A Course Management System Capable of Handling the Requisite Mathematics and Graphing of Engineering and Technology Problems: LON-CAPA

Prof. Gene L. Harding, Purdue Polytechnic Institute

GENE L. HARDING is an associate professor of Electrical and Computer Engineering Technology (ECET) at Purdue University, where he has taught since 2003. He has three years of industrial experience with Agilent Technologies, 28 years of combined active and reserve service in the United States Air Force, holds an MSEE from Rose-Hulman Institute of Technology, and is a licensed professional engineer. He has coded 1000+ LON-CAPA problems used in over a dozen ECET courses.

Dr. Stuart P. Raeburn, Michigan State University

Dr. Raeburn has been involved with instructional technology at Michigan State University for the past two decades. Currently he is affiliated with the Department of Physics & Astronomy as lead developer and release manager for the LON-CAPA Content Management and Assessment system.

A Course Management System Capable of Handling the Requisite Mathematics and Graphing of Engineering and Technology Problems: LON-CAPA

Abstract

This is the second in a series of papers to inform engineering and engineering technology (E/ET) educators about a powerful course management system (CMS) called LON-CAPA. LON-CAPA is an acronym for Learning Online Network with Computer Assisted Personalized Approach. Although it has been around for over two decades, it has been used primarily in science and mathematics disciplines, with virtually no penetration into E/ET fields. The purpose of this series of papers is to get the word out to interested educators in the E/ET arena.

LON-CAPA can do the relatively mundane tasks needed to manage a course, but the features that make it so unique and so powerful are the ones that allow content creation with

- an extensive library of mathematics functions for calculations,
- appropriate rendering of the mathematics notation,
- dynamically-generated graphs,
- interactive capabilities, and
- the ability to manage multi-part problems with linked calculations.

This paper begins with a brief synopsis of topics covered in the first paper, then continues the discussion with a detailed description of how to set up Numerical Response problems, including three different ways to set up the grading tolerances:

- within a specified percentage,
- with a set number of decimal places, or
- with a certain number of significant figures.

Next, it discusses two different techniques for setting up dynamically-generated graphs using gnuplot, as well as how to construct multi-part problems with calculations that carry through and correlate directly with the generated plots. Then it shows a pair of examples that mix response types. Finally, the paper concludes with a section describing options for implementing LON-CAPA at a new institution.

Introduction

Although LON-CAPA is used fairly widely in the science fields, and by some in mathematics, it is not well known in engineering and engineering technology circles. This is the second in a series of papers whose purpose is to get the word out to those communities about LON-CAPA's benefits and capabilities.

The first paper started with an overview of LON-CAPA's history, beginning with its inception in 1992 as CAPA, a pilot project in a physics class at Michigan State University. Today it is used in

Canada, Germany, Hong Kong, Indonesia, Israel, Kazakhstan, Netherlands, New Zealand, Oman, South Korea, Turkey, and the USA (with the majority of users in the US). Since its versions are backward compatible, and content is never really deleted, its 17 years of operation probably make it the longest-running learning management system around.

The previous paper described Radio Button Response, Option Response (i.e., multiple choice with more than one correct answer), and Numerical Response problem types, as well as how to combine them into multi-part problems with calculations that carried forward through some or all of the constituent problems. It also touched on available templates and how to use capabilities of the Maxima algebra system, R statistics package, gnuplot plot engine, and Perl scripting [1].

Other topics covered by the first paper include pedagogical benefits, such as fail without penalty while learning, immediate feedback, self-pacing, and student control; ability to create custom content; open architecture that allows easy problem sharing; time savings afforded by automatic grading; enabling distance learning; and the different learning curves associated with some of the problem types [1].

This paper picks up where the first one left off. It begins by digging deeper into construction and use of Numerical Response problems; then describes in detail two different approaches to creating gnuplot-generated graphs, one using functions and the other using datasets; next, it discusses mixed-type multi-part problems that illustrate some of the power LON-CAPA provides. Each of these sections includes illustrative examples. Finally, for those interested in pursuing it, the last section describes how to set up and manage LON-CAPA at a new institution.

Numerical Response Problems

Numerical Response problems [1] take answers as numbers, which can be formatted in various ways and may include units, as appropriate. The tolerance for an answer can be specified as a specific amount, such as ±0.05 V, or as a percentage, or as a certain number of significant figures. Displayed numbers, such as "given" parameters for a problem, can also be formatted as desired, such as with a specific number of decimal places, or in scientific notation. Figures can be added to a problem, such as a circuit schematic or free-body diagram. Moreover, dynamically generated plots can be included, using the same randomly-generated numbers that define the Numerical Response problem. (How to generate these plots is covered in the next two sections.)

