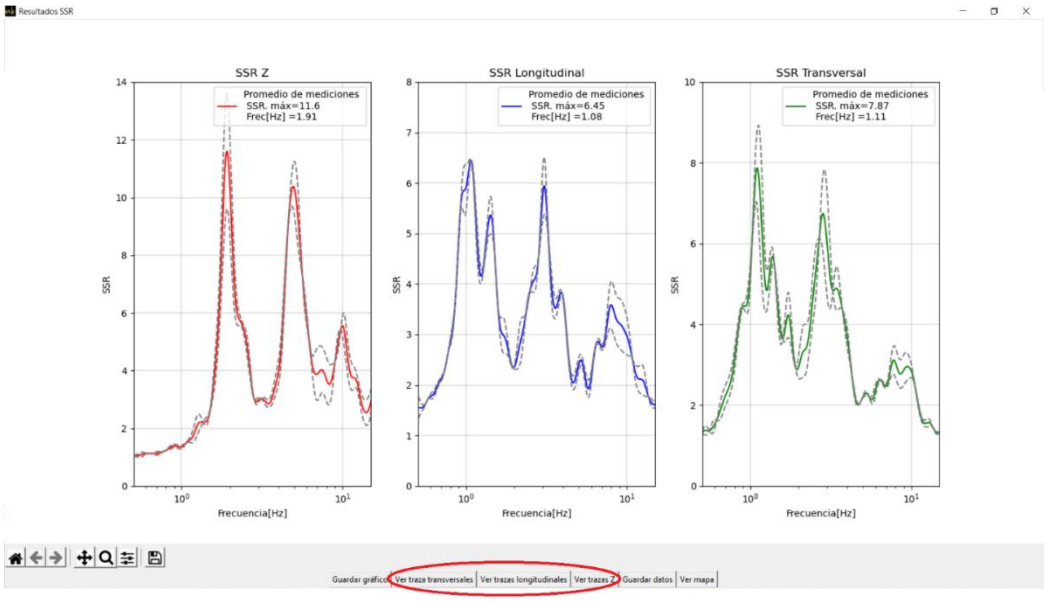


Figura 45: Ventana de resultados cálculo SSR.

Al hacer click en el botón “Ver trazas transversales” se muestra una ventana en donde se pueden ver las trazas procesadas de ambas estaciones, donde se indica el evento y la estación que registró el evento, como se ve en la Figura 46.

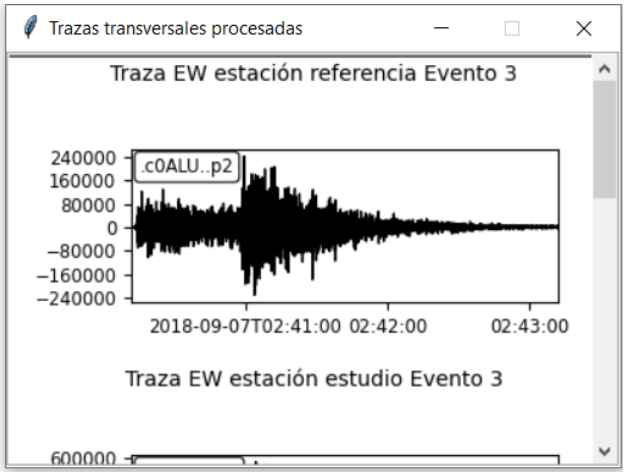


Figura 46: Ventana que muestra tranzas transversales del método SSR.

De esta forma, el diagrama de flujo para el uso del método SSR en la interfaz interactiva se muestra en la Figura 47.

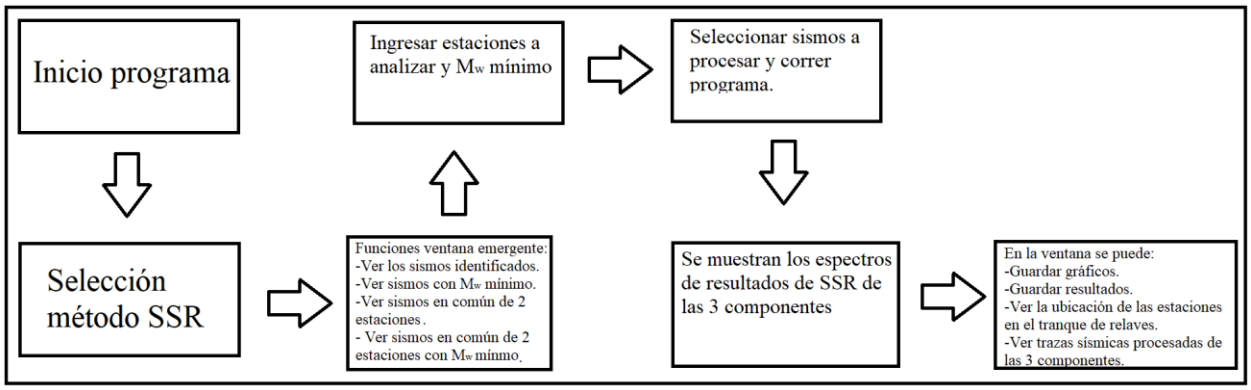


Figura 47: Esquema de uso de interfaz SSR.

Capítulo 5
Validación de resultados

Para la validación de resultados se compararon los resultados obtenidos mediante el software Geopsy y los resultados obtenidos mediante la interfaz desarrollada. De esta forma, para la validación de los resultados del método HVSR y SSR se seleccionaron 6 estaciones a lo largo del tranque, 3 ubicadas aguas abajo y 3 en el coronamiento. Estas estaciones corresponden a las T01, T05, T12, T14, T26 y T28. Para el análisis HVSR, se seleccionaron trazas de tiempo aleatorias en donde se procesó ruido ambiental y para el análisis SSR se generaron 3 perfiles transversales en donde se procesaron 2 sismos. Cabe señalar que las trazas procesadas no fueron proyectadas con mediante el ángulo que se muestra en la Figura 17. Los perfiles se muestran en la Figura 48.

Figura 48: Estaciones y perfiles transversales para validación de resultados.

Los parámetros que se emplearon para el procesamiento de señales en el software Geopsy se muestran en la Tabla 6.

Tabla 6: Variables ingresadas para procesamiento Geopsy.

Parámetro	Valor
Filtro Band-pass	0.05-25
Orden de filtro Butterworth	4
Ventana Tukey	5%
Width Konno y Ohmachi	20%

Cabe señalar que para el parámetro “Width” de Geopsy, un valor del 20% corresponde a un valor de 40 para el valor de “b” para el suavizado de Konno y Ohmachi.

5.1 Validación de resultados HVSR

La validación de resultados del método HVSR se realizó mediante una ventana de largo de 3[hr], en donde se tomaron sub-ventanas de un largo de 60[seg] para todas las estaciones de los perfiles.

De esta forma, en la Figura 49, se muestran los resultados del procesamiento HVSR del Perfil 1, en ambos gráficos se comparan los resultados obtenidos mediante Geopsy y del código desarrollado.

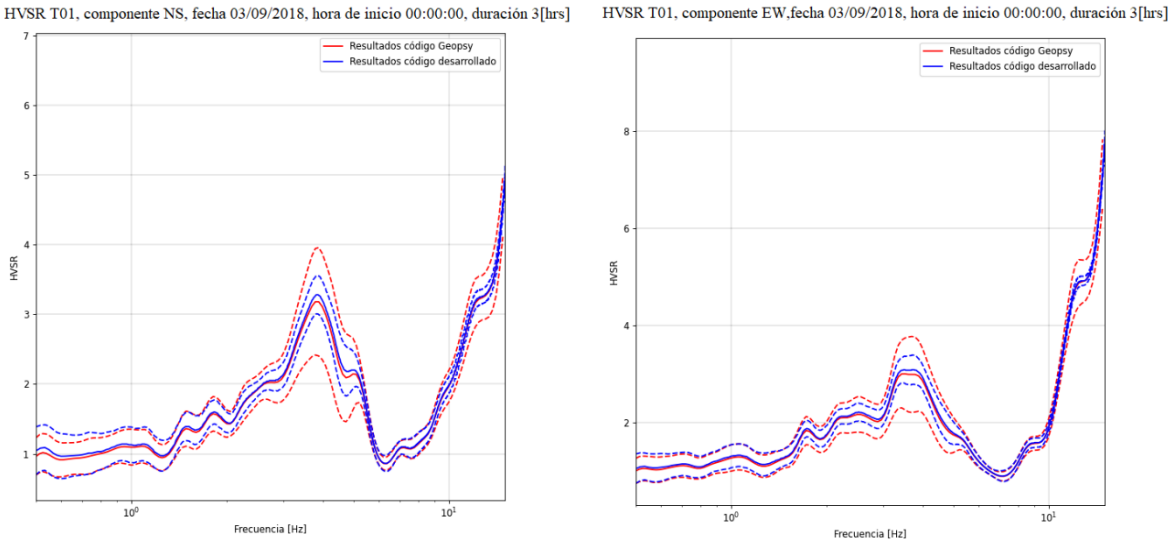


Figura 49: Comparación de resultados HVSR estación T01.

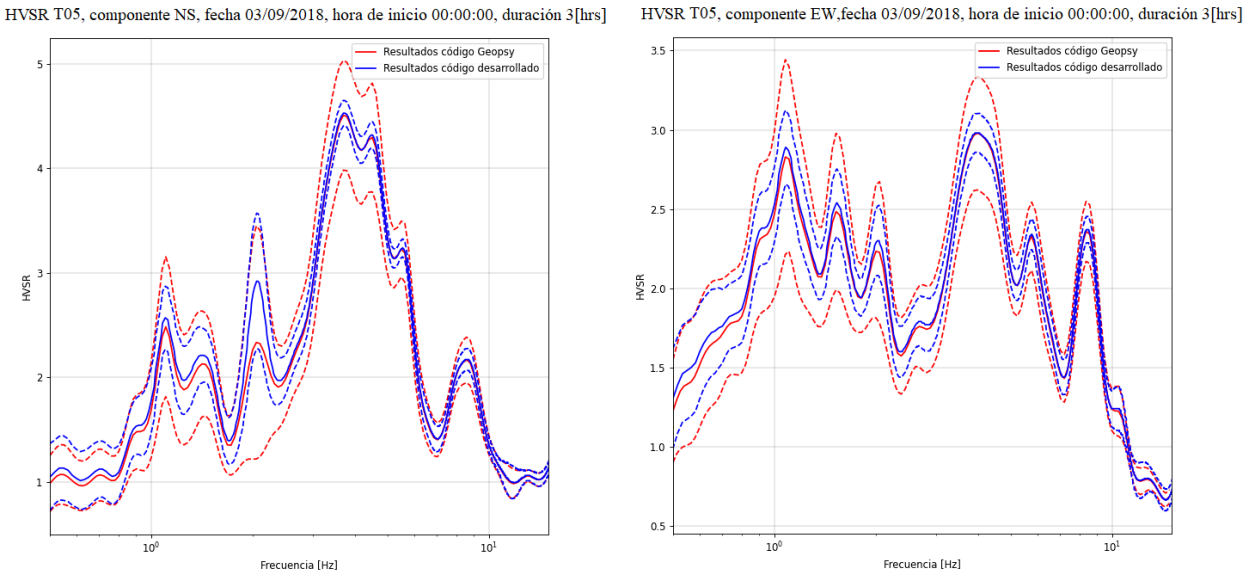


Figura 50: Comparación de resultados HVSR estación T05.

Para el Perfil 2, la comparación de resultados se muestra en la Figura 51 y Figura 52.

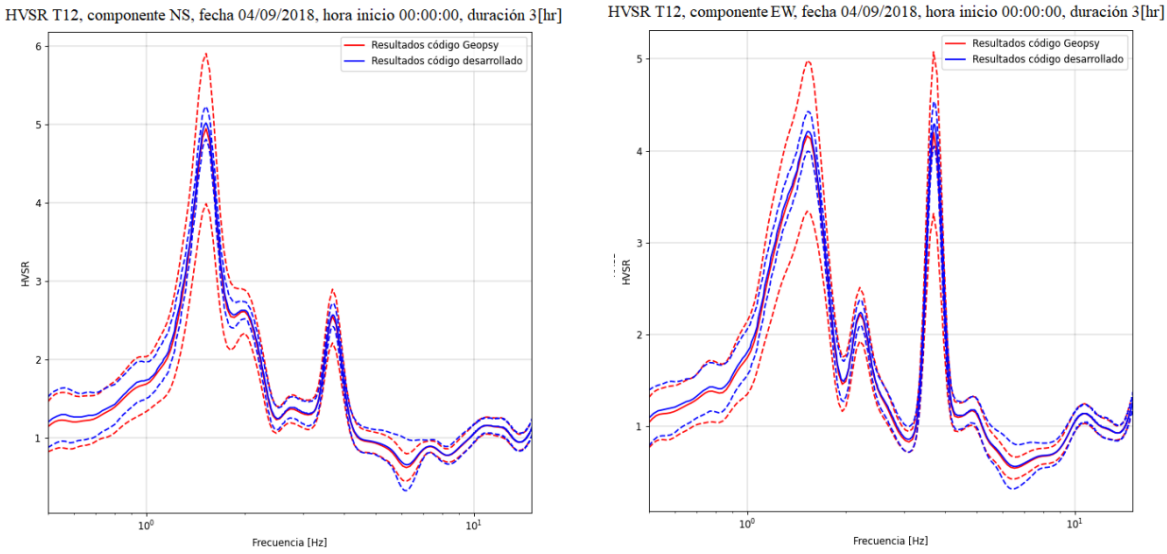


Figura 51: Comparación de resultados HVSR estación T12.

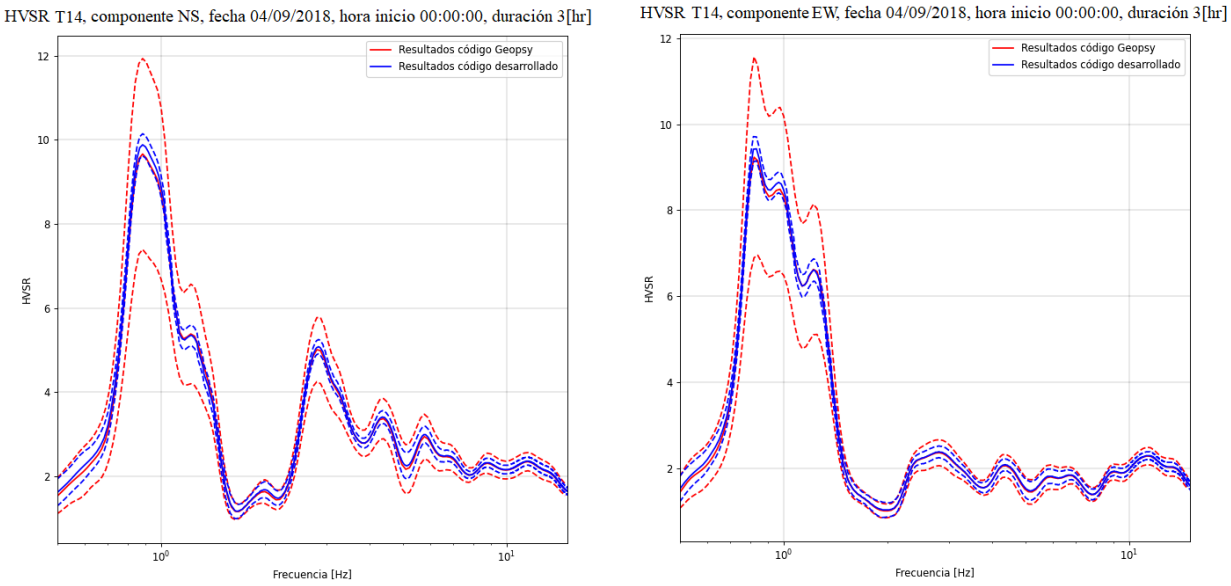


Figura 52: Comparación de resultados HVSR estación T14.

