McStas and McStasScript

3rd November, 2019

Mads Bertelsen, Thomas Holm Ro
European Spallation Source

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStas - Overview

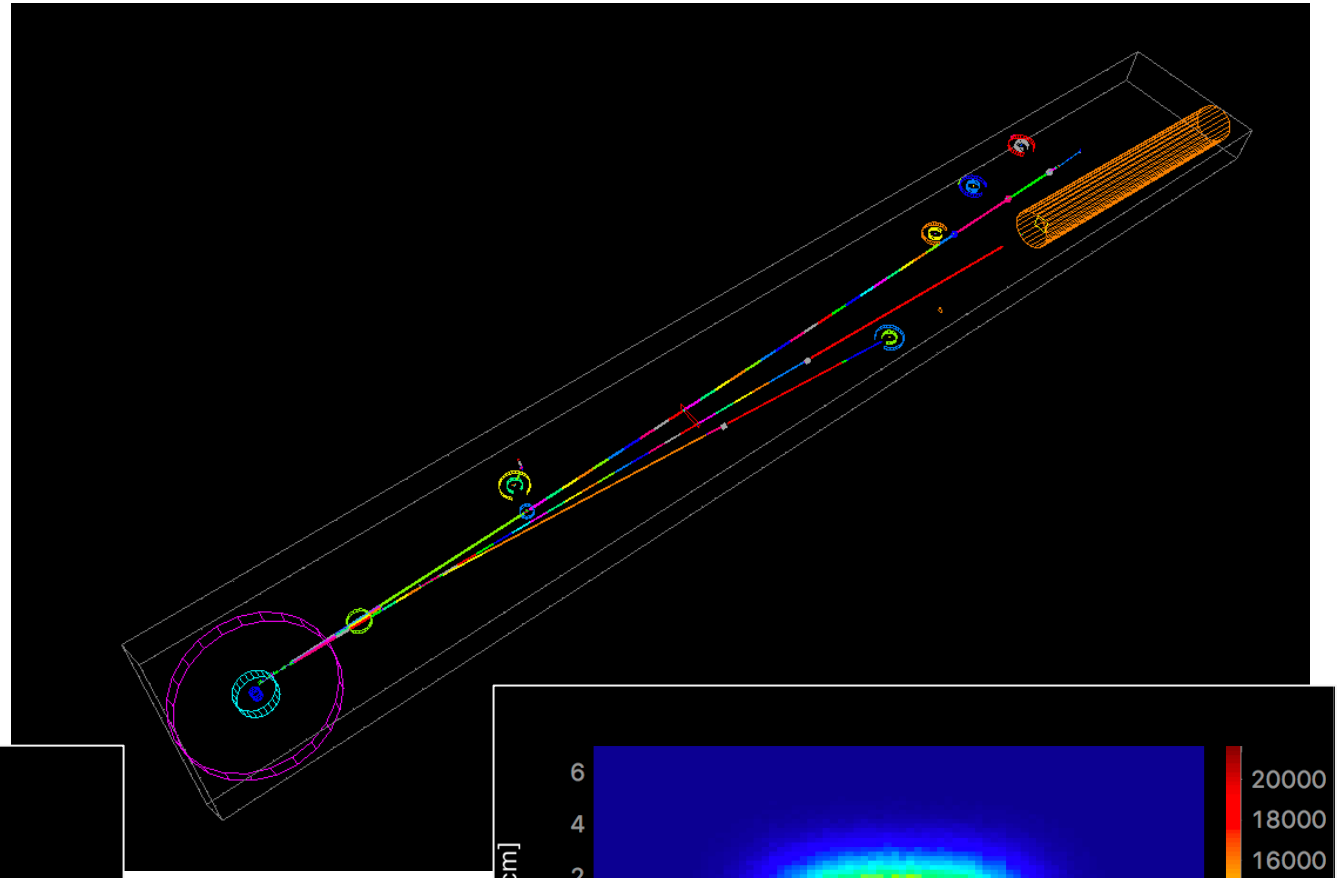
- Name: Monte Carlo Simulation of Triple Axis Spectrometer
- Now simulates all kinds of neutron scattering instrumentation
- From neutron source to detector

