Automated Grading of LabVIEW Files

Dr. Keith Hekman, California Baptist University

Dr. Keith Hekman is a full professor in Mechanical Engineering. He has been at California Baptist University for fifteen years. Prior to teaching at CBU, he taught at Calvin College and the American University in Cairo. His Ph.D. is from the Georgia Institute of Technology. His recent research has been focused on developing automated grading for engineering courses.

Automated Grading of LabVIEW Tutorial Files

Abstract

Instructors frequently use automated grading in programming classes. Institutions have developed graders for C++, Java, MATLAB, and many other programming languages. LabVIEW is a graphical programming language that people frequently use for data acquisition. Since there were no automated grading programs for LabVIEW, a computerized grading system has been developed. With the grading program, students email the LabVIEW files they have written, and the program provides their assignment score and feedback concerning missing program functions or wires. Students then can resubmit their work until the due date. The grading program was implemented in a LabVIEW programming course at California Baptist University using NI's LabVIEW Core 1 and Core 2 curricula. When using the grading system, students appreciated the immediate feedback from the program, and the instructor/teaching assistant's grading time was reduced.

Keywords

LabVIEW, Automated Grading.

Background

Automated grading for programming assignments has become quite common. ZyBooks [1] offers automatic grading in Java, C, C++, Python, and Web Programming. MATLAB also provides automated grading with MATLAB Grader [2]. In addition to a reduced TA/professor workload, the instant feedback helps students quickly discover what they are doing wrong, assisting the learning process.

Ihantola *et al.* did a literature review of 80 papers concerning the automatic assessment of programming assignments. [3]. They found that automated evaluations is used in programming courses to ensure that students get enough practice and feedback on the quality of their code. Testing included input-output comparison, scripting, and experimental approaches. They indicated that not all types of programming assignments lend themselves to automated grading. Caiza and Del Alamo [4] also provide a review of the various tools available for automated grading. More recently, Aldriye et al. [5] provided another literature review of automated grading systems for programming assignments. They found that grading could be based on unit testing, statistical error modeling, peer-to-peer feedback, random input test cases, and pattern matching. Clearly, there is a great interest in having a computer grade student's programming assignments. However, no research has been done in grading the LabVIEW programming language.

LabVIEW Automated grading program description

"LabVIEW is a graphical programming environment engineers use to develop automated research, validation, and production test systems." [6] Many people find the graphical programming structure more accessible than text-based coding. LabVIEW has built-in analysis functions and drivers for communicating with different instruments and acquisition hardware. The program is designed to grade LabVIEW tutorial problems where the textbook provides an image of the desired program. Since LabVIEW is an interactive language (e.g., click on a

button), the grading program only grades the student's program's structure, not the program's functionality. After the students complete the tutorial, they email their files to a dedicated email account for the automated grader. The program then determines the items in the program and the wire connections between them. The program can identify the following LabVIEW block diagram objects:

- Numeric Constant
- Enum Constant
- Cluster Constant
- Boolean Constant
- Control
- For Loop
- While Loop
- Event Structure
- Case Structure
- Named bundler
- Named Unbundler
- Function
- Growable Function
- VI
- Disable Structure
- Comparison
- In Range And Coerce
- SubVI
- Bundler
- Unbundler
- Build Array
- Compound Arithmetic
- Index Array
- Property Node
- Invoke Node
- Format Scan String
- Fixed Constant
- Local
- Digital Numeric Constant
- Error Ring
- Flat Sequence
- Array Constant
- Control Reference Constant
- Conditional output terminal

After identifying the block diagram objects and the wires, the program compares the student's file to the instructor-provided key. The program checks to ensure that every object is present and that the wires between the appropriate terminals are correctly connected.
When there is a difference between the student program and the instructor solution, the program highlights the different items in both codes. The program includes images of both codes for the student in an email reply. If there is a case structure, each case has a different picture in the LabVIEW Report. An example of the text of the email is below in Table 1, Sample Program Email Text. The email indicates that the student is using a different subVI than the key. This is also seen in the block diagram images sent along with the text, with the subVI highlighted in Figure 1 for the key and Figure 2 for the student. The program also highlighted the wire between the SubVI and the bundle by name terminal because it does not match since it has a different starting subVI. The other wiring error shows up in Figure 3 and Figure 4, where the student forgot to wire the "temperature warning text" in the unbundle by name to the FormatScanString. The mistake caused the program to highlight the tab constant, as the student did not include the last wire. The program also highlighted the "end of line" constant because the student wired it into a different terminal number than in the key.
Table 1, Sample Program Email Text