Finalmente, en las Figura 53 y Figura 54 se pueden ver la comparación de resultados del método HVSR para el procesamiento de Geopsy y del código desarrollado del Perfil 3.

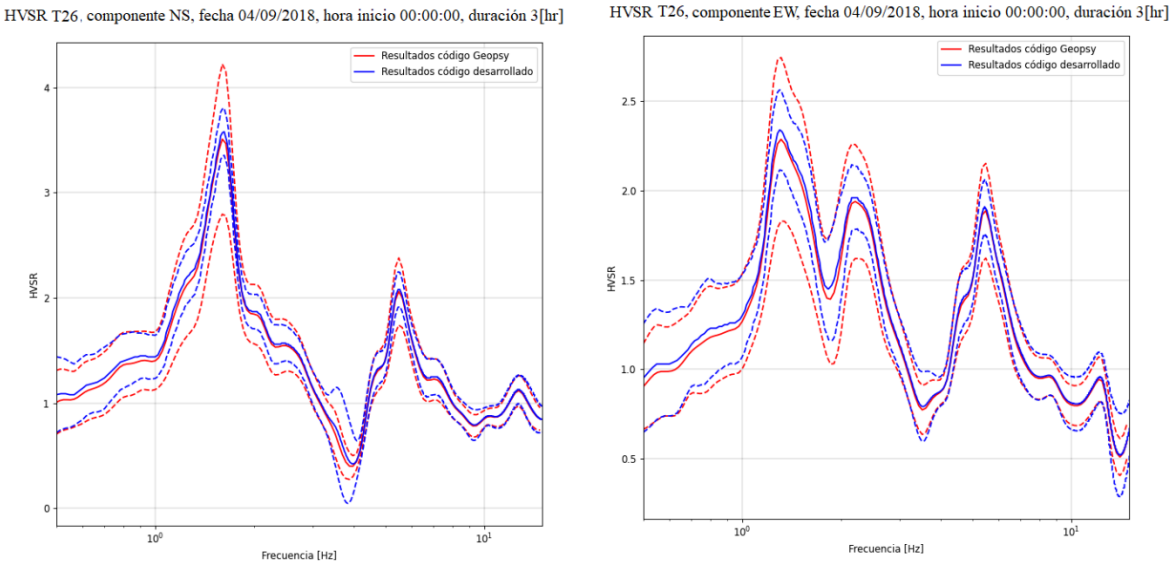


Figura 53: Comparación de resultados HVSR estación T26.

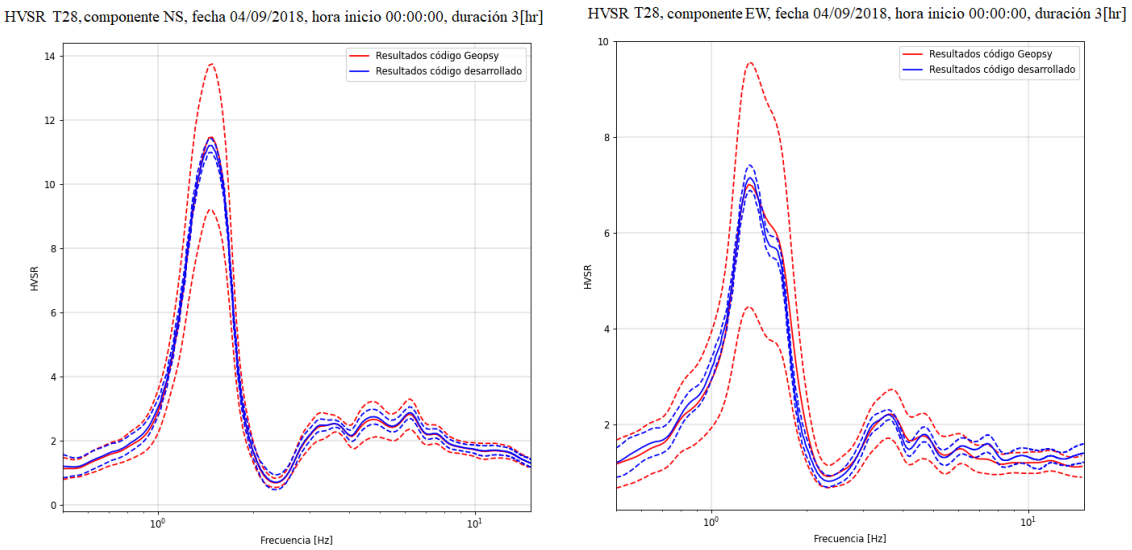


Figura 54: Comparación de resultados HVSR estación T28.

5.2 Validación de resultados SSR

Para la validación de resultados SSR se procesaron 2 sismos, que se muestran en la Tabla 7, los cuales son preidentificados, con un M_w mínimo igual a 5, en donde cada sismo se procesó de manera independiente. Geopsy no permite un procesamiento SSR en sus funciones, pero en la forma en la que se procedió fue procesar la traza de la estación de referencia como traza vertical y la de estudio como componente horizontal, en donde las 2 componentes ingresadas como horizontales es la misma, de esta forma se procesaron las componentes longitudinal y transversal de manera independiente.

Tabla 7: Sismos para validación de resultados SSR.

Evento	Fecha	Hora	Mw
1	07/09/2018	02:40:12	5.8
2	10/09/2018	08:25:10	5.1

Así, la comparación de resultados SSR del Perfil 1, en donde se estudian las componentes longitudinales y transversales, para los 2 sismos vienen dados por la Figura 55.

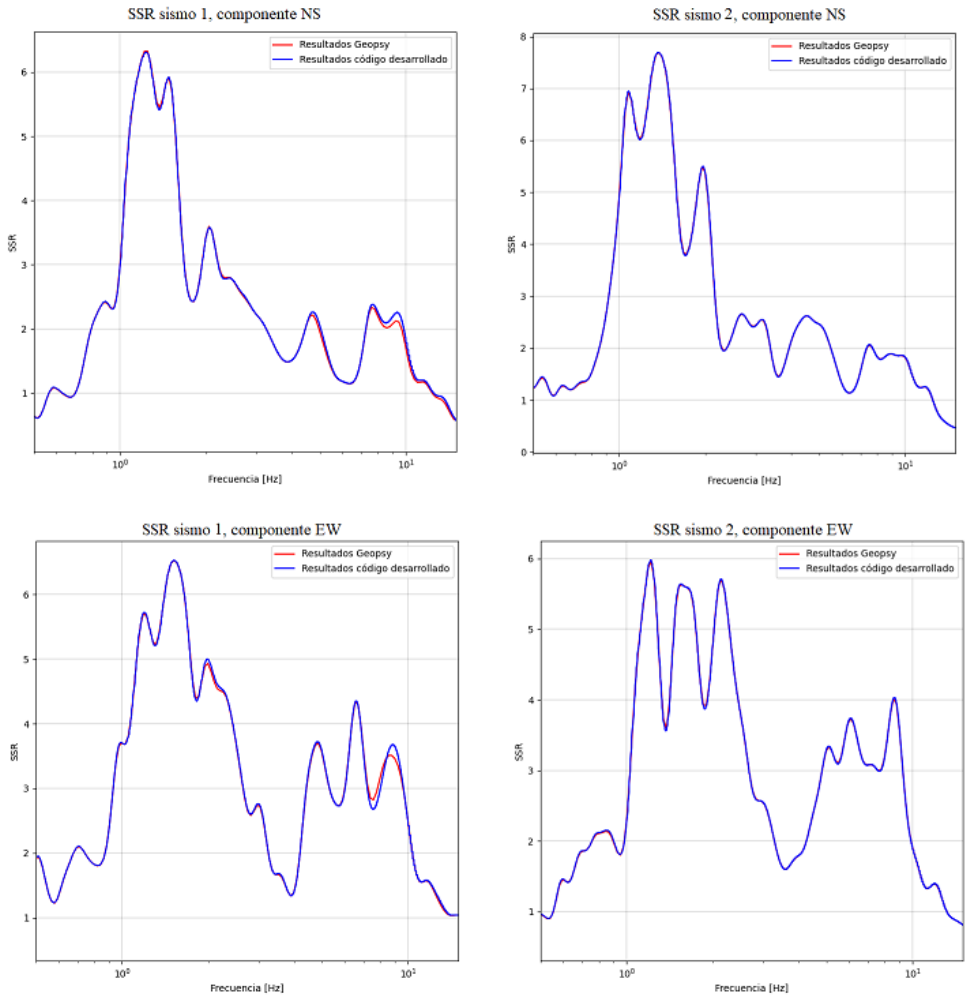


Figura 55: Resultados SSR para los sismos 1 y 2, Perfil 1.

Por otra parte, para el Perfil 2, la comparación de resultados se puede ver en la Figura 56.

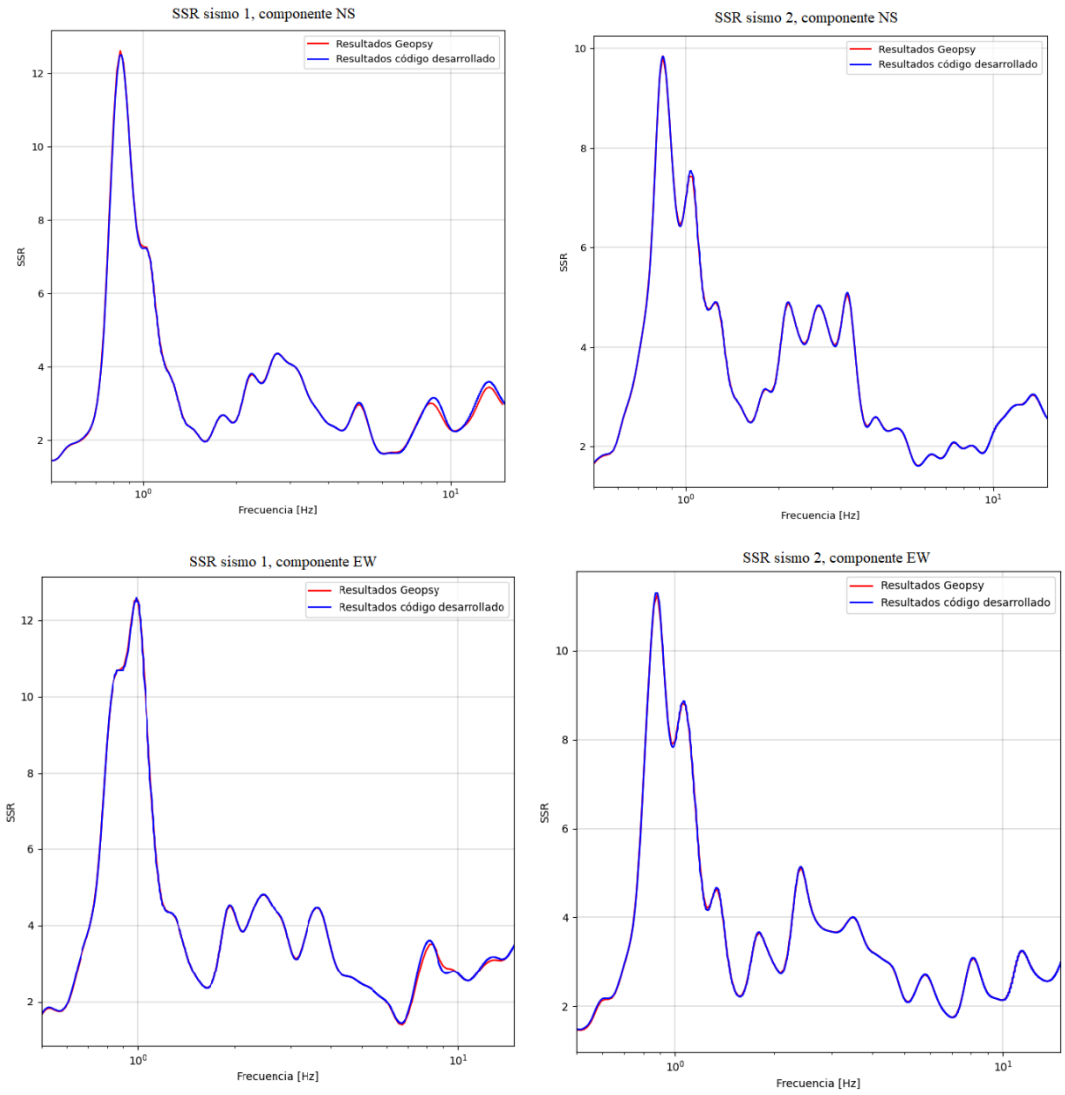


Figura 56:Resultados SSR sismo 1 y 2 para Perfil 2.

Capítulo 6
Discusión

El trabajo desarrollado cumple con sus objetivos generales y específicos, en donde se avanza en el desarrollo de una interfaz gráfica interactiva capaz de realizar un procesamiento de señales para los métodos HVSR y SSR, la cual pueda identificar la frecuencia predominante de vibración de la zona del tranque en estudio. A su vez, el programa desarrollado agiliza el procesamiento de datos que se desee realizar ya que no es necesaria la selección de manera manual de la traza a analizar y la selección de ventanas, si no que, en caso de ser el método HVSR se tienen que ingresar las variables como fecha, hora y duración de análisis para que el programa automáticamente seleccione la traza y la procese. Mientras que para el método SSR solo se tiene que ingresar las estaciones que se desean procesar y seleccionar los sismos preidentificados los cuales se quieran estudiar.

En el código desarrollado, el procesamiento de señales se simplifica al ingresar al inicio del script las variables globales que influyen en el procesamiento de las trazas y que pueden ser modificadas por el usuario, por ejemplo, las frecuencias mínima y máxima de filtrado, el largo de las subventanas (para el método HVSR), el parámetro “b” para el suavizado de Konno y Ohmachi, entre otras variables.

En cuanto a la automatización de procesos, el código avanza en el desarrollo de una interfaz que permita la automatización de procesamiento de datos, pero en el presente trabajo se realizaron actividades (procesos manuales) que son imprescindibles para un correcto funcionamiento de la interfaz, un ejemplo de lo anterior es que la ubicación de las coordenadas (en pixeles) de la ubicación de cada estación para poder ser marcada mediante el botón “Ver Mapa” fue obtenida de manera manual.

Siguiendo con los resultados del trabajo desarrollado, en cuanto a las funciones programadas, se lograron desarrollar funciones que permiten exportar los resultados de los métodos SSR y HVSR (resultados que son graficados en las ventanas de resultados), de esta forma el usuario puede almacenar información procesada para luego manipularla de manera universal (vale decir, procesada en otros softwares). De esta forma no es necesario implementar nuevamente la rutina para la obtención de datos que ya fueron obtenidos y así seguir la idea de agilizar el proceso de estudio de la propiedad dinámica que busca obtener el presente trabajo.

Por otra parte, en cuanto a los resultados que se obtuvieron los cuales fueron graficados y comparados en el Capítulo 5, se logran validar. Los valores promedios de obtenidos son idénticos, mientras que la dispersión es menor en el código desarrollado, esto se debe a que el proceso para obtener el estudio estadístico es diferente en los programas empleados. Lo anterior se debe posiblemente a que el procesamiento de data que realiza Geopsy no es similar al que realiza el código.

