# An Automated Grading System for Teaching MATLAB to Freshman Engineers

**James D. Bowen and Charles E. Price**
**University of North Carolina at Charlotte**

## Abstract

Systems for homework assignment, preliminary grading, homework submission, and final grading have been developed for a course unit that teaches MATLAB programming. During the MATLAB unit, which comprises approximately half of the semester, weekly homework assignments require all students to write and submit MATLAB scripts that test the programming concepts covered during the week's classroom lectures. Homeworks are assigned and submitted on the Internet using web-based forms. A MATLAB-based grading system developed for the course allows students to grade their own assignments before submission. This same system is used in an automated mode to grade the students' submitted assignments and to generate grading reports. Several measures have been taken to discourage cheating and to check for cheating in submitted scripts.

The automated grading system has proven to be a successful method for administering computer programming homework assignments. Students seem to quickly learn the systems for picking-up, grading, and submitting assignments, and have made heavy use of the grade-checking feature of the system. For instance, in the first ten weeks of the Fall 2002 semester, the system was used approximately 3000 times by 28 students and a teaching assistant as they worked on six MATLAB assignments. Script grading takes approximately ten minutes for a 100-student class. The system works particularly well in documenting the grading process, as the original submitted script, an annotated grading outcome, and a grade spreadsheet are all automatically saved securely during grading. The system has fostered student learning several ways; by allowing for more assignments, by allowing students to check their own work, and by allowing the instructor and the teaching assistants to focus their efforts on instruction rather than on grading and administering homework assignments.

## Introduction

Like many engineering schools, the University of North Carolina (UNC) at Charlotte has had a continuing debate regarding the freshman engineering curricula. One of the issues under discussion is the need for computer programming in the curricula. Like faculty elsewhere[5] nearly all faculty at UNC Charlotte believe that students need to learn early in their academic career "basic" computer skills such as word processing, spreadsheet analysis, and data presentation. Likewise, many faculty favor the approach taken by other schools[2] that offer instruction during the freshman year on computer software that can be used as part of traditional engineering problem solving. Finally, some other faculty at UNC Charlotte support a curricula that includes computer programming instruction.

Lately, the faculty in the Civil Engineering, in accordance with decisions made at other universities[3], have adopted a curricula for the second semester freshman engineering course that uses the numerical analysis software package MATLAB[9] to teach computer programming. With its many built-in functions and matrix analysis routines MATLAB can be a powerful engineering problem solver. In addition, MATLAB has many two-dimensional and three-dimensional plotting functions that can be useful in a variety of engineering courses. Finally, MATLAB also contains a programming language with all of the constructs found in other high-level languages. While discussion continues within the College on the relative merits of MATLAB and other quantitative analysis software packages such as MS Excel[10], MathCAD[7], or Mathematica[8], it was decided for the time being that MATLAB instruction would be taught to all Civil Engineering students during their second semester. In this article we report on a system that has been created to assist in teaching and grading of a course unit in MATLAB.

Because of heavy workload anticipated for instructors and teaching assistants involved with the MATLAB unit, it was decided that an automated assignment management system would be essential. The course was considered a challenge for several reasons. The MATLAB course unit was relatively short (approximately half of a two semester hour course) and the class size was large (approximately 100 students). Multiple assignments would be needed to give students repeated practice in writing, debugging, and running their programs. Because only a few of the student's have had previous programming experience, it was expected that many students would require lengthy, individualized instruction. The relatively short duration of the unit also meant that rapid grading of assignments was essential. It was decided therefore that automated systems be developed to facilitate every step in the assignment process.

Automated grading systems have previously been developed to assist with instruction in problem solving or computer programming. Green[6] developed an automated grading system as part of a junior-level electrical engineering course in signals and systems. In this case, automated grading was achieved using MATLAB's Webserver software, but students did not learn MATLAB as part of the course. Fithen[4] also developed a web-based system to grade computer modeling assignments. In this case students were required to learn MATLAB programming as part of the course. Here we describe an automated grading system that is incorporated directly into the MATLAB environment. Students are able to write, debug, run, and grade their MATLAB assignments from within the MATLAB environment. The result is a system that students quickly understand and adopt, is relatively easy to setup and administer by the instructor, and offers a wide range of potential computer programming assignments.

