

# Neutron Data Cleaner Preprocessing Filter

18 November 2025

---



**VIPR Consortium**

# The Problem

---

## Noisy Neutron Reflectometry Data

- Challenge 1: Negative Intensities ( $R < 0$ )
- Challenge 2: High Error Bars ( $dR/R > \text{threshold}$ )

**Goal:** Clean data before interpolating to model's  $q$  grid

## Integration into VIPR framework

---

### VIPR Inference Pipeline:

1. Load Data
2. Load Model
3. Normalize
4. Preprocess
  - └─ weight=-10: Neutron Data Cleaner ← NEW
  - └─ weight=0: Interpolation to model's q grid
5. Predict
6. Postprocess

#### Why weight=-10?

- Runs **before** interpolation

# Cleaning strategy

---

## Two step approach:

### Step 1: Remove negative intensities

```
mask = R > 0 # NumPy Boolean Indexing  
q, R = q[mask], R[mask]
```

### Step 2: Filter/ Truncate High Errors

```
rel_error = dR / R  
# Option A: Remove isolated high-error points  
# Option B: Truncate at N consecutive high-error points
```

# ViPR Integration

[https://codebase.helmholtz.cloud/vipr/vipr-reflectorch-plugin/-/blob/master/vipr\\_reflectometry/shared/preprocessing/neutron\\_data\\_cleaner.py](https://codebase.helmholtz.cloud/vipr/vipr-reflectorch-plugin/-/blob/master/vipr_reflectometry/shared/preprocessing/neutron_data_cleaner.py)

```
neutron_data_cleaner.py 12.06 KiB
1 """
2 Neutron reflectometry data cleaning filter.
3
4 Removes negative intensity values and filters/truncates points with high error bars.
5 """
6
7 import numpy as np
8 from typing import Tuple, Optional, Dict, List
9 from vipr.plugins.discovery.decorators import discover_filter
10 from vipr.plugins.inference.dataset import DataSet
11
12
13 class NeutronDataCleaner:
14     """
15     Preprocessing filter for cleaning neutron reflectometry data.
16
17     This filter removes problematic data points before further processing:
18     1. Removes points with negative intensity values ( $R < 0$ )
19     2. Filters and/or truncates curves at consecutive high error bars
20
21     The cleaning is applied row-by-row (per spectrum) and returns a ragged
22     array DataSet where each spectrum may have a different length after cleaning.
23
24     Uses weight=10 to run before interpolation (weight=0).
25     """
26
27     def __init__(self, app):
28         self.app = app
29
30     @discover_filter(
31         'INFERENCING_PREPROCESS_PNE_FILTER',
32         weight=10,
33         enabled_in_config=False,
34         parameters={
35             'error_threshold': {
36                 'type': 'float',
37                 'default': 0.5,
38                 'help': 'Relative error threshold (dR/R) for filtering high-error points (range: 0.0-1.0)'
39             },
40             'consecutive_errors': {
41                 'type': 'int',
42                 'default': 3,
43                 'help': 'Number of consecutive high-error points to trigger truncation (minimum: 1)'
44             },
45             'remove_single_errors': {
46                 'type': 'bool',
47                 'default': False,
48                 'help': 'Remove isolated high-error points before truncation'
49             }
50         }
51     )
52     def clean_experimental_data(self, data: DataSet, **kwargs) -> DataSet:
```

## VIPR Integration - Discovery & Config

### Discovery decorator:

```
@discover_filter(
    'INFERENCE_PREPROCESS_PRE_FILTER',
    weight=-10,
    enabled_in_config=False,
    parameters={
        'error_threshold': {
            'type': 'float',
            'default': 0.5,
            'help': 'Relative error threshold (dR/R) for filtering high-error points (range: 0.0-1.0)'
        },
        'consecutive_errors': {
            'type': 'int',
            'default': 3,
            'help': 'Number of consecutive high-error points to trigger truncation (minimum: 1)'
        },
        'remove_single_errors': {
            'type': 'bool',
            'default': False,
            'help': 'Remove isolated high-error points before truncation'
        }
    }
)
def clean_experimental_data(self, data: DataSet, **kwargs) -> DataSet:
```

