

New Allpix Squared Documentation and Website

Stephan Lachnit

3rd Allpix Squared User Workshop (11.05.2021)



Problems with the current documentation



- ▶ Looks awesome in PDF, but website has rendering issues
- ▶ \LaTeX is not WYSIWYG:
 - ▶ Needs build to review changes e.g. from merge requests
 - ▶ Documentation can't be previewed in GitLab nicely
- ▶ Inconsistencies:
 - ▶ Different formats: some parts are Markdown, some \LaTeX
 - ▶ Separated website: hugo (news) vs \LaTeX (docs) vs Doxygen (API ref)

Return to website

$$f(x, y, w) = \arctan\left(\frac{w_x y_1}{w_x \sqrt{x_1^2 + y_1^2 + d^2}}\right) + \arctan\left(\frac{w_x y_2}{w_x \sqrt{x_1^2 + y_2^2 + d^2}}\right) - \arctan\left(\frac{w_x y_1}{w_x \sqrt{x_2^2 + y_1^2 + d^2}}\right) - \arctan\left(\frac{w_x y_2}{w_x \sqrt{x_2^2 + y_2^2 + d^2}}\right)$$

with $w_{x,2} = x \pm \frac{w_x}{2}$, $y_{1,2} = y \pm \frac{w_y}{2}$. The parameters ' $w_{x,y}$ ' indicate the size of the collection electrode (i.e. the implant), ' V_w ' is the potential of the electrode and d is the thickness of the sensor.

Parameters

- `model`: Type of the weighting potential model, either `mesh` or `pad`.
- `file_name`: Location of file containing the weighting potential in one of the supported field file formats. Only used if the `model` parameter has the value `mesh`.
- `ignore_field_dimensions`: If set to true, a wrong dimensionality of the input field is ignored, otherwise an exception is thrown. Defaults to false.
- `output_plots`: Determines if output plots should be generated. Disabled by default.
- `output_plots_steps`: Number of bins along the z-direction for which the weighting potential is evaluated. Defaults to 500 bins and is only used if `output_plots` is enabled.
- `output_plots_position`: 2D Position in x and y at which the weighting potential is evaluated along the z-axis. By default, the potential is plotted for the position in the pixel center, i.e. (0, 0). Only used if `output_plots` is enabled.

Usage

An example to add a weighting potential from a field data file to the detector called "dut" is given below.

```
[WeightingPotentialReader]
name = "dut"
model = "mesh"
file_name = "example_weighting_field.apf"
```

[\[next\]](#) [\[prev\]](#) [\[prev-tail\]](#) [\[front\]](#) [\[up\]](#)

This manual is licensed under a Creative Commons Attribution 4.0 International License

- 8.1 CSADigitizer
- 8.2 CapacitiveTransfer
- 8.3 CarryvreckanWriter
- 8.4 DatabaseWriter
- 8.5 DefaultDigitizer
- 8.6 DepositionCosmics
- 8.7 DepositionGeant4
- 8.8 DepositionPointCharge
- 8.9 DepositionReader
- 8.10 DetectorHistogrammer
- 8.11 DopingConcentrationReader
- 8.12 ElectricFieldReader
- 8.13 GDMLOutputWriter
- 8.14 GenericPropagation
- 8.15 GeometryBuilderGeant4
- 8.16 InducedTransfer
- 8.17 LCIOWriter
- 8.18 MagneticFieldReader
- 8.19 ProjectionPropagation
- 8.20 PulseTransfer
- 8.21 RCEWriter
- 8.22 ROOTObjectReader
- 8.23 ROOTObjectWriter
- 8.24 SimpleTransfer

Figure 1: Sidebar not scrolling down: not all entries are visible

Return to website

detector relevant for this simulation, i.e. the sensing volume, the difference between effective and total concentration is expected to be negligible. Therefore the two values are treated as equivalent throughout the lifetime models and the doping concentration is taken as the absolute value $N = |N_D - N_A|$.

Whether a charge carrier has recombined with the lattice is calculated for every step of the simulation using the relation

$$p < 1 - e^{-dt/\tau(N)} \quad (6.8)$$

where p is a recombination probability, drawn from a uniform distribution with $[0, 1]$, dt is the last time step of the charge carrier motion and τ the lifetime for the local doping concentration calculated by the models described in the following. If Equation (6.8) evaluates to false, the charge carrier still exists, if it evaluates to true it has been recombined with the lattice.

