pynasonde — Methods & API Reference

Technical guide for ionospheric researchers

pynasonde Development Team

Version 1.0 — March 28, 2026

Contents

Abstract

pynasonde is an open-source Python toolkit for reading, analysing, and visualising ionosonde data from both **VIPIR** and **DIGISONDE DPS4D** instruments. This document gives the mathematical foundations for each analysis algorithm, the design rationale behind key implementation choices, and a compact API cheat-sheet. Target audience: researchers who want to understand the algorithms, extend the toolkit, or apply it to new data sets.

Published reference: Chakraborty S., Bullett T., Barjatya A., Mabie J. (2026). pynasonde: An open-source Python library for ionosonde data processing. *SoftwareX* 34, 102617. `https://doi.org/10.1016/j.softx.2026.102617`

1 Architecture Overview

1.1 Published v1.0 architecture

The library is organised into two instrument modules (`digisonde` and `vipir`) each with dedicated parsers and plotting routines, plus a new cross-instrument analysis layer added after publication.

Figure 2 of the Softwa shows the two-module (digisonde, vipir) with colour-coded data type plotting methods. The layer described here is new post-publication.

1.2 Complete file-format and class index

Table 1: All supported file formats, parser classes, and primary methods. **[new]** = added after SoftwareX publication.

Extension	Type	Class	Primary methods / output
DIGISONDE DPS4D			
`.SAO 4.3`	ASCII	`SaoExtractor`	`load_SAO_files()` with `height_profile` or `scaled_params`
`.SAO 5.0`	XML	`SaoExtractor`	same API; `SaoSummaryPlots.plot_TS()`
`.RSF`	Binary	`RsfExtractor`	`plot_direction_ionogram()`, `plot_directogram()` **[new]**
`.SBF`	Binary	`SbfExtractor`	`extract()`, `ionogram()`
`.DFT`	ASCII	`DftExtractor`	`plot_doppler_waterfall()`, `plot_doppler_spectra()` **[new]**
`.DVL`	ASCII	`DvlExtractor`	`load_DVL_files()`, `plot_dvl_drift_velocities()`
`.SKY`	ASCII	`SkyExtractor`	`extract()`, `to_pandas()`, multi-panel `plot_skymap()`
`.EDP`	ASCII	`EdpExtractor`	`extract()`, electron density profile
VIPIR			
`.RIQ`	Binary	`RiqDataset`	`create_from_file()`, SCT + PCT, `VIPIR_VERSION_MAP`
`.RIQ`	—	`EchoExtractor` **[new]**	7-param Dynasonde echoes (h, A, V*, EP, PP, XL, YL)
`.RIQ`	—	`IonogramFilter` **[new]**	6-stage filter: RFI, EP, multi-hop, DBSCAN, RANSAC, temporal
`.NGI`	NetCDF	`DataSource`	`load()`, multi-file data cube aggregation
`.NGI`	—	`AutoScaler`	noise profiling, segmentation, binary trace extraction
Analysis layer (all new post-v1.0)			
—	—	`PolarizationClassifier`	O/X mode separation via PP; hemisphere-aware
—	—	`TrueHeightInversion`	Titheridge/POLAN lamination; `EDPResult`
—	—	`AbsorptionAnalyzer`	LOF, O/X differential, calibrated, $\kappa(z)$
—	—	`SpreadFAnalyzer`	Range / frequency / mixed spread-F classification
—	—	`IrregularityAnalyzer`	EP structure function $\rightarrow \alpha(h)$, outer scale, anisotropy
—	—	`IonogramScaler`	foF2, foE, h'F, MUF from O-mode trace
—	—	`EsCaponImager`	Capon cross-spectrum, $\Delta r = r_0/K$
—	—	`RiqAggregator`	Multi-file A+B+C, $\sim$24 dB SNR gain
—	—	`NeXtYZInverter`	WSI model + Hamiltonian ray-tracing, 3-D $N(h, x, y)$
—	—	`FullPipeline`	End-to-end RIQ $\rightarrow$ N(h) automation

2 Instrument Overview

2.1 VIPIR

VIPIR (Vertical Incidence Pulsed Ionospheric Radar) records raw in-phase and quadrature (IQ) samples in binary `.RIQ` files. Each file contains a Sounding Control Table (SCT) followed by one Pulse Configuration Table (PCT) record per transmitted pulse. The IQ cube for one sounding frequency has shape (pulse_count, gate_count, rx_count).

Listing 1: VIPIR — load a RIQ file

```
from pynasonde.vipir.riq.parsers.read_riq import RiqDataset,
    VIPIR_VERSION_MAP

riq = RiqDataset.create_from_file(
    "data/WI937_20220821.RIQ",
    vipir_config=VIPIR_VERSION_MAP.configs[1],  # data_type=1 (older
        format)
)
sct = riq.sct    # SoundingControlTable
# Key fields: sct.timing.gate_count, sct.timing.gate_step (us),
#             sct.station.rx_count, sct.station.rx_position
```

WI937 (Wallops Islan pulses/freq, 960 gates ($r_0 = 1.499$ km), 8 Rx PL407 (Prudhoe Bay) format, data_type=2.

