

Numerical Methods in Physics

Computing Environment Setup

Linux Mint/Ubuntu, Taurus and WFiS Labs

Hamzeh Khanpour

hamzeh.khanpour@fis.agh.edu.pl

<https://view.fis.agh.edu.pl/staff/hamzeh.khanpour/>

Faculty of Physics and Applied Computer Science
AGH University of Krakow

5 October 2026

Laboratory instructor and setup materials



Laboratory instructor

Hamzeh Khanpour

High Energy Particle Physicist, AGH University of Krakow || CERN-TH

Contact

E-mail: hamzeh.khanpour@fis.agh.edu.pl; hamzeh.khanpour@agh.edu.pl

Office & Room: D10 & 218

Course and setup resources

[WFiS staff/course page](#) | [CERN academic page](#)

Setup bundle target: [nmpLab01.zip](#).

<https://view.fis.agh.edu.pl/staff/hamzeh.khanpour/nmpLab01.zip>

<https://home.agh.edu.pl/~hamzeh.khanpour/nmpLab01.zip>

<https://hamzeh-khanpour.web.cern.ch/nmpLab01.zip>

How to use this presentation



Self-study rule

For each technical step, identify the machine first. Then follow the same sequence: **command** → **expected result** → **meaning** → **what to do if it fails**.

When a step works

Record the result, keep the exact command, and continue. A header file alone is not enough: the decisive test is whether a representative program compiles, links and runs correctly.

When a step fails

Do not guess. Record **machine**, **directory**, **exact command**, and the **first error message**. On managed AGH systems, do not attempt administrator actions yourself.

Three environments - keep them separate



PERSONAL LINUX MINT/Ubuntu

Your own authorized machine. You may install packages with `sudo apt` and use it for local development, plotting and independent checks.

TAURUS

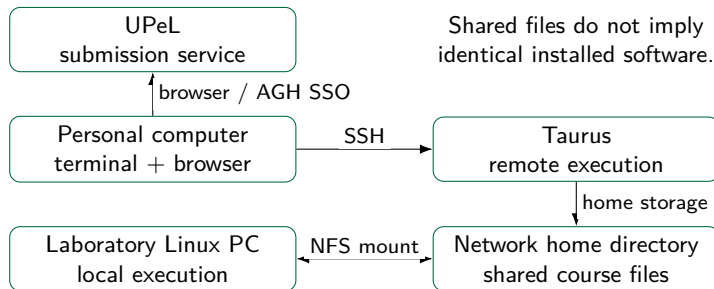
Managed WFiS Debian Linux server used for remote access and the network-home workflow. Use inspection, compilation and user-space files only.

WFiS LABORATORY PC

Managed classroom Linux workstation. Your Taurus home is mounted through NFS, but installed software is a separate environment and must be checked separately.

Key idea: the same files can be visible on two machines while the compilers and libraries installed on those machines are different.

Where are the files, and where does code run?



Storage and execution are different

SSH provides a remote terminal. **NFS** exposes the network home on classroom PCs. WFiS explicitly states that applications on Taurus and applications on classroom Linux PCs are independent.

Permissions and safety: what you may modify



Personal Linux Mint/Ubuntu only

Commands such as

```
sudo apt update  
sudo apt install ...
```

are appropriate only on a machine you are authorized to administer.

Taurus and laboratory PCs

Do **not** use `sudo`, `apt install`, or other system-modification commands. If required teaching software is absent, record the exact failure and report it to the instructor/WFiS administrator.

Security

Never share passwords, private keys or authentication tokens. Do not disable SSH host-key checking. A password typed into SSH normally produces no visible characters; this is expected.

Setup files: download instead of retyping



The student bundle contains

```
Presentation slides  
hello.cpp  
cblas_test.cpp  
lapacke_test.cpp  
fftw_test.cpp  
plot_test.py  
README.md  
optional      system_check.sh      and  
Makefile
```

nmp Lab01

Target URL:

<https://view.fis.agh.edu.pl/staff/hamzeh.khanpour/nmpLab01.zip>

<https://home.agh.edu.pl/~hamzeh.khanpour/nmpLab01.zip>

<https://hamzeh-khanpour.web.cern.ch/nmpLab01.zip>

LM1 | Identify your Linux Mint/Ubuntu system



PERSONAL LINUX MINT/Ubuntu | LOCAL MACHINE

Open a terminal with Ctrl+Alt+T. Record the operating system and architecture so you can later compare them with Taurus and the classroom workstation.