The structure of a common Numerical Response problem includes a section of Perl script to define parameters and perform calculations, and a section of extensible markup language (XML) to define how the problem is displayed to the student. An example of a basic Numerical Response problem, as displayed to the student, is shown in Figure 1. The first two lines are the

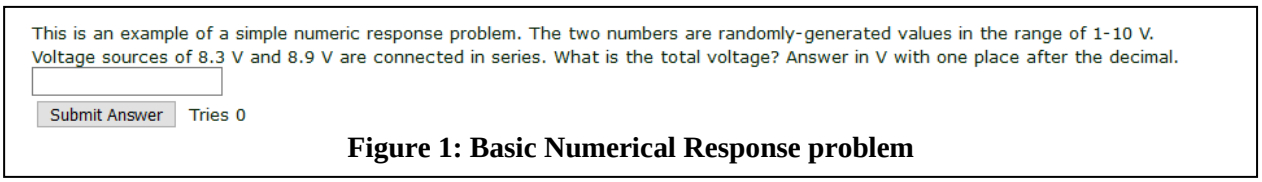


Figure 1: Basic Numerical Response problem

problem statement. The two numbers in the second line are randomly generated. The box below the text is where the student must enter the answer. If the units, V in this case, are left off, LON-CAPA will prompt the student to enter units. The correct answer for this randomization is 17.2 V. LON-CAPA accepts results with or without spaces and recognizes metric prefixes, so 17.2V, 17200mV, and 0.0172 kV would all be accepted.

The code for this problem is shown in Figure 2. Lines 4-8 contain the Perl script, which defines

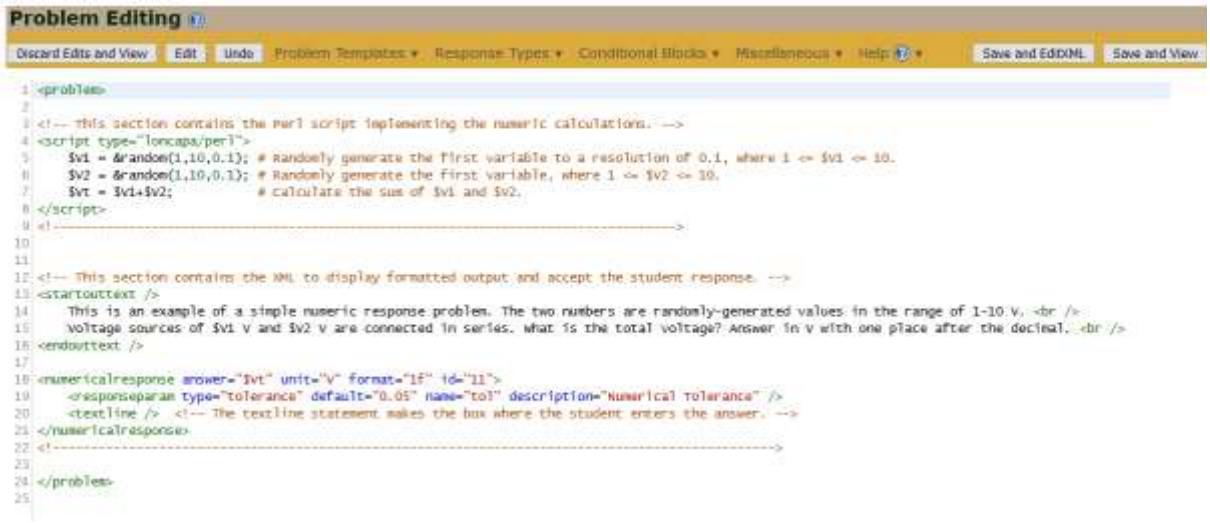


Figure 2: Code for basic Numerical Response problem shown in Figure 1

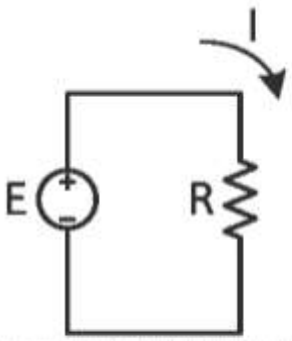
three variables:

$V1 is a randomly-generated voltage from 1-10 V in steps of 0.1 V
$V2 is also a randomly-generated voltage from 1-10 V in steps of 0.1 V
$Vt is the total voltage, defined as the sum of $V1 and $V2

Lines 13-21 contain the XML code to format and display the output. Line 18 specifies the answer to be $Vt with units of "V" and format of one place after the decimal, "1f". The format parameter controls how the correct answer will be displayed to the student after the answer is entered. The "id" parameter must be unique, and is useful for some code constructs, such as multi-part problems, which are described later in the paper. Line 19 specifies the tolerance allowed for the student's response, ±0.05 in this case, which requires an answer correct to one decimal place. If the answer is outside this tolerance it is marked wrong.

Another common approach for defining answer tolerances is percentage. In line 19, if "0.05" is replaced with "15%" then any answer within ±15% would be accepted.

