J. Appl. Math. & Informatics Vol. **38***(2020), No. 3 - 4, pp. 255 - 260*
http://dx.doi.org/10.14317/jami.2020.255

DEBUGGING TOOL FOR MOBILE NUMERICAL CODE LEARNING SYSTEM†

SANG-BAE KIM

Abstract. Today most students have a smartphone. Students learning numerical analysis can practice coding using their smartphone in the classroom without going to the computer lab. The tool NAPHP[1] makes it easy to practice coding using PHP in an internet web browser without installing a coding app on your smartphone. This paper introduces a debugging tool for the NAPHP.

AMS Mathematics Subject Classification : 65N15.
Key words and phrases : PHP, programming tool, Numercal Analysis.

1. Introduction

PHP is a kind of programming language. It was first unveiled by Rasmus Ludolf [2] in 1995 and is responsible for development and management by The PHP Group[3]. It is one of the typical server-side scripting language, and it is the basis of many web systems around the world. Similar languages include ASP and JSP. Using C-like syntax, it is easy and fast to create a small web page, so there are many users and applications of it.

Students in the numerical analysis course deal with very simple forms of algorithm to practice numerical calculations in textbooks. To practice coding, students had to move from the classroom to the PC room. However, with the recent spread of smart phones, it has become possible to code numerical analysis using smart phones in the classroom.

Although you can install a programming tool on your smartphone, you may practice your programming skills using PHP in internet web browser connected to a Linux server. The program tool NAPHP was introduced in the paper [1]. NAPHP uses a PHP internet server to provide examples to students and a convenient way to assign or collect homework to professors, but NAPHP lacked

Received March 31, 2020. Revised April 29, 2020. Accepted May 1, 2020.
†This work was supported by Hannam University Research Fund 2019.

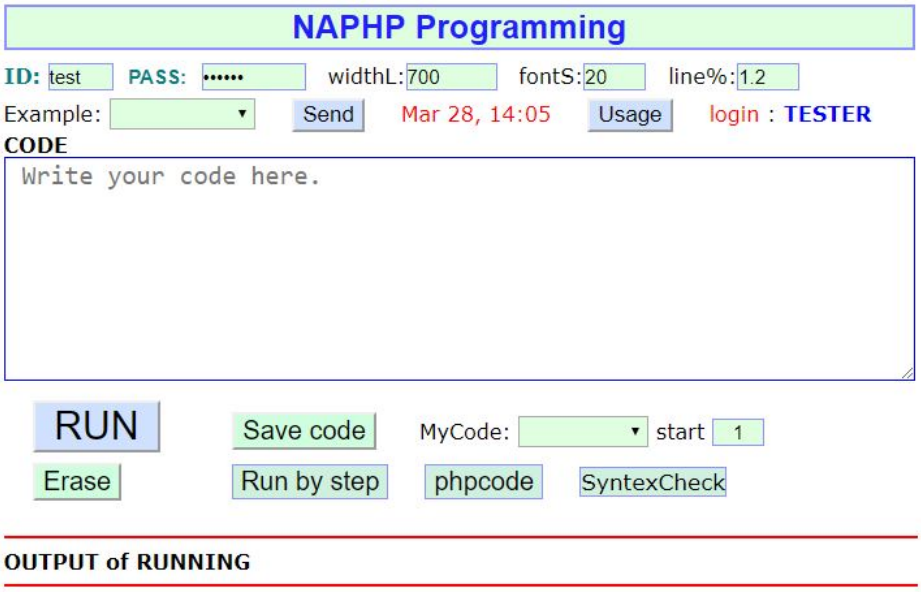

FIGURE 1. The first default screen of NAPHPv2.0

debugging tools. This paper introduces useful debugging tools for students using NAPHP.

