Magnetic Design Web Application
Michael Brim

Senior Project
ELECTRICAL ENGINEERING DEPARTMENT
California Polytechnic State University
San Luis Obispo
June 2010
Abstract

The automated Magnetic Design Web Application, available at http://www.magneticdesign.org, generates the full design process of a power transformer, gapped inductor and toroidal inductor given design specifications provided by the user. Calculations are accomplished using Area Product, $A_p$, or Core Geometry, $K_g$, procedures. All design processes mirror the format used in Dr. Taufik’s *Introduction to Magnetic Design* lecture-notes. An administrative backend is provided where instructors can add or modify core design values to expand the array of materials used. The project aims to aid student understanding of magnetic design and course concepts while allowing instructors to broaden the scope of material covered within EE 433, and any subsequent, magnetic design courses.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>II. Background</td>
<td>3</td>
</tr>
<tr>
<td>III. Requirements</td>
<td>5</td>
</tr>
<tr>
<td>IV. Design</td>
<td>7</td>
</tr>
<tr>
<td>Data Collection</td>
<td>7</td>
</tr>
<tr>
<td>Preliminary Design</td>
<td>9</td>
</tr>
<tr>
<td>Step Consolidation</td>
<td>10</td>
</tr>
<tr>
<td>Back-End Administration</td>
<td>13</td>
</tr>
<tr>
<td>Front-End User Interface Presentation</td>
<td>14</td>
</tr>
<tr>
<td>V. Implementation and Testing</td>
<td>20</td>
</tr>
<tr>
<td>VI. Conclusion and Recommendation</td>
<td>22</td>
</tr>
<tr>
<td>Reference List</td>
<td>23</td>
</tr>
</tbody>
</table>

## Appendix

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appendix A. Source Code Samples</td>
<td>24</td>
</tr>
<tr>
<td>Appendix B. Parts and Cost List</td>
<td>32</td>
</tr>
<tr>
<td>Appendix C. Analysis of Senior Project Design</td>
<td>33</td>
</tr>
<tr>
<td>Summary of Functional Requirements</td>
<td>33</td>
</tr>
<tr>
<td>Primary Constraints</td>
<td>33</td>
</tr>
<tr>
<td>Economic</td>
<td>33</td>
</tr>
<tr>
<td>Environmental / Manufacturability / Ethical / Health and Safety / Social and Political</td>
<td>33</td>
</tr>
<tr>
<td>Sustainability</td>
<td>34</td>
</tr>
<tr>
<td>Development</td>
<td>34</td>
</tr>
</tbody>
</table>
List of Tables and Figures

Figure 1 - Original Step Output for Gapped Inductor Design using Core Geometry .......... 9
Figure 2 - Consolidated Code for the Gapped Inductor using Area Product .................. 11
Figure 3 - Consolidated Code for the transformer’s total copper loss, $P_{CU}$, which totals 34 lines and includes the PHP calculation code and the HTML mark-up code .................. 12
Figure 4 - Administration Pane for Assigning Core CSVs .................................... 14
Figure 5 - Administration Pane for Editing Current Core Values ............................... 14
Figure 6 - Step Output w/ Formatting ........................................................................ 15
Figure 7 - Step Output w/o Formatting ....................................................................... 15
Figure 8 - Step with No Mouse Hover-Over ................................................................. 16
Figure 9 - Step with Mouse Hover-Over Displaying Background Highlight and Prototip Tooltip ................................................................................................................. 16
Figure 10 – What a user sees when first visiting magneticdesign.org ............................ 18
Figure 11 - The second options presented to a user ................................................. 19
Figure 12 - Specifying Input Variables for a Power Transformer Using Area Product (Default Values Shown) ................................................................. 19
I. Introduction

Proper circuit design requires appropriate component selection to satisfy certain design specifications. In the majority of designs, both passive components (resistors and capacitors) and active components (inductors and transformers) will be necessary to achieve desired results. All electrical engineers should be adequately familiar with the selection of passive components not only because they require limited input and calculation criteria (e.g. resistance and power ratings), but also that resistors and capacitors (of a wide-ranging variety) are massively produced and often available for purchase “over-the-counter.”