In the following section we describe the automated assignment management system that is now being used to teach computer programming to freshman Civil Engineers. To demonstrate the use of the system an example assignment is described. It will be seen that each assignment is comprised of several tasks that are performed by either the student or by the instructor. After explaining the assignment system and demonstrating its use, some observations are made on its use so far. The article concludes with some general conclusions on the advantages of automated assignment grading systems.

**Description of the Automated Grading System**

During the portion of the course devoted to MATLAB instruction, students are asked to complete a programming assignment roughly once a week. Each assignment requires the student to write a MATLAB script that utilizes the programming concepts and MATLAB functions presented in class that week. Each of these individual assignments can be seen as a process consisting of a number of tasks for the students, teaching assistants, and/or the instructor. A software system utilizing MATLAB and web based perl scripts has been created to assist students and instructors in completing each of these assignment tasks. An example assignment is provided here as a demonstration of the system and as a means for explaining its use. The tasks associated with a single assignment are as follows.

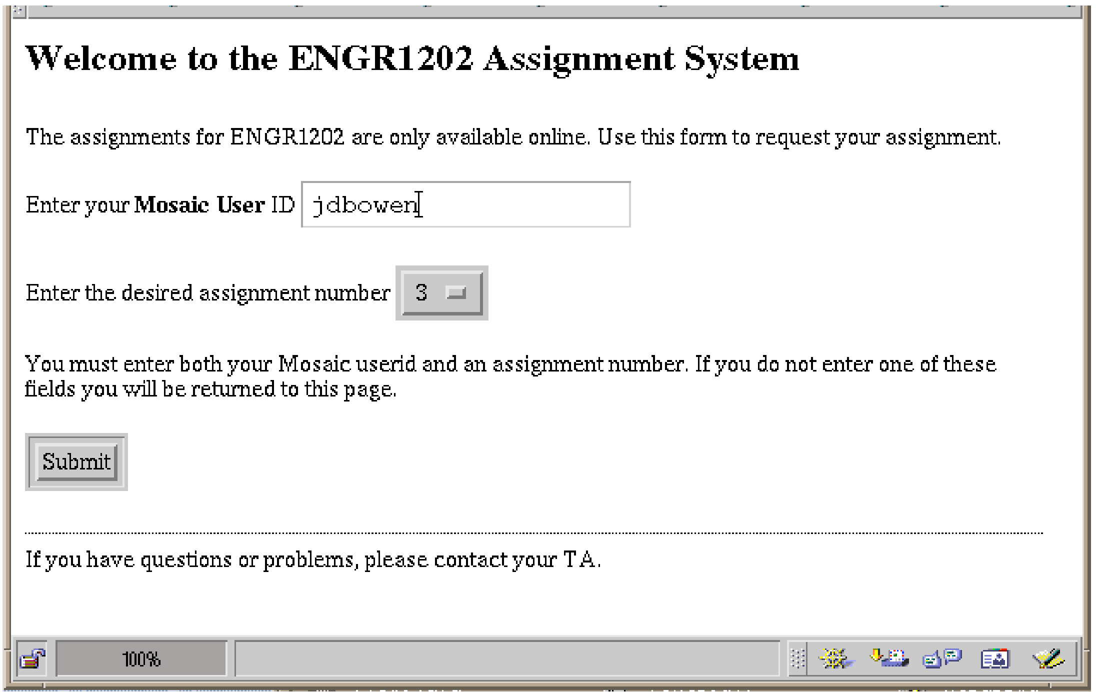
1. Student obtains his or her individual assignment off the class web site using an interactive form.
2. Student uses the MATLAB command line interface and the editor to write a MATLAB script (.m file).
3. Student debugs assignment and checks grade before submission.
4. Student submits the assignment script electronically.
5. Instructor uses automated grading system to grade all submitted scripts, to check for cheating, to print graded assignments, and to generate a grade report.
6. Student obtains graded assignment off class web site.

Each of these steps is described in more detail in the following sections.

*Step 1 - Student Obtains Assignment*

Assignments are provided to students using a web-based system. To discourage cheating, four distinct versions of each assignment are created. The overall programming objective is identical in each version, as are the skills needed to complete the assignment. Versions typically differ slightly in the order in which questions are asked, or in the data supplied, or in the MATLAB functions to be used. For instance, one version might ask for the minimum value of a vector while another version might ask for the mean value. One version might ask the student to generate a matrix of zeros with four rows and three columns, while another version would ask for a different number of rows and columns. Each version is designed to appear identical to other versions, while having completely different answers.