Finite charge carrier lifetime can be simulated by all propagation modules and comprise the following models:

6.2.1 Shockley-Read-Hall Recombination

This model describes the finite lifetime based on Shockley-Read-Hall or trap-assisted recombination of charge carriers with the lattice [31, 32]. The lifetime is calculated using the Shockley-Read-Hall relation as given by [33]:

$$\tau(N) = \frac{\tau_0}{1 + \frac{N}{N_0}} \quad (6.9)$$

where τ_0 and N_0 are reference lifetime and doping concentration, for electrons and holes respectively. The parameter values implemented in Allpix² are taken from [34] as:

6.2.2 Auger Recombination

At high doping levels exceeding

Figure 2: Some equation not rendering, preventing to scroll to the bottom

Return to website

How do I access data stored in a file produced with the ROOTObjectWriter from an analysis script?

Allpix² uses ROOT trees to directly store the relevant C++ objects as binary data in the file. This retains all information present during the simulation run, including relations between different objects such as assignment of Monte Carlo particles. In order to read such a data file in an analysis script, the relevant C++ library as well as its header have to be loaded.

In ROOT this can be done interactively by loading a data file, the necessary shared library objects and a macro for the analysis:

```
5 root->load_data_file root 2 root [1] 1 ->path/to/your/allpix-squared/16/16/Allpix2/objects.so 3 root [2] 6 -analysis/Macro.C
root [3] readTree -d160 -detector1
```

A simple macro for reading DepositedCharges from a file and displaying their position is presented below:

```
1 #include <TFile> 2 #include <TTree> 3 4 // FIXME: adapt path to the include file of APSQ installation 5 #include "path
to/your/allpix-squared/DepositedCharges.hpp" 6 7 // Read data from tree 8 void readTree(TFile* file, std::string detector) { 9 30
// Read tree of deposited charges: 11 TTree* dc_tree = static_cast<TTree*>(file->Get("DepositedCharge")); 12
if(!dc_tree) { 13 throw std::runtime_error("Could not read tree"); 14 } 15 16
// Find branch for the detector requested: 17 TBranch* dc_branch = dc_tree->FindBranch("detector_c_str"); 18
if(!dc_branch) { 19 throw std::runtime_error("Could not find detector branch"); 20 } 21 22
// Allocate object vector and link to ROOT branch: 23 std::vector<allpix::DepositedCharge> deposited_charges; 24
dc_branch->SetObjectStreamer(deposited_charges); 25 26 // Go through the tree event-by-event 27
for(int i = 0; i < dc_tree->GetEntries(); ++i) { 28 dc_tree->GetEntry(); 29 // Loop over all deposited charge objects
30 for(auto& charge : deposited_charges) { 31 std::cout << "Event " << i << ", " << "32
<< "charge = " << charge->getCharge() << ", " << "33
<< "position = " << charge->getGlobalPosition()
34
<< std::endl; 35 } 36 } 37 }
```

A more elaborate example for a data analysis script can be found in the tools directory of the repository and in Section 13.5 of this user manual. Scripts written in both C++ and in Python are provided.

How can I convert data from the ROOTObject format to other formats?

Since the ROOTObject format is the native format of Allpix², the stored data can be read into the framework again. To convert it to another format, a simple pseudo-simulation setup can be used, which reads in data with one module and stores it with another.

In order to convert for example from ROOTObjects to the data format used by the Corryvreckan reconstruction framework, the following configuration could be used:

```
1 [Allpix 2 number_of_events = 99999999 3 detector_file = "telescope.conf" 4 random_seed_core = 0 5 6
(ROOTObjectReader) 7 file_name = "input_data_rootobjects.root" 8 9 [CorryvreckanWriter] 10
file_name = "output_data_corryvreckan.root" 11 reference = "mydetector"
```

Table of Contents:

- 12.1 Installation & Usage
- 12.2 Configuration
- 12.3 Detector Models
- 12.4 Data Analysis
- 12.5 Development
- 12.6 Miscellaneous