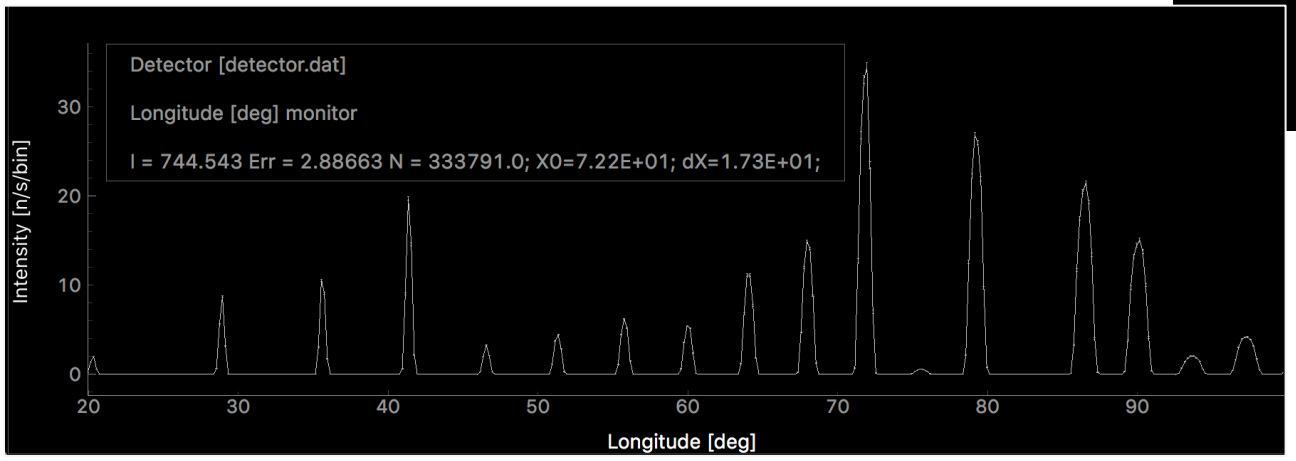
- Core developers at DTU / ESS
- Open source community project, many contributions

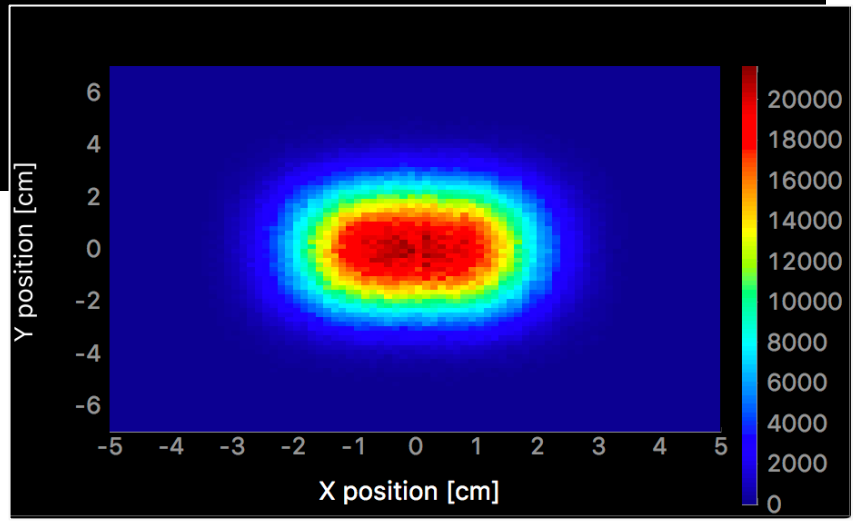
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStas - Overview

- 3D model and physics
- Construct instrument from pool of components
- Each component describe spatially separate part of the instrument



