Modelling Bonus Seismic Wave Arrivals Recorded on SchoolShake Seismographs from the March 2025 Orcas Island Earthquake

By

Sedona Reed

A Thesis Submitted in Partial Fulfillment
of the Requirements of the

HONOURS PROGRAM

in the School of Earth and Ocean Sciences

Supervisors: Camille Brillon, Lucinda Leonard

University of Victoria

We acknowledge and respect the Lək̓ʷəŋən (Songhees and Xʷsepsəm/Esquimalt) Peoples on whose territory the university stands, and the Lək̓ʷəŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I want to deeply thank my supervisors: Camille Brillon and Lucinda Leonard, for all of the time and support they dedicated throughout this project. Their guidance, feedback, and sharing of their expertise were integral to this research. Through their supervision I learned so much about how to conduct and communicate meaningful research, and every single component would not have been possible without their support.

I express my gratitude to those whose work that built the foundation for this project. Specifically, to Camille Brillon, Ed Nissen, and Andrew Schaeffer for the implementation of the SchoolShake network across Victoria, of which the recorded data led to the possibility of this project.

And finally, to those in my personal life who supported me through this project and above all came to all of the poster fairs.

Abstract

In the seismograms recorded at SchoolShake network seismographs across Greater Victoria, BC during the M4.5 Orcas Island earthquake on March 3rd, 2025, a bonus wave arrival was identified between the P and S arrivals at stations within an azimuthal range of 219-256° around the epicenter. Such arrivals had not been previously documented. The focus of this study was to determine the origin of the bonus seismic wave arrivals using seismic ray path and travel time modelling for increasing levels of model complexity. Initial modelling using the 1-D ObsPy.TauPy software successfully reproduced direct P and S wave arrivals but was insufficient to model the bonus phase arrivals. Subsequent 2-D modelling using Pykonal was implemented to better account for the subsurface structure, which includes dipping boundaries such as the top of the subducting Juan de Fuca plate. This modelling demonstrated that a reflection of seismic energy off the Juan de Fuca slab produced arrivals too late to match the bonus arrivals, therefore ruling out this boundary as their origin. Modelling reflections off a shallower boundary, consistent with the highly seismically reflective E-layer, produced arrival times consistent with the actual bonus phase arrivals. Residuals between the observed and modelled arrival times were minimized to lie within ±0.8 seconds across the analyzed stations, providing evidence that the bonus arrivals are consistent with P wave energy reflection off the top of the E-layer. These findings demonstrate the need for more spatially complex subsurface models, such as the 2-D Pykonal model and ideally a 3-D model, to model seismic wave travel in the presence of dipping subsurface boundaries. The bonus phase arrivals may contribute to longer and more intense shaking, which highlights the importance of understanding seismic wave travel to improve local hazard assessment and risk mitigation in Greater Victoria.

1 Introduction

Of the approximately 4000 earthquakes that occur in Canada each year that Natural Resources Canada (NRCan) locates, the largest and most frequent occur along the plate boundaries off the west coast (Cassidy et al., 2010). As a result, the seismic activity in southwestern British Columbia (BC) is a topic of significant ongoing research to better understand and monitor the hazards in the region. As plate boundaries and other faults in this region are capable of hosting large (M7+) and great megathrust earthquakes (M8+), it is important to understand the mechanisms that impact how earthquakes affect local communities.

In order to monitor the seismicity in the region, there are many different seismograph networks that record incoming seismic wave energy (for example CN – Canadian National Seismograph Network; PQ – Geological Survey of Canada Research Network; UW – University of Washington's Pacific Northwest Seismic Network; AM – Raspberry Shake). In this study, I will focus mainly on the impacts of introducing the SchoolShake network of Raspberry Shake seismographs across Victoria, which has increased the density of stations, allowing for interesting new observations to be made (RaspberryShake, 2024).

On March 3rd, 2025, a M_w 4.5 earthquake occurred at ~15 km depth beneath Orcas Island, Washington (east of Sidney, BC). This event was widely felt across the Puget Sound region in Washington, up to north Vancouver, as well as in southern Vancouver Island (USGS, 2025). It was reported after this event that many people in Victoria felt a longer duration of shaking than in previous earthquakes of similar size and distance (NRCan, 2026). While this event caused no significant damage, it was the strongest felt event on Vancouver Island since the introduction of the SchoolShake Seismograph network in the region, implemented in 2021.

During the March 3rd earthquake, seismograms from the SchoolShake stations recorded an unexpected seismic phase arriving between the P (Primary, Compressional) wave, and the S (Secondary, Shear) wave within an azimuthal range of 219-256°. Such arrivals had not been widely observed in previous earthquakes as the station coverage was too sparse prior to the implementation of the SchoolShake network. In this study, I test the hypothesis that these bonus arrivals are due to the reflection of P wave energy off the subducting Juan de Fuca plate. The Juan de Fuca plate is subducting northeastwardly below the region at ~30-50 km depth; reflection off the subducting slab could potentially explain why the bonus arrivals are only seen across a specific azimuth of stations and not at every station surrounding the event. The bonus wave arrival also explains the higher intensity of shaking as reported by the surrounding communities. A number of NRCan DYFI reports mentioned shaking or feeling two distinct pulses of energy (Camille Brillon, personal communication, February 19, 2025).

In order to test this hypothesis, this study modelled travel times in one and two dimensions for various seismic phases of the March 3rd earthquake, for comparison with the actual seismogram observations. A variety of stages of complexity and dimensionality of the system were tested, using two open source seismological data computation tools (TauPy: Beyreuther et al., 2009, and Pykonal: White et al., 2020). At each stage the modelled arrival times are compared to the actual recorded arrival times, on the SchoolShake stations, ultimately to determine the source of the bonus arrivals. A secondary objective of this study is to investigate the accuracy of the different modelling routines and velocity models used to calculate the arrival times of the event across an array of seismograph stations. This comparison can then be used to gain insight on which modelling program is more suitable for the region, and to motivate implementation in future local seismic analysis and research. A

better understanding of the March 3rd earthquake better informs the local seismic risk of an event such as this, as well as where we might expect to see similar arrivals and shaking during future earthquakes.

2 Background

2.1 Seismicity and Tectonics in the Study region

The Cascadia subduction zone (CSZ) is located off the west coast of British Columbia, spanning from northern Vancouver Island to northern California, (figure 1). It consists of the relatively young, small Juan de Fuca (JdF) plate subducting northeastward under the North American plate at a rate of ~3.5-4 cm/yr (e.g., Gao & Long, 2022). The JdF plate is bordered to the north and the south by two microplates, the Explorer and Gorda plates, respectively. The JdF plate is a few hundred kilometers wide west-east from spreading centre to trench and is relatively young (< 10 Ma) and is thus still quite warm and buoyant (Gao & Long, 2022). The CSZ is globally unique among other subduction zones, in its high temperature as well as being the quietest of all global subduction zones with low levels of seismicity (Gao & Long, 2022).

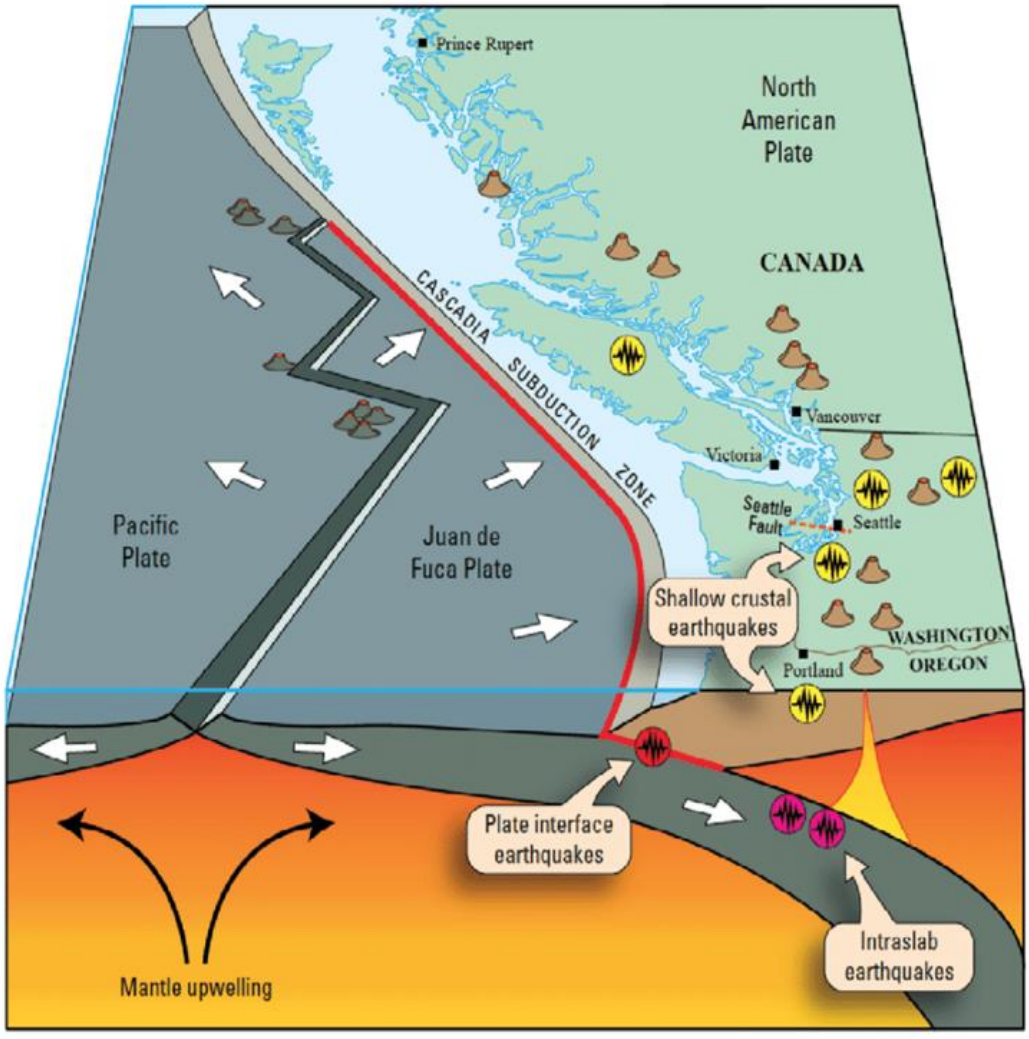


Figure 1 *– Three dimensional diagram of the northern portion of the Cascadia subduction zone. Different types of earthquakes produced in this region labelled: shallow crustal earthquakes, deep intra-slab earthquakes, and plate interface earthquakes. Here, the Juan de Fuca plate subducts northeastward beneath the North American plate. Sourced from McGuire et al., (2021). Reproduced without modification.*

The continent scale convergence between the JdF and North American plates is accommodated by an ~1000 km long megathrust fault, which is capable of producing great (M8) earthquakes. The paleoseismic record of the CSZ includes dozens of past megathrust earthquakes, with M8+ events occurring at a roughly 250 year interval (Stone et al., 2018). The last of these events in January 1700, wherein a believed M9 earthquake ruptured the full length of the subduction zone (Stone et al., 2018). This event caused large local tsunami waves and caused a large drop (1-2 m below sea level) in 600-1000 km of the coast of Washington and Oregon state. This region also hosts regular seismicity in the crust of the North American plate and earthquakes within the JdF plate in non-uniform distributions, but with higher concentrations off western Vancouver Island, in the Puget Sound region in Washington, and offshore northern California (Bostock et al., 2019). Specifically, as noted by Bostock et al. (2019), is that there are high concentrations of shallow (<40 km) depth crustal seismicity along the CSZ forearc between latitudes of~ 47-49°N, possibly due to high positive slab curvature at this mid-range the subduction zone, whereas in northern Vancouver Island, and in Oregon, the slab is dipping more gradually.

There are other significant boundaries (additional to the JdF slab) within the velocity structure of this region. An example of another layer that has been observed is the E-layer (or E conductor). This layer, first seen in LITHOPROBE studies (Green et al., 1986), is interpreted to be a conductive layer of low density and low P and S wave velocity (Nicholson et al., 2005). There have been several proposed explanations of the origin of the E-layer and its seismically reflective nature. These include that the E-layer is (1) part of the top of the subducting Juan de Fuca slab (a now largely discounted theory (Nicholson et al., 2005)), (2) sheared and imbricated sediments trapping fluid from the dehydrating slab (Calvert & Clowes, 1990), and (3) a layer that coincides with the top of the oceanic plate (Nedimović et al., 2003). The E-layer has been imaged in seismic reflection profiles to be 3-10 km thick and to lie above and parallel to the subducting JdF slab, reaching at least 45 km depth east of Vancouver Island (Ramachandran, 2001). Figure 2 shows the location and approximate thickness of the E-Layer, identified through seismic reflection methods (Bostock et al., 2025). This reflection profile was taken along a W-E profile across the Strait of Juan de Fuca just south of Vancouver Island (~48.3°N), where the deep red high velocity structure represents the subducting JdF slab, and the E-layer is seen to be lying above and parallel.

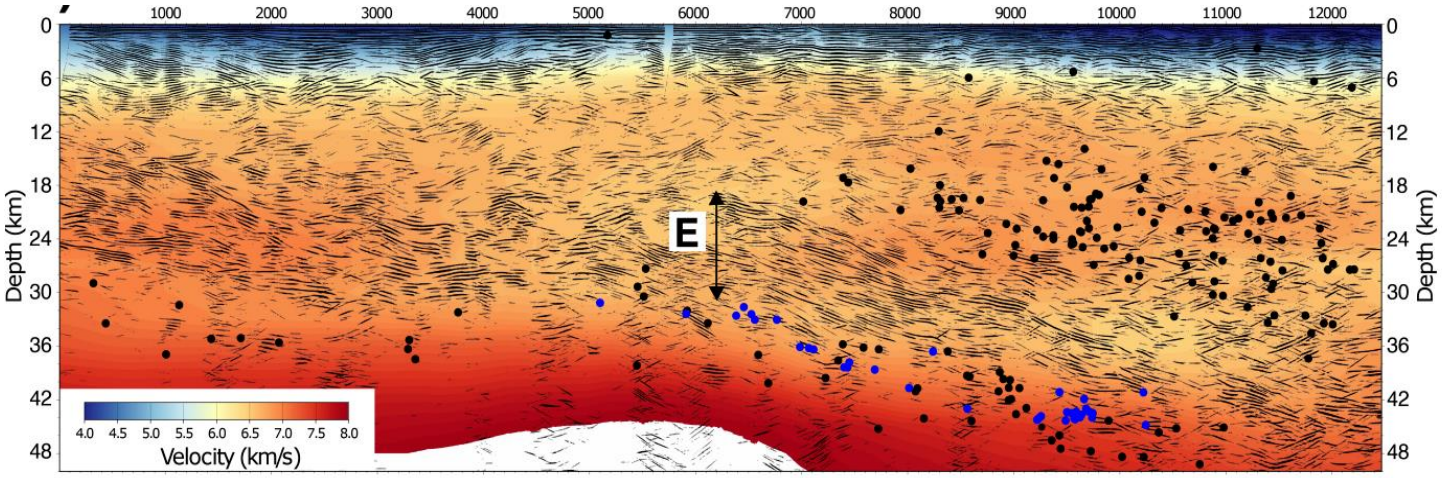


Figure 2 *- Seismic reflection profile (using SHIPS method). Near W-E cross section across Strait of Juan de Fuca (~48.3°N). This cross section presents the depth and thickness of the E-layer in this location, significantly thicker here than below Vancouver Island (~10 vs 5 km). Deep red high velocity structure represents subducting JdF slab, with highly reflective E-layer lying above and parallel. Figure sourced from Bostock et al., 2025, 'Tectonic tremor: The chatter of mafic underplating'. Seismica, Vol 2.4. Figure reproduced without modification.*

Due to the frequency of crustal and intra-slab earthquakes in the region, as well as the less frequent but larger megathrust earthquakes, Cascadia is extensively studied in order to assess the hazards and risk posed to the large population located in western Canada and the United States. To aid in the goal of studying the dynamics and seismicity in the region, there are many seismic networks in the area which are responsible for providing the majority of the seismic observations in the CSZ, such as the Pacific Northwest Seismic Network and the Canadian National Seismograph Network (Stone et al., 2018). The introduction of denser seismograph networks in the region allows for previously unseen observations to be made in the seismic recordings in the area. Specifically, the focus of this study is on the introduction of the SchoolShake Network of Raspberry Shake Seismographs in Victoria, B.C in 2021 (RaspberryShake, 2024).

2.2 Seismic Wave Theory and Earthquake Terminology

An earthquake is characterized as a release of elastic energy due to the sudden slip of two pieces of the earth's crust past each other on a fault. The stored energy emitted from an earthquake is transmitted through the earth as several types of seismic waves. Each type of wave emitted from the earthquake focus has a distinct particle motion, elastic deformation, and speed. Therefore, in recorded seismograms, it is possible to determine the arrival time of each of the wave types, as well as waves that have reflected and refracted off different earth boundaries. The speed of propagation of each wave depends on the elastic properties and density of the medium it travels through (Braile, 2010). Figure 3 illustrates the energy released from an earthquake as propagating waves in all directions from the source.

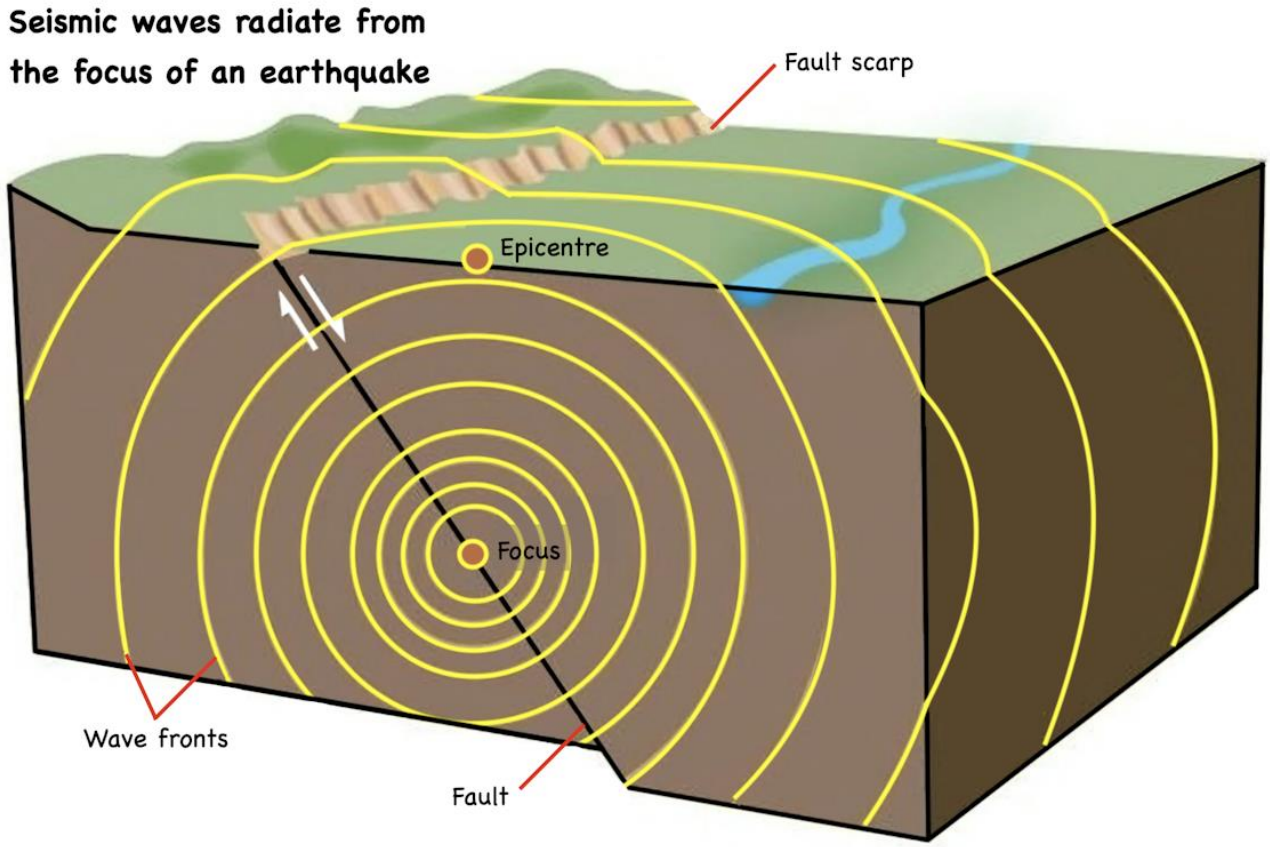


***Figure 3** – Seismic waves being emitted in all directions from the focus of an earthquake. Figure sourced from Science Learning Hub – Pokapū Akoranga Pūtaiao, The University of Waikato Te Whare Wānanga o Waikato, www.sciencelearn.org.nz, accessed 2026. Figure reproduced without modification.*

There are 2 main types of body waves, which are waves that propagate through the interior of the earth at speeds that depend on the elastic properties of the medium. The first type are *P waves,* which are compressional waves that cause particle motion in the same direction as the wave propagation, such that the disturbance propagates as a series of dilations and contractions. These waves are the first arrival to seismic stations after an earthquake as they have the fastest velocity. The velocity of P waves is given by:

$$V_p = \sqrt{\frac{K+\frac{4}{3}\mu}{\rho}} \tag{1}$$

where K is the bulk modulus, μ is the shear modulus, and ρ is the density of the material. The bulk modulus K defines the material's resistance to compression, and is given by

$$K = -V_o \frac{\Delta P}{\Delta V} \tag{2}$$

where V_o is the original volume of the material, ΔP is the change in pressure on the material, and ΔV is the resulting change in volume (Geophysics for Practicing Geoscientists, 2017). Materials with a high bulk modulus are more resistant to compression, characterized in this formula as having a small change in volume for a large change in pressure. A schematic diagram of a cell of material for a given K can be seen in figure 4.

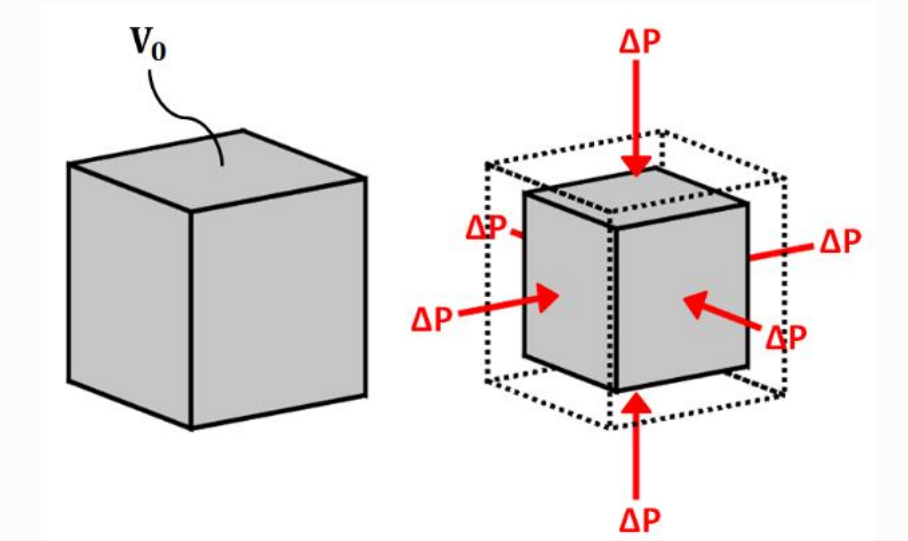


Figure 4 *- Bulk modulus diagram. Here, Vo represents initial volume of the cell, and the second panel shows the resulting change in volume (ΔV) of the cell due to an applied change in pressure (ΔP). Figure sourced from 'Seismic Velocity' (Geophysics for Practicing Geoscientists, 2017), https://gpg.geosci.xyz/c , accessed February 2026. Figure reproduced without modification.*

The shear modulus μ defines the material's resistance to shear stress, and is given by

$$\mu = \frac{stress}{strain} = \frac{Force}{Area} * \frac{l}{\Delta x} \tag{3}$$

where l is the perpendicular dimension of the cell of material, and Δx is the resulting shear displacement of the cell due to the tangent angle shear strain (Geophysics for Practicing Geoscientists, 2017). A schematic of a shear deformation of a cell of material with the corresponding labels of components of μ can be seen in figure 5.

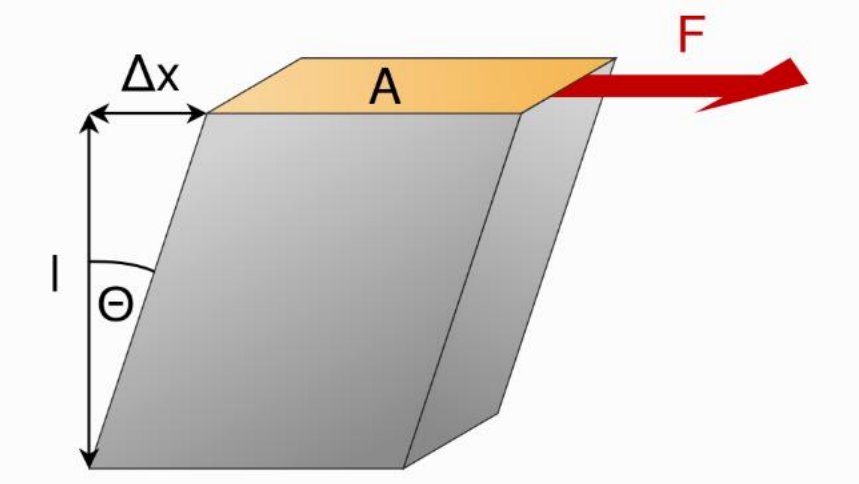


Figure 5 - *Shear Modulus diagram. Cell of material subject to an applied shear stress (Force F per area A) producing shear strain of the cell. Shear strain represents a tangent angle (θ) between the shear deformation (Δx) of the material along the direction of force and the perpendicular dimension l of the cell. Figure sourced from 'Seismic Velocity' (Geophysics for Practicing Geoscientists, 2017), https://gpg.geosci.xyz/c , accessed February 2026. Figure reproduced without modification.*

S waves, the second main type of body waves, are shear waves (also known as transverse waves) as they correspond to elastic deformation which shears the particles, causing oscillations perpendicular to the direction of propagation. The velocity of S waves is given by

$$V_s = \sqrt{\frac{\mu}{\rho}} \tag{4}$$

where μ is the shear modulus, and ρ is the density of the material (Geophysics for Practicing Geoscientists, 2017). As these waves depend solely on shear deformation, relating to the shear modulus μ, they are unable to propagate through fluids as the rigidity of such a material is zero.

A known consequence of comparing the P and S wave velocities, in equations 1 and 4, is that the P wave velocity is always greater than the S wave velocity for waves travelling in the same medium. This leads to a second naming convention for P and S waves: P waves are the *Primary* waves, arriving to the seismograph first, followed by the *Secondary* S waves, typically the second body wave arrival. This does not account for other phases that arrive between and after the P and S waves, as well as the seismic waves that are restricted to a free surface, *surface waves*, which arrive later than body waves. The factors that most impact seismic velocity are the mineralogy and structure of the material (bulk and shear modulus), as well as alterations to the material such as porosity, hydration, and lithification (Geophysics for Practicing Geoscientists, 2017). Figure 6 explicitly shows the particle motion inherent to each seismic wave type in relation to the direction of wave propagation.

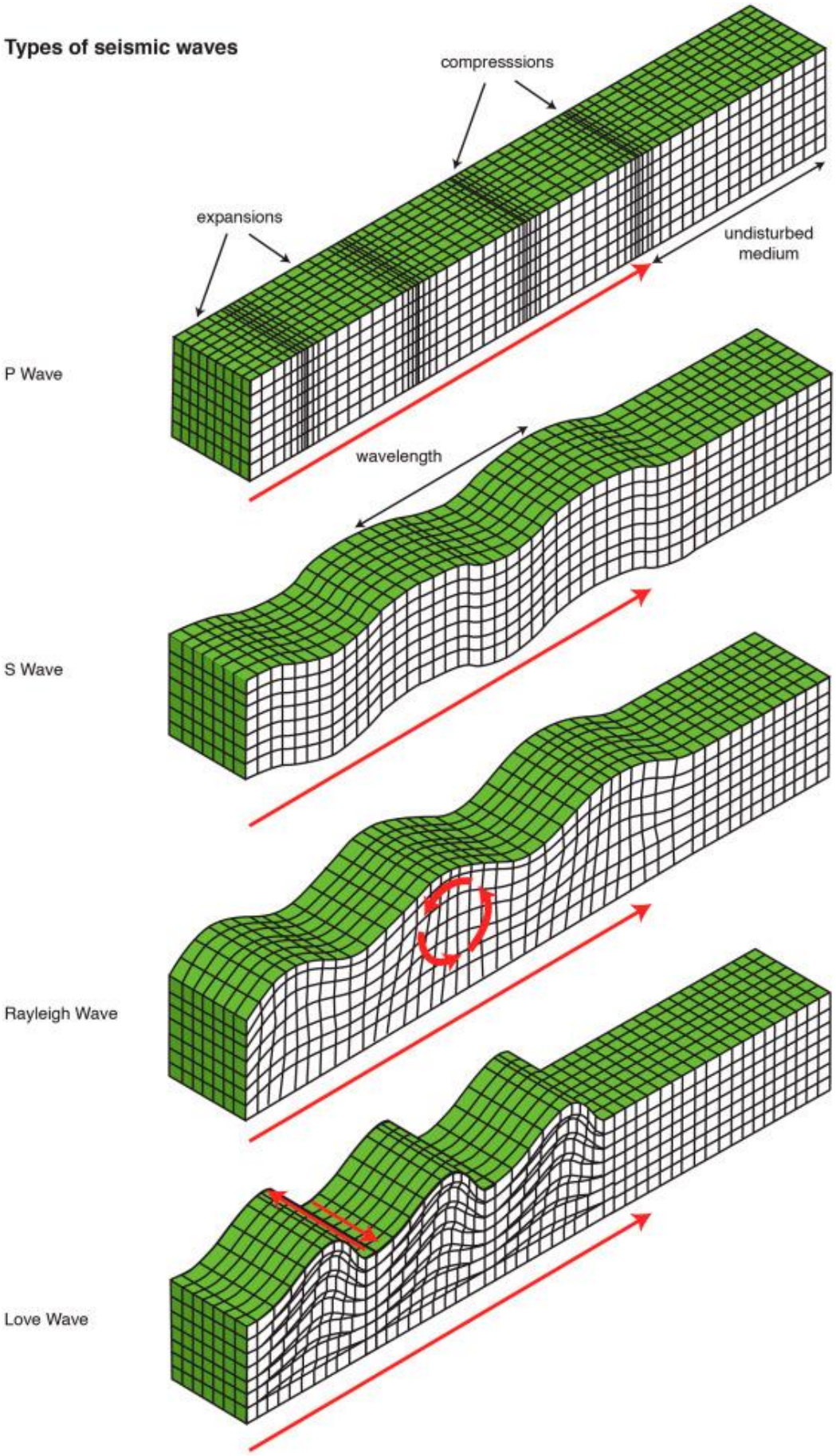


Figure 6 *- The main types of elastic seismic waves (Body and Surface waves), and their respective particle motion relative to direction of propagation. From top to bottom: Body waves: P, S, Surface waves: Rayleigh and Love waves, respectively (Shaw et al., 2018). Figure reproduced without modification.*

Modern seismometers have three separate recording directions to record horizontally and vertically arriving waves. These instruments can measure the N-S, E-W, and vertical components of each wave. Seismographs record ground motions due to seismic waves as electrical signals, which are then converted by a digitizer into a time series that can be digitally displayed as a seismogram (figure 7).

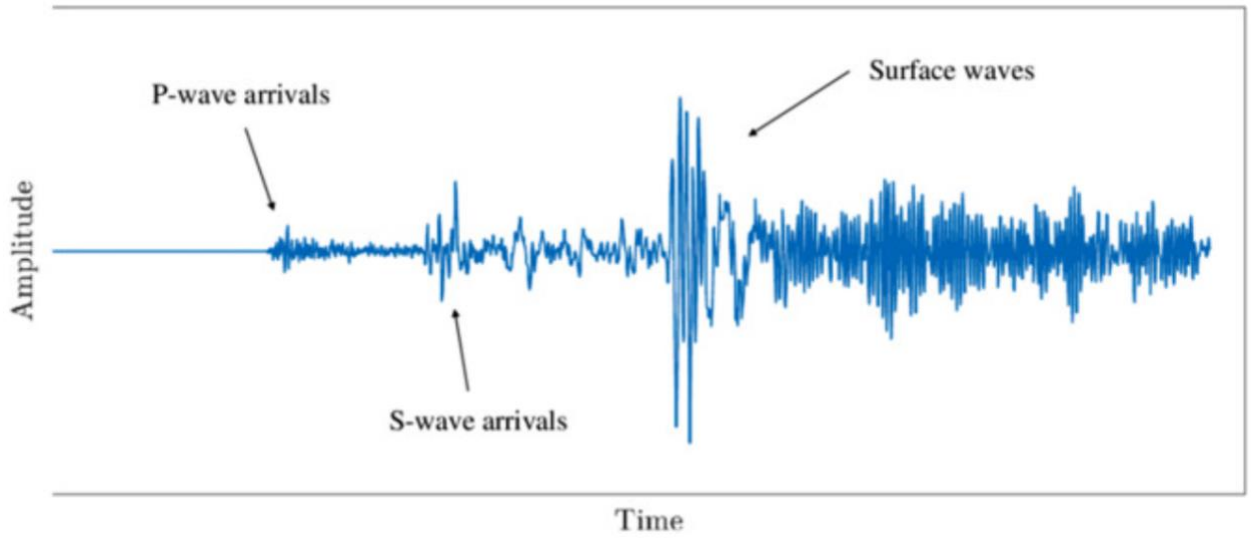


Figure 7 *– Example of a simple seismic waveform showing the P, S and surface wave arrivals (Hohensinn, 2019). Figure reproduced without modification.*

In typical earthquake arrivals seen in seismograms, the P wave is the first arrival of energy, followed by the S wave arrival, and then there are a variety of surface wave types which arrive later. When seismic waves encounter boundaries within the earth's interior, they respond by reflecting, refracting, and/or transmitting. For earthquakes that travel great distances, the waves encounter many boundaries between the hypocenter and the seismograph, giving them distinct and complex phase arrivals. These arrivals follow a naming convention, wherein the arrival is characterized by which boundaries it interacted with (figure 8). As the earth is known to be split into differing rheological layers, these body waves travel at different speeds within each layer and reflect and refract at different angles across each layer boundary. For small magnitude earthquakes, ray paths are more constrained to shallow depths and interactions with shallower boundaries (rather than the core, for example) and are not recorded at seismographs at great distances. The travel time of a given seismic phase can be predicted based on a travel timetable computed from the velocity model of the earth being used (University of Alberta, 2013).