Figure 3: Code blocks missing newlines, line numbers are in copied code

Current website setup



- ▶ Main website (landing page, news, etc) via hugo
- ▶ API reference created with Doxygen and rendered to HTML
- ▶ Main documentation in \LaTeX
- ▶ Module & example READMEs converted from Markdown to \LaTeX via pandoc
- ▶ Final \LaTeX document rendered to HTML page via TeX4ht
- ▶ HTML documentation and API reference pasted into hugo webdirectory

New Website



- ▶ Documentation ported to Markdown [1]
- ▶ Using hugo for news *and* documentation
- ▶ Docsy theme [2] (made for technical documentations)
- ▶ Consistent theme across website, no more rendering issues
- ▶ New features, e.g. pretty warning boxes, search

The screenshot shows a web browser window displaying the 'Modules' page of the Allpix Squared documentation. The browser's address bar shows 'localhost:1313/docs/07_modules/'. The page has a teal header with the 'Allpix Squared' logo and navigation links for 'About', 'Documentation', 'Blog', and 'Community'. A search bar is located in the top right of the header. On the left side, there is a scrollable sidebar titled 'Modules' containing a list of module names: CapacitiveTransfer, CorryvreckanWriter, CSADigitizer, DatabaseWriter, DefaultDigitizer, DepositionCosmics, DepositionGeant4, DepositionPointCharge, DepositionReader, Detector-Histogrammer, DopingProfileReader, Dummy, ElectricFieldReader, GDMLOutputWriter, GenericPropagation, GeometryBuilderGeant4, InducedTransfer, LQIOWriter, MagneticFieldReader, ProjectionPropagation, PulseTransfer, RCEWriter, ROOTObjectReader, ROOTObjectWriter, SimpleTransfer, TextWriter, and TransientPropagation. The main content area is titled 'Modules' and contains a description of all available modules, followed by a list of module descriptions with their names in blue links: CapacitiveTransfer (Transfer with cross-coupling between pixels), CorryvreckanWriter (Writes pixel hits in the Corryvreckan format), CSADigitizer (Digitizer emulating a Charge Sensitive Amplifier), DatabaseWriter (Writes simulation objects to a PostgreSQL database), DefaultDigitizer (Digitizer that creates a signal proportional to the collected charge), DepositionCosmics (Energy deposition from cosmic rays), DepositionGeant4 (Energy deposition with Geant4), and DepositionPointCharge. A 'Print entire section' link is visible in the top right of the main content area.

Figure 4: Separate chapter overview, scrollable sidebar

The screenshot shows a web browser displaying the Allpix Squared documentation page for 'Auger Recombination'. The page has a teal header with navigation links: About, Documentation, Blog, and Community. A search bar is present in the top right. On the left, there is a sidebar with a search box and a 'Documentation' section containing a list of topics like Introduction, Installation, Getting Started, Framework, Physics Models & Material Properties, Charge Carrier Lifetime & Recombination, and Objects. The main content area is titled 'Auger Recombination' and contains the following text:

At high doping levels exceeding $5 \times 10^{18} \text{ cm}^{-3}$ [fossun-lee], the Auger recombination model becomes increasingly important. It assumes that the excess energy created by electron-hole recombinations is transferred to another electron (e-e-h process) or another hole (e-h-h process). The total recombination rate is then given by [kerl]:

$$R_{\text{Auger}} = C_n n^2 p + C_p n p^2$$

where C_n and C_p are the Auger coefficients. The first term corresponds to the e-e-h process and the second term to the e-h-h process. In highly-doped silicon, the Auger lifetime for minority charge carriers can be written as:

$$\tau(N) = \frac{1}{C_a \cdot N^2}$$

where $C_a = C_n + C_p$ is the ambipolar Auger coefficient, taken as $C_a = 3.8 \times 10^{-21} \text{ cm}^6 \text{ s}^{-1}$ from [dziewior].

This recombination mode applies to minority charge carriers only, majority charge carriers have an infinite life time under this model and the recombination equation will always evaluate to true.

This model can be selected in the configuration file via the parameter `recombination_model = "auger"`.