2.2 DIGISONDE DPS4D

The Digisonde DPS4D family stores data in several formats:

Extension	Class	Contents
`.SAO`	`SaoExtractor`	Scaled parameters (foF2, hmF2, MUF) + N(h) profile
`.RSF`	`RsfExtractor`	Raw amplitude + phase per frequency/height/azimuth
`.SBF`	`SbfExtractor`	Binary raw sounding (alternative to RSF)
`.DFT`	`DftExtractor`	Per-height Doppler spectra (velocity vs. height)
`.DVL`	`DvlExtractor`	Drift velocity estimates (Vx, Vy, Vz)
`.SKY`	`SkyExtractor`	Sky map (polar arrival-direction map)

SAO is the primary p most science applicati RSF/SBF give access amplitude and phase. contains the full Dopp per height.

Listing 2: DIGISONDE — load SAO files in parallel

```
from pynasonde.digisonde.parsers.sao import SaoExtractor

df = SaoExtractor.load_SAO_files(
    folders=["data/KR835/"],
    func_name="height_profile",   # or "scaled_params"
    n_procs=4,
)
# df columns: datetime, height_km, plasma_freq_mhz,
    electron_density_cm3, ...
```

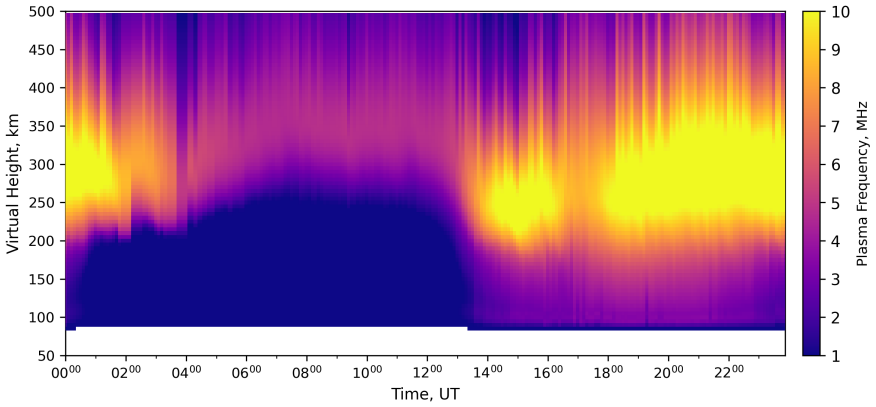


Figure 1: Daily isodensity contour from DPS4D SAO files at KR835. Each column is one N(h) profile; colour encodes electron density (cm^{-3}, log scale).

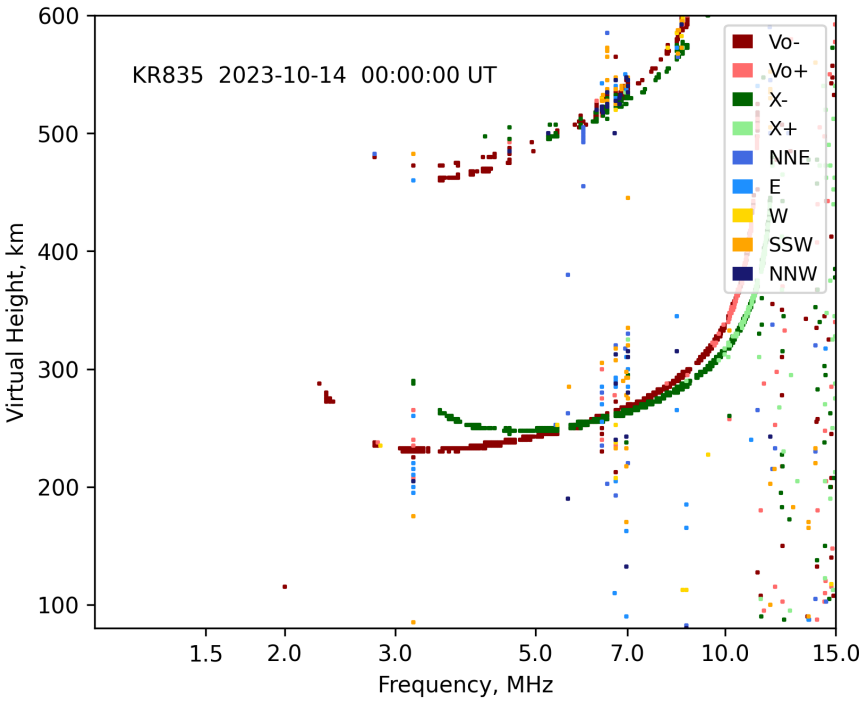


Figure 2: Direction-coded ionogram from a DPS4D RSF sounding at KR835. Colour encodes the echo arrival azimuth.

