In generating the module to be used in my pi.f90 file, I used implicit none to remove any implicitly defined variables. I defined pi_true as equaling the arccosine of -1. The subroutine performed was named pisolver with arguments thresh and N_max. All variables were declared explicitly. To approximate pi, a do loop from n=0 to N_max was used. The equation pisum was used to define the sum giving pi. The difference was also defined and if the difference became smaller than the threshold for a given value the process was exited and pi_appx was printed as well as the n value used to reach that difference. Within the code itself, the pimod module was used and pisolver was called for the four different threshold values. The sum converged more slowly with a smaller threshold. For 1.0d-4 it took 2 values of n, for 1.0d-8 it took 4, for 1.0d-12 it took 7, and for 1.0d-16 it took 10. The compiler flags gave all warnings, any extra warnings, implicit interface warnings, max errors being 1 for readability, position independent code, debugging info for gdb, run-time tests such as an array bounds check, and when a run-time error or a deadly signal is emitted the fortran run-time library
will output a backtrace of the error. A makefile helped to easily use all of these flags while also allowing the ease of compiling all necessary .f90 files and turning them into an executable. It also a loud for easy cleaning of *.o files.

As for debugging, my strategy involved reading over the subroutines to make sure that matched the necessary order of operations to perform row reduction of the given matrices. Next, I decided to create a separate module for the subroutines to avoid using an interface and make the code neater. I then scanned through both for syntax errors and grammar errors. The main program file needed to be changed to a *.F90 file with to enable the C-style preproccesor. The choice of compiler flags were similar to those used for the pi compilation. The intent was removed as they are inherently inout. -g3 gives more info than -g. The default flags for double precision were included so as to prevent having to explicitly state their kind as well as ensure their wasn't any kind=4 going into kind=8 and vice versa. The flag -Wno-maybe-uninitialized was added to circumvent a bug inherent to the GCC compiler which gives warnings for a common uninitialized 'a.dim[0].lbound'.