Dawn 4 Dawn workshop

Copenhagen – 20th October 2016

https://github.com/belkassaby/nobugs_dde

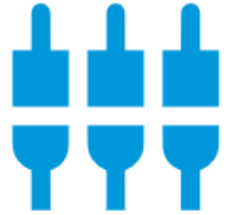
Data Analysis WorkbeNch

Visualisation

Comprehensive visualisation of data in 1D, 2D and 3D

Development Environments

Integrated Python, Jython and Eclipse plug-in development, debugging and execution



Processing and Workflows

for visual algorithms analysing scientific data

https://github.com/belkassaby/nobugs_dde

About

DAWN is funded primarily by Diamond Light Source and works within the Eclipse Science Working Group to allow a range of science facilities to make use of the product.

Commercial and non-commercial partners participate to the DAWN collaboration. **DAWN** also contributes to and uses PyDev.

Who we are and what we're doing: **DAWN** is an open source software (licence) built on the Eclipse/RCP platform in order to scale to address a wide range of applications and to benefit from the workbench and advanced plugin system implemented in Eclipse. The only limiting factors are ideas and human resources. For this reason contributors are welcomed!

DAWN is mainly developed at the Diamond Light Source.

Use of Dawn

If the analysis features available in Dawn are used significantly in your work, users are kindly requested to add acknowledgements to published work and to cite:

Basham, M., Filik, J., Wharmby, M.T., Chang, P.C.Y., El Kassaby, B., Gerring, M., Aishima, J., Levik, K., Pulford, B.C.A., Sikharulidze, I. et al. (2015). J. Synchrotron Rad. 22, doi:10.1107/S1600577515002283

VIEW ARTICLE

Links

Twitter
GitHub
Diamond Light Source
Eclipse Science Working Group
Eclipse Foundation

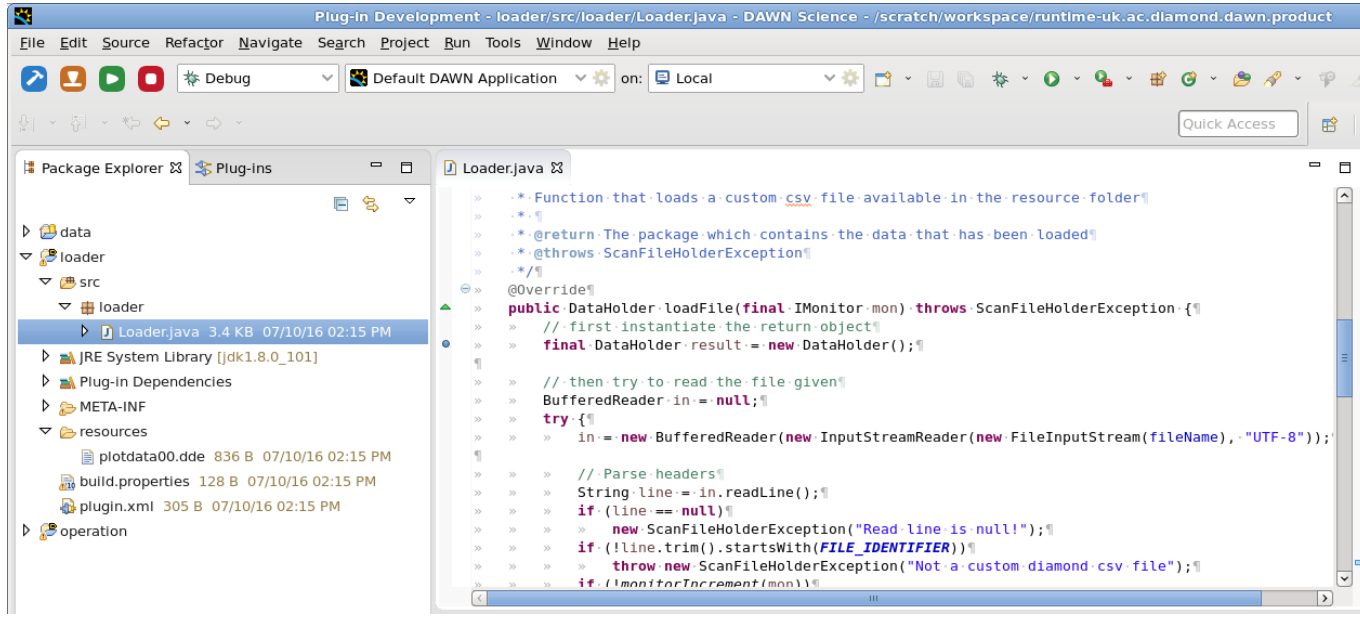
https://github.com/belkassaby/nobugs_dde