Commands

```
cat /etc/os-release  
lsb_release -a  
uname -m  
uname -a
```

Expected result

You should see distribution metadata, an architecture such as x86_64, and kernel/system information. Do not hard-code one Linux Mint/Ubuntu release into your notes; read the actual output from your machine.

LM2 | Refresh package metadata and repair only if APT asks



PERSONAL LINUX MINT/Ubuntu ONLY

First refresh the package database. A full system upgrade is general maintenance, not a mandatory prerequisite for this course.

```
sudo apt update
# Optional general maintenance:
sudo apt upgrade
```

If APT reports broken dependencies

Use repair commands only when APT explicitly reports a broken package state, and read the proposed changes before accepting them.

```
sudo dpkg --configure -a
sudo apt --fix-broken install
sudo apt update
```

Never run these repair/install commands on Taurus or a managed laboratory PC.

LM3 | Install the basic development environment



PERSONAL LINUX MINT/Ubuntu ONLY

Install the compiler/build tools, simple editor, archive tools and SSH utilities used in the course workflow.

```
sudo apt install \  
  build-essential gcc g++ make cmake pkg-config git nano \  
  zip unzip rsync openssh-client sshfs
```

Build tools

gcc: C-oriented compiler driver.
g++: C++ compile/link driver.
make: build automation.
cmake: build-system generator.
pkg-config: library metadata.

Workflow tools

git: version control.
nano: terminal editor.
zip/unzip: archive workflow.
openssh-client: SSH.
sshfs: optional remote mount.

PERSONAL LINUX MINT/Ubuntu | LOCAL MACHINE

Each command should print a version rather than `command not found`.

```
gcc --version
g++ --version
make --version
cmake --version
git --version
ssh -V
```

What this proves

A version string proves that the executable is available in your current PATH. It does not yet prove that numerical headers and libraries can be used; that requires compile/link/run tests.

PERSONAL LINUX MINT/Ubuntu ONLY

Install development packages for the interfaces used by the published projects.

```
sudo apt install \
    libblas-dev liblapack-dev liblapacke-dev libfftw3-dev
```

Optional optimized BLAS implementation for local testing:

```
sudo apt install libopenblas-dev
```

Course compatibility

Write ordinary BLAS/CBLAS and LAPACK source code. Do not make the project unnecessarily dependent on one local BLAS implementation. The primary course-style CBLAS link test uses `-lblas`; `-lopenblas` is only a local diagnostic fallback when needed.

LM6 | What the numerical libraries do



BLAS / CBLAS

BLAS: Basic Linear Algebra Subprograms – vector and matrix kernels.

CBLAS: C interface to BLAS. P01 uses `cblas_dgemm` for matrix multiplication.

OpenBLAS

One optimized BLAS implementation. Useful locally, but it is an implementation choice rather than a different numerical method.

LAPACK / LAPACKE

LAPACK: factorizations, linear solves, eigenproblems and related routines.

LAPACKE: high-level C interface to LAPACK. Later projects call LAPACKE routines.

FFTW3

Fast Fourier Transform in the West. P11 uses FFTW for discrete Fourier transforms in the noise-filtering project.

LM7 | Install Python for plotting and independent checks



PERSONAL LINUX MINT/Ubuntu ONLY

Python is recommended here for plotting, visualization and independent verification.

```
sudo apt install \  
    python3 python3-venv python3-numpy python3-scipy python3-matplotlib
```

Verify imports

```
python3 --version  
python3 -c "import numpy; print('NumPy:', numpy.__version__)"  
python3 -c "import scipy; print('SciPy:', scipy.__version__)"  
python3 -c "import matplotlib; print('Matplotlib:', matplotlib.__version__)"
```

Policy

Python validation does not silently replace a required C/C++ implementation of the numerical method. Follow the project handout and coordinator/instructor policy.