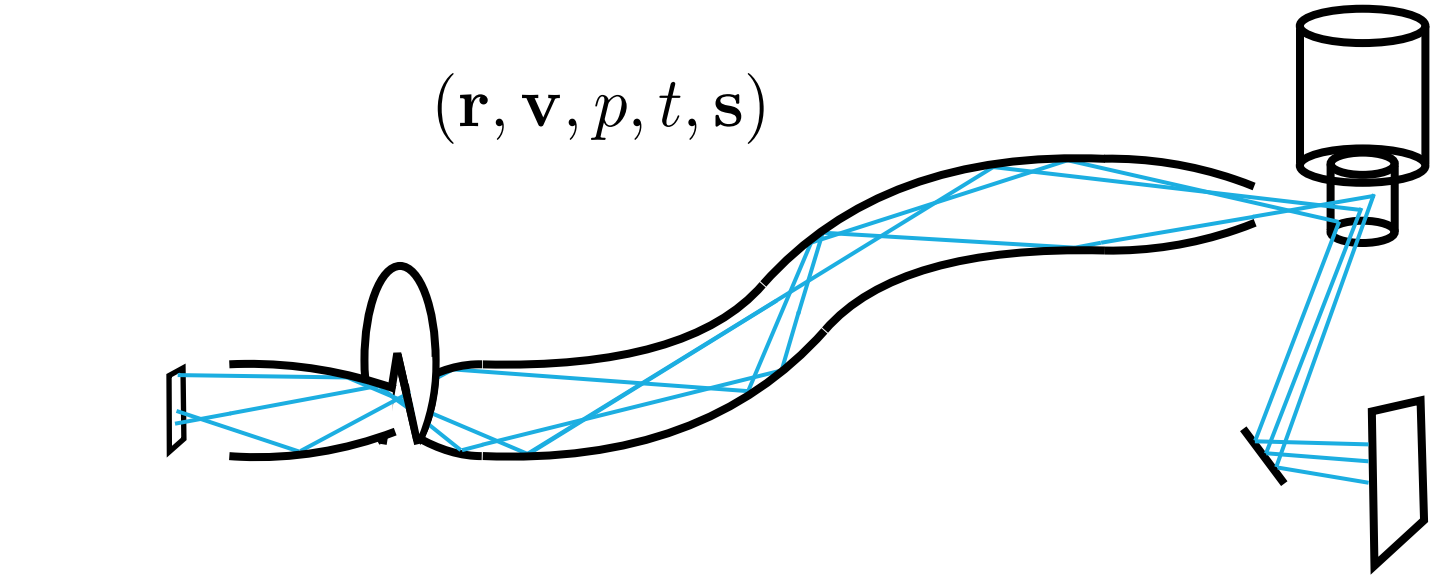


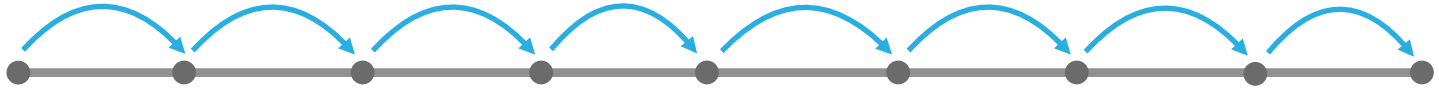


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStas - Components

- **Sources**
  - From file
  - Analytical
- **Optics**
  - Guides
  - Choppers
  - Monochromator
- **Sample**
  - Crystaline
  - Inelastic
  - Large scale structures
- **Monitors**