Magnetic design of active components is not usually a part of an electrical engineer’s core academic curriculum and thus, the concept of transformer or inductor design is mostly a foreign concept. Additionally active components are not produced on the same grand scale as passive components, often necessitating their individual design over the purchase of a packaged product.

The aforementioned educational discrepancy is not a problem unique to Cal Poly as there is little coverage of magnetic design across most American universities. This, unfortunately, creates a certain professional disconnect within industry as incoming electrical engineers, even those specializing in the power electronics field, are universally unversed in magnetic design. With a lack of readily manufactured active components available and lack of magnetic component design knowledge, the Magnetic Design Web Application aims to
provide a learning tool for both students and professionals to help them with the calculation process of transformer, gapped inductor and toroidal inductor design.
II. Background

The problems highlighted in the previous chapter present a pressing need for a tool to assist engineers in the design, creation and selection of magnetic components. The Magnetic Design Web Application addresses this need while additionally providing an educational foundation of magnetic design for those unfamiliar with the concepts.

Cal Poly recently re-introduced senior-level electrical engineering class EE 433 - Introduction to Magnetic Design to the Technical Elective curriculum; the class introduces prospective electrical engineers to the intricacies of design for both transformers and inductors. In the course, students are often required to progress through lengthy and laborious twenty-plus step design processes – with an expressed emphasis placed on achieving pre-determined efficiency, regulation and thermal constraint specifications. Altering the original design values, core material or core configuration requires that the entire process must be repeated; a very time-consuming endeavor. The benefits of streamlining the process through a computational algorithm are readily apparent as they provide the user a flexibility to instantaneously generate new, or modified, calculations.

The automated magnetic design web application seeks to supplement Dr. Taufik’s lecture notes in aiding a student’s understanding of the magnetic design process. Through its use, a student would not only be able to generate additional example problems, but also expedite and validate laboratory and project calculations. Through its provided administrative
backend, Dr. Taufik and other professors would be able to add additional core
configurations, core materials and core-loss variables. Such instantaneous implementation
would allow for instructors to further broaden the scope of material covered within the
course.
III. Requirements

The automated magnetic design web application generates the full design process of a power transformer, gapped inductor and toroidal inductor given design specifications provided by the user. Calculations are accomplished using Area Product, $A_p$, or Core Geometry, $K_g$, procedures. All design processes mirror the format used in Dr. Taufik’s *Introduction to Magnetic Design* lecture-notes.

The application is written in the web-development scripting language, PHP, with core specifications and wire sizing contained within comma separated variable (CSV) spreadsheets. The hierarchical selection process is controlled through simple JavaScript logic and animated using Prototype and Scriptaculous JavaScript framework scripts. The variable hover-over is accomplished by using the program, Protip2, which is JavaScript-based.

All presentation markups are accomplished through Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS); images were created in Adobe Photoshop and Adobe Illustrator.
Technical requirements for the magnetic design web application are minimal, only including:

- an internet server running PHP 5.0
- provided code required for referencing and extracting data from comma-separated variable (CSV) sheets
  

Requirements to accurately progress through the design process, aside from the mathematical calculations, include:

- proper floating points for precise variable value storage
- targeted core and wire data extraction through a simple CSV searching algorithm¹
- ability to use a variety of rounding rules (e.g. floor or ceiling a variable, rounding to next greater even number)

---

¹ using this method assumes that the column the value is located in has already been sorted by size (smallest to largest)
IV. Design

Due to the unique nature of this project, there were two separate design elements of the Magnetic Design Web Application. The first is the assumed programming involved required for the mathematical computations, referencing correct data-tables and the overall structure of the application. The second was the presentation element of the data. Because the application is to supplement educational materials, it would beneficial if the front-end design of the application mirrored the steps of Dr. Taufik’s Introduction to Magnetic Design lecture notes.

Data Collection
The magnetic design process pulls data from multiple sources and thus, before I could start on the project, I had to recreate and itemize all necessary tables and values in appropriate CSV tables and PHP files.

