SUPA C++ Labs 3+4

Jonathan Jamieson

November 29, 2023

The assignments here make up the second half of the lab portion of this course. In the first half you built up a complex piece of software in steps that allowed you to manipulate input data with a focus on functions, user input, and producing simple text output. We will now put what we learnt in the last labs into practice and build up some complex functionality without as many explicit steps. The focus this time will be on classes, inheritance and adapting pre-written code and should explore topics that we covered during the lessons on 29/11 and 06/12.

The assignment is due by the 13^{th} of December

Always write your name and date of creation as comments in the files and try to be "user friendly" when you write code:

- Comments are your friend, add them as you go to keep track of what the code (should) be doing, both for yourself and for anyone who has to read your code in the future
- Try and stress-test your code to account for any possible errors due to edge cases or mistakes in the input given by the user

1 Preliminary

In the repository for this assignment there is a pre-compiled executable called:

"GenerateRandomData"

run it in the terminal and it will create file at:

"Output/Data/MysteryDataXXXXX.txt"

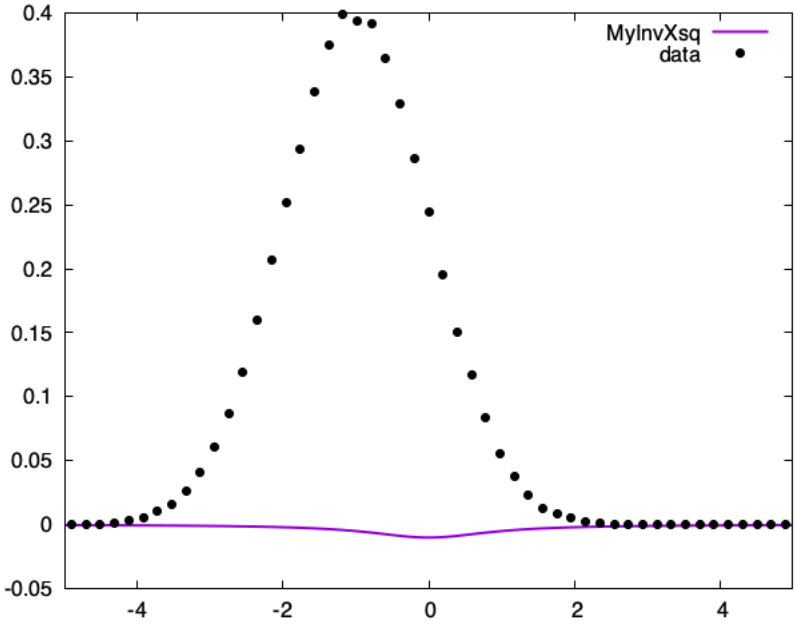
Where XXXXX is a random string of numbers. This file contains 1D datapoints which follow some unknown statistical distribution. The aim of this assignment is to work out what underlying distribution this data has been sampled from, with the end goal of using the same function to draw new simulated data points. Rather than trying to fit the data directly you have access to an existing base class "FiniteFunctions.cxx" which is used for evaluating and plotting finite functions. By default the function: $f(x) = \frac{1}{1+x^2}$ is used.

2 Main assignment

Write a script to test the default version of the FiniteFunctions class, there is no formal documentation so you will need to read the source code in order to work out how to use it. Without adding any new code to the class description you should be able to produce a plot of the function $f(x) = \frac{1}{1+x^2}$ and the data points which you will need to read in from the "MysteryData.txt" file.
Hint: *Make sure to pick limits within which the function is approximately finite.*

You can use the provided makefile to compile everything together by just running make. (If you don't want to use a makefile you will need to include the flags: -I ../../GNUplot/ -lboost_iostreams in your compiler command.)

You should get a plot which looks something like this:



Uh oh, looks like there is something wrong with the normalisation of the function, try to fix it and re-make the plot. **Hint:** *Check if any of the class methods are incomplete*

Once you have the normalisation fixed you will still see that the data still doesn't match the shape of: $f(x) = \frac{1}{1+x^2}$ very well so we will need to add in some new functional forms to test. Add these in as new classes which inherit from the base FiniteFunction class and which represent the following distributions:

1. The normal distribution: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$
2. The Cauchy-Lorentz distribution: $f(x) = \frac{1}{\pi\gamma\left[1+\left(\frac{x-x_0}{\gamma}\right)^2\right]},\ \gamma > 0,$
3. The negative Crystal Ball distribution: $f(x) = N \cdot \begin{cases} e^{-\frac{(x-\bar{x})^2}{2\sigma^2}}, & \text{for } \frac{x-\bar{x}}{\sigma} > -\alpha \\ A \cdot (B - \frac{x-\bar{x}}{\sigma})^{-n}, & \text{for } \frac{x-\bar{x}}{\sigma} \leq -\alpha \end{cases},$

   With:

   $n > 1, \alpha > 0$
   $A = \left(\frac{n}{|\alpha|}\right)^n \cdot e^{-\frac{|\alpha|^2}{2}},$
   $B = \frac{n}{|\alpha|} - |\alpha|,$
   $N = \frac{1}{\sigma(C+D)},$
   $C = \frac{n}{|\alpha|} \cdot \frac{1}{n-1} \cdot e^{-\frac{|\alpha|^2}{2}},$
   $D = \sqrt{\frac{\pi}{2}}\left(1 + \text{erf}\left(\frac{|\alpha|}{\sqrt{2}}\right)\right).$

Hint: *It might help to look at the member declarations in FiniteFunctions.h before you write your custom classes to see which functions we should override.*

In the end you should have a (single) .h and .cc file for the set of 3 custom function classes, as well as an executable file which you can use for testing. This should have a main function which calls the constructor for each of your classes. In each case you should then calculate the integral using a sensible number of intermediate sample points, print out info about your function, and then produce plots of your function vs the mystery data. Play around with the parameters for each function until you think you have found the distribution that the data was sampled from.

2.1 Sampling

Now we have the right distribution we want to be able to sample from it to produce more pseudo-data, for this we can use the metropolis algorithm. **Note:** this method only works for finite functions so make sure the range you choose for your function is wide enough.

1. Generate a random number x_i on the range where the function is defined, sampled from a uniform function.
2. Generate a second random number y from a normal distribution centred on x_i and using an arbitrarily chosen standard deviation.
3. Compute $A = \min\left(\frac{f(y)}{f(x_i)}, 1\right)$, where f is the function.
4. Generate a random number T, between 0 and 1, if $T < A$ then accept y.
5. If you accepted y set $x_{i+1} = y$, otherwise $x_{i+1} = x_i$.

Add a method to your classes which samples from the class function using the metropolis method **Hint:** *Can you get this to work for all of your custom classes without repeating any code?*.

As a final step produce some sampled data and create a plot which overlays the function, sampled data, and mystery data on the same axes. Does the metropolis sampling work well? Try adjusting the width of the gaussian used in the sampling to see if you can improve it.

3 Bonus

Write a script which takes two arguments at the command line: *radius*, and *n_random*, and uses these to calculate π using (pseudo) random numbers. Print the calculated value to the terminal with 10 decimal places.
Hint: *Think about comparing the areas of a square and a circle*