Versions are assigned to students using their unique user identification string (User ID) and the assignment number. Students are required to enter both these fields into the web-based form (Figure 1). To determine the version number, a perl script used by the form converts each character in the User ID to an integer ASCII code. These integers are then summed, divided by the assignment number, and then divided by four. The version number is then determined by adding one to the integer remainder of the division. Numerical testing demonstrated that the method divides the class into four nearly equal fractions regardless of the assignment number. The form then returns to the student their particular assignment with their User ID listed at the top. No indication is given to the student that their assignment is one of four possible versions.



**Welcome to the ENGR1202 Assignment System**

The assignments for ENGR1202 are only available online. Use this form to request your assignment.

Enter your **Mosaic User** ID jdbowen

Enter the desired assignment number 3

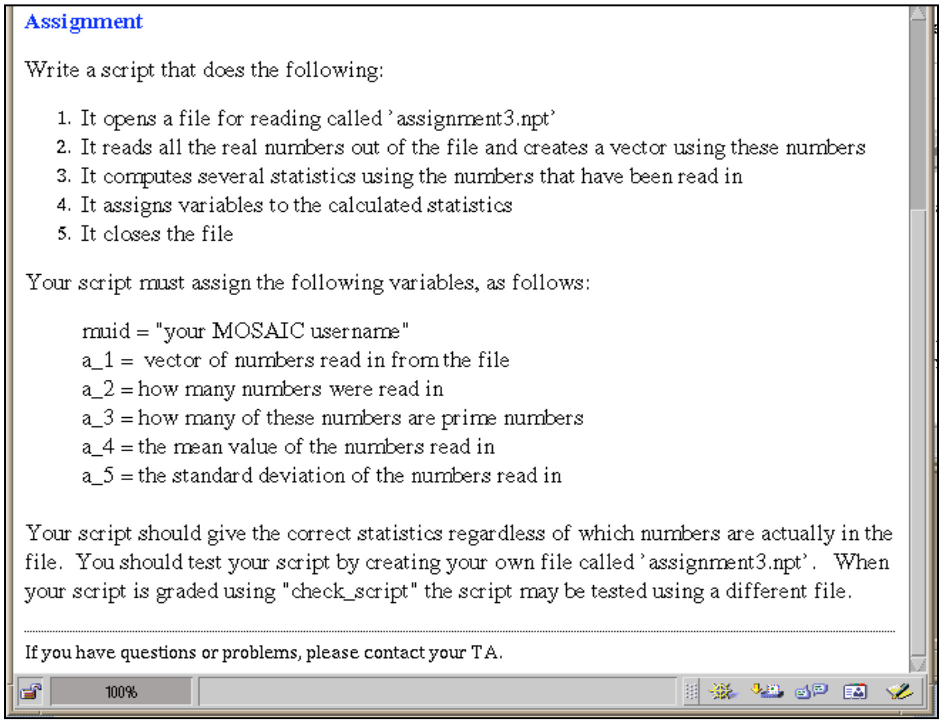
You must enter both your Mosaic userid and an assignment number. If you do not enter one of these fields you will be returned to this page.

**Figure 1.** Form used by students to obtain each assignment from class web site.

Each of the assignments is designed in such a way that the script created by the student is non-interactive, with the answers given by assigning values to designated variable names. Assignments generally ask students to assign values to four to ten variables, designated “a_1” through “a_10.” For example, the third assignment given to students in the Fall 2001 semester asked students to read numbers from a particular data file and then compute statistics using the data read in. In this assignment, students were required to assign the variable a_1 to the vector of numbers read in, and the variables a_2 through a_5 to four numerical statistics (Figure 2). In each assignment students are also required to assign to the string variable “muid” their User ID. This assignment requirement was created as one simple, albeit easily overcome way of detecting scripts that have been copied from another student.