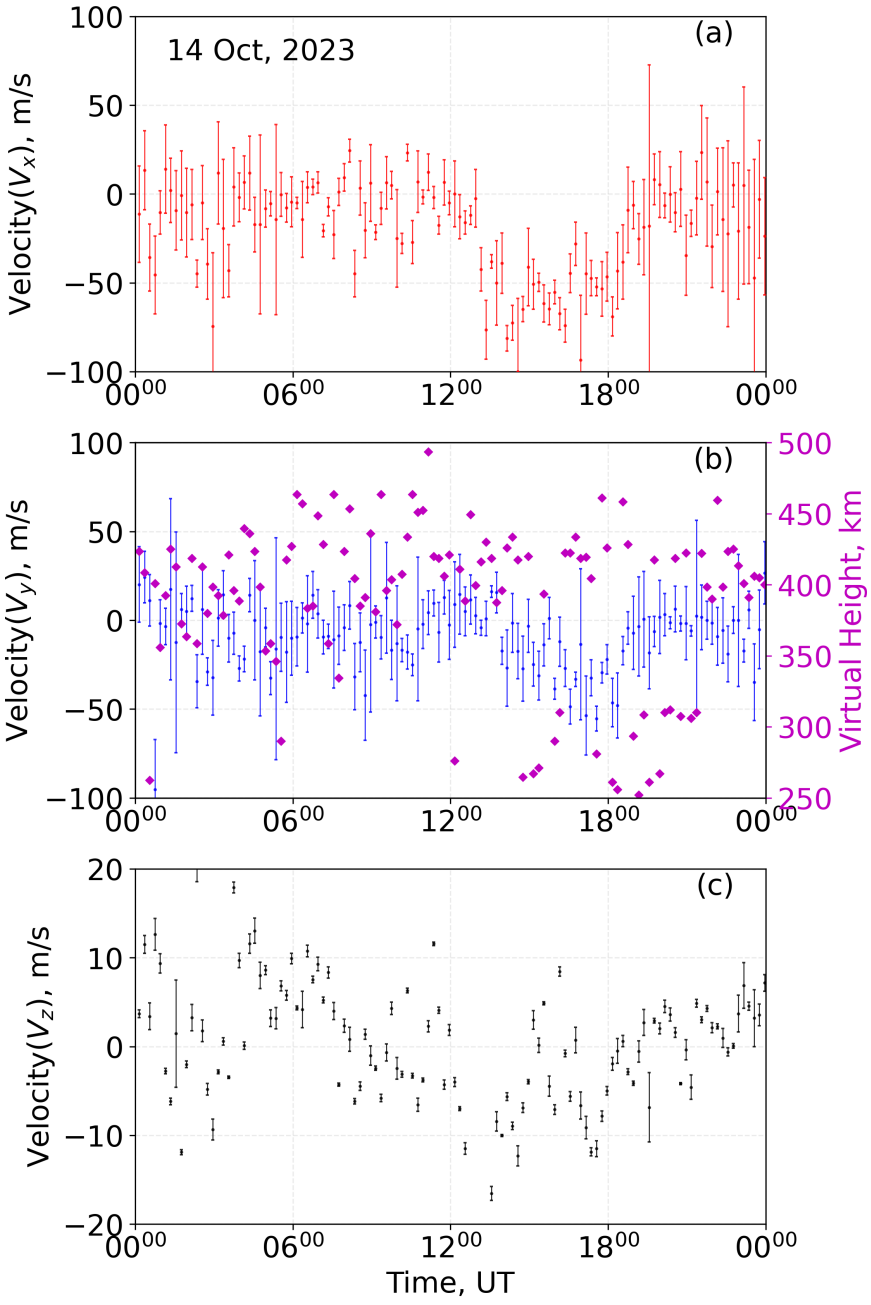


Figure 3: Three-panel drift velocity stack plot from DPS4D DVL files. Panels: V_x (east), V_y (north), V_z (vertical), with virtual-height overlay.

3 VIPIR Echo Extraction & Filtering

3.1 Seven-parameter echo extraction

`EchoExtractor` computes the seven Dynasonde-style echo parameters — height, amplitude, Doppler velocity V*, wavefront residual EP, polarization chirality PP, and direction cosines XL/YL — from the pulse-compressed IQ cube.

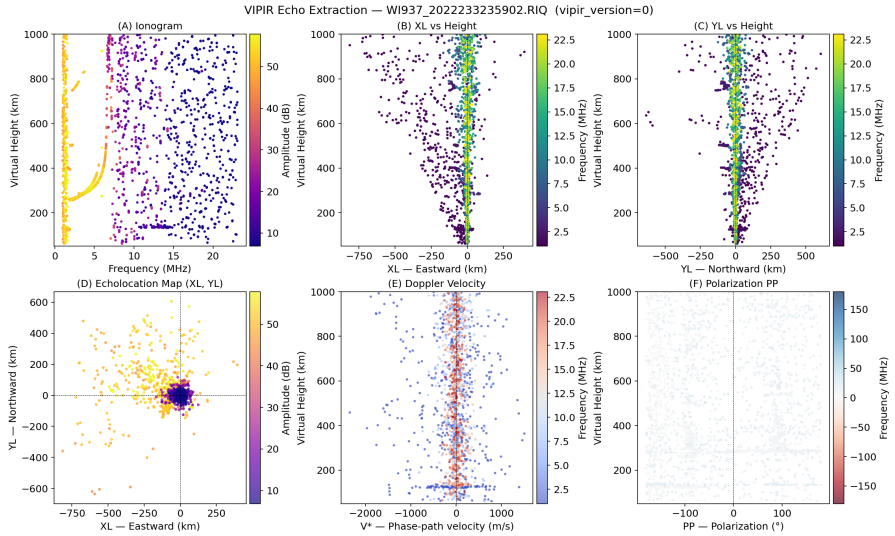


Figure 4: Six-panel diagnostic for WI937 echo extraction: ionogram (A), XL (B), YL (C), echo location map (D), Doppler V* (E), PP (F).

