**Developer Note for TransformerApp v0.1.0**

This is the developer note for TransformerApp v0.1.0, which includes formulae derivation, theory background, project architecture, etc. For a simple guide on how to use the app, please refer to User Manual.

The entire project is divided as three parts: circuit compiler, turns designer, and wire diameter designer. This document will first introduce the notations used, and later go through the aforementioned compartments of the project.

**I. Notations**

The following table presents the notations and their meaning throughout this document.

**Clarifications:**

* Let index denote the primary, and denote the secondary (output).
* Auxiliary winding is treated as a normal output.
* In this document, natural numbers does not include 0. For non-negative integers, is used.

|  |  |  |
| --- | --- | --- |
| **Symbol** | **meaning** | **unit** |
|  | minimum AC input rms voltage | V |
|  | maximum AC input rms voltage | V |
|  | minimum DC input voltage | V |
|  | maximum DC input voltage | V |
|  | the output voltage the output channel, | V |
|  | Minimum input voltage to transformer | V |
|  | Maximum input voltage to transformer | V |
|  | Reflected voltage to the primary side | V |
|  | The forward voltage of the diode at the nth channel, | V |
|  | Switching frequency of the MOSFET | Hz |
|  | AC input frequency | Hz |
|  | the output current of the output channel, | A |
|  | The middle point of the slope at primary current | A |
|  | The current ripple at primary side | A |
|  | The peak current flowing through the switch | A |
|  | The pulse by pulse current limit level | A |
|  | The RMS current flowing through the winding, | A |
|  | The current density of the winding, |  |
|  | the output power of the channel, | W |
|  | Total output power | W |
|  | Total input power | W |
|  | The load occupying factor of the output channel, |  |
|  | Efficiency |  |
|  | The maximum duty ratio |  |
|  | Magnetizing inductance | H |
|  | Saturation flux density of the core material | T |
|  | Cross sectional area of the core |  |
|  | The winding window area of the core |  |
|  | The cross sectional area of the wire of the channel, |  |
|  | The required winding window area |  |
|  | The width of the winding window of the core | m |
|  | The height of the winding window of the core | m |
|  | The required height for the total winding | m |
|  | The diameter of the wire of the winding, | m |
|  | The gap length of the core | m |
|  | The minimum number of primary turns |  |
|  | The number of turns of the channel, |  |
|  | Fill factor of the windings |  |

**II. Circuit Compiler**

The circuit compiler compiles circuit related parameters and produces input parameters for the transformer design. The current version supports two topologies.

1. Formulae
2. Voltage compilation

The application is able to handle both DC and AC input. We have the following expressions:

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-1) |
|  |  |  |
|  |  | (Eq. 2-2) |

For AC input, the root mean square value is provided. To get the peak value, we multiply the rms value by . As for the constant 0.8 in , it is an approximation of the lowest value in the voltage waveform after the input capacitor.

To perform a numerical calculation, we should also be given the input capacitance and the duty ratio during which the capacitor is being charged, denoted . If not given, is commonly chosen as for universal input and for European. A common figure for is 0.2.

With the given parameters, we can derive the minimum voltage for AC input by considering the energy conservation between the input power and the energy discharged by the capacitor:

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-3a) |

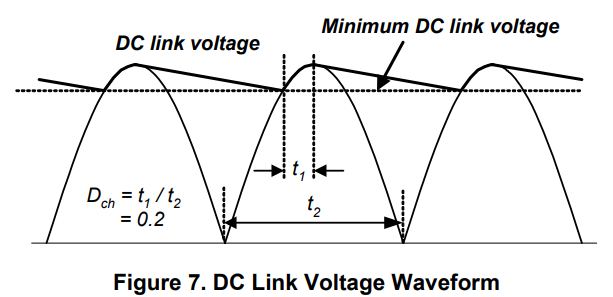
Use for the peak voltage at minimum AC input, and 2 in the denominator since is defined based on twice the line frequency, shown as follow:

Figure1: The DC link voltage waveform at input. (Reference: Application Note AN-4150, Design Guidelines for Flyback Converters Using FSQ-series Fairchild Power Switch (FPS™))

After simplifying Eq. 2-3a, we obtain

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-3) |

However, due to the fact that and are not always available and simplicity considerations, Eq. 2-3 is not used for now. The program includes only Eq. 2-2, and can be found in utils\formulae.py, function voltage\_compiler().

