Work in Group Adelmann: a how-to

Andreas Adelmann

December 17, 2025

1 Introduction

This document should help and – to some extent – guide the students in their Semester, Bachelor or Masters research.

2 Getting Started

Setup on the first day:

- Password setup: **this is the first step before login onto anything else!** You will be given a username and password on a sheet of paper. From a colleague's computer, `ssh` into any PSI computer, e.g. `ssh username@llc.psi.ch`. You will immediately be prompted to enter your new password. If this does not work, you can access this website using a computer connected to the Intranet (e.g. any PhD student's computer) to change your password: https://pw.psi.ch/PMUser.
- Wifi: connect to the Corp network with your PSI username and password.
- Email: outlook.office.com. Access with `firstname.surname@psi.ch` and your PSI password. We recommend that you redirect your PSI mail to your usual inbox (e.g. personal or ETH mail).
- Access to the Merlin cluster with `ssh username@login001.merlin7.psi.ch` or `login002.merlin7.psi.ch`. If you are immediately disconnected after login, it means that you have not been added to the correct unix groups. This happens often with new group members. Solve this by submitting a request on the following link and ask to be added to the following group `svc-cluster_merlin7`. You need to be in the PSI Intranet to access this link.

- Matrix chat: in our group we use the open-source chat `matrix.org`. There are several apps to access it, but most of us use the Element client `https://app.element.io`. Create an account and ask one of the group members to add you to the group chat.
- For issues and assistance with IT, you can open a ticket on ServiceNow (`http://psi.service-now.com/`); works only in the Corp network. For Merlin issues, you can email `merlin-admins@lists.psi.ch`.

2.1 Accessing Merlin from Home

- You will notice that `ssh` into Merlin only works from the Corp wifi network, but not from home. To access Merlin from home, you can either use the hopx system (see `https://www.psi.ch/en/computing/ssh-hop-ng`). Below some guidelines for this:
  1. Enable two-factor authentication "via a notification on your phone". See `https://www.psi.ch/en/computing/change-to-mfa`.
  2. `ssh username@hopx.psi.ch` This works even if you're not at PSI. You will need to approve the sign-in with your phone. You need to keep this terminal window open, and the session lasts 12 hours.
  3. Now, in another terminal session, you can access merlin by doing `ssh -J username@hopx.psi.ch username@merlin-l-001.psi.ch`.
- JupyterHub also only works from PSI. To use it from home:
  1. Dynamic tunnelling through port 9999 with `ssh -D 9999 username@hop.psi.ch`
  2. On your browser install a proxy switcher extension, e.g. *FoxyProxy*.
  3. On the proxy switcher add a new proxy with
     - Name: *myProxyForPsi*
     - Proxy Type: SOCKS5
     - Adress: localhost
     - Port: 9999
  4. Turn on the proxy *myProxyForPsi*
  5. Now JupyterHub will work, as long as your terminal with the dynamic tunnelling is open.
- You can also access the PSI network from outside via VPN. However, the VPN token must first be requested via ServiceNow from within the PSI network (`http://psi.service-now.com/`).

3 Report

Learning from examples is a proven way to success. The following reports are exemplary pieces of work:

- Semester work of Michael Ligotino
- Bachelor thesis of Ryan Ammann
- Master thesis of (now) Dr. Pirmin Berger. This work received the ETH medal

LaTeX templates and other tips are available here. We recommend the use Overleaf, checkout under *Templates* the ETH collection.

3.1 Literature Research

Visit the webpage of the ETH Library and in particular the article on AI-Tools.

3.2 Basic Writing Tips

Here's a short checklist of points to keep in mind when writing a report:

- acronyms need to be introduced once before used.
- no acronyms in titles
- Reference every figure and table in the text.
- Equation get a number iff they are referenced.
- Add captions to all figures and tables.
- Cite every image/plot/result that you have taken from someone else's work. The relevant guideline can be found here.
- If you make a statement that is not obvious, either prove it or cite a source. E.g. "...the Leapfrog integrator is used since it is symplectic [1]..."
- Use punctuation in your formulas and captions, e.g.

$$\frac{\partial N}{\partial t} = \lambda N(t)\,, \quad \text{with} \quad N(0) = N_0\,.$$

- no titles in figures, such information need to go to the figure caption.
- Figure 1 is an example of a high quality artwork.

The declaration of orginality must be added to the report as the last page.

3.3 Artwork

Presenting results are important. Do not use the default style of plotting tools. For example in Figure one you find a well crafted plot.

