

ScanTesla

V-1.10

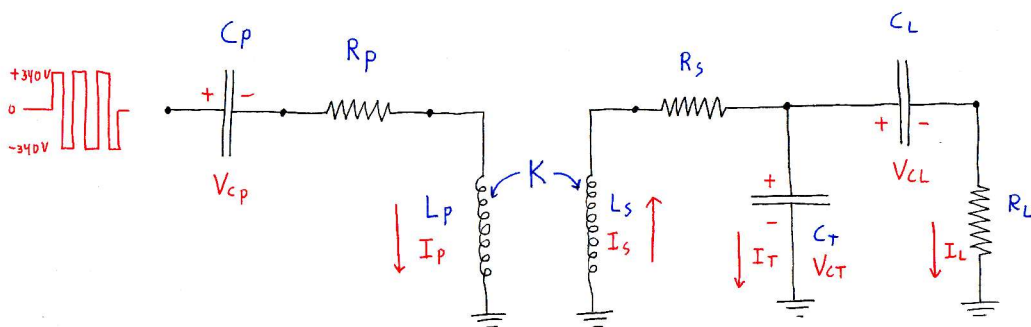
A program to scan Tesla coil parameters to find the best output.

Terry Fritz May 22, 2005

ScanTesla is a program that iterates through and tests a large number of Tesla coil parameters in search of those parameters that give the best output sparks.

The program normally is intended for the new DRSSTC type of Tesla coil, but minor changes to the inputs will allow typical disruptive coils to be studied too.

The Tesla coil is assumed to match the following model:



This is a standard Tesla coil model with a DRSSTC style input. Various equations are given which define how the circuit elements are applied in the program. C_L and R_L are the typical 1pF/foot +220k Ohm streamer load elements (these may improve over time). In the case of a conventional coil, the input is set to zero and node1 is set to the initial C_p voltage. With a very long T_1 time, CW coils could be directly modeled as well.

The program inputs are as follows:

Cp	start, stop, and increment
Rp	start, stop, and increment
Lp	start, stop, and increment
Ls	start, stop, and increment
K	start, stop, and increment
Rs	start, stop, and increment
Ct	start, stop, and increment
Cl	start, stop, and increment
Rl	start, stop, and increment
T1	start, stop, and increment
Vin	DRSSTC square wave voltage
Vn1	initial condition on Cp for disruptive coil case.

The program will scan the values as defined above. Normally, most of the variables will be fixed and only a few actually scanned.

The program outputs are as follows:

WRI	Power per T1 to load Rl
WRp	Power per T1 to Rp
WRs	Power per T1 to Rs
Win	Power per T1 to coil (WRI+WRp+WRs)
ICp-peak	Peak Cp current
VCp-peak	Peak Cp voltage
VCt-peak	Peak Ct voltage

Antonio tells us ("ScanTesla program -> Optimization", May 19, 2005):

Formulate a set of state equations with the form:
$$dX/dt = [A]X(t) + B \cdot vin(t)$$
where X is a vector with 5 elements
(Vc1, Vc2, Vc2, Il1, Il2), [A] is a
5x5 real matrix, and B is a vector with 5 elements. This
is quite easy.
Solve it by numerical integration with the trapezoidal
rule:
$$X(t_0+dt) = X(t_0) + (dt/2) * ([A] * X(t_0) + B * vin(t_0) + [A] * X(t_0+dt) + B * vin(t_0+dt))$$
where t0 is the present time, dt is a small time interval
ahead, X(t0)
is the present state, and X(t0+dt) is the next state.
The required calculation is:
$$X(t_0+dt) = [[I] - (dt/2) * [A]]^{-1} * (dt/2) * ([A] * X(t_0) + B * (vin(t_0) + vin(t_0+dt)))$$
The matrix inversion has to be done just once.
This results in good precision if you put about 50 "dt"
for each cycle.

Formulate a set of state equations with the form:
 $\frac{d\mathbf{X}}{dt} = [\mathbf{A}]\mathbf{X}(t) + \mathbf{B} \cdot v_{in}(t)$

We setup the equation with $a(x,y)$ and $b(x)$ being unknown for now:

$$\begin{array}{l} dV_{Cp}/dt \quad a_{11} \ a_{12} \ a_{13} \ a_{14} \ a_{15} \quad V_{Cp}(t) \quad b_1 \\ dV_{Ct}/dt \quad a_{21} \ a_{22} \ a_{23} \ a_{24} \ a_{25} \quad V_{Ct}(t) \quad b_2 \\ dV_{Cl}/dt = a_{31} \ a_{32} \ a_{33} \ a_{34} \ a_{35} \ x \ V_{Cl}(t) + \quad b_3 \ x \ V_{in}(t) \\ dI_p/dt \quad a_{41} \ a_{42} \ a_{43} \ a_{44} \ a_{45} \quad I_p(t) \quad b_4 \\ dI_s/dt \quad a_{51} \ a_{52} \ a_{53} \ a_{54} \ a_{55} \quad I_s(t) \quad b_5 \end{array}$$

The matrix \mathbf{X} is:

V_{Cp}
 V_{Ct}
 V_{Cl}
 I_p
 I_s

From the model:

$$\begin{array}{l} dV_{Cp}/dt = 1/C_p \ x \ I_p \\ dV_{Ct}/dt = 1/C_t \ x \ (I_s - (V_{Ct} - V_{Cl}) / R_I) \\ dV_{Cl}/dt = 1/C_l \ x \ (V_{Ct} - V_{Cl}) / R_I \\ dI_p/dt = 1/L_p \ x \ (V_{in} - V_{Cp} - I_p \ x \ R_p - K \ x \ I_s) \\ dI_s/dt = 1/L_s \ x \ (-V_{Ct} - I_s \ x \ R_s - K \ x \ I_p) \end{array}$$

Thus, the matrix \mathbf{A} is:

$$\begin{array}{ccccc} 0 & 0 & 0 & 1/C_p & 0 \\ 0 & 1/(C_t R_I) & -1/(C_t R_I) & 0 & 1/C_t \\ 0 & 1/(C_l R_I) & -1/(C_l R_I) & 0 & 0 \\ -1/L_p & 0 & 0 & -R_p/L_p & -K/L_p \\ 0 & -1/L_s & 0 & -K/L_s & -R_s/L_s \end{array}$$

The matrix \mathbf{B} is:

0
0
0
1/ I_p
0

The required calculation is:

$$\mathbf{X}(t_0 + dt) = [[\mathbf{I}] - (dt/2) * [\mathbf{A}]]^{(-1)} * (dt/2) * ([\mathbf{A}] * \mathbf{X}(t_0) + \mathbf{B} * (V_{in}(t_0) + V_{in}(t_0 + dt)))$$

The program will be written in C with LCC (<http://www.cs.virginia.edu/~lcc-win32/>). The following rules will apply:

1. It will be very easy to modify and recompile so anyone can add improvements or special functions over time as they wish and as the art improves.
2. No special libraries or complex programming will be used to keep it simple, understandable, and versatile.
3. The program should be general enough to be compiled with just about any C compiler under any operating system.
4. To keep it simple and easy, No GUI is supported. Input and output is typically done through direct input/output or data files to be analyzed with say spreadsheet programs.
5. There will be no copyright, GPL, copyleft, trademarks, or real owner for the Public Domain base version. Anyone can copy, steal, plagiarize, change, or use it as they wish. People may want to sell, copyright or do whatever with improved versions (say with an added GUI) but that is their problem, not ours.