EUROPEAN
SPALLATION
SOURCE

Spectroscopy Scientific and Technical Advisory Panel Revision: v2

Instrument Data Scientist Report

Gregory Tucker

April 29, 2022

A general workflow is being developed which will transform collected neutron event data into a useable form for a general neutron spectrometer, including all those in scope for the European Spallation Source.

1 General transformation workflow

All neutron spectrometers are designed to collect data which can be converted into $I(Q,E) \propto \frac{\partial^2\sigma}{\partial Q \partial E}$. Modern time-of-flight spectrometers record individual neutron capture events which may be treated to account for differing measurement effects dependent on detector, collection period, sample environment, or other parameters. Additionally, the recorded coordinates for each event can be transformed from, e.g., three spatial and one time for each event to (Q,E). The process of making the corrections and applying any change of coordinate system is data *transformation*.

A further step necessary for the visualisation of $I(Q,E)$ is data *reduction* in which one or more histogram of transformed data is collected. Routine reduction of all data from a typical ESS Spectrometer experiment into resolution-limited histograms for each detector element, fixed-energy, and collection period, is not tenable due to large file sizes and, using current techniques, a proliferation of individual data files. Instead collected event data will be transformed and stored, ready to be reduced as required by a user for visualisation and fitting.

The transformation process for neutron event data ultimately depends on details of the instrument used to collect it; however, there are commonalities between instruments which warrant description of a general data transformation workflow.

1.1 Incident neutron time windows

The time between subsequent neutron source pulses at a time-of-flight spectrometer is often referred to as a *frame*. On a fixed-incident-energy multiplexing spectrometer, the frame is subdivided into a number of *subframes*, equal to the number of different incident energy *reps*. The time between the earliest-arriving neutron in each (sub)frame depends on details of the flight-path between the sample and detector, and is therefore potentially unique for every detector element in an instrument.

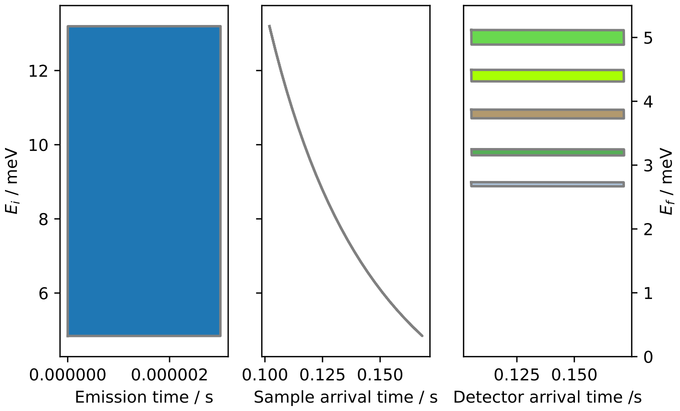
An important step in the data transformation process is converting the absolute timestamp of an event to time-of-flight. This can only be achieved with knowledge of the time from each source pulse to the start of each time window for every detector element. The first step in determining the absolute time windows is constructing an acceptance diagram for the chopper system of the primary spectrometer; a process which is common to all time-of-flight neutron scattering instruments.

The acceptance diagrams presented in figure 1 have been calculated from details of the planned primary spectrometers of BIFROST and CSPEC along with anticipated choices for chopper speeds and phases. The arrival times at the sample or each detector element can be used to define the time windows used in transforming the absolute event time to its time of flight as an automated step in the transformation workflow.