LM8 | Check headers, libraries and packages



Headers

```
find /usr/include /usr/local/include \  
  \( -name cblas.h -o -name lapacke.h -o -name fftw3.h \) \  
  2>/dev/null
```

Shared libraries and installed packages

```
ldconfig -p | grep -Ei 'blas|openblas|lapack|lapacke|fftw'  
dpkg -l | grep -Ei 'blas|openblas|lapack|lapacke|fftw'
```

Interpretation

Headers may live under paths such as `/usr/include/x86_64-linux-gnu/`. Finding a header proves only that the declaration is present. Successful compilation, linking and execution are the decisive tests.

LM9 | Create the Numerical Methods workspace



Create directories

```
mkdir -p ~/nmp
mkdir -p ~/nmp/setup
mkdir -p ~/nmp/Project{01..15}
mkdir -p ~/nmp/Project01/{src,data,plots,report}
```

Recommended student structure

```
~/nmp/
|-- setup/
|-- Project01/
|   |-- src/
|   |-- data/
|   |-- plots/
|   |-- report/
|-- Project02/
...
|-- Project15/
```

Instructor-only note

The supplied local manual also used a private reference/ directory for instructor results. Students do not need that directory unless explicitly asked to create it.

TEST1 | hello.cpp - complete source



nmp Lab01

Use the **wget** command to download nmpLab01.zip:

<https://view.fis.agh.edu.pl/staff/hamzeh.khanpour/nmpLab01.zip>

<https://home.agh.edu.pl/~hamzeh.khanpour/nmpLab01.zip>

<https://hamzeh-khanpour.web.cern.ch/nmpLab01.zip>

Purpose

This is the smallest compile/run smoke test. It tests the C++ toolchain only; it is not a numerical project.

```
#include <iostream>
```

```
int main()
```

```
{
```

```
    std::cout << "Numerical Methods in Physics\n";
```

```
    std::cout << "C++ setup OK\n";
```

```
    return 0;
```

```
}
```

TEST1 | Compile, run and interpret the result



Compile

```
cd ~/nmp/setup  
g++ -std=c++17 -Wall -Wextra -O2 hello.cpp -o hello
```

Run

```
./hello
```

Expected output

```
Numerical Methods in Physics  
C++ setup OK
```

What the command means

`-std=c++17` selects the demonstration language standard; `-Wall -Wextra` enable useful warnings; `-O2` enables optimization; `-o hello` names the executable; `./hello` executes the program in the current directory.

TEST2 | CBLAS matrix multiplication – know the answer first



Mathematical target

The smoke test multiplies

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}.$$

Therefore

$$AB = \begin{pmatrix} 19 & 22 \\ 43 & 50 \end{pmatrix}.$$

Why this is useful

P01 compares your own matrix operations with BLAS/CBLAS. The smoke test isolates the environment: small matrices, exact expected output, and one representative BLAS3 call.

Rule

Correct answer first, timing second.
Performance comparisons are meaningful only after the numerical results agree.

TEST2 | cblas_test.cpp – complete source (1/2)



```
#include <iostream>
#include <cblas.h>

int main()
{
    const int M = 2;
    const int N = 2;
    const int K = 2;

    double A[4] = {
        1.0, 2.0,
        3.0, 4.0
    };

    double B[4] = {
        5.0, 6.0,
        7.0, 8.0
    };

    double C[4] = {
        0.0, 0.0,
        0.0, 0.0
    };
};
```

TEST2 | cblas_test.cpp – complete source (2/2)



```
cblas_dgemm(  
    CblasRowMajor,  
    CblasNoTrans,  
    CblasNoTrans,  
    M,  
    N,  
    K,  
    1.0,  
    A,  
    K,  
    B,  
    N,  
    0.0,  
    C,  
    N  
);  
  
std::cout << C[0] << " " << C[1] << '\n';  
std::cout << C[2] << " " << C[3] << '\n';  
  
return 0;  
}
```

TEST2 | Compile, run and read cblas_dgemm



Primary course-compatible test

```
g++ -std=c++17 -Wall -Wextra -O2 \  
    cblas_test.cpp -lblas -lm -o cblas_test  
./cblas_test
```

Expected

```
19 22  
43 50
```

Main parameters

CblasRowMajor: row-wise layout.
CblasNoTrans: do not transpose.
M,N,K: result rows, result columns, shared inner dimension.
alpha=1, beta=0: compute $C = AB$.