1. Power

The power related parameters are calculated by the following expressions:

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-4) |
|  |  | (Eq. 2-5) |
|  |  | (Eq. 2-6) |
|  |  | (Eq. 2-7) |

They are calculated explicitly in circuit\{topology}.py, which can and should be refactored.

1. Duty ratio, reflected voltage and turns ratio

The reflected voltage is defined as

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-8) |

where is the number of primary turn, is the number of secondary turns, and is the voltage across the secondary turn.

1. Flyback converter

For flyback converters, we have the following fundamental equation:

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-9) |

where is the voltage across the primary turn and is the duty ratio.

1. Forward converter

For forward converters, we have the following fundamental equation:

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-10) |

where is the voltage across the primary turn and is the duty ratio.

By rearranging these equations, we get the formulae to derive the other two with one specified. However, since we didn’t prohibit the user to provide multiple of them, conflicts emerge. The current logic is to override the parameter(s) with lower priority. The priority is in the following order: duty ratio, turns ratio, reflected voltage.

The calculation formulae and logic can be found in function d\_n\_vro() in utils\formulae.py, with three returning parameters in the order as the identifier suggests.

1. Peak primary current and delta I at primary side

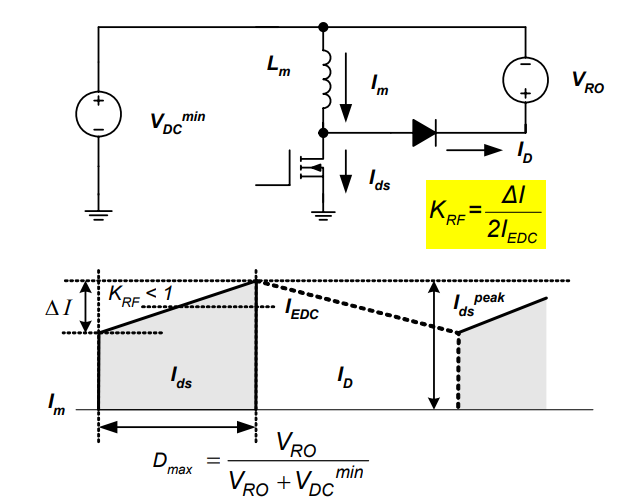
A visual aid for the current waveform is provided below. Since we only focus on the current at primary winding, we can use this figure for flyback converter w.l.o.g. Please note that the notations differ slightly from those used in this document.

Figure2: The current waveform and visualization. (Reference: Application Note AN-4150, Design Guidelines for Flyback Converters Using FSQ-series Fairchild Power Switch (FPS™))

Let denote the average input current. We have

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-11a) |

and

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-11b) |

which allows us to obtain

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-11) |

Also, the fundamental voltage-current relationship for an inductor is given by

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-12a) |

By substituting and with and , we obtain

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-12) |

The peak current can then be obtained:

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-13) |

These formulae can all be found in utils\formulae.py.

1. Magnetizing Inductance

Magnetizing inductance is only calculated in flyback converter in BCM operation. In other cases, this value is required to be given by the user.

Under BCM operation, the current happens to drop back to 0 when a new cycle starts. That is,

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-14a) |

Combining this constraint with Eq. 2-12, we can get

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-14) |

This expression can be found in circuit\flyback.py.

1. Critical Output Inductance

The critical output inductance is a derived value only for forward converter under BCM operation. Under BCM, we also have expression similar to Eq. 2-14a except that is now the output current. Focusing at the off time, we get

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-15a) |

Substituting with and include forward voltage of the diode, we get

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 2-15) |

Where is the minimum duty ratio, which is computed through Eq. 2-10 with . Please be aware that this expression assumes that the load does not fall under the specified output current for simplicity. The equation can be found in circuit\forward.py.