- Core Configuration Constants (page 123) are contained within `transformers_cores.php`
- Core Loss Equation Variables (pages 125-126, 291-292) are all contained in `transformers_material.php`
- Input Variables (controlled via form) are contained within `transformers_form.php`
- AWG Wire Table (page 225) is contained within `transformers_wire_size.csv`
- Magnetics Inc. Tape Toroidal Core Table (pages 241-242) is contained within `maginc_tape_toroidal_cores.csv`
- Magnetics Metals EI and EE Laminations Core Table (pages 243-244) is contained within `magnetics_metals_ei_and_ee_lamination_cores.csv`
- National Magnetics Corp. 2-mil C Cores Table (pages 245-246) is contained within `mag_corp_2_mil_c_cores.csv`
• Magnetics Inc. Toroidal Ferrites Core Table (pages 247-248) is contained within maginc_toroidal_ferrites_cores.csv
• Micrometals Powdered Iron Core Table (pages 289-290) is contained within micrometals_powdered_iron_cores.csv
• Magnetics Inc. MPP Powder Core Table (pages 289-290) is contained within maginc_mpp_powder_cores.csv
**Preliminary Design**

Original design plans called for six different self-contained PHP files, each representing one magnetic design process and containing between seventeen to twenty-two steps; averaging 700 lines of code and a 20KB file-size. The only globally referenced elements were the aforementioned data collection PHP and CSV files. This design featured no presentation elements, only a simple logger that would output calculated step-values to compare to the worked examples in the lecture notes; this format is shown in Figure 1.

Step 1
Current, \( I_{ref} = 5.25 \) [A]

Step 2
Energy = 0.01378125 [W-s]

Step 3
Electrical Conditions Constant, \( K_3 = 0.002842 \)

Step 4
Core Geometry Coefficient, \( K_0 = 0.033413591056 \) [cm^2]

Step 5
Using Core: CL-5

Step 6
Current Density, \( J = 647.615131579 \) [A/cm^2]

Step 7
Bare Wire Size, \( A_{wire} = 0.00810666666667 \) [cm^2]

Calculated Maximum Wire Size: 0.00891733333333 [cm^2]

**Figure 1 - Original Step Output for Gapped Inductor Design using Core Geometry**

The math required to complete the magnetic design process is simple algebra and thus, all necessary math functions are contained standard within the PHP core. The only challenging
programming aspect to complete the calculations was to correctly extract core values contained in the CSV tables; this was accomplished by using the PHP function fgetcsv().

While this design was simple and straight-forward, many of the individual steps are similar or the same amongst all six iterations. The current design did not lend itself to be easily scalable as adding additional magnetic designs would require hours of coding and the project could very easily become bloated.

**Step Consolidation**
Due to the gross redundancy of repeated code used for the individual steps, an easy solution to consolidate code and minimize the amount of work was to make the individual steps into their own independent files and “call” them in the six magnetic design files. Additionally, because each magnetic design required a different set of input variables, there were six different transformer_form.php files. The same logic used for step consolidation could also be applied to the input variables. Thus, the only code contained within the six magnetic design files is an array for the steps required, an array for the input variables and the option to set the default values for the input variables.

The finalized six transformer and inductor files are each 2KB and only contain 60 lines of code (the Gapped Inductor using Area Product file is shown in its entirety in Figure 2). The “missing” lines of code are now contained in the individual step files which include not only the PHP code to calculate the values, but also the HTML stylized code to display the step appropriately.
This is not to say that redundant code was not eliminated; using the previous design, there were 118 total steps. Through consolidation, there are now only 55 steps, each containing anywhere between 35 to 150 lines of code. Figure 3 shows the entire code for one step, the calculation of the total copper loss, which is only 33 lines long.
Additional benefits of the code consolidation include the ability to easily modify and troubleshoot problems with individual steps and the overall design. Because the majority of the code in the individual steps is HTML design markup, instead of staring at a wall of PHP code in the old master files, each individual step sometimes only contains 2-3 lines, making errors much easier to identify.

```php
<?php

// STEP 17

// Calculate the total copper loss

// Pcu = Pp + Ps

$this->pCu = $this->pP + $this->pS;

$title = "Calculate the transformer's total copper loss, P_\text{Cu}\";:

$calculations = array("Pcu" );

$output = &lt;&lt;&lt; FINISH

&lt;div class="equations">
&lt;span class="pcu"><span class="varhigh">P_\text{Cu}\</span&gt;&lt;/span&gt; = &lt;span class="pp" &gt;&lt;span class="varhigh">P_\text{p}\</span&gt;&lt;/span&gt; + &lt;span class="ps">P_\text{S}\</span&gt;&lt;/sup&gt; &lt;/span&gt; [ W ]

<br/>

$P_\text{Cu} = $this-&gt;Pp + $this-&gt;Ps [ W ]

<br/>

&lt;div style="margin-top: 10px;"&gt;&lt;span class="eq_highlight">P_\text{Cu}\</span&gt; = $this-&gt;Pcu [ W ]&lt;/span&gt;&lt;/div&gt;

&lt;/div&gt;

FINISH;

?>
```