## Filter design

---

### Signature of the callback function

```
def clean_experimental_data(self, data: DataSet, **kwargs) -> DataSet:
```

### Why **\*\*kwargs**?

- Flexible: Each filter picks the parameters it needs
- Example parameters : `error_threshold`, `consecutive_errors`, `remove_single_errors`

### Why **DataSet**?

- Type-safe: Ensures consistent data structure
- Chainable: Output of one filter → Input of next filter

## VIPR CLI usage

---

For using the new filter callback upon inference, it has to be added to the INFERENCE\_PREPROCESS\_PRE\_FILTER section

```
vipr:
  inference:
    filters:
    ...
    INFERENCE_PREPROCESS_PRE_FILTER:
    ...
    - class: vipr_reflectometry.shared.preprocessing.neutron_data_cleaner.NeutronDataCleaner
      method: clean_experimental_data
      enabled: true
      parameters:
        error_threshold: 0.5
        consecutive_errors: 3
        remove_single_errors: false
```



# NeutronDataCleaner can be tried in the web-app

1 Click “LOAD EXAMPLES”

SAVE LOAD **LOAD EXAMPLES** VIEW CONFIG

2 Select D17 example config

Select Example Configuration

After selecting a configuration, you must press the Run Prediction button to start the prediction.

PTCDI-C3\_XRR\_2025-06-12T11-23-25-881Z.json

Ni500\_NR\_2025-06-12T11-30-19-178Z.json

Ni\_on\_Glass\_NR\_2025-06-12T11-41-04-080Z.json

**D17\_SiO\_2025-11-11T16-22-02-551Z.json**

D17\_SiO\_pre\_filtered\_2025-06-16T14-14-53-306Z.json

ORSO\_example\_2025-06-18T10-08-12-671Z.json

XRR\_hdf5\_DIP\_1\_2025-08-22T12-19-21-976Z.json

MARIA\_Dataset\_s000003\_2025-09-12T12-20-14-645Z.js...

PANPE\_2layers\_XRR\_2025-01-10T13-46-00-000Z.json

mc1\_nsf\_2025-10-06.json

CANCEL

3 NeutronDataCleaner should be active

Preprocess

Filters:

INFERENCE\_PREPROCESS\_PRE\_FILTER:

Reflectorch\_preprocess\_interpolate

NeutronDataCleaner.clean\_experimental\_data

4 Click “RUN PREDICTION”

Configure Processing Steps

1

Load Data

Handler: ccs\_spectrereader

EDIT CONFIGURATION

2

Load Model

Handler: reflectorch

EDIT CONFIGURATION

3

Normalize

EDIT CONFIGURATION

4

Preprocess

Filters:  
INFERENCE\_PREPROCESS\_PRE\_FILTER:  
Reflectorch\_preprocess\_interpolate

EDIT CONFIGURATION

5

Predict

Handler: reflectorch\_predictor

EDIT CONFIGURATION

6

Postprocess

EDIT CONFIGURATION

RUN PREDICTION

Auto Execute

## Adjustable parameters for the NeutronDataCleaner in the web-app



1


Click button for config edit

Preprocess

Filters:

*INFERENCE\_PREPROCESS\_PRE\_FILTER:*


 Reflectorch\_preprocess\_interpolate  NeutronDataCleaner.clean\_experimental\_data

 EDIT CONFIGURATION

2

Open expansion panel

PREPROCESS

 INFERENCE\_PREPROCESS\_PRE\_FILTER

Available Callbacks


☒ Callback #1


Method: clean\_experimental\_data


Class: vipr\_reflectometry.shared.preprocessing.neutron\_data\_cleaner.NeutronDataCleaner

Weight: -10

Parameters:

error\_threshold  (float)

consecutive\_errors  (int)

remove\_single\_errors  (bool)

## Summary

---

- NeutronDataCleaner: Removes negative intensities & high-error points
- Integrated into VIPR (weight=-10)
- Configurable via YAML or Web-GUI
- Available in vipr-reflectometry-plugin