*Step 2 - Student Writes MATLAB Script*

Each assignment requires the student to write a MATLAB script (a .m file), which is a text file including descriptive comments and MATLAB commands. Students generally work on the assignments using the College of Engineering’s local area network of PC’s running Windows NT or XP and Sun workstations running UNIX. The College maintains approximately seven computer labs with approximately 200 machines that are available to any one of the 1800 engineering students. MATLAB version 6 is available on both platforms and has an identical



**Assignment**

Write a script that does the following:

1. It opens a file for reading called 'assignment3.npt'
2. It reads all the real numbers out of the file and creates a vector using these numbers
3. It computes several statistics using the numbers that have been read in
4. It assigns variables to the calculated statistics
5. It closes the file

Your script must assign the following variables, as follows:

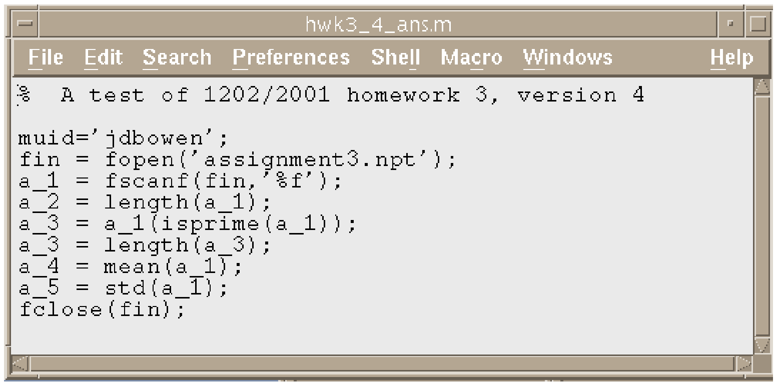
muid = "your MOSAIC username"
a_1 = vector of numbers read in from the file
a_2 = how many numbers were read in
a_3 = how many of these numbers are prime numbers
a_4 = the mean value of the numbers read in
a_5 = the standard deviation of the numbers read in

Your script should give the correct statistics regardless of which numbers are actually in the file. You should test your script by creating your own file called 'assignment3.npt'. When your script is graded using "check_script" the script may be tested using a different file.

If you have questions or problems, please contact your TA.

**Figure 2.** An example MATLAB assignment (assignment 3, version 4).

feature set and user interface. A student version of MATLAB is also available that is relatively cheap to buy and is functionally equivalent to the XP or UNIX versions available on campus. On both platforms the MATLAB environment includes a command line interpreter window and an editor window. Students are encouraged to use the command line window to test short portions of the script that they are writing using the editor. Since the scripts are created as text files, other text editors can also be used by students or instructors who have a particular favorite. Assignments are designed so that they can be completed using scripts that have ten to fifty lines of code (Figure 3).



```
%  A test of 1202/2001 homework 3, version 4

muid='jdbowen';
fin = fopen('assignment3.npt');
a_1 = fscanf(fin,'%f');
a_2 = length(a_1);
a_3 = a_1(isprime(a_1));
a_3 = length(a_3);
a_4 = mean(a_1);
a_5 = std(a_1);
fclose(fin);
```

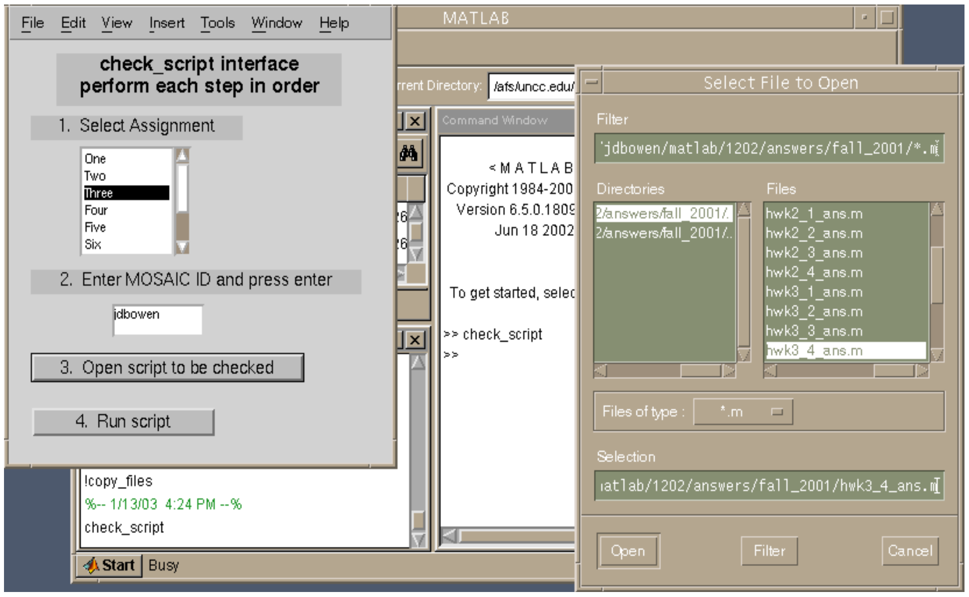
**Figure 3.** MATLAB script for assignment 3, version 4.