Leading dimensions and fallback

lda, ldb, ldc describe stored row/column strides. If generic -lblas fails on a personal Mint machine with OpenBLAS installed, -lopenblas may be used as a local diagnostic alternative. Do not automatically prescribe it on Taurus.

TEST3 | LAPACK, LAPACKe and the dependency chain



Library roles

LAPACKe: high-level C interface.

LAPACK: factorization/solve routines.

BLAS: underlying vector/matrix kernels.

Published projects P02, P03, P06, P09 and P10 use LAPACKe routines.

Link order

GNU tools process link inputs in command-line order. For the ordinary link used here, place dependencies after what uses them:

```
program -> LAPACKe -> LAPACK ->
                        BLAS
```

2026/27 decision rule

The current course webpage still contains an older local `lapacke.zip` workaround. Do not teach that fallback automatically. Test the current system first; use the workaround only if its current need, layout and binary compatibility are verified.

TEST3 | lapacke_test.cpp – complete source (1/2)



```
#include <iostream>
#include <lapacke.h>

int main()
{
    // Solve:
    // 3x + y = 9
    // x + 2y = 8
    // Exact solution: x = 2, y = 3
    double A[4] = {
        3.0, 1.0,
        1.0, 2.0
    };

    double b[2] = {
        9.0,
        8.0
    };

    lapack_int ipiv[2];

    lapack_int info = LAPACKE_dgesv(
```

TEST3 | lapacke_test.cpp – complete source (2/2)



```
LAPACK_ROW_MAJOR,  
2,  
1,  
A,  
2,  
ipiv,  
b,  
1  
);  
  
if (info != 0) {  
    std::cerr  
        << "LAPACKE_dgesv failed. info = "  
        << info << '\n';  
    return 1;  
}  
  
std::cout << "x = " << b[0] << '\n';  
std::cout << "y = " << b[1] << '\n';  
  
return 0;  
}
```

TEST3 | Compile, run and decide the LAPACKE route



Compile and run when system LAPACKE exists

```
g++ -std=c++17 -Wall -Wextra -O2 \  
    lapacke_test.cpp -llapacke -llapack -lblas -lm \  
    -o lapacke_test  
./lapacke_test
```

Expected

```
x = 2  
y = 3
```

Decision tree

Check `lapacke.h` → check `liblapacke` → compile → run. If all succeed, use the system library. If the header/library is absent on an AGH machine, stop and report it. A nonzero LAPACKE info is a routine status and must be interpreted; it is not automatically an installation failure.

TEST4 | FFTW3 - a unit impulse gives a simple exact check



Input

Use the four-point complex sequence

$$x = [1, 0, 0, 0].$$

For the forward discrete Fourier transform, every Fourier coefficient is $1 + 0i$.

What the program tests

Allocation with `fftw_malloc`; a one-dimensional complex DFT plan; `FFTW_FORWARD`; `FFTW_ESTIMATE`; execution; plan destruction; and memory cleanup.

Later project connection

P11 uses FFTW for noise filtering. Before interpreting a spectrum, verify a tiny transform whose result you already know.

TEST4 | fftw_test.cpp - complete source (1/3)



```
#include <iostream>
#include <fftw3.h>

int main()
{
    const int N = 4;

    fftw_complex* in =
        static_cast<fftw_complex*>(
            fftw_malloc(sizeof(fftw_complex) * N)
        );

    fftw_complex* out =
        static_cast<fftw_complex*>(
            fftw_malloc(sizeof(fftw_complex) * N)
        );

    if (!in || !out) {
        std::cerr << "FFTW allocation failed\n";
        if (in) {
            fftw_free(in);
        }
        if (out) {
```

