(PRACTICAL)
COMPUTATIONAL PHYSICS

Physics 551
Lecture 8

NOTATION

Extra Reading

Optional Exercise

Recommended

- This lecture slides for this course will attempt to use a uniform notation throughout. A normal paragraph looks like this.

☞ *Italicized paragraphs with pen bullets will indicate definitions, with the defined word or phrase shown in* ***SMALL-CAPS****.*

✏ Pencil bullets will indicate the introduction of **new notation**.

☞ Pointing hand bullets indicate important points that might otherwise be overlooked.

ANNOUNCEMENTS

ANNOUNCEMENTS

- To clone this week's **C++ demonstration materials** please invoke

```
$git clone https://github.com/hughdickinson/CompPhysL8CPP.git
/home/computationalphysics/Documents/cPlusPlus/lecture8
```

- To clone this week's **Python demonstration materials** please invoke

```
$git clone https://github.com/hughdickinson/CompPhysL8Python.git
/home/computationalphysics/Documents/python/lecture8
```

☞ You can also find these commands on the Blackboard Learn website.

ANNOUNCEMENTS

- **QUESTION 1:** Is everyone in the class a graduate student?
- If so, I am **not required** to set a **special** mid-term assessment exercise.
- **QUESTION 2:** Does anybody particularly **want** me to set a special mid-term assessment exercise?
- **Be aware** that this will result in a larger contribution from your weekly homework to your final grade for the class.
- I can provide an **anonymous survey** on **Blackboard Learn** to obtain **considered responses** to this question, if that is preferred.

CLARIFICATIONS

CLARIFICATIONS

- **Recall** that **multiple** source files can be used to build standalone binary executables e.g.

```
$ clang++ -std=c++11 -o pathToExecutable sourceFiles…
```

☞ It is **not** true that **all C++ source files** that **do not** contain a `main()` function **must** be used to create shared libraries.

☞ It **is** true that C++ source files that **are** used to create shared libraries **must not** contain a `main()` function.

☞ **Nonetheless**, it is often a **good idea** to create shared libraries if the source code you write is likely to be **reusable**.

C++

THE GNU SCIENTIFIC LIBRARY (GSL)

WHAT IS THE GSL?

☞ *The **GNU SCIENTIFIC LIBRARY** (GSL) is an **Open Source** software package that provides a rich suite of functionality for **scientific computing**.*

- Facilities provided by the GSL include: ***solution of differential equations***; ***vector and matrix manipulation***; ***function interpolation***; ***Monte-Carlo integration***; ***simulated annealing; numerical root-finding***; ***wavelet, Hankell and Fourier transforms*** and **many more**.

☞ The GSL provides **comprehensive online documentation** at:

www.gnu.org/software/gsl/manual/html_node

HOW TO USE THE GSL

- The GSL is distributed using **header files** and **shared libraries**.