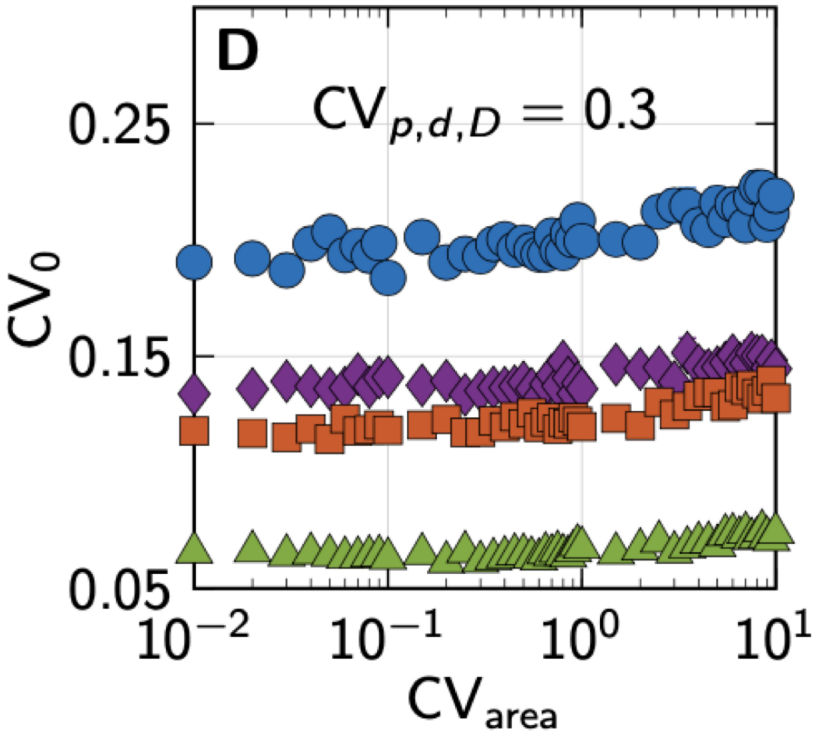

Figure 1: This is a nice figure made with TikTz, curtesy of J. Adelmann

4 Repository and Backup

Create a (ETH) Github project. It is mandatory that you put all source code, report and presentations into this repository. Do not push large files with simulation results. Push frequently (daily) as an additional benefit you have a backup for free! The url of the repository must be mentioned in your report.

5 Data Reproducibility

All results must be reproducible! As a consequence a clear recipe on how to create the data and how to analyse them must be given. The following steps are vital:

1. Seed the RNG if you are using random numbers
2. specify exactly the version of used libraries and compilers (all modules loaded)
3. explain exactly how to compile
4. explain how you performed the simulation i.e. provide a list of input files
5. for each plot/table in the report you have to provide an analysis script

5.1 Technical Advice

For each plot create a `csv` file, it is super easy:

```
# Reproducing the Idea of Figure 2.3.1 (Dahlquist & Björck)
#
# This Julia script computes absolute errors for three approximations to e:
# - (1 + 1/n)^n
# - exp(n*log(1 + 1/n))  (naïve)
# - exp(n*log1p(1/n))    (stable)
#
# It produces a log{log plot and writes three CSV files for each dataset.

using Printf
using DataFrames
using CSV
using Plots

ps = 1:14
ns = [10^p for p in ps]

err_power = Float64[]
err_exp_log = Float64[]
err_exp_log1p = Float64[]

for n in ns
    x = 1.0 / n
    val1 = (1.0 + x)^n
    val2 = exp(n * log(1.0 + x))
```

```
    val3 = exp(n * log1p(x))   # stable
    push!(err_power, abs(val1 - ))
    push!(err_exp_log, abs(val2 - ))
    push!(err_exp_log1p, abs(val3 - ))
end

df = DataFrame(p = collect(ps), n = ns,
               err_power = err_power,
               err_exp_log = err_exp_log,
               err_exp_log1p = err_exp_log1p)

println(first(df, 5))

# Write full data to CSV files
CSV.write("fig_2_3_1_full.csv", df)
CSV.write("fig_2_3_1_power.csv", select(df, :n, :err_power))
CSV.write("fig_2_3_1_exp_log.csv", select(df, :n, :err_exp_log))
CSV.write("fig_2_3_1_exp_log1p.csv", select(df, :n, :err_exp_log1p))

# Log{log plot
plot(ns, err_power, seriestype=:scatter, xscale=:log10, yscale=:log10,
     label="|(1+1/n)^n - e| (power)")
plot!(ns, err_exp_log, seriestype=:scatter,
      label="|exp(n*log(1+1/n)) - e| (naive log)")
plot!(ns, err_exp_log1p, seriestype=:scatter,
      label="|exp(n*log1p(1/n)) - e| (stable)")
xlabel!("n"); ylabel!("Absolute error")
title!("Reproducing Fig. 2.3.1 idea (log{log)")

savefig("fig_2_3_1_julia.png")

println("Saved CSV files and figure to current directory.")
```

6 Representing Results

Pay attention to the artwork and how you present your results. Do not use default plotting styles! There are many wonderful resources out there you can use and adapt. As an inspiration check this out. In general we suggest to use Beamer for the presentation. Templates for ETH/PSI are available.