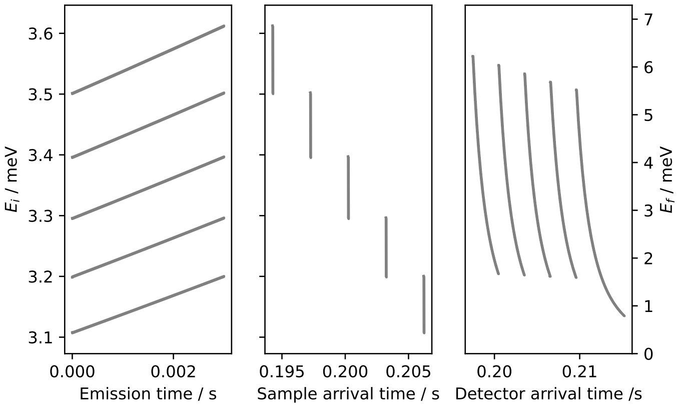
1.2 Possible prompt-pulse contamination

One concern for experiments at instruments located at a pulsed spallation source is the detection of very high energy spallation-produced neutrons which do not thermalise within the source moderator. Spallation-produced neutrons can have energies which are a significant fraction of the accelerated proton energy; if they are not thermalised within the moderator or target shielding, they will exit the target monolith with very high velocity. Some of these may ultimately thermalise within a detector tank and then subsequently be recorded as events by an instrument shortly after their spallation.

These prompt-pulse neutrons, if present, will appear once within each frame at an apparent energy-transfer depending on the instrument settings. By keeping track of the time-windows for each (sub)frame we can provide easy to understand feedback about possible prompt-pulse contamination when a user is deciding on the instrumental



(a) BIFROST



(b) CSPEC

Figure 1:
Acceptance diagrams calculated from chopper locations, speeds, and phases, for multiplexing indirect geometry spectrometer, 1a BIFROST, and multiplexing direct geometry spectrometer, 1b CSPEC.

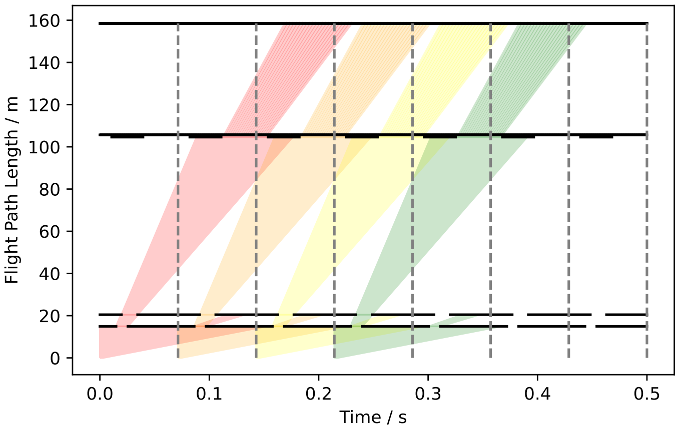


Figure 2:
Time distance diagram showing multiple frames; multiple subframes are present but too fine to see. Spallation occurs once per frame, as indicated by the dashed grey vertical lines, and may result in detection of prompt-pulse neutrons in one subframe.

configuration for their experiment.

The time-distance diagram shown in figure 2 is, effectively, a byproduct of an acceptance diagram calculation. It can be used to identify which subframe includes the prompt-pulse, shortly after which high-energy neutrons may thermalise in the detector. A straightforward calculation then provides a user with the apparent energy transfer of the possible contamination to aid in their experiment planning.

2 `scipp`

The transformation of collected data into a form ready for analysis will be done using the software `scipp`, which is being developed by members of the Data Reduction, Analysis & Modelling group within DMSC. The major advantages of `scipp` over alternatives are 1. scalars, vectors, and arrays with associated units; 2. dimension labels and label-based indexing and slicing; 3. the ability to store variances alongside values; 4. and an emphasis on event data. This forms the base of a hierarchy of packages including `scippneutron`, which adds transformation routines useful for all neutron techniques, and `ess`, which adds technique- and instrument specific-routines applicable at the ESS as well.

Combining `scipp` with Jupyter notebooks allows for interactive workflows with fast iteration. One such notebook for fitting binned incident-beam monitor peaks to extract monochromatic energies is shown in figure 3. The displayed notebook has been adapted into a bespoke GUI interface using Voila, which runs a Jupyter notebook while hiding-away input cells and only displaying their output.