Combined SRH/Auger Recombination

This model combines the charge carrier recombination from the Shockley-Read-Hall and the Auger model by inversely summing the individual lifetimes calculated by the models via

$$\tau^{-1}(N) = \tau_{\text{srh}}^{-1}(N) + \tau_a^{-1}(N) \quad \text{for minority charge carriers}$$

$$= \tau_{\text{srh}}^{-1}(N) \quad \text{for majority charge carriers}$$

where $\tau_{\text{srh}}(N)$ is the Shockley-Read-Hall and $\tau_a(N)$ the Auger lifetime. The latter is only taken into account for minority charge carriers.

This model can be selected in the configuration file via the parameter `recombination_model = "srh_auger"`.

Recombination with Constant Lifetimes

Some materials require constant lifetimes for charge carriers $\tau(N) = \tau$. This model requires the additional configuration keys `lifetime_electron` and `lifetime_hole` to be present in the module configuration section, for example:

Figure 5: No more broken math

The screenshot shows a web browser displaying the Allpix Squared documentation page. The page title is "Building Modules Outside the Framework". The left sidebar contains a navigation menu with categories like "Documentation", "Introduction", "Installation", "Getting Started", "Framework", "Physics Models & Material", "Properties", "Objects", "Modules", "Examples", "Module & Detector Development", "Building Modules Outside the Framework", "Implementing a New Module", "Writing Thread-Safe Code", "Adding a New Detector Model", "Development Tools & Continuous Integration", "Automated Testing", "FAQ", "Additional Tools & Resources", and "Appendix".

The main content area features the following text and code blocks:

Building Modules Outside the Framework

Allpix Squared provides CMake modules which allow to build modules for the framework outside the actual code repository. The macros required to build a module are provided through the CMake modules and are automatically included when using the `FIND_PACKAGE(Allpix)` CMake command. By this, modules can easily be moved into and out from the module directory of the framework without requiring changes to its `CMakeLists.txt`.

A minimal CMake setup for compiling and linking external modules to the core and object library of the Allpix Squared framework is the following:

```

CMAKE_MINIMUM_REQUIRED(VERSION 3.6.3 FATAL_ERROR)

FIND_PACKAGE(Allpix 2.2 REQUIRED)

ALLPIX_DETECTOR_MODULE(MODULE_NAME)
ALLPIX_MODULE_SOURCES(${MODULE_NAME} MySimulationModule.cpp)

```

All dependencies of the framework such as ROOT or Boost.Random are automatically added as CMake targets and can be used by the module. The required `CMAKE_CXX_STANDARD` is automatically inferred from the settings used to build the framework. Additional libraries can be linked to the module using the standard CMake command

```

TARGET_LINK_LIBRARIES(${MODULE_NAME} MyExternalLibrary)

```

A more complete CMake structure, suited to host multiple external modules, is provided in a separate repository [ap2-external-modules].

In order to load modules which have been compiled and installed in a different location than the ones shipped with the framework itself, the respective search path has to be configured properly in the Allpix Squared main configuration file:

```

[Allpix]
# Library search paths
library_directories = "-/allpix-modules/build", "/opt/ap2-modules"

```

On the right side of the page, there is a "Print entire section" link.

Figure 6: Beautiful code blocks

The screenshot shows a web browser displaying the Allpix Squared documentation. The page title is "Allpix Squared" and the URL is "localhost:1313/docs/03_getting_started/01_configuration_files/". The navigation menu includes "About", "Documentation", "Blog", and "Community". A search bar is present on the right. The left sidebar contains a "Documentation" section with a search bar and a list of topics including "Introduction", "Installation", "Getting Started", "Configuration Files", "Framework", "Physics Models & Material", "Properties", "Objects", "Modules", "Examples", "Module & Detector Development", "Development Tools & Continuous Integration", "Automated Testing", "FAQ", "Additional Tools & Resources", and "Appendix".

The main content area features a "Warning" section with a red vertical bar on the left. The warning text states: "If no units are specified, values will always be interpreted in the base units of the framework. In some cases this can lead to unexpected results. E.g. specifying a bias voltage as `bias_voltage = 50` results in an applied voltage of 50 MV. Therefore it is strongly recommended to always specify units in the configuration files."

Below the warning, there is a paragraph explaining that the internal base units are chosen for maximum precision and to avoid conversions. It also notes that combinations of base units can be specified using multiplication and division signs, and that units should be specified explicitly for all parameters.