*Step 3 - Student Checks Assignment Grade Before Submission*

An essential feature of the system is that students should be able to test their scripts and check their grades before the scripts are submitted. This was accomplished by creating a MATLAB script called “check_script.m” that includes a graphical user interface (GUI) to specify the student, assignment, and script together with additional MATLAB scripts and functions for running the student’s script, defining a set of correct answers, comparing the results against the correct answers, and calculating the grade based upon the results of the comparison.

To accomplish the entire grading process, approximately twenty specialized scripts and functions were written. Collectively these functions and scripts contain approximately 2000 lines of MATLAB code. All of these scripts have been compiled into a functional, binary format (.p files) and then placed into a special directory that is included in each student’s search path when they start either the PC or UNIX versions of MATLAB. All students are given read only access to this directory. Students who purchase the student version of MATLAB are able to download the contents of this directory and place it on their local machine so that they can also use the grading system. The graphical user interface that is the heart of the system was created using a GUI builder feature of MATLAB. The GUI includes boxes for the entry of the User ID and assignment numbers, and a separate GUI for selecting the file to check (Figure 4). This GUI allows students to interactively find and select any MATLAB script in their filespace.

Once the student selects the script to be checked, the grading system then attempts to run the student’s scripts. Errors that cause the script to abort prematurely are trapped and a minimum grade is assigned. The grading system runs the student’s script from within a function to prevent accidental assignment of variables used by check_script. Scripts with no syntax errors



jdbowen

```
<MATLAB
Copyright 1984-200
Version 6.5.0.1809
Jun 18 2002

To get started, selec

>> check_script
>>
```

```
!copy_files
%-- 1/13/03  4:24 PM --%
check_script
```

Filter: 'jdbowen/matlab/1202/answers/fall_2001/*.m

Directories: 2/answers/fall_2001/. | 2/answers/fall_2001/..

Files: hwk2_1_ans.m, hwk2_2_ans.m, hwk2_3_ans.m, hwk2_4_ans.m, hwk3_1_ans.m, hwk3_2_ans.m, hwk3_3_ans.m, hwk3_4_ans.m

Files of type: *.m

Selection: atlab/1202/answers/fall_2001/hwk3_4_ans.m

**Figure 4.** GUIs used as part of "check_script" for student grading of MATLAB assignments

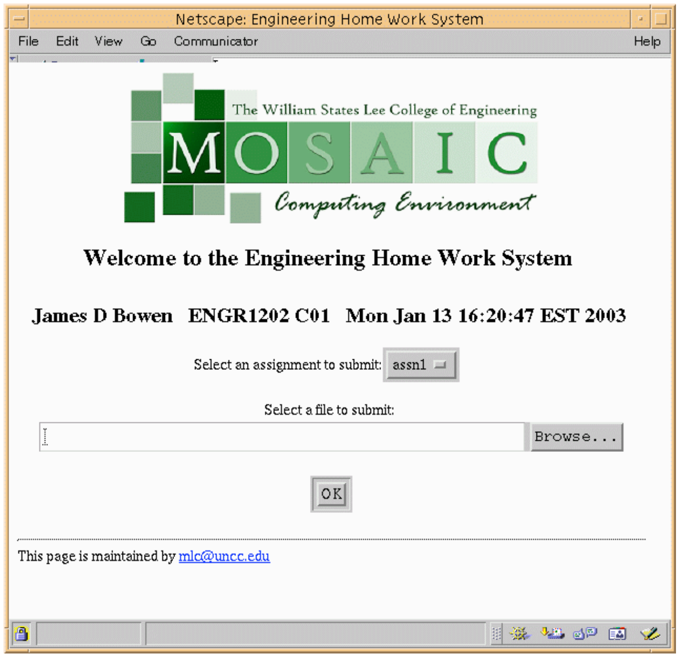
that lack an assigned User ID are given a slightly higher, but still unacceptable grade. Variables assigned by working scripts that contain a properly assigned User ID variable are compared to the set of correct answers for that assignment and version. The grading scheme is flexible, and is assigned by the instructor in one the MATLAB functions, as are the set of correct answers. For the example assignment shown, scripts with syntax errors receive a grade of 30 points; working scripts with no User ID receive a grade 40 points. Working scripts with defined User IDs receive grades from 50 to 100 points (10 points per correct answer). Answer checking is also done from within a function call so as to hide the correct answers from the students. Results of the checking are provided to the students immediately using the MATLAB command line interface.