Listing 3: Echo extraction and filtering

```
from pynasonde.vipir.riq.echo import EchoExtractor
from pynasonde.vipir.riq.parsers.filter import IonogramFilter

extractor = EchoExtractor(sct=riq.sct, pulsets=riq.pulsets,
                          snr_threshold_db=3.0).extract()
df = extractor.to_dataframe()

filtered = IonogramFilter(snr_threshold_db=6.0, ep_threshold_deg=30.0,
                          use_dbscan=True, use_ransac=True).fit(df)
```

4 Capon Cross-Spectrum Es Layer Imaging

4.1 Physical motivation

Sporadic-E (Es) layers are thin metallic-ion sheets at 90–130 km altitude. Their vertical thickness (< 1 km) is below the native range resolution of most ionosondes ($r_0 = 1.5$–4 km).

The pulse-compressed range profile $R_{ss}(t)$ is a complex sequence of V samples. Its FFT (the *cross-power spectrum*) is

Liu et al. (2023) demonstrated 384 m resolution from a 3.84 km native gate using 256 pulses from the WISS digisonde.

$$G_{ss}(f_m) = \mathrm{FFT}\{R_{ss}\}(f_m) = U(f_m)\, e^{j4\pi r f_m/c}, \tag{1}$$

where $U(f) = |S(f)|^2$ encodes signal power and the exponential encodes range r.

4.2 Hankel subband matrix and covariance

Partition G_{ss} into a Hankel matrix $\mathbf{G}$ of shape $(Z, V - Z + 1)$:

$$G[i, j] = G_{ss}[f_{i+j}], \quad i = 0 \ldots Z - 1,\ j = 0 \ldots V - Z. \tag{2}$$

The spatial covariance with diagonal loading ε:

$$\mathbf{R}_f = \frac{\mathbf{G}\mathbf{G}^H}{V - Z + 1} + \varepsilon\, \frac{\mathrm{tr}(\mathbf{R}_f)}{Z}\, \mathbf{I}_Z. \tag{3}$$

4.3 Capon pseudospectrum

Steering vector $\mathbf{a}(\omega_\ell) \in \mathbb{C}^Z$ with $a_k(\omega_\ell) = e^{jk\omega_\ell}$ and $\omega_\ell = 2\pi\ell/(KV)$:

$$\boxed{P\left(\frac{\ell r_0}{K}\right) = \frac{1}{\mathbf{a}^H(\omega_\ell)\,\mathbf{R}_f^{-1}\,\mathbf{a}(\omega_\ell)}}, \quad \ell = 1, \ldots, KV. \tag{4}$$

Effective range resolution: $\Delta r = r_0/K$.

4.4 Singularity constraints

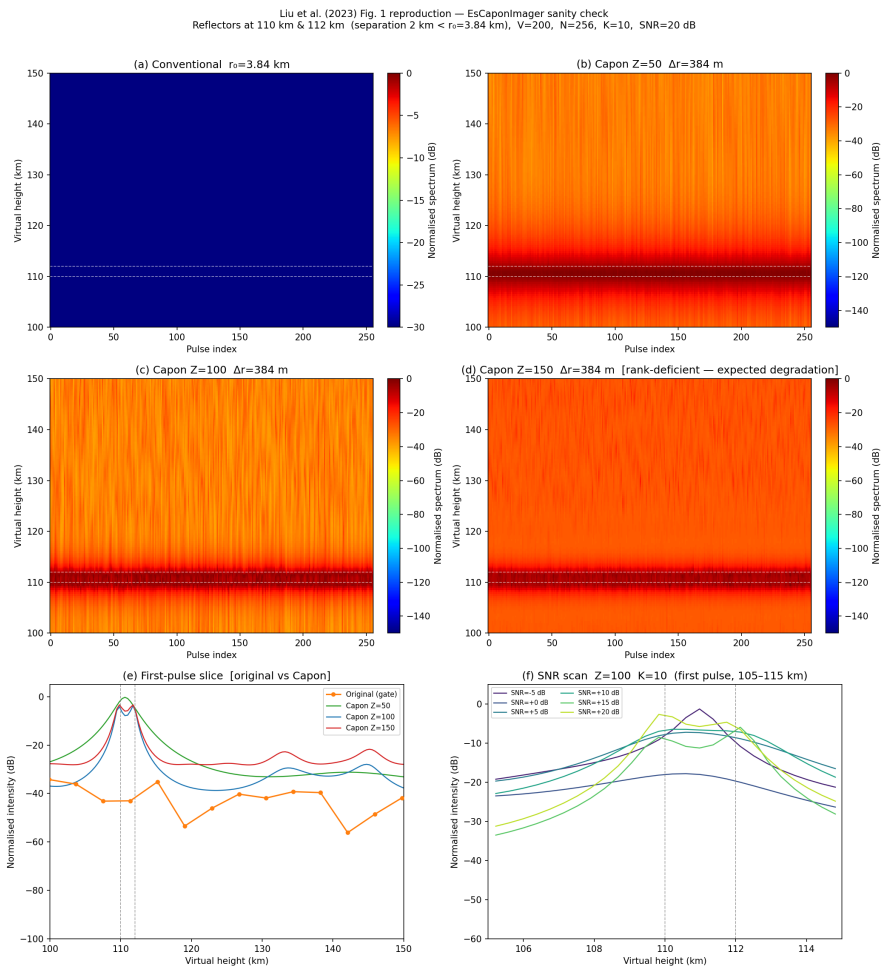
$\mathbf{R}_f$ has shape (Z, Z) and rank at most $V - Z + 1$. For full rank:

$$V - Z + 1 \geq Z \quad \Longrightarrow \quad Z \leq \frac{V+1}{2}. \tag{5}$$

Parameter roles:

- Z *Covariance size; affects rank of* $\mathbf{R}_f$
- K *Output grid only; does* ***not*** *enter* $\mathbf{R}_f$

K has no singularity constraint. Earlier implementations incorrectly clipped $K \leq \lfloor V/Z \rfloor$; pynasonde ≥1.0 removes this constraint entirely.



Liu et al. Fig. 1: $Z = 50$ (unresolved), $Z = 100$ (resolved, optimal), $Z = 150$ ($> (V+1)/2$, rank-deficient, partially recovered via diagonal loading).

Figure 5: Synthetic sanity check reproducing Liu et al. (2023) Fig. 1. Two layers at 110 and 112 km ($r_0 = 3.84$ km, $V = 200$, $K = 10$). (a) Conventional; (b) $Z = 50$; (c) $Z = 100$; (d) $Z = 150$; (e) first pulse; (f) SNR scan.

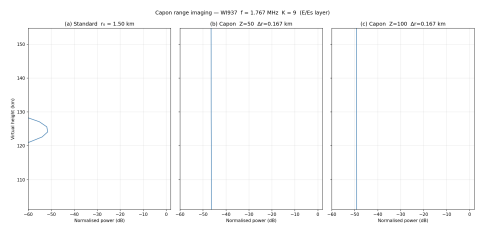


Figure 6: Single-file Es Capon image (1-D profile).

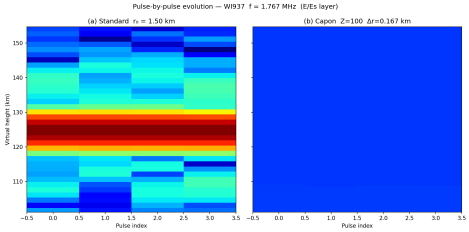


Figure 7: Slow RTI from `RiqAggregator` (A+B+C, 8 files).

Listing 4: EsCaponImager — single file

```
from pynasonde.vipir.analysis import EsCaponImager
imager = EsCaponImager(n_subbands=100, resolution_factor=10,
                       gate_spacing_km=1.499, gate_start_km=90.0)
result = imager.fit(iq_cube)
```

5 Multi-File A+B+C Combining (RiqAggregator)

5.1 Combining strategy

Option A — Coherent Rx beamforming:

$$R_{\text{beam}}[p,g] = \mathbf{S}[p,g,:]\,\mathbf{w}^*, \quad \Delta\text{SNR} = 10\log_{10}(n_{rx})\ \text{dB}. \tag{6}$$

Calibrated Rx weights from the antenna position table (SCT header `station.rx_position`) will be supported once the RIQ header reader exposes them.

Option B — Incoherent pulse averaging (per file):

$$P_{\text{file}} = \frac{1}{n_p}\sum_{p} P_{\text{Capon}}(R_{\text{beam}}[p,:]). \tag{7}$$

Option C — Incoherent file stacking:

$$P_{\text{final}} = \frac{1}{n_f}\sum_{f} P_{\text{file},f}. \tag{8}$$

SNR budget (8 files × 8 Rx):

Step	
Opt. A (8 Rx)	
Opt. B (4p avg)	
Opt. C (8f stack)	
Total	≈

Listing 5: RiqAggregator — multi-file A+B+C

```
from pynasonde.vipir.analysis import RiqAggregator
agg = RiqAggregator(n_subbands=100, resolution_factor=10,
                    output_mode="slow_rti")  # or "single"
result = agg.fit(file_list, freq_target_khz=5000.0, freq_tol_khz=50.0)
```

6 HF Radio Absorption

No-field, weak-collision Appleton-Hartree absorption rate (dB km^{-1}):

$$\kappa(h) = \frac{4.343\,\nu(h)\,X(h)}{c_{\text{km}}\sqrt{1-X(h)}}, \qquad X = (f_p/f)^2. \tag{9}$$

LOF index: $A = f_{\min}^2 - f_{\text{ref}}^2$ [MHz^2]. Differential O/X: $\Delta L(f) = \text{SNR}_O - \text{SNR}_X$ [dB].

