pynasonde — Methods & Capabilities Overview

An open-source Python library for ionosonde data processing

Chakraborty et al., *SoftwareX* 34, 102617 (2026)

S. Chakraborty T. Bullett A. Barjatya J. Mabie

Embry-Riddle Aeronautical University — Fourth State Communications

March 31, 2026

`pip install pynasonde` — github.com/shibaji7/pynasonde — pynasonde.readthedocs.io

Outline

Why pynasonde?

- **Problem**: Digisonde tools (Java) and VIPIR tools are instrument-specific, fragmented, and not always publicly available
- **Solution**: A unified, open-source Python framework for *both* instrument families
- **Key design goals**:
  - Modular architecture (read → plot → analyse)
  - Support 9 file formats (binary + ASCII + XML)
  - Lower barrier to reproducible analysis
  - Built-in analysis pipeline, not just plotting
- **Published**: Chakraborty et al., *SoftwareX* 34, 102617 (2026)
- **Current**: v1.0 — substantially extended beyond publication

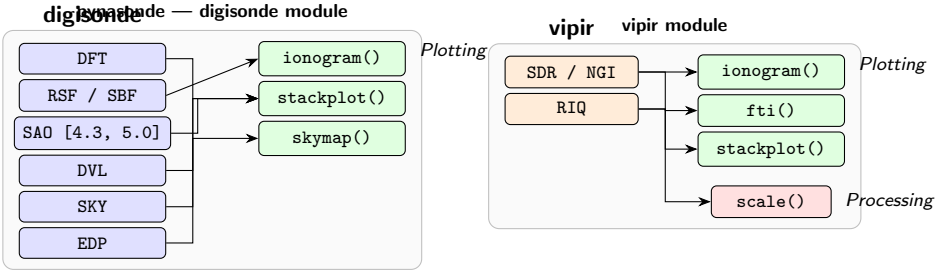
Installed capabilities

9 file formats read
7 digisonde parsers
6 VIPIR I/O classes
9 analysis modules
20+ public API classes
Full pipeline automation

Cite as

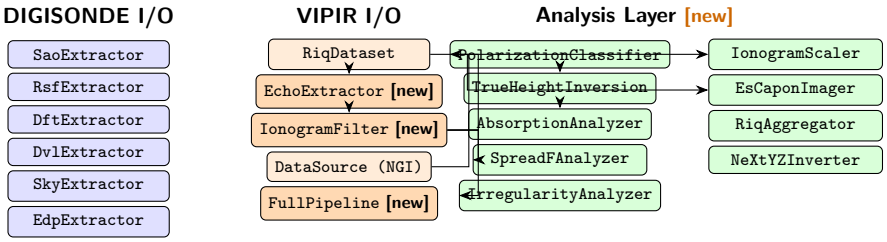
Chakraborty et al. (2026).
SoftwareX 34, 102617.
DOI: 10.1016/j.softx.2026.102617
Zenodo: 10.5281/zenodo.17476077

pynasonde v1.0 Architecture (Chakraborty et al. 2026)



Adapted from Fig. 2 of Chakraborty et al. (2026), *SoftwareX* 34, 102617.

Extended Architecture — Post-v1.0 Additions



Orange = new since publication. Green = analysis layer (entirely new post-v1.0).

DIGISONDE Module — File Formats & Parsers

Ext.	Type	Parser class	Key methods / output
.SAO 4.3	ASCII	SaoExtractor	load_SAO_files(): height_profile / scaled_params
.SAO 5.0	XML	SaoExtractor	same API; SaoSummaryPlots.plot_TS()
.RSF	Binary	RsfExtractor	plot_direction_ionogram(), plot_directogram()
.SBF	Binary	SbfExtractor	extract(), ionogram()
.DFT	ASCII	DftExtractor	plot_doppler_waterfall(), plot_doppler_spectra()
.DVL	ASCII	DvlExtractor	load_DVL_files(), plot_dvl_drift_velocities()
.SKY	ASCII	SkyExtractor	extract(), to_pandas(), plot_skymap()
.EDP	ASCII	EdpExtractor	extract(), electron density profile