Create a custom Loader

- Go to the Plugin-in Development perspective
  - Window->Perspective->Open perspective->Other->Plug-in…
- Create a New DAWN Plugin-in project
  - File->New->DAWN plug-in project
  - Give it a project name, identifier, name and institute
  - Select “uk.ac.diamond.scisoft.analysis.io.loader” as an extension point id
  - Click “next” and set the file extension field to “.dde”; click “Finish”
- Open the generated java class in the source folder in the new project located in your workspace to see a default loader for a custom csv file (in the resource folder of that project)

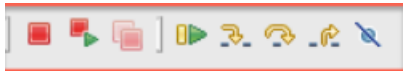
https://github.com/belkassaby/nobugs_dde

Debug through your code



- Add a break point at the beginning of "`public DataHolder loadFile(final IMonitor mon)`" method
- Select "Debug" next to the dev toolbar and press the green launch button

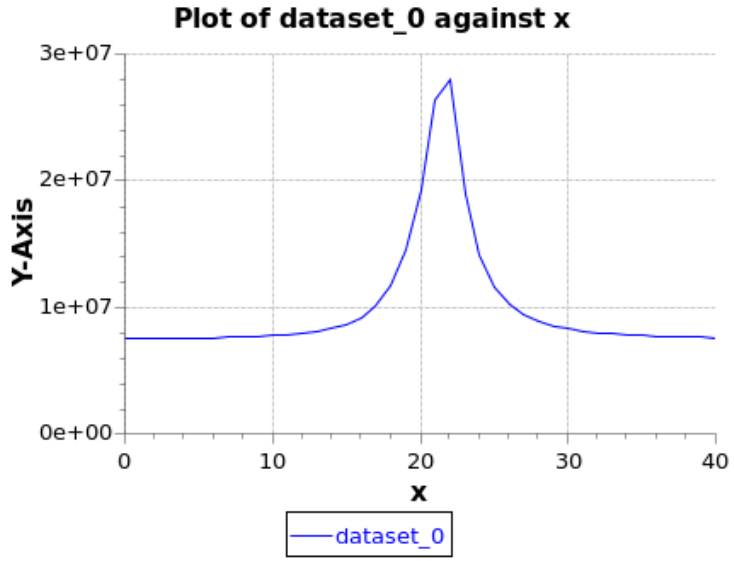
- Go to the "Processing perspective" in the new DAWN instance
- Drag and drop the custom csv file in the resource folder of the DAWN development instance ("plotdata00.dde") to the "Data Slice View" of the "processing perspective" in the new DAWN instance
- Step-over the loadFile method using the Eclipse Debug toolbar in the dev DAWN instance