A third approach that is useful in engineering and science is significant figures, although it requires more setup. Suppose the problem involved a multiplication, and the instructor wanted the student to respond with a certain number of significant figures. The problem depicted in Figure 3 requires two significant figures. Although the answer entered is numerically correct, it has four significant figures, so LON-CAPA prompts the student to try again with fewer digits. LON-CAPA is looking for 39 mA, 0.039A, or an equivalent expression with two significant figures. Many students struggle to internalize the concept of significant figures, so problems like this help to reinforce it.



This is an example of a numeric response problem requiring an answer in a certain number of significant figures, two in this case.
Find the current through a 75-Ω resistor if 2.9-V is applied across its terminals. Answer in mA with two significant figures.

38.67 mA

Submit Answer Submission not graded. Use fewer digits. Tries 0 Previous Tries

Figure 3: Numerical Response problem requiring two significant figures; incorrect

The code for the "sig fig" problem is shown in Figure 4. The setup is more complicated, but

Problem Editing

Discard Edits and View | Edit | Undo | Problem Templates | Response Types | Conditional Blocks | Miscellaneous | Help | Save and EditXML | Save and View

```
1  <problem>
2
3  <!-- This section contains the Perl script implementing the numeric calculations. -->
4  <script type="loncapa/perl">
5      $V      = &random(1,10,0.1);                          # Randomly generate voltage from 1.0-10.0 V in steps of 0.1 V.
6      $R      = &choose(&random(1,8,1),10,22,33,47,56,68,75,82); # Randomly pick the 2nd variable from a list.
7      $I      = $V/$R * 1e3;                                # Calculate the current in mA.
8      $sigfig = 2;                                          # Required number of sig figs for answer.
9      if (log10(abs($I))%1 == 0)  # Check for even power of ten
10         #  Reason for /2 in calculation --> +/- 1/2 of lowest sig fig
11         [ $tol = 10**(ceil(log10(abs($I)+1))-$sigfig)/2; ]  # If pwr of 10, add 1 to move left 1 decimal
12     else
13         [ $tol = 10**(ceil(log10(abs($I)))-$sigfig)/2; ]    # If not pwr of 10, do straight calculation
14 </script>
15 <!------------------------------------------------------------------------------>
16
17
18 <!-- This section contains the XML to display formatted output and accept the student response. -->
19 <!-- Display figure floating to the right side -->
20 <img src="Images/OneCkt.png"/>
21 <startouttext /><br />
22     This is an example of a numeric response problem requiring an answer in a certain number of significant figures, two in this case. <br />
23     Find the current through a $R-&Omega; resistor if $V-V is applied across its terminals. Answer in mA with two significant figures. <br />
24 <endouttext />
25
26 <numericalresponse answer="$I" unit="mA" format="2s" id="11">
27     <responseparam type="tolerance" default="$tol" name="tol" description="Numerical Tolerance" />
28     <responseparam name="sig" type="int_range" default="$sigfig" description="Significant Figures" />
29     <textline />  <!-- The textline statement makes the box where the student enters the answer. -->
30 </numericalresponse>
31 <!------------------------------------------------------------------------------>
32
33 </problem>
34
```

Figure 4: Code for problem of Figure 3, which requires an answer with two significant figures

straightforward. Lines 8-13 specify the number of significant digits and calculate an appropriate (absolute) tolerance based on the answer and number of sig figs required. In line 26 the format has changed from specifying decimal places to specifying sig figs: "2s". In line 27, the tolerance is now set to the variable "$tol", calculated above in the Perl script. An additional response parameter has been added in line 28, which specifies the number of sig figs, "$sigfig". This latter parameter could be hard-coded as the number 2, but in this case is defined as a variable for more code flexibility.

In addition to figures like schematics or diagrams, sometimes it is useful to include plots or graphs. The next section describes how to use gnuplot to plot a function and embed it into a problem.

Plotting Functions with Gnuplot

Reading and interpreting plots/graphs is an important skill in the engineering professions. Gnuplot is a graphing utility with versions for several different operating systems. Although copyrighted, it is available for download at no charge, and is used as a plotting engine for some third-party applications [2]. LON-CAPA uses gnuplot for creating graphs of functions and plots of datasets [1]. This section discusses graphs of functions.

Figure 5 shows an example of a problem that uses gnuplot to graph a sine wave. The graph uses