TEST4 | fftw_test.cpp - complete source (2/3)



```
        fftw_free(out);
    }
    return 1;
}

for (int i = 0; i < N; ++i) {
    in[i][0] = 0.0;
    in[i][1] = 0.0;
}

// Four-point unit impulse: x = [1, 0, 0, 0]
in[0][0] = 1.0;

fftw_plan plan =
    fftw_plan_dft_1d(
        N,
        in,
        out,
        FFTW_FORWARD,
        FFTW_ESTIMATE
    );

if (!plan) {
```

TEST4 | fftw_test.cpp - complete source (3/3)



```
        std::cerr << "FFTW plan creation failed\n";
        fftw_free(in);
        fftw_free(out);
        return 1;
    }

    fftw_execute(plan);

    std::cout << "FFT of unit impulse:\n";
    for (int i = 0; i < N; ++i) {
        std::cout
            << out[i][0]
            << " "
            << out[i][1]
            << '\n';
    }

    fftw_destroy_plan(plan);
    fftw_free(in);
    fftw_free(out);

    return 0;
}
```

TEST4 | Compile, run and interpret the FFTW result



Compile and run

```
g++ -std=c++17 -Wall -Wextra -O2 \  
    fftw_test.cpp -lfftw3 -lm -o fftw_test  
./fftw_test
```

Expected approximately

```
FFT of unit impulse:  
1 0  
1 0  
1 0  
1 0
```

Important convention

FFTW computes an unnormalized transform. In a later forward-plus-inverse round trip, divide the inverse result by the transform length N before comparing with the original input.

TEST5 | plot_test.py - complete source and plotting check



```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0.0, 2.0 * np.pi, 200)
y = np.sin(x)

plt.plot(x, y)
plt.xlabel("x")
plt.ylabel("sin(x)")
plt.title("Numerical Methods in Physics - Plot
          Test")
plt.grid(True)

plt.savefig("plot_test.pdf")
plt.show()
```

Personal Linux Mint/Ubuntu

```
python3 plot_test.py
ls -lh plot_test.pdf
```

Expected: a sine-wave plot and a new plot_test.pdf.

Headless remote session

If no graphical display is available, save the figure without an interactive window or run, for example:

```
MPLBACKEND=Agg \
python3 plot_test.py
```

This is a plotting/verification route, not a replacement for required C/C++ numerical implementation.

LM10 | Practice ZIP, confirm SSH, prepare optional SSHFS



ZIP workflow

```
cd ~/nmpLab01
zip -r setup_test.zip setup/
unzip -l setup_test.zip
unzip -t setup_test.zip
```

Create, list, then integrity-test the archive. This is the same discipline later used for activity ZIP submissions.

SSH client

```
ssh -V
```

Expected: an OpenSSH version string.

Optional SSHFS preparation

```
mkdir -p ~/AGH_Taurus
sshfs \
  YOUR_LOGIN@taurus.fis.agh.edu.pl:/ACTUAL_REMOTE_HOME \
  ~/AGH_Taurus
```

Do not guess the remote home path. SSHFS is optional convenience; the core course workflow is SSH plus the WFiS NFS-mounted home on classroom Linux PCs.

Linux Mint/Ubuntu readiness checkpoint



Core setup

- ☐ Linux Mint/Ubuntu/architecture recorded
- ☐ gcc and g++ available
- ☐ hello.cpp compiles and runs
- ☐ BLAS/CBLAS headers and libraries checked
- ☐ cblas_test.cpp gives the exact matrix
- ☐ /nmp workspace created

Later-project preparation

- ☐ LAPACK/LAPACKE status known
- ☐ FFTW3 status known
- ☐ Python/NumPy/SciPy/Matplotlib status known
- ☐ ZIP/unzip works
- ☐ SSH client works
- ☐ Optional system_check.sh can be run read-only

Status vocabulary

Mark each item **working**, **pending**, or **needs assistance**. A pending check needs an action, not a guessed pass.

TAURUS | MANAGED WFiS SERVER

Current official hostname: `taurus.fis.agh.edu.pl`. WFiS creates staff/doctoral accounts through the administrator and student-group accounts before classes. A working account is required before the remaining Taurus steps can be completed.

If login/password is unknown or fails

Do not keep guessing. Confirm the account with WFiS server administration.