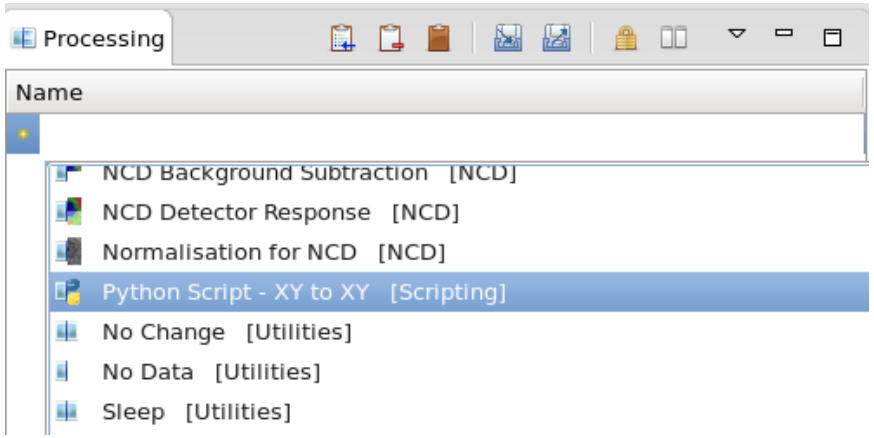


https://github.com/belkassaby/nobugs_dde

Processing

- Our custom csv file is loaded through the LoaderFactory mechanism in DAWN which now has our new loader in its list of available file loaders
- In the new window that pops-up, select “dataset_0[41]” as the dataset; the data plot should change; press “Finish”
- In the “Processing” view, select the “Python Script – XY to XY” operation



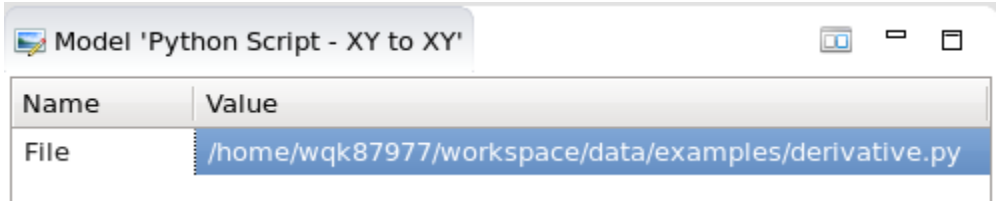


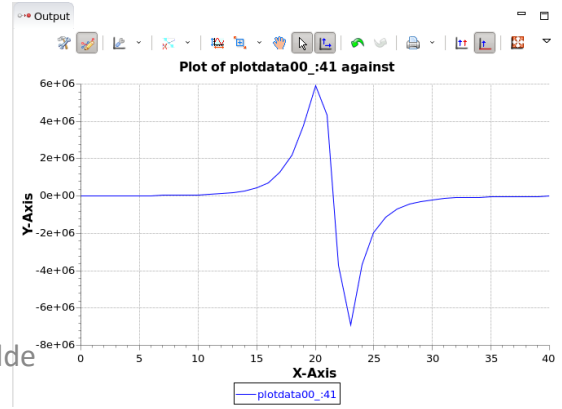
https://github.com/belkassaby/nobugs_dde

Apply a derivative with a Python script

- Let’s apply a derivative on the data loaded in “Processing”
  - Create a Pyton file to link with the operation previously added to the “Processing” view
    - Right-Click on a folder in a project in your workspace
    - Select New->File
    - type “derivative.py” as a file name
    - Add the following code to your file and save it
  - Link the python file to the operation
  - Click on the python script operation to apply the processing workflow to the data

```python
import numpy as np
'''
Derivative python script example
'''
def run(data, xaxis, **kwargs):
    result = np.gradient(data)
    output = {'data': result}
    output["xaxis"] = xaxis
    return output
```





https://github.com/belkassaby/nobugs_dde

Modify the custom loader to handle a stack

Pages / DAWN Training Home

Linking Datasets from Multiple Files [DAWN 2.2]

Created by Jacob Filik, last modified on Sep 01, 2016

In DAWN 2.2 it is possible to link datasets from multiple different files into one "virtual" dataset by creating a simple linking text file. For example this could be used to generate a 4D dataset from a grid scan of an imaging detector that collects a single tiff at each point (virtual dataset shape would be [scan y, scan x, tiff size y, tiff size x]) or combining multiple line scans in HDF files into a grid scan.

