Workshop 2: Lissajous curves

In this workshop you will use OpenMP to parallelise a program which calculates Lissajous curves. In order to parallelise the program correctly you will need to consider variable scoping and whether there are any loop-carried dependencies.

The workshop is designed to be carried out using the Linux PCs in the Lovelace laboratory. It is assumed that you are not on campus and will access these computers remotely. On the module ELE page there are instructions describing how to log in to a remote Linux computer under the "Workshop 2" section.

1 Getting set up

This section describes how to log in to a Linux PC and obtain the example programs.

- Look at the Linux computer availability dashboard (`http://students.emps.ex.ac.uk/dashboard/`) and choose a computer to log in to (you may need to use the VPN to view the dashboard). The exercises in the workshop are not resource intensive but you will get better interactive response if you choose a computer where the system load is low.
- If you are connecting using SSH then open a terminal (Linux/MacOS) or command prompt/power shell window (Windows) and log in by running the command

  ```
  ssh abc123@blueXX.ex.ac.uk
  ```

  where you should replace `abc123` with your university username and replace `blueXX` with the name of your chosen computer.

  If you are connecting using PuTTY you should follow the instructions in the "Remote access using PuTTY" document instead.
- The example programs for this workshop are provided as a tar file (a tar file is a type of archive file). Take a copy of the tar file for this workshop by running the command:

  ```
  cp /secamfs/userspace/ug/shared/ecm3446/workshop2.tar .
  ```

  where the dot at the end of the command causes the file to be copied into the current working directory.
- Unpack the tar file by running the following command:

  ```
  tar xvf workshop2.tar
  ```

  This will create a directory called `workshop2` which contains the example programs.

2 Lissajous curves

Lissajous curves are defined by the equations:

$$x = A\sin(\omega_x t + \delta_x) \tag{1}$$
$$y = B\sin(\omega_y t + \delta_y) \tag{2}$$

where ω_x, ω_y, δ_x and δ_y are constant parameters. These are parametric equations where the coordinates x and y are defined in terms of a parameter t. If ω_x/ω_y is rational (i.e. it can be written as a fraction) then x and y will describe a closed curve. Some example Lissajous curves are shown in figure 1