Parallel loading

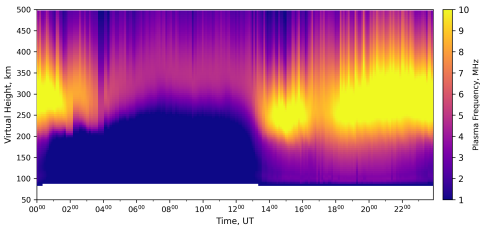
```
SaoExtractor.load_SAO_files(folders, n_procs=4)
DvlExtractor.load_DVL_files(folder, n_procs=4)
```

New post-v1.0

DFT waterfall, RSF direction-coded ionogram + directogram, multi-panel SKY sequence

DIGISONDE — SAO: Density Profiles & Scaled Parameters

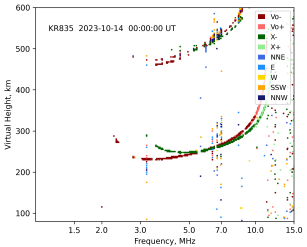
```python
from pynasonde.digisonde.parsers.sao \
    import SaoExtractor, SaoSummaryPlots

# N(h) profiles for a whole day
df = SaoExtractor.load_SAO_files(
    folders=["data/KR835/"],
    func_name="height_profile",
    n_procs=4,
)
# cols: datetime, height_km,
#       plasma_freq_mhz, Ne_cm3

# Scaled ionospheric parameters
params = SaoExtractor.load_SAO_files(
    folders=["data/KR835/"],
    func_name="scaled_params",
)
# cols: datetime, foF2, hmF2, foE, MUF

SaoSummaryPlots(df).plot_TS(
    params=["foF2","hmF2"])
```



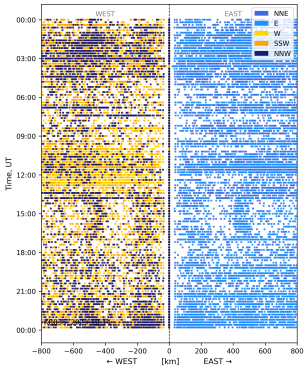
Daily isodensity contour, KR835.

DIGISONDE — RSF: Direction-Coded Ionogram

```python
from pynasonde.digisonde.parsers.
    rsf \
    import RsfExtractor

ex = RsfExtractor("KR835.RSF")
ex.extract()

# colour-coded by arrival azimuth
ex.plot_direction_ionogram()

# full day: TID / ground distance
ex.plot_directogram()
```



Direction ionogram



Directogram (daily)

What RSF adds

- Arrival azimuth per echo (4 Rx)
- Full amplitude + phase
- Directogram = TID signature

DIGISONDE — DFT, DVL, SKY

DFT

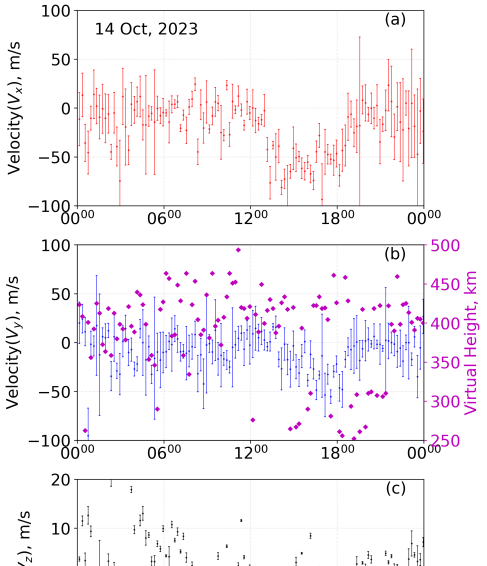
```
dft = DftExtractor("KR835.DFT")
dft.extract()
dft.plot_doppler_waterfall()
```

DVL

```
dvl = DvlExtractor.load_DVL_files(
    folder="data/KR835/", n_procs=4)
dvl.plot_dvl_drift_velocities()
# panels: Vx(E), Vy(N), Vz(up)
```

SKY

```
sky = SkyExtractor("KR835.SKY")
sky.extract()
SkySummaryPlots(sky.to_pandas())\
    .plot_skymap()
```