Current support route

E-mail: admin@fis.agh.edu.pl

Kompas/Ticket: [WFiS network issues](#)

Never send a password, private key or token in a support message.

TA2 | SSH client and first connection



Linux / macOS

```
ssh -V  
ssh YOUR_LOGIN@taurus.fis.agh.edu.pl
```

Windows PowerShell / Windows Terminal

```
ssh -V  
ssh YOUR_LOGIN@taurus.fis.agh.edu.pl
```

What to expect

Replace YOUR_LOGIN with the account confirmed by WFiS. On first connection, SSH may ask whether you trust a host key. Password entry normally shows no characters, dots or asterisks.

Do not bypass checks

Do not use shortcuts that disable host-key verification. If the account is not yet active, stop here and resolve the account problem first.

TA3 | Verify the SSH host key before accepting it



Current public WFiS notice checked 2 Oct 2026

New Taurus access was opened on 19 May 2026. WFiS currently publishes:

Algorithm: ecdsa-sha2-nistp256 256

SHA-256: fp4YXXliP+9hWBNhwZVNCpMn9d8iXn7UgtGtUFtdiKo

If your client shows a different key

One SSH server can advertise several host-key algorithms, so a client may show a different algorithm. Do not accept an unexpected fingerprint blindly and do not disable checking. Confirm it with WFiS/instructor.

If an old Taurus key is cached

Only after confirming the current key, remove the obsolete host entry and reconnect:

```
ssh-keygen -R taurus.fis.agh.edu.pl
```

TA4 | First successful login and home-directory activation



Why the first direct Taurus login matters

WFiS explicitly requires at least one direct SSH login to Taurus before the user's home directory becomes available on the Linux laboratory workstations.

After successful authentication

```
whoami
hostname
pwd
echo $HOME
ls -la ~
```

Record the result

These commands establish your actual username, the machine name, the current path and the home-directory path. Do not guess the remote home path for later SSHFS use.

TA5 | Identify Taurus and create the course workspace



Read the actual Taurus environment

```
cat /etc/os-release  
uname -m  
uname -a
```

Public WFiLS documentation identifies Taurus as a 64-bit Debian Linux system. Read the exact Debian release from `/etc/os-release`; do not guess it.

Create user-space course directories

```
mkdir -p ~/nmp/setup  
mkdir -p ~/nmp/Project{01..15}  
mkdir -p ~/nmp/Project01/{src,data,plots,report}  
ls ~/nmp
```

Managed-system rule

These are ordinary user-space operations. Do not install or modify system packages on Taurus.

TA6 | Check the compiler and numerical libraries on Taurus



Compiler

```
gcc --version
g++ --version
which gcc
which g++
```

Libraries and headers

```
ldconfig -p | grep -Ei 'blas|openblas|lapack|lapacke|fftw'
find /usr/include /usr/local/include \
  \( -name cblas.h -o -name lapacke.h -o -name fftw3.h \) \
  2>/dev/null
pkg-config --list-all 2>/dev/null | grep -Ei 'blas|lapack|fftw'
```

Interpretation

A library found on your personal Linux machine is not automatically present on Taurus. A library found on Taurus is not automatically present on the classroom PC. Record the status; do not use `sudo apt install` on Taurus.

TA7 | Run the basic C++ and CBLAS smoke tests on Taurus



Plain C++

```
cd ~/nmp/setup
g++ -std=c++17 -Wall -Wextra -O2 hello.cpp -o hello
./hello
```

Expected: Numerical Methods in Physics and C++ setup OK.

CBLAS – important for P01

```
g++ -std=c++17 -Wall -Wextra -O2 \
  cblas_test.cpp -lcblas -lm -o cblas_test
./cblas_test
```

Expected: 19 22 and 43 50.

If either test fails

Record the exact first error and machine. Do not install packages yourself. The test source is a setup smoke test only; it is not the solution to P01.