The GUI presents a user with a simple file-chooser, an opportunity to input expected monochromatic energies, and a button to load, display, and fit the monitor data.

The Voila-based GUI presented in figures **??** runs on top of Python code in the notebook, figure 3, tailored for details of the LET spectrometer. That Python notebook, however, itself uses a general underlying module which, with similar top-level tailoring, should be useable for the direct-geometry ESS spectrometers.

The precise GUI presented is almost certainly suboptimal since, thus far, it has a user-base of one and was converted from working Jupyter notebook to GUI in one evening. This, however, is a feature of using a flexible tool, like Voila, to construct the GUI; as it becomes better exercised by more users, modifications to layout, feedback, and function are relatively easy to implement in short time periods.

3 DMSC Instrument Scientist's workshop

A workshop has been organised, and will take-place shortly, to help afford ESS Instrument Scientists with a holistic view of the DMSC software stack. Organisation of the workshop was within the purview of a small committee consisting of two Instrument Scientists and two Instrument Data Scientists, pulled from SANS and Spectroscopy. The workshop will take place over two days at the DMSC offices in Copenhagen and will require in-person participation in order to better facilitate communication. Presentations will be given by DMSC staff on topics ranging from the contents and providence of an Event NeXus file through the advantages of using `scipp`. Hands-on tutorials will help participants construct concrete understandings of the various tools and their applications.

This workshop is likely to be the first of a series of in-person workshops and training sessions following lessons learned from online-only Python training courses offered by DMSC in connection with the last two IKON meetings. Other envisioned future workshops will focus on production-ready notebooks and GUIs, and the use of ray-tracing simulations in the planning and analysis of inelastic neutron spectroscopy measurements.

```
import scipp as sc
import numpy as np
from nexus_interface import monitors
```

```
def load_monitors(nx_file):
    monitor_1 = monitors.histogram_monitor(nx_file, 1)
    monitor_5 = monitors.histogram_monitor(nx_file, 5)
    return monitor_1, monitor_5

def determine_expectations(values, delay, monitor_1, monitor_5):
    # ideally this would be determined from the chopper phases and speeds:
    expected_energies = sc.array(values=values, unit='meV', dims=['rep'])
    moderator_emission_delay = sc.scalar(value=delay, unit='µs')
    m1_times = monitors.expected_monitor_times(expected_energies, monitor_1, moderator_emission_delay)
    m5_times = monitors.expected_monitor_times(expected_energies, monitor_5, moderator_emission_delay)

    out=f"Expected parameters:\n{'Energy':^11}  {'M1 time':^11}  {'M5 time':^11}\n"
    out+=f"{'------':^11}  {'-------':^11}  {'-------':^11}\n"
    for (en, t_1, t_5) in zip(expected_energies, m1_times, m5_times):
        out += f'{en.value:7.4f} {en.unit} : {t_1.value:7.1f} {t_1.unit} & {t_5.value:7.1f} {t_5.unit}\n'

    return out, expected_energies, moderator_emission_delay, m1_times, m5_times

def fit_monitors(values, delay, monitor_1, monitor_5):
    params, delay, fit_1, fit_5 = monitors.two_monitor_energies(monitor_1, monitor_5, delay=delay, guess=values)
    return params, delay, fit_1, fit_5

def plot_total(monitor_1, monitor_5, fit_1, fit_5):
    import matplotlib.pyplot as pp
    p = sc.plot({'monitor_1': monitor_1, 'fit_1': fit_1, 'monitor_5': monitor_5, 'fit_5': fit_5})

    return p

def plot_individual(monitor_1, monitor_5, fit_1, fit_5, params):
    import matplotlib.pyplot as pp
    nei = np.sum(params.shape)
    fig, axs = pp.subplots(2, nei, sharey='all', figsize=(6*1.618, 6))
    for i in range(nei):
        sc.plot({'monitor_1': monitor_1, 'fit_1': fit_1}, ax=axs[0,i])
        pp.setp(axs[0,i], xlim=np.array((-3, 3))*params['short_width'][i].value + params['short_x0'][i].value)
    for i in range(nei):
        sc.plot({'monitor_5': monitor_5, 'fit_5': fit_5}, ax=axs[1,i])
        pp.setp(axs[1,i], xlim=np.array((-3, 3))*params['long_width'][i].value + params['long_x0'][i].value)

    return fig
```

```
from pathlib import Path
from ipyfilechooser import FileChooser
nx_path = Path.home().joinpath('Documents', 'ISIS', 'LET')
nx_file = 'LET00076866.nxs'
fc = FileChooser(nx_path, select_default=True)
fc.default_filename = nx_file
fc.reset()
fc.sandbox_path = nx_path
fc.filter_patern = '*.nxs'
fc.title = '<b>Select an Event NeXus file</b>'
#fc.register_callback(file_chosen_callback)
display(fc)
```