VIPIR Module — File Formats & I/O Classes

Ext.	Type	I/O class	Key methods
.RIQ	Binary	`RiqDataset`	`create_from_file()`, SCT + PCT, `VIPIR_VERSION_MAP`
.RIQ	—	`EchoExtractor` **[new]**	7-param echoes (h, A, V*, EP, PP, XL, YL); `to_dataframe()`
.RIQ	—	`IonogramFilter` **[new]**	6-stage: RFI, EP, multi-hop, DBSCAN, RANSAC, temporal
.NGI	NetCDF	`DataSource`	`load()`, multi-SDR data cube aggregation
.NGI	—	`AutoScaler`	noise profile, segmentation, binary trace extraction
.RIQ	—	`FullPipeline` **[new]**	end-to-end: RIQ → echoes → filter → O/X → N(h)

Supported stations

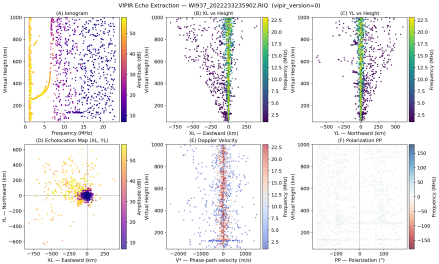
WI937 — Wallops Island (data_type=1)
PL407 — Prudhoe Bay (data_type=2)
Generic via `VIPIR_VERSION_MAP`

New since v1.0

EchoExtractor + IonogramFilter + FullPipeline — the complete signal-processing chain from raw I/Q to geophysical parameters

VIPIR — RIQ Loading, Echo Extraction & Filtering

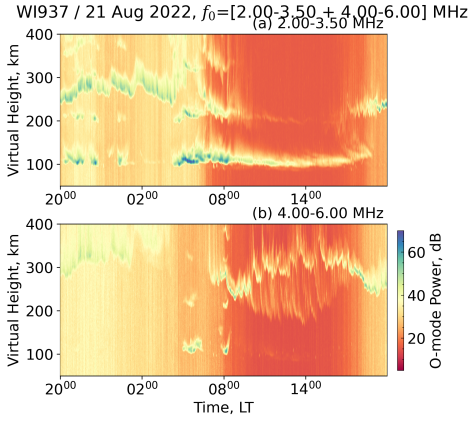
```
from pynasonde.vipir.riq.parsers.read_riq \
    import RiqDataset, VIPIR_VERSION_MAP
from pynasonde.vipir.riq.echo \
    import EchoExtractor
from pynasonde.vipir.riq.parsers.filter \
    import IonogramFilter

# 1. Load raw IQ
riq = RiqDataset.create_from_file(
    "WI937_20220821.RIQ",
    vipir_config=VIPIR_VERSION_MAP.configs
        [1])
# gates=960, r0=1.499 km, 8Rx, 4 pulses/freq

# 2. Extract 7-parameter echoes
ex = EchoExtractor(
    sct=riq.sct, pulsets=riq.pulsets,
    snr_threshold_db=3.0).extract()
df = ex.to_dataframe()
# cols: freq_mhz, height_km, amplitude_db,
#       doppler_mps, EP_deg, PP_deg, XL, YL

# 3. Six-stage noise filter
```



6-panel: ionogram (A), XL (B), YL (C), echo map (D), Doppler V* (E), PP (F).

VIPIR — NGI/SDR: FTI & AutoScaler

```
from pynasonde.vipir.ngi.source \
    import DataSource
from pynasonde.vipir.ngi.scale \
    import AutoScaler

# Load NGI data cube
ds = DataSource(files=sdr_file_list)
ds.load()

# Frequency-binned time-intensity plot
ds.fti(freq_bands=[
    (2.0, 3.5),
    (4.0, 6.0)])

# Automated scaling
scaler = AutoScaler(ds)
scaler.run()
# noise profile -> segment -> binary
# -> DBSCAN trace extraction
params = scaler.scaled_parameters
# .foF2, .foE, .hpF, .MUF, ...
```



FTI from WI937: persistent Es at 100 km; intensifies 08–12 UT.