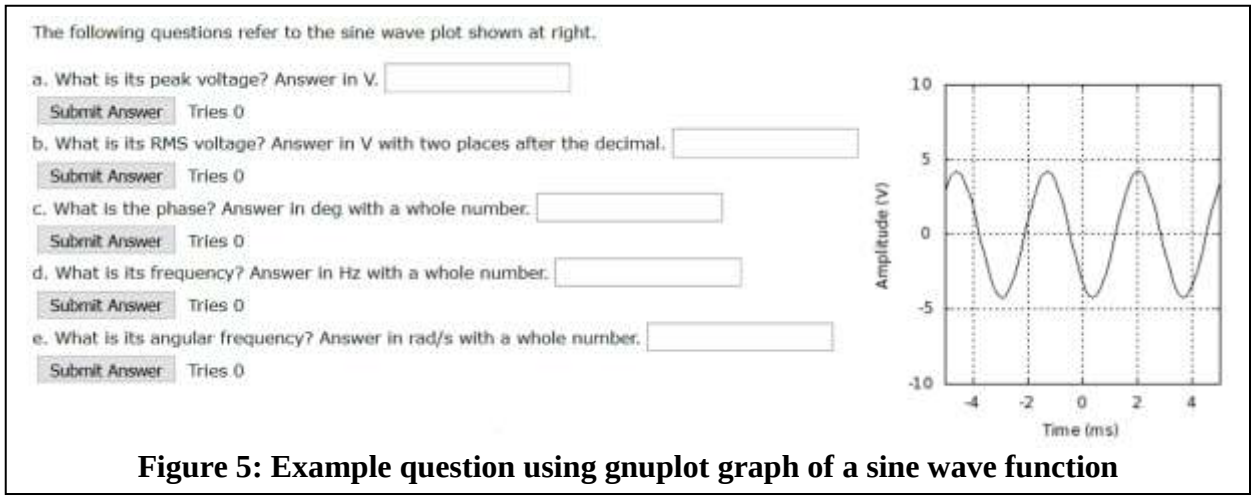


Figure 5: Example question using gnuplot graph of a sine wave function

randomly-generated values for the peak voltage, period, and phase angle. The student must identify the values of these parameters and enter them. Since the numbers are randomly generated, each student gets a different graph to evaluate. This problem simultaneously achieves two goals: giving the students practice interpreting graphs, and reinforcing sinusoidal voltage waveform characteristics.

The Perl script for this problem is shown below in Figure 6. Lines 5-7 generate random values

```
1  <problem>
2
3  <script type="loncapa/perl">
4  # Sine equation setup
5  $Vp     = &random(1,10,0.1);
6  $T      = &random(3,5,0.1);     # Period in ms
7  $theta  = &random(-179,179,1);
8
9  # Calculations
10 $Vrms       = $Vp/&sqrt(2);                    # RMS voltage
11 $f          = 1/($T/1000);                     # Frequency in Hz
12 $omega      = 2*$pi*$f;                        # Angular frequency in radians
13 $omegaplot  = 2*$pi*$f/1000;                   # Angular frequency in radians to plot in ms
14 $function   = "$Vp*sin($omegaplot*x+$theta*$pi/180)";
15 </script>
```

Figure 6: Perl script for problem using gnuplot-generated sine wave

for the sine wave parameters; $Vp and $theta are also the answers to parts a and c of the problem. Lines 10-12 perform calculations to generate answers for parts b, d, and e. Lines 13 and 14 define variables to be used by gnuplot: $omegaplot is in units of milliseconds for plotting, and $function is the function definition to be plotted. Note that all of the variables are Perl variables except for "x", which is the horizontal axis variable in the plot.

Figure 8 shows the XML code that calls gnuplot. Line 22 defines general parameters, such as the

```
22 <gnuplot width="300" grid="on" align="right" font="9" height="300" border="on" samples="100" bgcolor="xFFFFFF" fgcolor="x000000" transparent="off"
   plottype="Cartesian">
23     <axis xmin="-5" ymax="10" color="x000000" ymin="-10" xmax="5" />
24     <xlabel>Time (ms)</xlabel>
25     <ylabel>Amplitude (V)</ylabel>
26     <curve linestyle="linespoints" name="My Plot" color="x000000" pointtype="0">
27         <function>$function</function>
28     </curve>
29 </gnuplot>
```

Figure 8: XML code to call gnuplot and graph sine wave

type of plot, width, alignment, and font size to be used. The subsequent statements define the plotting range, axis labels, line style, and function to be plotted ($function from the Perl script). The XML editor is often the most expedient way of coding for experienced users, but when first learning something new, the "colorful editor" can be very helpful. Figure 7 shows part of the

Figure 7: Colorful editor view showing a portion of the gnuplot parameters

gnuplot section in the colorful editor. Although not nearly as compact, the colorful editor presents boxes to be filled in so the coder does not need a priori knowledge of the precise XML syntax. Even for experienced users, it can be handy when trying new parameters. Moreover, the many standard templates in LON-CAPA [1, p. 2] include one for creating a gnuplot graph of a function, so those using gnuplot for the first time can start with a working problem and modify it to create their own material.