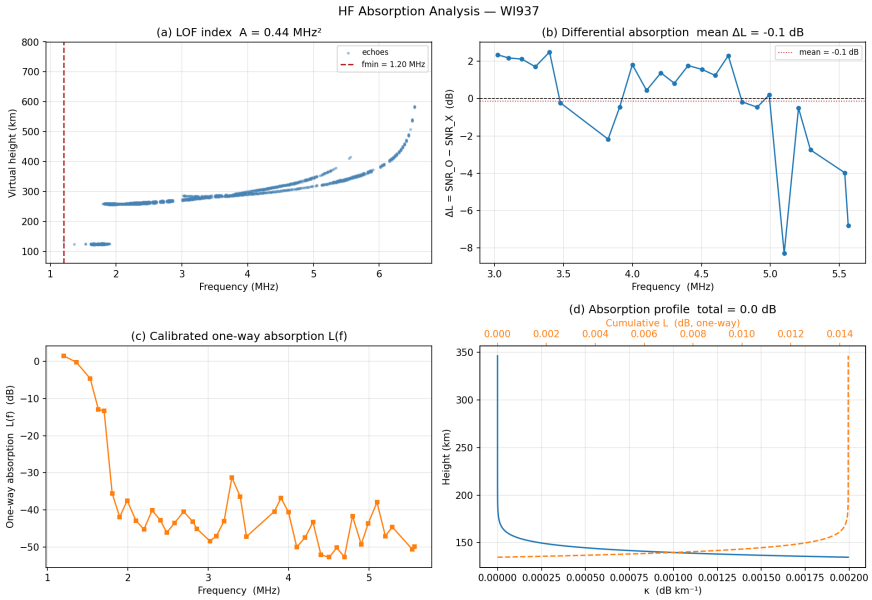


Figure 8: Absorption analysis panels: (a) LOF index, (b) differential O/X $\Delta L(f)$, (c) total calibrated $L(f)$, (d) height-resolved $\kappa(z)$.

Listing 6: AbsorptionAnalyzer — all four estimators

```
from pynasonde.vipir.analysis import AbsorptionAnalyzer
import numpy as np

aa = AbsorptionAnalyzer(f_ref_mhz=1.0)
lof  = aa.lof_absorption(filtered_df)                          # LOF
diff = aa.differential_absorption(pol.annotated_df)           # O/X
total= aa.total_absorption(filtered_df, tx_eirp_dbw=70.0)      #
    calibrated
prof = aa.absorption_profile(edp, nu_hz=lambda h: 1e6*np.exp(-h/8.0))
     # k(z)
```

7 True Height Inversion

The Titheridge (1967) lamination method discretises the Abel integral:

$$h_n = h'(f_n) - \sum_{j=1}^{n-1} \big(\mu'(f_n, f_j) - 1\big)\, \Delta h_j, \quad \mu'(f, f_p) = \frac{1}{\sqrt{1 - (f_p/f)^2}}. \tag{10}$$

Stability caveat: adjac steps < 0.1 MHz caus near-singular μ' contr pynasonde bins to ~ 0 before applying eq. (10 prescription).

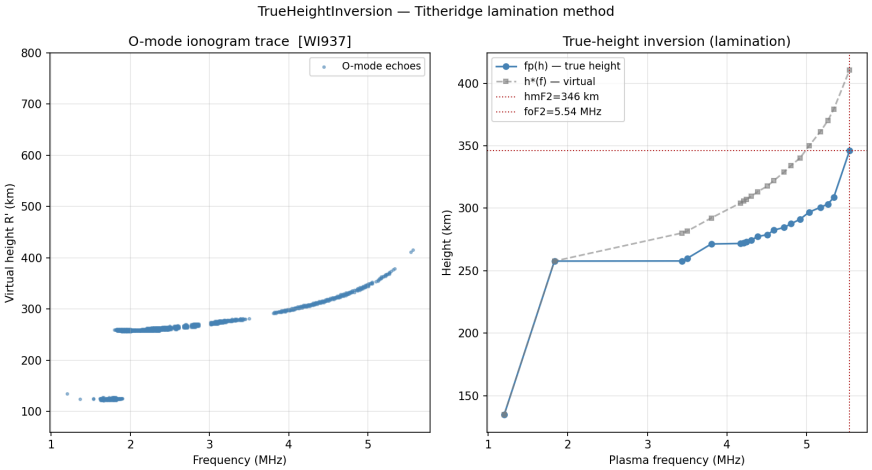


Figure 9: True height inversion output: N(h) electron density profile (left) and virtual vs. true height comparison (right).

Listing 7: TrueHeightInversion API

```
from pynasonde.vipir.analysis import TrueHeightInversion
inv = TrueHeightInversion(monotone_enforce=True, bin_width_mhz=0.05)
edp = inv.fit_from_df(pol.annotated_df)  # O-mode filter applied
    automatically
print(edp.summary())   # foF2=8.12 MHz  hmF2=287.4 km  NmF2=8.2e5 cm
    ^-3
```

8 Polarization & Spread-F

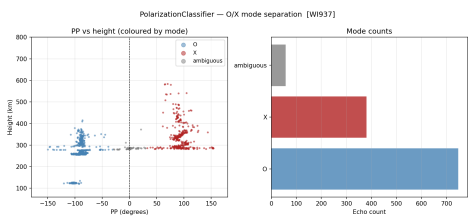


Figure 10: PP histogram with O/X classification regions.