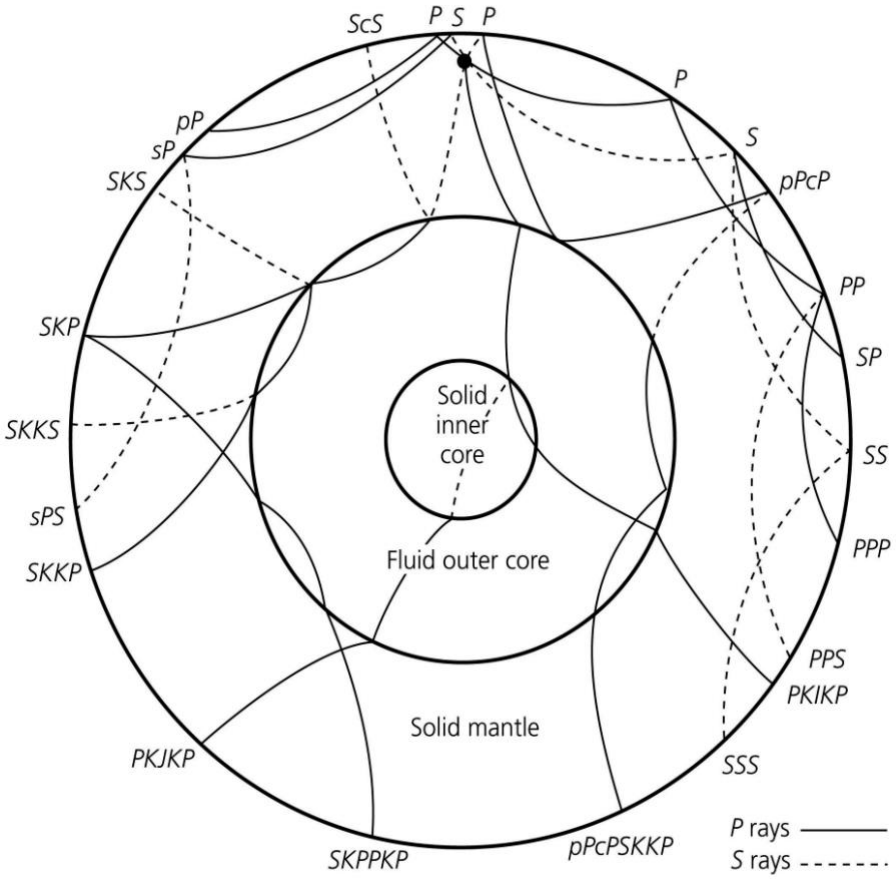


Figure 8 *– Diagram of ray paths and associated phase name for various examples of seismic wave arrivals. All P waves (solid lines) and S waves (dashed lines) emanate from the earthquake focus (black dot) and receive phase arrival names following the convention which identifies the path taken and boundaries interacted with. Figure sourced from UofA, 2013, reproduced without modification.*

To quantify the energy released by an earthquake, there are different magnitude scales used. The most accurate is the moment magnitude (M_w) scale, which is based on the physical properties of the earthquake and does not saturate for larger earthquakes as other scales do. The M_w value is a measure of the amount of energy released in the event of an earthquake and is related to the area of the fault rupture and the amount of slip on the rupture, as well as the strength of the material faulting (British Geological Survey, 2026). These values are derived from an analysis of the recorded seismograms from an earthquake (Earthquake Hazards Program, 2026). The formula for calculating the seismic moment (the quantification of the energy released) is

$$M_o = \mu A L \tag{5}$$

where μ is the shear modulus of the material, A is the area of the fault rupture, and L is the average slip distance. This value M_o is then used to calculate the moment magnitude M_w using

$$M_w = \frac{2}{3}\log(M_o) - 6.06 \tag{6}$$

where the constants are empirically determined (e.g., British Geological Survey, 2026).

Another type of measure of earthquake strength is the *Intensity.* Intensity is a subjective and site-specific measurement of the strength of shaking that occurred due to the event, and most commonly decreases as a function of distance from the epicenter. The reported intensity of shaking depends on how far away the

earthquake source is to the reporter, the depth of the earthquake hypocentre, the material making up the bulk of the location's geology, the type of wave arrivals at the location, and the direction of earthquake rupture (Earthquake Hazards Program, 2026). Intensity values are determined through publicly answered felt reports and can be estimated from instrumental data. CDI (Community Decimal Intensity) intensity maps are produced for an earthquake using public reported intensity (e.g., figure 9). As was mentioned, we see that the intensity from the event decreases with distance from the source. The scale for the intensity map is defined by increasing roman numerals, in line with the Mercalli Intensity scale with I being no caused damage and not felt, and increasing up to X+ for extreme perceived shaking and very heavy damage.

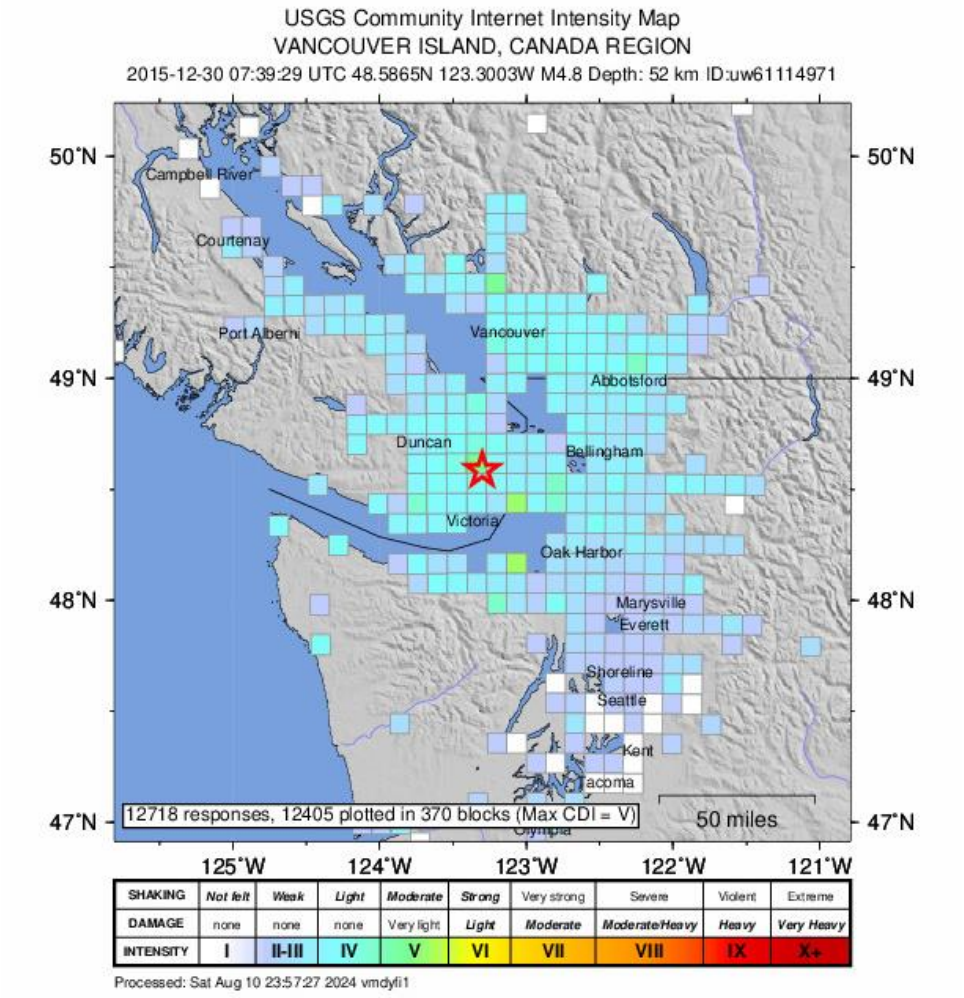


Figure 9 *– Example of a typical CDI map. This intensity map has been made using 12718 public responses on perceived intensity. Figure sourced from USGS, 2015, https://earthquake.usgs.gov/earthquakes/eventpage/uw61114971/executive , accessed March 2026. Reproduced without modification.*

2.3 The SchoolShake Network and Raspberry Shake Seismographs

The SchoolShake program began development in 2021 by the Geological Survey of Canada (GSC) and the University of Victoria (UVic) with the goal of bringing hands-on geoscience education to classrooms around Vancouver Island to engage youth in STEM (Science, Technology, Engineering, and Math) and promote earthquake preparedness in schools in the area, while also creating a new high density network of seismographs to increase understanding of local seismic hazards (Brillon et al., 2024). To accomplish these goals, many small, low-cost, user-friendly Raspberry Shake (RS) seismographs, were installed in schools around Vancouver Island.

Data from this network allow seismologists to better characterize poorly understood crustal faults in the region, such as the source of the highlighted event of this research, the March 3rd, 2025, Orcas Island event.

The Raspberry Shake network is a public, freely accessible network that provides high quality seismological data to researchers and scientists, in educational settings, and at home (RaspberryShake, 2024). Currently, there are over 2200 RS seismographs online globally. The wide response spectrum (0.5 - ~40 Hz geophones) means these seismographs can perform well for smaller local and regional seismicity, as well as global teleseismic larger magnitude events if installed in an ideal location. The SchoolShake seismographs are placed in locations that are as quiet as possible, but they are still in schools, meaning their background noise levels are higher than a typical seismograph station. A map of the Raspberry Shake seismographs installed over the west coast of Canada (figure 10A) shows a cluster of SchoolShake seismographs installed on the Saanich peninsula. The SchoolShake network has significantly increased the spatial density of seismographs on southern Vancouver Island (figure 10B).

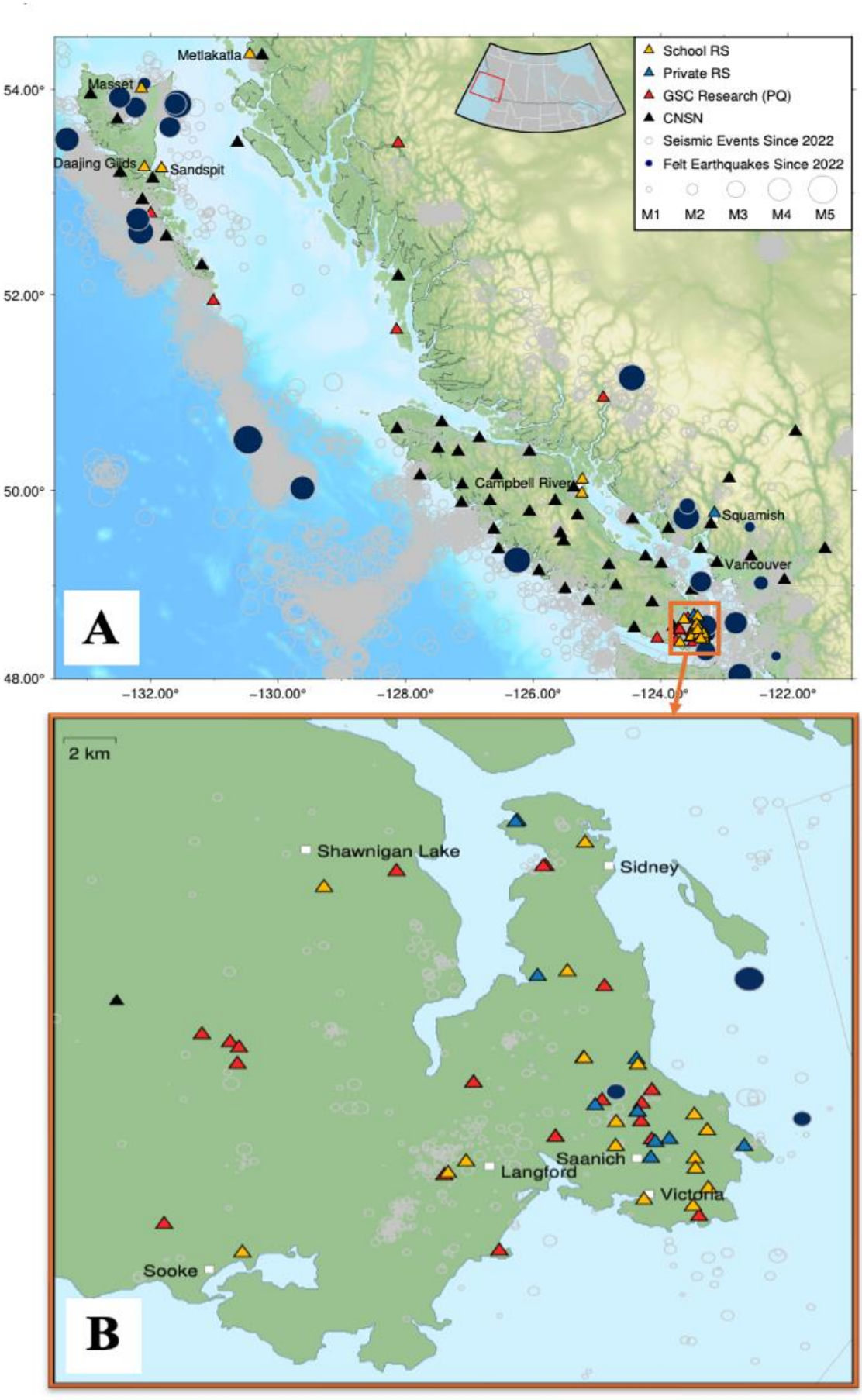


Figure 10 –

Panel A: *Map of Raspberry Shake seismographs along Canada's west coast. SchoolShake RS stations (yellow triangles), privately owned RS (blue triangles), GSC research stations (red triangles) and, CNSN stations (black triangles) plotted. Map also displays seismic events from 2022 until 2025 (grey circles), those felt in black.*

Panel B: *Zoomed in plot of seismic stations on southern Vancouver Island. Legend is the same as for panel A. Increased density of stations in this region with the introduction of the SchoolShake network (yellow triangles) can be clearly seen.*

Figures sourced from SchoolShake, 2025.

2.4 The March 3rd, 2025, Orcas Island Earthquake

The event of interest in this project is the M_w 4.5 crustal earthquake that occurred on March 3rd, 2025, at 13:02:37 UTC (5:02am PST). This event was located at 15.8 km depth, with an epicenter on Orcas Island, Washington US, just east of Sidney BC. The shaking due to this event was widely felt across the Puget Sound region, from north Seattle to north Vancouver, as well as across southern Vancouver Island (Stevens et al., 2025). It was observed that the shaking of this event lasted longer than previous events of the same depth and magnitude (NRCan DYFI, 2025). In the 24 hours after this event, Pacific Northwest Seismic Network (PNSN) located 47 tiny aftershocks within 10 km (figure 11), the largest of those being M 0.92 (Stevens et al., 2025). The focal mechanism of the main shock event (figure 11) implies east-west compression on a north-south trending fault (Stevens et al., 2025). The fault on which this earthquake occurred on is unknown. There are some known very old and 'inactive' faults exposed on Orcas Island aligned with this earthquake's mechanism; detailed analysis is needed to determine if this earthquake occurred on a deeper part of one of these known faults (Stevens et al., 2025).

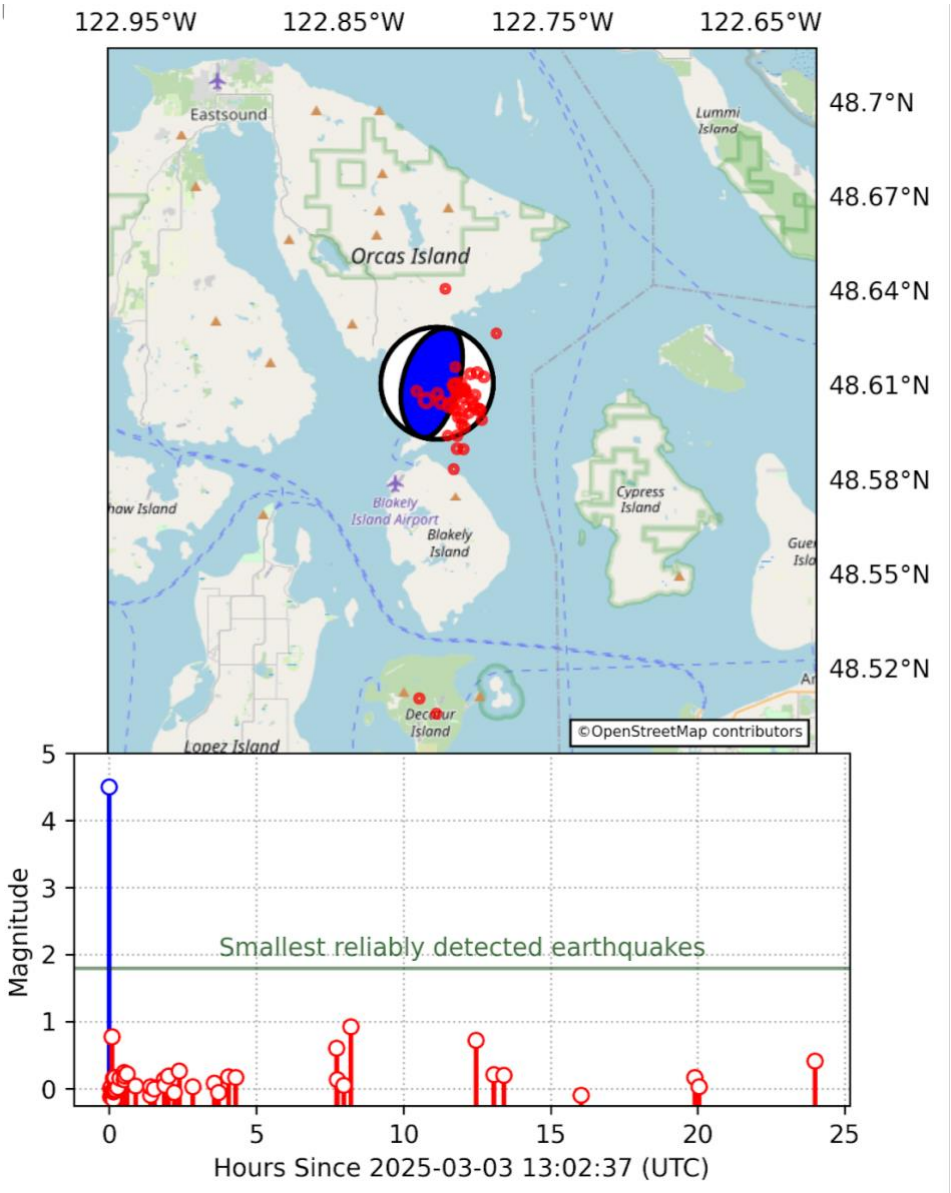


Figure 11 *– Focal mechanism and aftershocks of the M4.5 March 3rd, 2025, Orcas Island earthquake which occurred below Orcas Island at ~ 15 km depth. The 47 tiny aftershocks within 10 km of the March 3rd event within 24 hours after the main shock are visualized (red points). Below the map, the magnitude of these aftershocks plotted vs time since the main shock can be seen. Figure sourced from PNSN, Stevens et al., (2025). Figure reproduced without modification.*

The CDI map for this event (figure 12) shows light shaking was felt over part of southern BC as well as into northern Washington. The region of highest intensity extends from the epicenter to the west-southwest over the Greater Victoria Area, coincident with the azimuthal range that recorded the bonus phase arrival between the P and S waves (219-256°). Some seismograms in the region recorded an arrival of energy between the P and S wave (figure 13). The amplitude of this bonus arrival varies between stations, but for the majority with this bonus arrival it is of a larger amplitude than the P wave arrival. This extra arrival of energy would explain the comments left in the felt reports of this event, many of which remark on 2 distinct arrivals of shaking (Camille Brillon, personal communication, February 19, 2025). Therefore, the bonus arrival may be linked to longer and more intense shaking. Determining the specific phase and source of the bonus arrival is the goal of this project.

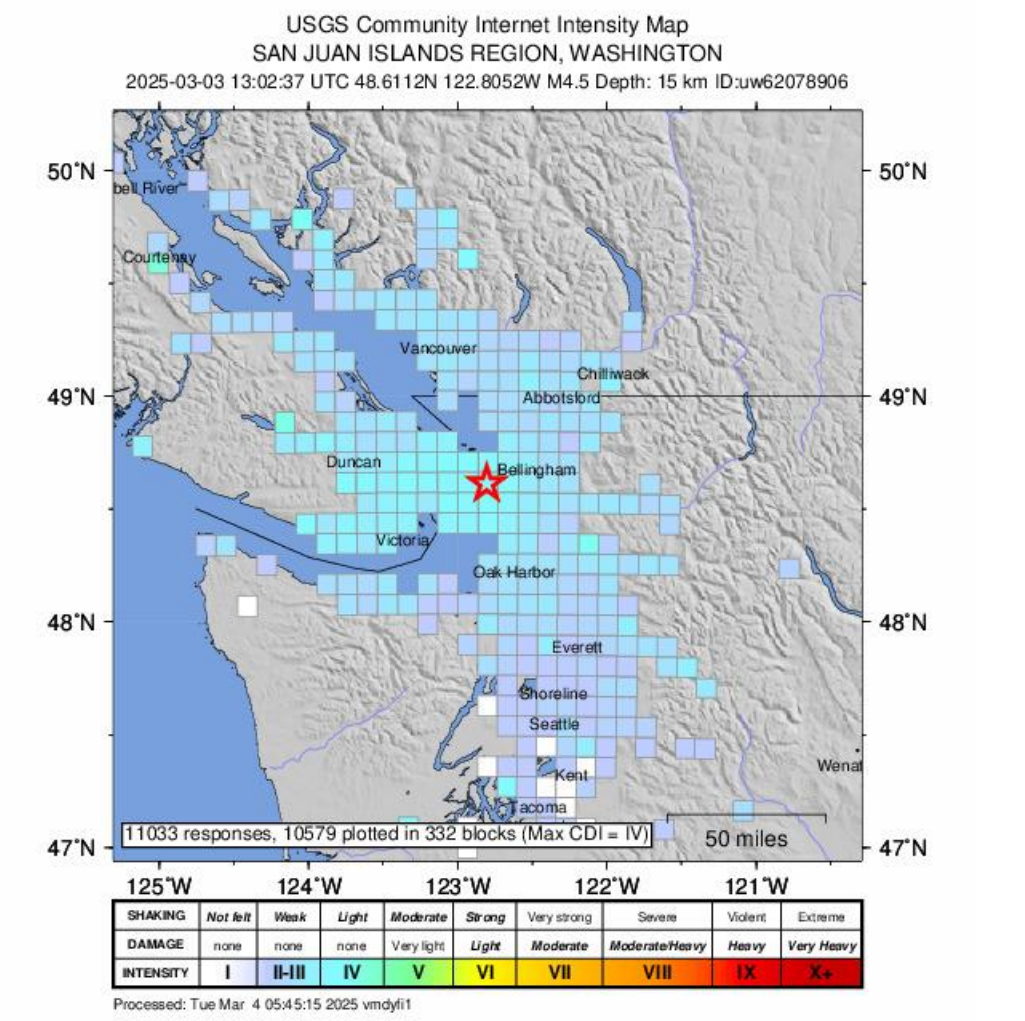


Figure 12 *– CDI map reported from the March 3rd, 2025, Orcas Island earthquake. Produced from over 11000 community responses from surrounding communities. CDI shows general decrease in perceived intensity with distance from the epicenter; however, region of highest perceived intensity is observed to coincide with azimuthal range which received the bonus arrival before the S wave arrival (219-256°). Sourced from USGS, (2025). Accessed March 2026, reproduced without modification.*

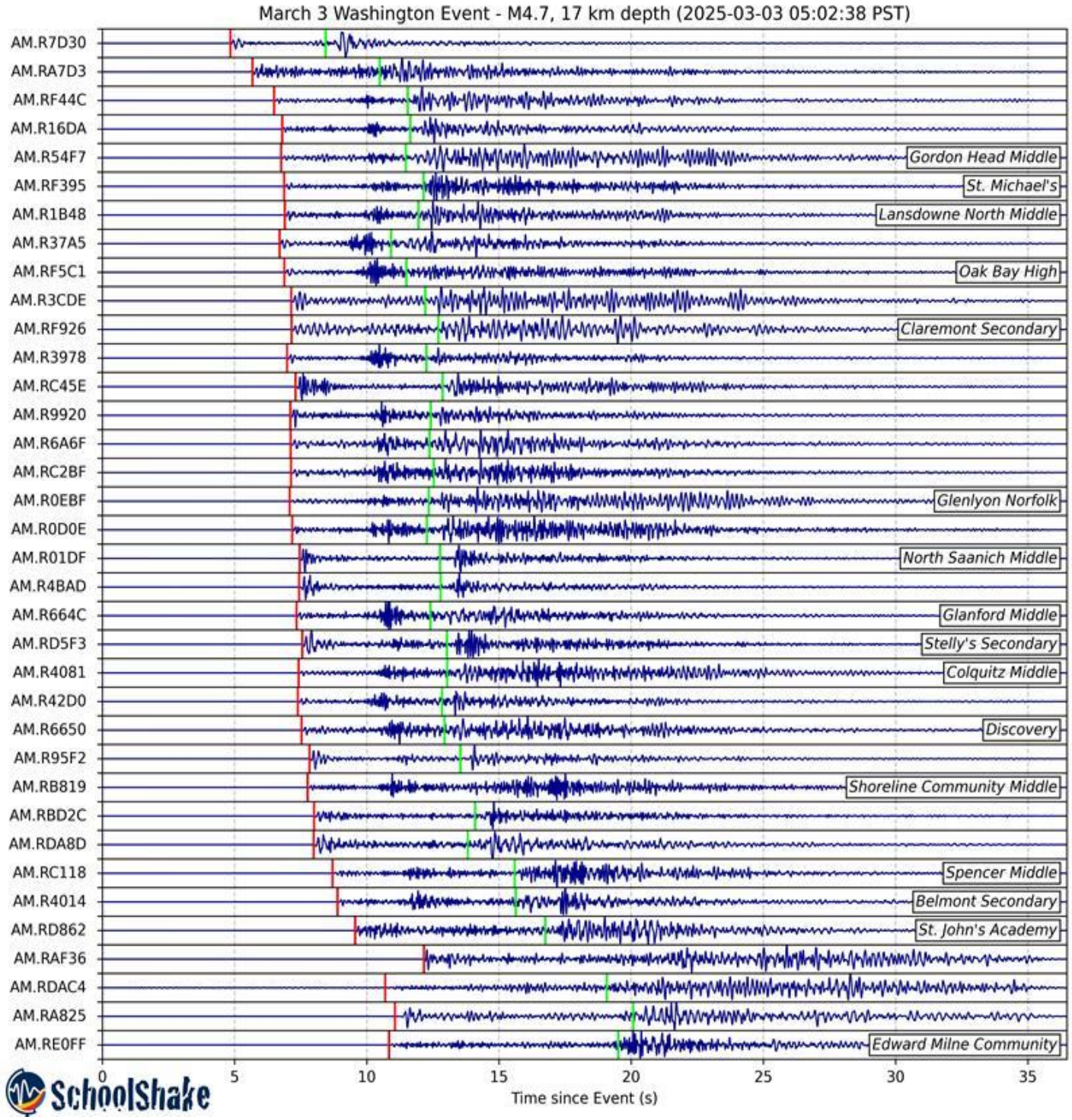


Figure 13 *– Seismograms recorded from the March 3rd, 2025, Orcas Island earthquake across all Raspberry Shakes in the SchoolShake network in the Greater Victoria area. The bonus wave arrival can be seen between the P wave arrivals (red line) and S wave arrivals (green line) for specific stations which lie in the azimuthal range of 219-256°. Stations which did not receive the bonus arrival also presented. Labels on the right identify the schools at which the station is installed. Geographical location of each station can be seen in table 1. Figure sourced from SchoolShake, (2025). Reproduced without modification.*

This study will test the hypothesis that the bonus arrivals from the March 3rd, 2025, Orcas Island earthquake, as measured by SchoolShake seismometers over a specific azimuthal range, resulted from the reflection of P wave energy off the subducting Juan de Fuca slab. In this region, the JdF slab has been imaged to

be subducting below Vancouver Island and the San Juan Islands, where this earthquake occurred. The subducting oceanic crust provides a northeast dipping boundary for the seismic energy to reflect off and is the hypothesized origin of the bonus arrivals. The JdF slab is at ~40 km depth beneath Greater Victoria, increasing northeastward, and is ~52 km beneath Orcas Island (figure 14).

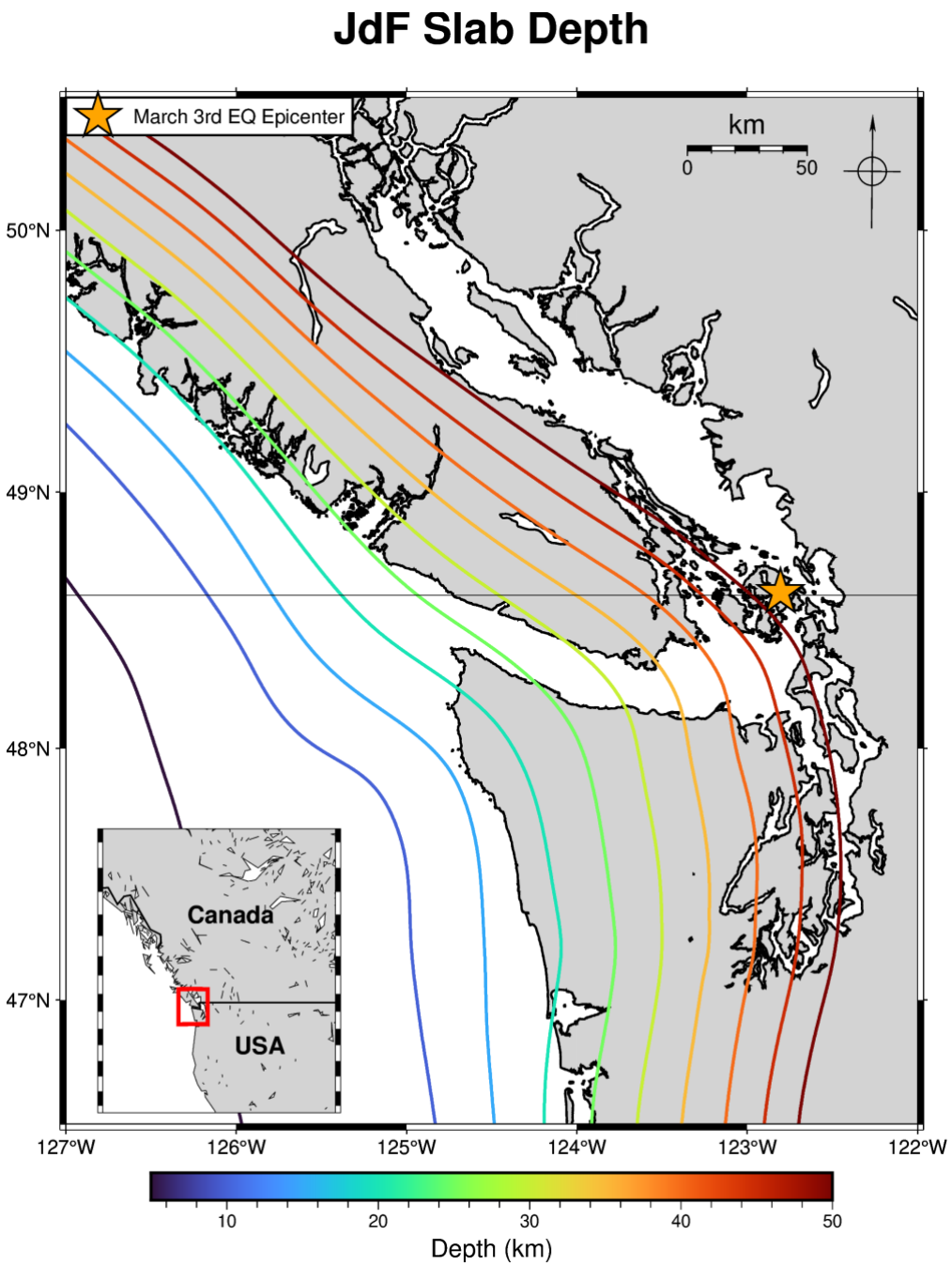


Figure 14 *- Depth contours of the Juan de Fuca plate subducting beneath the North American plate in the study region. Depth contours at 5 km intervals from 5 to 50 km below sea level. March 3rd, 2025, earthquake epicenter (yellow star) located on Orcas Island. Figure created using GMT (Generic Mapping Tools) 2026, JdF slab data from Sypus, et al., (2024).*

3 Methods

3.1 ObsPy.TauPy Software

ObsPy is a Python based toolbox created to simplify the usage of programming for seismologists (Beyreuther et al., 2009). ObsPy extends Python by providing direct access to modules used in seismological computing and cuts down the preprocessing of data necessary to perform programming without this software, while still incorporating powerful and commonly used Python programming modules such as NumPy (Harris et al., 2020), SciPy (Virtanen et al., 2020), and visualization tool Matplotlib (Hunter, 2007; Beyreuther et al., 2009). ObsPy provides access to basic functionalities needed in Python for seismological computations and is made to be easily combined with other more in depth computing modules. Specifically used in this research is the ObsPy package TauPy (Crotwell et al., 1999) – which provides tools for calculating seismic travel times and producing seismic ray path models. This package allows for the computation of a large range of seismic phases for defined 1-dimensional velocity model boundaries and computes the travel times by integrating ray paths analytically in each defined layer.

3.2 Pykonal Software

Pykonal is an open-source Python package for 2-D and 3-D seismological computations, such as travel times and ray tracing in heterogeneous media (White et al., 2020). Pykonal computes the ray paths and travel times by solving the Eikonal equation using the fast marching method (FMM) and can be used in either cartesian or spherical coordinates at any scale. This package enables a variety of seismic imaging applications, such as locating earthquakes, seismic tomography analysis, and in this case computing ray path and travel time parameters, all within a relatively simple and efficient coding structure (White et al., 2020).

The Eikonal equation is a non-linear first order partial differential equation defined as the high frequency limit of the full wave equation. It is derived from the homogeneous scalar wave equation, and by using the high-frequency approximation (limit where $\omega \to \infty$) to give:

$$|\nabla\tau|^2 = \frac{1}{v^2} \tag{7}$$

where $\mathbf{v}(\mathbf{r})$ is the given velocity field and $\boldsymbol{\tau}(\mathbf{r})$ is the travel time field, where $\boldsymbol{\tau}$ is found by solving equation 7. Here, the high-frequency approximation is acceptable as the wavelength of the propagating wave is much shorter than the scale of the velocity structure (White et al., 2020). In Pykonal, this equation is solved using the efficient numerical fast marching method, which was initially introduced as a method to solve this equation by Sethian (1996), who provides great detail into the derivation and proof of how the FMM is used to solve the Eikonal equation.

3.3 Velocity Models

3.3.1 The CN03 Model

The first and simplest velocity model used for the study in this research is the 1-D CN03 velocity model which includes discrete velocity values from 0 to 45 km depth for P and S wave velocities in km/s, for

southwestern BC (Rogers, 1983; Horner et al., 1983). This velocity model was derived from locating local earthquakes and adjusting velocities and densities to minimize residuals.

Furthermore, to be used with the more complex software Pykonal, this velocity model was spliced into the shallow layers of the global IASP91 velocity model (Kennett & Engdahl, 1991) and edited to include a 7th column including earth radius in kilometers.

3.3.2 The Savard Velocity Model

The 2-D velocity model used was extracted from the 3-D model of Savard (2018). In this work, the 3-D velocity structure of southern Vancouver Island was inverted using double-difference tomography techniques. For use in 2-D modelling, a profile of depth and longitude was extracted for a constant latitude of 48.6°N. This W-E profile was chosen as it passes through the region of highest station density as well as the event location. To extract this profile, the V_P and V_S data were extracted for that specific latitude for each defined depth unit in the velocity model files. The velocity gridding and coordinate system used in this modelling was extracted from the model grid info, with the z grid spacing used for depth and the x grid spacing used for longitude (Savard, 2018).

3.4 Study Location and Stations Used

To investigate the March 3rd, 2025, seismic wave arrivals, a span of receiver stations in the Raspberry Shake network were chosen around the region (figure 15). The majority of these stations are part of the SchoolShake network in Victoria, but some privately owned stations were also selected for the analysis, specifically to have at least one station in each ‘quadrant’ surrounding the epicenter. For each selected station, the P, S, and bonus arrival times (when present) were compiled from the recorded seismograms. A spatially representative subsection of these stations is used in subsequent modelling procedures.