One other characteristic of this problem is worth noting: It has multiple interrelated parts. This is a great feature of LON-CAPA that is very straightforward to implement. It allows for setting up complicated problems, breaking them into pieces, and carrying numbers and calculations forward within the larger problem. Each problem part is graded independently of the others, so students get some credit even if some of the parts are wrong. A pair of XML "part" statements delimits each part of a problem. Figure 9 depicts the code for the first two parts of this problem. Line 31 indicates the start of the first part, labeled "a" by the "id" qualifier. Next is the text for that part, followed by a numerical response similar to the one described in the previous section. Lines 42 and 51 delineate part b, which contains code similar to part a, but for a different parameter. The next section describes how to plot (x,y) datasets.

```
31 <part id="a">
32 <startouttext />
33 &currentpart.  What is its peak voltage? Answer in V.
34 <endouttext />
35 <numericalresponse answer="$Vp" unit="V" format="1f" id="11">
36         <responseparam name="tol" default="0.5" description="Numerical Tolerance" type="tolerance" />
37         <responseparam name="sig" default="0,15" description="Significant Figures" type="int_range" />
38         <textline readonly="no" />
39 </numericalresponse>
40 </part>
41
42 <part id="b">
43 <startouttext />
44 &currentpart.  What is its RMS voltage? Answer in V with two places after the decimal.
45 <endouttext />
46 <numericalresponse answer="$Vrms" unit="V" format="2f" id="12">
47         <responseparam name="tol" default="0.005" description="Numerical Tolerance" type="tolerance" />
48         <responseparam name="sig" default="0,15" description="Significant Figures" type="int_range" />
49         <textline readonly="no" />
50 </numericalresponse>
51 </part>
```

Figure 9: First two parts of gnuplot function graphing problem

Plotting Datasets with Gnuplot

Not all graphs are generated from pre-defined functions. LON-CAPA also incorporates gnuplot functionality to plot datasets of (x,y) coordinate values [1]. The standard LON-CAPA template, described in this section, uses a sine function to generate the data, but the data could be entered manually or read from a file. Since the template uses a sine to generate the (x,y) data, the graph looks similar to the one shown in the previous section. The Perl script, shown below in Figure 10, is also similar, although it uses a "for" loop and "push" statements in lines 11-16 to create the dataset point by point.

```
8  $amplitude = &random(1,4,0.5);
9  $x_min = -5;
10 $x_max =  5;
11 for ($x=$x_min;$x<=$x_max;$x=$x+0.05) {
12    push(@X,$x);
13    push(@Y,$amplitude*sin($x));
14 # Safeguard:
15 # The following line limits the size of the array to 1000 to avoid infinite loops
16    if (($#X>1000) || ($#Y>1000)) { last; }
17 }
```

Figure 10: Perl script to generate dataset for gnuplot problem

The gnuplot section is also similar to that of the problem in the previous section, except that two "data" statements replace the "function" statement, as shown in lines 26-27 of Figure 11. One caution: Note that the data points are not grouped as (x,y) pairs. Line 26 contains all of the "x" values and line 27 contains all of the "y" values. The first (x,y) pair consists of the first value in line 26 and the first value in line 27. Although this example uses data arrays, the values can also be specified as comma-separated lists of numbers or Perl-script variables. As before, the colorful editor can also be helpful, especially when setting up a problem for the first time.

```
20 <gnuplot width="300" transparent="off" samples="100" grid="on" font="9" bgcolor="xffffff"
21          height="300" align="left" fgcolor="x000000" border="on" plottype="Cartesian" >
22     <axis xmin="$x_min" ymin="-5" xmax="$x_max" ymax="5" color="x000000" />
23     <xlabel>Label X</xlabel>
24     <ylabel>Label Y</ylabel>
25     <curve linestyle="linespoints" name="My Plot" pointtype="0" color="x000000">
26         <data>@X</data>
27         <data>@Y</data>
28     </curve>
29 </gnuplot>
```

Figure 11: Gnuplot portion of dataset plotting problem

Multipart Problems

A very useful and powerful feature of LON-CAPA is its ability to form multi-part problems. This capability was introduced above in the *Plotting Functions with Gnuplot* section. In that example, all of the problem parts contained Numerical Responses. Although this is fine, LON-CAPA provides the flexibility to include other problem types, both within a single part and across multiple parts.

For instance, sometimes it is useful to ask both qualitative and quantitative questions in a single problem. The example shown in Figure 12 depicts a problem that generates a graph with