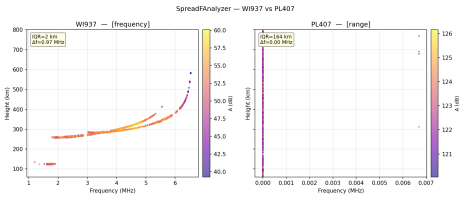


Figure 11: Spread-F detection: range/frequency/mixed type classification.

Listing 8: Polarization and Spread-F

```
from pynasonde.vipir.analysis import PolarizationClassifier,
    SpreadFAnalyzer

pol = PolarizationClassifier(o_mode_sign=-1).fit(filtered_df)
sf  = SpreadFAnalyzer().fit(filtered_df)
print(sf.summary())   # type=range  foEs=4.2 MHz  h_spread_km=38.1
```

9 Ionospheric Irregularity Analysis

9.1 Physical motivation

The EP (residual phase) parameter from the Dynasonde signal model carries information about sub-wavelength irregularities in the ionospheric reflection

layer. Irregularities imprint a structured signature on EP as a function of sounding frequency f.

EP is the wavefront r
phase: the part of the
not explained by the n
group-range and Dopp
Random phase perturb
evidence of small-scale
irregularities.

9.2 Structure function approach

The second-order structure function of EP as a function of frequency lag Δf:

$$D_{EP}(\Delta f) = \left\langle [EP(f + \Delta f) - EP(f)]^2 \right\rangle. \tag{11}$$

For a power-law irregularity spectrum with spectral index α:

$$D_{EP}(\Delta f) \propto \Delta f^{\alpha}. \tag{12}$$

A log-log fit of D_{EP} vs Δf yields α and amplitude A_0. The outer scale L_{outer} is estimated as the lag at which D_{EP} saturates.

9.3 Height-resolved and anisotropy analysis

Echoes are binned by virtual height, and the fit is computed independently per bin, giving a profile $\alpha(h)$. When both O-mode and X-mode EP are available:

$$\text{anisotropy proxy} = \frac{\sigma_{EP}(O)}{\sigma_{EP}(X)}. \tag{13}$$

A ratio near unity indicates isotropic scattering; deviations indicate field-aligned anisotropy.

Listing 9: IrregularityAnalyzer API

```
from pynasonde.vipir.analysis import IrregularityAnalyzer

irr = IrregularityAnalyzer(
    height_bin_km=10.0,       # virtual height bin width
    min_echoes_per_bin=5,     # minimum echoes for a valid fit
    lag_max=2.0,              # maximum frequency lag (MHz) for
        structure function
)
result = irr.fit(filtered_df)
# result.profile_df: columns = height_km, alpha, A0, outer_scale_km,
    anisotropy
# result.alpha_profile:   1-D alpha(h) array
# result.outer_scale_km:  1-D L_outer(h) array
result.plot_spectral_profile()   # alpha(h) + outer scale overlay
result.plot_structure_function(height_km=200.0)  # D_EP vs lag at one
    height
```

10 NeXtYZ 3-D Electron Density Inversion

10.1 Wedge-Stratified Ionosphere model

The NeXtYZ algorithm (Zabotin et al. 2006) represents the local electron density as a stack of plasma-frequency wedges bounded above by *frame planes*. Each wedge boundary is characterised by:

$$(h_{i+1},\ n_{x,i+1},\ n_{y,i+1}), \tag{14}$$

where h_{i+1} is the reflection height and (n_x, n_y) are the horizontal components of the outward normal — i.e., the ionospheric tilt.

NeXtYZ Lite solves o
wedge using constant
from mean angles of
≈ 6× faster than Full
sufficient for moderate

10.2 Hamiltonian eikonal ray tracing

Signal propagation through the WSI model is computed via the eikonal (method of characteristics) equations:

$$\dot{\mathbf{r}} = \partial H/\partial \mathbf{k}, \tag{15}$$

$$\dot{\mathbf{k}} = -\partial H/\partial \mathbf{r}, \tag{16}$$

with Hamiltonian $H = |\mathbf{k}|^2 - n^2(\mathbf{r}, f)$ and the full Appleton-Lassen refractive index n.

10.3 Inversion loop

Wedge parameters are solved bottom-up by alternately minimising:

1. **Group-range residual**: $\Delta R'_{i+1} = \sqrt{\sum_j (\rho'_{i+1,j} - R'_{i+1,j})^2}$
2. **Ground-return distance** of the mean-direction ray (tilt constraint).