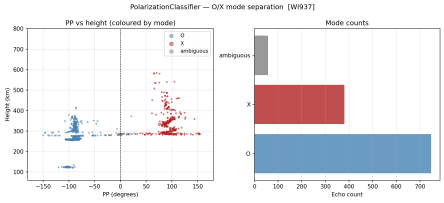
Analysis Layer — Nine Modules (All New Post-v1.0)

Class	Module	What it does
`PolarizationClassifier`	`polarization`	Separates O/X modes via PP chirality; hemisphere-aware
`TrueHeightInversion`	`inversion`	Virtual $\rightarrow$ true height via Titheridge lamination (POLAN)
`AbsorptionAnalyzer`	`absorption`	LOF, differential O/X, calibrated total, height-resolved $\kappa(z)$
`SpreadFAnalyzer`	`spread_f`	Range / frequency / mixed spread-F detection & classification
`IrregularityAnalyzer`	`irregularities`	EP structure function $\rightarrow$ spectral index $\alpha(h)$, outer scale
`IonogramScaler`	`scaler`	foF2, foE, h'F, MUF from O-mode trace
`EsCaponImager`	`es_imaging`	Capon cross-spectrum: 10$\times$ range resolution, single file
`RiqAggregator`	`es_imaging`	Multi-file A+B+C combining: $\sim$24 dB SNR, slow RTI
`NeXtYZInverter`	`nextyz`	3-D WSI model + Hamiltonian ray-tracing $\rightarrow$ N(h,x,y) + tilts

Import: `from pynasonde.vipir.analysis import` *`ClassName`*

O/X Polarization Separation

```python
from pynasonde.vipir.analysis import (
    PolarizationClassifier,
)

# O/X mode separation via PP chirality
pol = PolarizationClassifier(
    o_mode_sign=-1,    # N. hemisphere
    pp_ambiguous_threshold_deg=20,
).fit(filtered_df)

print(pol.summary())
# O=412  X=388  ambiguous=54

# Get O-mode only DataFrame
o_df = pol.o_mode_df()

# pol.annotated_df has 'mode' column
# (values: 'O', 'X', 'ambiguous')
```



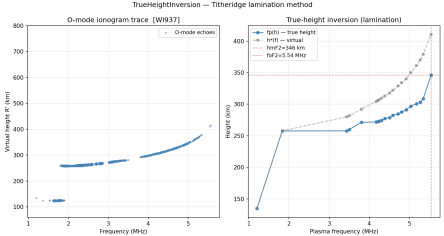
PP histogram with O/X classification regions.

Key parameters

`o_mode_sign`: -1 (N. hemisphere) / $+1$ (S. hemisphere)

True Height Inversion — N(h) Profile

```python
from pynasonde.vipir.analysis import (
    TrueHeightInversion,
)

# Titheridge lamination (POLAN)
inv = TrueHeightInversion(
    monotone_enforce=True,
    bin_width_mhz=0.05,
)
# O-mode filter applied automatically
edp = inv.fit_from_df(o_df)

print(edp.summary())
# foF2=8.12 MHz
# hmF2=287.4 km
# NmF2=8.2e5 cm^-3

edp.plot()
```



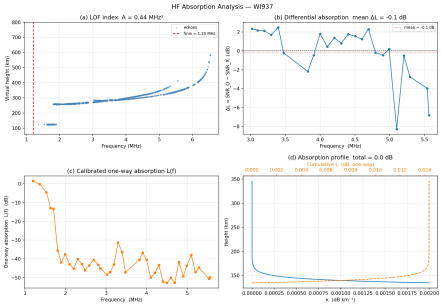
N(h) profile and virtual vs. true height comparison.

Key parameters

`bin_width_mhz`: frequency decimation for numerical stability

HF Absorption — Four Estimators

```python
from pynasonde.vipir.analysis import (
    AbsorptionAnalyzer,
)
import numpy as np

aa = AbsorptionAnalyzer(f_ref_mhz=1.0)

# 1. LOF index (no calibration needed)
lof  = aa.lof_absorption(filtered_df)

# 2. Differential O/X SNR
diff = aa.differential_absorption(
           pol.annotated_df)

# 3. Calibrated total absorption (dB)
total= aa.total_absorption(
           filtered_df, tx_eirp_dbw=70.0)

# 4. Height-resolved kappa(z) profile
prof = aa.absorption_profile(
    edp,
    nu_hz=lambda h: 1e6*np.exp(-h/8.0))
```



(a) LOF, (b) O/X differential, (c) calibrated total, (d) $\kappa(z)$ profile.