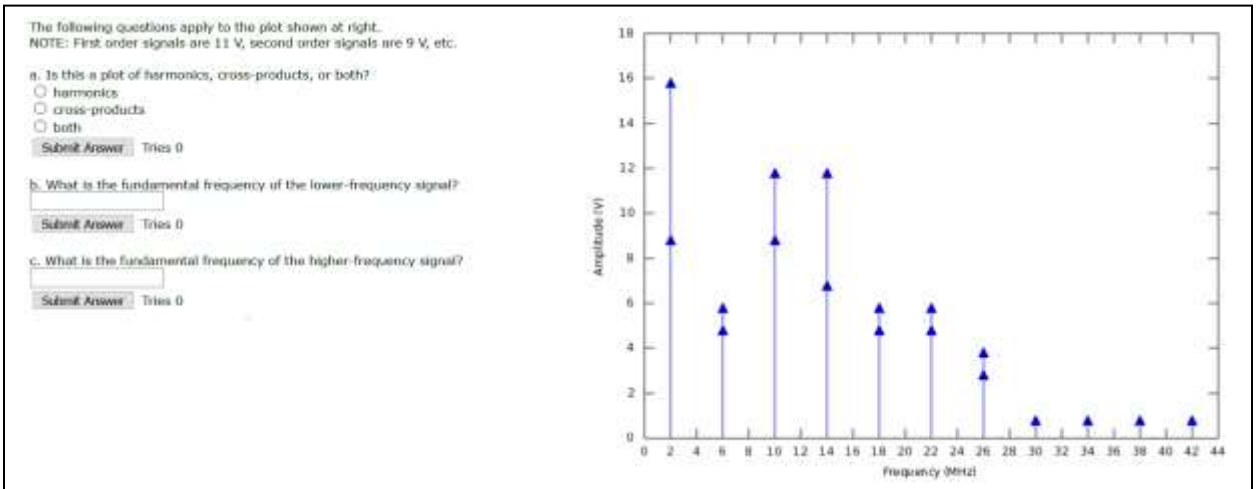


Figure 12: Multi-part problem with gnuplot-generated graph and parts of different types

randomly-generated vectors. Part "a" is a qualitative question of Radio Button Response type, which requires identifying the class of displayed vectors. Parts "b" and "c" are Numerical Response, requiring the student to interpret the graph and, in some cases, perform calculations.

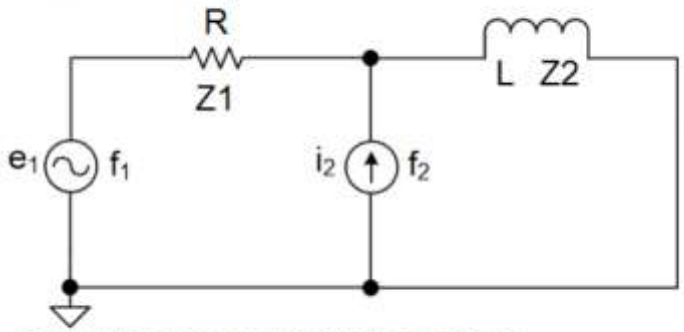
The possibilities for displaying multi-part problems are only limited by the author's needs and imagination. Another option that is sometimes useful is to show parts one at a time, only displaying a given part after the previous part has been answered. Yet another alternative is to only show something within a part if the part has been solved (or not solved). The example shown in Figure 13 is a problem with randomly-generated parameters that apply to the diagrammed circuit. The first two parts each include two Numerical Response values. Part "c" contains a Formula Response query, which requires entry of an algebraic equation. It also contains a "solved" section with a gnuplot graph that is only displayed after the question is answered. In Figure 13 it has not yet been answered, so the graph and further instructions are not yet shown.

The following parameters apply to the circuit shown.

e_1 = 900 mV_{rms} ∠64° f_1 = 190 kHz

I_2 = 1.9 mA_{rms} ∠67° f_2 = 80 kHz

R = 620 Ω L = 560 μH



a. Calculate the voltage across the inductor caused by e_1:
- Magnitude in first box in mV (rms) as a whole number;
- Phase in the second box in deg as a whole number.

Submit Answer Tries 0

b. Calculate the voltage across the inductor caused by I_2:
- Magnitude in first box in mV (rms) as a whole number;
- Phase in the second box in deg as a whole number.

Submit Answer Tries 0

c. Write the *time domain* equation for the composite voltage across the inductor.
Round the DC offset to three decimal places, sine coefficient to two decimal places, and frequency and phase to whole numbers.
NOTE: Leave units out of the equation.

mV

Submit Answer Tries 0

Figure 13: Multi-part problem with Numerical Responses, Formula Response, and "hidden" gnuplot that is only displayed after part "c" is answered

The Formula Response operation has a nifty feature that shows how LON-CAPA is interpreting the formula entry so the student can verify the formula before clicking the Submit button. This feature is illustrated in Figure 14. It is the pop-up box immediately below the text entry box.

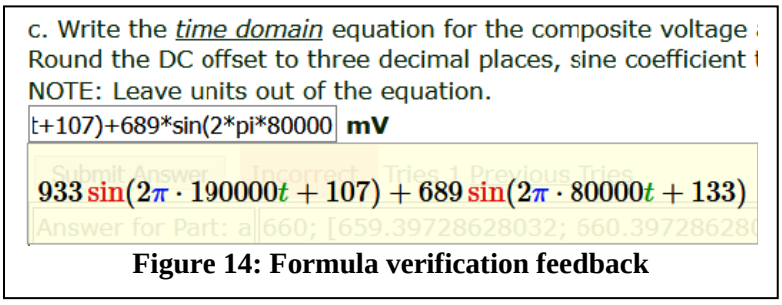


Figure 14: Formula verification feedback

In this case, after the problem is answered LON-CAPA displays instructions to plot the waveform described by the formula, using MATLAB. It also shows a gnuplot graph of the

formula so the student knows what the MATLAB plot should look like. This is depicted below in Figure 15. Also, remember that all of this uses numbers that are randomized for each student.

c. Write the *time domain* equation for the composite voltage across the inductor.
Round the DC offset to three decimal places, sine coefficient to two decimal places, and frequency and phase to whole numbers.
NOTE: Leave units out of the equation.
933*sin(2*pi*190000*t+107) + 689*sin(2*pi*80000*t+133) mV

Use MATLAB to plot the composite capacitor voltage in the time domain for two complete cycles of the lower-frequency waveform.