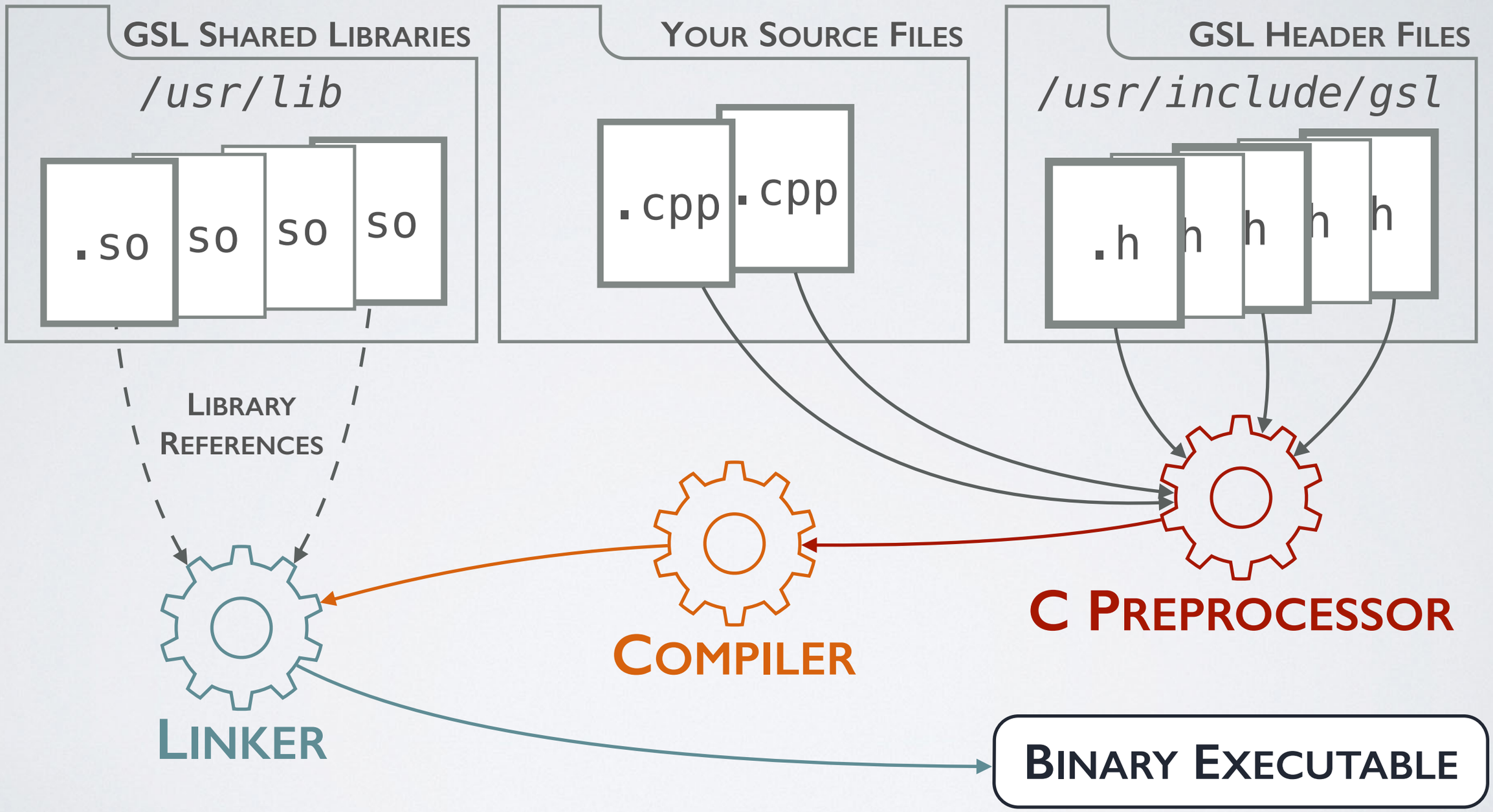
☞ **By default**, the GSL **header files** and the GSL **shared libraries** are installed in */usr/include/gsl* and */usr/lib* respectively.

☞ The **primary** GSL shared library is */usr/lib/libgsl.so*

☞ To **build** programs that use the **basic** GSL functionality, **include GSL header files** in your source code and invoke

```
$ clang++ -std=c++11 -o output -I/usr/include -L/usr/lib
  -lgsl other_flags?… other_inputs…
```

HOW TO USE THE GSL



GSL TIPS AND TRICKS

- **Recall** that invoking **standalone binaries** that are linked against shared libraries normally requires an update of the `LD_LIBRARY_PATH`.
- In order to use the **GSL shared libraries**, one **might expect** that the following shell command is required