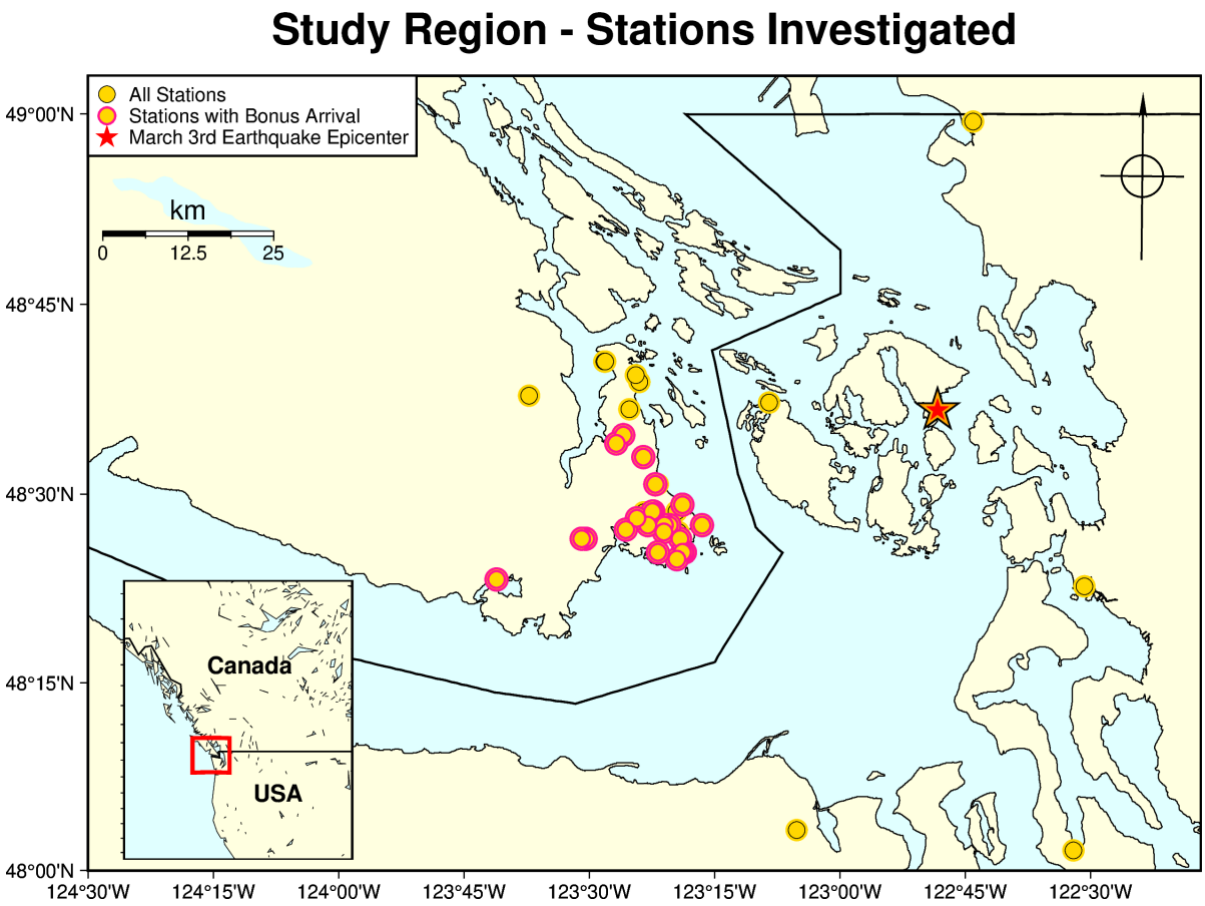


Figure 15 *– Map of study region including seismic stations (yellow circles). Stations which received bonus arrival (red outline) lie over the specific azimuthal range of 219-256°, up-dip from the March 3rd epicenter (red star). Majority of stations investigated are from the SchoolShake network, with a spatially representative subsection of these stations used in the modelling computations. Figure created using GMT, 2026.*

The RS seismograms chosen for this study all measure the EHZ (vertical) component and are all part of the AM network. Station coordinates were extracted from the SAGE (Seismological Facility for the Advancement of Geoscience) station viewer (EarthScope Consortium, 2026); coordinates have a random ~500 m offset applied to their true location to preserve users' privacy. Table 1 includes a summary of the 36 RS stations plotted in figures 13 and 15 - their location, azimuth relative to the March 3rd event, and whether the bonus arrival is observed or not. For use in the computational analysis, only a spatially representative array of stations were used; these 16 are bolded. Note that a map of these 16 stations labelled by name is provided in the appendix (A).

Table 1: Parameters of Stations Plotted on Study Location Map – (subsection of stations used in computational analysis bolded)

Station	Latitude (°N)	Longitude (°W)	Azimuth Rel. March 3rd, 2025, event (°)	Bonus Arrival Recorded (Y/N)
AM R7D30	48.621	123.141	272.704	N
AM RA7D3	48.378	122.512	140.084	N
AM RF44C	48.459	123.275	244.145	Y
AM R16DA	48.486	123.314	249.837	Y
AM R54F7	48.477	123.326	248.965	N
AM RF395	48.450	123.322	245.008	N
AM R1B48	48.411	123.319	243.655	Y
AM R37A5	48.423	123.309	240.804	Y
AM RF5C1	48.423	123.314	241.047	Y
AM R3CDE	48.513	123.363	255.347	N
AM RF926	48.513	123.368	255.475	Y
AM R3978	48.459	123.340	246.978	Y
AM RC45E	48.648	123.400	275.598	N
AM R9920	48.459	123.350	257.364	Y
AM R6A6F	48.477	123.372	250.567	Y
AM RC2BF	48.477	123.374	250.632	Y
AM R0EBF	48.414	123.325	240.431	Y
AM R0D0E	48.450	123.351	246.202	Y
AM R01DF	48.657	123.407	276.821	N
AM R4BAD	48.612	123.420	270.372	Y
AM R664C	48.476	123.391	251.033	Y
AM RD5F3	48.578	123.432	265.686	Y
AM R4081	48.459	123.383	248.556	Y
AM R42D0	48.423	123.363	243.250	Y
AM RB819	48.468	123.405	250.428	Y
AM R6650	48.567	123.446	264.316	Y
AM R95F2	48.453	123.427	249.247	Y
AM RBD2C	48.675	123.470	278.537	N
AM RDA8D	48.657	123.468	278.560	N
AM RC118	48.441	123.507	250.178	N
AM R4014	48.441	123.515	250.390	Y
AM RD862	48.630	123.620	272.326	Y
AM RA825	48.027	122.534	162.750	N
AM RE0FF	48.387	123.685	249.316	N
AM R382B	48.990	122.734	7.008	N
AM R09BB	48.054	123.086	198.646	Y

When producing the modelled ray paths and computing the travel paths for the 2-D case, two specific cross sections of the region were chosen. Profile A runs from west to east directly through the event location (at 48.6°N) and profile B runs SW-NE to match the NE dip of the JdF slab and to cross the dense region of stations that recorded the bonus arrival (figure 16).

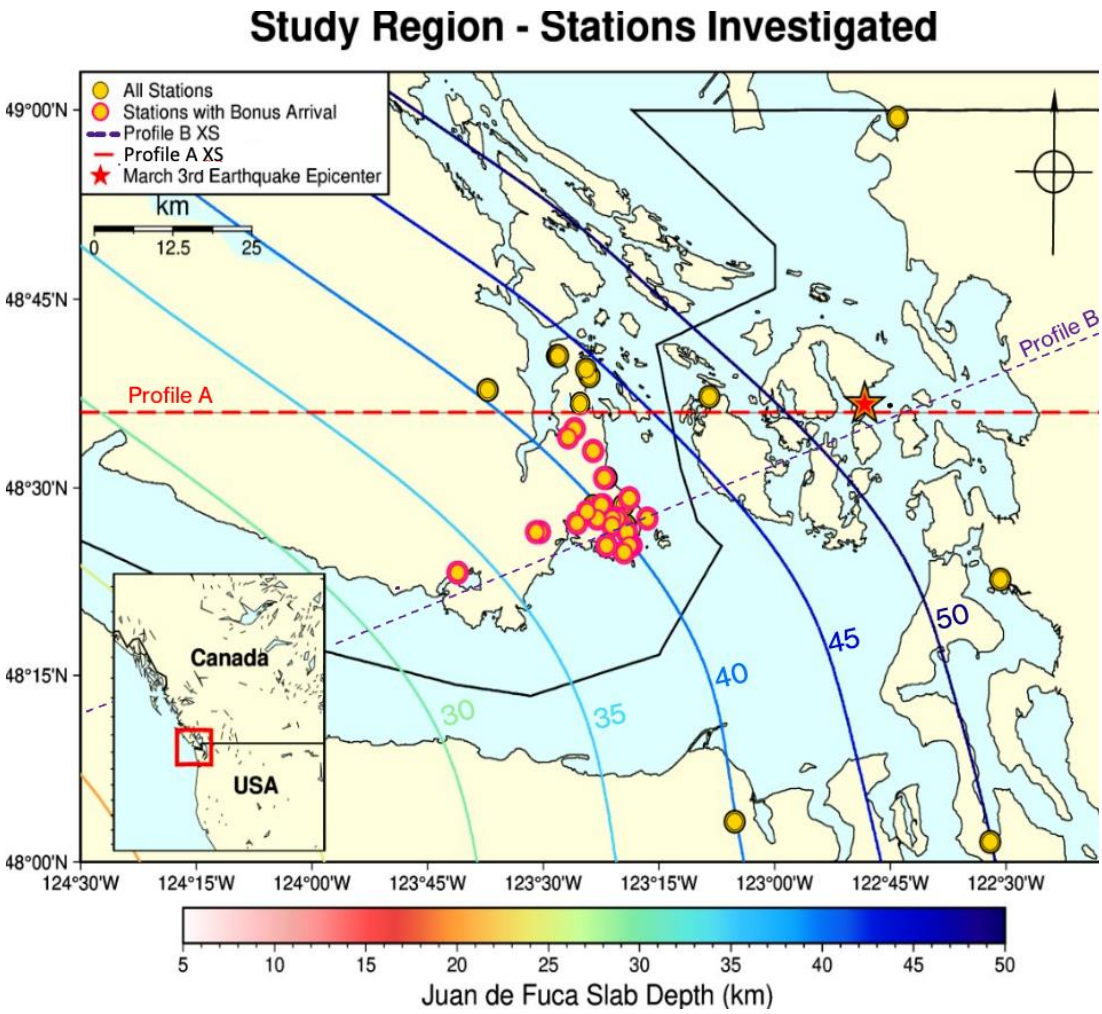


Figure 16 *– Map of study region including the cross sectional profiles used in the analysis: W-E Profile A at 48.6°N and the SW-NE profile B. Depth contours of the subducting JdF plate are overlain* (Sypus & Wang, 2024). *The plot includes stations investigated (yellow circles), and those which received the bonus arrival (red outlined stations). Figure created using GMT, 2026.*

3.5 Procedure

3.5.1 1-D ObsPy.TauPy Modelling Procedure

Using ObsPy.TauPy**,** seismic travel times and ray paths were calculated for the March 3rd, 2025, Orcas Island event by implementing algorithms from the TauPy toolkit. The code produced for the computational procedure in the ObsPy.TauPy procedure is included in Appendix B. The 1-D CN03 velocity model (Rogers, 1983) was read into the program and converted from a '.nd' file to a TauPy compatible format using the build_taup_model function, changing the file to a '.npz' format. This model was then loaded using the TauPyModel class for use in travel time calculations.

The specific parameters of the earthquake source (latitude, longitude, and depth), as obtained from the USGS earthquake catalogue (USGS, 2025), were defined. Next, the coordinates for the network of 16 chosen stations for analysis were defined, as listed in table 1 (bolded stations). Then, the theoretical seismic travel times for each station were calculated using the get_travel_times_geo method in the TauPyModel, which accounts for the geographic locations of both the earthquake focus and the receiver. These travel times were computed for phases: P (direct P wave arrival), S (direct S wave arrival), Pvmp (P wave topside reflection off the Moho), and Svms (S wave topside reflection off the Moho). This Moho boundary corresponds to a horizontal discontinuity in the velocity model at 30 km depth.

The ray paths taken by these specific seismic phases were computed using the get_ray_paths_geo function, which again takes as arguments the specific geographic location of the source and of each receiver. Separate ray path calculations were performed and plotted for P wave phases (both P and Pvmp) and S wave phases (S and Svms). The resulting ray path geometries were plotted as longitude vs depth profiles from the TauPy output, using Matplotlib (Hunter, 2007). The horizontal unit produced by the TauPy ray_paths.plot_rays function is expressed as angular distance in degrees from the source. For interpretation purposes, this axis was changed to longitude by projecting each point on the computed ray path onto a great circle using spherical trigonometry, to incorporate not only each station's angular distance from the source but also its azimuth relative to the source. The ray paths were then plotted as longitude vs depth. Horizontal reference lines were added to the plots at 0 km (surface) and at 30 km depth (Moho). Separate P and S wave plots were produced for each of the 16 stations.

This procedure was repeated for a custom shifted reflector boundary in the 1-D CN03 velocity model, wherein the boundary was shifted down to 35 km depth from the initial 30 km Moho reflector in the actual velocity model. Travel times were calculated, and ray paths were plotted for the same specific array of seismic phases for all 16 stations, using an identical process as for the original CN03 model.

3.5.2 1-D Pykonal Procedure

The following procedure for the 1-D Pykonal model was produced through a modification of the 'Tracing_PKIKP_rays' Pykonal tutorial written by Malcolm White and Joachim B Haga in conjunction with the 2020 publication on the software, as sourced from the Pykonal GitHub repository (White, et al., 2020). This tutorial was modified to trace reflected rays on a smaller scale in order to image the March 3rd, 2025, event. The code produced for the computational procedure using Pykonal is included in Appendix C. This 1-D model was produced using the CN03 velocity model, which was modified to include a column for earth radius and spliced into the shallow depths of the IASPI91 velocity model (Kennett et al., 1991), a needed format for use in Pykonal. This procedure was done for the most part to obtain a better idea of how to use Pykonal in general and how its tools can be applied to a study in this region. By first performing the Pykonal modelling with the 1-D velocity model, I was able to make sure I understood what Pykonal was doing and that the results made sense before proceeding with the more complex 2-D velocity model.

First, the direct P-wave travel times and ray paths were calculated following a similar workflow that was used in the ObsPy procedure. The 1-D P-wave velocity model from the CN03 model was read into Python and was separated into depth, radius, and P-wave velocity arrays. As Pykonal requires velocity data at every node across the grid, the 1-D model was interpolated onto the computational grid using a linear interpolation. This interpolation mapped the 1-D CN03 velocity values onto the radial coordinates of the computational domain as the 1-D Pykonal model was produced in spherical coordinates. This was done as the travel-time field was calculated in spherical coordinates, and the grid resolution was defined as 512 nodes in the radial and phi direction, and 5 nodes in the theta direction. The radial domain was defined to be 100 km deep and was set by taking the earth's radius to be a uniform 6371 km and subtracting 100 km to be the minimum radius, and the max radius being the entire 6371 km radius. This is how the model was modified to focus on the crustal and shallow mantle propagation relevant to this study. The longitudinal grid was determined from the maximum and minimum longitudes of the 16 seismic stations used in the computation. A small buffer of 0.5° was added on to each side to ensure the domain covered all ray paths. To approximate the 2-D slice, as we are working with a 1-

D velocity model interpolated onto a 2-D profile, the theta dimension was set to only 5 nodes centered on the event source to approximate the 2-D slice while retaining the Pykonal 3-D spherical coordinate framework required by the solver.

The 16 seismic stations included in the numerical computation were defined using their geographical coordinates and then converted into spherical coordinates to match the solver. Here, the radius was set to 6371 km as the stations are all on the surface, the theta coordinate was set to a constant $\pi/2$ slice to work in the 2-D domain, and the phi coordinate was set to the longitude value of each station in radians. The source location was defined using the known geographical coordinates and depth, the depth was converted to radial distance. To ensure smooth and consistent stability in the Eikonal solver, the source location was snapped to the nearest grid node in the computational grid.

Next, the travel time calculation was done using the point source solver in spherical coordinates implemented in Pykonal. This solver solves the Eikonal equation to compute the minimum travel time to every node in the grid for the wavefront propagating from the source node. Once the travel time field was computed, the travel time for the direct P wave arrival to each station was computed by interpolating the travel time field to each set of exact receiver coordinates.

Finally, in the direct P wave computation using the 1-D Pykonal method, the ray paths were computed and plotted. This was done by tracing the gradient of the travel time field from each station back towards the source, as done using the solver.trace_ray() function with the argument being the receiver location. This is done using the gradient to find the steepest direction in travel time which corresponds to the physical wave propagation taking the path of minimum time. The resulting ray paths were extracted as sequences of nodes in spherical coordinates and converted to longitude and depth for visualization on the 2-D plots. To focus on shallow crustal earthquakes the ray segment computational region was limited to 65 km depth. Then, the velocity model, earthquake source, seismic stations, and ray paths were plotted using Matplotlib (Hunter, 2007). The velocity model colormap represents the interpolated 1-D CN03 field in longitude depth space. The geometry of the Juan de Fuca slab was included on the plot to provide context of the 2-D slice, the Juan de Fuca slab position was extracted for the computational cross section (at 48.6°N) from Sypus et al. (2024).

For the direct S wave travel time and ray path computation using Pykonal and the interpolated 1-D CN03 model, the exact procedure as above was repeated for the column of V_S velocities extracted from the velocity file. An identical solving, modelling, and plotting routine was carried out to produce a figure of the direct arrival S waves as well as their travel times from the source to station.

3.5.3 2-D Pykonal Procedure

The next stage of model complexity integrated into this sequence of models is the use of a 2-D velocity model to produce calculated seismic phase travel times and ray paths for the direct P and S waves, and also to model the reflections off the JdF slab to test the hypothesis that this is the origin of the bonus arrivals. The 2-D model used in this procedure and all following 2-D models is the Savard model extracted to a profile of 48.6°N (Profile A, figure 16) (Savard, 2018). The code produced for the computational procedure using Pykonal is included in Appendix C.

3.5.4 2-D Pykonal Direct P and S wave Procedure

To produce the modelled arrival times and ray paths for the direct P wave arrivals in the 2-D Pykonal setup, the 2-D P wave velocity model was read in as a '.txt' file and converted to a '.npz' file. The profile longitude and depth grid spacing values were sourced from Savard's grid info (Savard, 2018), and were limited to include only a grid range for depth of 0 to 90 km spaced by 3 km for depth and a longitude grid of -200 to 208° spaced by 14° with the origin of this coordinate system corresponding to (48.75°N,124.0°W) with no coordinate system rotation. The depth values and longitudes of the 2-D profile were taken exactly from this grid definition to define the velocity structure. As the solver requires a 3-D velocity field, the 2-D field was replicated across a small number of nodes in the y direction (as was done with theta in the 1-D interpolation previously). This approach allows the cartesian coordinates used in this run of the solver to maintain effectively 2-D limited wave propagation. Using this grid the 2-D Pykonal model was done in cartesian coordinates. As the seismic stations and the earthquake location were originally defined in geographic coordinates, the next step taken was to convert the longitudes of the stations and the event to horizontal distances in kilometers relative to the reference longitude of 124°W (origin of coordinate system). The computational grid was then defined using a cartesian coordinate system with node spacing determined from the velocity model grid info. The origin of the grid (min_coords) was set to the minimum longitude grid node of the model and at the shallowest depth. Therefore, at this point the solver has been created to span the full horizontal extent of the velocity profile and for depths of 0-90 km.

Next, the source location was defined using its geographic location and depth, with longitude converted to distance along the profile as was done for the stations. The y coordinate of the source was set to 0 as this corresponds to the central plane of the quasi 2-D model. As was done in the 1-D Pykonal run, the location of the earthquake was snapped to the nearest node in the grid before solving the Eikonal equation. The seismic stations were positioned along the model profile with each station's location corresponding to x = longitude in kilometers from the origin, y=0 and z=0.

Following the location definition of the receivers and the source, the first arriving P wave travel times were calculated, using the point source solver in Pykonal in cartesian coordinates. Then, the seismic ray paths for the direct P arrival were calculated and produced by the solver as sequences of coordinates within the defined grid and were then converted back to longitude and depth for visualization. Again, the geometry of the subducting JdF slab was read in as manually defined points for this latitude slice, as this profile was defined for a range of longitudes per depth value, an average longitude was assigned to each depth value at an interval of 5 km and a linear interpolation was applied to generate a continuous slab for visualization (Sypus, et al., 2024). At this point, the plot was produced. The 2-D gridded velocity model, the earthquake source, the seismic stations, the ray paths, and the JdF slab geometry were all plotted on the same figure of longitude vs depth using Matplotlib (Hunter, 2007). The final plot represents the propagation of direct P waves through shallow depths beneath the study area over the 2-D velocity model. This figure was plotted for varying longitude ranges, one plot to visualize areas close to the earthquake source and one at a larger longitude range to view the entire velocity model area.

As in the 1-D Pykonal case, the same procedure was repeated to model the ray travel times and ray paths for the direct arrival S wave. The V_S model was similarly extracted for a 48.6°N cross section of the Savard velocity model (Savard, 2018) and loaded in and converted to a '.npz' file, using the same defined

grid spacing in x and z. Following an identical procedure as for the direct P wave arrival in the 2-D Pykonal model, the direct S wave travel times were calculated, and the same plot was made of the ray paths for the direct S wave arrival.

3.5.5 2-D Pykonal Reflected Pp and Ps wave Procedure

The next goal using the 2-D Pykonal V_P and V_S modelling routine was to produce travel times and ray paths for the hypothesized origin of the bonus arrivals to compare with the actual recorded arrival times. To do this both a Pp and Ps reflection off the JdF slab were modeled using the same solving routine format as for the direct arrivals. This had to be done by splitting the reflection path into a 2 segment solver and travel time calculation, where the total travel time was taken to be the sum of the downgoing ray to the reflection point and upgoing ray to the receiver separately due to Pykonal limitations.

For the Pp reflection, the travel time field, receiver location definition, and source location used were the same as for the direct arrival P wave defined previously. To compute the travel time for the upward propagating ray, a separate travel time solver was used for each station. For each receiver a new Pykonal point source solver was initialized on the same computational grid. Following this, each receiver was defined as a source, as Pykonal computes the travel times and ray paths from receiver to the source. Then, the Eikonal equation was solved to compute a travel time field to represent the minimum travel time from each station to every node in the grid. Next, to define where the rays are reflecting off the JdF slab, the slab was discretized into a series of points defined by their x and z locations on the grid and listed into a set of possible reflection points for the reflection to occur. Then, for each possible point of reflection on the slab, two travel times were calculated: that of the downgoing ray from the earthquake and that of the upgoing ray from slab to station. The total travel time was calculated as the sum of these two legs. For each station, the slab point that minimized the total travel time was selected as the reflection point for that ray. The coordinates of the determined reflection points for all stations were recorded for ray tracing. The minimum total travel time for each ray path is taken to be the predicted Pp phase arrival time at each station.

To plot the ray path of the two segment solver, the downgoing segment was traced from the reflection point back to the source and the upgoing segment was traced from the reflection point to the receiver for each station. Note that due to the fact that this ray path is determined by computing the gradient between nodes, the ray path could not be traced to the nodes at 0 km (surface) as there are no defined nodes above. Therefore, in the reflection plots where the reflection point is treated as the source to the receiver, the plotted rays stop at the nearest grid point (3 km depth). This does not affect the calculated travel times, which are based on the full ray path. Once again, Matplotlib was used to create the ray path plot on top of the 2-D V_P grid.

In addition to the Pp reflections, phases converted to S waves upon reflection (Ps) off the JdF slab were also modelled to investigate the possibility of Ps waves producing the bonus seismic wave arrivals. This Ps wave was modelled in an identical way as the Pp ray, in the multiple path segment; however, in this case the V_S model was used in the computation of the upgoing ray. The modelling approach followed the same minimized travel time reflection point determination and the same plotting routine. The total travel time of the Ps wave was then computed for each station to be the travel time of the downgoing wave in the P wave velocity model summed with the upgoing ray travel time in the S wave velocity model. The Ps reflections were plotted overlain on the V_P grid even though the upgoing ray was solved using the V_S model.

3.5.6 Pseudo 3-D modelling using Profile B

To determine if the use of a different cross sectional profile of the region had an effect on the accuracy of travel times for reflected bonus arrivals in this region, the SW-NE profile (profile B, figure 16) slab depths were used to re-compute the reflected arrival times for a reflection off the JdF slab. To do this, the same 2-D W-E cross section of the velocity model was used, as the definition of a diagonal 2-D cross section from the Savard (2018) model would have been difficult to extract, and we assume the velocity structure above the slab to be relatively homogeneously distributed in the N-S direction in this limited study area. To model a reflection off the JdF slab using this cross-sectional profile, the longitudinal ranges and depth values (at 5 km intervals) for each place the profile crossed the depth contours were extracted and used in plotting the JdF slab along that profile atop the profile A velocity model. The result of extracting a diagonal profile perpendicular to the slab and plotting it along a constant latitude is that the plotted profile has a step like appearance. Then, the same modelling routine used for the previous JdF reflections (Pp and Ps) was repeated for the profile B cross section.

3.6 Travel Time Residual Calculations and Visualization

Travel times were produced for each of the defined model routines above: 1-D ObsPy direct P and S, 1-D Pykonal direct P and S, and 2-D Pykonal direct P and S and reflected Pp and Ps waves, to each of the 16 stations in the analysis. For comparison of the predicted travel times with the actual phase arrival times, the travel times were first converted to arrival times by adding the modeled time to the earthquake origin time. This produced predicted UTC arrival times for each station in each model routine. The actual measured arrival time of the P and S wave for each station were obtained from C. Brillon (personal communication, March 2025). These actual times were converted to UTC timestamps for direct comparison with the predicted arrivals. Furthermore, for the bonus arrivals modeled from the JdF reflection, comparison was to be made with an array of actual arrival times of the bonus phase arrival at each of the stations that had the bonus arrival. These arrival times were manually picked from the recorded March 3rd, 2025, seismograms.

To quantify how well each model performed in producing arrival times for the P, S, and bonus arrivals, the difference between the modelled and observed arrival times was calculated for every predicted travel time as the residual between them:

$$Residual = t_{modelled} - t_{actual}$$

where $t_{modelled}$ is the event time plus the calculated travel time, and $t_{observed}$ is the actual recorded arrival time. Residuals were calculated separately for each modelling approach to identify the overall effectiveness of each model. To evaluate each model's performance, some summary statistics were calculated: mean residual, mean absolute residual, and the standard deviation of residuals. The mean residual was calculated including positive and negative residuals to show which way the model is biased in producing arrival times too long, too short, or of no bias compared to the actual arrival times. The mean absolute residual is a measure of the overall offset between the predicted and absolute arrival times, while the standard deviation reflects the spread of the residual distribution. These metrics, while over a statistically insignificant sample of stations, allows us to compare the performance of each of the modelling routines overall.

The reflection residuals were computed with the predicted Pp and Ps arrival times, using the manually

chosen bonus arrivals as the actual arrival time. As this residual was calculated even for stations that didn't observe the bonus arrival before the S arrival ($t_{actual} = 0$), we see that for these stations the residual ended up being extremely large and obviously not reflective of a true residual. These large residuals were excluded from the statistical calculations, which were only performed on stations that actually recorded a bonus arrival, and therefore had a conceivable residual. The same statistical values were calculated for the modelled bonus arrivals.

Next, in order to visualize the residuals of each model run, I plotted them as a function of azimuth relative to the March 3rd, 2025, earthquake epicenter; this was done to investigate whether the model showed azimuthal dependent trends in the residuals. For each model, the residuals were plotted as a scatter plot for each station on a plot of azimuth vs residual; direct residuals vs azimuth for each model were plotted separately from those of the reflected arrivals computed using the bonus arrival times. A horizontal dashed line was added to each figure representing the mean residual, and a horizontal dotted line was plotted as the mean absolute residual. Furthermore, error bars representing the standard deviation were plotted centered on the mean residual. Stations were each assigned a different color to distinguish individual data points and for use in identifying trends. Overall, the residual calculations were made to assess the accuracy of each of the seismic modelling routines for the March 3rd, 2025, event.

4 1-D Direct and Reflected Modelling

The initial results from this study are shown in figures 17-22, which plot the 1 dimensional modelled ray travel paths and travel times for the 2025 Orcas Island event. This 1-D modelling was done in both ObsPy.TauPy and in Pykonal as described in section 3.

4.1 1-D ObsPy.TauPy Modelling Results

The modelled rays using the ObsPy.TauPy model are plotted separately for each station, rather than superimposing all stations ray tracings on one figure, as this model produced almost identical ray tracings for each station. Figure 17 presents the results of modelling the direct P and reflected Pvmp ray tracing for each of the 16 stations. The Pvmp reflection represents a topside reflection off the Moho, which is defined at 30 km depth in the 1-D CN03 velocity model (Rogers, 1983). Similar plots show the modelled direct S ray path as well as the Svms reflection (figure 18).

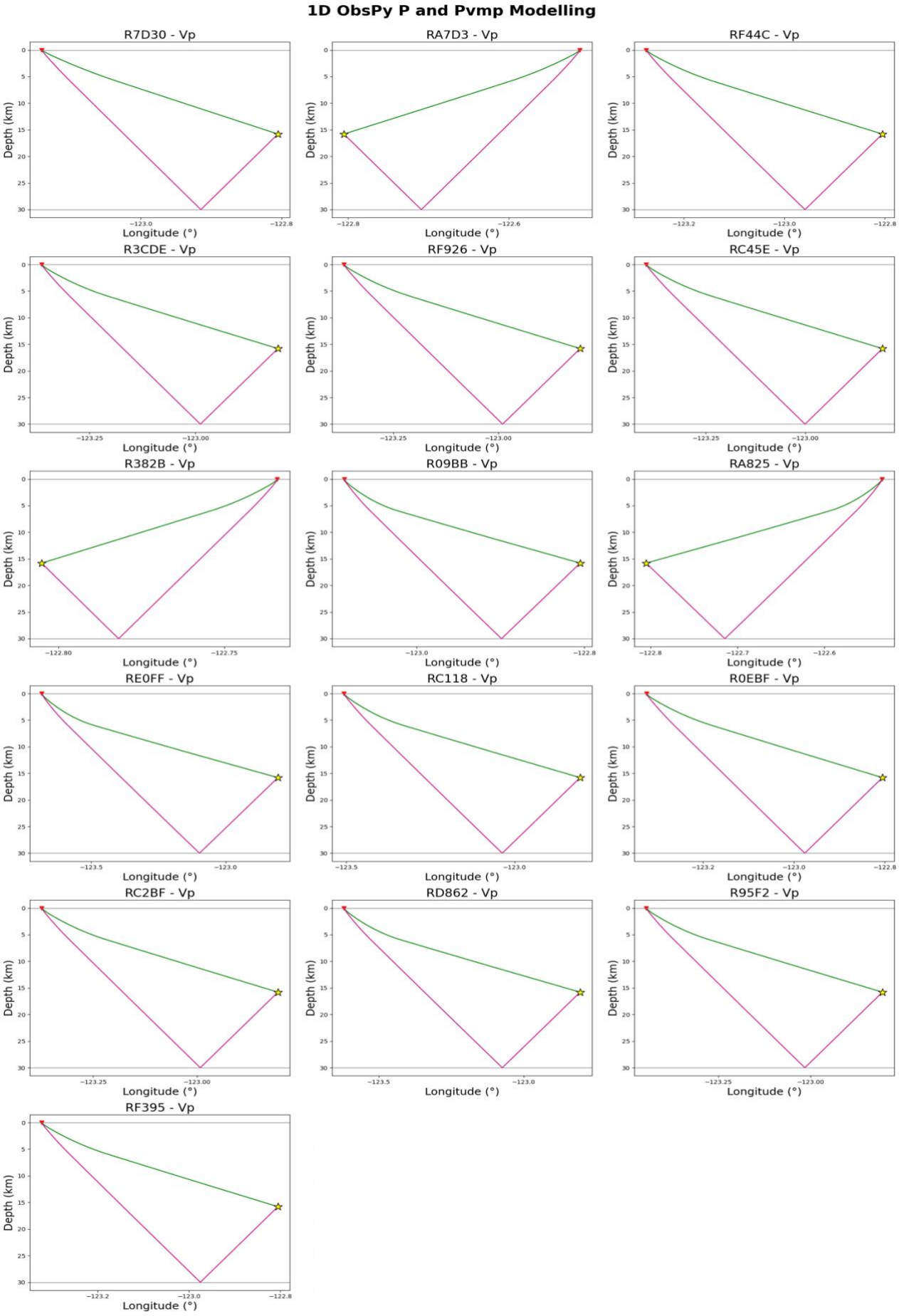


Figure 17 *– 1-D P (green path) and Pvmp (topside reflection off Moho, pink path) ray path modelling results using ObsPy.TauPy software (Crotwell et al., 1999) and CN03 V_P velocity model* (Rogers, 1983)*, from the March 3rd, 2025, earthquake hypocenter to each of the 16 seismic stations analyzed.*

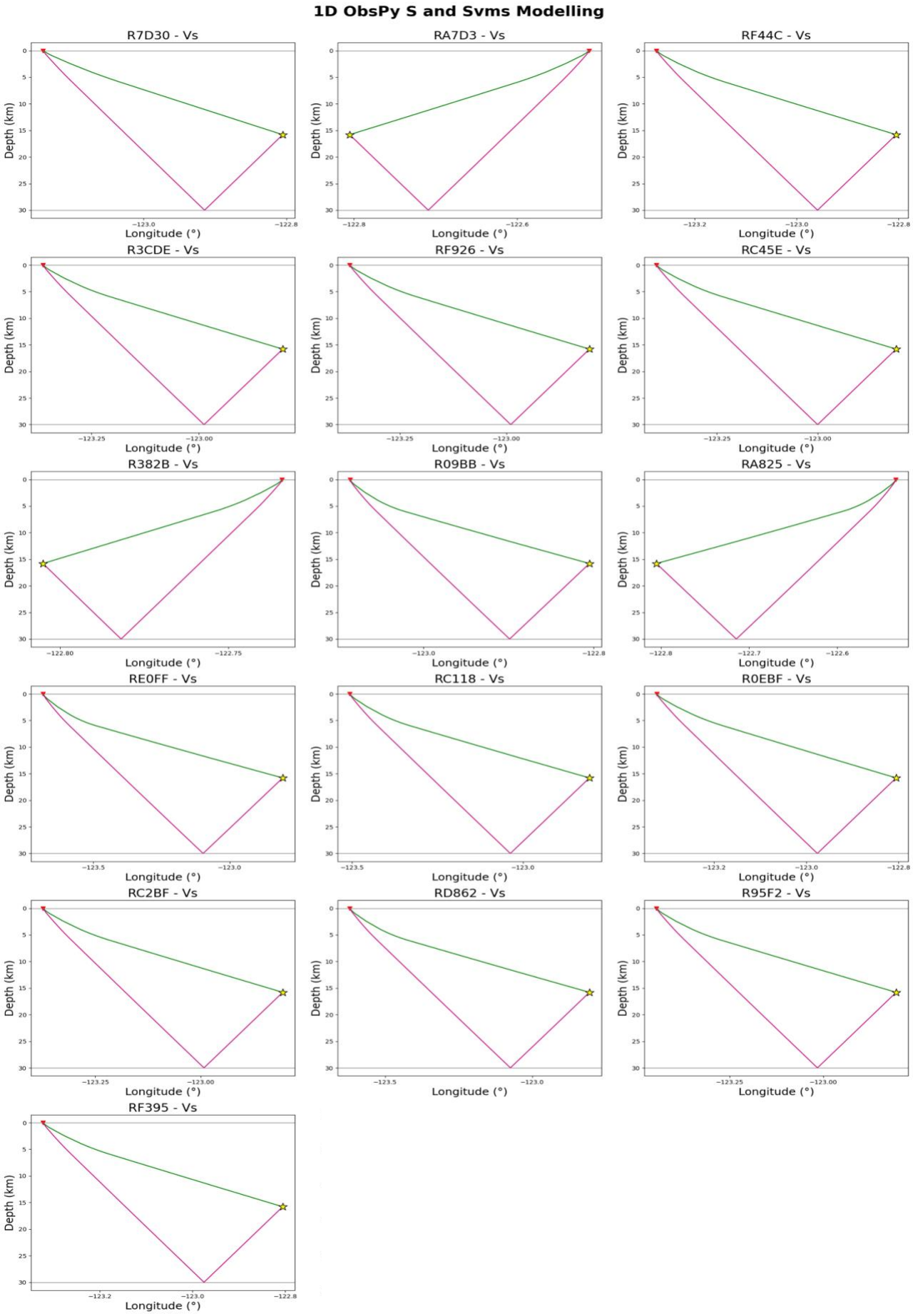


Figure 18 *– 1-D S (green path) and Svms (topside reflection off Moho, pink path) ray path modelling results using ObsPy.TauPy software (Crotwell et al., 1999) and CN03 V_S velocity model* (Rogers, 1983)*, from the March 3rd, 2025, earthquake hypocenter to each of the 16 seismic stations analyzed.*

For the 1-D ObsPy.TauPy modelling, the travel times were computed for ray paths to each of the 16 stations for each of the 4 seismic phases modelled. The computed 1-D ObsPy.TauPy travel times are summarized in table 2, which also provides the actual recorded travel times for the P, S and bonus wave arrivals (shown as '-' for stations where the bonus arrival is not observed between P and S arrivals).

