The computer labs provide computational experience related to the analytic theory presented in the lectures. The main tool for these exercises is a programming language called Julia designed for the implementation of numerical algorithms that combines the compiled efficiency of C and Fortran with the interactive and notational convenience of MATLAB and Python.

In science and engineering an important goal is to become a skilled practitioner by doing it yourself. To this end the computers in the lab have been provisioned with a Linux programming environment similar to what is deployed on the university high-performance cluster, all other supercomputers worldwide and for most cloud computing. To access Linux please restart the computer using the USB network boot key for this class.

Rather than using the lab equipment it is also possible to freely install Julia on your personal laptop. While using your own computer goes along well with doing it yourself, I will unfortunately be unable to help with any technical problems that might crop up in that case. Even so, I'd recommend trying to install Julia at home, if only to avoid coming in after hours to complete the homework. You may also use your laptop in the lab.

Note that it is possible to forgo Julia and perform all your computations using a different programming language. Although I would be happy to grade assignments completed using such alternatives, my opinion is Julia makes numerical methods much easier than a general-purpose programming language. I am also able to provide more help with Julia.

Quadratic Equations

Our lab exercises start with solving a quadratic equation. Suppose we seek to solve $x^2 + x = 5$. The quadratic formula tells us

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad \text{where} \quad a = 1, \quad b = 1 \quad \text{and} \quad c = -5.$$

While this formula is well known, it must be rearranged for numerical calculations to avoid loss of precision. This lab introduces the Julia programming environment and the specific details of turning your results in for grading. We also focus on numerical considerations when using the quadratic formula that apply to any computing device including a hand calculator.

In my opinion even if you know another language well, Julia will make numerical methods easier. Begin by using the interactive read-evaluate-print loop in Julia as a desktop calculator to plug in the numbers of the quadratic formula.

To start Julia open a terminal window and type the command `julia` followed by the enter key. If you are using your personal computer the technique for starting Julia may involving clicking a menu item. In either case the display on the screen should look like

```
$ julia
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.6.1 (2021-04-23)
 _/ |\__'_|_|_|\__'_|  |
|__/                   |

julia>
```

The prompt `julia>` indicates Julia is in interactive mode waiting to be used as a calculator. Enter the quadratic formula as two functions

```
xplus(a,b,c)=(-b+sqrt(b^2-4*a*c))/(2*a)
xminus(a,b,c)=(-b-sqrt(b^2-4*a*c))/(2*a)
```

Note that after entering the first function, the second one can be obtained by pressing the up-arrow key, making the needed changes and then pressing enter. Using the arrow keys to recall, edit and execute previous commands in the Julia REPL can save lots of typing. You can also cut and paste the necessary input from this document.

The new lines on the screen should look like

```
julia> xplus(a,b,c)=(-b+sqrt(b^2-4*a*c))/(2*a)
xplus (generic function with 1 method)

julia> xminus(a,b,c)=(-b-sqrt(b^2-4*a*c))/(2*a)
xminus (generic function with 1 method)
```

If you make an error, press the up-arrow key, edit the line to correct the error and then press enter to run it again. By default some errors result in a long series of messages called a stack trace. While overwhelming at first, such stack traces will become easier to understand over time.

When solving $x^2 + x = 5$ the values of a, b and c in the quadratic formula are $a = 1$, $b = 1$ and $c = -5$. Therefore, the two solutions may be obtained by typing `xplus(1,1,-5)` and `xminus(1,1,-5)` in the Julia read-evaluate-print loop. Your screen should now look like

```
julia> xplus(1,1,-5)
1.79128784747792

julia> xminus(1,1,-5)
-2.79128784747792
```

The above answers appear fine, but there is a difficulty that becomes more obvious when the size of b is large compared to a and c. In that case $\sqrt{b^2 - 4ac} \approx |b|$ and depending on whether b was positive or negative either `-b+sqrt(b^2-4*a*c)` or `-b-sqrt(b^2-4*a*c)` involve the subtraction of two nearly equal numbers and a resulting loss of precision.

To avoid this difficulty make a difference of squares and simplify as

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}} = \frac{2c}{-b - \sqrt{b^2 - 4ac}}$$

Combining the above with a similar simplification for the other root yields

$$x = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}}.$$

Note that $\pm$ in the original form of the quadratic formula has been switched to $\mp$ in new formula. Thus, loss of precision can be avoided by combining the formulas and computing the roots when $b < 0$ as

$$x_{\text{plus}} = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad \text{and} \qquad x_{\text{minus}} = \frac{2c}{-b + \sqrt{b^2 - 4ac}}$$

but when $b > 0$ as

$$x_{\text{plus}} = \frac{2c}{-b - \sqrt{b^2 - 4ac}} \qquad \text{and} \qquad x_{\text{minus}} = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Create two new functions called `xplusnew` and `xminusnew` with the square root in the denominator. Use Julia to verify that

```
xplus(1,1,-5) ≈ xplusnew(1,1,-5)
xminus(1,1,-5) ≈ xminusnew(1,1,-5)
```

So far we have done all our work interactively. There is no program nor program output but just the transcript of the interactive session. It is tempting to take a photograph of the screen and turn that in; however, in anticipation of more involved calculations there is a different way.