*Step 4 - Student Submits Assignment Electronically*

Once the student is satisfied with his or her grade as determined by check_script, they then submit their assignment electronically. The homework submission system uses web based forms and perl scripts to allow students to upload their homework files into a secure area of the college's shared file system. Using a web-based system makes it easy to allow students to upload their files from any location without granting them read access to the submissions of other students or allowing them to alter their own files after submission. To initialize the home work submission system, the instructor or a TA defines the number of assignments to be submitted online, assigns a range of dates during which each assignment may be submitted, sets the date and time when students can begin to pickup each graded assignment, and specifies whether or not late submissions will be accepted. The software automatically creates separate directories for the on time, late and graded submissions for each assignment and sets the access permissions so that the instructor and TAs have read access to the submitted files, and write access to the graded directory. When students submit an assignment, they must select the appropriate course and section numbers and input their User ID and password for authentication. The system then verifies the student's identity using the same ID and password authentication system that controls access to the college's shared computer network. Once a student has been authenticated they must select the desired assignment number, and input the name of the file to be uploaded (Figure 5). The selected file is then saved to the appropriate directory for that class, section and assignment, using a unique file name created by concatentating the assignment number with the student's User ID and a submission number (e.g. assn3jdbowen2.m). The file name extension (e.g. .m for MATLAB scripts) is retained in order to facilitate identification of the file type and support easy access from the associated application. With this system a consistent naming scheme is guaranteed and each student is allowed to make multiple submissions that are separately identified and tracked.

*Step 5 - Instructor Grades Assignment Using Automated System*

Once the deadline for a particular assignment has passed, all subsequent submissions are placed in a "late" directory so that they can be easily identified during grading. At this point, every student in the class is graded on the particular assignment using a MATLAB script (grade_all_students.m) written for this purpose. Once the instructor chooses the assignment number to be graded, the script then consults a student list. The grading script searches for the last submission of each student on the class list. Each of these scripts is then graded using the same set of functions used by the students to check their grade. Once the grades are assigned,


Netscape: Engineering Home Work System

File Edit View Go Communicator Help

The William States Lee College of Engineering MOSAIC Computing Environment

Welcome to the Engineering Home Work System

James D Bowen ENGR1202 C01 Mon Jan 13 16:20:47 EST 2003

Select an assignment to submit: assn1

Select a file to submit:

Browse...

OK

This page is maintained by mlc@uncc.edu

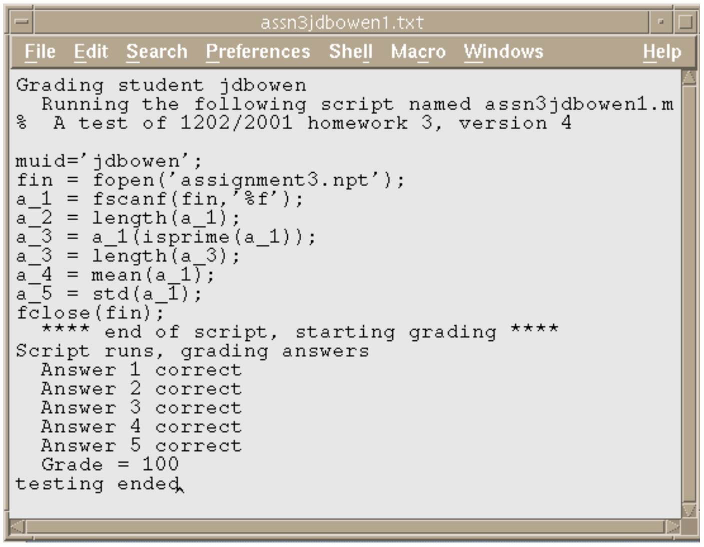
**Figure 5.** Web based form used by students to electronically submit homeworks

the grading script then checks for any evidence of cheating. The results of each step in the grading process are printed to a text file and saved to a "graded" subdirectory for that class, section, and assignment. Into this text file is first put the name of the file that is being graded. The grading script next prints a copy of the script to be graded to the text file. Finally, the results of the grading are copied to the file (Figure 6). A filename scheme like that employed in homework submission is employed to create unique filenames for each assignment and student.