Listing 10: NeXtYZInverter API

```
from pynasonde.vipir.analysis import NeXtYZInverter, NeXtYZResult

inv3d = NeXtYZInverter(
    mode="lite",           # "lite" (fast, const tilts) or "full" (per-
        wedge tilts)
    n_wedge_max=30,        # maximum number of WSI wedge layers
    ray_step_km=0.5,       # Hamiltonian integrator step size
    tilt_regularisation=0.01,  # weight of tilt constraint vs group-
        range fit
)
result: NeXtYZResult = inv3d.fit(filtered_df)

# Key output attributes
print(result.fp_profile)     # plasma frequency vs true height
print(result.tilt_x)         # E-W tilt per wedge (degrees)
print(result.tilt_y)         # N-S tilt per wedge (degrees)
print(result.height_error_km)  # estimated layer height uncertainty

result.plot_wedge_planes()  # 3-D visualisation of WSI frame planes
result.plot_fp_profile()    # fp(h) vs true height
```

11 API Quick-Reference

11.1 All public imports

```
# DIGISONDE I/O
from pynasonde.digisonde.parsers.sao import SaoExtractor,
    SaoSummaryPlots
from pynasonde.digisonde.parsers.rsf import RsfExtractor
from pynasonde.digisonde.parsers.sbf import SbfExtractor
from pynasonde.digisonde.parsers.dft import DftExtractor
from pynasonde.digisonde.parsers.dvl import DvlExtractor
from pynasonde.digisonde.parsers.sky import SkyExtractor,
    SkySummaryPlots
from pynasonde.digisonde.parsers.edp import EdpExtractor

# VIPIR I/O
from pynasonde.vipir.riq.parsers.read_riq import RiqDataset,
    VIPIR_VERSION_MAP
from pynasonde.vipir.riq.echo import EchoExtractor
from pynasonde.vipir.riq.parsers.filter import IonogramFilter
from pynasonde.vipir.ngi.source import DataSource
from pynasonde.vipir.ngi.scale import AutoScaler
```

```python
# Analysis layer
from pynasonde.vipir.analysis import (
    PolarizationClassifier, PolarizationResult,
    TrueHeightInversion, EDPResult,
    AbsorptionAnalyzer, LOFResult, DifferentialResult,
    TotalAbsorptionResult, AbsorptionProfileResult,
    SpreadFAnalyzer, SpreadFResult,
    IrregularityAnalyzer, IrregularityProfile,
    IonogramScaler, ScaledParameters,
    EsCaponImager, RiqAggregator, EsImagingResult,
    NeXtYZInverter, NeXtYZResult, WedgePlane,
)
```

11.2 Constructor summary

Table 2: Key constructor parameters for all analysis classes.

Class	Parameter	Default	Notes
`EsCaponImager`	`n_subbands`	100	Z; recommended $\leq (V+1)/2$
	`resolution_factor`	10	K; no singularity limit
	`gate_spacing_km`	3.84	r_0; VIPIR WI937 = 1.499 km
	`diagonal_loading`	1e-3	ε for $\mathbf{R}_f$ conditioning
`RiqAggregator`	`rx_weights`	None	None = uniform $1/n_{rx}$
	`output_mode`	single	or `slow_rti`
	`gate_spacing_km`	1.499	overridden from RIQ header
`TrueHeightInversion`	`method`	lamination	only supported method
	`bin_width_mhz`	0.05	trace decimation for stability
	`monotone_enforce`	True	remove non-monotone layers
`AbsorptionAnalyzer`	`f_ref_mhz`	1.0	quiet-day reference for LOF
`PolarizationClassifier`	`o_mode_sign`	-1	$+1$ for southern hemisphere
`IrregularityAnalyzer`	`height_bin_km`	10.0	virtual height bin width
	`min_echoes_per_bin`	5	minimum for valid power-law fit
	`lag_max`	2.0 MHz	max frequency lag for D_{EP}
`NeXtYZInverter`	`mode`	`lite`	`lite` (fast) or `full` (per-wedge tilts)
	`n_wedge_max`	30	maximum WSI wedge layers
	`ray_step_km`	0.5	Hamiltonian integrator step size
	`tilt_regularisation`	0.01	tilt constraint weight

11.3 Result dataclasses at a glance

Table 3: Key attributes of result objects.

Class	Key attributes	Shape

`EsImagingResult`	`pseudospectrum_db`
	`heights_km`
	`effective_resolution_km`
		(n_s, KV)
		$(KV,)$
		scalar

`EDPResult`	`true_height_km`, `plasma_freq_mhz` `foF2_mhz`, `hmF2_km`, `NmF2_cm3`	$(n_L,)$ scalar
`AbsorptionProfileResult`	`profile_df` (h, ν, f_p, X, κ) `total_absorption_db`	DataFrame scalar
`PolarizationResult`	`annotated_df` (`mode` column) `o_mode_count`, `x_mode_count`	DataFrame int

References

- Liu, T., Yang, G., & Jiang, C. (2023). High-resolution sporadic E layer observation. *Space Weather*, 21, e2022SW003195. `https://doi.org/10.1029/2022SW003195`
- Titheridge, J. E. (1967). *JATP*, 29, 763–778.
- Paul, A. K. (1975). POLAN. *NOAA Technical Report ERL 324-SEL 31*.
- Zabotin, N. A., Wright, J. W., & Zhbankov, G. A. (2006). NeXtYZ. *Radio Science*, 41, RS6S32.
- Davies, K. (1990). *Ionospheric Radio*. Peter Peregrinus.
- Budden, K. G. (1985). *The Propagation of Radio Waves*. Cambridge.
- Grubb, R. N., et al. (2011). New science with VIPIR. *Radio Science*.
- Reinisch, B. W., et al. (1997). The Digisonde 4D. *JASTP*, 59, 2181.