A su vez, los gráficos obtenidos en el presente trabajo, de manera preliminar, guardan estrecha relación por lo expuesto en el Capítulo 2, donde se plantea que los valores máximos de

amplificación de movimientos de una geo-estructura se encuentran en la dirección perpendicular a la cresta (Lovati et al., 2011). Esta dirección perpendicular a la cresta corresponde a la dirección transversal, lo anterior se puede ver en las Figura 31 y Figura 45, en donde los resultados para los métodos HVSR y SSR tienen sus mayores valores en la dirección transversal. Siguiendo con lo expuesto en el Capítulo 2, Pastén et al., 2022 plantean que la frecuencia predominante de vibración para un tranque de relaves varía a lo largo de este, como en su altura, esto se ve en la Figura 14. Esto se puede observar en los resultados que se ven en el Capítulo 5 en la validación del método H/V, donde las estaciones que se encuentran cercanas a los estribos (Perfil 1 y Perfil 3) tienen una frecuencia predominante de vibración mayor que las del Perfil 2. Es muy importante señalar que los resultados recién señalados no son concluyentes, si no que encaminan y dan una referencia de los gráficos obtenidos en el trabajo desarrollado.

Capítulo 7
Conclusiones.

Se logra avanzar en el desarrollo de una interfaz gráfica que permita la obtención de frecuencia predominante de vibración de un tranque de relaves, además se logró implementar un algoritmo que encamine la automatización de procesamiento de registros. Existen procesos realizados de manera manual, los cuales son expuestos, estos deben ser desarrollados para seguir en la línea de desarrollo del presente trabajo y así avanzar a la automatización total de todos los procesos.

Se verifica la variación de la frecuencia predominante de vibración del tranque de relaves en estudio, tanto en su eje longitudinal y transversal, como en la altura de este, al realizar procesamientos de HVSR y SSR en los 3 perfiles transversales elegidos. Además, se logra verificar el método HVSR mediante la comparación de los resultados obtenidos con los resultados obtenidos mediante el método SSR, en donde se observan frecuencias de vibración predominante similares en ambos métodos para los 3 perfiles.

En cuanto a los resultados obtenidos, los gráficos obtenidos mediante el software Geopsy son idénticos a los obtenidos en el código, tanto como para el método SSR, como para el método HVSR. El estudio estadístico desarrollado tiene menos dispersión que el que emplea Geopsy. Se logra identificar la frecuencia predominante de vibración por lo que el procesamiento de señales desarrollado es válido.

Capítulo 8
Comentarios.

Se recomienda a futuro poder conectar la interfaz gráfica a una base de datos que se esté actualizando diariamente con datos de una red de geófonos instalada en un tranque de relaves, para así poder tener un seguimiento diario de la frecuencia predominante de vibración y poder asociar variaciones de esta. Además, se recomienda a futuras personas que sigan la línea de investigación mantener el lenguaje de programación Python ya que es código abierto y amigable con el usuario, tanto el lenguaje, como la comunidad que siempre busca innovar en las funciones que son desarrolladas. Además, es fácil y sencillo de implementar, ya que no requiere una licencia para su uso.

Finalmente, se debe encaminar el desarrollo de la plataforma hacia una automatización de procesos para su funcionamiento, como la identificación de pixeles de la imagen donde se desee identificar cada estación en el tranque de relaves.

BIBLIOGRAFÍA

Bonneyfoy-Claude, S., Cotton, F., & Bard, P.-Y. (2006). The nature of noise wavefield and its applications for site effects studies: a literature review. *Earth Sci Rev 79*, 205-227.

Borcherdt, R. (1970). Effects of local geology on ground motion near San Francisco Bay. *Seism. Soc. Am., 60, 1,*, 29-61.

Calderón, F & Cordone, J.P. (2019). Determinación de la frecuencia fundamental de una presa CFRD mediante funciones de transferencia. *Congreso Chileno de Sismología e Ingeniería Sísmica.* Valdivia.

Campaña, J., Valenzuela, L., Bard, E., Verdugo, R., & Peters, G. (2017). Evaluation of Tailings Dams Subjected to Large Earthquakes., (págs. 1615-1618). Seoul.

Celebi, M. (1991). Topographical and geological amplification: case studies and engineering implications. *Structural Safety, 10*, 199-217.

Celebi, M., Prince, J., Dietel, C., Onate, M., & Chavez, G. (1987). The Culprit in Mexico City—Amplification of Motions. *Earthquake Spectra 3*, 315-328.

Chávez-García, J. L. (1993). Site effect evaluation using spectral ratios with only one station. Bulletin of the Seismological Society of America. *Bulletin of the Seismological Society of Americ*, 1574-1594.

Garofalo, F., Foti, S., Hollender, F., Bard, P., Cornou, C., Cox, B., . . . Vergniault, C. (2016). InterPACIFIC project: Comparison of invasive and non-invasive methods for seismic site characterization. Part II: Inter-comparison between surface-wave and borehole methods. *Soil Dynamics and Earthquake Engineering*, Pages 241-254.

Gazetas, G. (1987). Seismic response of earth dams: some recent developments. En *Soil Dynamics and Earthquake Engineering* (págs. 2-47).

González, G. (2019). *Comportamiento sísmico de perfiles de suelo de espesor menor a 30 metros y perfiles con inclusiones de alta o baja velocidad.* Santiago, Universidad de Chile, Departamento de Ingeniería Civil: Memoria para optar al título de ingeniero civil.

Kafadar, O. (2020). A geophone-based and low-cost data acquisition and analysis system designed for microtremor measurements. *Geosci. Instrum. Method. Data Syst*, 365-373.

Konno, K., & Ohmachi, T. (1998). Ground-Motion characteristics estimated from spectral ratio. *Bull. Seism. Soc. Am.*, 228-241.

Lovati, S., Bakavoli, M., Massa, M., Ferreti, G., Pacor, F., Paolucci, R., . . . Kamalian, M. (2011). Estimation of topographical effects at Narni ridge (Central Italy): comparisons between experimental. *Bull Earthquake Eng*, 1987–2005.

Molnar, S., Castellaro, S., Cornou, C., Crow, H., Cornou, C., Hunter, J., . . . Yong, A. (2018). Application of Microtremor Horizontal-to-Vertical Spectral Ratio (MHVSR) Analysis for Site Characterization: State of the Art. *Surv Geophys 39*, 613-631.

Morril, J. (2020). Safety First: Guidelines for Responsible Mine Tailings Management. *Earthworks and MiningWatch Canada,*.

Nakamura, Y. (1989). A method for dynamic characteristics estimation of subsurface using. *Quarterly Reports of the Railway Technical Research, 30.*, 25-33.

Nogoshi, M., & Igarashi, T. (1971). On the amplitude characteristics of microtremor. *(in Japanese with English abstract) J. Seismol. Soc. Japan,*, 26-40.

Pastén, C. (2007). *Respuesta Sísmica de la Cuenca de Santiago.* Santiago, Universidad de Chile, Departamento de Ingeniería Civil: Tesis para optar al grado de magister en ciencas de la ingeniería, mención Ingeniería Geotecnia, Memoria para optar al título de Ingeniero Civil.

Pastén, C., Comte, D., Peña, G., Burgos, J., & Rietbrock, A. (2019). Dynamic Characterization of a Tailings Dam Embankment Using a Dense Seismic Array: Preliminary Results. *6th International Seminar on Tailings Management.*

Pastén, C., Peña, G., Diana, C., Díaz, L., Burgos, J., & Rietbrock, A. (2022). On the use of the H/V spectral ratio method to estimate the fundamental frecuency of tailing dams. *Under review in Journal of Earthquake Engineering*.

Press, W., Flannery, B., Teukolsky, S., & Vetterling, W. (1992). Fast Fourier Transform. *Fast Fourier Transform. Ch. 12 in Numerical Recipes in FORTRAN: The Art of Scientific Computing,*, 490-529.

Sauter, F. (1989). *Fundamentos de ingeniería sísmica I: Introducción a la sismología.* Costa Rica: Editorial Tecnológica de Costa Rica.

ANEXO

A continuación, se adjunta el código desarrollado:

```
1.   #Importación de módulos.
2.  %matplotlib notebook
3.  from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
4.  #Importar modulo os:
5.  import os
6.  #Importar modulo numpy como np:
7.  import numpy as np
8.  #Importar read de modulo obspy:
9.  from obspy import read
10. #Importar signal de modulo obspy:
11. from obspy import signal
12. #Importar signal de modulo scipy:
13. from scipy import signal
14. #Importar modulo matplotlib.pyplot como plt:
15. import matplotlib.pyplot as plt
16. #Importar modulo tkinter:
17. from tkinter import *
18. #Importar función de suavizado de Konno & Ohmachi:
19. from obspy.signal.konnoohmachismoothing import konno_ohmachi_smoothing
20. #Se importa Figure para poder graficar datos:
21. from matplotlib.figure import Figure
22. #Se importa Image e ImageTk para poder mostrar figuras cargadas:
23. from PIL import Image, ImageTk
24. #Definición de variables globales para el procesamiento de señales.
25. ventana=60                    #segundos ventanas de analisis HVSR.
26. freq_last = int(20.0)         #ultima frecuencia a considerar.
27. tiempo_analisis_SSR= 180      #duración de analisis SSR.
28. frec_min_filtrado=0.05        #se determina la frecuencia mínima de filtrado[Hz].
29. frec_max_filtrado=25          #se determina la frecuencia máxima de filtrado[Hz].
30. ventana_tukey='5%'            # % con el que se trabaja en la ventana tukey.
31. bandwith_KyO=40               # valor de parámetro b para suaviazado de Konno y Ohmachi.
32.
33. #Se cargan parámetros para las funciones de marcar en el mapa de estaciones
34.
35. resizex=7    #Con estos resize se puede variar el resize del PNG del marcardor dependiendo
    de la imagen de fondo
36. resizey=7
37. errorx=11    #Hay cierto error al analizar la figura con matplotlib e ImageTk, por lo que
    hay que ajustar a mano esto
38. errory=8
39. #Variables para función SSR
40. columnas_del_archivo=11 #Columnas de archivo donde se muestra información de sismos
41. diccionario_int_vars=dict() #Diccionario que recibirá variables en función de selección de
    sismos en procesamiento SSR.
42. dic_trazas_eventos=dict()   #Diccionario que se usará para almacenar las trazas de SSR
43.
44. #Se procede a definir rutas de archivos externos
45. archivo_coordenadas='C:/Users/jose_/Script Tesis/Scripts desarrollados/Coordenadas
    Estaciones en foto empleada.csv' #Ruta de archivo que muestra pixel y ángulo de estaciones.
46. archivo_eventos='D:/Disco
    Duro/Tesis/Sismos_Importantes_Finales/Sismos_importantes_final_modificado.csv'
    #Ruta de archivo que muestra info de los sismos
47. path='D:/Disco Duro/Tesis/Estaciones/'
    #Ruta de carpeta donde se encuentran las trazas
48.
49.
```

```
'----------------Funciones para HVSR-------------------'

#Esta función tiene como objetivo reaizar procesamiento HVSR con los inputs que se pueden observar
#entrega como resultado un diccionario con gráfico de componentes ponderadas o independientes
#y resultados que fueron graficados.

def H_on_V_ambiental(estacion,tiempo_analisis,hora_inicio,combinar,fecha):
    n_ventanas=int(tiempo_analisis)/int(ventana) #Numero de ventanas a analizar
    #Se crea la variable de hora final en funcion de los segundos a analizar:
    hora_final=str(int(hora_inicio[0:2])+int(tiempo_analisis/3600))+':'+str(int(hora_inicio[3:5])+int(tiempo_analisis/60))+':'+str(int(hora_inicio[6:8]))+'0' #Se agrega el 0 a "mano"
    #Direccion de la estacion actual:
    estaciones = np.loadtxt(archivo_coordenadas,dtype=str,comments='Estaciones',delimiter=';',
                        usecols=0,skiprows=1)
    angulos = np.loadtxt(archivo_coordenadas,dtype=str,comments='Estaciones',delimiter=';',
                        usecols=3,skiprows=1)
    #Se define el angulo para trabajar (azimut)
    for i in np.arange(len(estaciones)):
        if estacion==estaciones[i]:
            angulo=float(angulos[i])
    path_station = path + estacion+'/'
    #Se obtienen los años y fechas en str para comparar con el archivo.
    año_fecha=str(int(fecha[6:10])-2000)
    mes_fecha=str(fecha[3:5])
    dia_fecha=str(fecha[0:2])
    #Se crea la variable seg_inicio a partir de hora_inicio.
    seg_inicio=int(hora_inicio[0:2])*3600+int(hora_inicio[3:5])*60+int(hora_inicio[6:8])
    Z=1
    #Vector de datos diarios de la estacion actual:
    daily_data = os.listdir(path_station)
    #Se obtienen los registros a analizar según la estación y fecha.
    for file in daily_data:
        #Se asigna el registro a la coordenada Z
        if (file.endswith('.pri0') and file.startswith(año_fecha+
                            mes_fecha+
                            dia_fecha,5)):


            Z = read(path_station+file)

        #Se asigna el registro a la coordenada NS
        if (file.endswith('.pri1') and file.startswith(año_fecha+
                            mes_fecha+
                            dia_fecha,5)):
            NS= read(path_station+file)

        #Se asigna el registro a la cordenada EW
        if (file.endswith('.pri2') and file.startswith(año_fecha+
                            mes_fecha+
                            dia_fecha,5)):
            EW= read(path_station+file)
      if Z==1:
          messagebox.showinfo('Error',"No existe información para procesar :(")
          return
      #Se corta la traza en el tiempo que se desea a analizar

    Z.trim(Z[0].stats.starttime+seg_inicio,Z[0].stats.starttime+tiempo_analisis+seg_inicio) #la funcion trim trabaja con segundos

    NS.trim(NS[0].stats.starttime+seg_inicio,NS[0].stats.starttime+tiempo_analisis+seg_inicio)

    EW.trim(EW[0].stats.starttime+seg_inicio,EW[0].stats.starttime+tiempo_analisis+seg_inicio)
```

```
107.
108.      #Se realiza el detrend y el filtrado
109.      Z[0].detrend()
110.      NS[0].detrend()
111.      EW[0].detrend()
112.
113.      #Filtro butterworth orden 4  0.05 - 30 Hz
114.      Z[0].filter('bandpass', freqmin=frec_min_filtrado, freqmax=frec_max_filtrado, corners
    = 4, zerophase=True)
115.      NS[0].filter('bandpass', freqmin=frec_min_filtrado, freqmax=frec_max_filtrado, corners
    = 4, zerophase=True)
116.      EW[0].filter('bandpass', freqmin=frec_min_filtrado, freqmax=frec_max_filtrado, corners
    = 4, zerophase=True)
117.
118.
119.
120.      #Se crean las listas donde irá cada ventana de la traza a analizar
121.      z_lista=[]
122.      ns_lista=[]
123.      ew_lista=[]
124.      H_on_V_pond=[]
125.      H_on_V_NS=[]
126.      H_on_V_EW=[]
127.
128.      for i in range(int(n_ventanas)):#el indice es hasta +1 para que tome la ultima ventana
129.          z=Z.copy().trim(Z[0].stats.starttime+i*ventana,Z[0].stats.starttime+(i+1)*ventana)
130.
    ns=NS.copy().trim(NS[0].stats.starttime+i*ventana,NS[0].stats.starttime+(i+1)*ventana)
131.
    ew=EW.copy().trim(EW[0].stats.starttime+i*ventana,EW[0].stats.starttime+(i+1)*ventana)
132.
133.          #Se procede a asignar la data a cada vector con el que se trabajará
134.          data_z=z[0].data
135.          data_ns=ns[0].data
136.          data_ew=ew[0].data
137.
138.          #Se define la ventana tukey
139.          windows=signal.tukey(len(data_z),0.05)
140.
141.          #Se multiplica la ventana tukey por la ventana cortada
142.          data_tukey_z=windows*data_z
143.          data_tukey_ns=windows*data_ns
144.          data_tukey_ew=windows*data_ew
145.          data_z=data_tukey_z
146.          data_ns=data_tukey_ns
147.          data_ew=data_tukey_ew
148.
149.          #Se procede a proyectar las trazas NS y EW en función del angulo del talud
150.          data_ns_pr=data_ns*np.cos((angulo)*np.pi/180.0) -
    data_ew*np.sin((angulo)*np.pi/180.0)
151.          data_ew_pr=data_ew*np.sin((angulo)*np.pi/180.0) +
    data_ew*np.cos((angulo)*np.pi/180.0)
152.
153.          #Se procede a realizar las transformadas de fourier
154.          data_transformed_z = np.abs(np.fft.rfft(data_z))
155.          frequency_rescued = np.fft.rfftfreq(len(data_z), d=1./Z[0].stats.sampling_rate)
    #Vector con frecuencias
156.          data_transformed_rescued_z = data_transformed_z
157.          data_transformed_ns = np.abs(np.fft.rfft(data_ns_pr))
158.          data_transformed_rescued_ns = data_transformed_ns
159.          data_transformed_ew = np.abs(np.fft.rfft(data_ew_pr))
160.          data_transformed_rescued_ew = data_transformed_ew
161.
162.
163.
164.
```

```
            #Se realiza el suavizado de Konno & Ohmachi
            data_smoothed_z =
    konno_ohmachi_smoothing(data_transformed_rescued_z,frequency_rescued,
    bandwidth=bandwith_KyO,max_memory_usage=4096)
            data_smoothed_ns =
    konno_ohmachi_smoothing(data_transformed_rescued_ns,frequency_rescued,
    bandwidth=bandwith_KyO,max_memory_usage=4096)
            data_smoothed_ew =
    konno_ohmachi_smoothing(data_transformed_rescued_ew,frequency_rescued,
    bandwidth=bandwith_KyO,max_memory_usage=4096)

            #Se vuelve a trabajar con su nombre original para mayor comodidad
            data_z=data_smoothed_z
            data_ns = data_smoothed_ns
            data_ew = data_smoothed_ew
            #Se guardan los valores en las listas
            z_lista.append(data_z)
            ns_lista.append(data_ns)
            ew_lista.append(data_ew)
            #Se crean los vectores de H/V ponderando NS y EW y se guardan como listas de
    listas
            H_on_V_pond.append(np.sqrt((data_ns*data_ns + data_ew*data_ew)/2)/ data_z)
            #Ahora se crean los vectores de H/V de donde cada componente es individual y se
    guardan idem
            H_on_V_NS.append(data_ns/data_z)
            H_on_V_EW.append(data_ew/data_z)


        #Se obtienen las desviaciones estandar del logaritmo de componentes ponderadas
        des_H_on_V_pond_log=np.std(np.log(H_on_V_pond), axis=0)
        #Se obtiene el promedio de todas las componentes en la lista de analisis
        prom_H_on_V_pond_log=np.mean(np.log(H_on_V_pond), axis=0)
        #Se suma el logaritmo del promedio con la desviacion estandar
        mas_des_H_on_V_pond_log=prom_H_on_V_pond_log+des_H_on_V_pond_log
        menos_des_H_on_V_pond_log=prom_H_on_V_pond_log-des_H_on_V_pond_log
        mas_des_H_on_V_pond=np.exp(mas_des_H_on_V_pond_log)
        menos_des_H_on_V_pond=np.exp(menos_des_H_on_V_pond_log)
        prom_H_on_V_pond=np.mean(H_on_V_pond, axis=0)


        #Se obtiene las desviaciones estandar del logaritmo de la componente longitudinal
        des_H_on_V_NS_log=np.std(np.log(H_on_V_NS), axis=0)
        #Se obtiene el promedio de todas las componentes en la lista de analisis
        prom_H_on_V_NS_log=np.mean(np.log(H_on_V_NS), axis=0)
        #Se suma el logaritmo del promedio con la desviacion estandar
        mas_des_H_on_V_NS_log=prom_H_on_V_NS_log+des_H_on_V_NS_log
        menos_des_H_on_V_NS_log=prom_H_on_V_NS_log-des_H_on_V_NS_log
        mas_des_H_on_V_NS=np.exp(mas_des_H_on_V_NS_log)
        menos_des_H_on_V_NS=np.exp(menos_des_H_on_V_NS_log)
        prom_H_on_V_NS=np.mean(H_on_V_NS, axis=0)


        #Se obtiene las desviaciones estandar del logaritmo de la componente transversal
        des_H_on_V_EW_log=np.std(np.log(H_on_V_EW), axis=0)
        #Se obtiene el promedio de todas las componentes en la lista de analisis
        prom_H_on_V_EW_log=np.mean(np.log(H_on_V_EW), axis=0)
        #Se suma el logaritmo del promedio con la desviacion estandar
        mas_des_H_on_V_EW_log=prom_H_on_V_EW_log+des_H_on_V_EW_log
        menos_des_H_on_V_EW_log=prom_H_on_V_EW_log-des_H_on_V_EW_log
        mas_des_H_on_V_EW=np.exp(mas_des_H_on_V_EW_log)
        menos_des_H_on_V_EW=np.exp(menos_des_H_on_V_EW_log)
        prom_H_on_V_EW=np.mean(H_on_V_EW, axis=0)