Table 2: Calculated Travel Times from 1-D ObsPy.TauPy Modelling Routine

Station	**Modelled P Travel Times** (s)	**Modelled S Travel Times** (s)	**Modelled Pvmp Travel Times** (s)	**Modelled Svms Travel Times** (s)	**Actual P Travel Time** (s)	**Actual S Travel Time** (s)	**Actual Bonus Travel Time** (s)
R7D30	4.5011	7.7950	7.6384	13.2257	6.0	9.0	-
RA7D3	5.7019	9.8742	8.3808	14.5112	7.0	11.0	-
RF44C	6.3666	11.02525	8.8348	15.2972	7.0	12.0	10.01
R3CDE	6.9267	11.9949	9.2370	15.9936	8.0	13.0	-
RF926	6.9772	12.0823	9.2740	16.0577	8.0	14.0	-
RC45E	7.1307	12.3482	9.3875	16.2541	8.0	14.0	-
R382B	6.9236	11.9896	9.2347	15.9896	8.3	14.4	-
R09BB	10.227	17.7092	11.8700	20.5524	11.6	19.0	-
RA825	10.612	18.3764	12.1988	21.1216	12.0	21.0	-
RE0FF	10.832	18.7568	12.3877	21.4487	12.0	21.0	-
RC118	8.7297	15.1168	10.6291	18.4040	10.0	16.0	12.4
R0EBF	7.1598	12.3986	9.4092	16.2917	8.0	13.0	11.0
RC2BF	7.2073	12.4808	9.4445	16.3529	8.0	13.0	10.4
RD862	9.4415	16.3491	11.2111	19.4116	10.0	18.0	-
R95F2	7.6275	13.2083	9.7621	16.9028	9.0	15.0	11.6
RF395	6.8678	11.8930	9.1939	15.9190	8.0	13.0	11.03

Next, the residuals for each of the seismic phases computed in this model were computed (table 3). Note that, for stations which didn't observe the bonus arrival, the true arrival time is set to 0 seconds; therefore, the residual is very large and not meaningful (these stations are thus excluded from statistical analysis on the residuals).

Table 3: Residuals of Seismic Phases Computed in 1-D ObsPy.TauPy Modelling Routine

Station	**P Residual** (s)	**S Residual** (s)	**Pvmp Residual** (s)	**Svms Residual**(s)
R7D30	-1.4988	-1.2049	+46964.638	+46970.226
RA7D3	-1.2980	-1.1257	+46965.381	+46971.511
RF44C	-0.6333	-0.9747	-1.175	+5.287
R3CDE	-1.0732	-1.0050	+46966.237	+46972.994
RF926	-1.0227	-1.9176	+46966.274	+46973.058
RC45E	-0.8692	-1.6517	+46966.388	+46973.254
R382B	-1.3763	-2.4103	+46966.235	+46972.990
R09BB	-1.3728	-1.2907	+46968.870	+46977.552
RA825	-1.3875	-2.6235	+46969.199	+46978.122
RE0FF	-1.1678	-2.2431	+46969.388	+46978.449
RC118	-1.2702	-0.8831	-1.771	+6.004
R0EBF	-0.840	-0.6013	-0.991	+5.89
RC2BF	-0.7926	-0.5191	-1.555	+5.353
RD862	-0.5584	-1.650	+46968.211	+46976.412
R95F2	-1.3724	-1.7916	-1.838	+5.303
RF395	-1.1321	-1.1069	-1.836	+4.889

The next modelling step involved repeating the 1-D ObsPy.TauPy analysis but with the reflector boundary placed 5 km deeper, at 35 km, using a modified version of the 1-D CN03 velocity model (Rogers, 1983). Results are plotted for the same 4 seismic phases using the V_P and V_S models in figures 19 and 20; the direct P and S wave travel times are identical to the previous modelling step.

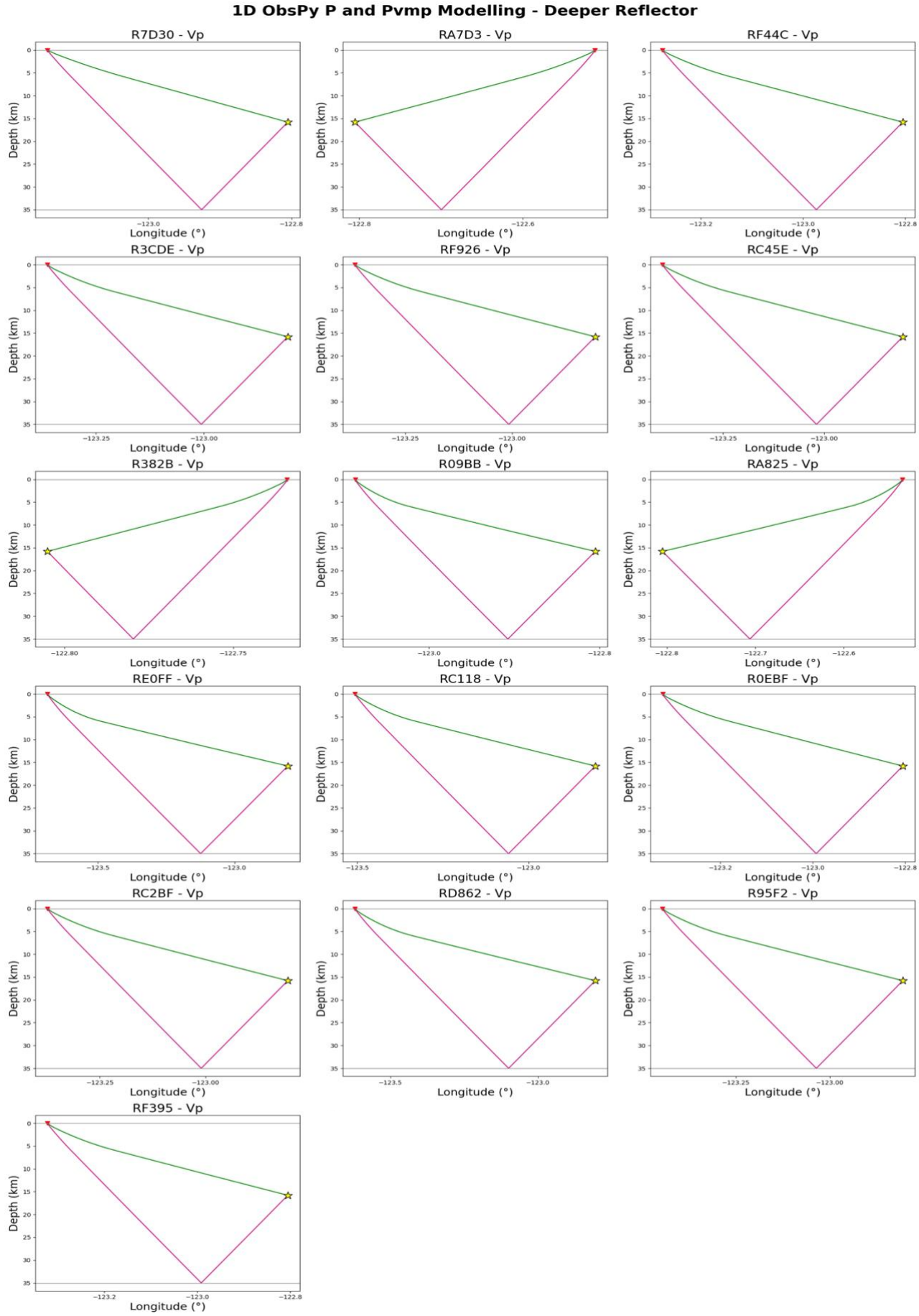


Figure 19 *– 1-D P (green path) and Pvmp (topside reflection off 'Moho', pink path) ray path modelling results using ObsPy.TauPy software (Crotwell et al., 1999) and modified CN03 V_P velocity model, with reflector boundary moved deeper to 35 km depth* (Rogers, 1983)*, from the March 3rd, 2025, earthquake hypocenter to each of the 16 seismic stations analyzed.*

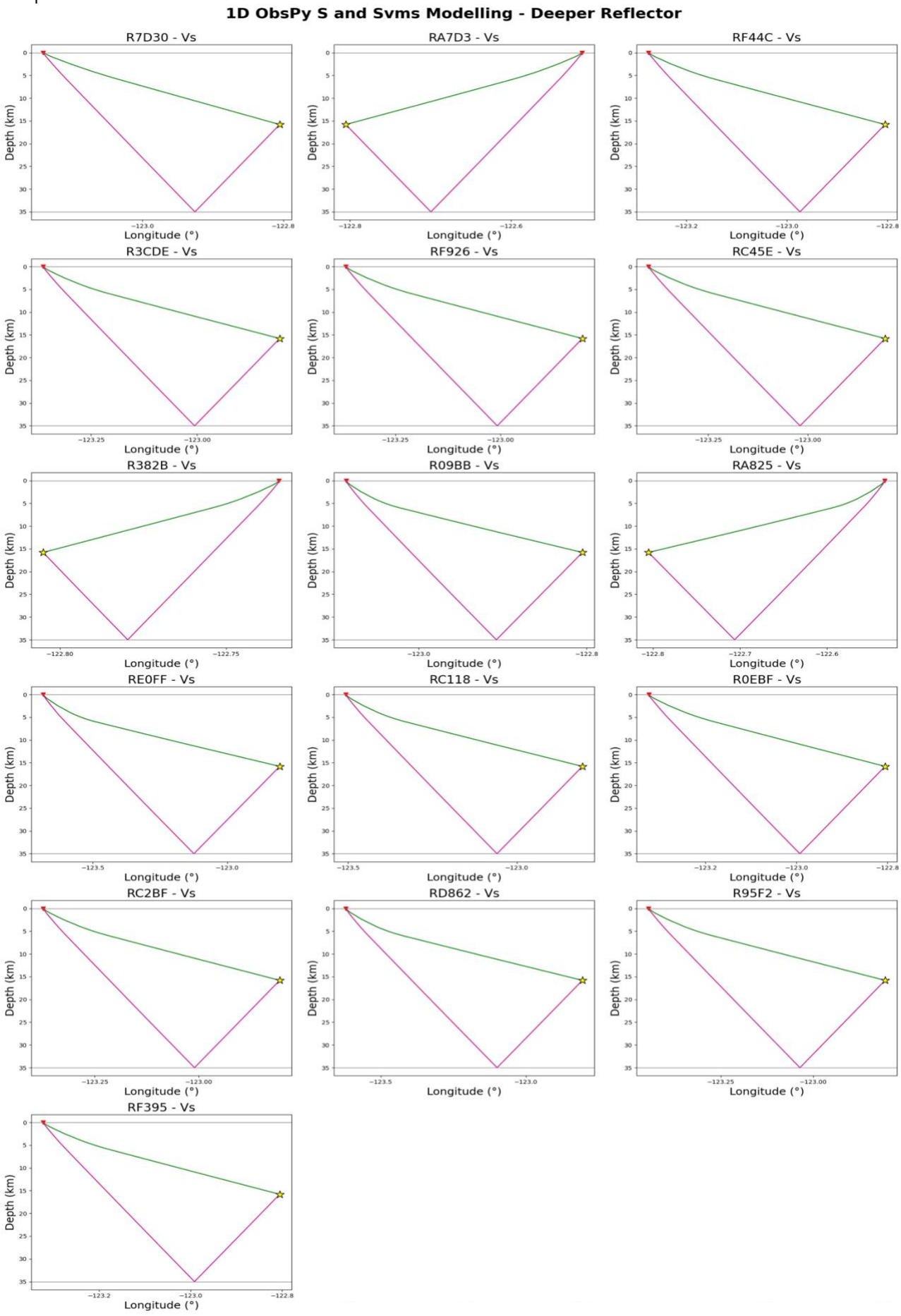


Figure 20 *– 1-D S (green path) and Svms (topside reflection off 'Moho', pink path) ray path modelling results using ObsPy.TauPy software (Crotwell et al., 1999) and modified CN03 V_S velocity model, with reflector boundary moved deeper to 35 km depth* (Rogers, 1983)*, from the March 3rd, 2025, earthquake hypocenter to each of the 16 seismic stations analyzed.*

The travel times from the 1-D ObsPy.TauPy deeper boundary reflection (35 km) are provided in table 4.

Table 4: Calculated Travel Times from 1-D ObsPy.TauPy Modelling Routine – Deeper Reflection

Station	**Modelled P Travel Times** (s)	**Modelled S Travel Times**(s)	**Modelled Pvmp Travel Times** (s)	**Modelled Svms Travel Times** (s)	**Actual P Travel Time** (s)	**Actual S Travel Time** (s)	**Actual Bonus Travel Time** (s)
R7D30	4.5011	7.7950	7.7950	15.5275	6.0	9.0	-
RA7D3	5.7019	9.8742	9.6055	16.6314	7.0	11.0	-
RF44C	6.3666	11.0252	10.0024	17.3186	7.0	12.0	10.01
R3CDE	6.9267	11.9949	10.3579	17.9341	8.0	13.0	-
RF926	6.9772	12.0823	10.3909	17.9911	8.0	14.0	-
RC45E	7.1307	12.3482	10.4918	18.1660	8.0	14.0	-
R382B	6.9236	11.9896	10.3559	17.9306	8.3	14.4	-
R09BB	10.2271	17.7092	12.7537	22.0822	11.6	19.0	-
RA825	10.6125	18.3764	13.0593	22.6111	12.0	21.0	-
RE0FF	10.8322	18.7568	13.2353	22.9160	12.0	21.0	-
RC118	8.7297	15.1168	11.6120	20.1053	10.0	16.0	12.4
R0EBF	7.1598	12.3986	10.5112	18.1994	8.0	13.0	11.0
RC2BF	7.2073	12.4808	10.5427	18.2540	8.0	13.0	10.4
RD862	9.4415	16.3491	12.1451	21.0283	10.0	18.0	-
R95F2	7.6275	13.2083	10.8270	18.7463	9.0	15.0	11.6
RF395	6.8678	11.8930	10.3197	17.8679	8.0	13.0	11.03

The residuals of the calculated depth change 1-D ObsPy.TauPy model are provided in table 5.

Table 5: Residuals of Seismic Phases Computed in 1-D ObsPy.TauPy Modelling Routine – Deeper Reflection

Station	**P Residual** (s)	**S Residual** (s)	**Pvmp Residual** (s)	**Svms Residual**(s)
R7D30	-1.4988	-1.2049	+46964.795	+46972.528
RA7D3	-1.2980	-1.1257	+46966.606	+46973.631
RF44C	-0.6333	-0.9747	-0.008	+7.309
R3CDE	-1.0732	-1.0050	+46967.358	+46974.934
RF926	-1.0227	-1.9176	+46967.391	+46974.991
RC45E	-0.8692	-1.6517	+46967.492	+46975.166
R382B	-1.3763	-2.4103	+46967.356	+46974.931
R09BB	-1.3728	-1.2907	+46969.754	+46979.082
RA825	-1.3875	-2.6235	+46970.059	+46979.611
RE0FF	-1.1678	-2.2431	+46970.235	+46979.916
RC118	-1.2702	-0.8831	-0.788	+7.705
R0EBF	-0.840	-0.6013	+0.111	+7.799
RC2BF	-0.7926	-0.5191	-0.457	+7.254
RD862	-0.5584	-1.650	+46969.145	+46978.028
R95F2	-1.3724	-1.7916	-0.773	+7.146
RF395	-1.1321	-1.1069	-0.710	+6.838

4.2 1-D Pykonal Modelling Results

Next, the 1-D Pykonal model results are presented for the ray paths of the direct P (figure 21) and S wave (figure 22) arrivals. The travel times were computed for these direct arrivals (table 6), as well as the residuals (table 7).

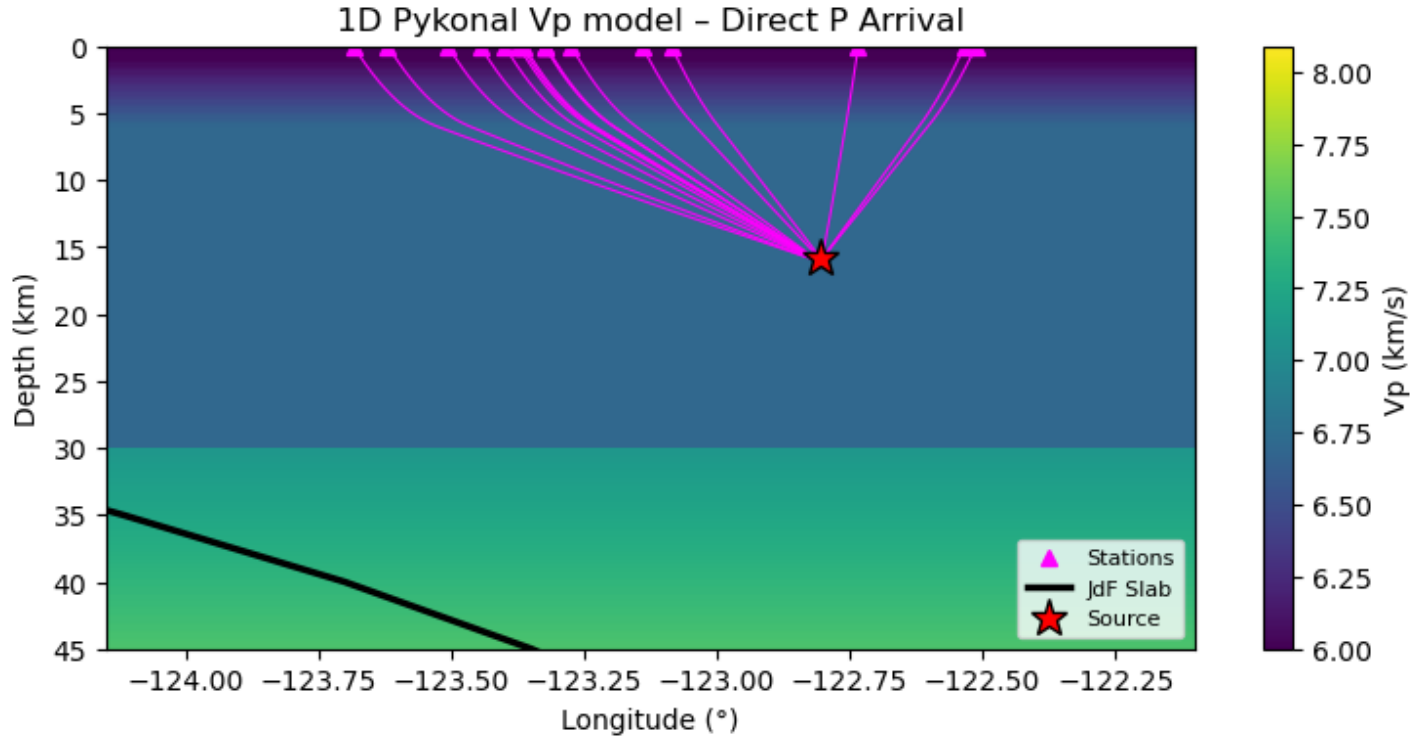


Figure 21 *– 1-D direct P wave arrival ray path modelling results using Pykonal (White et al., 2020) and the 1-D V_P CN03 velocity model interpolated over the 2-D region (Rogers, 1983), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed.*

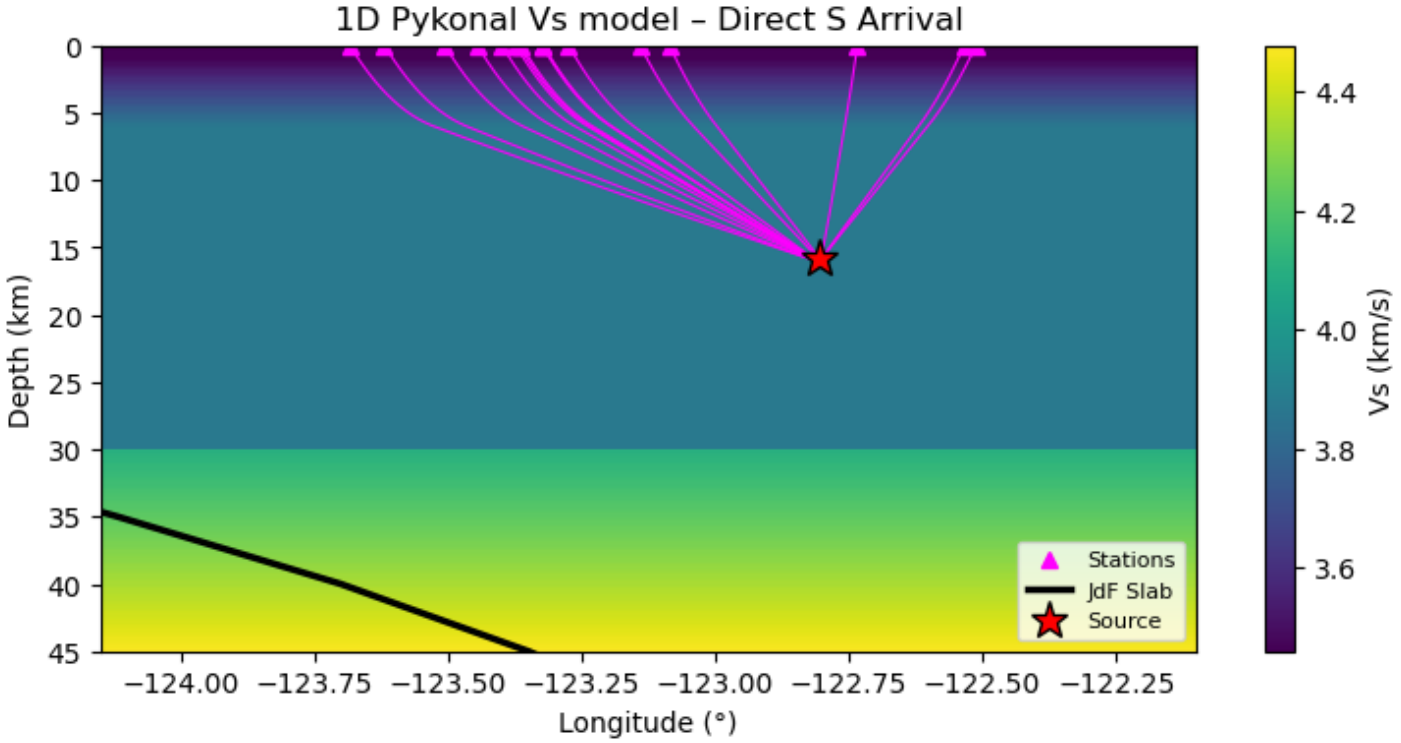


Figure 22 *– 1-D direct S wave arrival ray path modelling results using Pykonal (White et al., 2020) and the 1-D Vs CN03 velocity model interpolated over the 2-D region (Rogers, 1983), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed.*

Table 6: Calculated Travel Times from 1-D Pykonal Direct Arrival Modelling Routine

Station	**Modelled P Travel Times** (s)	**Modelled S Travel Times** (s)	**Actual P Travel Time** (s)	**Actual S Travel Time** (s)
R7D30	6.4768	11.2162	6.0	9.0
RA7D3	5.7773	10.0050	7.0	11.0
RF44C	8.5434	14.7944	7.0	12.0
R3CDE	9.9373	17.2078	8.0	13.0
RF926	10.0170	17.3459	8.0	14.0
RC45E	10.5287	18.2318	8.0	14.0
R382B	3.1238	5.4100	8.3	14.4
R09BB	5.6618	9.8051	11.6	19.0
RA825	5.4581	9.4522	12.0	21.0
RE0FF	15.1430	26.2206	12.0	21.0
RC118	12.2513	21.2142	10.0	16.0
R0EBF	9.3329	16.1613	8.0	13.0
RC2BF	10.1128	17.5117	8.0	13.0
RD862	14.0843	24.3876	10.0	18.0
R95F2	11.2673	19.5105	9.0	15.0
RF395	9.2853	16.0790	8.0	13.0

Table 7: Residuals of Seismic Phases Computed in 1-D Pykonal Direct Arrival Modelling Routine

Station	**P** (s)	**S** (s)
R7D30	0.4768	2.2162
RA7D3	-1.2226	-0.9949
RF44C	1.5434	2.7944
R3CDE	1.9373	4.2078
RF926	2.0170	3.3459
RC45E	2.5287	4.2318
R382B	-5.1761	-8.9899
R09BB	-5.9381	-9.1948
RA825	-6.5418	-11.5477
RE0FF	3.1430	5.2206
RC118	2.2513	5.2142
R0EBF	1.3329	3.1613
RC2BF	2.1128	4.5117
RD862	4.0843	6.3876
R95F2	2.2673	4.5105
RF395	1.2853	3.0790

4.3 Discussion of 1-D models

The primary goal of this research was to determine the origin of the bonus phase arrivals observed at local School Shake network seismographs across an azimuthal range of 219-256° during the March 3rd, 2025, Orcas Island earthquake. To do this, seismic ray path and travel time modelling was conducted using models of varying complexity.

This discussion of the 1-D modelling aims to evaluate the effectiveness of the simplest modelling routines in reproducing direct wave arrivals and to assess whether the bonus phase arrivals are consistent with a reflection, and at which reflector depth minimizes bonus arrival residuals. The functionality of the model type for modelling earthquake phase arrival times will also be discussed.

The initial 1-D ObsPy.TauPy model (section 4.1.1) is evaluated. Figures 17 and 18 show the 1-D ray path modelling for V_P and V_S using the CN03 model for the 16 selected stations. These figures include direct P and S waves, as well as Pvmp and Svms topside reflections off the Moho. The ray paths are nearly identical in shape across all stations, with differences only in propagation direction and reflection location to each individual station. This consistency between stations is expected, as the model has a defined 1-D velocity structure with discrete layers (boundaries at 0, 6, 30, 45 km depth), with a known horizontal velocity defined at 30 km depth corresponding to the Moho.

The computed travel times (table 2) show that direct P and S arrivals differ from observed arrival times by approximately 1–2 seconds. This small discrepancy indicates that the model performs well for modelling direct arrivals. Since these waves primarily travel through the relatively homogeneous shallow crust, the simplified 1-D velocity structure is sufficient to approximate their travel times accurately. In contrast, reflected phases are more sensitive to deeper structural complexity, considering this region overlies the dipping subducting JdF slab. Comparison of the Pvmp and Svms travel times with the observed bonus arrivals shows that Pvmp arrivals more closely match the observed timings, suggesting that the bonus phase is likely associated with a reflected P wave.

Residuals for each phase (table 3) further illustrate model performance. The residuals for direct P and S waves are consistently negative, ranging from approximately −0.5 to −2.5 seconds, indicating that the model systematically underestimates travel times. Despite this bias, the small magnitude of these residuals confirms that the model remains reasonably accurate for direct arrivals. For reflected phases, Pvmp residuals are also all negative but small (-0.9 to -1.8 s) for stations that recorded the bonus arrival between the P and S direct waves, reinforcing the interpretation of a P-wave reflection origin. In contrast, Svms residuals are positive and significantly larger (~5–6 seconds), indicating that these modelled arrivals occur too late to explain the observed bonus phase.

To further constrain the depth of the reflecting boundary, the Moho depth in the 1-D model was varied to minimize residuals to 35 km. The ray paths plotted for the deeper reflection (figures 19 and 20) are not very visually different than the 30 km reflection ray paths (figures 17 and 18). However, by increasing the reflector depth from 30 to 35 km, the Pvmp residuals were reduced to approximately −0.1 to −0.8 seconds (tables 4 and 5) representing a substantial improvement. This result suggests that a deeper reflecting interface than the Moho at 30 km depth better explains the observed bonus arrivals. Also, the need to artificially adjust the reflector

depth to minimize residuals highlights a key limitation of the 1-D model: it cannot account for lateral variations or dipping structures. This limitation motivates the use of more complex 2-D modelling to better represent wave propagation in the subduction zone region and more accurately constrain the origin of the bonus arrivals.

With the goal of increasing model complexity, the next step was to use the Pykonal software to model this event. Although Pykonal is designed for 2-D or 3-D models, the 1-D velocity model was first interpolated over the 2-D area. This was done to ensure the development of the Pykonal procedure worked properly for a simpler model prior to introducing the 2-D velocity model.

The results of the 1-D Pykonal model were presented in section 4.1.2. In the computed ray paths for the direct arrival P and S waves (figures 21 and 22) it can be seen that the computed ray paths from the source differ spatially to get to each station and are slightly bent by the velocity contrast defined at 6 km depth. In the P and S travel times and residuals (tables 6 and 7) this model demonstrates slightly worse performance in computing accurate direct arrival travel times as compared to the 1-D ObsPy.TauPy model; however, the 1-D Pykonal direct travel times are still accurate within 0.4 – 11.5 s. It can be noted that compared to the 1-D ObsPy.TauPy model that the Pykonal computation produced residuals less systematically too small and rather has a distribution of positive and negative residuals depending on station location. At this point the use of Pykonal to model this event in 1-D was better understood and was next customized for use with the 2-D velocity model.

5 2-D Direct and Reflected Modelling

This section presents the results produced using the 2-dimensional seismic velocity structure to model ray travel paths and travel times for the March 2025, Orcas Island event.

5.1 2-D Pykonal Direct Arrival Modelling Results

In modelling the 2-D ray paths, the direct P and S wave arrivals were produced on the profile A (48.6°N) velocity model. The direct P wave arrivals are plotted for both a zoomed out (figure 23) and zoomed in (figure 24) longitude and depth range, for reference of what the velocity model looks like on a larger scale. The direct S wave ray paths using this model are plotted similarly (figure 25).

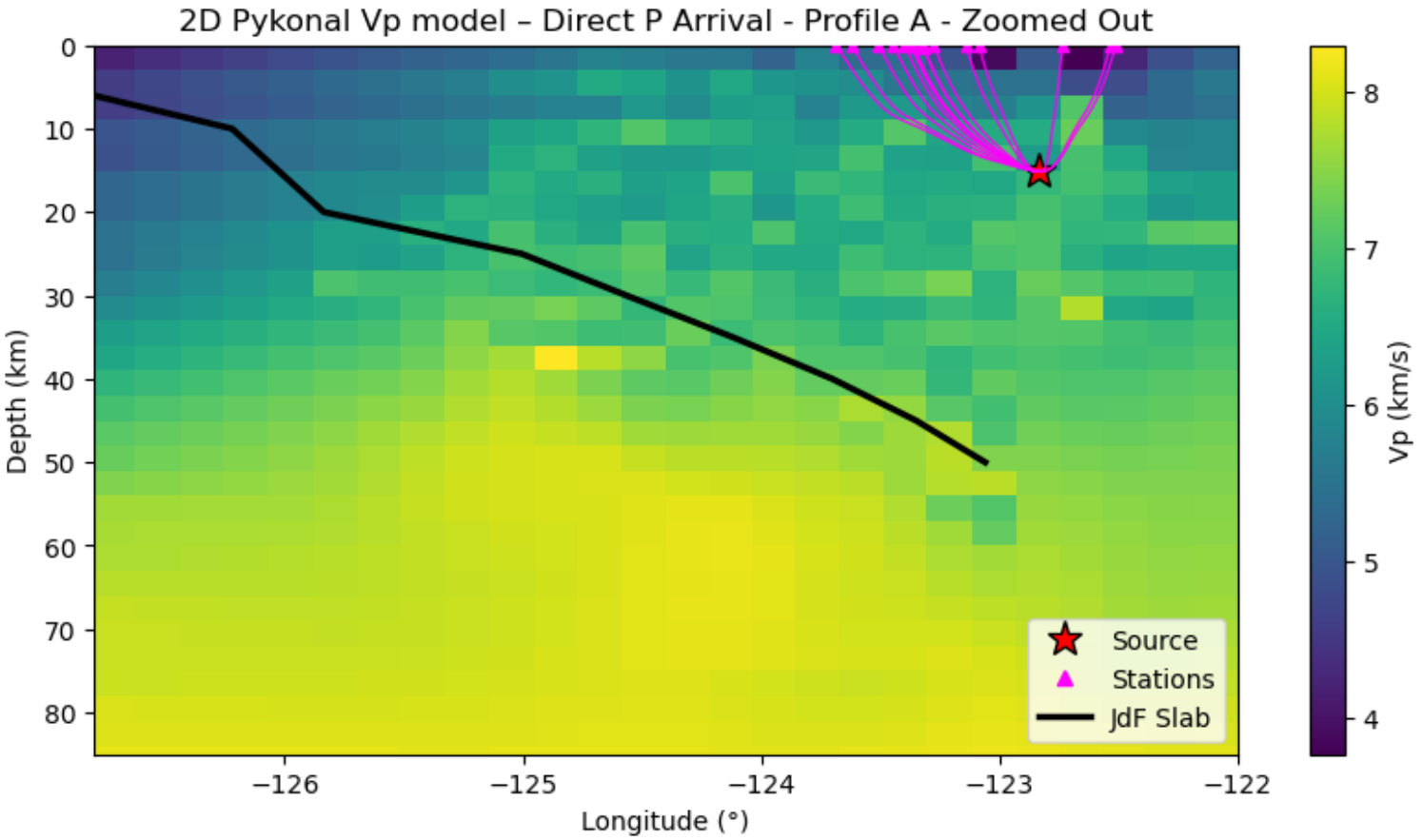


Figure 23 *– 2-D direct P wave arrival ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile A cross section plotted along with JdF slab depths (Sypus & Wang, 2024), plotted on a larger depth and longitude scale to visualize a greater region of the subsurface.*

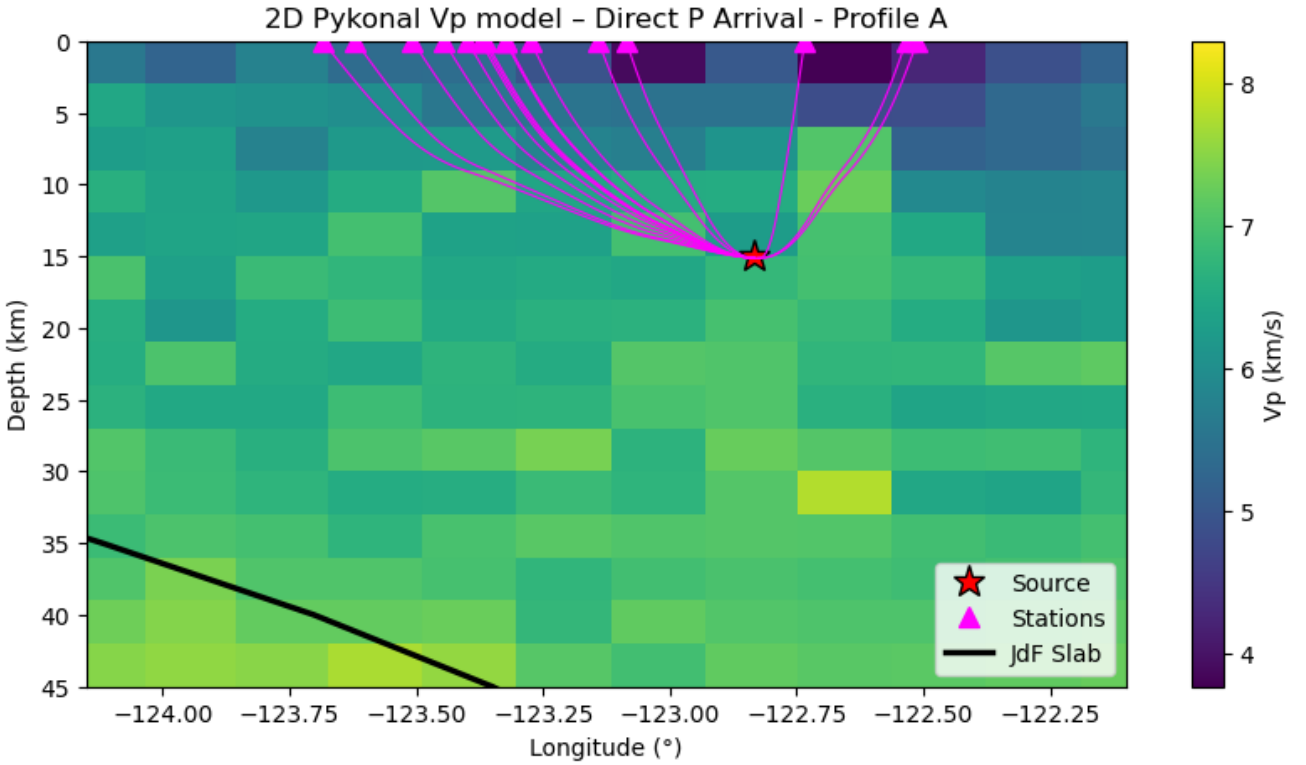


Figure 24 *– 2-D direct P wave arrival ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile A cross section plotted along with JdF slab depths (Sypus & Wang, 2024).*

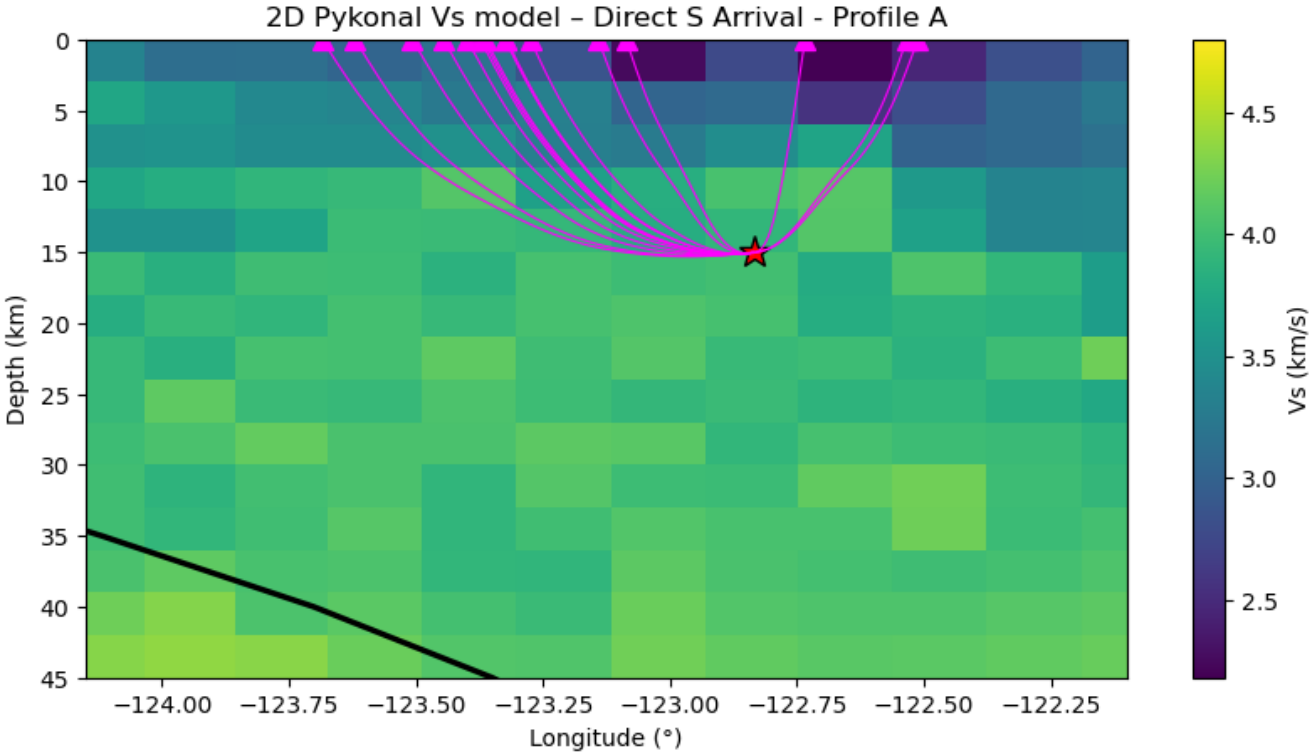


Figure 25 *– 2-D direct S wave arrival ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_S Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile A cross section plotted along with JdF slab depths (Sypus & Wang, 2024).*

The travel times were then computed for the direct P and S wave arrivals using the 2-D Pykonal model (table 8), as well as the residuals for these computed travel times (table 9).