2. PHP Numerical Analysis Program Tool (NAPHP)

When accessing the NAPHP homepage in a Linux server [4], the default screen of NAPHP version 2.0 appears as in Figture 1. Anyone can log in to the system with the test ID '`test`' and password '`000000`'(six zeros). If you enter a set of code in the box under `CODE` and click the `[RUN]` button, NAPHP will output in the same way as version 1.0 of the paper [1].

New features of version 2.0 are as follows. The first is that students can adjust the font size and line height for their own smartphone. Second, students can save their code using the `[Save Code]` button. They can use the selection box next to `'MyCode:'` to call the saved code and reuse it. Third, using the `[php code]` button, students can generate regular PHP code translated from NAPHP code. Fourth, students can check the code for syntex errors using the `[Syntex Check]` button. This syntex checking function is implimented by another tool in internet site[5] which is connected with the `[SyntexCheck]` button. Fifth, students can use the `[Run by Step]` button to run the code line by line for finding semantics error. Finally, the professor can run the code of the students who submitted it for homework.

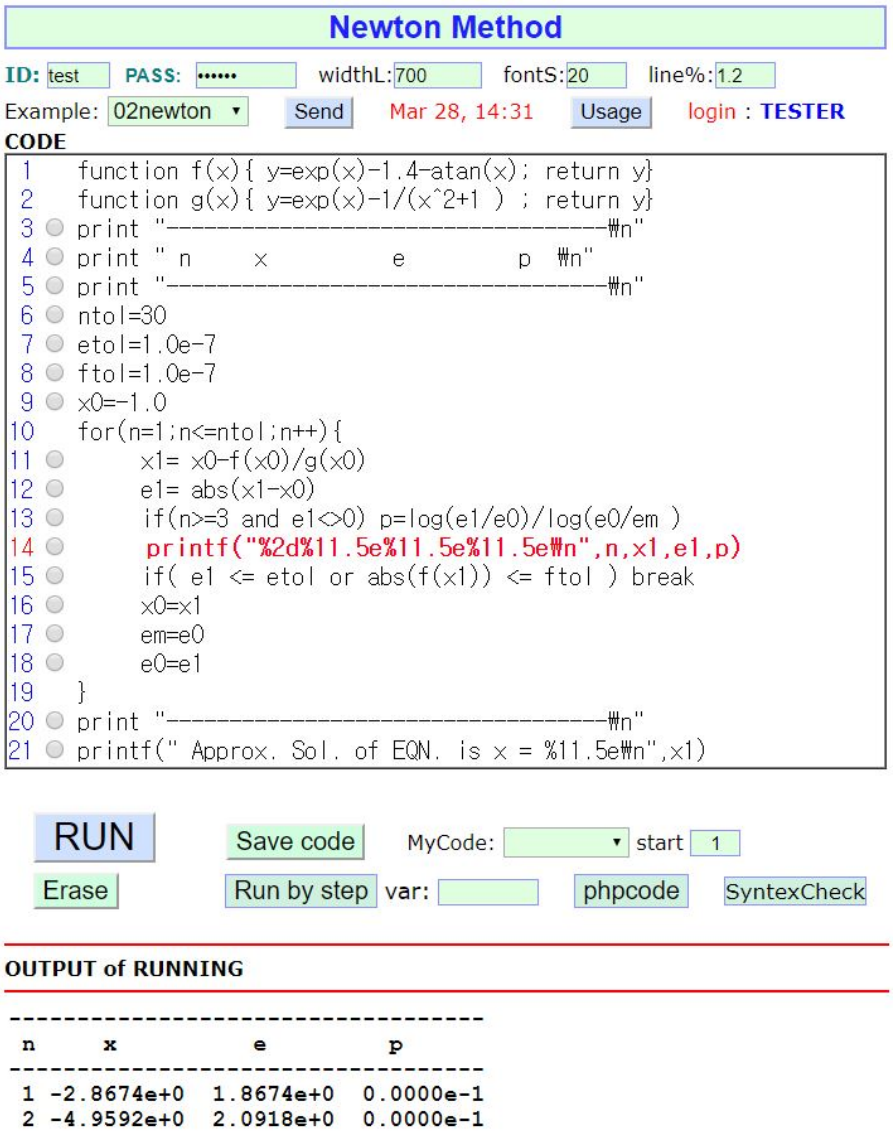

FIGURE 2. Step by Step Debugging Tool of NAPHP v2.0

Storing the code that students create is necessary for the case that there may be grammar error in the code. Students should return to the past state of their code and reconsider where they made grammar mistakes. In internet web browser, they may not be able to go to the past page for some reasons, so they