```

```
223.      #Se procede a crear un arreglo de Trues y Flases para indexar los elementos.
224.      indice_filtro=frequency_rescued>0.5 #Frecuencia mínima a graficar
225.      frequency_rescued=frequency_rescued[indice_filtro]
226.      prom_H_on_V_pond=prom_H_on_V_pond[indice_filtro]
227.      prom_H_on_V_EW=prom_H_on_V_EW[indice_filtro]
228.      prom_H_on_V_NS=prom_H_on_V_NS[indice_filtro]
229.      des_H_on_V_pond_log=des_H_on_V_pond_log[indice_filtro]
230.      des_H_on_V_EW_log=des_H_on_V_EW_log[indice_filtro]
231.      des_H_on_V_NS_log=des_H_on_V_NS_log[indice_filtro]
232.      mas_des_H_on_V_EW=mas_des_H_on_V_EW[indice_filtro]
233.      menos_des_H_on_V_EW=menos_des_H_on_V_EW[indice_filtro]
234.      mas_des_H_on_V_NS=mas_des_H_on_V_NS[indice_filtro]
235.      menos_des_H_on_V_NS=menos_des_H_on_V_NS[indice_filtro]
236.      mas_des_H_on_V_pond=mas_des_H_on_V_pond[indice_filtro]
237.      menos_des_H_on_V_pond=menos_des_H_on_V_pond[indice_filtro]
238.
239.      #Se obtienen los datos de mayores amplificaciones, con su respectiva frecuencia
240.      #para un analisis mas fácil (iran directo a una interfaz de resultados)
241.      #Se obtiene la posicion iesima de donde está el valorde las amplificaciones para así
    llevarlo al vector de frecuencias
242.      i_H_on_V_pond= np.where(prom_H_on_V_pond==max(prom_H_on_V_pond))
243.      frec_pond=frequency_rescued[i_H_on_V_pond]
244.      frec_pond_round=round(frec_pond[0],2)
245.      amp_max_pond=round(max(mas_des_H_on_V_pond),2)
246.      amp_max_pond_grafico=round(max(mas_des_H_on_V_pond),0)
247.      amp_max_NS=round(max(prom_H_on_V_NS),2)
248.      amp_max_NS_grafico=round(max(mas_des_H_on_V_NS),0)
249.      amp_max_EW=round(max(prom_H_on_V_EW),2)
250.      amp_max_EW_grafico=round(max(mas_des_H_on_V_EW),0)
251.      i_H_on_V_NS= np.where(prom_H_on_V_NS==max(prom_H_on_V_NS))
252.      frec_NS=frequency_rescued[i_H_on_V_NS]
253.      frec_NS_round=round(frec_NS[0],2)
254.      i_H_on_V_EW= np.where(prom_H_on_V_EW==max(prom_H_on_V_EW))
255.      frec_EW=frequency_rescued[i_H_on_V_EW]
256.      frec_EW_round=round(frec_EW[0],2)
257.
258.      #Se crea una matriz con resultados donde Columna1= Frec[hz], Columna2= Prom. Pond,
    Columna3= Des. Est. Pond, Columna4= Prom. NS, Columna5= Des. Est. NS, Columna6= Prom. EW,
    Columna7= Des. Est. EW'
259.      Resultados=np.concatenate((frequency_rescued.reshape(len(frequency_rescued),-
    1),prom_H_on_V_pond.reshape(len(frequency_rescued),-
    1),des_H_on_V_pond_log.reshape(len(frequency_rescued),-
    1),prom_H_on_V_NS.reshape(len(frequency_rescued),-
    1),des_H_on_V_NS_log.reshape(len(frequency_rescued),-
    1),prom_H_on_V_EW.reshape(len(frequency_rescued),-
    1),des_H_on_V_EW_log.reshape(len(frequency_rescued),-1)),1)
260.      #Acá se guardan los gráficos de las ventanas de tiempo que se obtuvieron a partir de
    la los archivos mseed
261.      #Se procede a plotear los gráficos:
262.      if combinar==2:
263.          figpond=Figure(figsize=(9,9))
264.          ax=figpond.add_subplot(111)
265.          ax.semilogx(frequency_rescued,prom_H_on_V_pond,label='Analisis '+estacion+ '
    direcciones ponderadas',linewidth=1.5,color='brown')
266.          ax.semilogx(frequency_rescued,mas_des_H_on_V_pond,linestyle='--',color='brown')
267.          ax.semilogx(frequency_rescued,menos_des_H_on_V_pond,linestyle='--',color='brown')
268.          ax.set_title('Analisis H/V, estacion '+estacion+ ', hora inicio:
    '+str(hora_inicio)+', duracion analisis '+ str(tiempo_analisis)+'[seg], '+
    fecha,fontsize=13)
269.          ax.set_xlabel('Frecuencia [Hz]')
270.          ax.set_xlim(min(frequency_rescued),15)
271.          ax.set_ylim(0,amp_max_pond_grafico+2)
272.          ax.set_ylabel('HVSR')
273.          ax.legend(['Promedio de mediciones \n HVSR. máx='+str(amp_max_pond)+'\n Frec[Hz]
    ='+str(frec_pond_round)],loc='upper right')
274.          #Incluir grilla:
```

```
275.          ax.grid(True)
276.          #Configuracion de grilla:
277.          ax.grid(color = '0.5', linestyle = '-', linewidth =0.3)
278.          Graf_y_Resultados={'grafico':figpond,'resultados':Resultados,'Traza Z':Z, 'Traza
    NS':NS,'Traza EW':EW}
279.          #Se crea un diccionario con gráfico en primer lugar y resultados en segundo lugar
280.          return Graf_y_Resultados
281.
282.      if combinar == 1:
283.          fig=Figure(figsize=(9,9))
284.          axNS=fig.add_subplot(121)
285.          axNS.semilogx(frequency_rescued,prom_H_on_V_NS,label='Analisis '+estacion+ '
    direccion NS',linewidth=1.5,color='r')
286.          axNS.semilogx(frequency_rescued,mas_des_H_on_V_NS,linestyle='--',color='r')
287.          axNS.semilogx(frequency_rescued,menos_des_H_on_V_NS,linestyle='--',color='r')
288.          axNS.set_xlabel('Frequencia [Hz]')
289.          axNS.set_xlim(min(frequency_rescued),15)
290.          axNS.set_ylim(0,amp_max_NS_grafico+2)
291.          axNS.set_ylabel('HVSR')
292.          axNS.legend(['Promedio de mediciones \n HVSR. máx='+str(amp_max_NS)+'\n Frec[Hz]
    ='+str(frec_NS_round)],loc='upper right')
293.          #Incluir grilla:
294.          axNS.grid(True)
295.          #Configuracion de grilla:
296.          axNS.grid(color = '0.5', linestyle = '-', linewidth =0.3)
297.          #Figura EW
298.          axEW=fig.add_subplot(122)
299.          axEW.semilogx(frequency_rescued,prom_H_on_V_EW,label='Analisis de estacion
    '+estacion+ ' direccion EW',linewidth=1.5,color='b')
300.          axEW.semilogx(frequency_rescued,mas_des_H_on_V_EW,linestyle='--',color='b')
301.          axEW.semilogx(frequency_rescued,menos_des_H_on_V_EW,linestyle='--',color='b')
302.          axEW.set_xlabel('Frecuencia [Hz]')
303.          axEW.set_xlim(min(frequency_rescued),15)
304.          axEW.set_ylim(0,amp_max_EW_grafico+2)
305.          axEW.set_ylabel('HVSR')
306.          axEW.legend(['Promedio de mediciones \n HVSR. máx='+str(amp_max_EW)+'\n Frec[Hz]
    ='+str(frec_EW_round)],loc='upper right')
307.          #Incluir grilla:
308.          axEW.grid(True)
309.          #Configuracion de grilla:
310.          axEW.grid(color = '0.5', linestyle = '-', linewidth =0.3)
311.          fig.suptitle('Analisis HVSR longitudinal (rojo) y transversal (azul), estacion
    '+estacion +',hora inicio: '+str(hora_inicio)+', duracion analisis '+ str(tiempo_analisis)+'
    [seg], '+ fecha,fontsize=10)
312.          Graf_y_Resultados={'grafico':fig,'resultados':Resultados,'Traza Z':Z, 'Traza
    NS':NS,'Traza EW':EW}
313.          #Se crea un diccionario con gráfico en primer lugar y resultados en segundo lugar
314.          return Graf_y_Resultados
315.
316.
317.  #Esta función se usa para crear el archivo .txt de los datos graficados HVSR
318.  def crear_txt_HVSR(matriz):
319.      def wrapper():
320.          name = filedialog.asksaveasfilename(initialdir = ".",title = "Seleccionar
    archivo",filetypes = (("txt file","*.txt"),("all files","*.*")))
321.          if not name.endswith('.txt'):
322.              name+='.txt'
323.          np.savetxt(name,matriz,fmt='%.2e',header='Columna1= Frec[hz], Columna2= Prom.
    Pond, Columna3= Des. Est. Pond, Columna4= Prom. Longitudinal, Columna5= Des. Est.
    Longitudinal, Columna6= Prom. Transversal, Columna7= Des. Est. Transversal')
324.      return wrapper
325.
326.
327.  # Se define la funcion auxiliar HVSR, la cual permitirá mostrar los resultados en una
    ventana
328.  def H_on_V_aux():
```

```
329.
    dict_HV=H_on_V_ambiental(EstacionEntryHV.get(),int(TiempoEntryHV.get()),HoraEntryHV.get(),se
    lectedHV.get(),FechaEntryHV.get())
330.      fig=dict_HV['grafico']
331.      resultados=dict_HV['resultados']
332.      if fig is None:
333.          LabelError=Label(ventana_graf_HV,text='No existe información para procesar ):')
334.          LabelError.pack(side=TOP)
335.          return
336.      fig.set_size_inches(4.5, 3.8)
337.      traza_NS=dict_HV['Traza NS']
338.      traza_EW=dict_HV['Traza EW']
339.      traza_Z=dict_HV['Traza Z']
340.      #Se crea la ventana en donde se muestran los resultados
341.      ventana_graf_HV = Toplevel(raiz)
342.      ventana_graf_HV.iconbitmap('Señales.ico')
343.      ventana_graf_HV.title('Resultados HVSR')
344.      #Se crea el frame donde irán los botones
345.      FrameBotonesHV=Frame(ventana_graf_HV,width=200, height=200)
346.      FrameBotonesHV.pack(side=BOTTOM)
347.      BotonSaveFig=Button(FrameBotonesHV,text='Guardar gráfico',command=save_fig(fig))
348.      BotonSaveFig.pack(side=LEFT)
349.      BotonMapa=Button(FrameBotonesHV,text='Ver mapa',command=ventana_estacion_HV_mapa)
350.      BotonMapa.pack(side=RIGHT)
351.      BotonSaveDatos=Button(FrameBotonesHV,text='Guardar
    datos',command=crear_txt_HVSR(resultados))
352.      BotonSaveDatos.pack(side=RIGHT)
353.      #Figura del gráfico
354.      canvas_fig_HV = FigureCanvasTkAgg(fig, master= ventana_graf_HV)
355.      canvas_fig_HV.get_tk_widget().pack(side=TOP, fill=BOTH,expand=1)
356.      canvas_fig_HV.draw()
357.      #Frame para trazas
358.      frame_trazas = Frame(ventana_graf_HV,width=500, height=100)
359.      frame_trazas.pack()
360.      #Figura de trazas NS
361.      fig_NS=traza_NS.plot(size=(400,200), show=False )
362.      fig_NS.suptitle('Traza NS',fontsize=10)
363.      canvas_fig_NS = FigureCanvasTkAgg(fig_NS, master=frame_trazas)
364.      canvas_fig_NS.get_tk_widget().pack(side=LEFT, fill=BOTH,expand=1)
365.      canvas_fig_NS.draw()
366.      #Figura de trazas EW
367.      fig_EW=traza_EW.plot(size=(400,200), show=False ,number_of_ticks=3)
368.      fig_EW.suptitle('Traza EW',fontsize=10)
369.      canvas_fig_EW = FigureCanvasTkAgg(fig_EW, master=frame_trazas)
370.      canvas_fig_EW.get_tk_widget().pack(side=LEFT, fill=BOTH,expand=1)
371.      canvas_fig_EW.draw()
372.      #Figura traza Z
373.      fig_Z=traza_Z.plot(size=(400,200),show=False )
374.      fig_Z.suptitle('Traza Z',fontsize=10)
375.      canvas_fig_Z = FigureCanvasTkAgg(fig_Z, master=frame_trazas)
376.      canvas_fig_Z.get_tk_widget().pack(side=LEFT, fill=BOTH,expand=1)
377.      canvas_fig_Z.draw()
378.      #Se agrega la barra
379.      toolbar = NavigationToolbar2Tk(canvas_fig_HV, ventana_graf_HV)
380.      toolbar.update()
381.      return
382.
383.
384.
385.  #Funcion para mostrar en mapa la estación HV analizada.
386.  def ventana_estacion_HV_mapa():
387.      #Se carga la imagen png que sirve para marcar en el mapa
388.      marcador_rojo = Image.open('C:/Users/jose_/Script Tesis/Scripts
    desarrollados/marcador_rojo.png')
389.      #Se busca obtener el tamaño de la imagen para posterior manipulación
390.      ancho_mapa_HV, alto_mapa_HV = marcador_rojo.size
```

```
391.      marcador_rojo = marcador_rojo.resize(((ancho_mapa_HV//2)-resizex,(alto_mapa_HV//2)-
    resizey))
392.      #Se abre la imagen de la vista en planta de la red de geófonos
393.      background_HV = Image.open('C:/Users/jose_/Script Tesis/Scripts
    desarrollados/estacionesHV.jpg')
394.      #Se emplea la función ubicacion_en_el_mapa
395.      background_HV.paste(marcador_rojo,
    (ubicacion_en_el_mapa(archivo_coordenadas,EstacionEntryHV.get())[0]-errorx,
    ubicacion_en_el_mapa(archivo_coordenadas,EstacionEntryHV.get())[1]-errory), marcador_rojo)
396.      #Se guarda la imagen en formato .png para posterior manipulación de la imagen
397.      background_HV.save('NewImg_HV.png',"PNG")
398.      #Se abre la imagen en formato.png
399.      NewImg_HV = Image.open('NewImg_HV.png')
400.      # Se usa la imagen
401.      ventana_mapa_HV=Toplevel()
402.      ventana_mapa_HV.iconbitmap('Señales.ico')
403.      ventana_mapa_HV.title('Ubicación estación procesada')
404.      #Se convierte la imagen a elemento amigable con tkinter.
405.      tkimage_HV = ImageTk.PhotoImage(NewImg_HV)