Table 8: Calculated Travel Times from 2-D Pykonal Direct Arrival Modelling Routine

Station	**Modelled P Travel Times** (s)	**Modelled S Travel Times** (s)	**Actual P Travel Time** (s)	**Actual S Travel Time** (s)
R7D30	4.9353	8.5077	6.0	9.0
RA7D3	5.0749	8.8977	7.0	11.0
RF44C	6.1691	10.5555	7.0	12.0
R3CDE	7.0259	11.9603	8.0	13.0
RF926	7.0746	12.0401	8.0	14.0
RC45E	7.3920	12.5642	8.0	14.0
R382B	3.1265	5.4750	8.3	14.4
R09BB	4.4614	7.7090	11.6	19.0
RA825	4.8679	8.5436	12.0	21.0
RE0FF	10.3638	17.5303	12.0	21.0
RC118	8.4942	14.4091	10.0	16.0
R0EBF	6.6559	11.3537	8.0	13.0
RC2BF	7.1330	12.1359	8.0	13.0
RD862	9.6730	16.3788	10.0	18.0
R95F2	7.8658	13.3574	9.0	15.0
RF395	6.6267	11.3058	8.0	13.0

Table 9: Residuals of Seismic Phases Computed in 2-D Pykonal Direct Arrival Modelling Routine

Station	**P** (s)	**S** (s)
R7D30	-1.0640	-0.4922
RA7D3	-1.9250	-2.1022
RF44C	-0.8308	-1.4444
R3CDE	-0.9740	-1.0396
RF926	-0.9253	-1.9598
RC45E	-0.6079	-1.4357
R382B	-5.1734	-8.9249
R09BB	-7.1385	-11.2909
RA825	-7.1320	-12.4563
RE0FF	-1.6361	-3.4696
RC118	-1.5057	-1.5908
R0EBF	-1.3440	-1.6462
RC2BF	-0.8669	-0.8640
RD862	-0.3269	-1.6211
R95F2	-1.1341	-1.6425
RF395	-1.3732	-1.6941

For the 2-D Pykonal model of a reflection off the JdF slab, the model was run for Pp reflections (figure 26), as well as Ps reflections (figure 27). The Ps reflection was computed using the V_S model for the upgoing ray path but is shown overlain on the V_P velocity model.

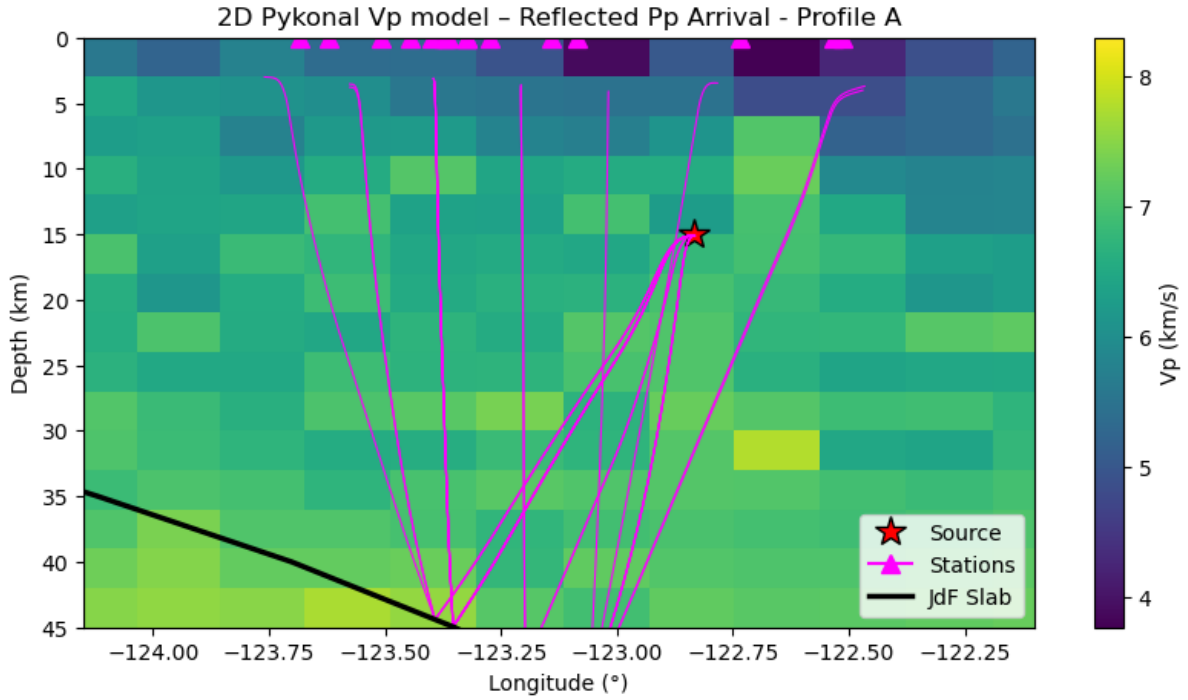


Figure 26 *– 2-D Pp reflection off JdF slab (profile A) ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile A cross section plotted along with JdF slab depths (Sypus & Wang, 2024).*

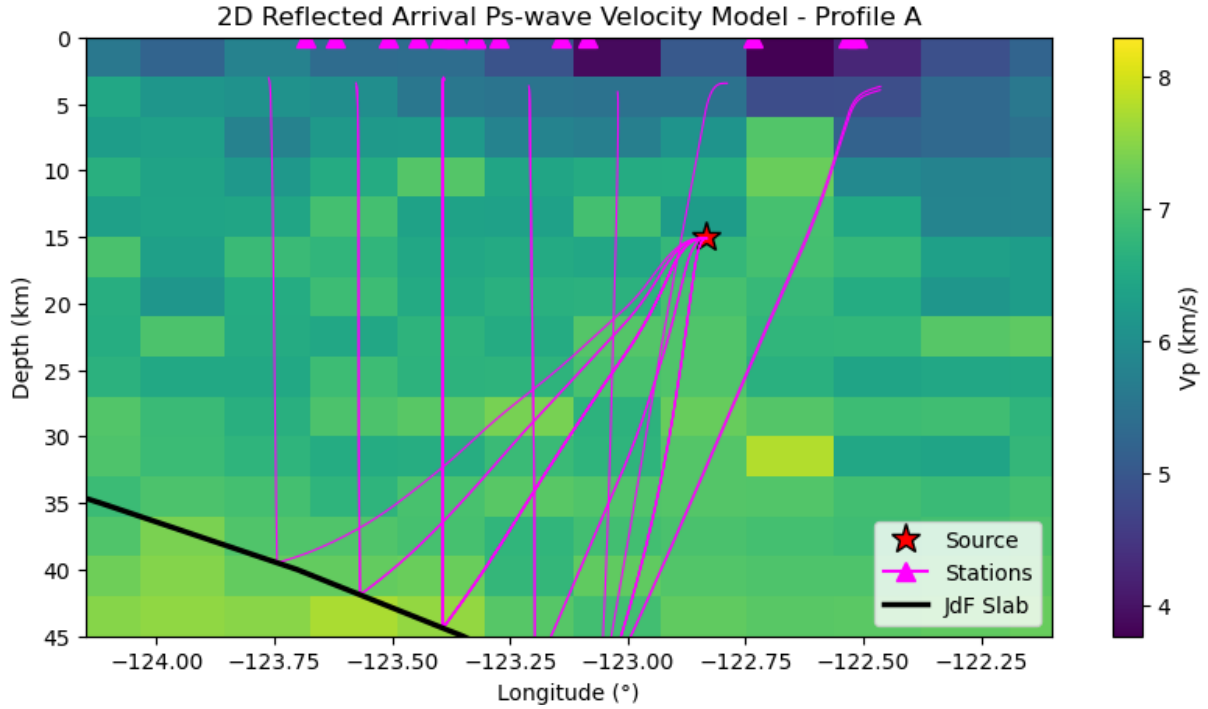


Figure 27 *– 2-D Ps reflection off JdF slab (profile A) ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile A cross section plotted along with JdF slab depths (Sypus & Wang, 2024).*

Next, the travel times (table 10) and residuals (table 11) are presented for the reflected ray paths, along with the actual travel time for the bonus arrival and S wave arrival for comparison.

Table 10: Calculated Travel Times from 2-D Pykonal Reflected Arrival Modelling Routine – Profile A

Station	**Modelled Pp Travel Times** (s)	**Modelled Ps Travel Times** (s)	**Actual Bonus Travel Time** (s)	**Actual S Travel Time** (s)
R7D30	13.9342	13.8546	-	9.0
RA7D3	16.6229	20.3883	-	11.0
RF44C	13.9482	13.8635	10.01	12.0
R3CDE	14.3188	14.2174	-	13.0
RF926	14.3073	14.1980	-	14.0
RC45E	14.2611	14.1424	-	14.0
R382B	14.9044	16.1555	-	14.4
R09BB	13.8310	14.0117	-	19.0
RA825	16.7955	20.7085	-	21.0
RE0FF	16.5898	17.8712	-	21.0
RC118	15.2357	15.7256	12.4	16.0
R0EBF	14.4952	14.5200	11.0	13.0
RC2BF	14.2843	14.1803	10.4	13.0
RD862	15.0939	15.4979	-	18.0
R95F2	14.4245	14.4184	11.6	15.0
RF395	14.5153	14.5548	11.03	13.0

Table 11: Residuals of Seismic Phases Computed in 2-D Pykonal Reflected Arrival Modelling Routine – Profile A

Station	**Pp** (s)	**Ps** (s)
R7D30	+46970.934	+46976.274
RA7D3	+46973.623	+46981.713
RF44C	+3.938	+9.273
R3CDE	+46971.319	+46976.198
RF926	+46971.307	+46976.179
RC45E	+46971.261	+46976.123
R382B	+46971.904	+46978.458
R09BB	+46970.831	+46976.522
RA825	+46973.796	+46982.018
RE0FF	+46973.590	+46979.032
RC118	+2.836	+7.818
R0EBF	+4.095	+9.101
RC2BF	+3.284	+8.161
RD862	+46972.094	+46976.990
R95F2	+2.825	+7.799
RF395	+3.485	+8.505

Following this, the JdF reflection model was repeated for the SW-NE profile B across the region. The modelling of a reflection off this profile of the JdF was done for both a Pp (figure 28) and Ps reflection (figure 29).

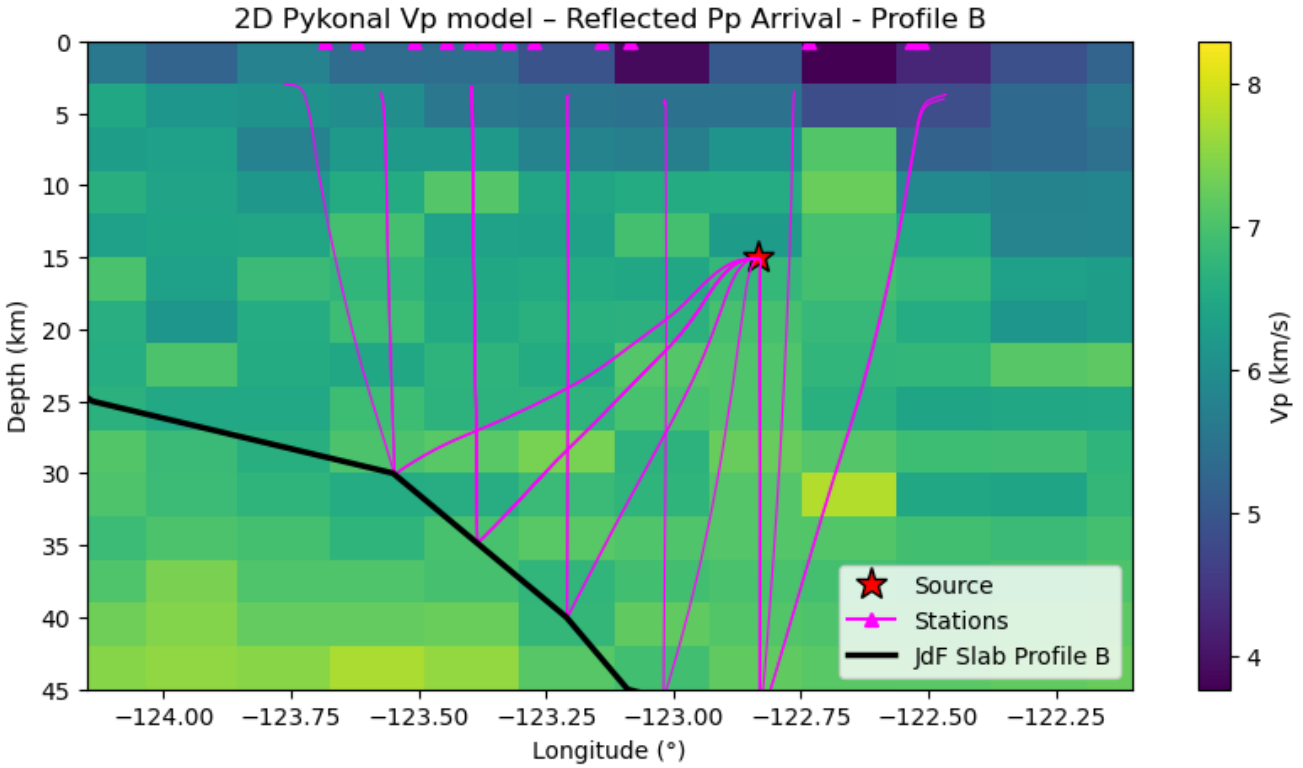


Figure 28 *– 2-D Pp reflection off JdF (profile B) slab ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile B cross section plotted along with JdF slab depths (Sypus & Wang, 2024).*

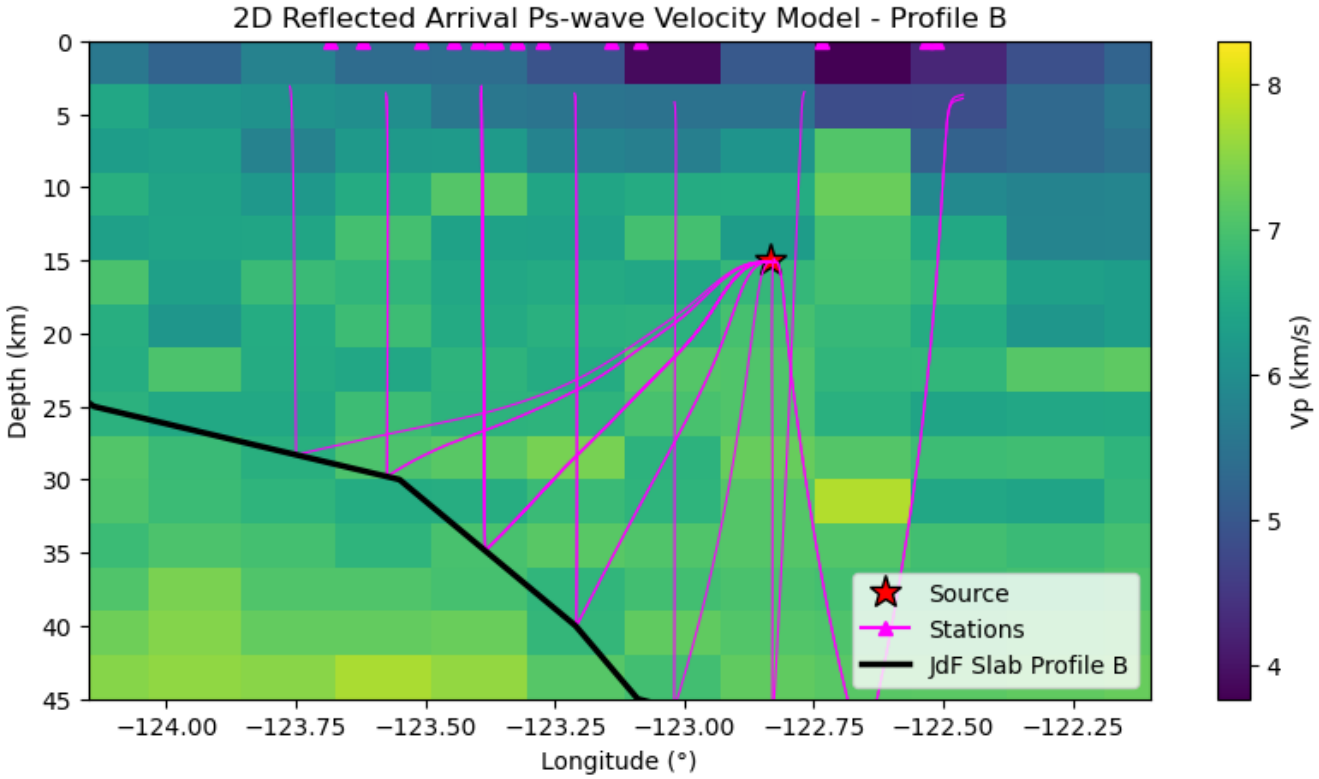


Figure 29 *– 2-D Ps reflection off JdF (profile B) slab ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile B cross section plotted along with JdF slab depths (Sypus & Wang, 2024).*

Next, the travel times and residuals are presented for the profile B reflected ray paths in table 12 and 13 respectively.

Table 12: Calculated Travel Times from 2-D Pykonal Reflected Arrival Modelling Routine – Profile B

Station	**Modelled Pp Travel Times** (s)	**Modelled Ps Travel Times** (s)	**Actual Bonus Travel Time** (s)	**Actual S Travel Time** (s)
R7D30	12.2822	16.8016	-	9.0
RA7D3	13.4972	20.1347	-	11.0
RF44C	12.2962	16.8105	10.01	12.0
R3CDE	12.3578	16.1612	-	13.0
RF926	12.3466	16.1418	-	14.0
RC45E	12.3005	16.0861	-	14.0
R382B	12.4516	18.4980	-	14.4
R09BB	12.3610	17.4831	-	19.0
RA825	13.6623	20.4185	-	21.0
RE0FF	14.7213	19.0031	-	21.0
RC118	13.3071	16.8194	12.4	16.0
R0EBF	12.5327	16.4639	11.0	13.0
RC2BF	12.3237	16.1241	10.4	13.0
RD862	13.1680	16.5922	-	18.0
R95F2	12.4622	16.3624	11.6	15.0
RF395	12.5527	16.4987	11.03	13.0

Table 13: Residuals of Seismic Phases Computed in 2-D Pykonal Reflected Arrival Modelling Routine – Profile B

Station	**Pp** (s)	**Ps** (s)
R7D30	+46969.282	+46973.802
RA7D3	+46970.497	+46977.135
RF44C	+2.286	+6.801
R3CDE	+46969.358	+46973.161
RF926	+46969.347	+46973.142
RC45E	+46969.301	+46973.086
R382B	+46969.452	+46975.498
R09BB	+46969.361	+46974.483
RA825	+46970.662	+46977.419
RE0FF	+46971.721	+46976.003
RC118	+0.907	+4.419
R0EBF	+2.133	+6.064
RC2BF	+1.324	+5.124
RD862	+46970.168	+46973.592
R95F2	+0.862	+4.762
RF395	+1.523	+5.469

5.3 Discussion of 2-D models

The 2-D velocity model along profile A was integrated into the Pykonal software and used to compute direct P and S ray paths and travel times from this event. In the 'zoomed out' profile (figure 23) of the velocity model, for a larger depth and longitude range, the velocity contrast of the dipping subducting JdF slab can be clearly seen. The computed P and S wave ray paths in the 2-D model are also presented at a closer scale (figures 24 and 25). This model shows a similar underestimation of travel times (table 8) for the direct arrivals, and a similar quality of residuals as produced by the previous models (table 9). To go forth to model the reflected arrivals off the JdF slab this 2-D velocity segment and Pykonal model were used.

The JdF reflection modelling using the 2-D Pykonal model was done by computing a 2 segment ray path as Pykonal does not have a reflection phase module built in. As ray tracing in Pykonal functions by computing the gradient of the travel time field in each defined cell, the upgoing ray path being traced from the reflection point on the JdF plate to the receiver (at 0 km depth), the gradient cannot be computed as there is no defined cell above 0 km depth; therefore the ray paths don't trace all the way to the stations in the plots. Although this affects the visual ray path plotting it does not affect the computed travel times.

In the ray path plotting for Pp and Ps reflections off the JdF plate (figures 26 and 27 respectively), as the slab reflection point was determined by choosing the point of reflection which minimized the travel time the ray paths to each of the 16 stations overlap and reflect at the same points leading to the appearance of fewer traced rays. In analyzing the computed travel times from this reflection modelling (table 10) it was found that this phase model for both the Pp and Ps reflections produced arrival times that are later than both the actual bonus arrival and the true S wave arrival. Therefore, it is deduced that for the bonus arrivals to arrive before the S wave arrivals for up-dip stations, they must be due to a reflection off a shallower boundary than the top of the JdF plate. Further confirmation that the bonus arrivals are not consistent with reflections off the JdF slab is that the computed residuals for each reflected phase are large in magnitude (2.8 to 4.1 s for the Pp reflection) and all positive (arriving too late) (table 11).

A similar reflection model was tested for the SW-NE profile B cross section of the JdF plate for both a Pp and Ps reflection (figures 28 and 29 respectively). This was done as a *pseudo 3-D model* to identify if a change in profile direction had the effect of improving bonus arrival residuals for along-profile stations.

The results of the profile B reflection modelling are similar to those of the profile A reflection model in that the travel times are also too long to reproduce the observed bonus arrivals (table 12). However, the residuals when computed using profile B decrease overall for stations located along profile B (table 13), which include all of the stations analyzed that received the bonus arrival (since profile B was chosen to cross the dense area of stations on southern Vancouver Island). This result demonstrates the need for a 3-D model in this region to further reduce residuals of calculated seismic phases overall.

At this point, an important conclusion can be made: the bonus arrivals must be due to a reflection off a shallower boundary than the subducting JdF plate. Another useful modelling result is that the Pvmp reflection residuals using ObsPy.TauPy were minimized for a reflector depth of 35 km (section 5.1.1). The next goal is to compute the reflected phases off a shallower boundary that occurs at ~35 km depth beneath the region. Such a boundary has been identified in seismic reflection data – the E layer, which overlies the JdF slab with a similar

dip (section 2.1). To minimize the overall residuals of the bonus phase arrivals, the next model run computed the reflections off the E-Layer.

5.4 The E-Layer

The shallower boundary which the bonus arrivals are hypothesized to reflect off is known as the E-layer (or E reflector). As mentioned in section 2.1, the E-layer is a conductive layer of low density and low P and S-wave velocity (Nicholson et al., 2005), lying above and parallel to the JdF slab with varying thickness across the region (3-10 km), reaching at least 45 km depth east of Vancouver Island (Ramachandran, 2001). The location and approximate thickness of the E-Layer below southern Vancouver Island is constrained by seismic reflection profiles (figure 2, section 2.1) (Rogers, 1983)

As the JdF slab is too deep to produce reflected bonus arrival times that arrive before the S wave for up-dip stations, the reflections must be occurring off a shallower boundary, which could be the known highly seismically reflective E-layer. The next goal is to model ray paths that reflect off a boundary at the approximate location of the E-layer with the goal of minimizing the bonus arrival residuals.

6 E-Layer Modelling

6.1 2-D E-layer Reflection Results

Here, I present the 2 dimensional modelled ray travel paths and modelled ray travel times for the reflection off the approximate location of the E-layer. This 2-D modelling was done in Pykonal as described in section 3.5.4 in the same method as performed for the reflection off the JdF slab previously. These ray paths were computed for a Pp (figure 30) and Ps (figure 31) reflection off the approximate E-layer location using the profile A cross section.

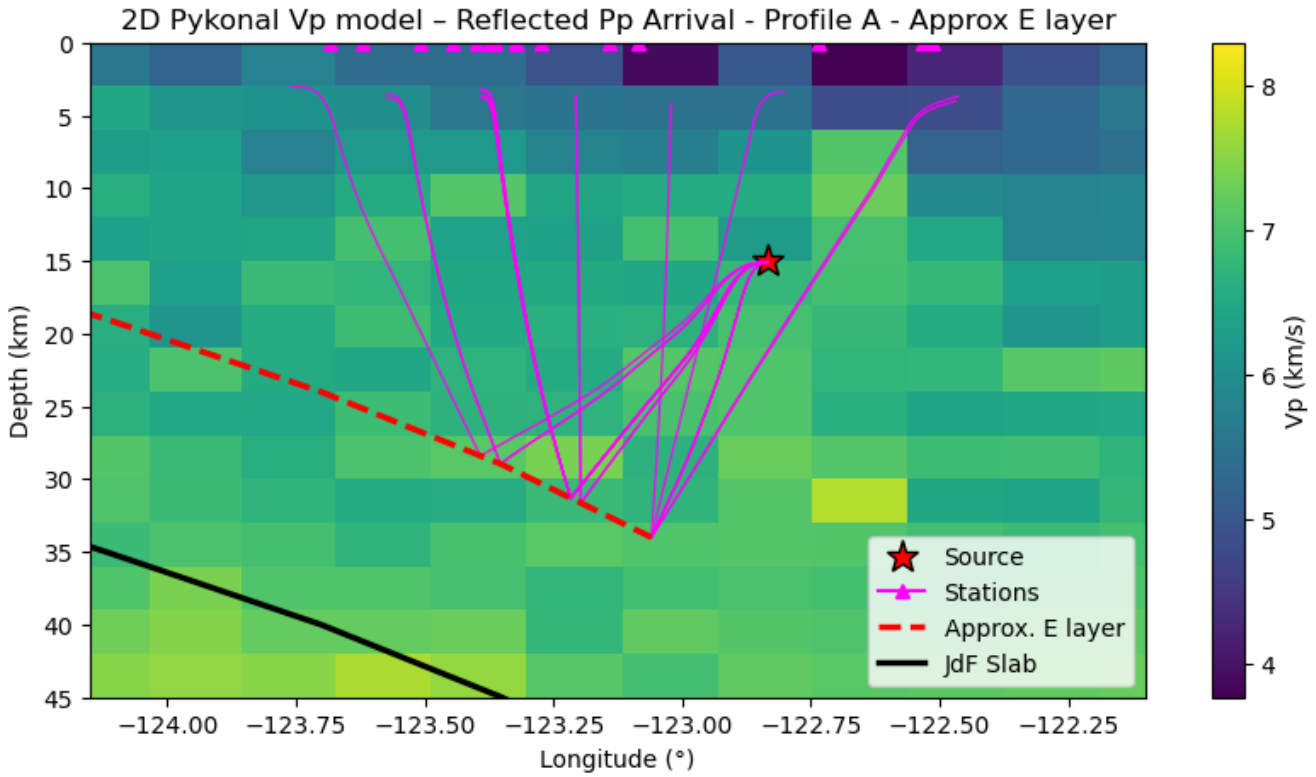


Figure 30 *– 2-D Pp reflection off approximate E-layer (profile A) ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile A JdF slab depths (Sypus & Wang, 2024), and approximate profile A E-layer (red dashed line) are shown.*

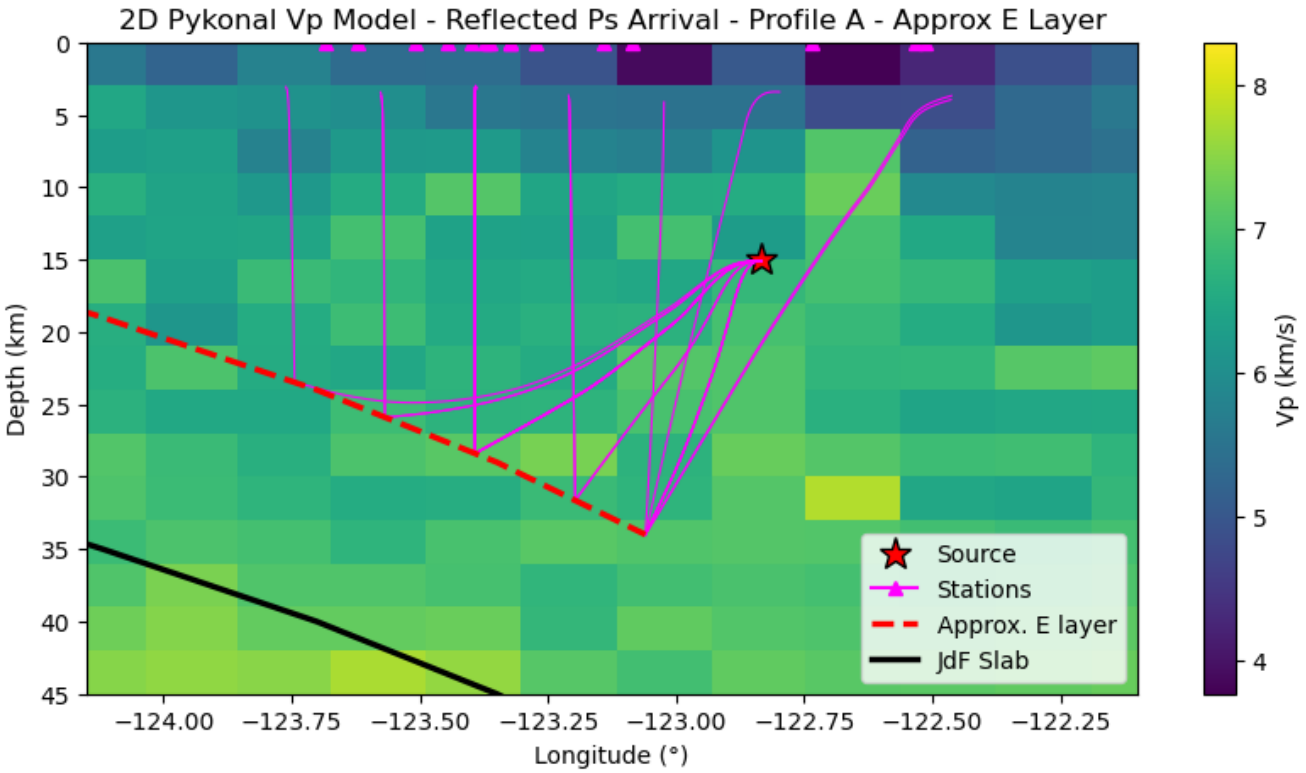


Figure 31 *– 2-D Ps reflection off approximate E-layer (profile A) ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile A JdF slab depths (Sypus & Wang, 2024), and approximate profile A E-layer (red dashed line) are shown.*

Next, in table 14, I present the modelled ray travel times for this modelled reflection off the E-layer along profile A.

Table 14: Calculated Travel Times from 2-D Pykonal E-Layer Reflected Arrival Modelling Routine – Profile A

Station	**Modelled Pp Travel Times** (s)	**Modelled Ps Travel Times** (s)	**Actual Bonus Travel Time** (s)
R7D30	10.2176	13.8546	-
RA7D3	13.3633	20.3883	-
RF44C	10.2317	13.8635	10.01
R3CDE	10.9482	14.2174	-
RF926	10.9347	14.1980	-
RC45E	10.8874	14.1424	-
R382B	11.1027	16.1555	-
R09BB	9.8642	14.0117	-
RA825	13.5452	20.7085	-
RE0FF	13.8766	17.8712	-
RC118	12.2749	15.7256	12.4
R0EBF	11.1394	14.5200	11.0
RC2BF	10.9106	14.1803	10.4
RD862	12.1284	15.4979	-
R95F2	11.0666	14.4184	11.6
RF395	11.1604	14.5548	11.03

For these calculated travel times, the corresponding residuals were calculated and are presented in table 15.