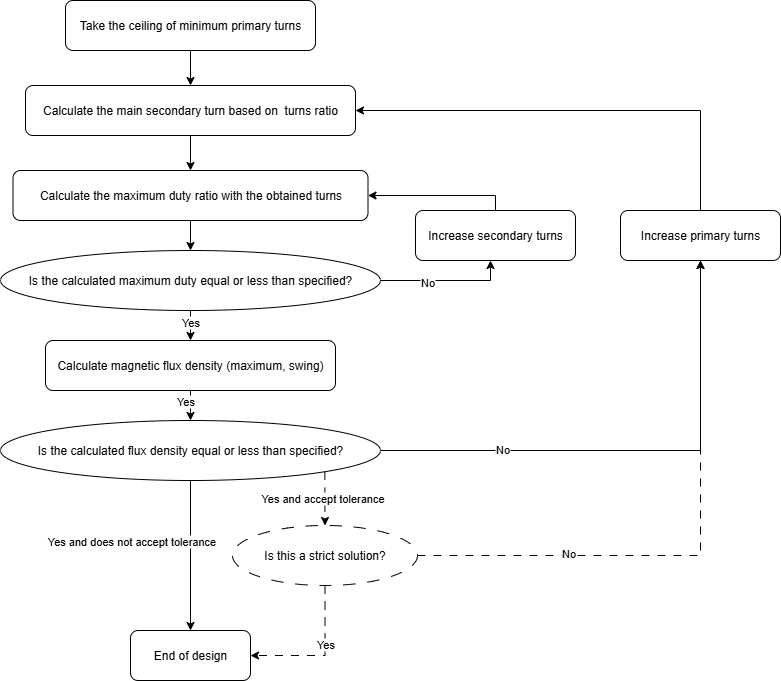
1. Code

In version 0.1.0, the converters are constructed with separate classes in separate files. They can be found in circuit\ and flyback.py, forward.py respectively. These classes include method compile\_params() that achieve the aforementioned behavior. The instances are constructed when the user clicks on the compile circuit button.

The developer has considered several architectures and falls on the current one. However, a single class with topology as a parameter/attribute, inheriting parent class are also possible solutions.

**III. Turns Designer**

The turns designer finds the appropriate design regarding the number of turns in an iterative process.

1. Flow Chart
2. Formulae
3. Minimum primary turns

Let λ denote the flux linkage, L denote the inductance, I denote the current flowing through, N denote the number of turns, B denote the flux density, A denote the effective area, V denote the voltage across a winding, and denote the time when V is applied on the winding. We have

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-1a) |
|  |  | (Eq. 3-1b) |

In differential form, we can write

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-2a) |
|  |  | (Eq. 3-2b) |
|  |  | (Eq. 3-2c) |

At primary side, we plug in the magnetizing inductance , maximum current , saturation flux density , minimum number of primary turns and core area for Eq. 3-1a and 3-1b. Equating the RHS, we yield

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-1d) |

Hence, we obtain the expression for minimum primary turns bound by maximum current is given as:

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-1) |

In some cases (e.g. high frequency), the flux density swing ( is a stricter constraint due to core loss consideration. By equating the RHS of Eq. 3-2a, 3-2b, and 3-2c, we get

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-2) |
|  |  | (Eq. 3-3) |

We take the largest value among Eq. 3-1, 3-2 and 3-3.

Since the real world does not accept non-integer number of turns, we take the ceiling of this value. This is written in transformer\tf\_draft.py with utility functions in utils\formulae.py.

1. Calculate number of main secondary winding

It is initially calculated as

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-4) |

where denotes rounding the number to the nearest integer.

1. Checking for maximum duty ratio

Due to the fact that number of turns can only be integers, our delicate turns ratio is usually not maintained. In this step, we check if the turns ratio calculated exceeds the limit we set. The duty ratio calculated can be obtained by Eq. 2-9 and 2-10 for different topologies. If violation occurs, the turns ratio is too low to keep the duty ratio down. Therefore, we increase the secondary turns by 1 until the duty ratio criteria is satisfied.

1. Checking for magnetic flux density

With the alteration of turns ratio and duty ratio, the magnetic flux density should be recalculated. We obtain the maximum current and delta I through Eq. 2-11, 2-12 and 2-13 with the value we have now and plug in Eq. 3-1, 3-2, and 3-3 to check if the limit is exceeded. If any of them fails to keep under, we increase the primary turn by 1. This is written in transformer\tf\_draft.py with utility functions in utils\formulae.py.

The iterative process above can be found in transformer\tf\_draft.py, method update\_draft\_windings() in class TransformerDraft.