Examples of specifying key/values pairs are given below:

```
# All whitespace at the front and back is removed
first_string = string_without_quotation
# All whitespace within the quotation marks is preserved
second_string = " string with quotation marks "
# Keys are split on whitespace and commas
string_array = "first element" "second element", "third element"
# Elements of matrices with more than one dimension are separated
# using square brackets
string_matrix_3x3 = [{"1", "0", "0"}, {"0", "cos", "-sin"}, {"0", "sin", "cos"}]
# If the matrix is of dimension 2xN, the outer brackets have to be
# added explicitly
integer_matrix_1x3 = [[18, 11, 12]]
# Integer and floats can be specified in standard formats
int_value = 42
float_value = 123.456e9
# Units can be passed to arithmetic types
energy_value = 1.22MeV
time_value = 42ns
# Units are combined in linear order without grouping or implicit brackets
acceleration_value = 1.0e/s/s
# Thus the quantity below is the same as 1.0deg*kV*K/s/s
random_quantity = 1.0deg*kV/s/s*K
# Relative paths are expanded to absolute paths, e.g. the following value
# will become "/home/user/test" if the configuration file is located
```

Figure 7: Fancy warnings

Live Demo



To try it yourself, you only need git and hugo [3]:

```
git clone --recurse-submodules https://gitlab.com/stephanlachnit/apsq-docs-hugo.git
cd apsq-docs-hugo
hugo server
```

Now you can explore the page by visiting <http://localhost:1313/>.

New format: Markdown



- ▶ Almost WYSIWYG
- ▶ Native to hugo
- ▶ Nicely rendered in GitLab
- ▶ Natively supports code highlighting
- ▶ Using GitLab Flavored Markdown (GFM) [4] for math

How to write Markdown?



```
# This is a large caption in Markdown
```

```
## This is also a caption but a bit smaller
```

Text in Markdown can be *italics* or **bold**.

Markdown supports [hyperlinks](<https://cern.ch/allpix-squared/>).

Markdown support code highlighting `inline` and via blocks:

```
```cpp
auto hist = CreateHistogram<TH1D>("name", "title", 100, 0., 100.);
```
```

How to write Markdown?



Figure 8: How the previous Markdown code renders on GitLab



How to write GLFM?

```
---  
# GLFM and hugo support front matter written in YAML  
title: "This is the title for the corresponding webpage"  
---
```

GLFM supports LaTeX inline math like $\frac{1}{2}$. GLFM also supports free standing equations:

```
```math  
\mu_e^{-1}(E) = 1 / \mu_{0,e} + E / v_{sat}
```
```


How to write GLFM?



The screenshot shows a web IDE interface for a file named 'talk2.md' (274 Bytes). The code editor contains the following GLFM code:

```
# GLFM and hugo support front matter written in YAML
title: "This is the title for the corresponding webpage"
```

Below the code, the rendered output is shown. It includes a text line: "GLFM supports LaTeX inline math like $\frac{1}{2}$. GLFM also supports free standing equations:" followed by a mathematical equation:

$$\mu_e^{-1}(E) = 1/\mu_{0,e} + E/v_{sat}$$

Figure 9: How the previous GLFM code renders on GitLab

L^AT_EX features missing in GLFM



- ▶ Automatic chapter and section numbering
- ▶ Proper captions for figures
- ▶ Citations using a bibliography
- ▶ `siunitx` (not supported by KaTeX)

What about the PDF?



- ▶ PDF will be created by converting GLFM to \LaTeX via pandoc [5]
- ▶ Sadly GLFM is not support by pandoc yet
- ▶ Some problems (most can be fixed with pandoc filters or regex):
 - ▶ GLFM's math format is not recognized
 - ▶ YAML front matter is ignored
 - ▶ Internal references are converted to `\href`
 - ▶ Pictures don't have max width / height set
 - ▶ Warning boxes don't work

State of the new implementaion



- ▶ GitLab MR !734
- ▶ Documentation fully converted
- ▶ Many tiny (already existing) mistakes fixed
- ▶ Website prototype working
- ▶ TODOs:
 - ▶ Fix CMake to recognize new files
 - ▶ Clean up hugo tree and create MR for website repo
 - ▶ Adjust CI to trigger correct pipelines for deployment
- ▶ Expect the new documentation to land in `master` in the next weeks!