An example text file can be found below. To work, the file should have the extension .dawn.

```
# DIR_NAME : /home/me/data/tiffs/
# DATASET_NAME: image-01
# SHAPE: 3,4
# FILE_NAME
00001.tif
00002.tif
00003.tif
00004.tif
00005.tif
00006.tif
00007.tif
00008.tif
00009.tif
00010.tif
00011.tif
00012.tif
```

The header needs to contain # DIR_NAME followed by a colon then the path to directory the files are in.

Then (possibly multiple) # DATASET_NAME entries, which are the names of the datasets you wish to link

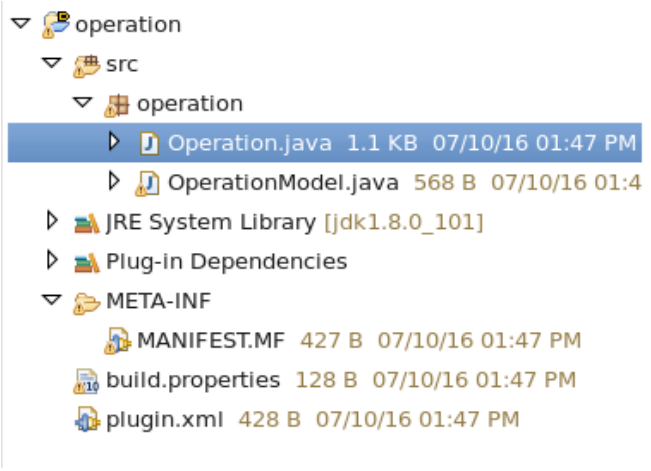
An optional # SHAPE entry if the data should not just be stacked

Then # FILE_NAME followed by the name of the file (found in the directory specified by DIR_NAME). If the files are spread across multiple directories, DIR_NAME can be the root directory (/) then the FILE_NAME would be home/me/data/tiffs/00001.tif etc

https://github.com/belkassaby/nobugs_dde

Create a custom derivative operation 1

- Close the new DAWN instance and go to the DAWN dev instance
- In the “Plug-in Development” perspective, select File->New->DAWN Plug-in project
- In the new window, select a project name, identifier, name and institute and choose the “org.dawnsci.analysis.api.operation” id
- Click “next” and give a description to your operation; press “Finish”
- In the new project created, you will fin two java classes, one class defining your operation and the other the operation model
- Open the operation java class



https://github.com/belkassaby/nobugs_dde

Create a custom derivative operation 2

- In the operation java class “process” method,
  - Convert IDataset to Dataset using org.eclipse.january.DatasetUtils.convertToDataset()
  - apply the org.eclipse.january.dataset.Maths.derivative() to the input IDataset
  - Set the name of the derivative dataset
  - Return the new OperationData

```
Dataset converted = … ;
Dataset deriv = Maths…;
// set name of deriv
…
// return derivative
return new OperationData(…);
```

  - As a bonus you can use the model class generated to get the spread parameter the derivative method takes; this parameter can be an input in the model ui :

```
@OperationModelField(hint="Enter value for spread", label = "Value")
private int derivSpread = 1;
```