1. Gap length calculation

Let denote the magnetic flux, ℱ denote the magnetomotive force, denote the number of turns wound on the magnetic core, denote the current flowing through the winding, denote the flux linkage, denote the length of the magnetic path of the core, denote the permeability of the magnetic core, and denote the air gap, core, and total reluctance respectively. Some basic magnetic circuit equations are given as follow:

|  |  |
| --- | --- |
|  | (Eq. 3-5a) |
|  | (Eq. 3-5b) |
|  | (Eq. 3-5c) |

where

|  |  |
| --- | --- |
|  | (Eq. 3-5d) |
|  | (Eq. 3-5e) |
|  | (Eq. 3-5f) |

As seen in Eq. 3-5e and 3-5f, high material permeability lead to low reluctance, i.e. ; hence, . Using Eqs. 3-5a to 3-5f and the approximation mentioned, we obtain

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-5) |

This is written as a utility function calculate\_gap() in utils\formulae.py.

1. RMS current calculation

For the primary RMS current, please only consider the shadowed part of the current waveform in figure 2. The initial current at the start of the interval is , and the slope is given by , where is the switching period. The current waveform can be formulated as

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-6a) |

By applying the definition of RMS

|  |  |
| --- | --- |
|  | (Eq. 3-6b) |

after calculation, we get

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-6) |

Please note that Eq. 3-6 still holds for DCM (although not discussed in the project for now), where

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-6c) |

1. As for the secondary RMS current, we have to discuss with respect to different topologies.
2. Flyback

From the current relationship between the primary windings and secondary windings:

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-7a) |

where we denote the peak current of the winding as . If we denote as the minimum current flowing through the windings, we can also get a similar equation:

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-7b) |

For simplicity, we denote as here. The maximum and minimum currents on the primary and secondary sides are related by a scaling factor . Since the current waveforms on both sides increase/decrease linearly, the secondary-side current can be regarded as a time-inverted, time-scaled, and magnitude-scaled version of the primary-side current. Specifically, if represents the primary current waveform, the secondary waveform can be expressed as . This allows us to directly relate their RMS values using the known RMS relationship for such signal transformations.

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-7c) |

We denote as the root mean square value of the waveform over one period. We have and , yielding

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-7) |

This expression can be found in utils\formulae.py, function calculate\_irms\_with\_ref().

1. Forward

It is simpler for forward converters. They have current waveform of the same shape but differ in magnitude by a factor of as noted in the flyback section.

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 3-8) |

This equation also belongs to calculate\_irms\_with\_ref() in utils\formulae.py.

1. Code

The above-mentioned design flow is implemented in function design\_turns() from app\transformer.py. The method creates a draft with user-submitted specification/options and go through the process shown in the flow chart. The program will only update RMS current for each winding after a solution is selected.

**IV. Wire Diameter Designer**

The wire diameter designer finds the optimal solution to selecting wire diameter by formulating practical concerns into an optimization problem. There can be many approaches to achieve our goal. In v1.0.0, we solve the problem with the help of convex optimization.

1. Optimization problem
2. Notations

Let the circuit include output channels.

1. , where denotes the RMS current flowing through the channel

, where denotes the current density limit of the channel.

1. , where denotes the wire diameter of the channel
2. , where denotes the number of windings for the channel
3. , where denotes the number of strands of the windings for the channel
4. , where denotes the layer needed for a single winding for the channel
5. : the bobbin winding width
6. : the bobbin winding height
7. : the thickness of the insulator around the wire
8. : the thickness of the Mylar tape
9. : the number of layers of the Mylar tape
10. : the maximum ratio that the wires can occupy in the winding width
11. : the maximum ratio that the stack can occupy in the winding height

Moreover, we let for variables without subscripts. Also, element-wise multiplication and division are written directly as in normal algebra style. That is, if and . (Although it is not a good habit, I’ll write them this way for simplicity here.) Matrix multiplications are specified explicitly.

1. Main Constraints
2. The current density of the channel should not exceed .
3. For each layer, the space occupied should not exceed of the bobbin winding width.
4. The stacked height should not exceed of the bobbin winding height.
5. Preliminary

The channel includes of total strands; hence, each strand has of RMS current flowing through. The minimum wire diameter allowed (calculated with the maximum allowed current density ) is . This is the best case where the wire diameter is the smallest. The minimum width that a winding need is . We determine how many layers this specific winding needs by applying constraint 2 on the best case. Hence, the number of layers that a single winding from the channel needed is given by

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 4-1) |

1. Main problem
2. Rules

A convex optimization problem has the form

where the objective and constraint functions are convex:

For all and all , with

If this form is satisfied, we can solve the problem easily with the help of the mature and well-established convex optimization framework and a powerful package cvxpy.