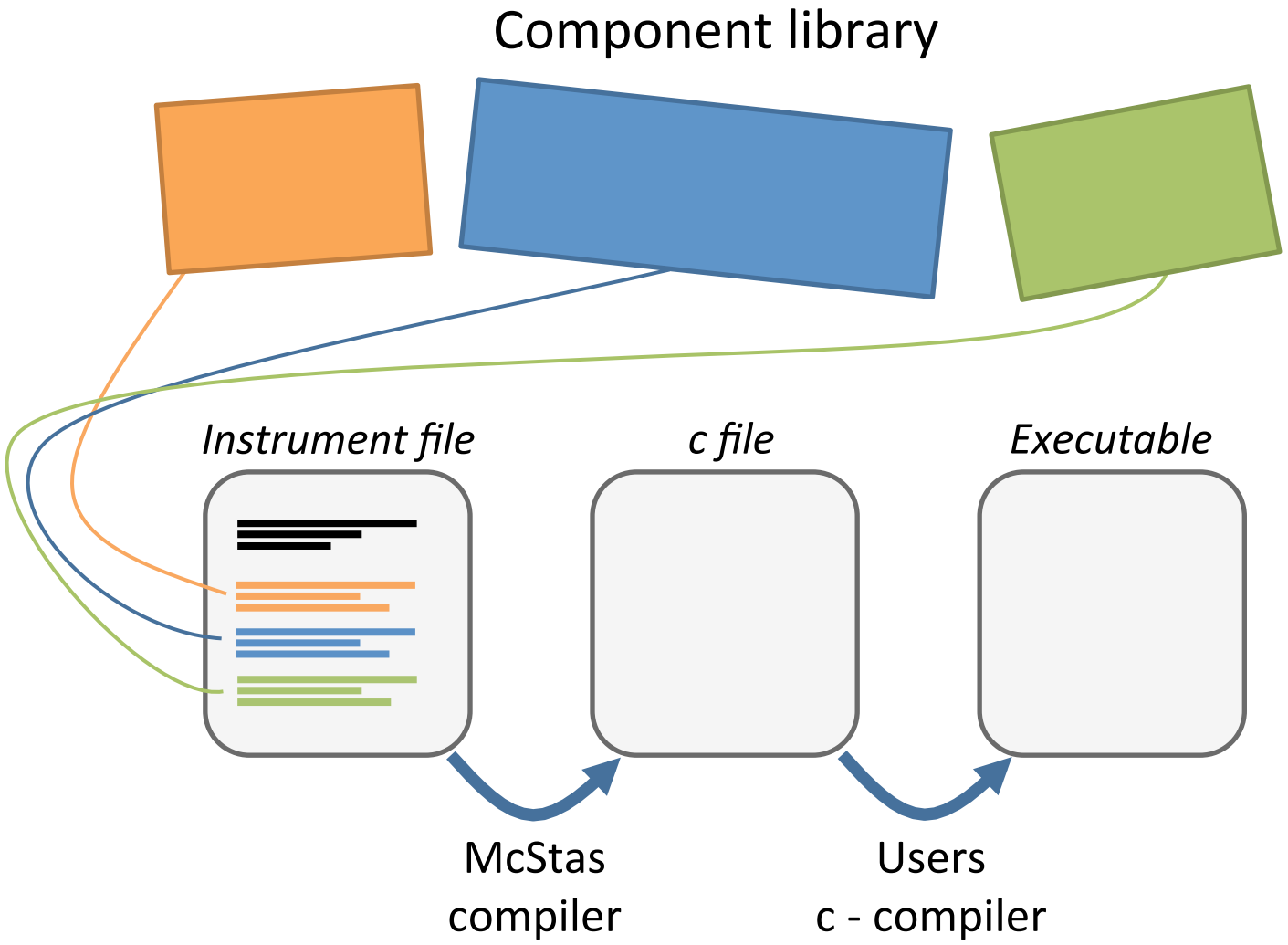
Sequence of components

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStas – Technical

- **Instrument file (meta language)**
  1. Instrument definition
  2. Declare section
  3. Initialize section
  4. Trace section
  5. Finally section

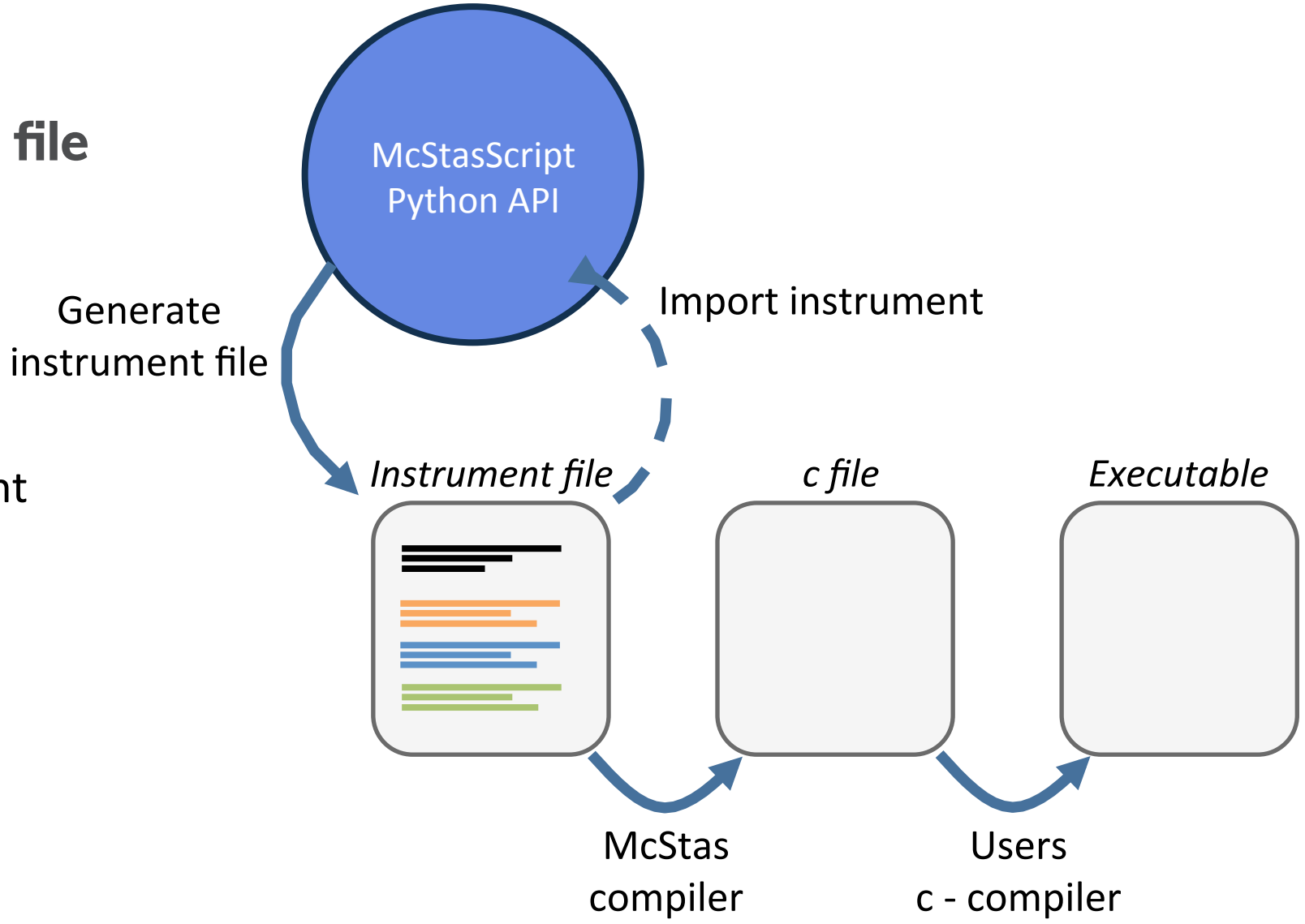
- **Instrument to data with one utility**
  - mcrun on command line
  - mcgui graphical user interface