Objects 80/82 Wires 132/139 Total score 4.80/ 5.00 The key has the following extra items: SubVI named "Thermometer (Demo).vi" The student has the following extra items: SubVI named "Thermometer.vi"

The key has the following extra wires:
Wires between:
SubVI <Thermometer (Demo).vi> (Terminal 1), Control <Temperature History> (Indicator), Named bundler <Bundle By Name> (Terminal 2),
Named Unbundler <Unbundle By Name> (Terminal 5), FormatScanString <Format Into String> (Terminal 13),
FixedConstant <
> ( ), FormatScanString <Format Into String> (Terminal 14),
FixedConstant < > ( ), FormatScanString <Format Into String> (Terminal 6), FormatScanString <Format Into String> (Terminal 8), FormatScanString <Format Into String> (Terminal 10), FormatScanString <Format Into String> (Terminal 12),

The student has the following extra wires:
Wires between:
SubVI <Thermometer.vi> (Terminal 2), Named bundler <Bundle By Name> (Terminal 2), Control <Temperature History> (Indicator),
FixedConstant < > ( ), FormatScanString <Format Into String> (Terminal 6), FormatScanString <Format Into String> (Terminal 8), FormatScanString <Format Into String> (Terminal 10),
FixedConstant <
> ( ), FormatScanString <Format Into String> (Terminal 12),