Table 15: Residuals of Seismic Phases Computed in 2-D Pykonal E-layer Reflected Arrival Modelling Routine – Profile A

Station	**Pp** (s)	**Ps** (s)
R7D30	+46967.218	+46970.855
RA7D3	+46970.363	+46977.388
RF44C	+0.222	+3.854
R3CDE	+46967.948	+46971.217
RF926	+46967.935	+46971.198
RC45E	+46967.887	+46971.142
R382B	+46968.103	+46973.156
R09BB	+46966.864	+46971.012
RA825	+46970.545	+46977.709
RE0FF	+46970.877	+46974.871
RC118	-0.125	+3.326
R0EBF	+0.739	+4.120
RC2BF	-0.089	+3.180
RD862	+46969.128	+46972.498
R95F2	-0.533	+2.818
RF395	+0.130	+3.525

Next, I present the results from the E-layer reflection using the profile B cross section, as was done previously for the JdF reflection (section 4.4.2). The profile B E-layer reflected ray paths were computed for a Pp (figure 32) and Ps (figure 33) reflection.

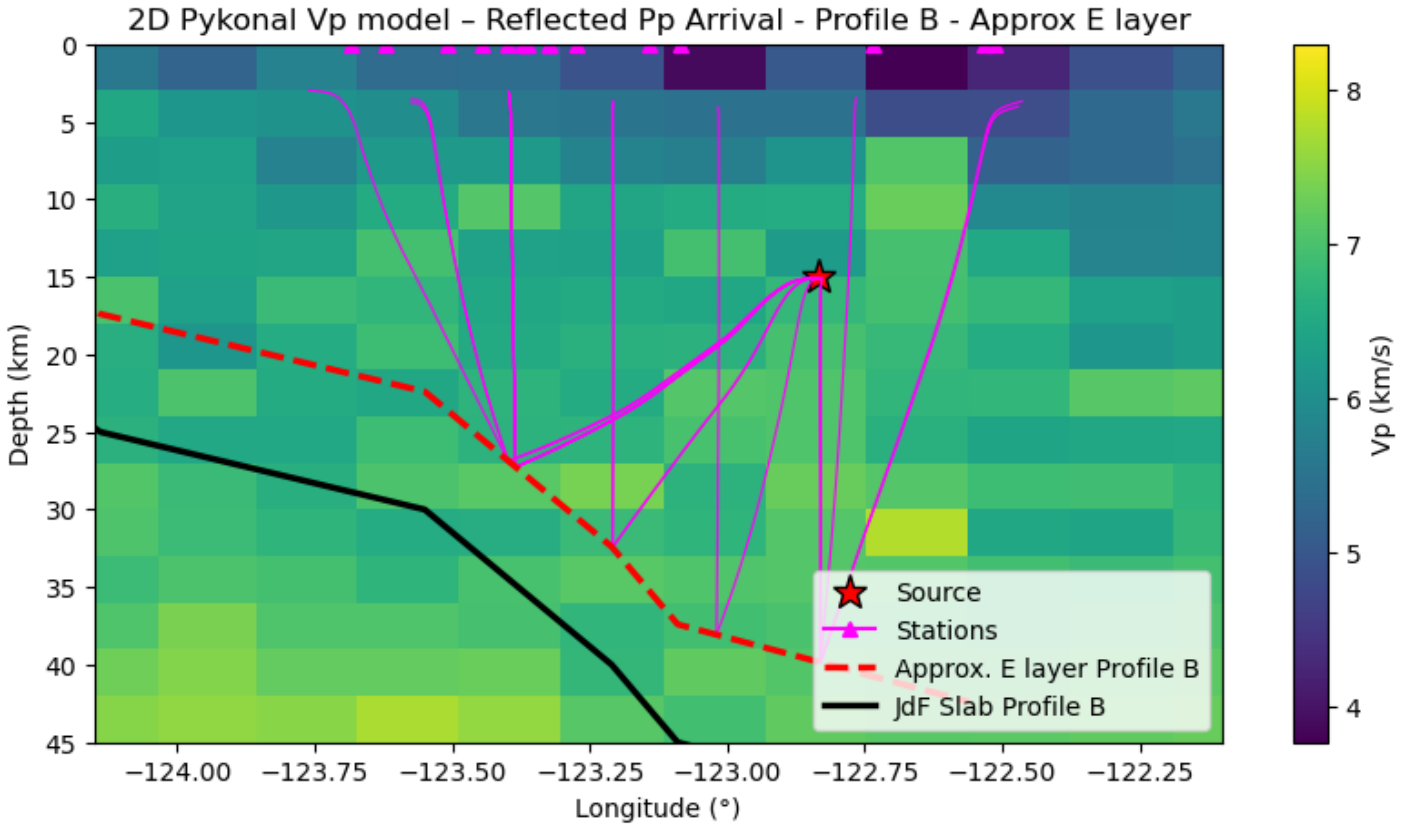


Figure 32 *– 2-D Pp reflection off approximate E-layer (profile B) ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile B JdF slab depths (Sypus & Wang, 2024), and approximate profile B E-layer (red dashed line) are shown.*

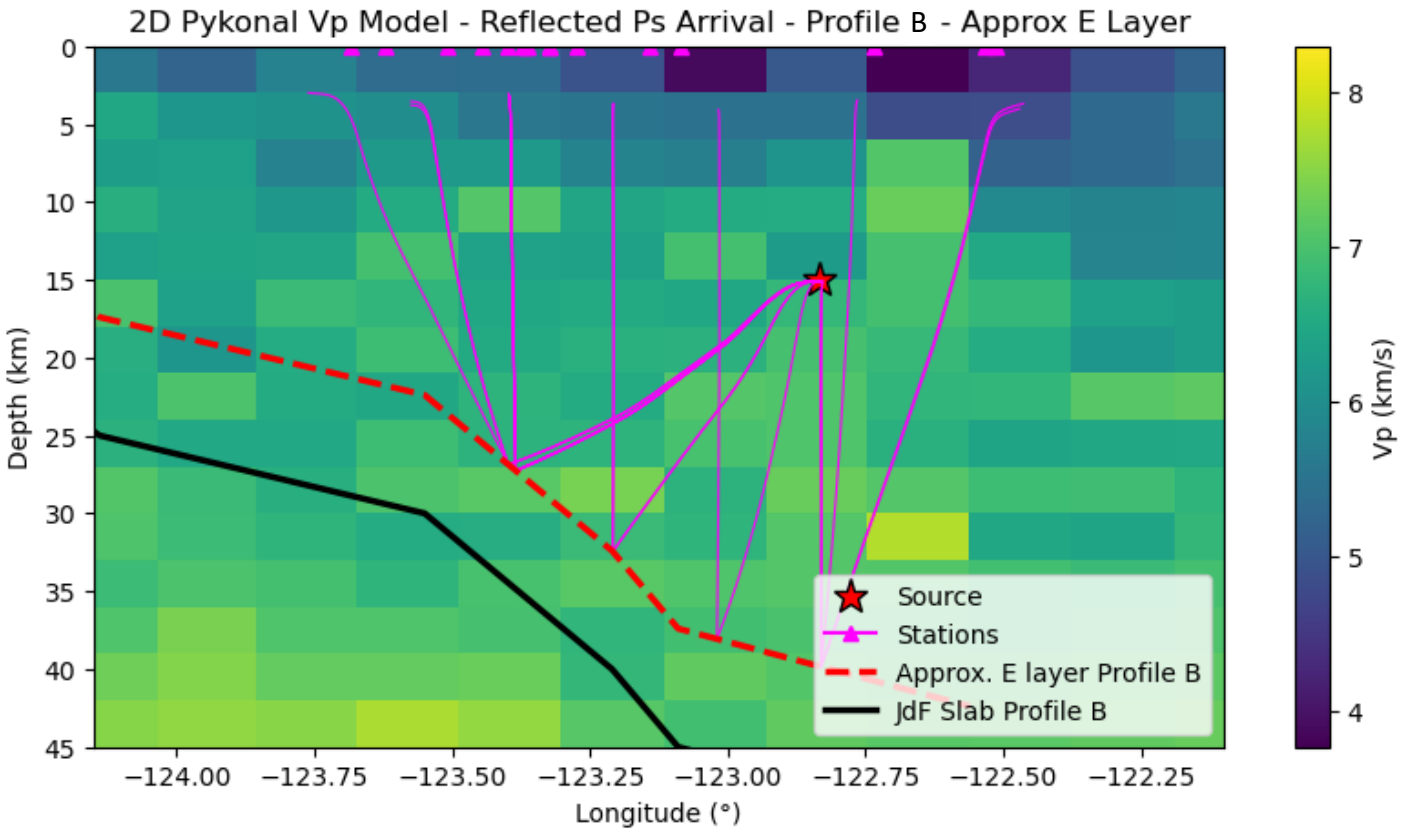


Figure 33 *– 2-D Ps reflection off approximate E-layer (profile B) ray path modelling results using Pykonal (White et al., 2020) and the 2-D V_P Savard velocity model (Savard, 2018), from the March 3rd, 2025 earthquake hypocenter to each of the 16 seismic stations analyzed. Profile B JdF slab depths (Sypus & Wang, 2024), and approximate profile B E-layer (red dashed line) are shown.*

Next, the travel times (table 16) and residuals (table 17) are presented for the profile B E-layer reflected ray paths.

Table 16: Calculated Travel Times from 2-D Pykonal E-Layer Reflected Arrival Modelling Routine – Profile B

Station	**Modelled Pp Travel Times** (s)	**Modelled Ps Travel Times** (s)	**Actual Bonus Travel Time** (s)
R7D30	10.5022	14.1896	-
RA7D3	11.6029	17.5777	-
RF44C	10.5163	14.1985	10.01
R3CDE	10.8576	13.8152	-
RF926	10.8463	13.7957	-
RC45E	10.8002	13.7400	-
R382B	10.3983	15.6492	-
R09BB	10.3998	14.7751	-
RA825	11.7711	17.8639	-
RE0FF	13.6664	17.2146	-
RC118	12.0514	14.8600	12.4
R0EBF	11.0326	14.1183	11.0
RC2BF	10.8234	13.7780	10.4
RD862	11.9060	14.6325	-
R95F2	10.9621	14.0166	11.6
RF395	11.0527	14.1531	11.03

Table 17: Residuals of Seismic Phases Computed in 2-D Pykonal E-Layer Reflected Arrival Modelling Routine – Profile B

Station	**Pp** (s)	**Ps** (s)
R7D30	+46967.50	+46971.190
RA7D3	+46968.603	+46974.578
RF44C	+0.506	+4.189
R3CDE	+46967.858	+46970.815
RF926	+46967.846	+46970.796
RC45E	+46967.800	+46970.740
R382B	+46967.398	+46972.649
R09BB	+46967.400	+46971.775
RA825	+46968.771	+46974.86
RE0FF	+46970.666	+46974.215
RC118	-0.349	+2.460
R0EBF	+0.633	+3.718
RC2BF	-0.177	+2.778
RD862	+46968.906	+46971.633
R95F2	-0.638	+2.417
RF395	+0.023	+3.123

6.2 Discussion of 2-D Pykonal E-Layer Reflection Modelling

As it was found previously that seismic rays reflecting off the JdF slab arrived too late to be the origin of the observed bonus arrival, the next step taken was to model this reflection off the top of the E-layer (shallower and highly seismically reflective boundary) as a proposed source of the bonus arrivals. First, to produce such a model, the profile A JdF slab data points (as defined at 5 km depth spacings across a range of longitudes and fit with a linear curve) were initially altered such that each depth contour was moved shallower to the estimated depth of the E-layer and the travel times and residuals were calculated. Following the initial bulk shift vertically in the original JdF slab data, the depths of the newly defined E-layer data points were each adjusted slightly, at the same interval to maintain the same spacing between data points, until the calculated bonus arrival residuals were minimized. This reflection was computed for both a Pp (figure 30) and Ps (figure 31) reflection off the E-layer; similar to earlier results, the calculated Ps reflection travel times do not match well (table 14), further demonstrating that the bonus arrival is likely due to a Pp reflection of seismic energy.

As the E-layer depths were finely adjusted to produce minimum residuals, the Pp reflection residuals proved to decrease significantly in this modelling routine (table 15). In analyzing the calculated residuals for this model, it is found that the Pp reflection residuals are all within the range of ± 0.7 s which demonstrates a significant improvement in bonus arrival travel time calculations compared to the JdF reflections.

The final step taken in the modelling process for this project was to model reflections using the profile B cross section of the E-layer, similar to the process followed for the JdF reflections in section 4.2.2. This was done once again as a pseudo 3-dimensional modelling process to determine if residuals improved for stations along profile B. The profile B E-layer reflections were modelled for both a Pp (figure 32) and Ps (figure 33) reflection.

The travel times (table 16) and residuals (table 17) for this model demonstrate once again that the bonus arrival is consistent with a Pp reflection rather than a Ps reflection. In analyzing the residuals from the profile B reflections, we find that each residual computed for stations which observed the bonus arrival succeeded in decreasing, in comparison to the modelling done with profile A. The Pp residuals for the profile B model each lie within ± 0.6 s, which again shows that calculated residuals are decreased, and therefore the models would be more accurate using a 3-D model.

Overall, as the residuals were minimized for the 1-D ObsPy.TauPy Pvmp reflection for a reflector at 35 km depth (similar to that of the E-layer below this region), and given the further decrease of the residuals in using Pykonal to model both the profile A and profile B Pp reflection off the E-layer, it can be concluded that within the bounds of each of these models the source of the bonus arrivals is consistent with a reflection of P wave energy off the E-Layer.

7 Residual Analysis and Model Comparisons

7.1 Summarizing Some Simple Statistics of Each Model

While the sample of stations analyzed in this study is too limited to perform an in-depth statistical analysis, some simple statistics on the model residuals aid in model assessment and comparison. Statistical computations performed include: the mean residual, mean absolute residual, and the standard deviation. Note that, for the bonus arrival statistical analysis, stations which did not observe the bonus arrival between the P and S wave arrival are not included in the statistical computations. The statistical results for the direct arrival models are presented in table 18, while those of the bonus arrival modelling are presented in table 19.

Table 18: Statistical Analysis Results for Direct Arrival Residuals

	Mean Residual (s)	**Mean Absolute Residual** (s)	**Standard Deviation** (s)
1-D ObsPy.TauPy - P	-1.1041	1.1041	0.2821
1-D ObsPy.TauPy - S	-1.4375	1.4375	0.6114
1-D Pykonal - P	0.3813	2.7412	3.216
1-D Pykonal - S	1.1346	4.9755	5.5568
2-D Pykonal - P	-2.1224	2.1224	2.1650
2-D Pykonal - S	-3.3546	3.3546	3.7265

Table 19: Statistical Analysis Results for Bonus Arrival Residuals

	Mean Residual (s)	**Mean Absolute Residual** (s)	**Standard Deviation** (s)
1-D ObsPy.TauPy - Pvmp	-1.5276	1.5276	0.3325
1-D ObsPy.TauPy - Svms	5.4546	5.4546	0.3816
1-D ObsPy.TauPy - Deeper Reflector - Pvmp	0.4374	0.4745	0.3642
1-D ObsPy.TauPy - Deeper Reflector - Svms	7.3419	7.3419	0.3272
2-D Pykonal – JdF Pp reflection – Profile A	3.4105	3.4105	0.4903
2-D Pykonal – JdF Ps reflection – Profile A	8.4428	8.4428	0.5787
2-D Pykonal – JdF Pp reflection – Profile B	1.5058	1.5058	0.5489
2-D Pykonal – JdF Ps reflection – Profile B	5.4399	5.4399	0.8001
2-D Pykonal – E-layer Pp reflection – Profile A	0.0573	0.3065	0.3876
2-D Pykonal – E-layer Ps reflection – Profile A	3.4704	3.4704	0.4285
2-D Pykonal – E-layer Pp reflection – Profile B	-0.0002	0.3874	0.4503
2-D Pykonal – E-layer Ps reflection – Profile B	3.1141	3.1141	0.6515

7.2 Visualizing Residuals vs Azimuth around the March 3rd Epicenter

Finally, to visualize how the direct and bonus arrival residuals computed in each model change relative to azimuth around the March 3rd event's epicenter, plots of residual vs azimuth were made for each model run for the direct (figure 34) and bonus (figure 35) arrival residuals separately.

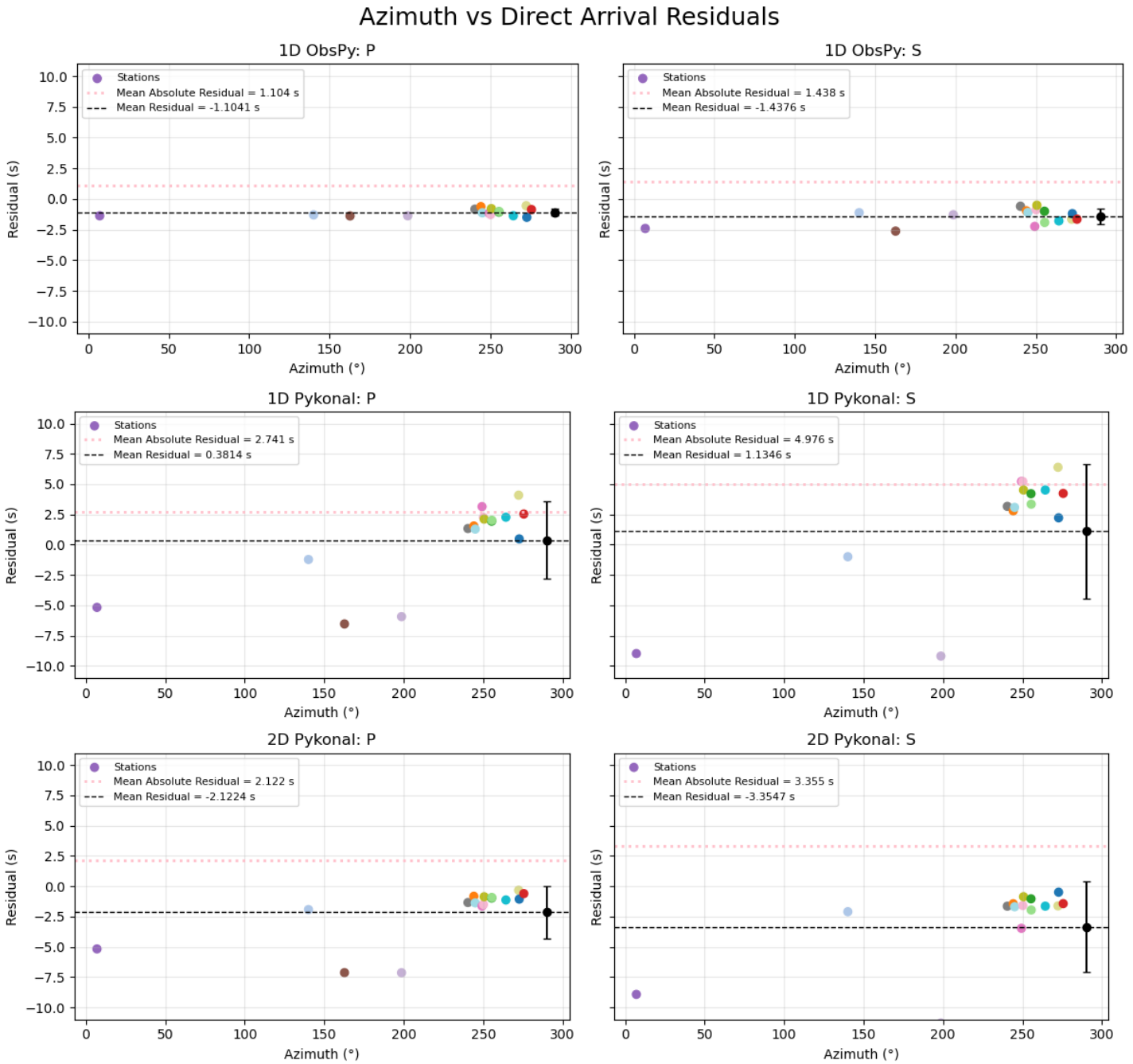


Figure 34 *– Residual vs azimuth for each of the direct arrival models. Mean residual (black dashed line), mean absolute residual (pink dotted line), and standard deviation (error bar) are overlain on each plot. Figure created using Matplotlib, 2026 (Hunter, 2007).*

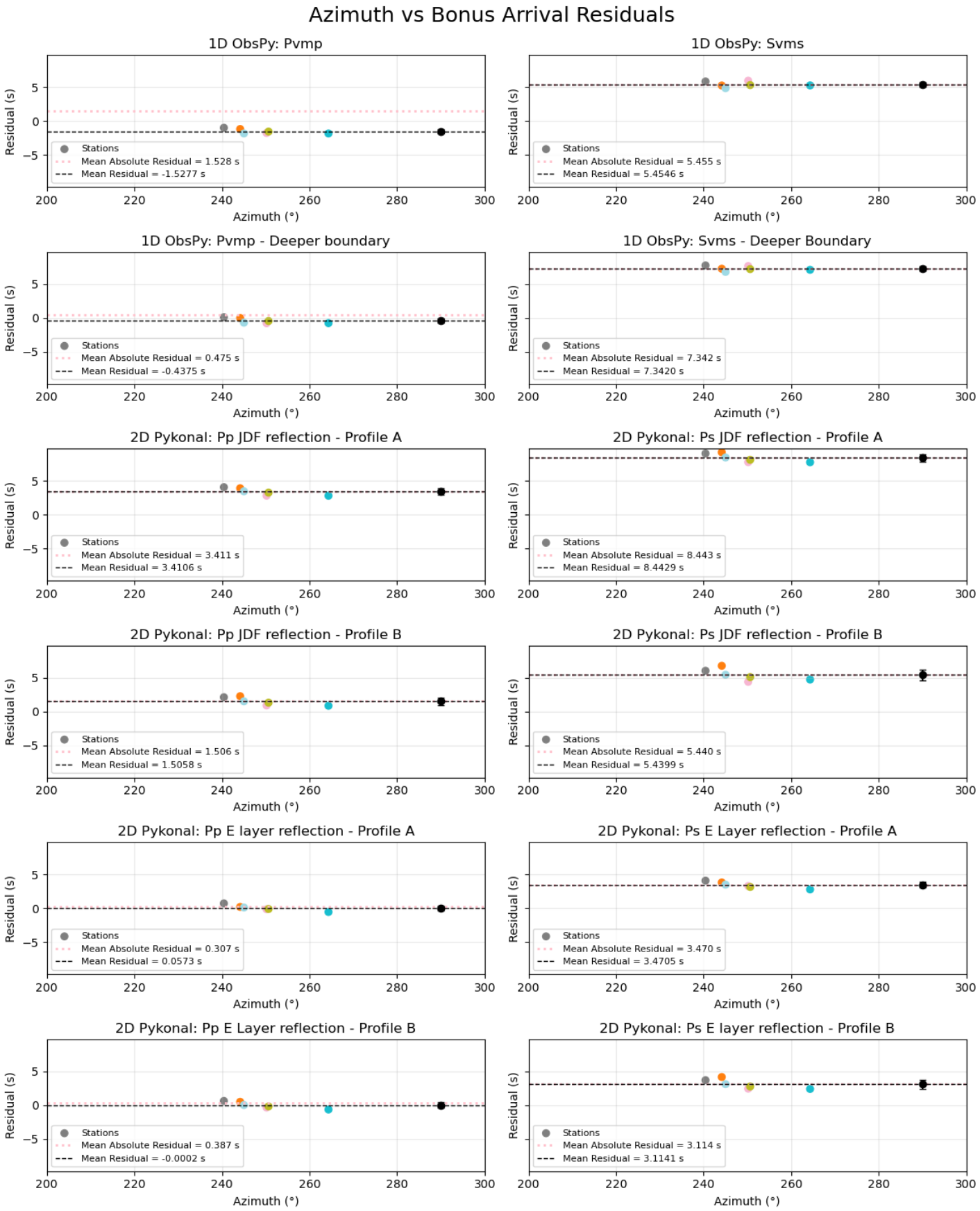


Figure 35 – *Residual vs azimuth for each of the bonus arrival models, plotted on a limited azimuthal range as these stations lie only W to N-W of the event. Mean residual (black dashed line), mean absolute residual (pink dotted line), and standard deviation (error bar) are overlain on each plot. Figure created using Matplotlib, 2026 (Hunter, 2007).*

7.3 Discussion of Model Performance

To further identify the strengths and limitations of each modelling routine used to produce calculated ray travel times and ray paths in this study, we can analyze the simple statistical calculations made on the direct (table 18) and bonus arrival (table 18) residuals. Only 16 stations were used in the computation for each model, with a large majority of those stations lying on the Saanich peninsula (figure 15); therefore, the sample of stations analyzed is too limited (both in quantity and spatial distribution) to perform an in-depth statistical analysis on the calculated travel times. However, calculating the mean residual, mean absolute residual, and standard deviation of residuals in each model aids in identifying patterns, bias, and a glance at the functionality of the model overall.

First to discuss the accuracy of modelled direct arrival travel times (table 18), it can be identified that each of the models performed relatively well, with the largest mean absolute residual being ~5 s for the 1-D Pykonal direct S wave arrivals. As for the 1-D ObsPy.TauPy model, the residuals were surprisingly low given the use of the very simple velocity model. However, as was seen in the direct arrival ray paths traced in each model, these rays propagate from the source (at ~15 km depth) upwards to the stations through a relatively spatially homogenous region in velocity even when using the more complex 2-D velocity model; therefore, it is reasonable for the 1-D model to perform quite well. In the mean and mean absolute residuals for the 1-D direct arrival, it can be observed that the magnitudes of these values are identical. This demonstrates that the model systematically calculated travel times too small compared to the actual travel time at every station. Furthermore, the standard deviations for these models are quite low, demonstrating that this model performed equally well for each station, regardless of location. To further demonstrate this, in the residual vs azimuth plots for the 1-D direct arrivals (row 1, figure 34) the residuals of the stations do not demonstrate a dependence on azimuth, with each station's result lying close to the mean residual. Regarding the functionality of the Pykonal software in computing the direct arrival residuals, it can be seen that these models performed adequately well in producing small residuals but show a larger standard deviation overall. In the direct arrival residuals vs azimuth (rows 2 and 3, figure 34) it can be seen that a small dependence on azimuth is introduced into the residuals – where in each Pykonal direct arrival model the stations lying W to NW of the event are positive, whereas the only negative residuals are from stations in the 0-200° azimuth range (corresponding to stations furthest from the modelling profiles A and B).

Overall, it was found that the 1-D ObsPy.TauPy model performed the best in producing smallest residuals for the direct arrival travel times for this event; however, limitations in the dimensionality of the model motivate the need for a 2-D model to be used going forth, to be able to compute rays interacting with the dipping boundaries in the subsurface in this region.

Regarding the bonus arrival modelling a similar analysis is made. A larger number of reflected phases were computed, for the 2 rounds of 1-D reflection modelling, JdF reflection, and E-layer reflection each computed for a P and S wave reflection, and the 2-D models for profile A and B. In general, for each modelling routine run for an S reflection (Svms or Ps) the residual is noticeably larger than its modelled P phase counterpart (Pvmp or Pp) (table 19). This further confirms that the bonus arrival is most likely due to a reflection of P wave energy, and that the S wave reflections arrive too late and are thus ruled out.

For the 1-D Pvmp models, travel times were systematically too low for the reflection off the Moho, with a similar distribution of each station around the mean (row 1, figure 35). However, the key takeaway from this procedure was that in moving the reflector deeper (35 km depth) the residuals were diminished (~ -1.5 to 0.4 s). This provided the key motivation for the 2-D model, as the 1-D model accounting only for a velocity contrast at the Moho simply did not describe the observed bonus arrivals well enough, in terms of the depth and planar nature of this reflector.

Thus, going forward with the 2-D model, as it was identified that the JdF reflections produced travel times greater than the S wave travel time, the shallower E-layer reflections are meaningful in residual analysis. For the E-layer Pp reflections for profile A, initially it can be seen that the mean absolute residual and mean residual are strikingly lower than for any of the other models along this profile. For the profile B Pp E-layer reflections the residuals are similarly low and improve for stations along the profile. Looking to the distribution of residual vs azimuth for this model (rows 5 and 6, figure 35), a slight trend in azimuth vs residual can be seen across each plot, with an overall small standard deviation for each of the models due to the close spatial proximity of the stations which received the bonus arrival.

Overall, these modelling results demonstrate the need to use, at minimum, a 2-D velocity model for seismic research and analysis in southwest British Columbia, present the likely source of the observed bonus arrivals as a Pp reflection off the E-layer, and demonstrate a potential improvement in seismic ray tracing routines with the use of a quasi-3-D model.

8 Uncertainties and Future Work

8.1 Uncertainties in the Study

There are known sources of systematic error as well as random uncertainties within the computations performed in this research. First, error is introduced into the modelling scheme in the definition of the geographical location of the stations as well as of the earthquake hypocenter. For the stations, a known approximate ±500 m random jitter is applied to the public source documentation of each location, as these devices are owned privately. Thus, each definition of station location for the modelled ray paths has an associated systematic spatial error. Also, there is uncertainty in the true location of the earthquake hypocenter (latitude, longitude, and depth).

Furthermore, there is uncertainty in the modelling software as well as in the velocity models. While the modelling routines function by integrating the ray path analytically or solving the Eikonal equation in each cell, there is error in the functionality of the model and in the coarseness of the defined velocity models. Also, there is uncertainty in the velocity models themselves, as they are approximations of the true earth structure. These uncertainties propagate into the computed values and contribute to an overall uncertainty in the produced travel times. Spatial uncertainty is also present in the reflector depths used in the 2-D reflection models - in particular, the true geometry of the JdF plate, and in the estimated depths of the E-layer which were finely adjusted to minimize residuals and thus are not strictly constrained. Another source of error in this study are the approximations made in the development of the modelling program. These include the use of a spherical earth model in the projection of angular distance to longitudes in the ObsPy.TauPy model ray path plotting and in the radius of the earth being set to a constant 6371 km in the Pykonal modelling.

Lastly, I comment on the uncertainty in the choosing of the actual arrival times from each station's seismograms. This picking uncertainty is deemed to be ~ ±1 s based on repeated measurements of true arrival time wherein the maximum variation was 0.8 s.

These sources of error propagate through the determination of both modelled and actual travel times, and therefore into the calculated residuals used in the analysis. However, as the major conclusions drawn by the results were generally consistent across each modelling scheme, they are considered to be acceptable within the bounds of these uncertainties.

8.2 Future work

It was found that the implementation of modelling reflections using the NE-SW profile B as opposed to the W-E profile A had the effect of decreasing the residuals for the stations closest to the profile, therefore motivating the need for a 3-D velocity model to best model the arrival times of seismic waves in this region. As the study region is tectonically complex, with the northeast-dipping JdF plate as well as other seismically reflective layers such as the E-layer, results would be improved using a more complex model compared to the 1-D and 2-D modelling conducted in this research. As it was found that Pykonal worked well for the extent of the 2-D modelling, a 3-D model could be implemented into Pykonal for seismic modelling in this region.

Furthermore, a conduction of said modelling with a larger array of stations in the region would allow for a more intensive and meaningful statistical analysis of the results. Also, to further study the unexpected bonus arrivals, the amplitude of these arrivals could be analysed to further constrain their origin. An analysis of which stations received the bonus arrival at large amplitude vs those which received a muted and almost unidentifiable bonus arrival between the P and S wave arrivals could improve the understanding of its phase origin.

9 Conclusions

This study aimed to determine the origin of bonus seismic wave arrivals observed across SchoolShake network seismographs during the March 3rd, 2025, Orcas Island earthquake by using seismic ray path and travel time modelling of various levels of complexity.

The results of this modelling demonstrate that the bonus arrivals are most consistent with a reflection of P wave energy off the E-layer. The 1-D ObsPy.TauPy modelling was successful in reproducing direct P and S wave arrivals; however, it was insufficient to reproduce the bonus arrival reflections, and the implementation of the 2-D model was necessary. 2-D modelling with Pykonal demonstrated that reflections off the top of the JdF slab arrive later than both the bonus arrivals and the S wave arrivals, ruling out this boundary as the source of the bonus arrivals.

However, modelling reflections off a shallower boundary consistent with the E-layer produced significantly improved agreement with the actual bonus arrival times. Residuals for the Pp reflection off the top of the E-layer were minimized to within ±0.8 seconds for both profile A and profile B models, providing strong evidence that the bonus arrivals are consistent with P wave energy reflecting off the top of the E-layer.

These results demonstrate the importance of using spatially complex models to accurately simulate seismic wave propagation in tectonically complex regions, such as the Cascadia subduction zone. While the simple 1-D models were effective for direct arrivals, accurately modelling reflected phases requires at least 2-D, and ideally 3-D, models. Future work incorporating a 3-D model and larger station distribution would further refine these results and improve constraints on the origin of the bonus seismic phase arrivals.

These findings are particularly important as the March 3rd, 2025, Orcas Island earthquake was the largest local event since the implementation of the SchoolShake network in 2021, leading to the observation of the previously unseen bonus arrival in this region. The bonus phase arrivals appear to have contributed to longer and more intense shaking, which has the potential to impose greater seismic hazard to the area in the event of a larger magnitude event. Ultimately, having a better understanding of the possible seismic ray travel paths in the region improves seismic hazard models and contributes to improved risk mitigation for the Greater Victoria area.

10 References

Beyreuther, M., Barsch, R., Krischer, L., Megies, T., Behr, Y., & Wassermann, J. (2009). An XML-SEED Format for the Exchange of Synthetic Seismograms. In *EOS Transactions of American Geophysical Union* (Vol. 81, Number 3). http://numpy.scipy.org

Bostock, M. G., Christensen, N. I., & Peacock, S. M. (2019). Seismicity in Cascadia. *Lithos*. https://api.semanticscholar.org/CorpusID:134015941

Bostock, M., Sammis, C., Littel, G., Peacock, S., & Calvert, A. (2025). Tectonic tremor: the chatter of mafic underplating. *EGU General Assembly Conference Abstracts,* EGU25-7179. https://doi.org/10.5194/egusphere-egu25-7179

Braile, L. (2010, February 26). *Seismic Wave Demonstrations and Animations*. Purdue University. Explorations In Earth Science. https://web.ics.purdue.edu/~braile/edumod/waves/WaveDemo.htm

Brillon, C., Schaeffer, A. S., & Nissen, E. K. (2024). SchoolShake - connecting schools to an international, open data seismic network. *Canadian Dam Association Bulletin Magazine*, *35*(3), 12–19.

British Geological Survey. (2026). *How are earthquakes detected, located and measured?*. British Geological Survey. https://www.bgs.ac.uk/discovering-geology/earth-hazards/earthquakes/how-are-earthquakes-detected/

Calvert, A. J., & Clowes, R. M. (1990). Deep, high-amplitude reflections from a major shear zone above the subducting Juan de Fuca plate. *Geology*, *18*(11), 1091–1094. https://doi.org/10.1130/0091-7613(1990)018<1091:DHARFA>2.3.CO;2

Cassidy, J., Rogers, G., Lamontagne, M., Halchuk, S., & Adams, J. (2010). Canada's Earthquakes: 'The Good, the Bad, and the Ugly'. *Geoscience Canada*, *37*(1), 1–16.

Crotwell, H., Owens, T., & Ritsema, J. (1999). The TauP Toolkit: Flexible seismic travel-time and raypath utilities. *Seismological Research Letters*, *70*. https://doi.org/10.1785/gssrl.70.2.154

U.S. Geological Survey Earthquake Hazards Program. (2026). *Earthquake Magnitude, Energy Release, and Shaking Intensity*. USGS. https://www.usgs.gov/programs/earthquake-hazards/earthquake-magnitude-energy-release-and-shaking-intensity

EarthScope Consortium. (2026). *SAGE Station Viewer*. NSF Seismological Facility for the Advancement of Geoscience, SAGE. https://ds.iris.edu/gmap/

Gao, H., & Long, M. D. (2022). Tectonics and geodynamics of the Cascadia Subduction Zone. *Elements*, *18*(4), 226–231. https://doi.org/10.2138/gselements.18.4.226

Geophysics for Practicing Geoscientists. (2017). *Seismic Velocity*. GPG. https://gpg.geosci.xyz/content/physical_properties/seismic_velocity_duplicate.html

Green, A. G., Clowes, R. M., Yorath, C. J., Spencer, C., Kanasewich, E. R., Brandon, M. T., & Brown, A. S. (1986). *Seismic reflection imaging of the subducting Juan de Fuca plate*. *319*(6050), 210–213. https://doi.org/10.1038/319210a0

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Horner, R. B., Rogers, C., & Wetmiller, R. (1983). Canadian earthquakes. *Seismological Series of the Earth Physics Branch*, (87).