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

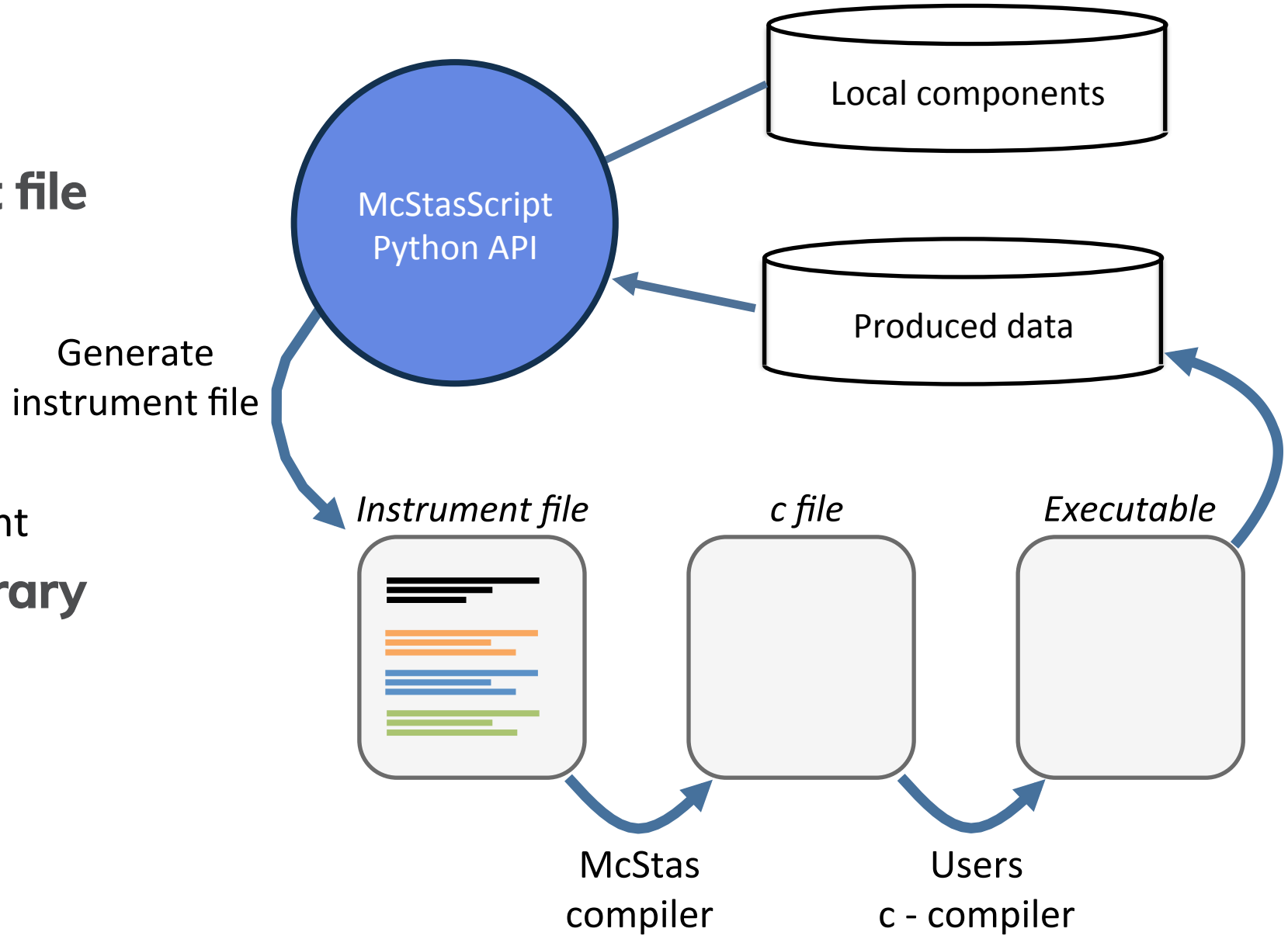
- **Generate instrument file from python API**
- **Advantages:**
  - for loops
  - Functions
  - Flexible code placement