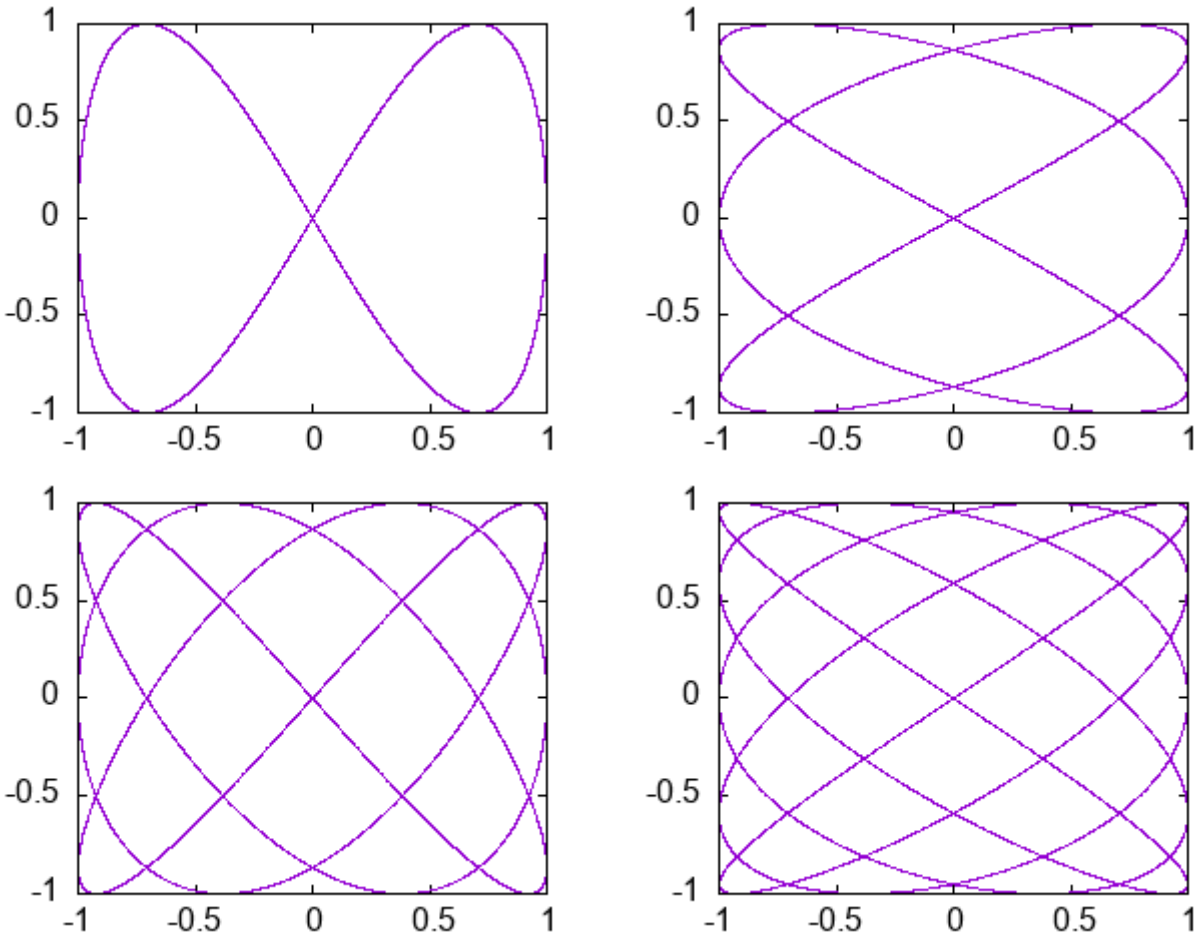

Figure 1: Example Lissajou curves: Top left $\omega_x = 1$, $\omega_y = 2$, $\delta_x = \pi/2$; top right $\omega_x = 3$, $\omega_y = 2$, $\delta_x = \pi/2$; lower left $\omega_x = 3$, $\omega_y = 4$, $\delta_x = \pi/4$; lower right $\omega_x = 5$, $\omega_y = 4$, $\delta_x = \pi/4$ ($\delta_y = 0$ for all curves).

3 The serial program

- Start by changing into the `workshop2` directory, and compile and run the example program `lissajous.c`:

  ```
  cd workshop 2
  gcc -o lissajous -std=c99 lissajous.c -lm
  ./lissajous
  ```

  When you run the program it will write a file called `output.dat` which contains the data for a Lisssajous curve with the parameters $\omega_x = 1$, $\omega_y = 2$, $\delta_x = \pi/2$, $\delta_y = 0$.

- You can now plot the Lisssajous curve using the gnuplot script provided. To plot the curve run the command:

```
gnuplot plot_lissajous
```

this will create a file in PNG format called `lissajous.png`.

- To view the file copy it to your local computer by running the following `scp` command **on your local computer**:

  ```
  scp abc123@blueXX.ex.ac.uk:/home/links/abc123/workshop2/lissajous.png .
  ```

  where you should replace `abc123` with your university username and replace `blueXX` with the name of your chosen computer. If the PNG file is not in a directory called `workshop2` in your home directory then you will need to replace `/home/links/abc123/workshop2/lissajous.png` with the path to the file. The dot at the end of the command causes the file to be copied into the current directory/folder.

  If you are connecting with PuTTY then you will need to use the command-line application `pscp` instead of `scp`. Documentation for PSCP can be found at `https://the.earth.li/~sgtatham/putty/0.74/htmldoc/Chapter5.html`.

  You can now open the file with an appropriate PNG viewer on your local device.

- The file `output.dat` is your known good output which you can use later to test that your parallel version of the program is working correctly. To prevent this file being overwritten by later versions of the program rename the file:

  ```
  mv output.dat output_kgo.dat
  ```

4 Parallelising the program

You are now ready to parallelise the program using OpenMP.

- You will need to choose a a text editor to edit the program. The `nano`, `emacs` and `vi` editors are available on the blue room PCs computers. If you are not familiar with any of these editors I recommend using `nano`. If you get stuck in `vi` type `:!q` to quit.

- Open the program in your chosen text editor and find the `for` loop in the `main` function.

  Question 1: Are there any loop carried dependencies in this loop which prevent it being parallelised?

- There is also a `for` loop in the `write_to_file` function.

  Question 2: Can the `for` loop in the `write_to_file` function be parallelised?

- Now parallelise the program using OpenMP making sure that the variables are correctly scoped. You should check that the results are correct by comparing against the known good output. You can use the `diff` command to check that the output matches the known good output from the serial version:

  ```
  diff --brief output.dat output_kgo.dat
  ```

  If there is no output from the diff command then the files are identical.

  Question 3: How did you scope the variables to make sure the parallel version gives identical output to the serial version?

5 Generating multiple curves

The `workshop2` directory contains another version of the program called `lissajous_multi.c`. This version of the program calculates four Lissajous curves for the parameters shown in figure 1. There is also a plotting script called `plot_lissajou_multiple` for plotting the results.

- Open the `lissajous_multi.c` program in a text editor and find the part of the `main` function where the Lissajous curves are calculated. This calculation is now carried out by two nested `for` loops: an inner loop over `N_POINTS` points and an outer loop over `N_CURVES` curves.

  Question 4: Which loop would you choose to parallelise and why?

- OpenMP provides a clause called `collapse` which is used to combine loops so that more than one loop can be parallelised with a single OpenMP pragma. Section 2.9.2 of the OpenMP 5.0 specification says

  > If a collapse clause is specified with a parameter value greater than 1, then the iterations of the associated loops to which the clause applies are collapsed into one larger iteration space ...

  If you add the clause `collapse (2)` to a `#pragma omp parallel for` directive it will result in the following two loops being combined for parallelisation.

  Question 5: Could you use the collapse clause with these loops and does it change your answer to the previous question?

- Lastly parallelise the `lissajous_multi.c` program using OpenMP and check that the results match the serial version.