InFoMM C++ Skills Training

Practical 1 (Lectures 1-3)

If this is your first time coding in C++ then make sure you can compile the example in `examples/hello_world` from either the command line or an IDE of your choice, for example Eclipse (http://www.eclipse.org/cdt), or Clion (http://www.jetbrains.com/clion). Note that when you are working on small, similar questions then you might need to consider how to organise your work. You might prefer

- one file per question each with its own `main()` function. You may need to tweak the build rules so that each file produces an executable
- one function per question, with the `main()` function only calling the function you are testing (when you are used to writing functions)

If you need to check or visualize your results then feel free to use Matlab or any scripting tools that you know.

1. Create two `double` variables x and y. Set $x = 1$ and $y = 1$ and calculate $r = \sqrt{x^2 + y^2}$. Write the result r to standard output.
2. Generate N uniform random numbers x_i and y_i between -1 and 1. Count the number of points where $\sqrt{x_i^2 + y_i^2} < 1$, and use this to estimate the value of π.
3. Code up another estimator for π by calculating the sum of the reciprocals of square numbers (The Basel problem) for N terms, which converges to $\pi^2/6$ for large enough N.

$$S = \sum_{n=1}^{n=N} \frac{1}{n^2} \to \frac{\pi^2}{6}$$

4. Finally, code up the Gauss-Legendre algorithm for estimating π, which has quadratic convergence.
5. Write code to implement the backward Euler method to solve the ODE

$$\frac{dy}{dx} = -y \quad y(0) = 1.$$

on the interval [0 1]. Your code should print a file called `xy.dat` that has two columns: the calculated values of `x`; and the calculated values of `y`. Read this file into Matlab or Python and plot the solution.

[*The backward Euler method for this problem results in the difference relation*

$$\frac{y_{n+1} - y_n}{h} = -y_{n+1}.$$

where h is step size]

6. Write code that reads the file `xy.dat` created in the previous exercise, and computes the error at each point (the true solution is $y = e^{-x}$. Print to the screen the maximum error.
7. Write code to calculate the scalar (dot) product of two `std::array<double,3>` variables
8. Write code to multiply two 3 x 3 matrices. Think about how you would store your matrices. You could use a flat array `std::array<double,9>`, or you could use nested arrays `std::array<std::array<array,3>,3>`.