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

- **Generate instrument file from python API**
- **Advantages:**
  - for loops
  - Functions
  - Flexible code placement
- **Read component library**
- **Execute simulation**
- **Load data**



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

- **Installation**

  Python –m pip install McStasScript

- **Configuration**
  - Set path to mcrun utility
  - Set path to components

```
In [2]: from mcstasscript.interface import instr, plotter, functions
```

```
In [3]: my_config = functions.Configurator()
        my_config.set_line_length(72)
        my_config.set_mcrun_path("/Applications/McStas-2.5.app/Contents/Resources/mcstas/2.5/bin/")
        my_config.set_mcstas_path("/Applications/McStas-2.5.app/Contents/Resources/mcstas/2.5/")
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

- Create a new instrument object

```
In [3]: demo = instr.McStas_instr("presentation_demo")
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

- Add instrument parameter
- Add a component
- Print component state

```
In [4]: demo.add_parameter("wavelength", value=4.0, comment="[AA] Wavelength simulated")
        demo.show_parameters()
```

```
wavelength  =  4.0  // [AA] Wavelength simulated
```

```
In [5]: src = demo.add_component("source", "Source_simple")
        src.xwidth = 0.11
        src.yheight = 0.11
        src.focus_xw = 0.05
        src.focus_yh = 0.05
        src.dist = 2.0
        src.lambda0 = "wavelength"
        src.dlambda = "0.1*wavelength"
        src.flux = 1E13
```

```
In [6]: src.print_long()
```

```
COMPONENT source = Source_simple
  yheight = 0.11 [m]
  xwidth = 0.11 [m]
  dist = 2.0 [m]
  focus_xw = 0.05 [m]
  focus_yh = 0.05 [m]
  lambda0 = wavelength [AA]
  dlambda = 0.1*wavelength [AA]
  flux = 10000000000000.0 [1/(s*cm**2*st*energy unit)]
AT [0, 0, 0] ABSOLUTE
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

- **Include code in the simulation, declare and initialize**

```
In [7]: monochromator_Q = 1.8734
        demo.add_declare_var("double", "mono_Q", value=monochromator_Q)
        demo.add_declare_var("double", "wavevector")
        demo.append_initialize("wavevector = 2*PI/wavelength;")

        demo.add_declare_var("double", "mono_rotation")
        demo.append_initialize("mono_rotation = asin(mono_Q/(2.0*wavevector))*RAD2DEG;")
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