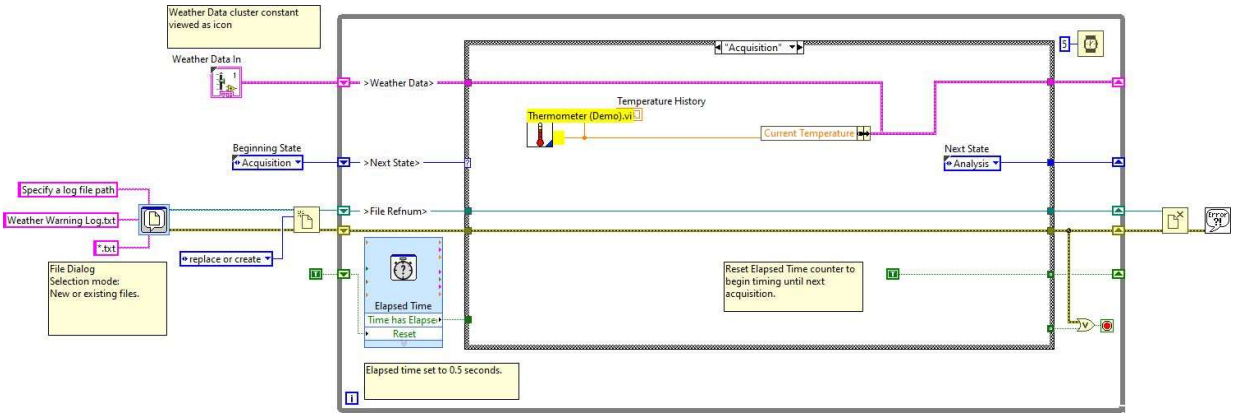

Figure 1, Key with wrong subVI highlighted

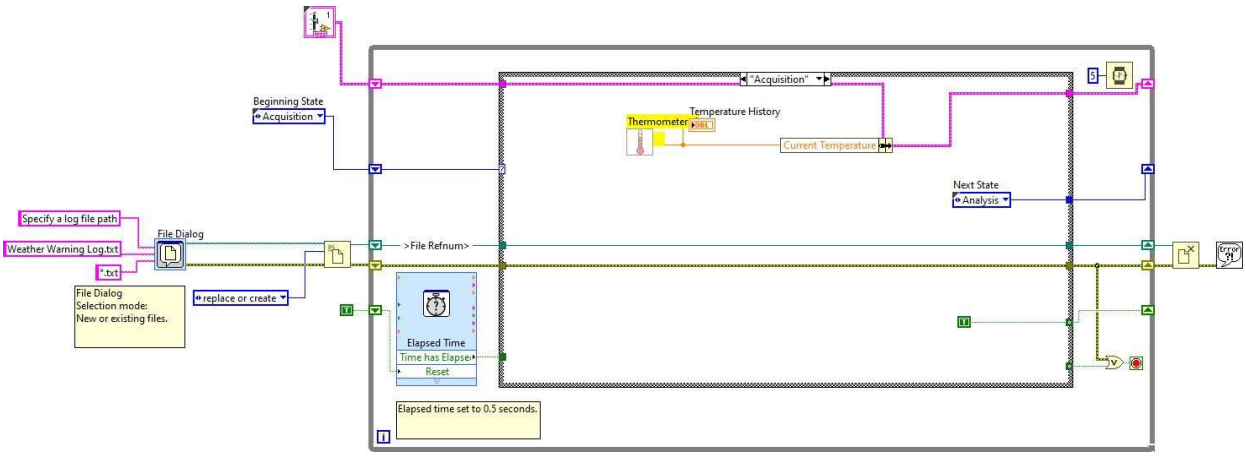

Figure 2, Student file with wrong SubVI highlighted

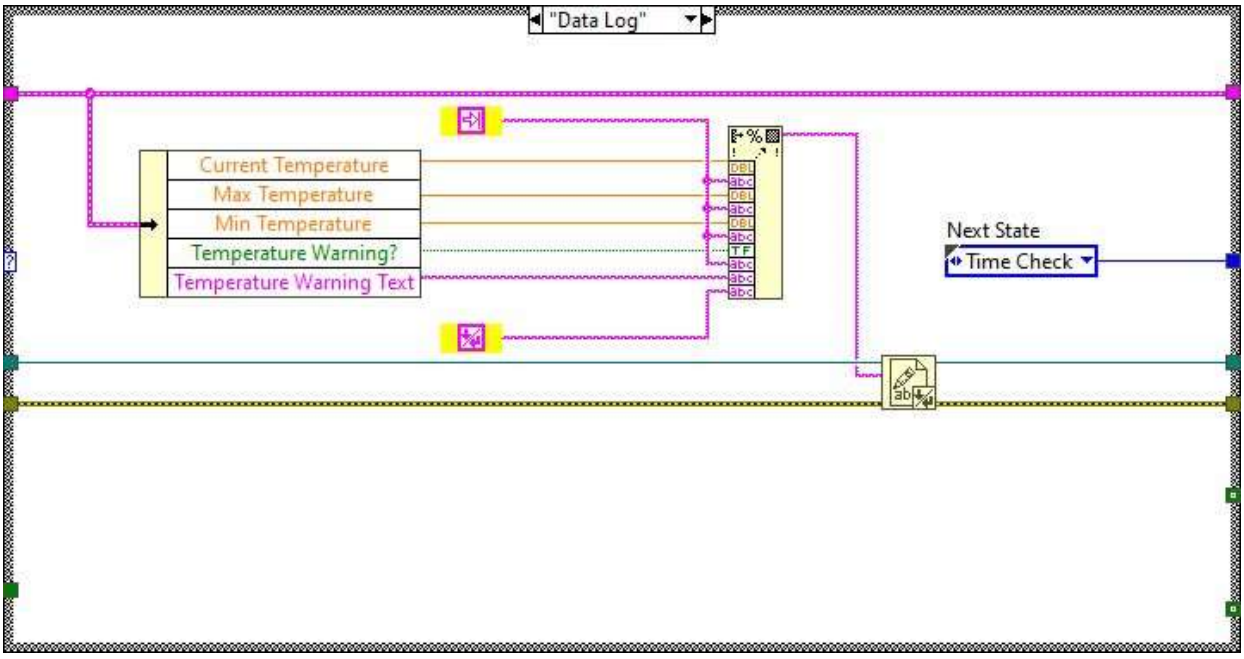

Figure 3, Key file with wiring problem highlighted

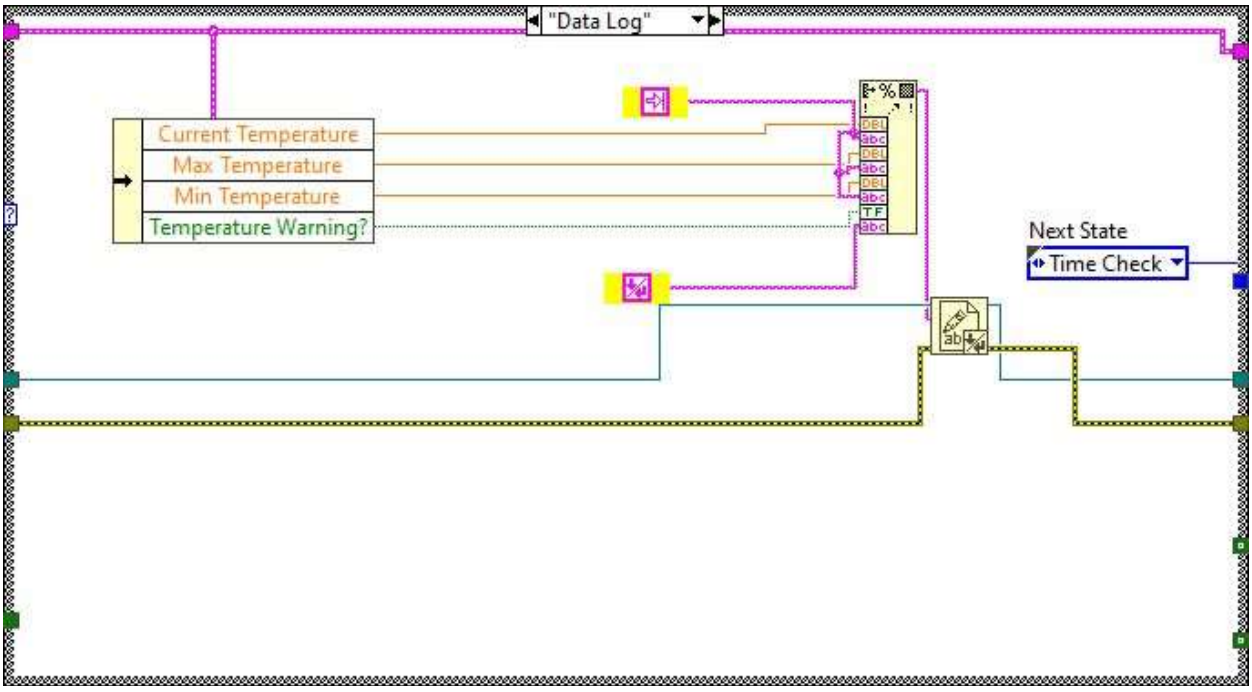

Figure 4, Student File with wiring problem highlighted.

Another example of the program's feedback involves a case where the student has a missing wire. The text feedback can be seen below in Table 1. From the text, there is a problem with the wiring. Figure 5 shows the solution case in the case structure with the error, while Figure 6 shows the corresponding student case in the block diagram. In the graphical feedback of the block diagram, the program highlighted the wire coming out of the build array function, indicating the mistake in that wire. The student forgot to wire the Concatenated Data Array output on the front panel.