Figure 3 - Consolidated Code for the transformer’s total copper loss, $P_{\text{Cu}}$, which totals 34 lines and includes the PHP calculation code and the HTML mark-up code
**Back-End Administration**

When agreeing to advise this project, Professor Taufik expressed a desire for the ability to add additional core materials in the future. To accomplish this, an administrative backend was created allowing one the following options:

- **Add a New Core**
- **Edit a Current Core** (see Figure 5)
- **Change Current CSV Assignments** (see Figure 4)
  
  *e.g. Ferrite F Material currently uses the mag_corp_2_mil_c_cores.csv, one can change this value to any of the available CSVs*

- **Download Current CSVs**
  
  *This is important if one wants to amend values to the current CSV: download, edit, then re-upload*

- **Upload New or Updated CSVs**
  
  *If one doesn’t want to add a core, but rather change a CSV assignment of a current core*

To prevent accidental destructive behavior, all of the CSVs mentioned in the Data Collection section have write-protections enabled to prevent them from being over-written or deleted.

The code required to drive the backend is explained more in depth in the following Implementation and Testing section.
**Front-End User Interface Presentation**

While the original design that only output results would be adequate for someone only desiring design values or looking to validate their own results, it fails to provide an instructional element to the project. Because Professor Taufik’s lecture notes are straightforward and easy to follow, I decided to completely mirror his presentation. Figure 6 and Figure 7 demonstrate the new “lecture notes” formatting and the old “logger” formatting, respectively.
Many of the steps contain fractions and HTML isn’t the most adept at handling them (because of the combination of in-line and block elements in the same line), I originally attempted to parse the equations through LaTeX. Unfortunately, while LaTeX’s presentation is appealing, I was unable to figure out a way to feed values through an online parser. Had I chosen to go with LaTeX, there would have been no intermediate steps in the equations: just the variable set-up and the final value. Ultimately, I found it tremendously easier to stick with using HTML tables.

I started the Magnetic Design Web Application a year removed from taking EE 433 and thus, to re-familiarize myself with the material, I redid all of the lecture note examples. While following the steps, I found myself circling variables and writing down which step number
they came from for future reference. Because I wanted the application to also be used as a learning tool, I thought implementing this sort of “reminder” would be beneficial for those not entirely familiar with magnetic design. To accomplish this, I created a span class that had a hover-over which would change the text’s background, creating a “highlight” effect. I then acquired a non-commercial single-domain license for Prototip2, a tooltip applet that uses the Prototype JavaScript framework, to create hover-over tooltips which displayed the variable’s title and the step it originated from. A demonstration of this effect is shown in Figures 8 (no hover-over or tool-tip) and 9 (with hover-over and tool-tip).

\[
R_p = \frac{\mu \text{M}}{\text{cm}} \times 10^{-6} \left[ \Omega \right]
\]

\[
R_p = 4.7 \times 25 \times 264 \times 10^{-6} \left[ \Omega \right]
\]

\[
R_p = 0.0310 \left[ \Omega \right]
\]

Figure 8 - Step with No Mouse Hover-Over

\[
R_p = \text{MLT} \times \frac{\mu \text{M}}{\text{cm}} \times 10^{-6} \left[ \Omega \right]
\]

\[
R_p = 4.7 \times 25 \times \text{Number of Turns on the Primary Side (step 5)} \times 10^{-6} \left[ \Omega \right]
\]

\[
R_p = 0.0310 \left[ \Omega \right]
\]

Figure 9 - Step with Mouse Hover-Over Displaying Background Highlight and Prototip Tooltip
Prototip2 is set up so that each defined variable is assigned a span class. For easy referencing, I wanted the hover-over tip to mention the variable’s originating step number. Unfortunately, while many of the magnetic designs contain the same variables, these variables aren’t necessarily generated on the same step. With over twenty variables per magnetic design, creating individual classes for each design would result in a bloated styling file. To make the code more efficient, I modified each step to report which variables were being modified; each step contains the following line of code, specifying the generated/modified variable:

```php
$calculations = array("Bac");
```