Submitting Your Work

In general two things should be uploaded for grading:

- A working computer program.
- The output from running the program.

Keeping your work organized to avoid mistakes and confusion is important for any type of science. Therefore, it may be useful to place your files in working directories specific for each lab assignment. I have found naming the directory `lab01` and following a consistent pattern works well.

Although it is possible to create new directories and then organize your files by means of drag and drop with a mouse, consider the alternative of using the command line. Advantages of the command line are that

- Tasks performed from a text-based interface are possible to automate as scripts.
- After this class you may find yourself using a cloud or high-performance computing cluster with only a terminal interface.
- Using the command line for simple things flattens the learning curve for when we need it to perform other tasks in the future.

To gradually become more familiar with the command line leave the window with Julia running and open a new terminal window. Type `mkdir lab01` at

the prompt to make a subdirectory and then change your working directory to the new one by typing `cd lab01`.

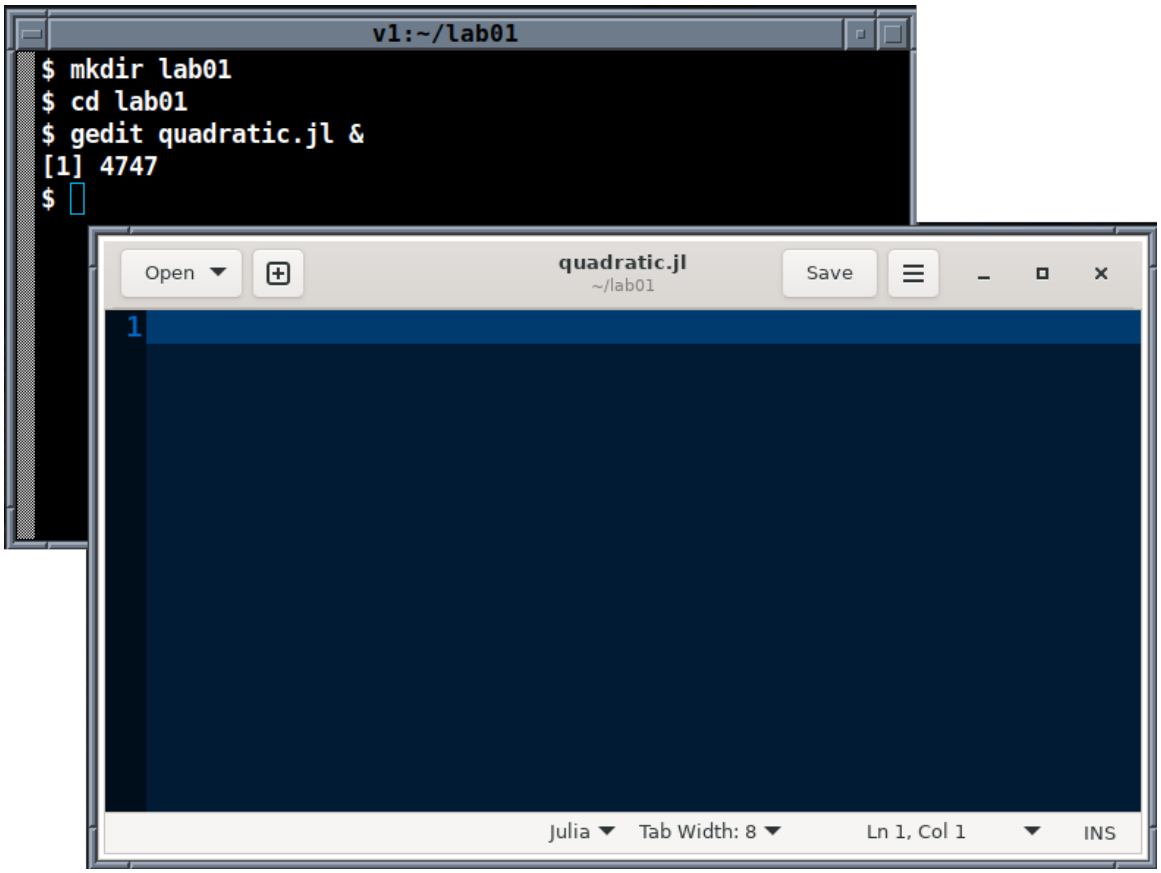
At this point the output should look like

```
$ mkdir lab01
$ cd lab01
$
```

Note the commands `mkdir` and `cd` produce output only when they fail. Thus, the fact that no additional output appears indicates they were successful.

We now convert the interactive Julia session for solving the quadratic equation into a real computer program. Writing a program also provides something to submit for grading.

To this end create a file called `quadratic.jl` in an editor such as `gedit` or `pluma`. This may be done by typing `gedit quadratic.jl &` at the prompt in the terminal window. An editor window should open up. The relevant windows on the screen might look like



Note carefully the `&` typed at the end of the `gedit` command above. This is necessary so the terminal window will continue to accept commands while the editor is running. Not typing the `&` can lead to confusion. If this happens close the editor window and try again.