It should look very similar to the plot below. Remember to add a title and axis labels.
Capture this screen (Alt-PrintScreen) including the plot below with the date and your name.
Paste it into a Word document and crop it appropriately. Do the same with your MATLAB plot.
Turn in the plots and MATLAB code at the beginning of the next class period.

Name: Gene Harding Date: Thu Jan 31 19:02:12 2019

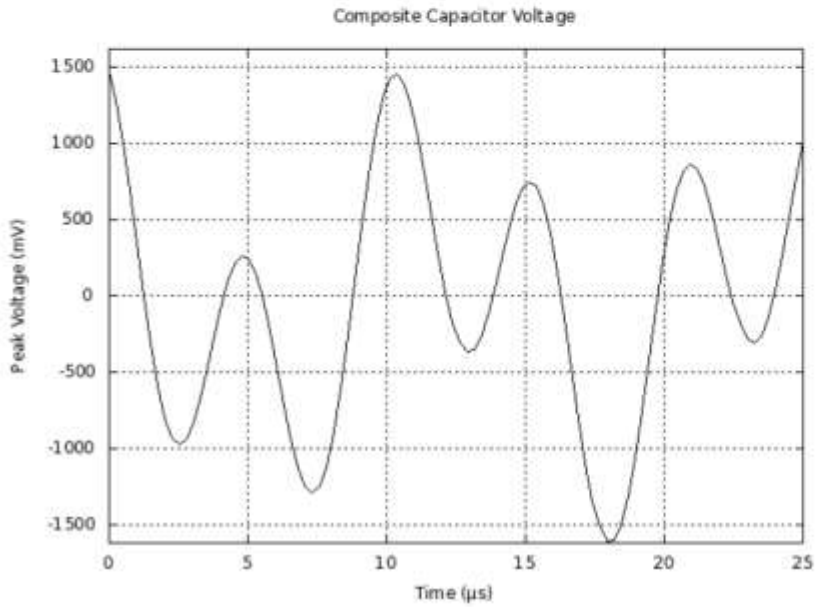


Figure 15: Plot and instructions to do MATLAB plot, displayed after problem is answered

Implementing LON-CAPA at a New Institution

Implementation of LON-CAPA requires either a dedicated server or a virtual machine (VM) running Linux (RHEL/CentOS/Scientific Linux, SuSE Linux Enterprise Server, or Ubuntu LTS), or use of a hosting provider [3,4]. The LON-CAPA application code is open source, and there is no cost to download or run it. LON-CAPA relies on other free software, such as the Apache web server with mod_perl and memcache, several Perl modules from CPAN, and also LaTeX, MySQL/MariaDB, MAXIMA, CKEditor, CodeMirror, gnuplot and R.

If an institution chooses to manage its own LON-CAPA instance, then the starting point is to set up a Linux server or VM as a “library” node for the institution's LON-CAPA domain. The recommendation is to establish a single domain for each institution, and (initially) to deploy a

single library node for that domain. A library node is used for permanent storage of user and course data, as well as any resources generated by content creators in the domain. In any domain there can be more than one LON-CAPA node, and each node will be one of two types: “library” or “access”. Access nodes can be thought of as web application servers, i.e., they host user sessions and will store data temporarily, but they need to connect to the library node identified as the “home server” of the user (or the course) for data to be preserved permanently. A server class machine (e.g., 2 CPUs; 8 GB RAM; 60 GB storage), or a VM allocated similar resources, are both good starting points for a LON-CAPA library node for a small number of courses, and up to a couple hundred students.

The procedure for installing LON-CAPA is:

(a) Install a minimal version of one of the supported Linux distros;
(b) Add the LON-CAPA package repository for the particular Linux distro to the list of available sources;
(c) Use the distro's package manager, i.e., yum, zypper, or apt-get to install the LONCAPA-prerequisites.rpm or loncapa-prerequisites.deb package, to install dependencies;
(d) Download a small tar file, and extract a Perl script, which when run (as root) will prepare a system for LON-CAPA, including download of the current stable LON-CAPA release;
(e) Run the ./UPDATE command in the downloaded release to create directories, copy files and set permissions, and the initial domain configuration;
(f) Use a script in the downloaded release to create a file system authenticated user, who will be assigned a domain coordinator role;
(g) Start the LON-CAPA daemons, and the web server; and
(h) Use a web browser to log-in as the user created at step (f) [5].

Installation of library and access nodes is identical, except step (f) is only needed for the library node.