Table 2, Sample Program Email Text with missing wire

Objects 110/110 Wires 118/120 Total score 4.96/ 5.00 The key has the following extra wires: Wires between: BuildArray <Build Array> (Terminal 0), Control <Concatenated Data> (Indicator), Control <Concatenated Data Array> (Indicator), The student has the following extra wires: Wires between: BuildArray <Build Array> (Terminal 0), Control <Concatenated Data> (Indicator),

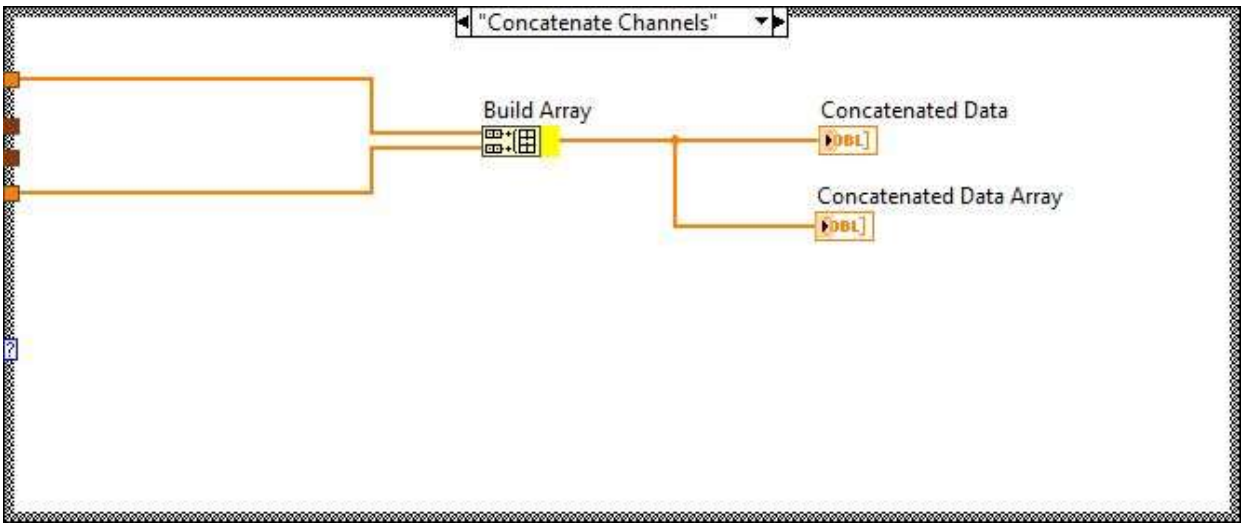

Figure 5, Solution program block diagram

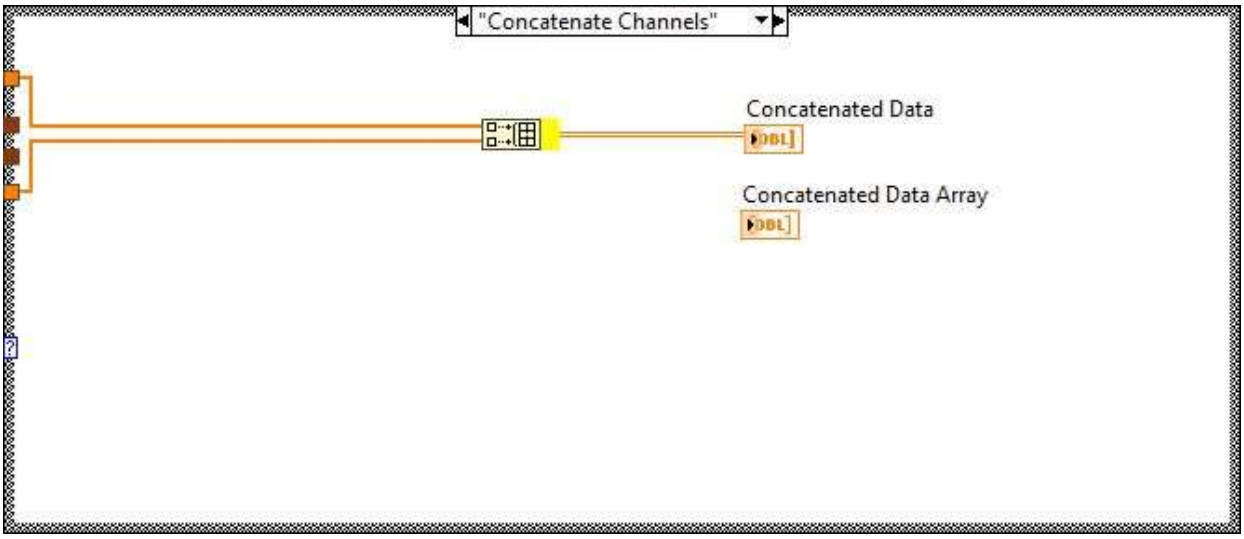

Figure 6, Student program block diagram

Implementation

Students used the automated grading in a Data Acquisition class at California Baptist University. The course covers the LabVIEW programming language and how engineers use LabVIEW for data acquisition. LabVIEW instruction follows National Instruments LabVIEW Core 1 [7] and Core 2 [8] course material. The class is required for Electrical and Computing Engineering majors, and some Mechanical Engineering Majors take the class as a technical elective. Students first used the grading program in the Fall semester of 2020 in a class of 18 Students. Based on conversations with the students during the semester, I modified the grading program to give them full credit if they scored at least a 95% on the assignment. This way, students were not spending extra time trying to fix minor errors since the goal of the automated grading was to have the students work through the tutorials. Students continued to use the grading program in 2021 (27 Students) and 2022 (36 Students). The computer-graded tutorial assignments were supplemented by other programming tasks with principles from the tutorial that the instructor or a teaching

assistant graded. Before implementing automatic grading, the tutorial assignments were not graded, and many students skipped them.
A statistical analysis of the students' test grades before and after implementing automated grading showed no significant effect. The change in education brought about by the response to COVID-19 also significantly changed the learning environment.

Student survey

After receiving IRB approval, students were surveyed regarding their opinions of the program at the end of each semester from 2020 to 2022. Students rated their experiences on a Likert scale to the following prompts:

- I found the program helpful
- I found the text description of the errors easy to understand
- I found the text description of the errors helpful
- I found the highlighted LabVIEW block diagrams easy to understand
- I found the highlighted LabVIEW block diagrams helpful
- I found the program easy to use
- The program improved my LabVIEW coding skills
- The grading reply from the program came in a timely manner.
- Based on my experiences with the program, I would rather use the grading program instead of having a TA grade my homework by hand.

Students were given a small amount of extra credit for their homework grades to incentivize them to participate in the survey. Students could complete an alternate task to get the points as well.
Figure 7 shows the results of the survey for each question by year. The percent positive number is the percentage of students who either Strongly Agreed or Somewhat Agreed with each statement from all students. Overall the students had a good experience with the program. They found that the text was not as easy to understand or helpful as the highlighting on the block diagram, which is understandable. Overall 96% of the students who responded preferred the automated grading program over having a TA grade their work.