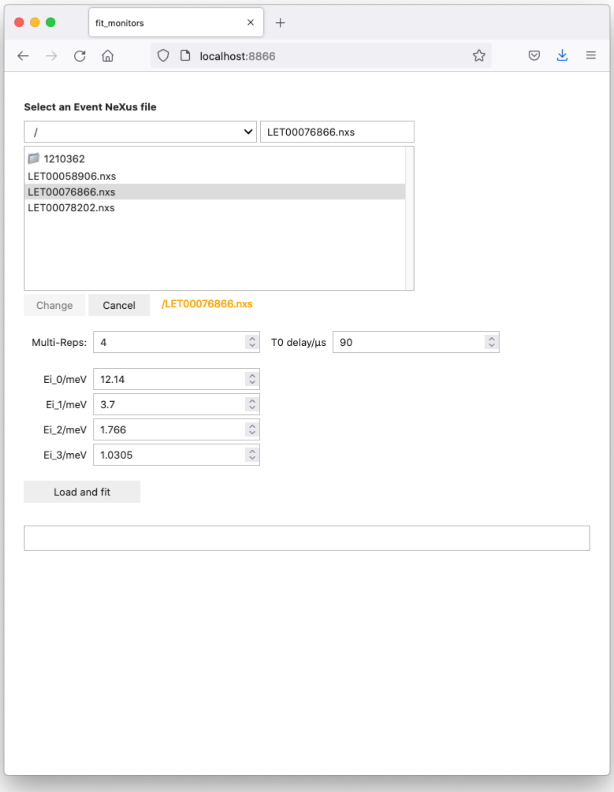
```
import ipywidgets as w
reps = w.BoundedIntText(value=4, min=1, max=15, step=1, description='Multi-Reps:')
wdel = w.BoundedIntText(value=90, min=0, max=10**3, step=1, description='T0 delay/µs')
w.Box([reps, wdel])
```

```
expected_energies = [12.14, 3.7, 1.766, 1.0305]
ens = [w.BoundedFloatText(value=x, min=0, max=100, step=0.01, description=f'Ei_{i}/meV') for i, x in enumerate(expected_energies)]
w.VBox(ens)
```

```
%matplotlib inline
button = w.Button(description='Load and fit')
display(button)

labels = ['Parameters', 'Monitors', 'Peaks']
outs = [w.Output() for _ in labels]
tab = w.Tab()
tab.titles = labels

def load_and_fit_callback(b):
    import matplotlib.pyplot as pp
    m1, m2 = load_monitors(fc.selected)

    tab.children = [outs[1]]
    tab.set_title(0, labels[1])
    with outs[1]:
        two = sc.plot({'monitor_1': m1, 'monitor_5': m2})
        pp.show(two)

    info, eis, delay, t1, t2 = determine_expectations([x.value for x in ens], wdel.value, m1, m2)
    tab.children = [outs[0], outs[1]]
    [tab.set_title(i, t) for i, t in enumerate([labels[0], labels[1]])]
    outs[0].append_stdout(info)

    p, delay, f1, f2 = fit_monitors(eis, delay, m1, m2)
    wdel.value = delay.value
    with outs[0]:
        display(p['energy'])

    total = plot_total(m1, m2, f1, f2)
    outs[1].clear_output()
    with outs[1]:
        print("\nAll fit peaks")
        pp.show(total)

    indiv = plot_individual(m1, m2, f1, f2, p)
    with outs[2]:
        print("\nIndividual fit peaks")
        pp.show(indiv)

    tab.children = outs
    [tab.set_title(i, t) for i, t in enumerate(labels)]

button.on_click(load_and_fit_callback)
display(tab)
```