The next activity is to copy and paste the commands from the interactive Julia session into the editor. Since ⟨ctrl⟩-`c` means interrupt the process in the terminal window, then ⟨ctrl⟩-⟨shift⟩-`c` must be used to copy text from the Julia REPL. To find the definition of `xplus` type first two letters `xp` and repeatedly press the up-arrow key until the definition appears. Highlight the definition and press ⟨ctrl⟩-⟨shift⟩-`c` to copy it to the mouse. Finally click on the editor and press ⟨ctrl⟩-`v` to paste it.

Adding ⟨shift⟩ to the usual copy and paste keystrokes is not needed for the editor window but only the terminal. In particular, to copy text the other direction press ⟨ctrl⟩-`c` in the editor, select the terminal window and then press ⟨ctrl⟩-⟨shift⟩-`v` to paste the result into the interactive session.

Continue copying commands from the Julia REPL to the editor until the contents of `quadratic.jl` looks like

```
1 xplus(a,b,c)=(-b+sqrt(b^2-4*a*c))/(2*a)
2 xminus(a,b,c)=(-b-sqrt(b^2-4*a*c))/(2*a)
3 xplusnew(a,b,c)=0   # fix this formula
4 xminusnew(a,b,c)=0  # fix this formula
```

Note that the definitions of `xplusnew` and `xminusnew` are obviously wrong in lines 3 and 4. Please change these lines to the correct formulas. The line numbers which appear to the left are not part of the file. Depending on the settings the line numbers may not be displayed by the editor. As such line numbers are often referred to in the error messages reported by Julia please turn them on if they do not appear.

Numbering the lines also provides a convenient way to identify different parts of the program for further discussion in this document.

The above are the definitions of the functions which compute the quadratic formula. To make a complete program, we need to add some lines which produce output. In the REPL the return value of each command is automatically displayed in the interactive session. In a program one can use `println` or `display` to obtain output. Use `println` for today.

The finished program is now

```
xplus(a,b,c)=(-b+sqrt(b^2-4*a*c))/(2*a)
xminus(a,b,c)=(-b-sqrt(b^2-4*a*c))/(2*a)
xplusnew(a,b,c)=0   # fix this formula
xminusnew(a,b,c)=0  # fix this formula
println("xplus=",xplus(1,1,-5))
println("xplusnew=",xplusnew(1,1,-5))
println("xminus=",xminus(1,1,-5))
println("xminusnew=",xminusnew(1,1,-5))
```

Press ⟨ctrl⟩-`s` in the editor to save the file. Leave the editor window open and switch to terminal window that does not have Julia running in it. Type `ls` at the prompt to check the code has been saved. Modulo any mistakes made earlier the contents of that window should now be

```
$ mkdir lab01
$ cd lab01
$ gedit quadratic.jl &
[1] 4747
$ ls
quadratic.jl
$
```

If `quadratic.jl` doesn't appear in the `lab01` directory, chances are that it had been saved elsewhere. Go back to the editor and check; otherwise, it is time to run the program.

Type `julia quadratic.jl` to run the program. The output should be

```
$ julia quadratic.jl
xplus=1.79128784747792
xplusnew=1.7912878474779201
xminus=-2.79128784747792
xminusnew=-2.7912878474779204
$
```

Once the program is working properly, it is time to make an output file to upload along with the program source for grading. This can be done

by means of a slight modification to the command just used to run the program. Type `julia quadratic.jl >quadratic.out` to place the output in the file `quadratic.out`. Then type `cat quadratic.out` to check that the output is as expected.

The results should look like

```
$ julia quadratic.jl  >quadratic.out
$ cat quadratic.out
xplus=1.79128784747792
xplusnew=1.7912878474779201
xminus=-2.79128784747792
xminusnew=-2.7912878474779204
$
```

If you have reached this point, then congratulations you have finished the lab. Only one more step needs to be completed in order to submit your work: The files `quadratic.jl` and `quadratic.out` need to be converted to PDF format so they can be viewed and annotated by the course management system. In the lab the command

```
j2pdf -o submit01.pdf quadratic.jl quadratic.out
```

may be used to produce a file `submit01.pdf` suitable for uploading.

On Microsoft Windows suitable files for upload can be generated by printing `quadratic.jl` and `quadratic.out` to PDF from within Notepad. For example, if `code01.pdf` corresponds to the printout of `quadratic.jl` and `run01.pdf` to `quadratic.out`, then upload both `code01.pdf` and `run01.pdf` for grading. A similar technique will work for MacOS.

Subsequent lab assignments will require more independent work but follow a similar sequence of steps to produce files suitable for upload that can be viewed and annotated by the grader.

Before leaving don't forget to close the applications open on your desktop and logout. Exit the Julia REPL by typing ⟨ctrl⟩-`d` and then ⟨ctrl⟩-`d` again to close the terminal. The editor has a menu at the top. If using one of the lab computers, please reboot it into Microsoft Windows.