Within the Prototip styling file, it calls a function which searches for the specified variable (in this instance, $B_{AC}$) and then returns the step number. Thus, when one hovers-over the variable $B_{AC}$, they will see the tool-tip text of “AC Flux Density (Step #).”

```php
"bac" => "AC Flux Density " .this->s("Bac")
```

The shortcut function, s("variable"), is located in Appendix A.
While researching user-interface design methodology, it was almost universally recommended to present users with options in steps, rather than all at once. This idea works on the assumption that if one gives their user a chance to overlook something, they probably will. The Magnetic Design Web Application doesn’t have an abundance of options, but regardless, I thought the selection process should be as linear as possible. When one visits magneticdesign.org, they are presented only with the question of which magnetic component they want to design.

![magnetic design web application](image)

**Figure 10 – What a user sees when first visiting magneticdesign.org**

Clicking on “Power Transformer,” “Gapped Inductor” or “Toroidal Inductor” will highlight that selection and present two more options below: “area product” or “core geometry.”

(See Figure 11)
Figure 11 - The second options presented to a user

Once either is clicked, the user is asked to fill out input variables (default values are automatically entered). When the user clicks on the “Calculate” button, the entire magnetic design process is generated and shown below the aforementioned options. (See Figure 12)

with the following design specifications:

<table>
<thead>
<tr>
<th>Input Voltage, $V_{IN}$</th>
<th>28 [V]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Voltage, $V_O$</td>
<td>28 [V]</td>
</tr>
<tr>
<td>Output Current, $I_O$</td>
<td>3 [A]</td>
</tr>
<tr>
<td>Switching Frequency, $f$</td>
<td>20000 [Hz]</td>
</tr>
<tr>
<td>Efficiency, $\eta$</td>
<td>0.98</td>
</tr>
<tr>
<td>Regulation, $\alpha$</td>
<td>1</td>
</tr>
<tr>
<td>Temperature Rise</td>
<td>25 [°C]</td>
</tr>
<tr>
<td>Flux Density, $B_m$</td>
<td>0.3 [T]</td>
</tr>
<tr>
<td>Diode Voltage Drop, $V_D$</td>
<td>1 [V]</td>
</tr>
<tr>
<td>Core Material</td>
<td>Permalloy 2-mil</td>
</tr>
<tr>
<td>Core Configuration</td>
<td>C Core</td>
</tr>
<tr>
<td>Waveform</td>
<td>squarewave</td>
</tr>
</tbody>
</table>

Figure 12 - Specifying Input Variables for a Power Transformer Using Area Product (Default Values Shown)
V. Implementation and Testing

Luckily, due to the nature of this senior project, implementation, testing and troubleshooting were all easy tasks. Prior to programming, I had already reworked the six examples contained within the lecture notes and had values to compare to the web application. As previously mentioned, all of the mathematical calculations and the programming for the overall process were both relatively simple. Few problems occurred and those that did were usually minor in nature.

I tried to avoid using a SQL database to store data values (I used CSVs instead) which created a larger problem. When programming the step that selects an appropriate Area Product or Core Geometry, I overlooked my original assumption that the values would be sorted for the column header I had specified. Because the available core tables were already sorted in order from smallest to largest (in both $A_p$ and $K_g$), this error did not occur until later when I was experimenting with adding additional cores. My current search algorithm would search for the column header that matched (either $A_p$ or $K_g$) and would then progress linearly down the list until it found a value that was larger than the value calculated. Unfortunately, PHP lacks the inherit ability to sort data in a CSV prior to executing a demand and I was unable to get an open-source CSV parser\(^2\) to work as required.

\(^2\) http://code.google.com/p/parsecsv-for-php/
My engineered fix for this problem was a sort_csv.php script, which reads the CSV and splits the headers (titles) and data up. It then locates the appropriate column header, pulls the data into an array and then sorts the array. Within the individual steps, I create a temporary file that inherits the contents of the referenced core CSV (which varies depending on the magnetic design selection), calls the sort_csv script and specifies the sort-and-search parameters (either Ap or Kg and ascending or descending) and then saves the file to a temporary directory. With this fix, I was still able to use my original code because the sort_csv script will always provide an ascending-ordered Ap or Kg column CSV array.