A number of separate checks are performed to detect cheating. As mentioned earlier, each assignment is required to specify the student's User ID. During grading the student's actual User ID is compared to that assigned in the script being graded. If the assigned ID is not the student's but was in fact another student's ID, then both students are considered to be cheating. A second check for cheating is performed on scripts that have the correct User ID specified. The subset of these scripts that received grades less than 100% are graded again using the other three assignment version numbers. If another version of the assignment produces a higher grade, then the student is assumed to have cheated. For these cases where cheating is detected, the graded assignment returned to the student states that cheating has been detected and a grade of zero has been assigned. In the text file returned to the student, the student is asked to consult the instructor should they have any questions or concerns about the grading.

*Step 6 - Student Obtains Graded Assignment*

Graded assignments are returned to students in two ways. First, for the first few assignments, where the scripts are relatively short, all of the text files created by the automated grading system (see Figure 6 for example) are printed and handed back to the students in class.



```
Grading student jdbowen
  Running the following script named assn3jdbowen1.m
%  A test of 1202/2001 homework 3, version 4

muid='jdbowen';
fin = fopen('assignment3.npt');
a_1 = fscanf(fin,'%f');
a_2 = length(a_1);
a_3 = a_1(isprime(a_1));
a_3 = length(a_3);
a_4 = mean(a_1);
a_5 = std(a_1);
fclose(fin);
  **** end of script, starting grading ****
Script runs, grading answers
  Answer 1 correct
  Answer 2 correct
  Answer 3 correct
  Answer 4 correct
  Answer 5 correct
  Grade = 100
testing ended
```

**Figure 6.** An example graded assignment returned to student.

Students are also able to pick up their graded assignments using a secure web-based system. Students login to the system by supplying their User ID and system password. Once admitted, students enter the course number, the section, and the assignment number. The graded assignment corresponding to these choices is then provided to the student.

## Results of System Use

As of December 2002 the automated assignment management system had been used for three semesters worth of freshman engineering classes. In its first use in the Fall 2001 semester, the MATLAB unit was taught to more than 100 students during a single four-week time period. In this initial use four assignments were created that were quite similar in concept and execution. Over a four-week time period at the beginning of the semester nearly 500 MATLAB scripts were created, checked, submitted, and graded. In this initial use design of suitable assignments was found to be one of the more challenging parts of course development. As initially designed the system required that all assignments be non-interactive, and have exactly five variables to be graded. Even with this restriction however, it was found that assignments requiring tasks such as file input and output and two-dimensional graphics could be accommodated.

Students seemed to quickly adapt to the system, and appreciated being able to check their grades before submitting the assignment. Use of the checking script was monitored during the last two weeks of the course. Every time check_script was run a record was created giving the date and time, type of computer, and assignment being checked. Over this two-week time period students checked their scripts over 1600 times, which corresponds to approximately eight uses per student, per assignment. Not surprisingly, usage peaked in the afternoon and early evening on the day before the assignment was due. Students seemed to prefer using the Windows based machines, as approximately 70% of the usage came from the PC machines, even though there are nearly identical numbers of PCs and UNIX machines in the College's network.

The systems used to deter and detect cheating were also found to be effective. Some students had problems initially in obtaining the proper assignment version. Several of the international students confused their User ID and their student ID, a nine-digit integer that is usually also their social security number. This problem was eliminated by modifying the perl script so that it would not accept numbers for the User ID. Having multiple versions of every assignment was found to complicate the work of the teaching assistants. In assisting students whose scripts scored less than 100 they had to be aware that students sitting next to each other in the computer lab might have different assignments. Two students were caught cheating on one assignment in the first semester. One student had copied the assignment from another. Both students were assigned the same version of the assignment, but the student who copied the assignment failed to change the script to have his own User ID. No cheaters were found using the system that checked if a higher grade could be obtained for the assignment if graded using a different version number.

