[The following is based on the recommended g++ options from the Reference Manual for the GNU Scientific Library --- Debugging Numerical Programs. Note that compilers updates may render some of these obselete.]

C++ options for numerical programs

Writing reliable and portable numerical programs in C++ requires care. It is a good practice to compile and test your program with (at least) two different compilers.

The following minimum g++ (GNU compiler) and icpc (Intel compiler) options are recommended when compiling numerical programs:

```
g++ -Werror -Wall -Wextra -Wshadow -fno-common -g -O2

icpc -Werror -Wall -fno-common -g -O2
```

The following list gives a brief explanation of what types of errors these options catch.

-Werror
Consider warnings to be errors, so that compilation stops. This prevents warnings from scrolling off the top of the screen and being lost. You won't be able to compile the program until it is completely warning-free. You may be tempted to remove this. DON'T!

-Wall
This turns on a set of warnings for common programming problems. You need -Wall, but it is not enough on its own (for g++).

-Wextra
This turns on some extra warnings in g++ not included in -Wall, such as missing return values and comparisons between signed and unsigned integers.

-Wshadow
This warns whenever a local variable shadows another local variable. If two variables have the same name then it is a potential source of confusion.

-fno-common
This option prevents global variables being simultaneously defined in different object files (you get an error at link time). Such a variable should be defined in one file and referred to in other files with an extern declaration.

-O2
Turn on optimization. The warnings for uninitialized variables in -Wall rely on the optimizer to analyze the code. If there is no optimization then the warnings aren't generated. When first debugging, use -O0 (no optimization) until ready for production runs.

-g
It always makes sense to put debugging symbols in the executable so that you can debug it using gdb. The only effect of debugging symbols is to increase the size of the file, and you can use the "strip" command to remove them later if necessary.

```
The following is a more complete list of g++ warning options, which catch
less common problems and ensure greater portability.

g++ -Werror -Wall -W -Wshadow -fno-common -g -O4
  -ansi -pedantic -Wconversion -Wpointer-arith -Wcast-qual
  -Wcast-align -Wwrite-strings -fshort-enums

The following list gives a brief explanation of what types of errors
the additional options catch.

  -Wpedantic -std=c++03 [or -std=c++11 or -std=c++14 for other ISO standards]
     Use ISO C++, and reject any non-ANSI extensions. These flags help in
     writing portable programs that will compile on other systems.

  -Wconversion
     The main use of this option is to warn about conversions from signed
     to unsigned integers. For example, unsigned int x = -1. If you need
     to perform such a conversion you can use an explicit cast.

  -Wpointer-arith -Wcast-qual -Wcast-align
     These options warn if you try to do pointer arithmetic for types
     which don't have a size, such as void, if you remove a const cast
     from a pointer, or if you cast a pointer to a type which has a
     different size, causing an invalid alignment.

  -Wwrite-strings
     This option gives string constants a const qualifier so that it will
     be a compile-time error to attempt to overwrite them.

  -fshort-enums
     This option makes the type of enum as short as possible. Normally
     this makes an enum different from an int. Consequently any attempts
     to assign a pointer-to-int to a pointer-to-enum will generate a
     cast-alignment warning.
```