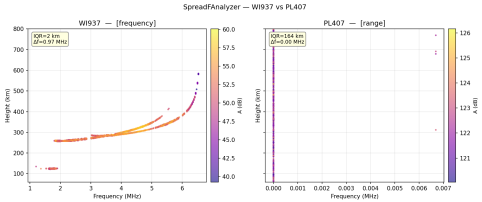
Spread-F & Irregularity Analysis

Spread-F — type classification

```
from pynasonde.vipir.analysis import (
    SpreadFAnalyzer,
    IrregularityAnalyzer,
)

sf = SpreadFAnalyzer().fit(filtered_df)
print(sf.summary())
# type=range  foEs=4.2 MHz
# h_spread_km=38.1
```

Irregularities — spectral index profile

```
irr = IrregularityAnalyzer(
    height_bin_km=10.0,
    min_echoes_per_bin=5,
).fit(filtered_df)

# result: alpha(h), outer_scale_km,
#         anisotropy proxy 0/X
irr.plot_spectral_profile()
```



Spread-F detection: type + foEs + height spread metrics.

Sporadic-E Imaging — Capon Cross-Spectrum (Liu et al. 2023)

Physical motivation

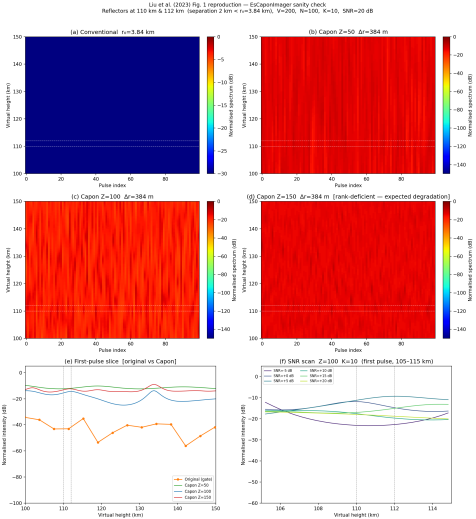
- Es: thin metallic-ion layer, 90–130 km, <1 km thick
- VIPIR gate: $r_0 = 1.499$ km — cannot resolve
- Capon (Liu et al. 2023): **10× finer resolution**, no extra hardware required
- Effective resolution: $\Delta r = r_0/K$

Resolution

	r_0	Δr (K=10)
WISS	3.84 km	384 m
VIPIR	1.499 km	**150 m**

Key constraint (Liu et al.)

All snapshots must be at the **same carrier frequency**. Different frequencies see different ionospheric reflectors —



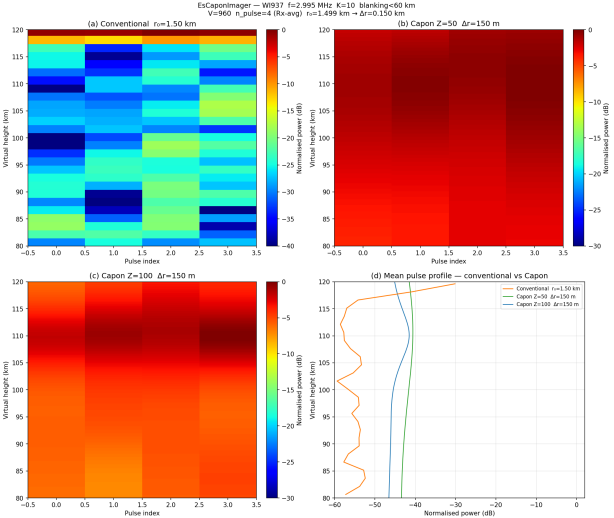
Synthetic sanity check (Liu et al. Fig. 1). Two layers at

Es Imaging — Single-File Sanity Check (Real Data)

```
# 1. Load closest pulset, avg 8 Rx
cube = _load_cube(
    "WI937.RIQ",
    freq_khz=3000.0)     # (4, 960)

# 2. Blank below 60 km (direct
    wave)
blank = int((60 - gate_start) / r0
    )
cube[:, :blank] = 0

# 3. Capon Z=50 and Z=100
res50  = EsCaponImager(
    n_subbands=50,  K=10).fit(cube
        )
res100 = EsCaponImager(
    n_subbands=100, K=10).fit(cube
        )
```