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kennett, B. L. N., & Engdahl, E. R. (1991). Traveltimes for global earthquake location and phase identification. *Geophysical Journal International*, *105*(2), 429–465. https://doi.org/10.1111/j.1365-246X.1991.tb06724.x

Nedimović, M., Hyndman, R., Ramachandran, K., & Spence, G. (2003). Reflection signature of seismic and aseismic slip on the northern Cascadia subduction interface. *Nature*, *424*, 416–420. https://doi.org/10.1038/nature01840

Nicholson, T., Bostock, M., & Cassidy, J. F. (2005). New constraints on subduction zone structure in northern Cascadia. *Geophysical Journal International*, *161*(3), 849–859. https://doi.org/10.1111/j.1365-246X.2005.02605.x

Ramachandran, K. (2001). *Velocity Structure of S.W. British Columbia, and N.W. Washington, from 3-D Non-linear Seismic Tomography*. PhD dissertation, University of Victoria.

RaspberryShake. (2024, February). *Raspberry Shake seismometers monitoring human-produced and geological Earth impacts is a global success for Raspberry Shake*. Raspberry Pi. https://www.raspberrypi.com/app/uploads/2024/02/Raspberry-Shake-seismometers-2.pdf

Rogers, G. C. (1983). *Seismotectonics of British Columbia*. PhD dissertation, University of British Columbia. https://open.library.ubc.ca/collections/831/items/1.0052938

Sethian, J. A. (1996). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, *93*(4), 1591–1595. https://doi.org/10.1073/pnas.93.4.1591

Stevens, N., Malone, S., Hutko, A., Hartog, R., Tobin, H., Gibbons, D., & Wright, A. (2025). *M4.5 Earthquake at Orcas Island, Washington, March 3rd, 2025*. PNSN. https://pnsn.org/blog/m4-5-earthquake-at-orcas-island-washington-march-3rd-2025

Stone, I., Vidale, J., Han, S., & Roland, E. (2018). Catalog of offshore seismicity in Cascadia: insights into the regional distribution of microseismicity and its relation to subduction processes. *Journal of Geophysical Research: Solid Earth*, *123*. https://doi.org/10.1002/2017JB014966

University of Alberta. (2013). *Seismic Phases*. Ppt slide deck. https://sites.ualberta.ca/~ygu/courses/geoph624/notes/lecture3-4_2013.pdf

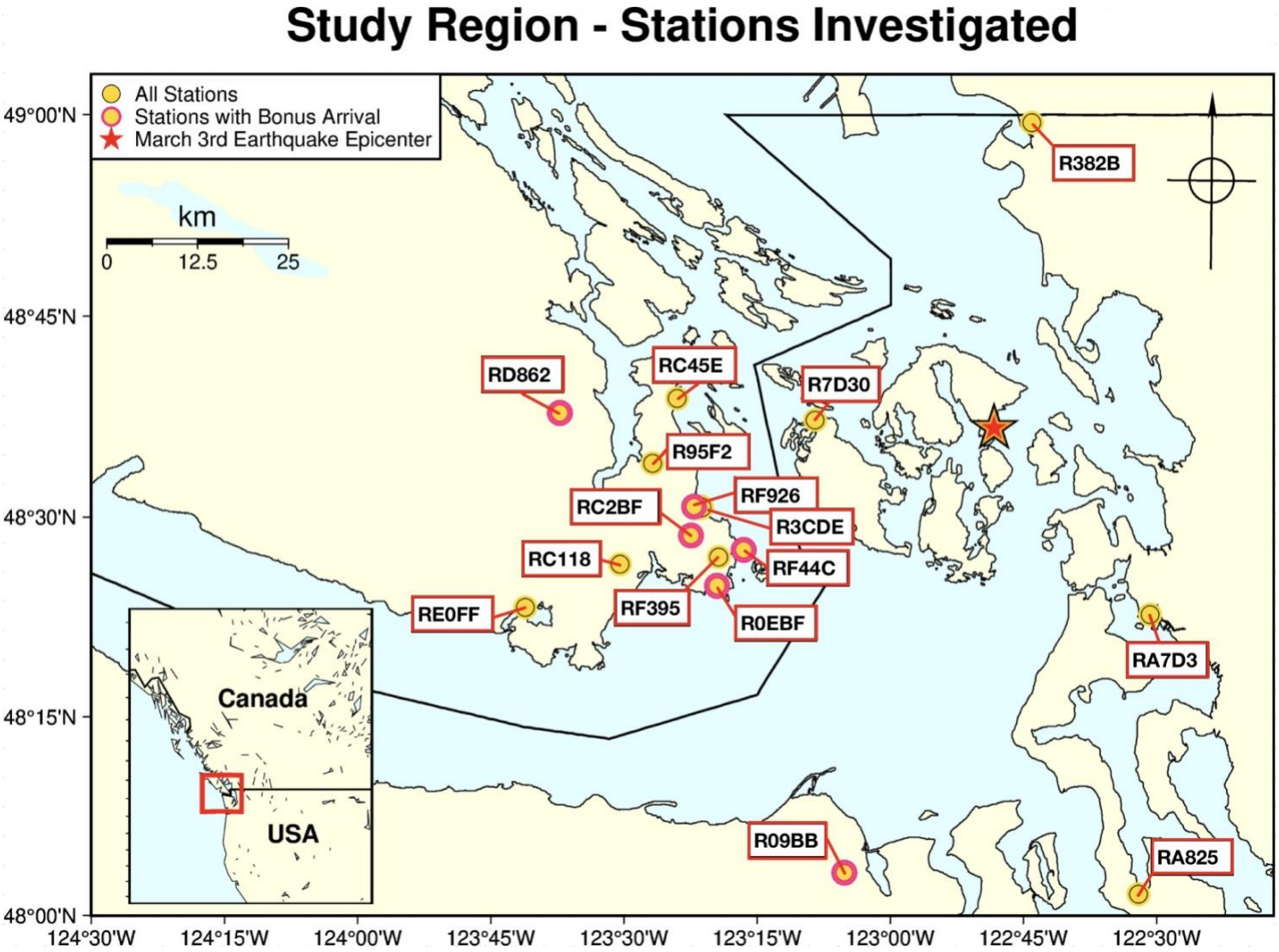
USGS. (2025). *M 4.5 - 2025 Orcas Island, Washington*. USGS. https://earthquake.usgs.gov/earthquakes/eventpage/uw62078906/executive

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … Contributors, S. 1. 0. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*(3), 261–272. https://doi.org/10.1038/s41592-019-0686-2

White, M. C. A., Fang, H., Nakata, N., & Ben-Zion, Y. (2020). PyKonal: A python package for solving the eikonal equation in spherical and cartesian coordinates using the fast marching method. *Seismological Research Letters*, *91*(4), 2378–2389. https://doi.org/10.1785/0220190318

Appendix

A – Map of Stations used in computations, labelled by name



[ ]:
```
from obspy.taup.tau_model import VelocityModel
from obspy.taup import TauPyModel
from obspy.taup.taup_create import build_taup_model
import matplotlib.pyplot as plt
```

1 1D ObsPy.TauP Modelling

[ ]:
```
#Loading in velocity models - 1D CN03 and 1D CN03 Deeper Boundary and converted␣
 ↪to .npz
#format and loaded into TauPyModel for use in travel time calculations

filename = "/opt/anaconda3/envs/pykonal/lib/python3.8/site-packages/obspy/taup/
 ↪data/CN03.nd"
filename_dchange = "V models/CN03_d_change.nd"
taup_model = build_taup_model(filename)
taup_model_d_change = build_taup_model(filename_dchange)


model = TauPyModel(model="CN03")
model_dchange = TauPyModel(model="CN03_d_change")
```

[ ]:
```
#Defining function to produce travel times and ray paths using TauP using␣
 ↪'get_travel_time_geo'
#and 'get_ray_paths_geo' which account for the geographical location of the␣
 ↪earthquake focus
#and each receiver

def travel_times(model_name, source_depth_in_km, source_lat, source_long,␣
 ↪receiver_lat, receiver_long):

    times = model_name.get_travel_times_geo(source_depth_in_km, source_lat,␣
 ↪source_long, receiver_lat, receiver_long, phase_list␣
 ↪=["p","P","Pvmp","s","S","Svms"])
    ray_paths_p = model_name.get_ray_paths_geo(source_depth_in_km, source_lat,␣
 ↪source_long, receiver_lat, receiver_long, phase_list=("p","P","Pvmp"))
```

```
    ray_paths_s = model_name.get_ray_paths_geo(source_depth_in_km, source_lat,␣
 ↪source_long, receiver_lat, receiver_long, phase_list=("s","S","Svms"))


    return times,ray_paths_p, ray_paths_s
```

[ ]:
```
#Defining physical parameters, geographical locations of stations and of March␣
 ↪3rd event
mar3_lat = 48.611
mar3_lon = -122.805
mar3_depth = 15.8

stn_names =␣
 ↪['R7D30','RA7D3','RF44C','R3CDE','RF926','RC45E','R382B','R09BB','RA825','RE0FF','RC118','R
stn_longs = [-123.141,-122.512,-123.275,-123.363,-123.368,-123.400,-122.
 ↪734,-123.086,-122.534,-123.685,-123.507,-123.325,-123.374,-123.620,-123.
 ↪446,-123.322]
stn_lats = [48.621,48.378,48.459,48.513,48.513,48.648,48.990,48.054,48.027,48.
 ↪387,48.441,48.414,48.477,48.630,48.567,48.450]
```

[ ]:
```
#Computing travel times for p,s,pvmp, and svms for CN03 Model

stn_travel_times = []
for i in range(len(stn_names)):
    tt =␣
 ↪travel_times(model,mar3_depth,mar3_lat,mar3_lon,stn_lats[i],stn_longs[i])[0]
    stn_travel_times.append(tt)

p_arrivals = []
s_arrivals = []
pvmp_arrivals = []
svms_arrivals = []
for i in range(len(stn_travel_times)):
    p_arrivals.append(stn_travel_times[i][0])
    s_arrivals.append(stn_travel_times[i][2])
    pvmp_arrivals.append(stn_travel_times[i][1])
    svms_arrivals.append(stn_travel_times[i][3])
```

[ ]:
```
#Repeating travel time calculations for depth change model
#- reflector boundary at 35 km depth
stn_travel_times_dchange = []
for i in range(len(stn_names)):
    tt =␣
 ↪travel_times(model_dchange,mar3_depth,mar3_lat,mar3_lon,stn_lats[i],stn_longs[i])[0]
    stn_travel_times_dchange.append(tt)
```

```
p_arrivals_dchange = []
s_arrivals_dchange = []
pvmp_dchange  = []
svms_dchange = []
for i in range(len(stn_travel_times_dchange)):
    p_arrivals_dchange.append(stn_travel_times_dchange[i][0])
    pvmp_dchange.append(stn_travel_times_dchange[i][1])
    s_arrivals_dchange.append(stn_travel_times_dchange[i][2])
    svms_dchange.append(stn_travel_times_dchange[i][3])
```

[ ]:
```
#Plotting ray paths for 1D model - Vp and Vs
#As horizontal unit produced by 'get_ray_paths.plot_rays' is in angular␣
 ↪distance degrees
#the axis was changed to longitude by projecting each point computed ray path␣
 ↪onto a great circle using
#but also its azimuth relative to the source
#spherical trigonometry, to incorporate not only the stations angular distance␣
 ↪from the source.

#Create subplot grids
fig_p, axes_p = plt.subplots(6, 3, figsize=(20, 35))
fig_s, axes_s = plt.subplots(6, 3, figsize=(20, 35))
fig_p.suptitle("1D ObsPy P and Pvmp Modelling",fontsize=25,fontweight='bold',␣
 ↪y=0.95)
fig_s.suptitle("1D ObsPy S and Svms Modelling",fontsize=25,fontweight='bold',␣
 ↪y=0.95)
axes_p = axes_p.flatten()
axes_s = axes_s.flatten()

#Azimuth each station
azs = []
for i in range(len(stn_names)):
    _, az, _ = gps2dist_azimuth(
        mar3_lat, mar3_lon,
        stn_lats[i], stn_longs[i]
    )
    azs.append(np.radians(az))

# Source in radians
lat1 = np.radians(mar3_lat)
lon1 = np.radians(mar3_lon)

# Loop over stations
for i in range(len(stn_names)):

    # P wave plots
    ax = axes_p[i]
```

```
    for j, ray in enumerate(stn_wave_paths_p[i]):
        colours = ['green', 'mediumvioletred']
        phase = ray.name
        dist = ray.path["dist"]
        depth = ray.path["depth"]
        az = azs[i]

        # Convert ray path dist in degrees to longitude
        lat2 = np.arcsin(
            np.sin(lat1)*np.cos(dist) +
            np.cos(lat1)*np.sin(dist)*np.cos(az)
        )

        lon2 = lon1 + np.arctan2(
            np.sin(az)*np.sin(dist)*np.cos(lat1),
            np.cos(dist) - np.sin(lat1)*np.sin(lat2)
        )

        lons = np.degrees(lon2)

        ax.plot(lons, depth, label=phase, color=colours[j])

    # Source
    ax.scatter(
        mar3_lon, mar3_depth,
        marker="*",
        s=200,
        color="yellow",
        edgecolor="black",
        zorder=10
    )
# Reference lines added at the surface (0 km) and at reflector (30 km)
    ax.axhline(0, color='grey', linewidth=1)
    ax.axhline(30, color='grey', linewidth=1)

    ax.invert_yaxis()
    ax.locator_params(axis='x', nbins=5)
    ax.ticklabel_format(useOffset=False)
    ax.scatter(stn_longs[i], 0,marker="v", color='red', s=50, label='Station')
    ax.set_title(f"{stn_names[i]} - Vp", fontsize=20)
    ax.set_xlabel("Longitude (°)",fontsize=18)
    ax.set_ylabel("Depth (km)",fontsize=18)


fig_p.tight_layout(rect=[0, 0, 1, 0.95])
for ax in axes_p:
```

```
    ax.locator_params(axis='x', nbins=3)

    # S wave plots

for i in range(len(stn_names)):

    ax = axes_s[i]

    for j, ray in enumerate(stn_wave_paths_s[i]):
        colours = ['green', 'mediumvioletred']
        phase = ray.name
        dist = ray.path["dist"]
        depth = ray.path["depth"]
        az = azs[i]

        lat2 = np.arcsin(
            np.sin(lat1)*np.cos(dist) +
            np.cos(lat1)*np.sin(dist)*np.cos(az)
        )

        lon2 = lon1 + np.arctan2(
            np.sin(az)*np.sin(dist)*np.cos(lat1),
            np.cos(dist) - np.sin(lat1)*np.sin(lat2)
        )

        lons = np.degrees(lon2)

        ax.plot(lons, depth, label=phase, color=colours[j])

    ax.scatter(
        mar3_lon, mar3_depth,
        marker="*",
        s=200,
        color="yellow",
        edgecolor="black",
        zorder=10
    )
# Reference lines added at the surface (0 km) and at reflector (30 km)
    ax.axhline(0, color='grey', linewidth=1)
    ax.axhline(30, color='grey', linewidth=1)

    ax.invert_yaxis()
    ax.locator_params(axis='x', nbins=5)
    ax.ticklabel_format(useOffset=False)
    ax.scatter(stn_longs[i], 0,marker="v", color='red', s=50, label='Station')

    ax.set_title(f"{stn_names[i]} - Vs", fontsize=20)
```

```
    ax.set_xlabel("Longitude (°)",fontsize=18)
    ax.set_ylabel("Depth (km)",fontsize=18)

#Layout

fig_s.tight_layout(rect=[0, 0, 1, 0.95])
for ax in axes_s:
    ax.locator_params(axis='x', nbins=3)
plt.show()
```

[ ]:
```
#Repeating above plotting scheme for deeper reflection model

#Create subplot grids
fig_p, axes_p = plt.subplots(6, 3, figsize=(20, 35))
fig_s, axes_s = plt.subplots(6, 3, figsize=(20, 35))
fig_p.suptitle("1D ObsPy P and Pvmp Modelling - Deeper␣
 ↪Reflector",fontsize=25,fontweight='bold', y=0.95)
fig_s.suptitle("1D ObsPy S and Svms Modelling - Deeper␣
 ↪Reflector",fontsize=25,fontweight='bold', y=0.95)
axes_p = axes_p.flatten()
axes_s = axes_s.flatten()

#Azimuth each station
azs = []
for i in range(len(stn_names)):
    _, az, _ = gps2dist_azimuth(
        mar3_lat, mar3_lon,
        stn_lats[i], stn_longs[i]
    )
    azs.append(np.radians(az))

# Source in radians
lat1 = np.radians(mar3_lat)
lon1 = np.radians(mar3_lon)

#Loop over stations
for i in range(len(stn_names)):

    #P wave plots
    ax = axes_p[i]

    for j, ray in enumerate(stn_wave_paths_p_dchange[i]):
        colours = ['green', 'mediumvioletred']
        phase = ray.name
        dist = ray.path["dist"]
        depth = ray.path["depth"]
        az = azs[i]
```

```
        # Convert ray path dist in degrees to longitude
        lat2 = np.arcsin(
            np.sin(lat1)*np.cos(dist) +
            np.cos(lat1)*np.sin(dist)*np.cos(az)
        )

        lon2 = lon1 + np.arctan2(
            np.sin(az)*np.sin(dist)*np.cos(lat1),
            np.cos(dist) - np.sin(lat1)*np.sin(lat2)
        )

        lons = np.degrees(lon2)

        ax.plot(lons, depth, label=phase, color=colours[j])

    # Source
    ax.scatter(
        mar3_lon, mar3_depth,
        marker="*",
        s=200,
        color="yellow",
        edgecolor="black",
        zorder=10
    )
# Reference lines added at the surface (0 km) and at deeper reflector (35 km)
    ax.axhline(0, color='grey', linewidth=1)
    ax.axhline(35, color='grey', linewidth=1)

    ax.invert_yaxis()
    ax.locator_params(axis='x', nbins=5)
    ax.ticklabel_format(useOffset=False)
    ax.scatter(stn_longs[i], 0, marker='v',color='red', s=50, label='Station')
    ax.set_title(f"{stn_names[i]} - Vp", fontsize=20)
    ax.set_xlabel("Longitude (°)",fontsize=18)
    ax.set_ylabel("Depth (km)",fontsize=18)


fig_p.tight_layout(rect=[0, 0, 1, 0.95])
for ax in axes_p:
    ax.locator_params(axis='x', nbins=3)



    # S wave plots
for i in range(len(stn_names)):
```

```
    ax = axes_s[i]

    for j, ray in enumerate(stn_wave_paths_s_dchange[i]):
        colours = ['green', 'mediumvioletred']
        phase = ray.name
        dist = ray.path["dist"]
        depth = ray.path["depth"]
        az = azs[i]

        lat2 = np.arcsin(
            np.sin(lat1)*np.cos(dist) +
            np.cos(lat1)*np.sin(dist)*np.cos(az)
        )

        lon2 = lon1 + np.arctan2(
            np.sin(az)*np.sin(dist)*np.cos(lat1),
            np.cos(dist) - np.sin(lat1)*np.sin(lat2)
        )
        lons = np.degrees(lon2)

        ax.plot(lons, depth, label=phase, color=colours[j])

    ax.scatter(
        mar3_lon, mar3_depth,
        marker="*",
        s=200,
        color="yellow",
        edgecolor="black",
        zorder=10
    )
# Reference lines added at the surface (0 km) and at deeper reflector (35 km)
    ax.axhline(0, color='grey', linewidth=1)
    ax.axhline(35, color='grey', linewidth=1)

    ax.invert_yaxis()
    ax.locator_params(axis='x', nbins=5)
    ax.ticklabel_format(useOffset=False)
    ax.scatter(stn_longs[i], 0,marker="v", color='red', s=50, label='Station')

    ax.set_title(f"{stn_names[i]} - Vs", fontsize=20)
    ax.set_xlabel("Longitude (°)",fontsize=18)
    ax.set_ylabel("Depth (km)",fontsize=18)

#Layout

fig_s.tight_layout(rect=[0, 0, 1, 0.95])
for ax in axes_s:
```

```
    ax.locator_params(axis='x', nbins=3)
plt.show()
```

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import pykonal
from scipy.interpolate import interp1d
```

0.1 1D Pykonal Modelling

[2]:
```
#Defining locations of event and stations
EARTH_RADIUS = 6371 # Earth radius in km.
CMB = 3480 # Core-mantle boundary radius in km.
ICB = 1220 # Inner-core outer-core boundary radius in km.
SUB_PLATE = 6341 #Plate

#Defining event location
event_lat = 48.611
event_long = -122.805
event_depth = 15.8

#Defining station locations
stn_names =␣
 ↪['R7D30','RA7D3','RF44C','R3CDE','RF926','RC45E','R382B','R09BB','RA825','RE0FF','RC118','R
stn_longs = [-123.141,-122.512,-123.275,-123.363,-123.368,-123.400,-122.
 ↪734,-123.086,-122.534,-123.685,-123.507,-123.325,-123.374,-123.620,-123.
 ↪446,-123.322]
stn_lats = [48.621,48.378,48.459,48.513,48.513,48.648,48.990,48.054,48.027,48.
 ↪387,48.441,48.414,48.477,48.630,48.567,48.450]
stn_depths = np.zeros_like(stn_longs)

#Defining station + event locations in spherical coordinates for 1-D Pykonal␣
 ↪model
#Here, the radius was set to 6371 km as the stations are all on the surface,
#the theta coordinate was set to a constant /2 slice to work in the 2-D domain,
#and the phi coordinate was set to the longitude value of each station in␣
 ↪radians.

model_theta = np.pi/2
```

```
station_coords={}
for i in range(len(stn_names)):
    station_coords[stn_names[i]] = (
        (EARTH_RADIUS),
        model_theta,
        (np.deg2rad(stn_longs[i]))
    )

src_loc = (EARTH_RADIUS-event_depth, model_theta, np.deg2rad(event_long))
```

[ ]:
```
#Defining JdF slab depths and longitudes
#Profile A
slab_data = [
    ((-126.9, -127.0),  5.0),
    ((-126.17, -126.27), 10.0),
    ((-125.79, -125.88),  20.0),
    ((-124.94, -125.07),  25.0),
    ((-124.46, -124.67), 30.0),
    ((-124.005, -124.233), 35.0),
    ((-123.59, -123.813), 40.0),
    ((-123.26, -123.44), 45.0),
    ((-122.97, -123.15), 50.0),

]
slab_lons = []
slab_depths = []

#Averaging longitude range for each defined depth
for (lon_min, lon_max), depth in slab_data:
    slab_lons.append(0.5 * (lon_min + lon_max))
    slab_depths.append(depth)

slab_lons = np.array(slab_lons)
slab_depths = np.array(slab_depths)


#Interpolating discrete slab data over smooth line
f = interp1d(slab_lons, slab_depths, kind='linear')

slab_lons_dense = np.linspace(min(slab_lons), max(slab_lons), 200)
slab_depths_dense = f(slab_lons_dense)

slab_x_km = lon_to_x(slab_lons_dense)
slab_z_km = slab_depths_dense

#Profile B
slab_pb = [
```

```
    ((-125.25), 10.0),
    ((-124.58),  15.0),
    ((-124.34),  20.0),
    ((-124.14), 25.0),
    ((-123.55), 30.0),
    ((-123.38), 35.0),
    ((-123.21), 40.0),
    ((-123.09), 45.0),
    ((-122.56), 50.0)

]
slab_lons_pb = []
slab_depths_pb = []

for (lon), depth in slab_pb:
    slab_lons_pb.append(lon)
    slab_depths_pb.append(depth)

slab_lons_pb = np.array(slab_lons_pb)
slab_depths_pb = np.array(slab_depths_pb)

#Interpolating discrete slab data over smooth line
fpb = interp1d(slab_lons_pb, slab_depths_pb, kind='linear')

slab_lons_dense_pb = np.linspace(min(slab_lons_pb), max(slab_lons_pb), 200)
slab_depths_dense_pb = f(slab_lons_dense_pb)

slab_x_km_pb = lon_to_x(slab_lons_dense_pb)
slab_z_km_pb = slab_depths_dense_pb
```

0.2 Loading in the 1D velocity model

[ ]:
```
vv_pkikp = pd.read_csv("V models/CN03_pyk_PKIKP.nd",delim_whitespace=True)
vv_pkikp.columns = ["depth","radius","vp","vs"]
depths = vv_pkikp["depth"].values
vp = vv_pkikp["vp"].values
vs = vv_pkikp["vs"].values
radius = vv_pkikp["radius"].values
```

0.3 Defining solver grid and writing solver

[ ]:
```
# Defining number of nodes in the radial and azimuthal directions.
#To approximate the 2-D slice as we are working with a 1-D velocity model
#interpolated onto a 2-D slice, the theta dimension was set to only 5 nodes
#centered on the events source to approximate the 2-D slice while retaining
#the Pykonal 3-D spherical coordinate framework required by the solver.
```

```
nrho, nphi = 512,512
ntheta = 5
dtheta = np.radians(0.1)
path = (EARTH_RADIUS, SUB_PLATE) # This defines the first segment of the path.

plt.close("all")
#Longitudinal Domain of solver set from extent of station coordinates ± 0.5␣
 ↪degrees for buffer
#to ensure the domain covered all ray paths
phis = np.array([c[2] for c in station_coords.values()])
phi_min = min(phis.min(), src_loc[2]) - np.radians(0.5)
phi_max = max(phis.max(), src_loc[2]) + np.radians(0.5)

solver = pykonal.solver.PointSourceSolver(coord_sys="spherical")

# Defining the computational domain.
r_min = EARTH_RADIUS - 100
r_max = EARTH_RADIUS

solver.vv.min_coords = (
    r_min,
    np.pi/2 - ntheta*dtheta/2,
    phi_min
)
solver.vv.node_intervals = (
    (r_max - r_min) / (nrho - 1),
    dtheta,
    (phi_max - phi_min) / (nphi - 1),
)
solver.vv.npts = nrho, ntheta, nphi

# Interpolating the CN03 model onto the computational grid.
#Pykonal requires a velocity at every node across the grid, so the 1-D model␣
 ↪was interpolated
#onto the computational grid using a linear interpolation

idx = np.argsort(radius)

solver.vv.values = np.interp(
    solver.vv.nodes[...,0],
    radius[idx],
    vp[idx]
)

#To ensure smooth and consistent stability in the Eikonal solver,
#the source location was snapped to the nearest grid node in the computational␣
 ↪grid.
```

```
solver.src_loc = src_loc
nodes = solver.vv.nodes
dist = np.linalg.norm(nodes - np.array(src_loc), axis=-1)
idx = np.unravel_index(np.argmin(dist), nodes.shape[:-1])
solver.src_loc = tuple(nodes[idx])
solver.solve()

#Direct P Travel Time calculations
arrival_times = {}
for name, coord in station_coords.items():
    coord = np.array(coord).reshape(1,3)
    tt = solver.tt.resample(np.array(coord))[0]
    arrival_times[name] = tt
print(arrival_times) #Direct arrival P wave 1D Pykonal travel times
```

[ ]:
```
#Direct S wave arrival
#Creating Vs solver
solver.vv.values = np.interp(solver.vv.nodes[..., 0],vv_pkikp["radius"].
 ↪values[-1::-1],vv_pkikp["vs"].values[-1::-1])
solver.src_loc = src_loc
solver.solve()

nodes = solver.vv.nodes
dist = np.linalg.norm(nodes - np.array(src_loc), axis=-1)
idx = np.unravel_index(np.argmin(dist), nodes.shape[:-1])
solver.src_loc = tuple(nodes[idx])
solver.solve()

arrival_times = {}
for name, coord in station_coords.items():
    coord = np.array(coord).reshape(1,3)
    tt = solver.tt.resample(np.array(coord))[0]
    arrival_times[name] = tt

print(arrival_times) #Direct arrival P wave 1D Pykonal travel times
```

0.4 Tracing P and S direct arrivals on interpolated 1D model

[ ]:
```
#Direct P wave arrival Ray path tracing
#done by tracing the gradient of the travel time field from each station back␣
 ↪towards the source, as done using the 'solver.trace_ray()'
#function with the argument being the receiver location.

fig, ax = plt.subplots(figsize=(9, 4))
itheta = ntheta // 2   # middle theta slice
```

```
nodes_2d = solver.vv.nodes[:, itheta, :, :]
vp_2d    = solver.vv.values[:, itheta, :]
#Background Vp model (2D)
r_grid = nodes_2d[..., 0]
phi_grid = nodes_2d[..., 2]
depth_grid = EARTH_RADIUS - r_grid
lon_grid = np.degrees(phi_grid)
#Plotting Vp map
pcm = ax.pcolormesh(
    lon_grid,
    depth_grid,
    vp_2d,
    cmap="viridis",
)

# Vp Ray paths

#The resulting ray paths were extracted as sequences of nodes in spherical␣
 ↪coordinates
#and converted to longitude and depth for visualization on the 2-D plots

for name, rx_loc in station_coords.items():
    rx_loc = np.array(rx_loc)
    ray = solver.trace_ray(rx_loc)
    # Keep only top 65 km
    mask = (EARTH_RADIUS - ray[:, 0]) <= 65.0
    ray = ray[mask]
    r_ray = ray[:, 0]
    phi_ray = ray[:, 2]
    depth_ray = EARTH_RADIUS - r_ray
    lon_ray = np.degrees(phi_ray)
    ax.plot(lon_ray, depth_ray, linewidth=0.8,color='magenta')
    rx_lon = np.degrees(rx_loc[2])
# Station depth (surface = 0 km)
    rx_depth = 0.0
# Plot as small triangle
    ax.plot(
        rx_lon,
        rx_depth,
        marker='^',        # triangle
        markersize=6,
        color='magenta',
        linestyle='None',
    )
ax.plot(rx_lon,rx_depth,marker='^',markersize=6,
        color='magenta',
        linestyle='None', label = 'Stations')
```

```
# JdF Slab
#included on the plot to provide context of the 2-D slice
ax.plot(
    slab_lons,
    slab_depths,
    color="black",
    linewidth=2.5,
    label="JdF Slab"
)
#Plotting hypocenter
ax.plot(
    event_long,
    event_depth,
    'r*',
    ms=14,
    label='Source',markeredgecolor = 'k'
)

# Formatting
ax.set_ylim(45, 0)
ax.set_xlim(-124.15,-122.1)
ax.set_xlabel("Longitude (°)")
ax.set_ylabel("Depth (km)")
ax.set_title("1D Pykonal Vp model - Direct P Arrival")

ax.legend(fontsize=8,loc = 'lower right')
plt.colorbar(pcm, ax=ax, label="Vp (km/s)")
```

[ ]:
```
#Direct S wave arrival Ray path tracing
itheta_mid = 1
nodes = solver.tt.nodes[:, itheta_mid, :, :]
vs = solver.vv.values[:, itheta_mid, :]

#Background Vp model (2D)
r_grid = nodes[..., 0]
phi_grid = nodes[..., 2]
depth_grid = EARTH_RADIUS - r_grid
lon_grid = np.degrees(phi_grid)

fig, ax = plt.subplots(figsize=(9, 4))
#Plotting Vp map
pcm = ax.pcolormesh(
    lon_grid,
    depth_grid,
    vs,
    cmap="viridis",
```

```
)
# Vs Ray paths
for name, rx_loc in station_coords.items():

    rx_loc = np.array(rx_loc)
    ray = solver.trace_ray(rx_loc)

    # Keep only top 65 km
    mask_ray = (EARTH_RADIUS - ray[:, 0]) <= 65.0
    ray = ray[mask_ray]

    r_ray = ray[:, 0]
    phi_ray = ray[:, 2]

    depth_ray = EARTH_RADIUS - r_ray
    lon_ray = np.degrees(phi_ray)

    ax.plot(lon_ray, depth_ray, linewidth=0.8,color='magenta')
    rx_lon = np.degrees(rx_loc[2])
# Station depth (surface = 0 km)
    rx_depth = 0.0
    ax.plot(
        rx_lon,
        rx_depth,
        marker='^',
        markersize=6,
        color='magenta',
        linestyle='None',
    )
ax.plot(
        rx_lon,
        rx_depth,
        marker='^',
        color='magenta',
        linestyle='None',label='Stations'
    )
ax.plot(
    slab_lons,
    slab_depths,
    color="black",
    linewidth=2.5,
    label="JdF Slab"
)
#Plotting source
ax.plot(
    event_long,
    event_depth,
```

```
    'r*',
    ms=14,
    label='Source',markeredgecolor = 'k'
)
#Formatting
ax.set_ylim(45, 0)
ax.set_xlim(-124.15,-122.1)
ax.set_xlabel("Longitude (°)")
ax.set_ylabel("Depth (km)")
ax.set_title("1D Pykonal Vs model - Direct S Arrival")

ax.legend(fontsize=8,loc ='lower right')
plt.colorbar(pcm, ax=ax, label="Vs (km/s)")
```

1 2D Pykonal Modelling

[7]:
```
from pykonal import solver
```

1.1 Loading in 2D velocity model

[ ]:
```
data = np.loadtxt("Savard/2D_Vp_model_lim.txt")

#the 2-D P wave velocity model read in as a .txt file and converted to a .npz␣
 ↪file
np.savez("Savard/2D_Vp_model_lim.npz", data=data)
file_path = 'Savard/2D_Vp_model_lim.npz'

vp_2d = np.load("Savard/2D_Vp_model_lim.npz")["data"]

#Defining Longitude and depth grid, from Savard 2018 grid_info
#origin of this coordinate system corresponding to (48.75°N,124.0°W)
#with no coordinate system rotation angle
longs = np.array([-200.00, -186.00, -172.00, -159.00 ,-145.00 ,-131.00, -118.
 ↪00, -104.00 ,-91.00 ,-77.00 ,-63.00 ,-50.00 ,-36.00 ,-23.00 ,-9.00  ,4.00␣
 ↪,17.00 ,31.00 ,44.00 ,58.00 ,72.00 ,85.00 ,99.00 ,112.00 ,126.00 ,140.00␣
 ↪,153.00 ,167.00 ,180.00, 194.00 ,208.00])
depths = np.array([0.0, 3.0, 6.0, 9.0, 12.0, 15.0, 18.0, 21.0,
              24.0, 27.0, 30.0, 33.0, 36.0, 39.0, 42.0, 45.0,
              48.0, 51.0, 54.0, 57.0, 60.0, 63.0, 66.0, 69.0,
              72.0, 75.0, 78.0, 81.0, 84.0, 87.0, 90.0])

#Defining grid

#computational grid defined using a Cartesian coordinate
#system with node spacing determined from the velocity model grid info
```

```
x =(longs)
z = depths
nx, nz = vp_2d.shape[1], vp_2d.shape[0]
dx = np.mean(np.diff(x))
dz=3
lon0 = -124 # model reference longitude
lat0 = 48.75
z0 = 0.0

#Defining solver
#As the solver requires a 3-D velocity field, the 2-D field
#was replicated across a small number of nodes in the y direction

solver_obj = solver.PointSourceSolver(coord_sys='cartesian')
ny = 11
dy = 1.0
solver_obj.vv.npts = (nx, ny, nz)
solver_obj.vv.node_intervals = (dx, dy, dz)

#'min_coords' set to the minimum longitude grid node of the model and at the␣
 ↪shallowest depth.
#at this point the solver has been created to span
#the full horizontal extent of the velocity profile and for depths of 0-90 km

solver_obj.vv.min_coords = (x.min(), -dy*(ny//2), depths.min())
vv = np.zeros((nx, ny, nz))
#Fill all y slices with velocity
for j in range(ny):
    vv[:, j, :] = vp_2d.T

solver_obj.vv.values = vv

#Creating function to convert longitude in degrees to distance in x
# To convert the longitudes of the stations and the event to horizontal␣
 ↪distances in km relative
#to the reference longitude of 124°W (origin of coordinate system)

km_per_deg_lon = 111.32 * np.cos(np.deg2rad(lat0))
def lon_to_x(lon):
    return (lon - lon0) * km_per_deg_lon
x_src = lon_to_x(event_long)
z_src = event_depth