Figure 3:

A Jupyter notebook utilising `scipp` and Mantid to fit monochromatic energies present in two incident-beam monitors using LET data.



Select an Event NeXus file

/ LET00076866.nxs

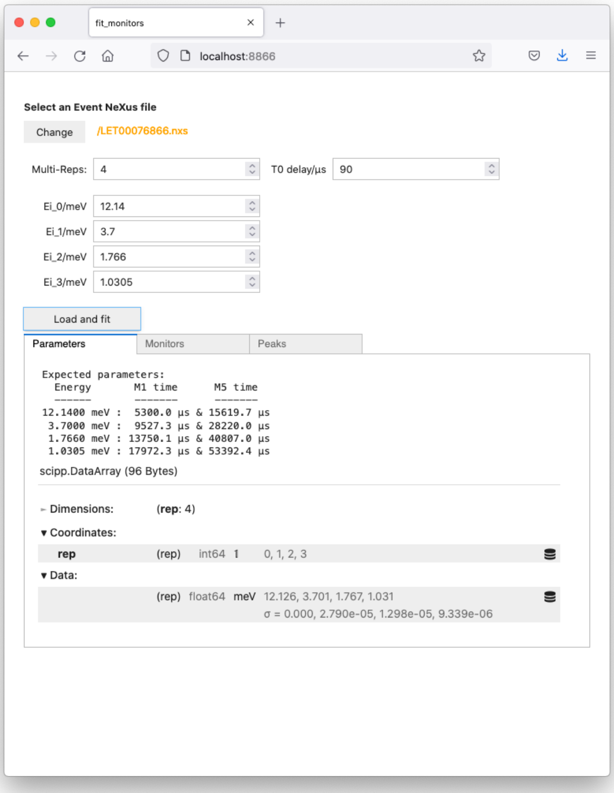
1210362
LET00058906.nxs
LET00076866.nxs
LET00078202.nxs

/LET00076866.nxs

Multi-Reps: 4
T0 delay/µs 90

Ei_0/meV 12.14
Ei_1/meV 3.7
Ei_2/meV 1.766
Ei_3/meV 1.0305

Figure 4:
The Jupyter notebook based Voila GUI file selection interface.