- Include code in the simulation, declare and initialize
- Use in component

```
In [7]: monochromator_Q = 1.8734
        demo.add_declare_var("double", "mono_Q", value=monochromator_Q)
        demo.add_declare_var("double", "wavevector")
        demo.append_initialize("wavevector = 2*PI/wavelength;")

        demo.add_declare_var("double", "mono_rotation")
        demo.append_initialize("mono_rotation = asin(mono_Q/(2.0*wavevector))*RAD2DEG;")

In [8]: mono = demo.add_component("mono", "Monochromator_flat")

In [9]: mono.zwidth = 0.05
        mono.yheight = 0.05
        mono.Q = monochromator_Q
        mono.set_AT([0, 0, 2], RELATIVE="source")
        mono.set_ROTATED([0, "mono_rotation", 0], RELATIVE="source")
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

- Add a monitor to collect data

In [10]:
```
beam_direction = demo.add_component("beam_dir", "Arm", AT_RELATIVE="mono")
beam_direction.set_ROTATED([0, "mono_rotation", 0], RELATIVE="mono")

detector = demo.add_component("detector", "PSD_monitor")
detector.xwidth = 0.1
detector.yheight = 0.1
detector.filename = "\"data.dat\""
detector.nx = 100
detector.ny = 100
detector.set_AT([0,0,2], RELATIVE="beam_dir")
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

- Run the simulation from python
- Data returned as simple object with numpy data arrays

```
In [16]: data = demo.run_full_instrument(ncount=1E7, foldername="demo", increment_folder_name=True,
                                         parameters={"wavelength" : 3.5})
```

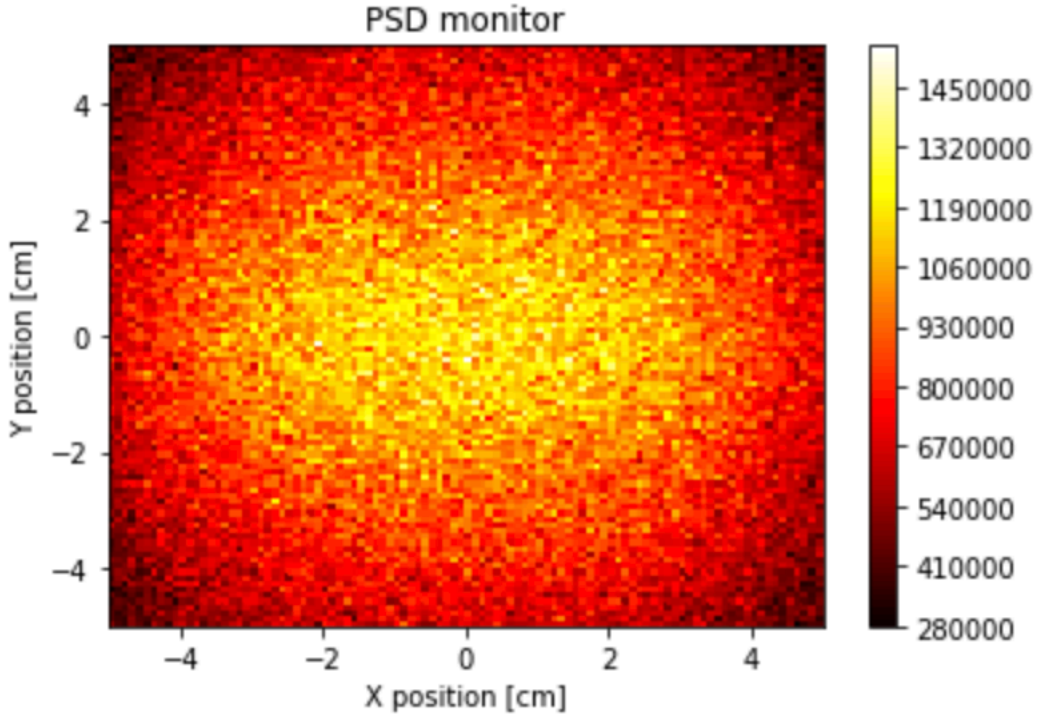
```
INFO: Using directory: "demo_0"
INFO: Regenerating c-file: presentation_demo.c
CFLAGS=
INFO: Recompiling: ./presentation_demo.out
INFO: ===
INFO: Placing instr file copy presentation_demo.instr in dataset demo_0

Detector: detector_I=8.35572e+09 detector_ERR=2.10199e+07 detector_N=158020 "data.dat"
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