406.      ventana_mapa_HV.tkimage= tkimage_HV #Para que tkinter no deseche la imagen.
407.      panel1_HV = Label(ventana_mapa_HV, image=tkimage_HV)
408.      panel1_HV.grid(row=0, column=2, sticky=E)
409.
410.
411.  #Muestra una ventana con la info de procesamiento de HVSR
412.  def InfoProcesamiento_HVSR():
413.
414.      info_procesamiento_HVSR=Toplevel(raiz)
415.      info_procesamiento_HVSR.title('Información de procesamiento HVSR')
416.      info_procesamiento_HVSR.iconbitmap('Señales.ico')
417.      Frame_Info=Frame(info_procesamiento_HVSR)
418.      Frame_Info.pack()
419.      #Tipo de filtro
420.      Label_Tipo_Filtro=Label(Frame_Info,text='Tipo de filtro',relief="ridge",width=18,
    height=2)
421.      Label_Tipo_Filtro.grid(row=0,column=0,sticky='n')
422.      Label_Filtro=Label(Frame_Info,text='Bandpass',relief="ridge",width=16, height=2)
423.      Label_Filtro.grid(row=0,column=1,sticky='n')
424.      #Frec min filtrado
425.      Label_Frec_Min_Filtro=Label(Frame_Info,text='Frec. min.
    filtrado',relief="ridge",width=18, height=2)
426.      Label_Frec_Min_Filtro.grid(row=1,column=0,sticky='n')
427.
    Label_Frec_Min_Filtro_valor=Label(Frame_Info,text=(str(frec_min_filtrado)+'[Hz]'),relief="ri
    dge",width=16, height=2)
428.      Label_Frec_Min_Filtro_valor.grid(row=1,column=1,sticky='n')
429.      #Frec max filtrado
430.      Label_Frec_Max_Filtro=Label(Frame_Info,text='Frec. max.
    filtrado',relief="ridge",width=18, height=2)
431.      Label_Frec_Max_Filtro.grid(row=2,column=0,sticky='n')
432.
    Label_Frec_Max_Filtro_valor=Label(Frame_Info,text=(str(frec_max_filtrado)+'[Hz]'),relief="ri
    dge",width=16, height=2)
433.      Label_Frec_Max_Filtro_valor.grid(row=2,column=1,sticky='n')
434.      #Ventana Tukey
435.      Ventana_tukey=Label(Frame_Info,text='Ventana tukey',relief="ridge",width=18, height=2)
436.      Ventana_tukey.grid(row=3,column=0,sticky='n')
437.      Ventana_tukey_valor=Label(Frame_Info,text=(ventana_tukey),relief="ridge",width=16,
    height=2)
438.      Ventana_tukey_valor.grid(row=3,column=1,sticky='n')
439.      #Suavizado de Konno y Ohmachi
440.      KyO=Label(Frame_Info,text='Suavizado de Konno \n y Ohmachi',relief="ridge",width=18,
    height=2)
441.      KyO.grid(row=4,column=0,sticky='n')
442.      KyO_valor=Label(Frame_Info,text=str(bandwith_KyO),relief="ridge",width=16, height=2)
443.      KyO_valor.grid(row=4,column=1,sticky='n')
```

```
444.      #Largo de ventana
445.      Largo_de_ventana=Label(Frame_Info,text='Largo de sub-
    ventanas',relief="ridge",width=18, height=2)
446.      Largo_de_ventana.grid(row=4,column=0,sticky='n')
447.
    Largo_de_ventana_valor=Label(Frame_Info,text=str(ventana)+'[seg]',relief="ridge",width=16,
    height=2)
448.      Largo_de_ventana_valor.grid(row=4,column=1,sticky='n')
449.
450.
451.  '----------------Funciones para SSR-------------------'
452.
453.
454.
455.
456.  #Esta función muestra los sismos y sus características del archivo
457.  #'archivo_eventos' (variable definida al comienzo)
458.  def ventana_todos_sismos():
459.      ventana_info_sismos=Toplevel()
460.      ventana_info_sismos.iconbitmap('Señales.ico')
461.      ventana_info_sismos.title('Eventos registrados')
462.      for j in range(int(columnas_del_archivo)):
463.          columna_archivo=np.loadtxt(archivo_eventos,dtype=str,delimiter=';',
464.                         usecols=j)
465.          frame_1=Frame(ventana_info_sismos,width=1000, height=700)
466.          for i in np.arange(len(columna_archivo)):
467.              Label_1 = Label(frame_1,
    text=str(columna_archivo[i]),borderwidth=1,relief="ridge",width=16,
468.            height=2)
469.              Label_1.grid(row=i,column=j,sticky='n')
470.          frame_1.pack(side='left')
471.
472.  #Esta función muestra los sismos con Mw mayor a la que se ingresa en la casilla Mw min
473.  def ventana_sismos_filtrados():
474.      ventana_info_sismos_filtrados=Toplevel()
475.      ventana_info_sismos_filtrados.iconbitmap('Señales.ico')
476.      ventana_info_sismos_filtrados.title('Sismos con Mw>'+str(Mw_minEntrySSR.get()))
477.      intensidades= np.loadtxt(archivo_eventos,dtype=str,delimiter=';',usecols=6)
478.      Mw_min =  Mw_minEntrySSR.get()
479.      for j in range(int(columnas_del_archivo)):
480.          columna_archivo=np.loadtxt(archivo_eventos,dtype=str,delimiter=';',usecols=j)
481.          frame_1=Frame(ventana_info_sismos_filtrados,width=1000, height=700)
482.          for i in np.arange(len(columna_archivo)):
483.              if intensidades[i]>Mw_min:
484.                  Label_1 = Label(frame_1,
    text=str(columna_archivo[i]),borderwidth=1,relief="ridge",width=16, height=2)
485.                  Label_1.grid(row=i,column=j,sticky='n')
486.                  frame_1.pack(side='left')
487.
488.
489.  # Esta función muestra los sismos que tienen en común 2 carpetas de estaciones
490.  #al ver si existe el archivo de la componente NS en ambas carpetas
491.  def sismos_en_comun():
492.      fecha_excel= np.loadtxt(archivo_eventos,dtype=str,delimiter=';',usecols=1,skiprows=1)
493.      estacion1=EstacionRefEntry.get()
494.      estacion2=EstacionEstudioEntry.get()
495.      path_station1=path+estacion1+'/'
496.      path_station2=path+estacion2+'/'
497.      daily_data1 = os.listdir(path_station1)
498.      daily_data2 = os.listdir(path_station2)
499.      eventos_1=[]
500.      eventos_2=[]
501.      eventos_comun=[0] #Se le asigna el 0 ya que mas adelante será necesario para el ciclo
    for de crear la ventana.
502.      for i in np.arange(len(fecha_excel)):
503.          año_excel=str(fecha_excel[i][2:4])
```

```
504.          mes_excel=str(fecha_excel[i][5:7])
505.          dia_excel=str(fecha_excel[i][8:10])
506.          hora_excel=int(fecha_excel[i][11:13])
507.          for file in daily_data1:
508.              if (file.startswith(año_excel+mes_excel+dia_excel,5)) and
    (int(file[11:13])<hora_excel) and (file.endswith('.pri1')):
509.                  eventos_1.append(i+1) #se le suma 1 ya que python trabaja desde el indice
    0, en caso de cumplir la condición el indice 0 corresponde al sismo 1
510.          for file in daily_data2:
511.              if (file.startswith(año_excel+mes_excel+dia_excel,5)) and
    (int(file[11:13])<hora_excel) and (file.endswith('.pri1')):
512.                  eventos_2.append(i+1)
513.      for num in eventos_1:
514.             if num in eventos_2:
515.                  eventos_comun.append(num)
516.      '-------------------------------------'
517.      ventana_info_sismos_comun=Toplevel(raiz)
518.      ventana_info_sismos_comun.iconbitmap('Señales.ico')
519.      ventana_info_sismos_comun.title('Sismos en común entre estaciones '+estacion1+' y
    '+estacion2)
520.      eventos=np.loadtxt(archivo_eventos,dtype=int,delimiter=';',usecols=0,skiprows=1)
521.      for j in range(int(columnas_del_archivo)):
522.          columna_archivo=np.loadtxt(archivo_eventos,dtype=str,delimiter=';',usecols=j)
523.          frame_1=Frame(ventana_info_sismos_comun,width=1000, height=700)
524.          for i in np.arange(len(eventos_comun)):
525.              if i == 0:
526.                  Label_1 = Label(frame_1,
    text=str(columna_archivo[i]),borderwidth=1,relief="ridge",width=16, height=2 )
527.                  Label_1.grid(row=i,column=j,sticky='n')
528.              if eventos_comun[i] in eventos:
529.                  Label_1 = Label(frame_1,
    text=str(columna_archivo[eventos_comun[i]]),borderwidth=1,relief="ridge",width=16, height=2
    )
530.                  Label_1.grid(row=i,column=j,sticky='n')
531.          frame_1.pack(side='left')
532.
533.
534.  # Esta función muestra los sismos que tienen en común 2 carpetas de estaciones
535.  #al ver si existe el archivo de la componente NS en ambas carpetas,
536.  #incorpora el filtro de magnitud Mw y agrega el botón correr
537.  #que realiza SSR entre las estaciones ingresadas
538.  def sismos_en_comun_filtrados():
539.      estacion1=EstacionRefEntry.get()
540.      estacion2=EstacionEstudioEntry.get()
541.      path_station1=path+estacion1+'/'
542.      daily_data1 = os.listdir(path_station1)
543.      path_station2=path+estacion2+'/'
544.      daily_data2 = os.listdir(path_station2)
545.      fecha_excel= np.loadtxt(archivo_eventos,dtype=str,delimiter=';',usecols=1,skiprows=1)
546.      eventos_1=[]
547.      eventos_2=[]
548.      eventos_comun=[0] #Se le asigna el 0 ya que mas adelante será necesario para el ciclo
    for de crear la ventana.
549.      Mw_min = Mw_minEntrySSR.get()
550.      if Mw_min == '':
551.          Mw_min=0
552.      Mw_min_float=float(Mw_min)
553.      for i in np.arange(len(fecha_excel)):
554.          año_excel=str(fecha_excel[i][2:4])
555.          mes_excel=str(fecha_excel[i][5:7])
556.          dia_excel=str(fecha_excel[i][8:10])
557.          hora_excel=int(fecha_excel[i][11:13])
558.          for file in daily_data1:
559.              if (file.startswith(año_excel+mes_excel+dia_excel,5)) and
    (int(file[11:13])<hora_excel) and (file.endswith('.pri1')):
```

```
                    eventos_1.append(i+1) #se le suma 1 ya que python trabaja desde el indice
    0, en caso de cumplir la condición el indice 0 corresponde al sismo 1
          for file in daily_data2:
              if (file.startswith(año_excel+mes_excel+dia_excel,5)) and
    (int(file[11:13])<hora_excel) and (file.endswith('.pri1')):
                    eventos_2.append(i+1)
      for num in eventos_1:
             if num in eventos_2:
                    eventos_comun.append(num)
      '-------------------------------------'
      ventana_info_sismos_comun_filtrados=Toplevel(raiz)
      ventana_info_sismos_comun_filtrados.iconbitmap('Señales.ico')
      ventana_info_sismos_comun_filtrados.title('Sismos en común entre estaciones
    '+estacion1+' y '+estacion2+', con con Mw>'+str(Mw_minEntrySSR.get()))
      eventos=np.loadtxt(archivo_eventos,dtype=int,delimiter=';',usecols=0,skiprows=1)

    intensidades=(np.loadtxt(archivo_eventos,dtype=str,delimiter=';',usecols=6,skiprows=0))
      for j in range(int(columnas_del_archivo)):
          columna_archivo=np.loadtxt(archivo_eventos,dtype=str,delimiter=';',usecols=j)
          frame_1=Frame(ventana_info_sismos_comun_filtrados,width=1000, height=700)
          for i in np.arange(len(eventos_comun)):
              if i == 0:
                  Label_1 = Label(frame_1,
    text=str(columna_archivo[i]),borderwidth=1,relief="ridge",width=16, height=2 )
                  Label_1.grid(row=i,column=j,sticky='n')
              if (eventos_comun[i] in eventos) and ((float(intensidades[i])>Mw_min_float)) :
                      Label_1 = Label(frame_1,
    text=str(columna_archivo[eventos_comun[i]]),borderwidth=1,relief="ridge",width=16, height=2
    )
                      Label_1.grid(row=i,column=j,sticky='n')
          frame_1.pack(side=LEFT)


          '------------------------------'
          # En este ciclo for es para agregar los checkbuttons
      for i in np.arange(len(eventos_comun)):
          if i == 0:
              Label_1 = Label(frame_1,
    text='Seleccionar',borderwidth=1,relief="ridge",width=16, height=2)
              Label_1.grid(row=i,column=12,sticky='n')
          if (eventos_comun[i] in eventos) and ((float(intensidades[i])>float(Mw_min))) :
              diccionario_int_vars['Evento '+str(i)]=IntVar()
              CheckButton= Checkbutton(frame_1,variable=diccionario_int_vars['Evento
    '+str(i)],onvalue=i, offvalue=0)
              CheckButton.grid(row=i, column=12,sticky='n')

      BotonCorrer=Button(ventana_info_sismos_comun_filtrados,text='Cálculo
    SSR',command=SSR_sismos)
      BotonCorrer.pack(side=RIGHT)

  #Esta función retorna la transformada de Fourier de las trazas estudiadas