Solution

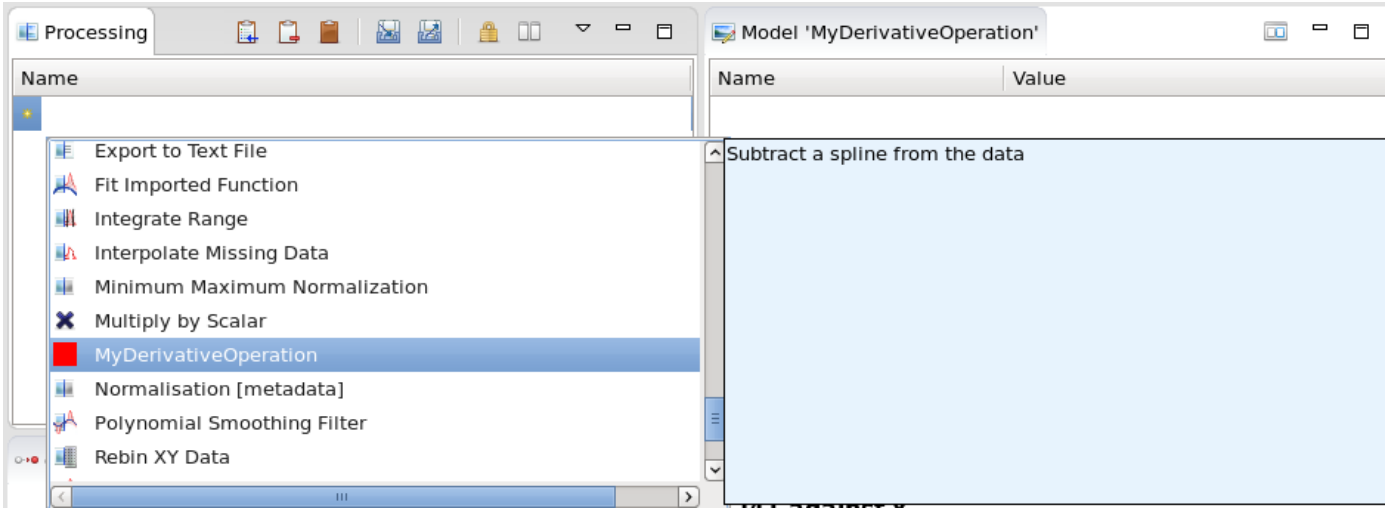
```
//In the Operation class:
@Override
protected OperationData process(IDataset input, IMonitor monitor) throws OperationException {
            Dataset data = DatasetUtils.convertToDataset(input);
            // retrieve the ui input from the model for the derivative spread parameter
            int spread = model.getDerivSpread();
            // apply derivative using Maths class in Eclipse January
            Dataset indices = data.getIndices().squeeze();
            Dataset deriv = Maths.derivative(indices, data, spread);
            deriv.setName("Derivative Data");
            copyMetadata(input, deriv);
            // return derivative
            return new OperationData(deriv);
}
//In the Model class:
@OperationModelField(hint="Enter value for operation", label = "Value")
private int derivSpread = 1;
public int getDerivSpread() {
            return derivSpread;
}
public void setDerivSpread(int derivSpread) {
            firePropertyChange("derivSpread", this.derivSpread, this.derivSpread = derivSpread);
}
```

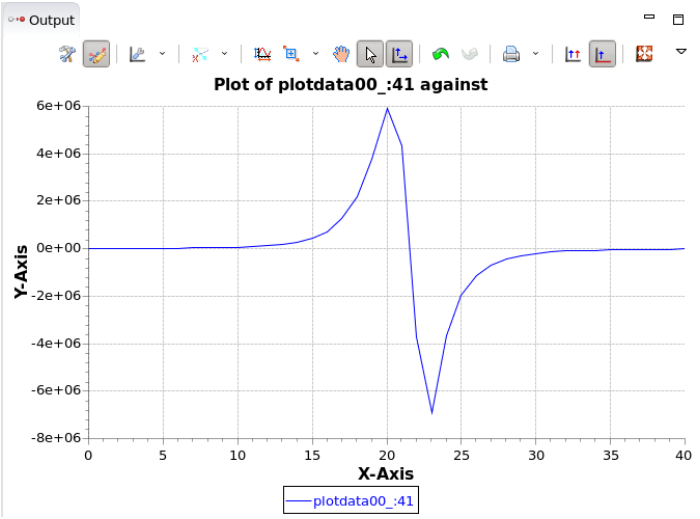
https://github.com/belkassaby/nobugs_dde

Use derivative operation created

- Build your projects by typing the following keys: CTRL+B
- Run the new debug instance of DAWN by pressing the launch button

- Drag and drop the csv file to the data slice view
- Select the custom derivative operation created
- Apply the operation to the processing workflow and visualize your data in the output





https://github.com/belkassaby/nobugs_dde