TA8 | LAPACKe, FFTW, Python status and the second login



Test what is actually installed

Try the standard commands from the earlier slides for `lapacke_test.cpp` and `fftw_test.cpp`. Also record optional Python status:

```
python3 --version
python3 -c "import numpy; print('NumPy:', numpy.__version__)"
python3 -c "import scipy; print('SciPy:', scipy.__version__)"
python3 -c "import matplotlib; print('Matplotlib:', matplotlib.__version__)"
```

LAPACKe special rule

If `lapacke.h` or `-llapacke` is missing, stop and report it. The instructor must decide whether current system LAPACKe, the old course fall-back, or an administrator installation is the correct route.

Second login test

```
exit
ssh YOUR_LOGIN@taurus.fis.agh.edu.pl
ls ~/nmp
ls ~/nmp/setup
```

Persistence confirms that the course files remain in the network home.

LAB1 | Log in to the WFiS Linux workstation and verify NFS



WFiS LABORATORY PC | MANAGED CLASSROOM MACHINE

After the first direct Taurus login succeeds, select/boot Linux on the classroom computer and use the same Taurus username and password at the graphical login screen.

Open Terminal and check

```
whoami  
hostname  
echo $HOME  
pwd  
ls ~/nmp
```

Pass condition

Files created on Taurus should be visible through the NFS-mounted home directory. If the account cannot log in or the home is absent, stop and report the problem rather than attempting local account/system changes.

LAB2 | Shared files do not mean shared software



Re-check the classroom PC itself

```
cat /etc/os-release
uname -m
g++ --version
ldconfig -p | grep -Ei 'blas|openblas|lapack|lapacke|fftw'
find /usr/include /usr/local/include \
  \( -name cblas.h -o -name lapacke.h -o -name fftw3.h \) \
  2>/dev/null
```

Critical distinction

WFiLS states that classroom-PC applications are independent of the software installed on Taurus and should be used first where appropriate. A program compiling on one machine does not prove that it compiles on the other.

If teaching software is missing

Record the machine and exact error. Licensed teaching software is installed by the network administrator after testing; students should not modify the managed workstation.

Troubleshooting: identify the failure before changing anything

Account / SSH

Permission denied: wrong login/password, inactive account, or password issue. Stop repeated guesses and confirm the account.

REMOTE HOST IDENTIFICATION HAS CHANGED: verify the current official host key first; only then remove an obsolete `known_hosts` entry if appropriate.

command not found: executable is absent or not in `PATH`; identify the machine before acting.

Compiler / linker / numerical routine

cblas.h / lapacke.h / fftw3.h not found: header is not visible in the compiler include path.

cannot find -lblas / -llapacke / -lfftw3: linker cannot find that library.

undefined reference: link flag/order or missing dependency.

nonzero LAPACK info: interpret the routine status; it is not automatically a package problem.

Always report four things

Machine | directory | exact command | first error message.

UPeL and official resources



UPeL

AGH UPeL uses AGH Single Sign-On. Do not assume Taurus credentials are interchangeable with AGH SSO credentials. The instructor must confirm the current NMP course/group and assignment names.

Submission reminder

The public NMP page describes two distinct tracks: source code/numerical data as a ZIP activity submission, and the scientific report as a PDF report submission.

Clickable official resources

[Numerical Methods in Physics course page](#)

[WFiS Taurus instructions](#)

[WFiS network / current host-key notice](#)

[WFiS D-10/D-7 laboratory rules](#)

[UPeL portal](#)

Before Project 01: final readiness checkpoint



Access and environment

- ☐ I know which machine I am using.
- ☐ My Taurus account works.
- ☐ I completed the first direct Taurus SSH login.
- ☐ My home directory is active.
- ☐ I can log in to the classroom Linux workstation.
- ☐ My Taurus files are visible there through NFS.
- ☐ I know the compiler/library status on each machine separately.

Project 01 preparation

- ☐ g++ works.
- ☐ hello.cpp compiles and runs.
- ☐ CBLAS status is known.
- ☐ cblas_test.cpp returns the exact matrix.
- ☐ LAPACK and FFTW status is known.
- ☐ I know where /nmp is.
- ☐ I know whom to contact if something fails.

Next: P01 | Vector and Matrix Operations

The setup CBLAS program verifies the environment only. P01 is the actual assessed project: students write their own program, perform the assigned matrix/vector tasks, save numerical outputs, and compare their own matrix multiplication with `cblas_dgemm`.