  def TransFourier(estacion,tiempo_analisis,hora_inicio,fecha):
      estaciones =
    np.loadtxt(archivo_coordenadas,dtype=str,comments='Estaciones',delimiter=';',
                        usecols=0,skiprows=1)
      angulos =
    np.loadtxt(archivo_coordenadas,dtype=str,comments='Estaciones',delimiter=';',
                        usecols=3,skiprows=1)
      #Se define el angulo para trabajar (azimut)
      for i in np.arange(len(estaciones)):
          if estacion==estaciones[i]:
              angulo=float(angulos[i])
      path_station = path + estacion+'/'
      #Se obtienen los años y fechas en str para comparar con el archivo
      año_fecha=str(fecha[2:4])
      mes_fecha=str(fecha[5:7])
```

```
614.      dia_fecha=str(fecha[8:10])
615.      #Se crea la variable seg_inicio a partir de hora_inicio con la que se trabajará en
    segundos
616.      seg_inicio=int(hora_inicio[0:2])*3600+int(hora_inicio[3:5])*60+int(hora_inicio[6:8])
617.      Z=1
618.      #Vector de datos diarios de la estacion actual:
619.      daily_data = os.listdir(path_station)
620.      #Se obtienen los registros a analizar según lo pedido.
621.      for file in daily_data:
622.           #Se asigna el registro a la coordenada Z
623.          if (file.endswith('.pri0') and file.startswith(año_fecha+
624.                              mes_fecha+
625.                              dia_fecha,5)):
626.
627.
628.              Z = read(path_station+file)
629.
630.          #Se asigna el registro a la coordenada NS
631.          if (file.endswith('.pri1') and file.startswith(año_fecha+
632.                              mes_fecha+
633.                              dia_fecha,5)):
634.              NS= read(path_station+file)
635.
636.          #Se asigna el registro a la cordenada EW
637.          if (file.endswith('.pri2') and file.startswith(año_fecha+
638.                              mes_fecha+
639.                              dia_fecha,5)):
640.              EW= read(path_station+file)
641.      if Z==1:
642.          print ('No existen datos de esta fecha :(')
643.          return
644.      #Se procede a cortar la traza en el tiempo que se desea a analizar
645.
    Z.trim(Z[0].stats.starttime+seg_inicio,Z[0].stats.starttime+tiempo_analisis+seg_inicio)
646.
    NS.trim(NS[0].stats.starttime+seg_inicio,NS[0].stats.starttime+tiempo_analisis+seg_inicio)
647.
    EW.trim(EW[0].stats.starttime+seg_inicio,EW[0].stats.starttime+tiempo_analisis+seg_inicio)
648.
649.      #Se realiza el filtrado y detrends
650.      Z[0].detrend()
651.      NS[0].detrend()
652.      EW[0].detrend()
653.
654.      #Se asigna el registro a la coordenada Z
655.      Z[0].filter('bandpass', freqmin=frec_min_filtrado, freqmax=frec_max_filtrado, corners
    = 4, zerophase=True)
656.      NS[0].filter('bandpass', freqmin=frec_min_filtrado, freqmax=frec_max_filtrado, corners
    = 4, zerophase=True)
657.      EW[0].filter('bandpass', freqmin=frec_min_filtrado, freqmax=frec_max_filtrado, corners
    = 4, zerophase=True) #Se cambió 30 por 25
658.
659.      #Se procede a asignar la data a cada vector con el que se trabajará
660.      data_z=Z[0].data
661.      data_ns=NS[0].data
662.      data_ew=EW[0].data
663.
664.      #Se define el tukey
665.      windows=signal.tukey(len(data_z),0.05)
666.
667.      #Se multiplica la ventana tukey por
668.      data_tukey_z=windows*data_z
669.      data_tukey_ns=windows*data_ns
670.      data_tukey_ew=windows*data_ew
671.      data_z=data_tukey_z
672.      data_ns=data_tukey_ns
```

```
673.      data_ew=data_tukey_ew
674.
675.      #Se procede a proyectar las trazas NS y EW en función del angulo del talud
676.      data_ns_pr=data_ns*np.cos((angulo)*np.pi/180.0) - data_ew*np.sin((angulo)*np.pi/180.0)
677.      data_ew_pr=data_ns*np.sin((angulo)*np.pi/180.0) + data_ew*np.cos((angulo)*np.pi/180.0)
678.
679.
680.      #Se procede a realizar las transformadas de fourier
681.      data_transformed_z = np.abs(np.fft.rfft(data_z))
682.      frequency_rescued = np.fft.rfftfreq(len(data_z), d=1./Z[0].stats.sampling_rate)
683.      data_transformed_rescued_z = data_transformed_z
684.      data_transformed_ns = np.abs(np.fft.rfft(data_ns_pr))
685.      data_transformed_rescued_ns = data_transformed_ns
686.      data_transformed_ew = np.abs(np.fft.rfft(data_ew_pr))
687.      data_transformed_rescued_ew = data_transformed_ew
688.
689.
690.
691.
692.      #Se realiza el suavizado de Konno & Ohmachi
693.      data_smoothed_z =
    konno_ohmachi_smoothing(data_transformed_rescued_z,frequency_rescued,
694.
    bandwidth=bandwith_KyO,max_memory_usage=4096)
695.      data_smoothed_ns =
    konno_ohmachi_smoothing(data_transformed_rescued_ns,frequency_rescued,
696.
    bandwidth=bandwith_KyO,max_memory_usage=4096)
697.      data_smoothed_ew =
    konno_ohmachi_smoothing(data_transformed_rescued_ew,frequency_rescued,
698.
    bandwidth=bandwith_KyO,max_memory_usage=4096)
699.
700.      #Se vuelve a trabajar con su nombre original para mayor comodidad
701.      data_z=data_smoothed_z
702.      data_ns = data_smoothed_ns
703.      data_ew = data_smoothed_ew
704.      #Se crean los vectores de fourier ponderando NS y EW
705.      Fourier_pond=np.sqrt((data_ns*data_ns + data_ew*data_ew)/2)
706.      #Ahora se crean los vectores de fourier de donde cada componente es individual
707.      Fourier_NS=data_ns
708.      Fourier_EW=data_ew
709.
710.
711.
712.      #Se crea una matriz de resultados.
713.      Resultados_Fourier= np.concatenate((frequency_rescued.reshape(len(frequency_rescued),-
    1),data_z.reshape(len(frequency_rescued),-1),data_ns.reshape(len(frequency_rescued),-
    1),data_ew.reshape(len(frequency_rescued),-1)),1)
714.
715.      Res_y_Estacion={'resultados': Resultados_Fourier, 'estacion':estacion, 'Traza Z':Z,
    'Traza NS':NS,'Traza EW':EW}
716.      #Se retorna un diccionario con los resultados y la estación analizada
717.      return Res_y_Estacion
718.
719.
720.  #Esta función aplica SSR, retornando gráficos.
721.
722.  def SSR_sismos():
723.      dic_fun=diccionario_int_vars
724.      estacion1=EstacionRefEntry.get()
725.      estacion2=EstacionEstudioEntry.get()
726.      lista_sismos_a_procesar=[]
727.      fecha_excel= np.loadtxt(archivo_eventos,dtype=str,delimiter=';',usecols=1)
728.      hora_sismos=np.loadtxt(archivo_eventos,dtype=str,delimiter=';',usecols=11)
729.      resultados_z=[]
```

```
730.      resultados_ns=[]
731.      resultados_ew=[]
732.      frecuencias=[]
733.      dic_trazas_eventos.clear()
734.      for value in dic_fun.values():
735.          if value.get()!= 0:
736.
    TransFourier2=TransFourier(estacion2,tiempo_analisis_SSR,hora_sismos[value.get()],fecha_exce
    l[value.get()])
737.
    TransFourier1=TransFourier(estacion1,tiempo_analisis_SSR,hora_sismos[value.get()],fecha_exce
    l[value.get()])
738.              SSR_z=TransFourier2['resultados'][:,1]/TransFourier1['resultados'][:,1]
739.              frecuencias=TransFourier1['resultados'][:,0]
740.              resultados_z.append(SSR_z)
741.              SSR_ns=TransFourier2['resultados'][:,2]/TransFourier1['resultados'][:,2]
742.              resultados_ns.append(SSR_ns)
743.              SSR_ew=TransFourier2['resultados'][:,3]/TransFourier1['resultados'][:,3]
744.              resultados_ew.append(SSR_ew)
745.              dic_trazas_eventos['Evento '+str(value.get())]={
746.                  'Trazas Z ':{
747.                      'referencia':TransFourier1['Traza Z'],
748.                      'estudio':TransFourier2['Traza Z']}
749.                      ,
750.                  'Trazas NS ':{
751.                      'referencia':TransFourier1['Traza NS'],
752.                      'estudio':TransFourier2['Traza NS']}
753.                      ,
754.                  'Trazas EW ':{
755.                      'referencia':TransFourier1['Traza EW'],
756.                      'estudio':TransFourier2['Traza EW']}
757.              }
758.
759.      prom_z=np.mean(resultados_z, axis=0)
760.      prom_ew=np.mean(resultados_ew, axis=0)
761.      prom_ns=np.mean(resultados_ns, axis=0)
762.      indice_filtro=frecuencias>0.5 #Frecuencia mínima a graficar
763.      frecuencias=frecuencias[indice_filtro]
764.      prom_z=prom_z[indice_filtro]
765.      prom_ew=prom_ew[indice_filtro]
766.      prom_ns=prom_ns[indice_filtro]
767.      #Se obtienen los datos de mayores SSR, con su respectiva frecuencia para un analisis
    mas fácil (iran directo a una interfaz de resultados)
768.      #Para la componente z
769.      i_SSR_z= np.where(prom_z==max(prom_z)) #Se obtiene la posicion iesima de donde está el
    valorde las amplificaciones para así llevarlo al vector de frecuencias
770.      frec_z=frecuencias[i_SSR_z]
771.      SSR_z_max=round(max(prom_z),2)
772.      SSR_z_max_grafico=round(max(prom_z),0)+2
773.      frec_z_round=round(frec_z[0],2)
774.
775.      #Para la componente ns
776.      i_SSR_ns= np.where(prom_ns==max(prom_ns)) #Se obtiene la posicion iesima de donde está
    el valorde las amplificaciones para así llevarlo al vector de frecuencias
777.      frec_ns=frecuencias[i_SSR_ns]
778.      SSR_ns_max=round(max(prom_ns),2)
779.      SSR_ns_max_grafico=round(max(prom_ns),0)+2
780.      frec_ns_round=round(frec_ns[0],2)
781.
782.      #Para la componente z
783.      i_SSR_ew= np.where(prom_ew==max(prom_ew)) #Se obtiene la posicion iesima de donde está
    el valorde las amplificaciones para así llevarlo al vector de frecuencias
784.      frec_ew=frecuencias[i_SSR_ew]
785.      SSR_ew_max=round(max(prom_ew),2)
786.      SSR_ew_max_grafico=round(max(prom_ew),0)+2
787.      frec_ew_round=round(frec_ew[0],2)
```

```
788.
789.      resultados_SSR=np.concatenate((frecuencias.reshape(len(frecuencias),-
    1),prom_z.reshape(len(frecuencias),-1),prom_ns.reshape(len(frecuencias),-
    1),prom_ew.reshape(len(frecuencias),-1)),1)
790.      # Se procede a plotear
791.      fig = plt.figure(figsize=[18,6])
792.      axNS = fig.add_subplot(132)
793.      axEW = fig.add_subplot(133)
794.      axZ = fig.add_subplot(131)
795.
796.      #se procede a plotear los promedios
797.      axZ.semilogx(frecuencias,prom_z,linewidth =1.5 , color='r')
798.      axNS.semilogx(frecuencias,prom_ns,linewidth =1.5 , color='b')
799.      axEW.semilogx(frecuencias,prom_ew,linewidth =1.5 , color='g')
800.      #Se añade los valores máximos SSR y su frecuencia.
801.      axEW.legend(['Promedio de mediciones \n SSR. máx='+str(SSR_ew_max)+'\n Frec[Hz]
    ='+str(frec_ew_round)],loc='upper right')
802.      axNS.legend(['Promedio de mediciones \n SSR. máx='+str(SSR_ns_max)+'\n Frec[Hz]
    ='+str(frec_ns_round)],loc='upper right')
803.      axZ.legend(['Promedio de mediciones \n SSR. máx='+str(SSR_z_max)+'\n Frec[Hz]
    ='+str(frec_z_round)],loc='upper right')
804.
805.      #Se plotea cada SSR de cada sismo
806.      for espectroZ in resultados_z:
807.          espectroZ=espectroZ[indice_filtro]
808.          axZ.semilogx(frecuencias,espectroZ,linestyle='--',color='grey')
809.      for espectroNS in resultados_ns:
810.          espectroNS=espectroNS[indice_filtro]
811.          axNS.semilogx(frecuencias,espectroNS,linestyle='--',color='grey')
812.      for espectroEW in resultados_ew:
813.          espectroEW=espectroEW[indice_filtro]
814.          axEW.semilogx(frecuencias,espectroEW,linestyle='--',color='grey')
815.
816.
817.      # Se procede a configurar cada gráfico
818.      # Configuración del eje X
819.      axEW.set_xlim(min(frecuencias),15)
820.      axEW.set_xlabel('Frecuencia[Hz]')
821.      axNS.set_xlim(min(frecuencias),15)
822.      axNS.set_xlabel('Frecuencia[Hz]')
823.      axZ.set_xlim(min(frecuencias),15)
824.      axZ.set_xlabel('Frecuencia[Hz]')
825.      # Configuración del eje Y
826.      axEW.set_ylim(0,SSR_ew_max_grafico)
827.      axEW.set_ylabel('SSR')