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib
```

- In fact, if the GSL libraries are installed in their **default locations**, this step is **not** necessary!

☞ The `/usr/lib` directory is one of **very few** locations that the **dynamic linker** searches **by default** when a binary is invoked.

GSL TIPS AND TRICKS

- The GSL provides a handy utility called `gsl-config` to help you determine the appropriate flags to use when invoking `clang++`.
- A **list of flags** that `gsl-config` accepts can be obtained by invoking it with **no arguments**.
- To print all the flags you ***may*** require, invoke

```
$ gsl-config --cflags –libs
```

- If your code **does not use BLAS functionality** (see demonstration material), invoke

```
$ gsl-config --cflags --libs-without-cblas
```

DEMONSTRATION

Using the **GNU Scientific Library**

Clone the Lecture 8 C++ demonstration material from Github:

```
$git clone https://github.com/hughdickinson/CompPhysL8CPP.git
 /home/computationalphysics/Documents/cPlusPlus/lecture8
```

PYTHON

IPYTHON NOTEBOOK

☞ **Recall** that the ***IPython Notebook*** interface can be launched from the shell command line using

```
$ ipython notebook
```

☞ **Experimenting** with the myriad features that Python provides is a great way to **learn the language** and **find facilities** to aid your research.

- You may want to experiment with Python or the IPython Notebook **without modifying the demonstration material**.

☞ There is a "**New Notebook**" button in the **top, right-hand corner** of the IPython Notebook ***Home*** page that you can use to **create a new notebook** to work in.

DEMONSTRATION

More Python Basics

***If necessary*, clone the *Lecture 7* Python demonstration material from Github:**

```
$ git clone https://github.com/hughdickinson/CompPhysL7Python.git
  /home/computationalphysics/Documents/python/lecture7
```

NUMPY
(CRUNCHING NUMBERS)

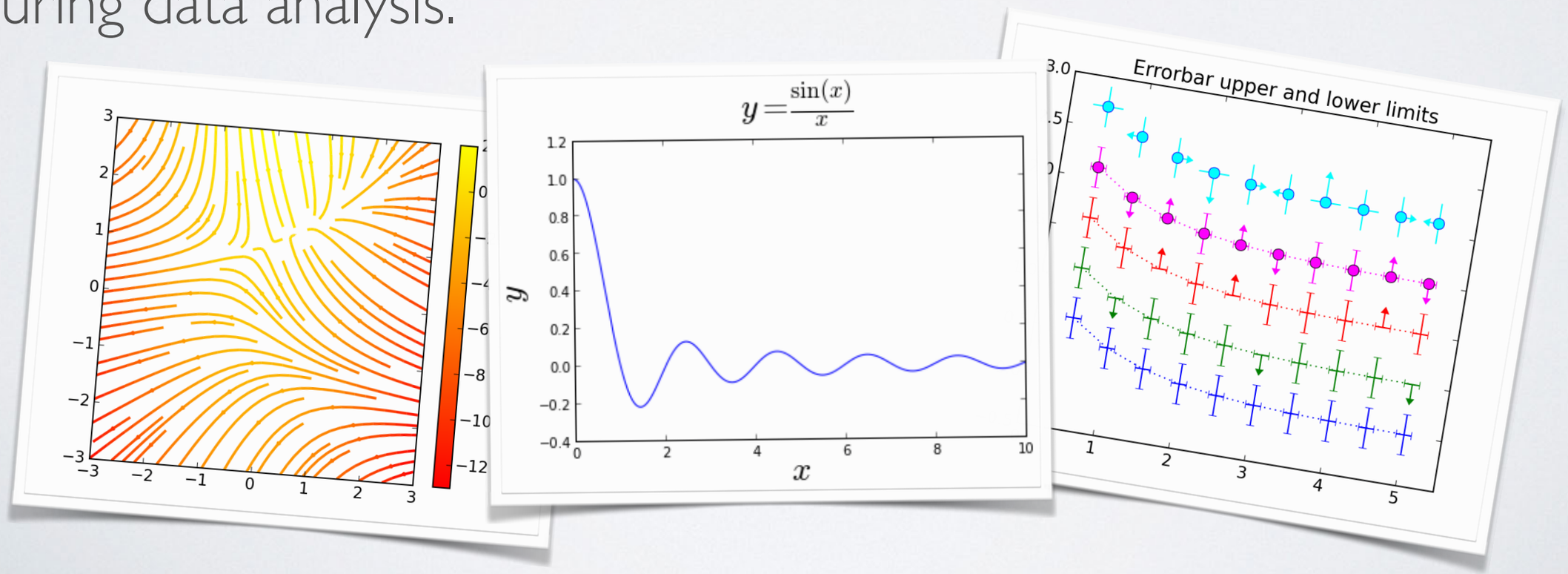
☞ *The **NumPy** package provides invaluable **data-handling** support for numerous **scientifically targeted** Python utilities.*

- Its primary functionality is the provision of a **homogeneous** (i.e. elements all have the **same type**), **multidimensional** array type.
- This array type supports **indexing**, **slicing** and **reshaping** as well **highly efficient** array-wise mathematical operations.

☞ **Mathematical operations** can be performed on whole multidimensional arrays **without the need for explicit loops**.

MATPLOTLIB (PLOTTING RESULTS)

- ***MATPLOTLIB*** *is a feature-rich 2-dimensional* ***plotting package*** *for Python.*
- Matplotlib can be used to generate **publication quality figures** in a **variety of formats**.
- It can also be used to plot and **review results** at intermediate stages during data analysis.



DEMONSTRATION

Using **NumPy** and **Matplotlib**

Clone the Python demonstration material from Github:

```
$git clone https://github.com/hughdickinson/CompPhysL8Python.git
 /home/computationalphysics/Documents/python/lecture8
```

LECTURE 8 SUMMARY

- After reviewing the material from this lecture (including the demonstration material) **and completing the reading exercises** you should know:
  1. How to **invoke** and **use** the **basic Python interpreter**, the terminal-based **IPython interpreter** and the web-browser-based **IPython Notebook interface**.
  2. The basic properties of Python's built-in **`set`** type.
  3. The basic properties of Python's built-in **`map`** type.

LECTURE 8 SUMMARY

4. How to specify **conditional branching statements** in Python using **`if`**, **`elif`** and **`else`** clauses as well as the **ternary branching construct**.
5. How **control the flow** of a Python program using **`for`**-loops and **`while`**-loops.
6. How to **define** and **call functions** within in your Python programs.

LECTURE 8 SUMMARY

7. That **exceptions** are used to react to problems that arise from weak variable typing.
8. How to **handle exceptions** that are raised by Python code.
9. How to **efficiently initialize** Python lists using the **list comprehension** syntax.
10. How to **extend the functionality** of Python by **import**ing modules.

LECTURE 8 SUMMARY

11. Some simple ways in which to **instantiate** multidimensional NumPy arrays.
12. How to **index** and **slice** NumPy arrays and the built-in Python sequence types.
13. How to apply **array-wise mathematical operations** to instances of the NumPy array type.
14. The **basics** of generating figures using **Matplotlib**.

OPTIONAL READING

The Python Online Tutorial
docs.python.org/2/tutorial/

The Python Online Reference
docs.python.org/2/library/

The GSL Library Reference
www.gnu.org/software/gsl/manual/

The NumPy Online Tutorial
wiki.scipy.org/Tentative_NumPy_Tutorial

LECTURE 7 HOMEWORK

Review the C++ demonstration material from Lecture **8**!

***If neccessary*, re-read sections:**

5.7. Set Types (set)

5.8. Mapping Types

From the **Python Standard Library Reference**

https://docs.python.org/2.7/library/index.html

Then read

Section 8: Errors and Exceptions

From the **Online Python tutorial**

https://docs.python.org/2/tutorial

- Complete the **Lecture 8 Homework Quiz** that you will find on the course Blackboard Learn website.