Pipeline

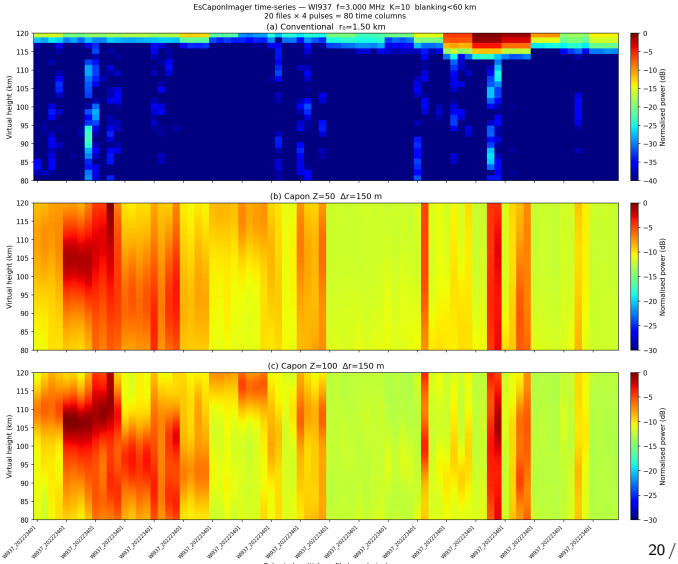
RIQ → Rx-avg → blank < 60 km
→ Capon (Z=50, Z=100, K=10)



(a) Conventional, (b) Capon Z=50, (c) Capon Z=100, (d) mean

Es Imaging — 20-File Time-Series RTI

```
from concurrent.futures import
     (
    ThreadPoolExecutor,
    as_completed)

ordered = {}
with ThreadPoolExecutor() as
    ex:
    futures = {
        ex.submit(
            _process_file, src
                ): i
        for i, src in
            enumerate(
            paths[:20])}
    for f in as_completed(
        futures):
        i = futures[f]
        ordered[i] = f.result
            ()


# re-sort, concatenate -> (80,
    n_hr)
```

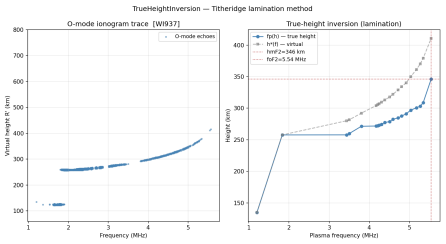


NeXtYZ 3-D Electron Density Inversion

Algorithm: Wedge-Stratified Ionosphere (WSI) model (Zabotin et al. 2006)

- Ionosphere = stack of plasma-frequency wedges
- Each boundary: height + tilt (n_x, n_y)
- Hamiltonian eikonal ray-tracing through WSI
- **Lite**: constant tilts, 6× faster
- **Full**: (h, n_x, n_y) per wedge

```
from pynasonde.vipir.analysis import (
    NeXtYZInverter,
)

inv3d = NeXtYZInverter(
    mode="lite",      # or "full"
    n_wedge_max=30,
)
result = inv3d.fit(filtered_df)
# .fp_profile: plasma freq vs true h
# .tilt_x, .tilt_y: per wedge (deg)
# .height_error_km: uncertainty
```



3-D vs 1-D

Standard inversion assumes horizontal stratification. NeXtYZ recovers tilts and produces full 3-D $N(h, x, y)$ from a single vertical sounder.

What Has Been Added Since v1.0 Publication

VIPIR signal-processing chain (new)

- `EchoExtractor` — 7-param Dynasonde echoes
- `IonogramFilter` — 6-stage pipeline
- `FullPipeline` — end-to-end automation

Analysis layer (entirely new)

- `PolarizationClassifier` — O/X separation
- `TrueHeightInversion` — Titheridge/POLAN
- `AbsorptionAnalyzer` — 4 estimators
- `SpreadFAnalyzer` — range/freq/mixed
- `IrregularityAnalyzer` — $\alpha(h)$ profile
- `IonogramScaler` — foF2, MUF

Es super-resolution (new, not on roadmap)