828.      axNS.set_ylim(0,SSR_ns_max_grafico)
829.      axNS.set_ylabel('SSR')
830.      axZ.set_ylim(0,SSR_z_max_grafico)
831.      axZ.set_ylabel('SSR')
832.      #Incluir grilla:
833.      axEW.grid(True)
834.      axNS.grid(True)
835.      axZ.grid(True)
836.      #Configuracion de grilla:
837.      axEW.grid(color = '0.5', linestyle = '-', linewidth =0.3)
838.      axNS.grid(color = '0.5', linestyle = '-', linewidth =0.3)
839.      axZ.grid(color = '0.5', linestyle = '-', linewidth =0.3)
840.      #Se añade título a cada gráfico
841.      axEW.set_title('SSR Transversal')
842.      axNS.set_title('SSR Longitudinal')
843.      axZ.set_title('SSR Z')
844.
845.
846.      #Se crea la ventana
847.      ventana_SSR_procesado=Toplevel(raiz)
848.      ventana_SSR_procesado.iconbitmap('Señales.ico')
```

```
849.      ventana_SSR_procesado.title('Resultados SSR')
850.       #Figura del gráfico
851.      canvas_fig_SSR = FigureCanvasTkAgg(fig, master= ventana_SSR_procesado)
852.      canvas_fig_SSR.get_tk_widget().pack(side=TOP, fill=BOTH,expand=1)
853.      canvas_fig_SSR.draw()
854.      #Botones para interfaz resultados SSR
855.      FrameBotonesSSR=Frame(ventana_SSR_procesado,width=200, height=200)
856.      FrameBotonesSSR.pack(side=BOTTOM)
857.      BotonSaveFig=Button(FrameBotonesSSR,text='Guardar gráfico',command=save_fig(fig))
858.      BotonSaveFig.pack(side=LEFT)
859.      BotonMapa=Button(FrameBotonesSSR,text='Ver mapa',command=ventana_estacion_SSR_mapa)
860.      BotonMapa.pack(side=RIGHT)
861.      BotonSaveDatos=Button(FrameBotonesSSR,text='Guardar
    datos',command=crear_txt_SSR(resultados_SSR))
862.      BotonSaveDatos.pack(side=RIGHT)
863.      BotonVerTrazasZ=Button(FrameBotonesSSR,text='Ver trazas Z',command=ver_trazas_Z_SSR)
864.      BotonVerTrazasZ.pack(side=RIGHT)
865.      BotonVerTrazasNS=Button(FrameBotonesSSR,text='Ver trazas
    longitudinales',command=ver_trazas_NS_SSR)
866.      BotonVerTrazasNS.pack(side=RIGHT)
867.      BotonVerTrazasEW=Button(FrameBotonesSSR,text='Ver traza
    transversales',command=ver_trazas_EW_SSR)
868.      BotonVerTrazasEW.pack(side=RIGHT)
869.
870.
871.      #Se agrega toolbar
872.      toolbar = NavigationToolbar2Tk(canvas_fig_SSR, ventana_SSR_procesado)
873.      toolbar.update()
874.
875.
876.      return
877.
878.  #Muestra una ventana con la info de procesamiento de SSR
879.  def InfoProcesamiento_SSR():
880.
881.      info_procesamiento_SSR=Toplevel(raiz)
882.      info_procesamiento_SSR.title('Información de procesamiento SSR')
883.      info_procesamiento_SSR.iconbitmap('Señales.ico')
884.      Frame_Info=Frame(info_procesamiento_SSR)
885.      Frame_Info.pack()
886.      #Tipo de filtro
887.      Label_Tipo_Filtro=Label(Frame_Info,text='Tipo de filtro',relief="ridge",width=16,
    height=2)
888.      Label_Tipo_Filtro.grid(row=0,column=0,sticky='n')
889.      Label_Filtro=Label(Frame_Info,text='Bandpass',relief="ridge",width=16, height=2)
890.      Label_Filtro.grid(row=0,column=1,sticky='n')
891.      #Frec min filtrado
892.      Label_Frec_Min_Filtro=Label(Frame_Info,text='Frec. min.
    filtrado',relief="ridge",width=16, height=2)
893.      Label_Frec_Min_Filtro.grid(row=1,column=0,sticky='n')
894.
    Label_Frec_Min_Filtro_valor=Label(Frame_Info,text=(str(frec_min_filtrado)+'[Hz]'),relief="ri
    dge",width=16, height=2)
895.      Label_Frec_Min_Filtro_valor.grid(row=1,column=1,sticky='n')
896.      #Frec max filtrado
897.      Label_Frec_Max_Filtro=Label(Frame_Info,text='Frec. max.
    filtrado',relief="ridge",width=16, height=2)
898.      Label_Frec_Max_Filtro.grid(row=2,column=0,sticky='n')
899.
    Label_Frec_Max_Filtro_valor=Label(Frame_Info,text=(str(frec_max_filtrado)+'[Hz]'),relief="ri
    dge",width=16, height=2)
900.      Label_Frec_Max_Filtro_valor.grid(row=2,column=1,sticky='n')
901.      #Ventana Tukey
902.      Ventana_tukey=Label(Frame_Info,text='Ventana tukey',relief="ridge",width=16, height=2)
903.      Ventana_tukey.grid(row=3,column=0,sticky='n')
```

```
904.      Ventana_tukey_valor=Label(Frame_Info,text=(ventana_tukey),relief="ridge",width=16,
    height=2)
905.      Ventana_tukey_valor.grid(row=3,column=1,sticky='n')
906.      #Suavizado de Konno y Ohmachi
907.      KyO=Label(Frame_Info,text='Suavizado de Konno \n y Ohmachi',relief="ridge",width=16,
    height=2)
908.      KyO.grid(row=4,column=0,sticky='n')
909.      KyO_valor=Label(Frame_Info,text=str(bandwith_KyO),relief="ridge",width=16, height=2)
910.      KyO_valor.grid(row=4,column=1,sticky='n')
911.
912.
913.
914.  # Esta función marca el mapa con las estaciones que se
915.  #realizó el SSR
916.  def ventana_estacion_SSR_mapa():
917.      marcador_rojo = Image.open('C:/Users/jose_/Script Tesis/Scripts
    desarrollados/marcador_rojo.png')
918.      marcador_verde = Image.open('C:/Users/jose_/Script Tesis/Scripts
    desarrollados/marcador_verde.png')
919.      ancho_marcador_SSR, alto_marcador_SSR = marcador_rojo.size
920.      marcador_rojo = marcador_rojo.resize(((ancho_marcador_SSR//2)-
    resizex,(alto_marcador_SSR//2)-resizey))
921.      marcador_verde = marcador_verde.resize(((ancho_marcador_SSR//2)-
    resizex,(alto_marcador_SSR//2)-resizey))
922.      background_SSR = Image.open('C:/Users/jose_/Script Tesis/Scripts
    desarrollados/estacionesSSR.jpg')
923.      background_SSR.paste(marcador_rojo,
    (ubicacion_en_el_mapa(archivo_coordenadas,EstacionEstudioEntry.get())[0]-errorx,
    ubicacion_en_el_mapa(archivo_coordenadas,EstacionEstudioEntry.get())[1]-errory),
    marcador_rojo)
924.      background_SSR.paste(marcador_verde,
    (ubicacion_en_el_mapa(archivo_coordenadas,EstacionRefEntry.get())[0]-errorx,
    ubicacion_en_el_mapa(archivo_coordenadas,EstacionRefEntry.get())[1]-errory), marcador_verde)
925.
926.      background_SSR.save('NewImg_SSR.png',"PNG")
927.      NewImg_SSR = Image.open('NewImg_SSR.png')
928.      # Use Image
929.      ventana_mapa_SSR=Toplevel()
930.      ventana_mapa_SSR.iconbitmap('Señales.ico')
931.      ventana_mapa_SSR.title('Ubicación de las estaciones procesadas')
932.      tkimage_SSR = ImageTk.PhotoImage(NewImg_SSR)
933.      ventana_mapa_SSR.tkimage= tkimage_SSR #Para que tkinter no deseche la imagen.
934.      panel1_SSR = Label(ventana_mapa_SSR, image=tkimage_SSR)
935.      panel1_SSR.grid(row=0, column=2, sticky=E)
936.
937.  # Crea el archivo .txt de SSR
938.  def crear_txt_SSR(matriz):
939.      def wrapper():
940.          name = filedialog.asksaveasfilename(initialdir = ".",title = "Seleccionar
    archivo",filetypes = (("txt file","*.txt"),("all files","*.*")))
941.          if not name.endswith('.txt'):
942.              name+='.txt'
943.          np.savetxt(name,matriz,fmt='%.2e',header='Columna1= Frec[hz], Columna2= Prom. Z,
    Columna3= Prom. NS, Columna4= Prom. EW')
944.      return wrapper
945.
946.
947.
948.
949.  #Esta función muestra las trazas que fueron procesadas en la dirección Z
950.  def ver_trazas_Z_SSR():
951.      ventana_trazas_Z=Toplevel()
952.      ventana_trazas_Z.title('Trazas Z procesadas')
953.
954.      wrapperZ=LabelFrame(ventana_trazas_Z)
955.
```

```
956.      mycanvasZ=Canvas(wrapperZ)
957.      mycanvasZ.pack(side=LEFT)
958.
959.      yscrollbarZ=Scrollbar(wrapperZ,orient='vertical',command=mycanvasZ.yview)
960.      yscrollbarZ.pack(side=RIGHT,fill='y')
961.      mycanvasZ.configure(yscrollcommand=yscrollbarZ.set)
962.      mycanvasZ.bind('<Configure>', lambda
    e:mycanvasZ.configure(scrollregion=mycanvasZ.bbox('all')))
963.      myframeZ=Frame(mycanvasZ,width=500, height=500)
964.      mycanvasZ.create_window((0,0),window=myframeZ,anchor='nw')
965.
966.
967.      wrapperZ.pack(fill='both',expand='yes')
968.
969.
970.      for eventos,dic_direcciones in dic_trazas_eventos.items():
971.          for direccion, estaciones in dic_direcciones.items():
972.              if direccion == 'Trazas Z ':
973.                  for estacion, trazas in estaciones.items():
974.                      for traza in trazas:
975.                          fig_Z=traza.plot(size=(400,200),show=False)
976.                          fig_Z.suptitle('Traza Z estación '+estacion+'
    '+eventos,fontsize=10)
977.                          #fig_Z.set_size_inches(4.1, 1.5)
978.                          canvas_fig_Z = FigureCanvasTkAgg(fig_Z, master=myframeZ)
979.                          canvas_fig_Z.get_tk_widget().pack(side=TOP, fill=BOTH,expand=1)
980.                          canvas_fig_Z.draw()
981.
982.      ventana_trazas_Z.resizable(False,False)
983.
984.  #Esta función muestra las trazas que fueron procesadas en la dirección NS
985.  def ver_trazas_NS_SSR():
986.      ventana_trazas_NS=Toplevel()
987.      ventana_trazas_NS.title('Trazas longitudinales procesadas')
988.
989.      wrapperNS=LabelFrame(ventana_trazas_NS)
990.
991.      mycanvasNS=Canvas(wrapperNS)
992.      mycanvasNS.pack(side=LEFT)
993.
994.      yscrollbarNS=Scrollbar(wrapperNS,orient='vertical',command=mycanvasNS.yview)
995.      yscrollbarNS.pack(side=RIGHT,fill='y')
996.      mycanvasNS.configure(yscrollcommand=yscrollbarNS.set)
997.      mycanvasNS.bind('<Configure>', lambda
    e:mycanvasNS.configure(scrollregion=mycanvasNS.bbox('all')))
998.      myframeNS=Frame(mycanvasNS,width=500, height=500)
999.      mycanvasNS.create_window((0,0),window=myframeNS,anchor='nw')
1000.
1001.
1002.     wrapperNS.pack(fill='both',expand='yes')
1003.
1004.
1005.     for eventos,dic_direcciones in dic_trazas_eventos.items():
1006.         for direccion, estaciones in dic_direcciones.items():
1007.             if direccion == 'Trazas NS ':
1008.                 for estacion, trazas in estaciones.items():
1009.                     for traza in trazas:
1010.                         fig_NS=traza.plot(size=(400,200),show=False)
1011.                         fig_NS.suptitle('Traza NS estación '+estacion+'
    '+eventos,fontsize=10)
1012.                         #fig_Z.set_size_inches(4.1, 1.5)
1013.                         canvas_fig_NS = FigureCanvasTkAgg(fig_NS, master=myframeNS)
1014.                         canvas_fig_NS.get_tk_widget().pack(side=TOP, fill=BOTH,expand=1)
1015.                         canvas_fig_NS.draw()
1016.
1017.     ventana_trazas_NS.resizable(False,False)
```

```
1018.
1019.
1020. #Esta función muestra las trazas que fueron procesadas en la dirección EW
1021. def ver_trazas_EW_SSR():
1022.     ventana_trazas_EW=Toplevel()
1023.     ventana_trazas_EW.title('Trazas transversales procesadas')
1024.
1025.     wrapperEW=LabelFrame(ventana_trazas_EW)
1026.
1027.     mycanvasEW=Canvas(wrapperEW)
1028.     mycanvasEW.pack(side=LEFT)
1029.
1030.     yscrollbarEW=Scrollbar(wrapperEW,orient='vertical',command=mycanvasEW.yview)
1031.     yscrollbarEW.pack(side=RIGHT,fill='y')
1032.     mycanvasEW.configure(yscrollcommand=yscrollbarEW.set)
1033.     mycanvasEW.bind('<Configure>', lambda
    e:mycanvasEW.configure(scrollregion=mycanvasEW.bbox('all')))
1034.     myframeEW=Frame(mycanvasEW,width=500, height=500)
1035.     mycanvasEW.create_window((0,0),window=myframeEW,anchor='nw')
1036.
1037.
1038.     wrapperEW.pack(fill='both')
1039.
1040.
1041.
1042.
1043.     for eventos,dic_direcciones in dic_trazas_eventos.items():
1044.         for direccion, estaciones in dic_direcciones.items():
1045.             if direccion == 'Trazas EW ':
1046.                 for estacion, trazas in estaciones.items():
1047.                     for traza in trazas:
1048.                         fig_EW=traza.plot(size=(400,200),show=False)
1049.                         fig_EW.suptitle('Traza EW estación '+estacion+'
    '+eventos,fontsize=10)
1050.                         #fig_Z.set_size_inches(4.1, 1.5)
1051.                         canvas_fig_EW = FigureCanvasTkAgg(fig_EW, master=myframeEW)
1052.                         canvas_fig_EW.get_tk_widget().pack(side=TOP, fill=BOTH,expand=1)