For reasons unknown to me, Prototip2 will not display any tooltips if you incorrectly reference a span class. Because I’m not extremely well versed in JavaScript, I did not attempt to debug the error however I did report it to the author. I was able to remedy the problem by adding the, previously uncreated, class.

Due to my overall inexperience with creating an administrative backend that allowed uploading new materials and editing current materials, some of my code is amateur and could be a small security concern. When one submits the form on the administrative backend, I have it writing new PHP code to the server, which apparently is not the best way to handle the action. Hackers could use SQL injection tactics because there is no check to ensure that the CSVs are formatted correctly. The CSV could contain illegal characters and could attempt to write to directories they aren’t supposed to which does represent a sizeable threat. Luckily, the chance of anyone attempting to hack the backend is infinitely miniscule.
VI. Conclusion and Recommendation

The Magnetic Design Web Application successfully provides a user with a replicated and automated step-by-step design process for power transformers, gapped inductors and toroidal inductors. It additionally provides an administrative backend for professors to add new and edit current core information and values. With the overall structure of the application, it lends itself to be easily scalable should future developers want to add onto the project.

If given the chance to start the project over from the beginning, I would have stored the core values in MySQL databases instead of CSVs. While the current design is extremely versatile and completely independent and self-contained, the exclusive use of CSV files forced me to create workaround methods for actions that would have been straight-forward if using SQL databases. I originally thought that having a self-contained design (the project doesn’t reference any outside variables, data or functions and requires no setup) would be easier, but this thought was extremely short-sighted. By the time I began to realize the benefits of using SQL databases, I was already too entrenched in the current code and would have had to scrap a lot of my current progress.

The use of SQL databases would have eliminated the security concern but it would have also required a server running MySQL and creation of such databases on every server the Magnetic Design Web Application is installed on.
Reference List

http://wiki.github.com/madrobby/scriptaculous/effect-slidedown


Appendix A. Source Code Samples

Sample code from /transformers_cores.php

```php
$core_form_options = array(  
  "pot_core" => "Pot Core",  
  "powder_core" => "Powder Core",  
  "lamination_core" => "Lamination",  
  "c_core" => "C Core",  
  "single_coil_core" => "Single-Coil",  
  "tape_wound_core" => "Tape-Wound Core"
);

class core
{
  var $name;
  var $Kj25;  // Kj at 25 deg C
  var $Kj50;  // Dj at 50 deg C
  var $x;     // x constant
  var $y;     // y constant
}
```

Sample code from /transformers_material.php

```php
class core_material
{
  // Description of core material
  var $description = "Enter a description for this core material";
  var $group = "Group name";  // Power Transformers, Gapped Inductors, Torodial Inductors

  var $coeff;
  var $F_pow;
  var $Bm_pow;
```
Entire code from /sort_csv.php

```php
<?php

/*
Method for sorting a CSV by a column as specified in the first row of
the CSV
returns the contents of the CSV file, sorted
*/

function sort_csv($filename, $column, $direction = "asc")
{
    $row = 0;
    $headers = array();
    $data = array();

    if (($handle = fopen($filename, "r")) !== FALSE)
    {
        while (($d = fgetcsv($handle, 1000, ",")) !== FALSE)
        {
            if($row == 0)
                $headers = $d;
            else
                $data[] = $d;
            $row++;
        }
    }
    fclose($handle);

    // Actually sort the CSV;
    $sorted_data = array();
    $sorting_column = false;
```
// Find the column that we want to sort on
$count = 0;
foreach($headers as $c)
{
    if(strtolower($column) == strtolower($c))
    {
        $sorting_column = $count;
        break;
    }
    $count++;
}

if(!$sorting_column)
    return false;

// Get the column of data that we need
$temp_data = array();
$count = 0;
foreach($data as $d)
{
    $temp_data[$count] = $d[$sorting_column];
    $count++;
}

// Sort the data
if($direction == "asc")
    asort($temp_data);
else // desc
    arsort($temp_data);