In the Fall 2002 semester the system was improved to allow for more flexibility in problem assignment and grading schemes. Currently, MATLAB instruction is offered in portions of the entire two semester-hour course. Six MATLAB homeworks are assigned over this time period. Several of the later assignments require the students to perform with their MATLAB scripts structural and cost analyses of candidate truss designs used as part of a balsa wood bridge design competition[1]. This has proven to be a challenging use of the system, as there is no single set of correct answers for a given assignment, since each student is free to design their own truss. A significant amount of development time was required to allow the system to handle a variable set of correct answers. Students continue to heavily use the script grading system. For instance, in the first ten weeks of the Fall 2002 semester, the system was used approximately 3000 times by 28 students and a teaching assistant as they worked on six MATLAB assignments. This corresponds to roughly sixteen uses of the system per student per assignment. The higher number of student uses in the latest semester is probably due to several factors including the more challenging nature of the truss analysis assignments, the longer time span for MATLAB instruction, and the overall improvement of the grading system made over its first year of use.

The homework submission and automated grading systems have been found to be terrific time savers. Homework grading is now quick and painless. When the assignments were originally designed there was a sizable time commitment (approximately 30 hours) to write the assignments, to define the correct answers, to develop the grading schemes for each assignment, and to test and debug the system. Having accomplished this, however, homework grading for a 100-student class can now be done in approximately ten minutes. In addition, time-consuming file management chores are now nearly completely eliminated. The grade reports automatically generated for the students give them all the information that they require, and the system also produces a grade report that can easily be imported into a spreadsheet. Having all the graded assignments saved as text files with "read only" file permissions for the students also makes these files useful as an archive of student work. On a couple of occasions questions have arisen more than a month later about the grade for a particular assignment. In every case the graded scripts were available to answer these questions.

## Conclusions

There are few tasks more challenging than teaching programming to a large class of freshman engineers. Thankfully, many of the administrative chores of such a class can now be accomplished using a system like that described here. This leaves more of the instructor's time available for teaching. Students also benefit, as they are able to exercise more control over their course outcome. Assignments are precisely specified, and their work is graded immediately, allowing them to check, revise, and recheck their scripts until they are satisfied with their grade. The overall result is that student learning is improved, and both the instructor and the student are more satisfied with the course outcome.

### Bibliography

1. Bowen, J.D. 2003. Using a Hands-On, Project-Based Approach to Introduce Civil Engineering to Freshman. *Proceedings of the 2003 American Society for Engineering Education*. Washington, D.C.: American Society for Engineering Education
2. Clough, D.E., S.C. Chapra, and G.S Huvard. 2001. A Change in Approach to Engineering Computing for Freshman – Similar Directions at Three Dissimilar Institutions. *Proceedings of the 2001 American Society for Engineering Education*. Washington, D.C.: American Society for Engineering Education
3. Devens, P.E. 2000. MATLAB & Freshman Engineering. *Proceedings of the 2000 American Society for Engineering Education*. Washington, D.C.: American Society for Engineering Education
4. Fithen, B. 2001. Building a Web Based System for Automated Grading of Computer Modeling Assignments. *Proceedings of the 2001 American Society for Engineering Education*. Washington, D.C.: American Society for Engineering Education.
5. Gottfried, B.S. 1996. Should Computer Programming Be Taught to All First-Year Engineering Students? *Proceedings of the 1996 American Society for Engineering Education*. Washington, D.C.: American Society for Engineering Education
6. Green, R.A. Mastery Learning with the MATLAB Webserver. 2000. *Proceedings of the 2000 American Society for Engineering Education*. Washington, D.C.: American Society for Engineering Education
7. "MathCAD" is a trademark of MathCAD, Inc.
8. "Mathematica" is a trademark of Wolfram Research, Inc.
9. MATLAB", MATrix LABoratory is a trademark of The Mathworks, Inc..
10. "MS Excel", Microsoft Excel is a trademark of Microsoft Corporation

JAMES D. BOWEN
James D. Bowen is an Assistant Professor in the Civil Engineering Department at UNC Charlotte. He received his Ph.D. degree from the Massachusetts Institute of Technology. Dr. Bowen teaches MATLAB programming, hydraulics, aquatic chemistry, and water quality modeling. His research interests include water quality and eutrophication modeling, model uncertainty analysis, and the microscale fluid motions around phytoplankton cells.

CHARLES E. PRICE
Charles E. Price is the Assistant Dean for Engineering Computing in the College of Engineering at the University of North Carolina at Charlotte. He has a Bachelor of Arts degree from Northwestern University and M.S. and Ph.D. degrees from Indiana University specializing in Theoretical Nuclear Physics. He is a member of ASEE, the American Physical Society, and Sigma Xi.