Useful resources:

- Collection of example plots for various plot types, including code how to create them in Python: https://www.python-graph-gallery.com/
- Official PSI Logo: https://intranet.psi.ch/de/ako/logos-corporate
- Latex template for PSI themed presentations: http://amas.web.psi.ch/people/aadelmann/ETH-Accel-Lecture-1/projects/good-to-know.html

7 Short Status Updates

The weekly short status updates is a way to communicate progress, success but also to discuss challenges and ask for opinions. The key to success – i.e. obtain good advice – is a clear and concise presentation of the matter. This includes

1. setup the stage: introduction or repetition from last week
2. clearly state the problem
3. what did you already try out?

Bear in mind your audience is working on several other projects! In case there is progress or a topic for discussion we suggest to prepare a short presentation. The presentation then can be uploaded to the Wiki of the project. We recommend maintaining a *running presentation* to document your progress. This presentation should include new results, challenges, or specific issues you wish to discuss. When introducing a topic for discussion, ensure your explanations are clear and concise to help everyone follow along and contribute effectively.

8 Coding

Depending on the language you are using, use one of the following coding guidlines:

- C++: https://gitlab.psi.ch/OPAL/src/-/wikis/For-Developers/Codingstyle
- Python https://peps.python.org/pep-0008/
- Julia https://docs.julialang.org/en/v1/manual/style-guide/

For IPPL and OPAL we are using Doxygen, checkout the following cheat-sheat.

Consider the following binding rules:

- open frequently a MR, in minimum once a week !
- in case a function is larger than one A4 page you need to rewrite !

8.1 Guidelines for Students Working on IPPL

IPPL is hosted on GitHub. Below, you will find key resources to help you get started and stay organized:

- Instructions for building and running IPPL on PSI clusters: Merlin (CPU) and Gwendolen (GPU).
- An overview of the workflow for contributing to IPPL. If you are working on a bigger project, try to break down your project into chunks that can be merged into IPPL on a frequent basis such that we do not reach the end of the project with a big Pull Request to look through which might break the master. As soon as you have any working code with a unit test, please open a pull request.
- Guidelines for creating a pull request from a fork.
- Detailed coding guidelines to ensure consistency and quality.
- Additional information about the PSI computing clusters (specifications, etc.) and Slurm basics can be found on the High Performance Computing and Emerging Technologies @ PSI website.

8.1.1 IPPL Meetings

Students working on IPPL are required to participate in the weekly IPPL meetings. These are held every Tuesday at 16:00 on Zoom. You will receive a calendar invitation via your PSI or ETH email.

During the meetings, students are expected to provide short status updates. While updates in the form of a few slides are encouraged, they are not mandatory every week. Sharing slides can facilitate discussions and enable more focused feedback and support, check the Section 7 for more details.

All status updates need to be saved in this shared document.

9 C++

A very good introduction is Scott Meyers introduction (up tp C++14). A selection of relevant modern C++ features is collected at *www.cppstories.com*.

9.1 Minimal Documentation of Code

For each function a Doxygen documentation is required, follow and adapt the example below:

```
/**
 * @brief Computes the value of a mathematical function.
 *
 * This function calculates the result using the formula:
 * \f[
 *    f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x - \mu)^2}{2\sigma^2}}
 * \f]
 *
 * @param x Input value.
 * @param mu Mean value (default = 0.0).
 * @param sigma Standard deviation (default = 1.0).
 * @return Computed value based on the formula.
 */
double computeValue(double x, double mu = 0.0, double sigma = 1.0) {
    const double pi = 3.141592653589793;
    double coefficient = 1.0 / (sqrt(2.0 * pi * sigma * sigma));
    double exponent = exp(-((x - mu) * (x - mu)) / (2.0 * sigma * sigma));
    return coefficient * exponent;
}
```

10 Other important websites

1. https://intranet.psi.ch/de
2. https://lsm-hpce.gitpages.psi.ch/merlin6/faq.html#how-do-i-register-for-merlin
3. https://lsm-hpce.gitpages.psi.ch/merlin6/slurm-examples.html
4. https://kokkos.github.io/kokkos-core-wiki/index.html
5. https://github.com/kokkos/kokkos-tutorials/wiki/Kokkos-Lecture-Series
6. https://github.com/tmux/tmux/wiki(forperstistentterminalsessionsonMerlin)
7. Notebooks on Merlin https://merlin-jupyter.psi.ch:8000/hub/
8. https://www.psi.ch/en/computing/ssh-hop
9. KI-gestützte Tools für wissenschaftliches Arbeiten – jetzt als Selbstlernkurs