Once the domain coordinator has logged in via the web interface, additional configuration, creation of users and courses, and assignments of roles to users can all be accomplished from links on the domain coordinator's main menu page [7]. The LON-CAPA application supports integration into an institution's IT environment in order to leverage centralized services for authentication, enrollment, validation of instructors of record (for course creation requests) and updates to user information [6,7].

Although a LON-CAPA domain can include more than one library node, common practice is to only have a single library node in each domain. Scalability is achieved by adding access nodes and load-balancing user sessions across them. A library node will also need to be able to host user sessions for content creators, as the workspace used when authoring or modifying content, referred to as the “Authoring Space” in LON-CAPA, is only accessible via web browser access to a user session hosted on the library node, or via webDAV access to the library node. Although a LON-CAPA domain can be set up as an island (i.e., a single library node, or a cluster of library and access nodes in the same domain), one of the key features of the system is the ability to share content between institutions. In order to do that a domain needs to join the LON-CAPA network of collaborating institutions. Once a domain has joined the network, resources published to the cross-institutional repository will be available for use in LON-CAPA courses in the domain. There is no cost to join the network, or to use the shared content available from the

cross-institutional repository. Content authors set distribution rights for resources they add to the repository, when they "publish" content created in an Authoring Space. The default is to allow system-wide use, i.e., instructors in other domains can browse and search for the content, and then import into a course, but more granular distribution rights can be set, as needed, e.g., restricted to specific domain(s), and/or specific course(s).

The LON-CAPA Academic Consortium Board maintains the authoritative list of nodes and domains that belong to the network [8]. All nodes in the network retrieve this authoritative list daily and update locally cached information about network membership. Both forward and reverse DNS mapping is required for the hostname and IP address of each LON-CAPA node, and a static IP address needs to have been assigned. LON-CAPA requires that standard web ports 80 and 443 are open, and in addition, a dedicated port (5663) needs to be open to other nodes in the network. LON-CAPA uses OS-level firewall rules to restrict which remote hosts can connect to port 5663, and the daemon-based process that listens for connections to port 5663 (to handle transactions from other nodes) only permits connections from known nodes in the network. A domain can also choose to only allow connections from hosts that use an SSL tunnel as part of connection initiation. The SSL certificates used to create the secure tunnel need to have been signed by the LON-CAPA Certificate Authority (run by the Academic Consortium). Some institutions have security policies in place that require campus border firewall exceptions to permit access to port 5663 for inter-node communication from other LON-CAPA domains.

Conclusion

LON-CAPA is an outstanding online tool that is uniquely well-suited for use by educators in the science, technology, engineering, and mathematics (STEM) disciplines. Although it has been around for many years, it is not well known outside of the science and mathematics communities. The goal of this series of papers is to spread the word among engineering and engineering technology educators. This paper focused on LON-CAPA's capabilities with Numerical Response questions, dynamic plotting/graphing using the gnuplot interface, and mixed-type multi-part problems. It also described what is involved in setting up LON-CAPA for the first time.

There is also a very active and helpful LON-CAPA listserv. A wealth of information is available at the LON-CAPA web site, www.loncapa.org. Interested readers are encouraged to attend the annual LON-CAPA conference, normally held at the end of May each year. It is a great place to meet other users and learn more about using the system!

---

[1] *Learning Online Network with CAPA: Author's Tutorial and Manual*. LON-CAPA Group, Michigan State University, East Lansing, MI, May 11, 2017. . Available at: https://loncapa.msu.edu/adm/help/author.manual.pdf. [Accessed January 31, 2019]

[2] Anon, (2019). gnuplot homepage. [online] Available at: http://www.gnuplot.info/. [Accessed January 27, 2019.]

[3] *The Learning Online Network with CAPA: "Before You Start"*. LON-CAPA Academic Consortium (2013). Available at: https://loncapa.org/beforeyoustart.html. [Accessed February 2, 2019.]
[4] *The LearningOnline Network with CAPA: "Commercial Hosting and Support"*. LON-CAPA Academic Consortium (2013). Available at: https://loncapa.org/hosting.html. [Accessed February 2, 2019.]
[5] *The Learning Online Network with CAPA: "Install LON-CAPA"*. LON-CAPA Academic Consortium (2013). Available at: http://install.loncapa.org/. [Accessed February 2, 2019.]
[6] *Learning Online Network with CAPA: Domain Coordination Tutorial And Manual*. LON-CAPA Group, Michigan State University, East Lansing, MI, May 11, 2017. Available at: https://loncapa.msu.edu/adm/help/domain.manual.pdf. [Accessed February 2, 2019.]
[7] *The LearningOnline Network with CAPA: "Integration"*. LON-CAPA Academic Consortium (2013). Available at: https://loncapa.org/integration.html. [Accessed February 2, 2019.]
[8] *The LearningOnline Network with CAPA: "Member Institutions"*. LON-CAPA Academic Consortium (2013). Available at: https://loncapa.org/consortiummembers.html. [Accessed February 2, 2019.]