Newton Method

ID: test PASS: •••••• widthL:700 fontS:20 line%:1.2

Example: 02newton Send Mar 28, 14:58 Usage login : TESTER

CODE

```
function f(x){ y=exp(x)-1.4-atan(x); return y}
function g(x){ y=exp(x)-1/(x^2+1 ) ; return y}
print "-----------------------------------\n"
print " n     x          e          p  \n"
print "-----------------------------------\n"
ntol=30
etol=1.0e-7
ftol=1.0e-7
x0=-1.0
for(n=1;n<=ntol;n++){
     x1= x0-f(x0)/g(x0)
     e1= abs(x1-x0)
     if(n>=3 and e1<>0) p=log(e1/e0)/log(e0/em )
     printf("%2d%11.5e%11.5e%11.5e\n",n,x1,e1,p)
     if( e1 <= etol or abs(f(x1)) <= ftol ) break
     x0=x1
     em=e0
     e0=e1
}
print "-----------------------------------\n"
printf(" Approx. Sol. of EQN. is x = %11.5e\n",x1)
```

RUN Save code MyCode: start 1

Erase Run by step phpcode SyntexCheck

OUTPUT of RUNNING

```
function f($x){ $y=exp($x)-1.4-atan($x); return $y;}
function g($x){ $y=exp($x)-1/(pow($x,2)+1 ) ; return $y;}
print "-----------------------------------\n"; #3
print " n     x          e          p  \n"; #4
print "-----------------------------------\n"; #5
$ntol=30; #6
$etol=1.0e-7; #7
$ftol=1.0e-7; #8
$x0=-1.0; #9
for($n=1;$n<=$ntol;$n++){
     $x1= $x0-f($x0)/g($x0); #11
     $e1= abs($x1-$x0); #12
     if($n>=3 && $e1!=0) $p=log($e1/$e0)/log($e0/$em ); #13
     printf("%2d%11.5e%11.5e%11.5e\n",$n,$x1,$e1,$p); #14
     if( $e1 <= $etol or abs(f($x1)) <= $ftol ) break; #15
     $x0=$x1; #16
     $em=$e0; #17
     $e0=$e1; #18
}
print "-----------------------------------\n"; #20
printf(" Approx. Sol. of EQN. is x = %11.5e\n",$x1); #21
```