Select an Event NeXus file

/LET00076866.nxs

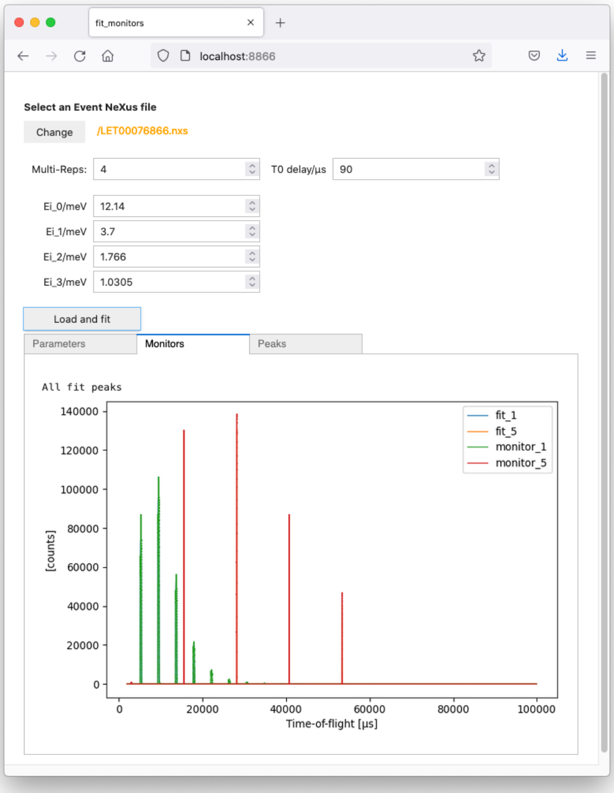
Multi-Reps: 4
T0 delay/µs 90

Ei_0/meV 12.14
Ei_1/meV 3.7
Ei_2/meV 1.766
Ei_3/meV 1.0305

```
Expected parameters:
  Energy       M1 time      M5 time
  ------       -------      -------
12.1400 meV :  5300.0 µs & 15619.7 µs
 3.7000 meV :  9527.3 µs & 28220.0 µs
 1.7660 meV : 13750.1 µs & 40807.0 µs
 1.0305 meV : 17972.3 µs & 53392.4 µs
```

```
scipp.DataArray (96 Bytes)

► Dimensions:      (rep: 4)
▼ Coordinates:
   rep             (rep)   int64    𝟙     0, 1, 2, 3
▼ Data:
                   (rep)   float64  meV   12.126, 3.701, 1.767, 1.031
                                          σ = 0.000, 2.790e-05, 1.298e-05, 9.339e-06
```

Figure 5:
The Jupyter notebook based Voila GUI: initial guess for peak times based on monitor positions and user-provided incident energy guesses, updated to show the fit energies as a `scipp` variable.



Select an Event NeXus file

/LET00076866.nxs

Multi-Reps: 4

T0 delay/µs 90

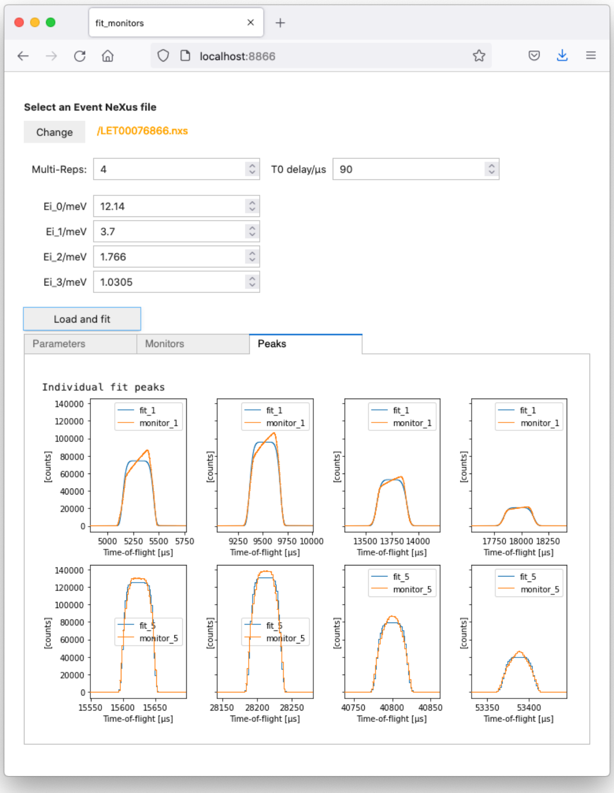
Ei_0/meV 12.14

Ei_1/meV 3.7

Ei_2/meV 1.766

Ei_3/meV 1.0305

Figure 6:
The Jupyter notebook based Voila GUI: monitor spectra, updated to show the fit results, which do not visibly deviate from the histogram monitors in this view.



Select an Event NeXus file

/LET00076866.nxs

Multi-Reps: 4

T0 delay/µs 90

Ei_0/meV 12.14

Ei_1/meV 3.7

Ei_2/meV 1.766

Ei_3/meV 1.0305

Figure 7:
The Jupyter notebook based Voila GUI: monitor spectra and fit peaks in the regions surrounding the monochromatic energies.