June 2022 Software

Dataset file format

RS²D
ZA des Maréchaux
13 rue Vauban
67450 Mundolsheim
France
www.rs2d.com

Contents

1 Files

Spinlab datasets are stored in two files:

- header.xml - a xml file containing all acquisition parameters
- data.dat – a binary file containing the data points

2 Header

The header file is a simple xml file containing all parameters and their attributes. Most of these attributes can be ignored when reading the data, only the parameter name and value are needed.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<header>
    <params>
        <entry>
            <key>SEQUENCE_TIME</key>
            <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="numberParam">
                <name>SEQUENCE_TIME</name>
                <description></description>
                <displayedName>SEQUENCE_TIME</displayedName>
                <locked>false</locked>
                <lockedToDefault>false</lockedToDefault>
                <group>Delay</group>
                <category>Acquisition</category>
                <rolesEnum>User</rolesEnum>
                <value>44.816384</value>
                <defaultValue>0.0</defaultValue>
                <restrictedToSuggested>false</restrictedToSuggested>
                <maxValue>1.0E9</maxValue>
                <minValue>0.0</minValue>
                <numberEnum>Time</numberEnum>
            </value>
        </entry>
        <entry>
            <key>DYNAMIC_MIN_TIME</key>
            <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="booleanParam">
                <name>DYNAMIC_MIN_TIME</name>
                <description>Execute the dynamic scan without delay between
repetitions</description>
                <displayedName>DYNAMIC_MIN_TIME</displayedName>
                <locked>false</locked>
                <lockedToDefault>false</lockedToDefault>
                <group>Delay</group>
                <category>Acquisition</category>
                <rolesEnum>User</rolesEnum>
                <value>true</value>
                <defaultValue>false</defaultValue>
            </value>
        </entry>
      <entry>
            <key>ACQUISITION_TIME_OFFSET</key>
            <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="listNumberParam">
                <name>ACQUISITION_TIME_OFFSET</name>
                <description>Relative acquisition start times in Dynamic or
MultiSeries scans</description>
                <displayedName>ACQUISITION_TIME_OFFSET</displayedName>
                <locked>false</locked>
                <lockedToDefault>false</lockedToDefault>
                <group>Delay</group>
                <category>Acquisition</category>
                <rolesEnum>User</rolesEnum>
```

```
                <value>0.0</value>
                <defaultValue>0.0</defaultValue>
                <maxValue>1.0E9</maxValue>
                <minValue>0.0</minValue>
                <numberEnum>Time</numberEnum>
            </value>
        </entry>
        <!-- ... snip lots of parameters -->
    </params>
    <variationParams1D/>
    <variationParams2D/>
    <variationParams3D/>
    <variationParams4D/>
</header>
```

The “variationParams1D” to “variationParams4D” elements can also contains parameters, with the same xml representation.

2.1 XPath

Here is a XPath expression to list all parameter names:

```
/header/params/entry/key
```

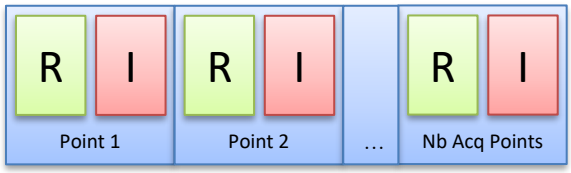
Here is a XPath expression to get a parameter value. In this example, the parameter RECEIVER_COUNT is used:

```
/header/params/entry/key[text()='RECEIVER_COUNT']/../value/value
```

3 Binary data

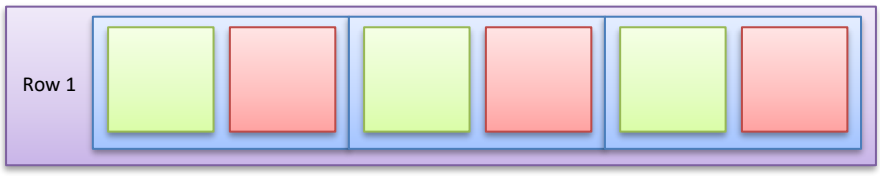
The binary “data.dat” file contains all points stored as 32 bit floating points using the standard IEEE 754 representation, using the big endian byte order. There is no separator between points, rows, slices, … Therefore, the reader must know beforehand the size of each dimensions, specified in the header in the MATRIX_DIMENSION_1D to MATRIX_DIMENSION_4D parameters and the number of receivers, specified in the RECEIVER_COUNT parameter.

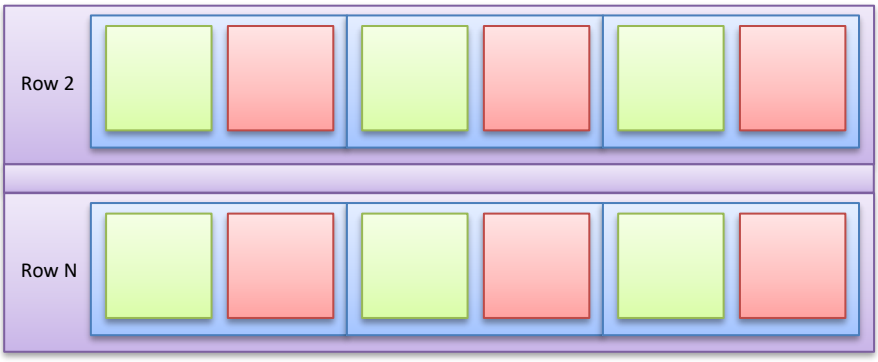
For each point, its real part and its imaginary part are stored. Thus, a row is stored as a list of points.



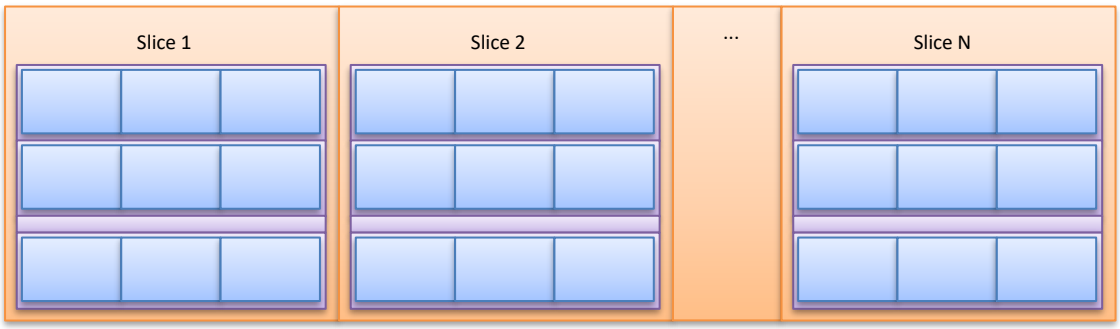
The number of point per row is defined in the header, with the MATRIX_DIMENSION_1D parameter.

The number of rows is defined with the MATRIX_DIMENSION_2D parameter.



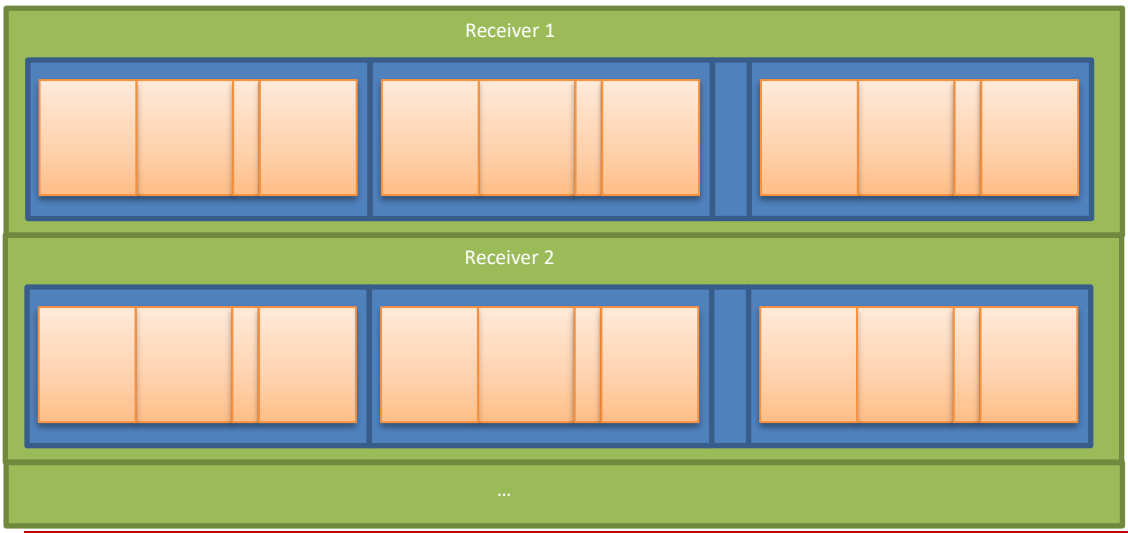


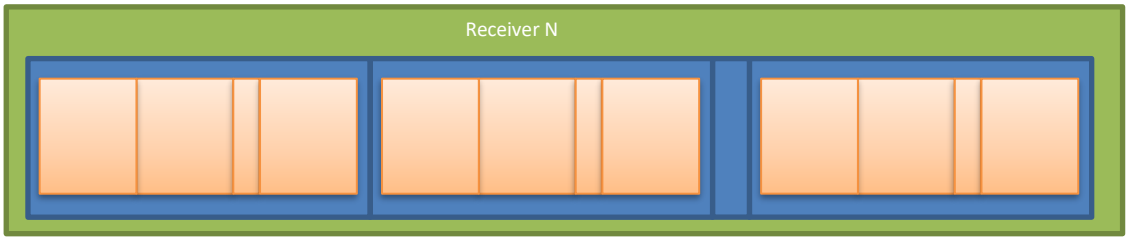
This block corresponds to a slice. The number of slices is defined with the MATRIX_DIMENSION_3D parameter.



This block corresponds to a volume. The number of volumes is defined with the MATRIX_DIMENSION_4D parameter.

This block contains all the data for a receiver. The number of receivers is defined in the RECEIVER_COUNT parameter.





Finally, this block of receivers makes the whole dataset.

4 Pseudo code

Here is some pseudo code to get a parameter:

```
function getParamValue(xml, paramName) {
  expr = "/header/params/entry/value/name[text()='" + paramName + "']/../value";
  return xpath(xml, expr);
}
```

Here is some pseudo code to read a dataset:

```
header = open("header.xml");
receiverCount = getParamValue(header, "RECEIVER_COUNT");
dimension1d = getParamValue(header, "MATRIX_DIMENSION_1D");
dimension2d = getParamValue(header, "MATRIX_DIMENSION_2D");
dimension3d = getParamValue(header, "MATRIX_DIMENSION_3D");
dimension4d = getParamValue(header, "MATRIX_DIMENSION_4D");

dataset = open("data.dat");

for(int rx=0 ;rx < receiverCount ; rx++) {
  for(int volume=0; volume < dimension4d ; volume++) {
    for(int slice=0 ; slice < dimension3d ; slice++) {
      for(int row=0 ; row < dimension2d ; row++) {
        for(int point=0 ; point < dimension1d ; point++) {
          float real = readFloat(dataset) ;
          float imaginary = readFloat(dataset) ;
          //do something here with the points
        }
      }
    }
  }
}
```

5 Java example

Here is a complete Java example, matching the previous pseudo code:

```
import org.w3c.dom.Document;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.xpath.XPath;
import javax.xml.xpath.XPathFactory;
import java.io.File;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

public class DatasetReadSample {
    private static String getParamValue(Document xml, String paramName) throws Exception {
        String expression = "/header/params/entry/key[text()='" + paramName +
"']/../value/value";

        XPath path = XPathFactory.newInstance().newXPath();
        return path.evaluate(expression, xml.getDocumentElement());
    }

    private static void readDataset(File headerFile, File dataFile) throws Exception {
        Document xml =
DocumentBuilderFactory.newInstance().newDocumentBuilder().parse(headerFile);
        int receiverCount = Integer.valueOf(getParamValue(xml, "RECEIVER_COUNT"));
        int dimension1d = Integer.valueOf(getParamValue(xml, "MATRIX_DIMENSION_1D"));
        int dimension2d = Integer.valueOf(getParamValue(xml, "MATRIX_DIMENSION_2D"));
        int dimension3d = Integer.valueOf(getParamValue(xml, "MATRIX_DIMENSION_3D"));
        int dimension4d = Integer.valueOf(getParamValue(xml, "MATRIX_DIMENSION_4D"));

        RandomAccessFile dataset = new RandomAccessFile(dataFile, "r");
        ByteBuffer buffer = dataset.getChannel().map(FileChannel.MapMode.READ_ONLY, 0,
dataset.getChannel().size());

        for(int rx=0 ;rx < receiverCount ; rx++) {
            System.out.println("======== rx: " + rx);
            for(int volume=0; volume < dimension4d ; volume++) {
                System.out.println("====== volume: " + volume);
                for(int slice=0 ; slice < dimension3d ; slice++) {
                    System.out.println("==== slice: " + slice);
                    for(int row=0 ; row < dimension2d ; row++) {
                        System.out.println("== row: " + row);
                        for(int point=0 ; point < dimension1d ; point++) {
                            float real = buffer.getFloat();
                            float imaginary = buffer.getFloat();
                            //do something here with the points
                            System.out.println(real + ", " + imaginary);
                        }
                    }
                }
            }
        }
    }

    public static void main(String[] args) throws Exception {
        File datasetDir = new File("...");
        File headerFile = new File(datasetDir, "header.xml");
        File dataFile = new File(datasetDir, "data.dat");

        readDataset(headerFile, dataFile);
    }
}
```

6 Python example

Here is the same example using the Python language:

```
from xml.etree.ElementTree import parse
from struct import unpack

def get_param(xml, param_name):
    expr = "./params/entry[key='" + param_name + "']/value/value"
    return xml.getroot().findtext(expr)

def read_dataset(header_path, data_path):
    xml = parse(header_path)
    receiver_count = int(get_param(xml, 'RECEIVER_COUNT'))
    dimension1d = int(get_param(xml, 'MATRIX_DIMENSION_1D'))
    dimension2d = int(get_param(xml, 'MATRIX_DIMENSION_2D'))
    dimension3d = int(get_param(xml, 'MATRIX_DIMENSION_3D'))
    dimension4d = int(get_param(xml, 'MATRIX_DIMENSION_4D'))

    data_file = open(data_path, mode='rb')
    for rx in range(0, receiver_count):
        print("====== rx: " + str(rx))
        for l in range(0, dimension4d):
            print("==== 4d: " + str(l))
            for k in range(0, dimension3d):
                print("== 3d: " + str(k))
                for j in range(0, dimension2d):
                    print("== row(2d):" + str(j))
                    for i in range(0, dimension1d):
                        real, imaginary = unpack('>ff', data_file.read(8))
                        # do something here
                        print(str(real) + ", " + str(imaginary))


dir_path = '...'
header_path = dir_path + '/header.xml'
data_path = dir_path + '/data.dat'
read_dataset(header_path, data_path)
```