```
In [12]: plotter.make_plot(data)
```

```
number of elements in data list = 1
Plotting data with name detector
```

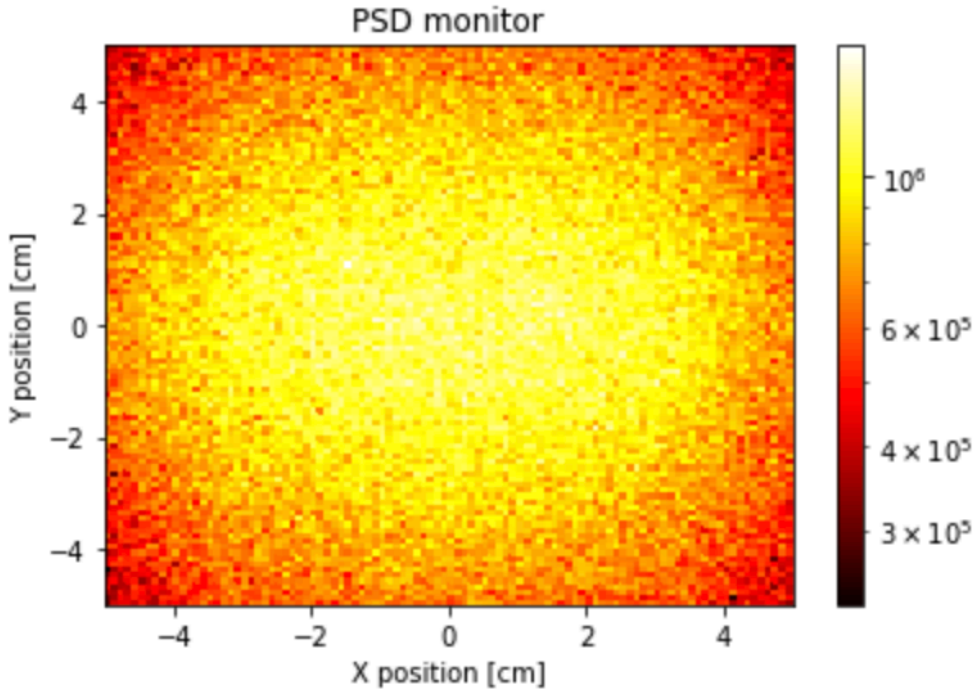


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

```
In [13]: functions.name_plot_options("detector", data, log=True)

In [14]: plotter.make_plot(data)
```

```
number of elements in data list = 1
Plotting data with name detector
```



Plot preferences included in data object

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

In [1]:
```
from mcstasscript.interface import instr, plotter, functions, reader
```

In [2]:
```
InstrReader = reader.McStas_file("presentation_demo.instr")
```

In [3]:
```
InstrReader.write_python_file("generated_presentation.py", force=True)
```

In [4]:
```
demo = instr.McStas_instr("generated_presentation_demo")
InstrReader.add_to_instr(demo)
```

In [5]:
```
demo.print_components()
```

```
source    Source_simple       AT      (0, 0, 0)              ABSOLUTE
                              ROTATED (0, 0, 0)              ABSOLUTE
mono      Monochromator_flat  AT      (0, 0, 2)              RELATIVE source
                              ROTATED (0, mono_rotation, 0)  RELATIVE source
beam_dir  Arm                 AT      (0, 0, 0)              RELATIVE mono
                              ROTATED (0, mono_rotation, 0)  RELATIVE mono
detector  PSD_monitor         AT      (0, 0, 2)              RELATIVE beam_dir
                              ROTATED (0, 0, 0)              RELATIVE beam_dir
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

McStasScript

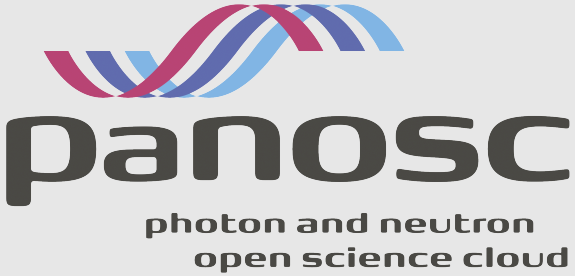
- Python API as an alternative to standard McStas workflow
- Still work to do on API, may rename some methods / functions / keywords
- Improvements on plotting and data handling
- Review pending from co-workers
- Continous integration will be set up

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852

PaNOSC WP5 deliverables

- **openPMD**
  - Has been postponed slightly in favor of building an API
  - Have example dataset and extension document
  - Need to make McStas in/out components
- **Documented API**
  - Have an API, will need further improvements
  - Both docstrings and pdf documentation
  - 199 unit tests of core functionallity, lacks test of plotting / instrument reader
  - Wish to add continous integration
- **Release of examples in Jupyter Notebook**
  - Can generate python scripts for McStas instruments (Hundreds available)
  - Can create more examples that emphesises python advantages

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852



Thank you

Mads.Bertelsen@esss.se

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852