1053.                         canvas_fig_EW.draw()
1054.
1055.     ventana_trazas_EW.resizable(False,False)
1056.
1057.
1058. '----------------Funciones para SSR y HVSR-----------------------'
1059. #Función para guardar gráficos
1060. def save_fig(fig):
1061.     def wrapper():
1062.         name = filedialog.asksaveasfilename(initialdir = ".",title = "Seleccionar
    archivo",filetypes = (("png files","*.png"),("all files","*.*")))
1063.         if not name.endswith('.png'):
1064.             name+='.png'
1065.         fig.savefig(name)
1066.     return wrapper
1067.
1068. #Se define la función para ubicar la estación en la imagen del tranque a traves del path
    Archivo_coordenadas, esta funcion
1069. #sirve para HV como para SSR.
1070. def ubicacion_en_el_mapa(archivo_coordenadas,estacion):
1071.     datosY = np.loadtxt(archivo_coordenadas,dtype=int,comments='Coordenada
    Y',delimiter=';',
1072.                         usecols=2,skiprows=1)
1073.     datosX = np.loadtxt(archivo_coordenadas,dtype=int,comments='Coordenada
    X',delimiter=';',
1074.                         usecols=1,skiprows=1)
1075.     estaciones =
    np.loadtxt(archivo_coordenadas,dtype=str,comments='Estaciones',delimiter=';',
1076.                         usecols=0,skiprows=1)
```

```
1077.
1078.     for i in np.arange(len(estaciones)):
1079.         if estacion==estaciones[i]:
1080.             pixel_x=datosX[i]
1081.             pixel_y=datosY[i]
1082.     return (pixel_x,pixel_y)
1083.
1084. #Función que abre una ventana con la información acerca de...
1085. def AcercaDe():
1086.
1087.     messagebox.showinfo('Acerca de...','Código desarrollado por estudiante de ingeniería
    José Arce Beltrán, bajo la supervición del Profesor César Pastén Puchi Phd. Las trazas
    sísmicas empleadas en el presente trabajo provienen de una red de geófonos instalados en
    tranque de relaves "El Torito", durante campaña geofísica realizada el 2018. Esta abarca el
    tranque tanto en la base como en el coronamiento, usando un total de 28 estaciones.')
1088. #Función que abre una ventana que muestra correo frente a dudas
1089. def AyudaMenu():
1090.
1091.     messagebox.showinfo('Ayuda','Frente a consultas sobre el código desarrollado enviar
    correo a jose.arce@ug.uchile.cl')
1092.
1093.
1094. #-Interfaz main.
1095.
1096. #----Se crea la ventana de Interfaz
1097. raiz=Tk()
1098. #Comando para cambiar el tamaño de la interfaz
1099. raiz.resizable(1,1)
1100. #----Se le da un nombre a la interfaz
1101. raiz.title('Analisis SSR y HVSR')
1102. raiz.iconbitmap('Señales.ico')
1103. #----Se adorna con imagen FCFM
1104.
1105. FrameImg=Frame(raiz,width=520, height=160)
1106. FrameImg.pack()
1107. NewImg_FCFM = Image.open('Fcfm.png')
1108. ancho_img_FCFM, alto_img_FCFM = NewImg_FCFM.size
1109. NewImg_FCFM = NewImg_FCFM.resize(((ancho_img_FCFM//2),(alto_img_FCFM//2)))
1110. tkimage_FCFM = ImageTk.PhotoImage(NewImg_FCFM)
1111. raiz.tkimage= tkimage_FCFM #Para que tkinter no deseche la imagen.
1112. panel1_FCFM = Label(FrameImg, image=tkimage_FCFM)
1113. panel1_FCFM.grid(row=0, column=0, sticky=E)
1114.
1115. #----Se pregunta que tipo de analisis se quiere realizar y se inserta DIRECTO EN RAIZ
1116. AnalisisLabel=Label(raiz, text="¿Que tipo de analisis desea
    realizar?",font=('Hervatica',12))
1117. AnalisisLabel.pack()
1118.
1119. #----Tamaño del frame donde se insertarán los botones
1120. FrameBotones=Frame(raiz,width=100, height=100)
1121. FrameBotones.pack()
1122.
1123. #Se crean las funciones para crear ventanas acorde a la selección del usuario
1124. #-Se crea la ventana SSR
1125. def ventana_SSR():
1126.     global Mw_minEntrySSR
1127.     global EstacionRefEntry
1128.     global EstacionEstudioEntry
1129.     ventana_nueva_SSR = Toplevel(raiz)             #Se crea nueva ventana
1130.     ventana_nueva_SSR.title('Análisis SSR') #Se añade título a la ventana SSR
1131.     ventana_nueva_SSR.iconbitmap('Señales.ico')
1132.     #----Se adorna con imagen FCFM
1133.
1134.     FrameImg_SSR=Frame(ventana_nueva_SSR,width=100, height=80)
1135.     FrameImg_SSR.pack()
1136.     NewImg_SSR = Image.open('adorno_SSR.png')
```

```
1137.     ancho_img_SSR, alto_img_SSR = NewImg_SSR.size
1138.     NewImg_SSR = NewImg_SSR.resize(((ancho_img_SSR//2),(alto_img_SSR//2)))
1139.     tkimage_SSR = ImageTk.PhotoImage(NewImg_SSR)
1140.     ventana_nueva_SSR.tkimage= tkimage_SSR #Para que tkinter no deseche la imagen.
1141.     panel1_SSR = Label(FrameImg_SSR, image=tkimage_SSR)
1142.     panel1_SSR.grid(row=0, column=0, sticky=E)
1143.
1144.     #Se crea el menú de SSR
1145.     #Primero, la sección ayuda
1146.     barraMenuSSR= Menu(ventana_nueva_SSR)
1147.     ventana_nueva_SSR.config(menu=barraMenuSSR)
1148.     AyudaMenuSSR=Menu(barraMenuSSR,tearoff=0)
1149.     barraMenuSSR.add_cascade(label='Ayuda',menu=AyudaMenuSSR)
1150.     AyudaMenuSSR.add_command(label='Ayuda',command=AyudaMenu)
1151.     InfoMenuSSR=Menu(barraMenuSSR,tearoff=0)
1152.     barraMenuSSR.add_cascade(label='Información',menu=InfoMenuSSR)
1153.     InfoMenuSSR.add_command(label='Información de
    procesamiento',command=InfoProcesamiento_SSR)
1154.     InfoMenuSSR.add_command(label='Acerca de...',command=AcercaDe)
1155.
1156.
1157.
1158.
1159.     FrameSSR=Frame(ventana_nueva_SSR,width=500, height=500)
1160.     FrameSSR.pack(side=LEFT)
1161.     #-Se crean las viñetas para las estaciones a comparar
1162.     #---Para la estacion de referencia
1163.     EstacionRefEntry=Entry(FrameSSR)
1164.     EstacionRefEntry.grid(row=1, column=1)
1165.     EstacionRefLabel=Label(FrameSSR, text='Estacion de referencia')
1166.     EstacionRefLabel.grid(row=1, column=0)
1167.     #---Para la estacion a estudiar
1168.     EstacionEstudioEntry=Entry(FrameSSR)
1169.     EstacionEstudioEntry.grid(row=0, column=1)
1170.     EstacionEstudioLabel=Label(FrameSSR, text='Estacion a estudiar')
1171.     EstacionEstudioLabel.grid(row=0,column=0)
1172.     #---Mw_min por intensidad
1173.     Mw_minLabel=Label(FrameSSR, text='Mw mínimo')
1174.     Mw_minLabel.grid(row=2,column=0)
1175.     Mw_minEntrySSR=Entry(FrameSSR)
1176.     Mw_minEntrySSR.grid(row=2, column=1)
1177.     FrameBotonesSSR=Frame(ventana_nueva_SSR,width=500, height=500)
1178.     FrameBotonesSSR.pack(side=RIGHT)
1179.     Boton_Sismos_SSR=Button(FrameBotonesSSR,text='Ver todos los
    sismos',command=ventana_todos_sismos) #si agrego en el codigo esto: command=codigoBoton,
    llamo a la funcion codigo boton
1180.     Boton_Sismos_SSR.pack()
1181.     Boton_Sismos_Filtrados=Button(FrameBotonesSSR,text='Ver sismos Mw > Mw
    min',command=ventana_sismos_filtrados) #si agrego en el codigo esto: command=codigoBoton,
    llamo a la funcion codigo boton
1182.     Boton_Sismos_Filtrados.pack()
1183.     Boton_Sismos_Comun=Button(FrameBotonesSSR,text='Ver sismos en
    común',command=sismos_en_comun) #si agrego en el codigo esto: command=codigoBoton, llamo a
    la funcion codigo boton
1184.     Boton_Sismos_Comun.pack()
1185.     Boton_Sismos_Comun_filtrados=Button(FrameBotonesSSR,text='Ver sismos en común con Mw >
    Mw min',command=sismos_en_comun_filtrados) #si agrego en el codigo esto:
    command=codigoBoton, llamo a la funcion codigo boton
1186.     Boton_Sismos_Comun_filtrados.pack()
1187.
1188. #---Se crea la ventana HV
1189. def ventana_HV():
1190.     global EstacionEntryHV
1191.     global FechaEntryHV
1192.     global HoraEntryHV
1193.     global TiempoEntryHV
```

```
1194.     global selectedHV
1195.     ventana_nueva_HV = Toplevel(raiz)             #Se crea nueva ventana
1196.     ventana_nueva_HV.title('Análisis HVSR')     #Se añade título a la ventana HV
1197.     ventana_nueva_HV.iconbitmap('Señales.ico')
1198.     #Se adorna la interfaz HVSR
1199.     #---Se adorna con imagen FCFM
1200.
1201.     FrameImg_HVSR=Frame(ventana_nueva_HV,width=100, height=80)
1202.     FrameImg_HVSR.pack()
1203.     NewImg_HVSR = Image.open('adorno_HVSR.png')
1204.     ancho_img_HVSR, alto_img_HVSR = NewImg_HVSR.size
1205.     NewImg_HVSR = NewImg_HVSR.resize(((ancho_img_HVSR//2),(alto_img_HVSR//2)))
1206.     tkimage_HVSR = ImageTk.PhotoImage(NewImg_HVSR)
1207.     ventana_nueva_HV.tkimage= tkimage_HVSR #Para que tkinter no deseche la imagen.
1208.     panel1_HVSR = Label(FrameImg_HVSR, image=tkimage_HVSR)
1209.     panel1_HVSR.grid(row=0, column=0, sticky=E)
1210.     #-Se crea el menú de HVSR
1211.     barraMenuHV= Menu(ventana_nueva_HV)
1212.     ventana_nueva_HV.config(menu=barraMenuHV)
1213.     #---Primero, la sección ayuda
1214.     AyudaMenuHV=Menu(barraMenuHV,tearoff=0)
1215.     barraMenuHV.add_cascade(label='Ayuda',menu=AyudaMenuHV)
1216.     AyudaMenuHV.add_command(label='Ayuda',command=AyudaMenu)
1217.     #---Luego, la sección Información
1218.     InfoMenuHV=Menu(barraMenuHV,tearoff=0)
1219.     barraMenuHV.add_cascade(label='Información',menu=InfoMenuHV)
1220.     InfoMenuHV.add_command(label='Información de
    procesamiento',command=InfoProcesamiento_HVSR)
1221.     InfoMenuHV.add_command(label='Acerca de...',command=AcercaDe)
1222.
1223.     #----Tamaño del frame donde se trabajará el analsis HVSR
1224.     FrameHV=Frame(ventana_nueva_HV,width=500, height=500)
1225.     FrameHV.pack()
1226.     #----Se crean las viñetas para la estacion a analizar
1227.     EstacionEntryHV=Entry(FrameHV)
1228.     EstacionEntryHV.grid(row=0, column=1)
1229.     EstacionLabelHV=Label(FrameHV, text="Estacion a analizar")
1230.     EstacionLabelHV.grid(row=0, column=0,sticky='w',pady=5)
1231.     #----Se crean las viñetas para la fecha a analizar
1232.     FechaEntryHV=Entry(FrameHV)
1233.     FechaEntryHV.grid(row=1, column=1)
1234.     FechaLabelHV=Label(FrameHV, text="Fecha de Analisis (DD/MM/AAAA)")
1235.     FechaLabelHV.grid(row=1, column=0,sticky='w',pady=5)
1236.
1237.
1238.     #----Se crean las viñetas para la hora a analizar
1239.     HoraEntryHV=Entry(FrameHV)
1240.     HoraEntryHV.grid(row=2, column=1)
1241.     HoraLabelHV=Label(FrameHV, text="Hora de inicio analisis (HH:MM:SS)")
1242.     HoraLabelHV.grid(row=2, column=0,sticky='w',pady=5)
1243.
1244.
1245.     #----Se crean las viñetas para el tiempo que dure el analisis
1246.     TiempoEntryHV=Entry(FrameHV)
1247.     TiempoEntryHV.grid(row=3, column=1)
1248.     TiempoLabelHV=Label(FrameHV, text="Duracion del analisis (seg)")
1249.     TiempoLabelHV.grid(row=3, column=0, sticky='w', pady=5)
1250.
1251.     #-----Se crean las ventanas para preguntar si se desea combinar o realizar un analisis
    independiente de resultados en las horizontales
1252.
1253.     AnalisisLabelHV=Label(FrameHV, text="Componentes horizontales")
1254.     AnalisisLabelHV.grid(row=4, column=0,sticky='w',pady=5)
1255.     selectedHV = IntVar()
1256.
1257.     #----Se crean botones tipo "círculo"
```

```
1258.     rad1 = Radiobutton(FrameHV,text='Combinadas', value=2, variable=selectedHV)
1259.     rad2 = Radiobutton(FrameHV,text='Independientes', value=1, variable=selectedHV)
1260.
1261.     rad1.grid(column=1, row=4)
1262.     rad2.grid(column=2, row=4)
1263.
1264.     #-----Se crea el boton para correr el programa H/V
1265.     BotonesFrameHV=Frame(ventana_nueva_HV,width=400, height=70)
1266.     BotonesFrameHV.pack()
1267.     BotonRunHV=Button(BotonesFrameHV,text='Correr',command=H_on_V_aux) #si agrego en el
    codigo esto: command=codigoBoton, llamo a la funcion codigo boton
1268.     BotonRunHV.pack()
1269.
1270. #----Se crean los botones para seleccionar analisis SSR o H/V en la VENTANA RAIZ.
1271.
1272. BotonSSR=Button(FrameBotones,text='Standard spectral
    ratio',command=ventana_SSR).grid(row=1,column=0)
1273. BotonHV=Button(FrameBotones,text='Horizontal to vertical spectral
    ratio',command=ventana_HV).grid(row=1,column=1)
1274.
1275.
1276.
1277.
1278. raiz.mainloop()
```