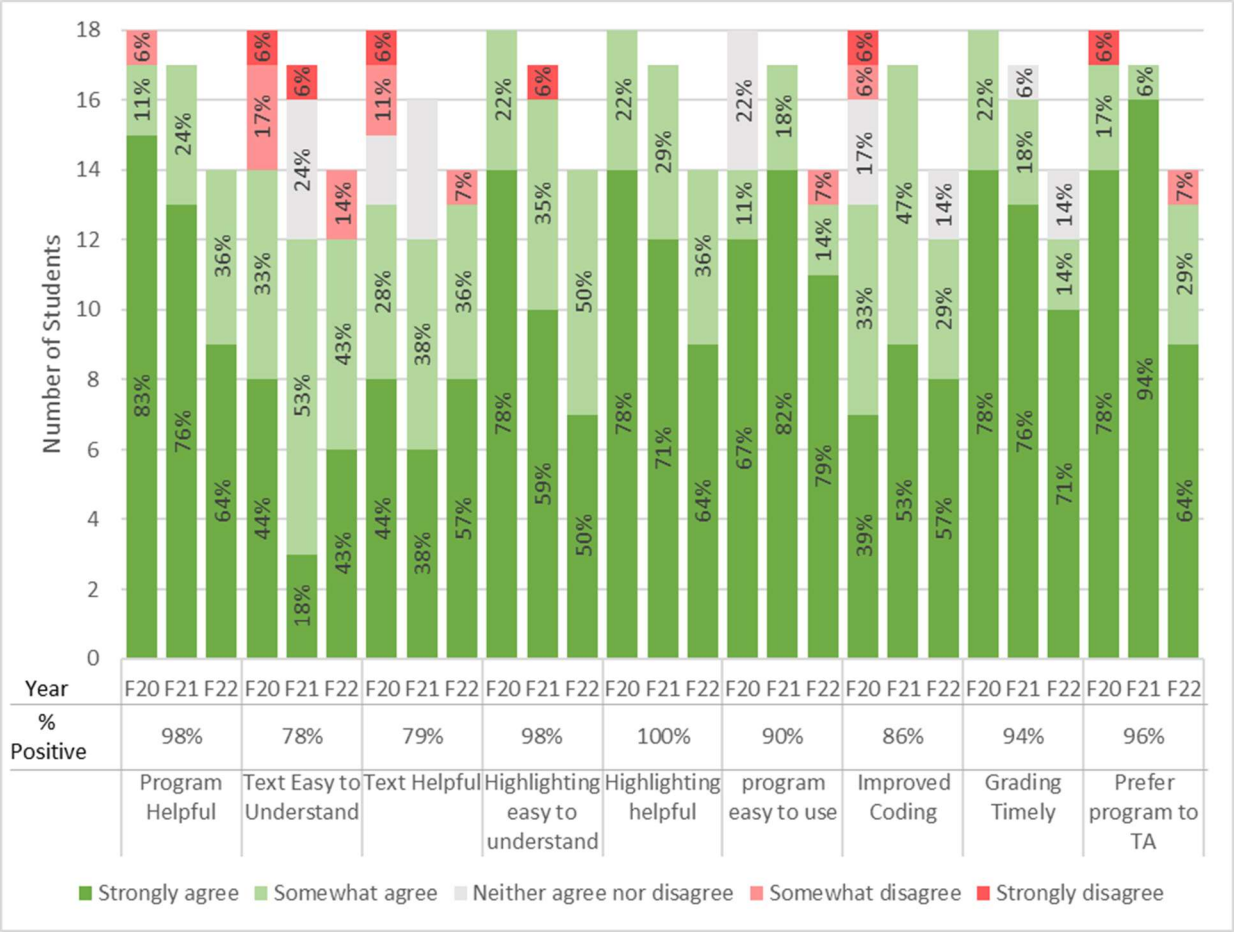

Figure 7, Student Survey results

Conclusion

An automated grading program has been developed for LabVIEW programs. The program receives the students' work by email and responds with their assignment score and a text and graphical description of the differences between the student's work and the instructor-provided solution. When surveyed, students had a positive experience with the program and preferred the program grading to having a TA grade their work. Future work will be to develop a web-based interface for the program and transfer the program to the cloud to improve reliability compared to running the program on a dedicated computer. A similar approach could be taken for Simulink files, though in informal discussions with people from Mathworks, they are developing automatic grading for Simulink.

References

[1] "Catalog - zyBooks," 16 7 2021. [Online]. Available: https://www.zybooks.com/catalog/.

[2] "MATLAB Grader," [Online]. Available: https://grader.mathworks.com/.

[3] P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä, "Review of Recent Systems for Automatic Assessment of Programming Assignments," in *Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, Koli Finland, 2010.

[4] J. C. Caiza and J. M. Del Alamo, "Programming Assignments Automatic Grading: Review of Tools and Implementations," in *7th International Technology, Education and Development Conference*, Valencia Spain, 2013.

[5] H. Aldriye, A. Alkhalaf, and M. Alkhalaf, "Automated Grading Systems for Programming Assignments: A Literature Review," *International Journal of Advanced Computer Science and Applications,* vol. 10, no. 3, pp. 215-222, 2019.

[6] "What is LabVIEW? Graphical Programming," National Instruments, [Online]. Available: https://www.ni.com/en-us/shop/labview.html.

[7] "LabVIEW Core 1 Overview," National Instruments, [Online]. Available: https://www.ni.com/en-us/shop/services/education-services/customer-education-courses/labview-core-1-course-overview.html.

[8] "LabVIEW Core 2 Course," National Instruments, [Online]. Available: https://www.ni.com/en-us/shop/services/products/labview-core-2-course.html.