Generic Checklist for Code Reviews

Structure

☑ Does the code completely and correctly implement the design?

Yes, the code is a part of the MVC design

☑ Does the code conform to any pertinent coding standards?

Yes: Humpback notation ued consistently, no global variables

☑ Is the code well-structured, consistent in style, and consistently formatted?

Yes, it is consistently indented, in-line comments are aligned

☒ Are there any uncalled or unneeded procedures or any unreachable code?

No, all code is reachable and subsequently called

☒ Are there any leftover stubs or test routines in the code?

No, there are no leftover stubs or test routines

☒ Can any code be replaced by calls to external reusable components or library functions?

No, this header file has called and used applicable library functions

☑ Are there any blocks of repeated code that could be condensed into a single procedure?

Yes - types could be put into a data structure that would allow access to each var via looping (array)

☑ Is storage use efficient?

Yes, despite being ugly and complex the structure is efficient

☒ Are symbolics used rather than "magic number" constants or string constants?

No, there is usage of undefined numbers ('44')

☑ Are any modules excessively complex and should be restructured or split into multiple routines?

Yes, when reading in the from csv, the strings could be stored in a vector

Documentation

☒ Is the code clearly and adequately documented with an easy-to-maintain commenting style?

No, there are very few comments and some of them are unhelpful "killme"

☑ Are all comments consistent with the code?

Yes, the formatting is consistent and the comments refer to the following code

Variables

☒ Are all variables properly defined with meaningful, consistent, and clear names?

No, temp1 - temp44 are not very descriptive

☑ Do all assigned variables have proper type consistency or casting?

Yes, there are no type conflicts

☒ Are there any redundant or unused variables?

No, all variables are used and have a purpose

Arithmetic Operations

☑ Does the code avoid comparing floating-point numbers for equality?

Yes, no usage of floats

☑ Does the code systematically prevent rounding errors?

Yes, no usage within code

☑ Does the code avoid additions and subtractions on numbers with greatly different magnitudes?

Yes, no such actions occur within the code

☒ Are divisors tested for zero or noise?

No, not applicable to code

Loops and Branches

☑ Are all loops, branches, and logic constructs complete, correct, and properly nested?

Yes, there are no functional errors regarding the logic in the code

☒ Are the most common cases tested first in IF- -ELSEIF chains?

No, due to the nature of code, the occurrence of a common case is unpredictable

☑ Are all cases covered in an IF- -ELSEIF or CASE block, including ELSE or DEFAULT clauses?

Yes, all cases are covered by if-else if statements

☒ Does every case statement have a default?

NOT applicable -- no case statements used

☑ Are loop termination conditions obvious and invariably achievable?

Yes, all loops end predictably

☑ Are indexes or subscripts properly initialized, just prior to the loop?

Yes, indexes are initialized within or before the loop

☒ Can any statements that are enclosed within loops be placed outside the loops?

No, the inner loops are logically placed for the required functionality of the code

☑ Does the code in the loop avoid manipulating the index variable or using it upon exit from the loop?

Yes, the code only uses the index variable as a reference point, within the loop.