Things to come



- ▶ Fix wrongly converted Markdown in \LaTeX source for the PDF
- ▶ Improved quick start, about page, screenshots page, etc.
- ▶ Documentation for each version (starting from the next one)
- ▶ Doxygen API Reference in hugo with Doxybook2 [6]

References

- [1] John Gruber. *Markdown*. 2004. URL: <https://daringfireball.net/projects/markdown/>.
- [2] *Docsy*. URL: <https://www.docsy.dev/> (visited on 2022-05-03).
- [3] *hugo*. URL: <https://gohugo.io/> (visited on 2022-05-03).
- [4] *GitLab Flavored Markdown (GLFM)*. URL: <https://docs.gitlab.com/ee/user/markdown.html> (visited on 2022-05-03).
- [5] John MacFarlane. *pandoc*. URL: <https://pandoc.org/> (visited on 2022-05-03).
- [6] *Doxybook2*. URL: <https://github.com/matusnovak/doxybook2> (visited on 2022-05-03).

Internal references in Markdown

```
# Some Chapter
```

```
## Some section
```

```
[This is a reference to this chapter.](#some-chapter)
```

```
[This is a to the section below.](#another-section)
```

```
## Another section
```

```
[Reference to chapter in other file.](./file.md#another-chapter)
```

The screenshot displays the Allpix Squared documentation page for the `allpix::SensorCharge` class. The page is structured as follows:

- Navigation:** Includes a search bar on the left and navigation links (About, Documentation, Blog, Community) at the top right.
- Breadcrumbs:** Shows the path: Documentation / Allpix*2 / Classes / allpix::SensorCharge.
- Section Headers:**
 - allpix::SensorCharge** (main title)
 - allpix::SensorCharge** (sub-title)
 - Module: Objects**
- Description:** "Base object for charge deposits and propagated charges in the sensor."
 - `#include "SensorCharge.hpp"`
 - Inherits from `allpix::Object`, `TObject`
 - Inherited by `allpix::DepositedCharge`, `allpix::PropagatedCharge`
- Public Functions:** A table listing various methods:

| Name | Description |
|--|---|
| <code>SensorCharge(ROOT::Math::XYZPoint local_position, ROOT::Math::XYZPoint global_position, CarrierType type, unsigned int charge, double local_time, double global_time)</code> | Construct a set of charges in a sensor. |
| <code>ROOT::Math::XYZPoint getLocalPosition() const</code> | Get local position of the set of charges in the sensor. |
| <code>ROOT::Math::XYZPoint getGlobalPosition() const</code> | Get the global position of the set of charges in the sensor. |
| <code>CarrierType getType() const</code> | Get the type of charge carrier. |
| <code>unsigned int getCharge() const</code> | Get total amount of charges stored. |
| <code>long getSign() const</code> | Get the sign of the charge for set of charge carriers. |
| <code>double getGlobalTime() const</code> | Get time after start of event in global reference frame. |
| <code>double getLocalTime() const</code> | Get local time in the sensor. |
| <code>virtual void print(std::ostream & ou) const override</code> | Print an ASCII representation of <code>SensorCharge</code> to the given stream. |
- Additional Info:** A link for `ClassDefOverride<SensorCharge>` is provided at the bottom.
- Right Sidebar:** Contains a "Print entire section" button and a list of "Public Functions" with links to their documentation.

Figure 10: API Reference in Docsy via Doxybook2

Lua filter for recognizing GLFM math with pandoc

-- From <https://gist.github.com/lierdakil/00d8143465a488e0b854a3b4bf355cf6#file-gitlab-math-lua>

```
function Math(el)
  if el.mathtype == "InlineMath" then
    if el.text:sub(1,1) == '`' and el.text:sub(#el.text) == '`' then
      local text = el.text:sub(2,#el.text-1)
      return pandoc.Math(el.mathtype, text)
    else
      local cont = pandoc.read(el.text)
      return { pandoc.Str("$") } .. cont.blocks[1].content .. { pandoc.Str("$") }
    end
  end
end

function CodeBlock(el)
  if el.classes[1] == "math" then
    return pandoc.Para({ pandoc.Math("DisplayMath", el.text) })
  end
end
```