ScanTesla

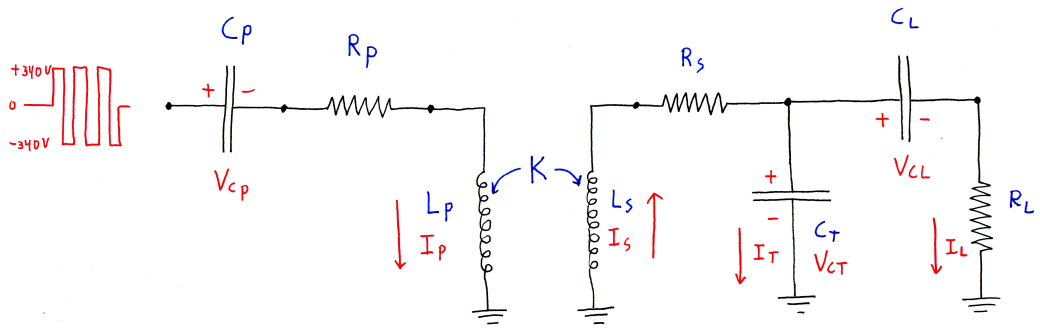
A program to scan Tesla coil parameters to find the best output.

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. Cl and Rl 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 Cp voltage. With a very long T1 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:

WRl	Power per T1 to load Rl
WRp	Power per T1 to Rp
WRs	Power per T1 to Rs
Win	Power per T1 to coil (WRl+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*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(t0+dt)=X(t0)+(dt/2)*([A]*X(t0)+B*vin(t0)+[A]*X(t0+dt)+B
*vin(t0+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(t0+dt)=[[I]-(dt/2)*[A]]^(-
1)*(dt/2)*([A]*X(t0)+B*(vin(t0)+vin(t0+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:
dX/dt = [A]X(t)+B*vin(t)

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

dVCp/dt		a11	a12	a13	a14	a15	VCp(t)	b1	
dVCt/dt		a21	a22	a23	a24	a25	VCt(t)	b2	
dVCl/dt	=	a31	a32	a33	a34	a35	x VCl(t) +	b3	x Vin(t)
dIp/dt		a41	a42	a43	a44	a45	Ip(t)	b4	
dIs/dt		a51	a52	a53	a54	a55	Is(t)	b5	

The matrix **X** is:

VCp
VCt
VCl
Ip
Is

From the model:

dVCp/dt = 1/Cp x Ip
dVCt/dt = 1/Ct x (Is - (VCt - VCl) / Rl))
dVCL/dt = 1/Cl x (VCt - VCl) / Rl
dIp/dt = 1/Lp x (Vin - VCp - Ip x Rp - K x Is)
dIs/dt = 1/Ls x (-VCt - Is x Rs - K x Ip)

Thus, the matrix **A** is:

0	0	0	1/Cp	0
0	1/(CtxRl)	-1/(CtxRl)	0	1/Ct
0	1/(ClxRl)	-1/(ClxRl)	0	0
-1/Lp	0	0	-Rp/Lp	-K/Lp
0	-1/Ls	0	-K/Ls	-Rs/Ls

The matrix **B** is:

0
0
0
1/Ip
0

The required calculation is:
X(t0+dt)=[[I]-(dt/2)*[A]]^(-
1)*(dt/2)*([A]*X(t0)+B*(Vin(t0)+Vin(t0+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. The 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.