#source location was defined using its geographic location and depth
#with longitude converted to distance along the profile as was done
#for the stations
```

```
nodes = solver_obj.vv.nodes.reshape(-1, 3)

#y coordinate of the source was set to 0 as this corresponds
#to the central plane of the quasi 2-D model
#the location of the earthquake was snapped to the nearest node in
#the grid before solving the eikonal equation

src_guess = np.array([x_src, 0.0, z_src])
src_idx = np.argmin(np.linalg.norm(nodes - src_guess, axis=1))
solver_obj.src_loc = nodes[src_idx]
```

[ ]:
```
#Defining station location in distance
#seismic stations were positioned along the model profile with each station's
#location corresponding x = longitude in km from the origin, y=0 and z=0

station_coords = {}

for i, lon in enumerate(stn_longs):
    x_rx = lon_to_x(lon)
    station_coords[f"STA{i}"] = np.array([x_rx, 0.0, 0.0])

#Solving
solver_obj.solve()
```

[ ]:
```
#Printing direct P travel times

#the first arriving P wave travel time calculation
#using the point source solver in Pykonal in Cartesian coordinates

arrival_times_2d = {}
for k, coord in station_coords.items():
    tt = solver_obj.tt.resample(coord.reshape(1,3))[0]
    arrival_times_2d[k] = tt

print("P wave Travel times (s):")
for k,v in arrival_times_2d.items():
    print(f"{k}: {v}")
```

Repeating for S model

[ ]:
```
data = np.loadtxt("Savard/2D_Vs_model.txt")
# Save as npz
np.savez("Savard/2D_Vs_model.npz", data=data)
file_path = 'Savard/2D_Vs_model.npz'
vs_2d = np.load("Savard/2D_Vs_model.npz")["data"]
depths = np.array([0.0, 3.0, 6.0, 9.0, 12.0, 15.0, 18.0, 21.0,
                   24.0, 27.0, 30.0, 33.0, 36.0, 39.0, 42.0, 45.0,
                   48.0, 51.0, 54.0, 57.0, 60.0, 63.0, 66.0, 69.0,
```

```
             72.0, 75.0, 78.0, 81.0, 84.0, 87.0, 90.0])
longs = np.array([-200.00, -186.00, -172.00, -159.00 ,-145.00 ,-131.00, -118.
 ↪00, -104.00 ,-91.00 ,-77.00 ,-63.00 ,-50.00 ,-36.00 ,-23.00 ,-9.00  ,4.00␣
 ↪,17.00 ,31.00 ,44.00 ,58.00 ,72.00 ,85.00 ,99.00 ,112.00 ,126.00 ,140.00␣
 ↪,153.00 ,167.00 ,180.00, 194.00 ,208.00])

vs_2d.shape
x = longs
z = depths
nx, nz = vs_2d.shape[1], vs_2d.shape[0]
dx = np.mean(np.diff(longs))
dz = np.mean(np.diff(depths))

lon0 =  -124     # model reference longitude
lat0 = 48.75
z0   = 0.0

solver_obj_vs = solver.PointSourceSolver(coord_sys='cartesian')

ny = 11
dy = 1.0

solver_obj_vs.vv.npts = (nx, ny, nz)
solver_obj_vs.vv.node_intervals = (dx, dy, dz)
solver_obj_vs.vv.min_coords = (longs.min(), -dy*(ny//2), depths.min())

vv = np.zeros((nx, ny, nz))

# 3. Fill all y slices with velocity
for j in range(ny):
    vv[:, j, :] = vs_2d.T  # now shapes match exactly

solver_obj_vs.vv.values = vv

km_per_deg_lon = 111.32 * np.cos(np.deg2rad(lat0))
def lon_to_x(lon):
    return (lon - lon0) * km_per_deg_lon


x_src = lon_to_x(event_long)
z_src = event_depth
nodes = solver_obj_vs.vv.nodes.reshape(-1, 3)
src_guess = np.array([x_src, 0.0, z_src])
src_idx = np.argmin(np.linalg.norm(nodes - src_guess, axis=1))
solver_obj_vs.src_loc = nodes[src_idx]

solver_obj_vs.solve()
```

[ ]:
```
arrival_times_2d_vs = {}

for k, coord in station_coords.items():
    tt = solver_obj_vs.tt.resample(coord.reshape(1,3))[0]
    arrival_times_2d_vs[k] = tt

print("Arrival times Vs (s):")
for k,v in arrival_times_2d_vs.items():
    print(f"{k}: {v}")
```

1.2 Plotting Direct P and S waves on 2D model Profile A

[ ]:
```
#Plotting direct P waves

#seismic ray paths for the direct P arrival calculated
#and produced by the solver as sequences of coordinates within
#the defined grid and then converted back to longitude and depth
#for visualization

vp_plot = solver_obj.vv.values[:, ny//2, :].T

z_plot = depths
x_plot_lon = x_to_lon(longs)

dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
#Cells are adjusted to lie along proper depth and lonitude grids
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)
x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2
zmin = depths.min() - dz/2
zmax = depths.max() + dz/2
z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))

#Plotting Vp map
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')
plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Pykonal Vp model - Direct P Arrival - Profile A')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon
```

```
ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)
for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon

    ax.plot(
        rx_lon,
        0.0,                  # surface depth
        marker='^',
        markersize=8,
        color='magenta',
        linestyle='None',
        zorder=9
    )
ax.plot(rx_lon,0.0, marker='^',
        markersize=8,
        color='magenta',
        linestyle='None',
        zorder=9,label='Stations')
for rx_loc in station_coords.values():
    ray = solver_obj.trace_ray(rx_loc)
    ray_lon = lon0 + ray[:, 0] / km_per_deg_lon
    ax.plot(ray_lon, ray[:, 2], 'magenta', linewidth=0.8)

ax.plot(
    slab_lons,
    slab_depths,
    color='black',
    linewidth=2.5,
    label='JdF Slab'
)

plt.legend(loc='lower right')
ax.set_ylim(45, 0)
ax.set_xlim(-124.15,-122.1)
plt.show()
```

[ ]:
```
#Plotting direct P waves - Zoomed out
vp_plot = solver_obj.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths
```

```
dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)

x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2
zmin = depths.min() - dz/2
zmax = depths.max() + dz/2
z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')
plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Pykonal Vp model - Direct P Arrival - Profile A - Zoomed Out')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon

ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)
for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon

    ax.plot(rx_lon,
        0.0,
        marker='^',
        markersize=4,
        color='magenta',
        linestyle='None',
        zorder=9)
ax.plot(
        rx_lon,
        0.0,
        marker='^',
        markersize=6,
        color='magenta',
```

```
        linestyle='None',
        zorder=9,label='Stations')

for rx_loc in station_coords.values():
    ray = solver_obj.trace_ray(rx_loc)
    ray_lon = lon0 + ray[:, 0] / km_per_deg_lon
    ax.plot(ray_lon, ray[:, 2], 'magenta', linewidth=0.8)
ax.plot(
    slab_lons,
    slab_depths,
    color='black',
    linewidth=2.5,
    label='JdF Slab'
)

plt.legend(loc='lower right')
plt.xlim(-126.8,-122)
plt.ylim(85,0)
plt.show()
```

[ ]:
```
#Plotting direct S waves

vs_plot = solver_obj_vs.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths

dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)

x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2

z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vs_plot,shading='flat')
plt.colorbar(pcm, ax=ax, label='Vs (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Pykonal Vs model - Direct S Arrival - Profile A')
src_x_km = solver_obj_vs.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon
```

```
ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)
for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon
    ax.plot(rx_lon, rx[2], 'magenta',marker='^', ms=8)
ax.plot(rx_lon, rx[2], 'magenta', ms=6,marker='^',label='Stations')
for rx_loc in station_coords.values():
    ray = solver_obj_vs.trace_ray(rx_loc)
    ray_lon = lon0 + ray[:, 0] / km_per_deg_lon
    ax.plot(ray_lon, ray[:, 2], 'magenta', linewidth=0.8)
ax.plot(
    slab_lons,
    slab_depths,
    color='black',
    linewidth=2.5,
    label='JdF Slab'
)

plt.legend(loc = 'lower right')
ax.set_xlim(-124.15,-122.1)
plt.ylim(45,0)
plt.show()
```

1.3 2D Reflection Pp and Ps Modelling

[ ]:
```
solver_obj.solve()
T_src = solver_obj.tt

grid_params = dict(
    npts=solver_obj.vv.npts,
    node_intervals=solver_obj.vv.node_intervals,
    min_coords=solver_obj.vv.min_coords,
    values=solver_obj.vv.values.copy(),
)

#Computing solver for upgoing ray path in reflected ray
#To compute the travel time for the upward propagating ray,
#a separate travel time solver was used for each station

solvers_sta = {}
T_sta = {}
for sta, rx_coord in station_coords.items():
```

```
    s = solver.PointSourceSolver(coord_sys="cartesian")

    s.vv.npts = grid_params["npts"]
    s.vv.node_intervals = grid_params["node_intervals"]
    s.vv.min_coords = grid_params["min_coords"]
    s.vv.values = grid_params["values"]
# each receiver defined as a source
# as Pykonal computes the travel times and ray paths from receiver to the source
    s.src_loc = rx_coord
    s.solve()

    solvers_sta[sta] = s
    T_sta[sta] = s.tt

#Defining slab points of possible reflection - Profile A and B
#to define where the rays are reflecting off of the JdF slab, the slab was␣
 ↪discretized into a
#series of points defined by their x and z locations on the grid and listed␣
 ↪into a set of possible
#reflection points for the reflection to occur

slab_pts = np.column_stack([
    slab_x_km,
    np.zeros_like(slab_x_km),
    slab_z_km
])
slab_pts_pb = np.column_stack([
    slab_x_km_pb,
    np.zeros_like(slab_x_km_pb),
    slab_z_km_pb
])

#Finding total travel times
#for each possible point of reflection on the slab, two travel times were␣
 ↪calculated:
#that of the down going ray from the earthquake and that of the upgoing ray␣
 ↪from slab to station.
#The total travel time was calculated as the sum of these two legs
reflection_pts = {}
T_reflect = {}
T_down = T_src.resample(slab_pts)

#the slab point that minimized the total travel time was selected as the␣
 ↪reflection point for that ray.
#The coordinates of the determined reflection points for all stations were␣
 ↪recorded for ray tracing
```

```
for sta in station_coords:
    T_up = T_sta[sta].resample(slab_pts)

    T_tot = T_down + T_up
    idx = np.nanargmin(T_tot)

    reflection_pts[sta] = slab_pts[idx]
    T_reflect[sta] = T_tot[idx]
```

1.4 Plotting reflected rays

[ ]:
```
#Reflected Pp ray - Profile A
vp_plot = solver_obj.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths

dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)

x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2

z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')

plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Pykonal Vp model - Reflected Pp Arrival - 48.6°N Profile - Zoomed␣
 ↪Out')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon

ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)
```

```
for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon
    ax.plot(rx_lon, rx[2], 'magenta',marker='^', ms=6)
ax.plot(rx_lon, rx[2], 'magenta',marker='^', ms=6,label='Stations')
#plotting the ray path of the two segment solver, the down going segment traced␣
 ↪from the reflection point back to the source and the upgoing segment
#traced from the reflection point to the receiver for each station

for sta, s_sta in solvers_sta.items():
    ref_pt = reflection_pts[sta]


    ray_down = solver_obj.trace_ray(ref_pt)


    ray_up = s_sta.trace_ray(ref_pt)[::-1]


    ray_down_lon = lon0 + ray_down[:, 0] / km_per_deg_lon
    ray_up_lon   = lon0 + ray_up[:, 0]   / km_per_deg_lon

    ax.plot(ray_down_lon, ray_down[:, 2], 'magenta', lw=0.8)
    ax.plot(ray_up_lon,   ray_up[:, 2],   'magenta', lw=0.8)

ax.plot(
    slab_lons,
    slab_depths,
    color='black',
    linewidth=2.5,
    label='JdF Slab'
)
plt.legend(loc='lower right')
plt.xlim(-126.8,-122)

plt.show()
```

[ ]:
```
#Reflection ray Profile A travel times
print("Reflected ray travel times:")
ref_tt = []
for sta, ref_pt in reflection_pts.items():

    T_down = T_src.resample(ref_pt.reshape(1,3))[0]

    T_up = T_sta[sta].resample(ref_pt.reshape(1,3))[0]

    T_total = T_down + T_up
```

```
    print(f"{sta}: tt = {T_total} s")
    ref_tt.append(T_total)
```

[ ]:
```
#Repeating for profile B Pp JdF reflection
grid_params = dict(
    npts=solver_obj.vv.npts,
    node_intervals=solver_obj.vv.node_intervals,
    min_coords=solver_obj.vv.min_coords,
    values=solver_obj.vv.values.copy(),
)

solvers_sta = {}
T_sta_pb = {}
for sta, rx_coord in station_coords.items():
    s = solver.PointSourceSolver(coord_sys="cartesian")

    s.vv.npts = grid_params["npts"]
    s.vv.node_intervals = grid_params["node_intervals"]
    s.vv.min_coords = grid_params["min_coords"]
    s.vv.values = grid_params["values"]


    s.src_loc = rx_coord
    s.solve()

    solvers_sta[sta] = s
    T_sta_pb[sta] = s.tt
reflection_pts_pb = {}
T_reflect_pb = {}


T_down_pb = T_src.resample(slab_pts_pb)

for sta in station_coords:
    T_up_pb = T_sta_pb[sta].resample(slab_pts_pb)

    T_tot_pb = T_down_pb + T_up_pb
    idx = np.nanargmin(T_tot_pb)

    reflection_pts_pb[sta] = slab_pts_pb[idx]
    T_reflect_pb[sta] = T_tot_pb[idx]

vp_plot = solver_obj.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths

dx = np.mean(np.diff(x_plot_lon))
```

```
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)

x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2

z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')
plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Pykonal Vp model - Reflected Pp Arrival - Profile B')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon

ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)
for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon
    ax.plot(rx_lon, rx[2], 'magenta',marker='^', ms=6)
ax.plot(rx_lon, rx[2], 'magenta',marker='^', ms=6,label='Stations')
for sta, s_sta in solvers_sta.items():
    ref_pt = reflection_pts_pb[sta]


    ray_down = solver_obj.trace_ray(ref_pt)


    ray_up = s_sta.trace_ray(ref_pt)[::-1]
    ray_down_lon = lon0 + ray_down[:, 0] / km_per_deg_lon
    ray_up_lon   = lon0 + ray_up[:, 0]   / km_per_deg_lon

    ax.plot(ray_down_lon, ray_down[:, 2], 'magenta', lw=0.8)
    ax.plot(ray_up_lon,   ray_up[:, 2],   'magenta', lw=0.8)


ax.plot(
```

```
    slab_lons_pb,
    slab_depths_pb,
    color='k',
    linewidth=2.5,
    label='JdF Slab Profile B'
)


plt.legend(loc='lower right')
ax.set_ylim(45, 0)
ax.set_xlim(-124.15,-122.1)
plt.show()
plt.show()
```

[ ]:
```
#Reflection Pp ray Profile B travel times
print("Reflected ray travel times PB:")
ref_tt = []
for sta, ref_pt in reflection_pts_pb.items():

    T_down_pb = T_src.resample(ref_pt.reshape(1,3))[0]

    T_up_pb = T_sta_pb[sta].resample(ref_pt.reshape(1,3))[0]

    T_total_pb = T_down_pb + T_up_pb

    print(f"{sta}: tt = {T_total_pb} s")
    ref_tt.append(T_total_pb)
```

[ ]:
```
#Repeating for Ps reflection - Profile A
solvers_sta_vs = {}
T_sta_vs = {}


for sta, rx_coord in station_coords.items():
    s_vs = solver.PointSourceSolver(coord_sys="cartesian")



    s_vs.vv.npts = solver_obj_vs.vv.npts
    s_vs.vv.node_intervals = solver_obj_vs.vv.node_intervals
    s_vs.vv.min_coords = solver_obj_vs.vv.min_coords
    s_vs.vv.values = solver_obj_vs.vv.values.copy()


    s_vs.src_loc = rx_coord
    s_vs.solve()
```

```
    solvers_sta_vs[sta] = s_vs
    T_sta_vs[sta] = s_vs.tt
T_ps = {}
T_P_down = solver_obj.tt.resample(slab_pts)


for sta in station_coords:
    T_S_up = T_sta_vs[sta].resample(slab_pts)


    T_tot = T_P_down + T_S_up
    idx = np.nanargmin(T_tot)


    reflection_pts[sta] = slab_pts[idx]
    T_ps[sta] = T_tot[idx]

#Plotting Ps reflection
vs_plot = solver_obj_vs.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths


dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)

x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2

z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')
plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Reflected Arrival Ps-wave Velocity Model - 48.6°N Profile')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon

ax.plot(
    src_lon,
```

```
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)
for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon
    ax.plot(rx_lon, rx[2], 'magenta',marker="^", ms=6)
ax.plot(rx_lon, rx[2], 'magenta',marker="^", ms=6,label="Stations")
for sta, s_sta in solvers_sta.items():
    ref_pt = reflection_pts[sta]


    ray_down = solver_obj.trace_ray(ref_pt)

    ray_up = solvers_sta_vs[sta].trace_ray(ref_pt)[::-1]

    ray_down_lon = lon0 + ray_down[:, 0] / km_per_deg_lon
    ray_up_lon   = lon0 + ray_up[:, 0]   / km_per_deg_lon

    ax.plot(ray_down_lon, ray_down[:, 2], 'magenta', lw=0.8)
    ax.plot(ray_up_lon,   ray_up[:, 2],   'magenta', lw=0.8)

ax.plot(
    slab_lons,
    slab_depths,
    color='black',
    linewidth=2.5,
    label='JdF Slab'
)
plt.legend(loc='lower right')
ax.set_xlim(-124.15,-122.1)
plt.ylim(45,0)
plt.show()
```

[ ]:
```
#Reflected Ps Travel times
print("Ps arrival times (s):")
for sta, tt in T_ps.items():
    print(f"{sta}: {tt}")
```

[ ]:
```
#Repeating for profile B Ps reflection
solvers_sta_vs = {}
T_sta_vs = {}


for sta, rx_coord in station_coords.items():
```

```
    s_vs = solver.PointSourceSolver(coord_sys="cartesian")



    s_vs.vv.npts = solver_obj_vs.vv.npts
    s_vs.vv.node_intervals = solver_obj_vs.vv.node_intervals
    s_vs.vv.min_coords = solver_obj_vs.vv.min_coords
    s_vs.vv.values = solver_obj_vs.vv.values.copy()


    s_vs.src_loc = rx_coord
    s_vs.solve()


    solvers_sta_vs[sta] = s_vs
    T_sta_vs[sta] = s_vs.tt
T_ps = {}
T_P_down = solver_obj.tt.resample(slab_pts_pb)


for sta in station_coords:
    T_S_up = T_sta_vs[sta].resample(slab_pts_pb)


    T_tot = T_P_down + T_S_up
    idx = np.nanargmin(T_tot)


    reflection_pts[sta] = slab_pts_pb[idx]
    T_ps[sta] = T_tot[idx]
vs_plot = solver_obj_vs.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths


dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)

x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2

z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')
```

```
plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Reflected Arrival Ps-wave Velocity Model - Profile B')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon

ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)
for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon
    ax.plot(rx_lon, rx[2], 'magenta',marker="^", ms=6)
ax.plot(rx_lon, rx[2], 'magenta',marker="^", ms=6,label="Stations")
for sta, s_sta in solvers_sta.items():
    ref_pt = reflection_pts[sta]

    ray_down = solver_obj.trace_ray(ref_pt)

    ray_up = solvers_sta_vs[sta].trace_ray(ref_pt)[::-1]

    ray_down_lon = lon0 + ray_down[:, 0] / km_per_deg_lon
    ray_up_lon   = lon0 + ray_up[:, 0]   / km_per_deg_lon

    ax.plot(ray_down_lon, ray_down[:, 2], 'magenta', lw=0.8)
    ax.plot(ray_up_lon,   ray_up[:, 2],   'magenta', lw=0.8)


ax.plot(
    slab_lons_pb,
    slab_depths_pb,
    color='black',
    linewidth=2.5,
    label='JdF Slab Profile B'
)
plt.legend(loc='lower right')
ax.set_xlim(-124.15,-122.1)
plt.ylim(45,0)
plt.show()
```

[ ]:
```
#Relected Ps wave travel times - Profile B
print("Ps arrival times (s):")
for sta, tt in T_ps.items():
    print(f" {tt}")
```

1.5 Next, Code for modelling ray path reflected off the E layer

[ ]:
```
#Defining E layer depths, same as was done for JdF slab above
Elayer_data = [
    ((-126.9, -127.0),  0),
    ((-126.17, -126.27), 2.0),
    ((-125.79, -125.88),  4.0),
    ((-124.94, -125.07),  9.0),
    ((-124.46, -124.67), 14.0),
    ((-124.005, -124.233), 19.0),
    ((-123.59, -123.813), 24.0),
    ((-123.26, -123.44), 29.0),
    ((-122.97, -123.15), 34.0),

]
E_lons = []
E_depths = []

for (lon_min, lon_max), depth in Elayer_data:
    E_lons.append(0.5 * (lon_min + lon_max))
    E_depths.append(depth)

E_lons = np.array(E_lons)
E_depths = np.array(E_depths)

from scipy.interpolate import interp1d

f = interp1d(E_lons, E_depths, kind='linear')

slab_lons_dense = np.linspace(min(E_lons), max(E_lons), 200)
slab_depths_dense = f(slab_lons_dense)

E_x_km = lon_to_x(slab_lons_dense)
E_z_km = slab_depths_dense
```

[ ]:
```
#Solving ray paths for reflection off the E layer
solver_obj.solve()
T_src = solver_obj.tt

grid_params = dict(
    npts=solver_obj.vv.npts,
    node_intervals=solver_obj.vv.node_intervals,
```

```
    min_coords=solver_obj.vv.min_coords,
    values=solver_obj.vv.values.copy(),
)

solvers_sta = {}
T_sta = {}
for sta, rx_coord in station_coords.items():
    s = solver.PointSourceSolver(coord_sys="cartesian")

    s.vv.npts = grid_params["npts"]
    s.vv.node_intervals = grid_params["node_intervals"]
    s.vv.min_coords = grid_params["min_coords"]
    s.vv.values = grid_params["values"]


    s.src_loc = rx_coord
    s.solve()

    solvers_sta[sta] = s
    T_sta[sta] = s.tt

slab_pts = np.column_stack([
    E_x_km ,
    np.zeros_like(E_x_km),
    E_z_km
])
reflection_pts = {}
T_reflect = {}


T_down = T_src.resample(slab_pts)

for sta in station_coords:
    T_up = T_sta[sta].resample(slab_pts)

    T_tot = T_down + T_up
    idx = np.nanargmin(T_tot)

    reflection_pts[sta] = slab_pts[idx]
    T_reflect[sta] = T_tot[idx]
```

[ ]:
```
#Plotting Reflection off E layer - Profile A
vp_plot = solver_obj.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths
```

```
dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)

x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2

z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')
plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Pykonal Vp model - Reflected Pp Arrival - Profile A - Approx E␣
 ↪layer')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon

ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)

for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon
    ax.plot(rx_lon, rx[2], 'magenta',marker='^', ms=6)
ax.plot(rx_lon, rx[2], 'magenta',marker='^', ms=6,label='Stations')
for sta, s_sta in solvers_sta.items():
    ref_pt = reflection_pts[sta]


    ray_down = solver_obj.trace_ray(ref_pt)


    ray_up = s_sta.trace_ray(ref_pt)[::-1]

    ray_down_lon = lon0 + ray_down[:, 0] / km_per_deg_lon
    ray_up_lon   = lon0 + ray_up[:, 0]   / km_per_deg_lon

    ax.plot(ray_down_lon, ray_down[:, 2], 'magenta', lw=0.8)
```

```
    ax.plot(ray_up_lon,   ray_up[:, 2],   'magenta', lw=0.8)


ax.plot(
    E_lons,
    E_depths,
    color='red',
    linewidth=2.5,
    label='Approx. E layer',
    linestyle= 'dashed'
)
ax.plot(
    slab_lons,
    slab_depths,
    color='black',
    linewidth=2.5,
    label='JdF Slab'
)
plt.legend(loc='lower right')
#plt.xlim(-126.8,-122)
ax.set_xlim(-124.15,-122.1)
plt.ylim(45,0)
plt.show()
```

[ ]:
```
#Pp E layer reflection travel times - Profile A
print("Reflected ray travel times - E layer:")
ref_tt = []
for sta, ref_pt in reflection_pts.items():

    T_down = T_src.resample(ref_pt.reshape(1,3))[0]

    T_up = T_sta[sta].resample(ref_pt.reshape(1,3))[0]

    T_total = T_down + T_up

    print(f"{sta}: tt = {T_total} s")
    ref_tt.append(T_total)
```

[ ]:
```
#Repeating for Ps E layer reflection
solvers_sta_vs = {}
T_sta_vs = {}


for sta, rx_coord in station_coords.items():
    s_vs = solver.PointSourceSolver(coord_sys="cartesian")
```

```
    s_vs.vv.npts = solver_obj_vs.vv.npts
    s_vs.vv.node_intervals = solver_obj_vs.vv.node_intervals
    s_vs.vv.min_coords = solver_obj_vs.vv.min_coords
    s_vs.vv.values = solver_obj_vs.vv.values.copy()


    s_vs.src_loc = rx_coord
    s_vs.solve()


    solvers_sta_vs[sta] = s_vs
    T_sta_vs[sta] = s_vs.tt
T_ps = {}
T_P_down = solver_obj.tt.resample(slab_pts)


for sta in station_coords:
    T_S_up = T_sta_vs[sta].resample(slab_pts)


    T_tot = T_P_down + T_S_up
    idx = np.nanargmin(T_tot)


    reflection_pts[sta] = slab_pts[idx]
    T_ps[sta] = T_tot[idx]
```

[ ]:
```
#Plotting Ps E layer reflection
vp_plot = solver_obj.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths

dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)

x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2

z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')
plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
```

```
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Pykonal Vp Model - Reflected Ps Arrival - Profile A - Approx E␣
 ↪Layer')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon

ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)
for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon
    ax.plot(rx_lon, rx[2], 'magenta',marker="^", ms=6)
ax.plot(rx_lon, rx[2], 'magenta',marker="^", ms=6,label="Stations")
for sta, s_sta in solvers_sta.items():
    ref_pt = reflection_pts[sta]


    ray_down = solver_obj.trace_ray(ref_pt)

    ray_up = solvers_sta_vs[sta].trace_ray(ref_pt)[::-1]

    ray_down_lon = lon0 + ray_down[:, 0] / km_per_deg_lon
    ray_up_lon   = lon0 + ray_up[:, 0]   / km_per_deg_lon

    ax.plot(ray_down_lon, ray_down[:, 2], 'magenta', lw=0.8)
    ax.plot(ray_up_lon,   ray_up[:, 2],   'magenta', lw=0.8)
ax.plot(
    E_lons,
    E_depths,
    color='red',
    linewidth=2.5,
    label='Approx. E layer',
    linestyle= 'dashed'
)
ax.plot(
    slab_lons,
    slab_depths,
    color='black',
    linewidth=2.5,
    label='JdF Slab'
)
```

```
plt.legend(loc='lower right')
ax.set_xlim(-124.15,-122.1)
plt.ylim(45,0)

plt.show()
```

[ ]:
```
#Ps reflection E layer travel times
print("Ps arrival times (s):")
for sta, tt in T_ps.items():
    print(f"{sta}: {tt}")
```

[ ]:
```
#Repeating E layer reflection - Profile B
#Defining profile B E layer

E_pb = [
    ((-125.25), 0.0),
    ((-124.58),  7.4),
    ((-124.34),  12.4),
    ((-124.14), 17.4),
    ((-123.55), 22.4),
    ((-123.38), 27.4),
    ((-123.21), 32.4),
    ((-123.09), 37.4),
    ((-122.56), 42.4)

]
e_lons_pb = []
e_depths_pb = []

for (lon), depth in E_pb:
    e_lons_pb.append(lon)
    e_depths_pb.append(depth)

e_lons_pb = np.array(e_lons_pb)
e_depths_pb = np.array(e_depths_pb)

from scipy.interpolate import interp1d

f = interp1d(e_lons_pb, e_depths_pb, kind='linear')

e_lons_dense_pb = np.linspace(min(e_lons_pb), max(e_lons_pb), 200)
e_depths_dense_pb = f(e_lons_dense_pb)

e_x_km_pb = lon_to_x(e_lons_dense_pb)
e_z_km_pb = e_depths_dense_pb
```

[ ]:
```
#Solving Pp reflection off profile B E layer
solver_obj.solve()
T_src = solver_obj.tt

grid_params = dict(
    npts=solver_obj.vv.npts,
    node_intervals=solver_obj.vv.node_intervals,
    min_coords=solver_obj.vv.min_coords,
    values=solver_obj.vv.values.copy(),
)

solvers_sta = {}
T_sta = {}
for sta, rx_coord in station_coords.items():
    s = solver.PointSourceSolver(coord_sys="cartesian")

    s.vv.npts = grid_params["npts"]
    s.vv.node_intervals = grid_params["node_intervals"]
    s.vv.min_coords = grid_params["min_coords"]
    s.vv.values = grid_params["values"]


    s.src_loc = rx_coord
    s.solve()

    solvers_sta[sta] = s
    T_sta[sta] = s.tt

slab_pts_epb  = np.column_stack([
    e_x_km_pb ,
    np.zeros_like(e_x_km_pb),
    e_z_km_pb
])

reflection_pts_epb = {}
T_reflect_epb  = {}


T_down_epb  = T_src.resample(slab_pts_epb)

for sta in station_coords:
    T_up_epb = T_sta[sta].resample(slab_pts_epb)

    T_tot = T_down_epb + T_up_epb
    idx = np.nanargmin(T_tot)

    reflection_pts_epb[sta] = slab_pts_epb[idx]
```

```
    T_reflect_epb[sta] = T_tot[idx]
```

[ ]:
```
#Plotting Pp reflection off E layer Profile B
vp_plot = solver_obj.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths


dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)

x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2

z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')
plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Pykonal Vp model - Reflected Pp Arrival - Profile B - Approx E␣
 ↪layer')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon

ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)

for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon
    ax.plot(rx_lon, rx[2], 'magenta',marker='^', ms=6)
ax.plot(rx_lon, rx[2], 'magenta',marker='^', ms=6,label='Stations')
for sta, s_sta in solvers_sta.items():
    ref_pt = reflection_pts_epb[sta]


    ray_down = solver_obj.trace_ray(ref_pt)
```

```
    ray_up = s_sta.trace_ray(ref_pt)[::-1]

    ray_down_lon = lon0 + ray_down[:, 0] / km_per_deg_lon
    ray_up_lon   = lon0 + ray_up[:, 0]   / km_per_deg_lon

    ax.plot(ray_down_lon, ray_down[:, 2], 'magenta', lw=0.8)
    ax.plot(ray_up_lon,   ray_up[:, 2],   'magenta', lw=0.8)


ax.plot(
    e_lons_pb,
    e_depths_pb,
    color='red',
    linewidth=2.5,
    label='Approx. E layer Profile B',
    linestyle= 'dashed'
)
ax.plot(
    slab_lons_pb,
    slab_depths_pb,
    color='black',
    linewidth=2.5,
    label='JdF Slab Profile B'
)
plt.legend(loc='lower right')

ax.set_xlim(-124.15,-122.1)
plt.ylim(45,0)
plt.show()
```

[ ]:
```
#Travel times Profile B E layer reflection
print("Reflected ray travel times - E layer Profile B:")
ref_tt_epb = []
for sta, ref_pt in reflection_pts_epb.items():

    T_down = T_src.resample(ref_pt.reshape(1,3))[0]

    T_up = T_sta[sta].resample(ref_pt.reshape(1,3))[0]

    T_total = T_down + T_up

    print(f"{sta}: tt = {T_total} s")
    ref_tt_epb.append(T_total)
```

[ ]:
```
#Repeating for Ps reflection off the E layer Profile B
solvers_sta_vs = {}
T_sta_vs = {}


for sta, rx_coord in station_coords.items():
    s_vs = solver.PointSourceSolver(coord_sys="cartesian")



    s_vs.vv.npts = solver_obj_vs.vv.npts
    s_vs.vv.node_intervals = solver_obj_vs.vv.node_intervals
    s_vs.vv.min_coords = solver_obj_vs.vv.min_coords
    s_vs.vv.values = solver_obj_vs.vv.values.copy()


    s_vs.src_loc = rx_coord
    s_vs.solve()


    solvers_sta_vs[sta] = s_vs
    T_sta_vs[sta] = s_vs.tt
T_ps = {}
T_P_down = solver_obj.tt.resample(slab_pts_epb)


for sta in station_coords:
    T_S_up = T_sta_vs[sta].resample(slab_pts_epb)


    T_tot = T_P_down + T_S_up
    idx = np.nanargmin(T_tot)


    reflection_pts[sta] = slab_pts_epb[idx]
    T_ps[sta] = T_tot[idx]
```

[ ]:
```
#Plotting Profile B Ps E layer reflection
vp_plot = solver_obj.vv.values[:, ny//2, :].T
x_plot_lon = x_to_lon(longs)
z_plot = depths

dx = np.mean(np.diff(x_plot_lon))
dz = np.mean(np.diff(depths))
x_centers = x_plot_lon
x_edges = np.zeros(len(x_centers)+1)
```

```
x_edges[1:-1] = (x_centers[:-1] + x_centers[1:]) / 2
x_edges[0] = x_centers[0] - (x_centers[1]-x_centers[0]) / 2
x_edges[-1] = x_centers[-1] + (x_centers[-1]-x_centers[-2]) / 2

z_edges = np.arange(0, depths.max() + dz + 1e-6, dz)
fig, ax = plt.subplots(figsize=(10,5))
pcm = ax.pcolormesh(x_edges, z_edges, vp_plot,shading='flat')
plt.colorbar(pcm, ax=ax, label='Vp (km/s)')
ax.set_xlabel('Longitude (°)')
ax.set_ylabel('Depth (km)')
ax.invert_yaxis()

plt.title('2D Pykonal Vp Model - Reflected Ps Arrival - Profile A - Approx E␣
 ↪Layer')
src_x_km = solver_obj.src_loc[0]
src_lon = lon0 + src_x_km / km_per_deg_lon

ax.plot(
    src_lon,
    solver_obj.src_loc[2],
    'r*',
    ms=14,
    label='Source',markeredgecolor='k'
)
for rx in station_coords.values():
    rx_lon = lon0 + rx[0] / km_per_deg_lon
    ax.plot(rx_lon, rx[2], 'magenta',marker="^", ms=6)
ax.plot(rx_lon, rx[2], 'magenta',marker="^", ms=6,label="Stations")
for sta, s_sta in solvers_sta.items():
    ref_pt = reflection_pts[sta]


    ray_down = solver_obj.trace_ray(ref_pt)

    ray_up = solvers_sta_vs[sta].trace_ray(ref_pt)[::-1]

    ray_down_lon = lon0 + ray_down[:, 0] / km_per_deg_lon
    ray_up_lon   = lon0 + ray_up[:, 0]   / km_per_deg_lon

    ax.plot(ray_down_lon, ray_down[:, 2], 'magenta', lw=0.8)
    ax.plot(ray_up_lon,   ray_up[:, 2],   'magenta', lw=0.8)


ax.plot(
    e_lons_pb,
    e_depths_pb,
    color='red',
```

```
    linewidth=2.5,
    label='Approx. E layer Profile B',
    linestyle= 'dashed'
)
ax.plot(
    slab_lons_pb,
    slab_depths_pb,
    color='black',
    linewidth=2.5,
    label='JdF Slab Profile B'
)
plt.legend(loc='lower right')
ax.set_xlim(-124.15,-122.1)
plt.ylim(45,0)

plt.show()
```

[ ]:
```
#Ps E layer reflection travel times - profile B
print("Ps arrival times (s):")
for sta, tt in T_ps.items():
    print(f"{tt}")
```