- `EsCaponImager` — 150 m resolution, gate blanking
- `RiqAggregator` — multi-snapshot Capon, `blank_min_km`
- Rx-averaging per pulse $\Rightarrow$ (n_{pulse}, V) cube
- 20-file parallel time-series: 80 columns via `ThreadPoolExecutor`

Versioning roadmap status

Ver.	Planned item	Status
v2.0	Modified POLAN	Done
v2.0	Oblique ionograms	Partial
v3.0	Directional info	Done
v3.0	Polarization info	Done
v4.0	3-D ray tracing	Done
v4.0	Cisium plotting	Planned
Extra	Es Capon imaging	Done
Extra	Absorption analysis	Done
Extra	Irregularities	Done

v2.0 release is overdue.
Core v3.0 + v4.0 features are already done.

Complete Public API — All Classes

DIGISONDE I/O

`SaoExtractor`	SAO profiles + params
`SaoSummaryPlots`	Isodensity, time-series
`RsfExtractor`	Direction ionogram, directogram
`SbfExtractor`	SBF binary
`DftExtractor`	Doppler waterfall + spectra
`DvlExtractor`	Drift velocities
`SkyExtractor`	SKY polar maps
`SkySummaryPlots`	Multi-panel skymap
`EdpExtractor`	Electron density profile

VIPIR I/O

`RiqDataset`	Load .RIQ + SCT/PCT
`EchoExtractor`	7-param echoes
`IonogramFilter`	6-stage filter
`DataSource`	NGI/SDR data cube
`AutoScaler`	Segment + scale
`FullPipeline`	End-to-end

Analysis

`PolarizationClassifier`	O/X via PP
`PolarizationResult`	annotated df, counts
`TrueHeightInversion`	N(h), foF2, hmF2
`EDPResult`	profile dataclass
`AbsorptionAnalyzer`	4 estimators
`LOFResult`	LOF index
`SpreadFAnalyzer`	Type + metrics
`SpreadFResult`	classification
`IrregularityAnalyzer`	$\alpha(h)$, L_{outer}
`IonogramScaler`	foF2, foE, MUF
`EsCaponImager`	Capon 1-file
`EsImagingResult`	pseudospectrum
`RiqAggregator`	Multi-file RTI
`NeXtYZInverter`	3-D N(h,x,y)
`NeXtYZResult`	wedge planes

pynasonde.readthedocs.io

Summary

pynasonde provides a complete end-to-end workflow:

1. Read **9 file formats** across two instrument families
2. **7-parameter echo extraction** + **6-stage filtering**
3. **9 analysis modules**: O/X, inversion, absorption, spread-F, irregularities, scaler, Es imaging, NeXtYZ
4. **Es Capon imaging**: 150 m from 1.499 km gates
5. **NeXtYZ**: 3-D $N(h, x, y)$ + tilts from single sounder
6. **FullPipeline**: raw I/Q $\rightarrow$ geophysical parameters

Quick start

```
pip install pynasonde
```

github.com/shibaji7/pynasonde

pynasonde.readthedocs.io

Cite as

Chakraborty et al. (2026).
SoftwareX 34, 102617.
DOI: 10.1016/j.softx.2026.102617

Next: v2.0

Full oblique ionogram parsing, POLAN-v2 solver, multi-instrument fusion

References I

- Chakraborty S., Bullett T., Barjatya A., Mabie J. (2026). pynasonde: An open-source Python library for ionosonde data processing. *SoftwareX* 34, 102617.
- Liu, T., Yang, G., & Jiang, C. (2023). High-resolution sporadic E via Capon cross-spectrum. *Space Weather* 21, e2022SW003195.
- Zabotin, N. A., Wright, J. W., & Zhbankov, G. A. (2006). NeXtYZ. *Radio Science* 41, RS6S32.
- Titheridge, J. E. (1967). *JATP* 29, 763–778.
- Grubb, R. N. et al. (2008). VIPIR. *URSI General Assembly*.
- Reinisch, B. W. et al. (2009). New Digisonde. *Radio Science* 44.
- Chakraborty S., Qian L., Mrak S. et al. (2025). G-condition 2017 GAE. *Earth and Space Science* 12, e2024EA004007.