$sorted_data[] = $headers;
foreach($temp_data as $key => $value)
{
    $sorted_data[] = $data[$key];
}
$handle = fopen("php://temp", "r+"/
rewind($handle);

foreach($sorted_data as $line)
{
    fwrite($handle, implode("","$line");
    if(end($sorted_data) != $line)
        fwrite($handle, "\n");
}

rewind($handle);
$csv = "";

while (!feof($handle))
    $csv .= fgets($handle, 4096);

fclose($handle);

return $csv;
}

Sample code from /prototips.php

$tips = array(
    /* step 1 */
    "vs" => "Voltage on Secondary Side " . $this->s("V"),
    "vo" => "Output Voltage (given)",
    "vd" => "Diode Voltage (given)",
    "po" => "Output Power ". $this->s("Po"),
    "io" => "Output Current (given)",
);
Shortcut code for Prototip Step Numbers (in /transformer.php)

```php
function s($var_name)
{
    if(isset($this->calculated_values_steps[$var_name]) && $this->calculated_values_steps[$var_name] > 0)
        return " (Step ", $this->calculated_values_steps[$var_name].")";
    else
        return "";
}
```

Example transformer code (/transformers/power_transformer_area_product.php)

```php
<?php

class power_transformer_area_product extends transformer
{
    var $steps_required = array(

        "transformer_output_power", // 1
        "apparent_power", // 2
        "area_product", // 3
        "select_area_product_table", // 4
        "number_turns_primary", // 5
        "primary_current", // 6
        "current_density", // 7
        "bare_wire_size_primary", // 8
        "wire_size_primary", // 9
        "winding_resistance_primary", // 10
        "copper_loss_primary", // 11
        "number_turns_secondary", // 12
        "bare_wire_size_secondary", // 13
        "wire_size_secondary", // 14
        "winding_resistance_secondary", // 15
        "copper_loss_secondary", // 16
        "copper_loss_total", // 17
    )
```
"power_loss_combined",       // 18
"iron_loss",                 // 19
"core_loss",                 // 20
"wkg_core_graph_loss",       // 21
"psi",                       // 22
);

var $form_inputs = array(
    "Input Voltage",
    "Output Voltage",
    "Output Current",
    "Switching Frequency",
    "Efficiency",
    "Regulation",
    "Temperature Rise",
    "Flux Density",
    "Diode Voltage Drop",
    "Core Material",
    "Core Configuration",
    "Waveform",
);

function defaults()
{
    $this->Vin  = 28;
    $this->Vo   = 28;
    $this->Io   = 3;
    $this->Fs   = 20000;
    $this->EFFIC= .98;
    $this->reg  = 1;
    $this->TRISE= 25;
    $this->Bm   = .3;
    $this->CORE_CONFIG = "c_core";
    $this->CORE_MATERIAL = "permalloy_2_mil";
}
$this->Vd = 1;
$this->WAVEFORM = "squarewave";
}
}

Example step code (/steps/winding_resistance_ind.php)

<?php

// these are the PHP calculations

// Calculate winding resistance
// R = MLT * N * resistance * 10^-6

$this->R = $this->core_mlt * $this->N * $this->wire_resistance * pow(10,-6);

// dipset is purely a display variable, it allows us to round all answers to six decimal points while retaining all for further calculations (if necessary)

$this->dipset = round($this->R, 6);

// this is for the Prototip step-recall function
$calculations = array("R");

// this is similar as it gets the step number for the step description for the selected core’s mean-turn-length and wire resistance.
$core_mlt_step = $this->get_step_no("core_mlt");
$wire_resistance_step = $this->get_step_no("wire_resistance");

$title = "Calculate the winding resistance. Use MLT from step $core_mlt_step and micro-ohms per centimeter from step $wire_resistance_step.";

$output = <<<FINISH

// continued on next page
// this is the HTML styling

// the first line of code is contained in a table to create the µΩ/cm fraction in text form

<div class="equations">
<table class="frac_eq">
<tr>
<td rowspan="2"><span class="r_wind"><span class="varhigh">R</span></span> = <span class="mlt"><span class="varhigh">MLT</span></span> * <span class="n"><span class="varhigh">N</span></span> * 
</td>
<td><table style="display: inline;" width="100%">
<tr><td style="border-bottom: 1px solid #000;"><span class="wireres"><span class="varhigh">&mu;&Omega;</span></span></td></tr>
<tr><td>cm</td></tr>
</table></td>
<td rowspan="2">* 10<sup>-6</sup> [ &Omega; ]</td>
</tr>
<br />
R = $this->core_mlt * $this->N * $this->wire_resistance * 10<sup>-6</sup> [ &Omega; ]
</div>

