Below is a log of the debugging done for Armadillo. Most of the bugs stemmed from small oversigths made in the translation phase (using the log_det function as opposed to the logdet variable) or simple typos (having a Y1 where Y2 was meant to be), however a few stem from inherent differences between Matlab and C++. I have highlighted the major difference between the two coding languages in the debugging log for each of these issues, and below is a brief discussion of them:

The log function is different between Matlab and C++. Matlab's log function implements a complex log, so it can return a value when taking the log of a negative number whereas C++ uses a normal log and so returns NaN (not a number) when trying to take the log of a negative number.

Matlab matrix indices start at 1 and C++ indices start at 0. That is, to reference the first element in a matrix, the Matlab would be referenced as MAT(1,1) and the C++ would be referenced as C(0,0)

C++ requires the programmer to declare what type of variable each variable is when it is declared. That is, when you declare a variable, you have to make it an int, double, etc. Matlab does not require this, so there can be some ambiguity in what type of variable something is in Matlab.

- The first issue I'm debugging is coming from the fact that abs(eta) is returning NaN (not a number) instead of a value
  - i've tracked the reason for this down to logdetLadYnew
  - i think the issue is that log(eigLadY) is not a double, so the sum of it doesn't fit into a double
  - what's causing the issue is that eigLadYnew is getting some negative values, and then since we're trying to take the logarithm of it, it's resulting in an undefined value and that throws everything off
  - found the root of the issue: matlab's log function is a complex log, so it returns a complex value for the log of a negative number. researching how to implement that in c++ now
  - making a variable mat ladYnew that's initialized by what is currently the intput to vec eigladYnew = real(eig_gen()) and then using log_det(ladYnew) outputs the correct logdetLadYnew
- 3 lines after the if options == 1, there's a function called trace. C++ is returning a value 0.3 greater than the value that matlab is returning. looking for cause of issue
  - issue resolved by fixing indexing issue—Matlab index starts at 1 whereas C++ index starts at 0. trace function now returns correct value
- for loop with j 1:50. for N = 2 matlab runs through the loop more times than c++
  - else if condition had a Y1 where there should have been a Y2
- Duality gap eta calculation returns wrong value in C++
  - eta calcuation was using log_det(X) instead of the value stored in variable logdetX.

- first iteration works and produces the same values for everything in matlab and c++. however, c++ code is not every converging
    - Rnew1 calculates incorrectly during the second iteration of ama (and presumably for subsequent iterations as well)
    - eta calculation also yields different result in second iteration; seems to e not dependent on Rnew1 in any way (so separate issue)
    - eta issue stemming from miscalculation of dualYnew
    - (1 - a / abs(Svec)) is calculating wrong in Svecnew
    - although this wasn't an issue for the first iteration... and Svec is right during this second iteration.... this is weird
    - variable a was being declared as an int instead of a double