ECM2433 CA1 – Report

Program Design

plotPNG.c is an ANSI standard C program that creates plots of mathematical functions in files in the Portable Network Graphics (PNG) format.

It was originally a program that just wrote three vertical stripes to a PNG file (using the *libpng* library), but has since been modified to output the plot of a mathematical function defined at runtime. Changes have only been made to the *main* and *makeImageData* functions, along with an additional include for the *tinyexpr* library. Everything else was left unchanged (credit: Dr Jacqueline Christmas and Guillaume Cottenceau).

The *main* function was changed so that it writes to the *stdout* and *stderr* streams in the case of warnings or errors, and aborts the program if any fatal errors are encountered. This way it can trap errors and leave meaningful messages in the appropriate streams before attempting to write to the file, rather than just running into a segmentation fault. If the program successfully creates the PNG file, it writes a meaningful message to the *stdout* stream. The program has also been modified so that it now takes two command-line arguments (file name and expression), rather than just one.

The *makeImageData* function is where all the calculation happens, and corresponding data is put into pixels to construct the image. To evaluate the expression, it now takes an array of characters (a string) as its fourth argument. This allows us to pass in the second command-line argument (the expression) to *makeImageData* in the *main* function. It also traps some more errors before attempting to evaluate the expression. If there have been no fatal errors, it then determines whether the expression is of the form $f(x)$ or $f(x,y)$. Using the *tinyexpr* library, it evaluates the expression and assigns RGB values to the corresponding pixels.

If the expression is of the form $f(x)$, it first colours the entire image white (sets all pixels RGB to {255,255,255}). Then it evaluates the expression on the interval $[0,1)$, in increments of $\frac{1}{50w}$ where w is the width of the PNG image. This prevents there being any gaps in the plotting, and ensures it scales up properly for different PNG resolutions. Then it colours corresponding pixels blue (RGB {0,0,255}), by multiplying the x values by the image width and the $f(x)$ values by the image height.

If the expression is of the form $f(x,y)$, it evaluates the expression on the interval $[0,1)$, incrementing x by $\frac{1}{50w}$ and y by $\frac{1}{50h}$, where w and h are the width and height of the image, and stores the resulting $f(x,y)$ values in a two-dimensional array. At the same time, it finds the maximum and minimum values that $f(x,y)$ takes on this interval. Then it writes RGB triplets to the corresponding pixels, based on the minimum and maximum values, and on the value of $f(x,y)$. The minimum is represented by red ({255,0,0}), the maximum is represented by blue ({0,0,255}), and values of $f(x,y)$ between these two values are assigned colours linearly between red and blue.

Assumptions

There are several assumptions I made when it comes to user input and program use:

- The program will only run if you give it exactly 3 arguments.
- The second command-line argument (file name) must be a valid PNG file. This means it must have the *.png* file extension.
- The third command-line argument (expression) is either of the form $f(x)$ or $f(x,y)$. This means the variables must be in terms of x and y, and that equations like $y = f(x)$ or $z = f(x,y)$ are invalid.
- If an expression is written in terms of y but not x, it will be treated as an $f(x,y)$ form expression (e.g. y will be treated as $y + 0(x)$).
- For expressions of the form $f(x,y)$, only square resolutions will be used (e.g. 300x300 is valid, but 300x200 isn't).
- The program will only be used at resolutions less than 1400x1400. This is because segmentation faults occur at higher resolutions (likely due to *libpng* library or the original implementation).

Error trapping

main

Fatal errors:

- Checks if exactly three command-line arguments were given.
- Checks if the filename ends with '.png'.
- Checks if the expression contains '='.

Warnings:

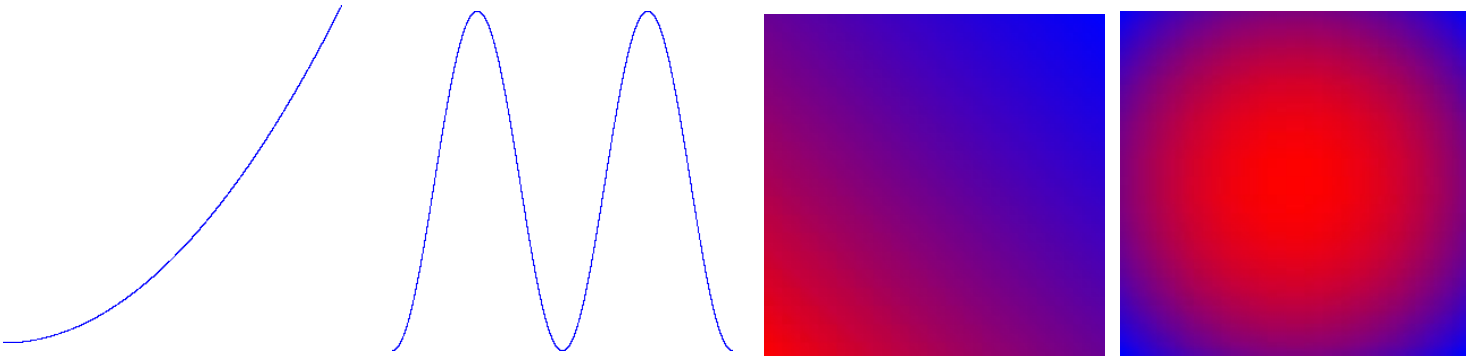
- Warns if expression is written only in terms of y (will be treated as $f(x,y)$).
- Warns if (hard-coded) width or height exceed 1400.

makeImageData

Fatal errors:

- Checks both if the expression is of form $f(x,y)$, and if the (hard-coded) resolution is non-square.
- Checks if *tinyexpr* can evaluate the expression. This handles any expressions given in terms of variables other than x or y.

Output (fig.1):



Instructions:

Note: These instructions are for Linux machines only.

The program relies on the C standard library, the *libpng* library, and the *tinyexpr* library. Therefore, make sure you have these installed before attempting to run the program.

Standard libraries come with the *GNU Compiler Collection*, which can be found here: https://gcc.gnu.org/

To install *libpng* on Linux, use the following command: *sudo apt-get install libpng-dev*

To allow the program to use the *tinyexpr* library, ensure that the *tinyexpr.h* and *tinyexpr.o* files are in the same directory as *plotPNG.c*.

To compile and link the program, open a terminal window and use the '*cd'* command to change the working directory to the directory where you extracted the contents of the zip file. Then run the shell script named *'comp'* to compile and link the program. First make it executable, by issuing the command '*chmod u+x comp*'. It can now be used to compile and link the program, with the command '*./comp plotPNG.c*'.

The directory should now contain the plotPNG executable, which can be used as with the following command: '*./plotPNG* <*file_out*> <*math_expr*>', where <*file_out*> and <*math_expr*> are to be replaced with a valid file name and math expression.