<br />

<div style="margin-top: 10px;" class="eq_highlight">R = $this->dipset [ &Omega; ]</div>

</div>

FINISH;
?>
### Appendix B. Parts and Cost List

<table>
<thead>
<tr>
<th>Part</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAMP Hosting</td>
<td>N/A (Already Purchased)</td>
</tr>
<tr>
<td>Adobe Photoshop and Dreamweaver CS</td>
<td>N/A (Already Purchased)</td>
</tr>
<tr>
<td>Venera Night Sky Abstract Background Usage License</td>
<td>$2</td>
</tr>
<tr>
<td>Single-Use Non-Commercial Prototip 2 Usage License</td>
<td>$3.58 (€ 3)</td>
</tr>
<tr>
<td>Taufik Tip Illustration</td>
<td>$40</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>$45.58</strong></td>
</tr>
</tbody>
</table>
Appendix C. Analysis of Senior Project Design

Summary of Functional Requirements
Successfully automates the magnetic design processes for Power Transformers, Gapped Inductors and Toroidal Inductors using either Area Product or Core Geometry calculation methods via an online application. Also provided is an administrative backend where instructors can add or modify core design values and settings.

It aims to aid student understanding of course concepts and allow instructors to broaden the scope of material covered within EE 433 and any subsequent magnetic design courses.

Primary Constraints
The primary constraint was my original insistence to avoid using a MySQL database to store core values, instead favoring the use of multiple CSV files. This required additional research on my part for the creation of the administration backend. Aside from this, most problems experienced were minor hiccups that were quickly and easily resolved. I proposed a project that was within my capabilities, strictly adhered to the timeline and fully completed my senior project with no “limiting factors” or compromises.

Economic
Already leasing a LAMP (Linux, Apache, MySQL and PHP) server for non-senior project purposes, there are no server costs because of the small footprint of the Magnetic Design Web Application. I, also, own a web-development application (Dreamweaver) and an image editor (Photoshop), reducing all major web-development costs to zero. If I did not already have access to these services and utilities, there are comparable (and free) programs available; the yearly server cost would be approximately $50.

The usage license for the background isn’t necessary to the performance of the application and its cost can be avoided by choosing a free background. The usage license for Prototip2 could also be avoided by using the HTML tag, <acronym>; however, both Prototip2 and the background chosen are a bit more visually appealing and come at a minimal cost.

Environmental / Manufacturability / Ethical / Health and Safety / Social and Political
None of these topics are applicable to my project. There are no environmental, manufacturing, ethical, health and safety, or social and political problems associated with an online magnetic design website.
Sustainability
One of the major issues addressed during the second half of the project was the allowance for continued development of the web application to be as easy and straight-forward as possible. The file system is very hierarchical (e.g. /steps/ folder contains all the steps, /transformers/ contains all six transformer and inductor files). Contained within each step file is both the PHP code for the calculations and the HTML code for the presentation. Each transformer file is simply an ordered list of steps and inputs required to generate the desired output. (See examples in Appendix B). All code is excessively commented which should allow someone completely unfamiliar with PHP programming to be able to understand what each section of code accomplishes.

The administrative backend allows easy access to current CSV-core assignment pairings, meaning professors can easily amend the CSVs to include additional core values. Also provided is the ability to edit current core configuration and core loss constants and add completely new cores.

The only suggested improvement to the current design would be the implementation and usage of a MySQL database to store all the aforementioned values instead of CSVs. This proposed improvement would eliminate the need to download and upload CSVs; with MySQL one would be allowed to live-edit the data tables. The only downside to this improvement is that if one moved the web application to a different server, there would be a longer set-up period required (to create the MySQL database and port over all the existing values).

Development
Going into the project, I had a rudimentary understanding of the PHP language. I don’t feel that I necessarily broadened the scope of my PHP knowledge, but rather solidified my current understanding of the limited code required for this senior project. I was able to experiment with existing JavaScript coding snippets (Prototip2 and Scriptaculous), but mostly relied on using the already established, open-source code.