FIGURE 3. Regular PHP code from NAPHP

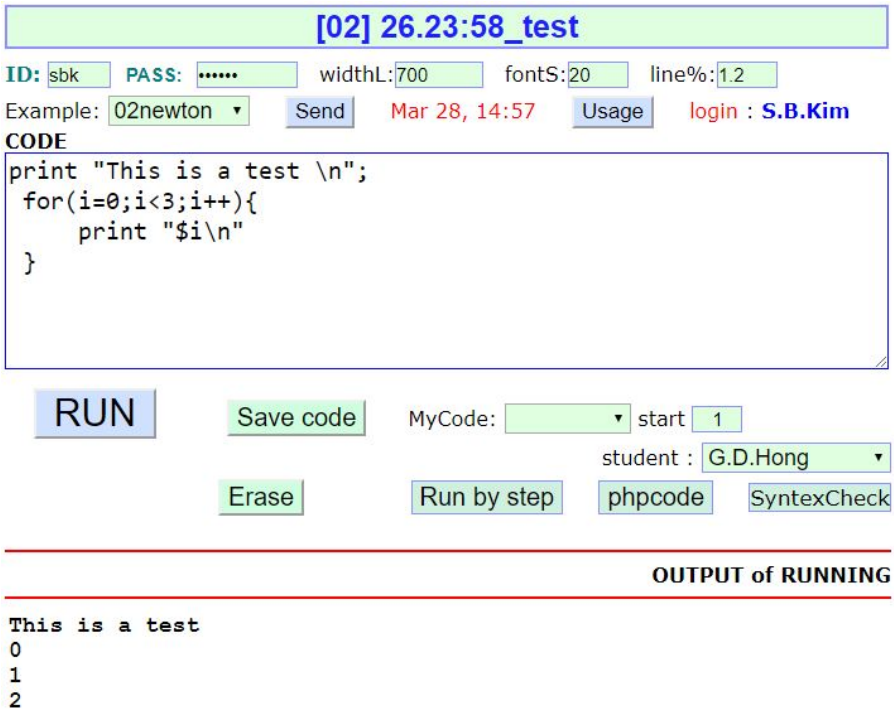

FIGURE 4. Implementation of Students' Homeworks

must save their code securely for the emergency. In addition, if the code was created for students' homework and later submitted on homework paper, there could be a problem with confidentiality. So students should save their code at the spot.

Sometimes, it is not easy for students to find grammar(syntex) errors in their code. First, students can convert NAPHP code to normal PHP code with the `[phpcode]` button. An example of the result is shown at the bottom of Figure 3. With the `[Syntex Check]` button, students can access the site [5] where they can look for grammar errors of their code. The moment the button `[Syntex Check]` is pressed, the PHP code is automatically copied to the buffer memory and you are directed to the site. Just paste it in the input box there and click `[Analyze]` button. Then you can find where the grammar errors are.

The basic of debugging for semantics error is step-by-step execution. What is needed for debugging is to check the value of the variable in question while the students step through the code. If you press the button `[Run by step]`, a radio button is created for each row of the code, as shown in Figture 2. It is designed so that the execution of code proceeds to the point where the button is pressed. The place where it is currently stopped is changed to red and bold

face. A text input box next to '`ver :`' is provided so that a specific variable for intermediate output could be designated.

Finally, the professor can check and execute the homework code which was created and saved by the students. If the professor log in as the designated superuser, the list of students is displayed in the selection box next to '`student:`' in the Figure 4, where the professor can choose a name of student. Students should submit a separate paper report in advance with a file name for a specific homework assignment. By selecting the file name from the selection box next to '`My Code: `', professor can grade the homework by running the student's code as a super user.

3. Conclusion

Our goal is for students to practice numerical analysis coding on internet web browser using their smartphones. NAPHP provides easier grammar form than the original PHP under the point of view of entering code. In NAPHP version 2.0, students can save their code and load it later. If there is a grammar error in the code, no results are produced. Then students can check the PHP syntax at the grammar check tool on the internet site [5]. On the other hand, for finding semantic errors, the code can be run step by step.

References

1. Sang-Bae Kim, *Mobile Learning System for Numerical Analysis by using PHP*, J. Appl. Math. & Informatics **37** (2019), 157-162.
2. Rasmus Lerdorf, accessed Nov 29, 2018, https://en.wikipedia.org/wiki/Rasmus_Lerdorf.
3. History of PHP, accessed Nov 29, 2018, http://php.net/manual/en/history.php.php.
4. NAPHP, accessed Nov 29, 2018, http://sbk.hnu.kr/NA/test.php.
5. PHP Code Checker, accessed Mar 3, 2020, https://phpcodechecker.com/.

Sang-Bae Kim received a Ph.D. from Purdue University, USA, in 1993. He is a professor of Department of Mathematics, Hannam University, Korea. His research interests are numerical linear algebra and scientific computing.

Department of Mathematics, Hannam University, Daejeon 34430, Korea.
e-mail: sbk@hnu.kr