1. Formulation

i. Optimization variable

The optimization variable of this problem is . (Only in the current version. If the optimization problem is more complicated, it can definitely be more than this.)

ii. Objective function

Suppose that our target is to fill the wire of the primary winding into the bobbin width as much as we can **(which is very rough and needs improvement).** Our objective function is

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 4-2) |

iii. Constraint functions

The functions are listed below with accordance to the constraints listed in b.

1. Current Density

The minimum wire diameter is given by as in c. Hence, we obtain

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 4-2a) |

Assume , we obtain

|  |  |
| --- | --- |
|  | (Eq. 4-2) |

1. Winding width

The winding width needed for a single layer of a winding in channel is , where can be seen as “turns per layer.” By constraint 2, we have inequality:

|  |  |
| --- | --- |
|  | (Eq. 4-3a) |

Denote , we get

|  |  |
| --- | --- |
|  | (Eq. 4-3) |

Note that the ceiling operator is applied element-wise to the vector inside.

1. Winding height

The total winding height should take in both the thickness of the insulator coated around the wire and the Mylar tape. The expression is given as . This value should be smaller than . Giving the inequality:

|  |  |
| --- | --- |
|  | (Eq. 4-4a) |

Assume , we obtain

|  |  |
| --- | --- |
|  | (Eq. 4-4) |

are all linear functions, which are affine and thus convex. Eqs. 4-2 to 4-4 are rearranged to match the problem form written in 1, but we can write it in less strict format as the following section and in python.

iv. Summary

We have linear objective and constraint functions. Linear functions are affine and therefore both convex and concave, making it a valid convex optimization problem.

With Eqs. 4-1 to 4-4, the final problem is

In fact, it is a linear programming problem. However, due to extensibility concerns, cvxpy is still used to solve the problem. Please refer to \bobbin\ector.py for implementation.

1. Implementing with wire catalog

In reality, the wire diameters are not arbitrary real numbers, but limited to several possibilities. To incorporate this situation, we transform the problem into a mixed integer programming problem.

1. Additional variables

i. Diameter catalog

Suppose we have available wire diameter, we set a catalog vector of length with each entry being a specific wire diameter.

ii. Optimization variable

We set the optimization variable to a Boolean matrix of shape , denoted as . is set to 1 if and only if winding uses the diameter locating at index in the catalog vector.

Therefore, the expression of the diameter of the channel is (matrix multiplication). By substituting above with , we get our optimization problem.

1. Additional constraint

The summation of the entries in each row must equal to 1.

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 4-5) |

where is a column vector of ones of length .

Please note that although this method is set as default in the application and it is closer to the real world, there are more constraints with built-in solvers in cvxpy. Please refer to the documentation of cvxpy for its abilities and limits.

1. KF method

This method only provides a primal guess on whether the winding can fit the core or not. We calculate the required window area as

|  |  |
| --- | --- |
|  | (Eq. 4-6) |

where can be obtained by

|  |  |  |
| --- | --- | --- |
|  |  | (Eq. 4-7) |

Please notice that in this method, current density is actually applied to the calculated number as opposed to the previous method that it serves as a constraint instead of an applied value.

1. Code

All the optimization related scripts can be found in \bobbin. The optimization part are implemented in ector.py (called ector\_continuous and ector\_discrete throughout the application) and KF method in kf\_method.py.

**V. Code Related**

This section explains the architecture and how the data are passed around throughout the project.

1. Architecture

\app # UI, application logic, state management, workspace

\bobbin # wire diameter related backend logic

\circuit # circuit topology classes and methods

\data # database (core repository) related helper

\transformer # turns design related backend logic and physical model classes

\utils # constants, formulae, tooltips, etc.

1. Design State

An object of class DesignState is passed around in the application. It stores specifications, cores and repository, compiled circuit, (selected) solutions to allow the state of the design to be passed around tabs.

**VI. Miscellaneous**

Contact Daniel Wang if you encounter any question. Email: [wcpwcpd@gmail.com](mailto:wcpwcpd@gmail.com)

Changelog:

2025.08.08 First edition done