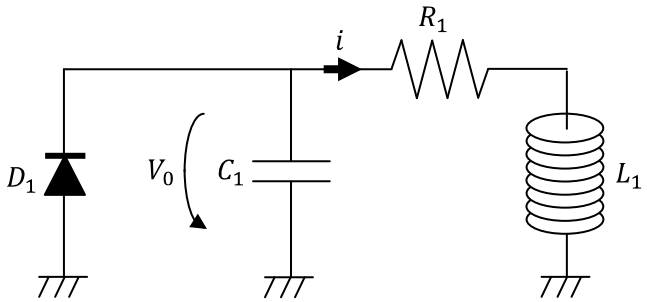
Placing bypass diodes across the capacitors

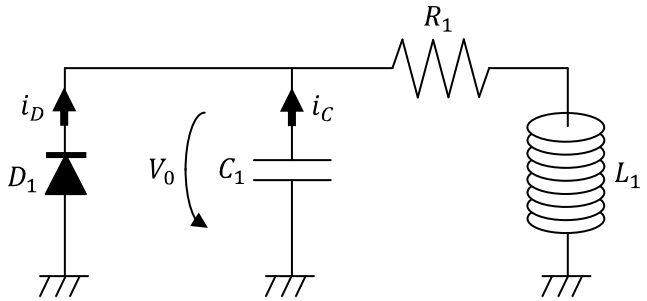
The following snippet of a schematic shows the modification I will examine in this paper. This is the circuit for a single barrel coil, say, barrel coil #1. The SCR which completes the circuit is assumed to close at time $t = 0$. The armature and other barrel coils are being ignored entirely. With one exception, this is the same L-R-C circuit I looked at in the previous paper, titled *Selecting barrel coils for an induction coil gun*. The exception is the diode D_1 now wired across the capacitor.



As before, the circuit starts up from rest, powered by the initial voltage V_0 over the capacitor. Being reverse biased, diode D_1 does not do anything during at first. The capacitor forces current i through the barrel coil, whose self-inductance is L_1 and whose Ohmic resistance constitutes almost all of the series resistance R_1 in the circuit. The current is the aggregate motion of electrons escaping from the capacitor.

Depending on the relative values of L_1, R_1 and C_1, the circuit (without the diode) exhibits two types of behaviour. In one type (the "non-oscillatory" response), the current rises exponentially up to some peak value and then decays exponentially back down to zero. If that happens, diode D_1 never does anything at all. The other type of response (called the "oscillatory" response), is a sinusoid, with a twist. The twist is that the amplitude of the sinusoid first rises exponentially and then decays exponentially. Except when the component values are pretty close to those of a non-oscillatory case, the sinusoidal nature of current i means that it will cycle back and forth between positive and negative values. This corresponds to energy flowing from the capacitor to the inductor (when i is positive) or flowing from the inductor back into the capacitor (when i is algebraically negative). There is nothing unusual about that. The capacitor stores energy in an electrostatic field which is generated by electric charges stored on its internal plates. The inductor stores energy in an magnetostatic field which is generated by electric current flowing through the turns in the coil. Energy can be converted from one type to the other, and it is the flow of current from one device to the other which carries the energy. Resistance R_1 uses up some of the energy as current flows through it, in either direction. The resistor grabs some of the electrical energy from the passing current, and converts it into another form of energy which is useless in this application: heat.

The addition of the diode completely changes the oscillatory response. It does two things. It prevents the capacitor from discharging more than its original charge. And it prevents the capacitor from accepting any current the inductor might wish to send back to it. Since there are now two pathways for the current, it is necessary to identify two different currents, i_C and i_D, as is done here.



For the time being, I am going to assume that diode D_1 is "ideal". When it is reverse-biased, it completely blocks the flow of current in the reverse direction (from top to bottom in the schematic). When it is forward-biased, it offers no resistance to the flow of current in the forward direction. Real diodes are not ideal, but the imperfection is small enough that it is not worth taking into account here. In the section below extending the concept to multiple barrel coils, I will include a small voltage drop that captures most of the non-ideality.

The current which flows through resistance R_1 and inductance L_1 is the sum $i_D + i_C$. I am using subscripts "D" and "C" for the currents flowing through the diode branch and the capacitor branch, respectively, rather than "1" and "2", for example, to ensure that there is never any confusion between currents flowing in barrel coil #1 and barrel coil #2.

It happens, though, that currents i_D and i_C never flow together. Current flows through the diode, or through the capacitor, but not both. It is either one branch or the other. That means that the circuit will have two different modes of operation. I will call the two modes "reverse-biased" and "forward-biased", distinguished by the voltage drop over the diode.

When the diode is reverse-biased, the capacitor has a non-zero positive voltage. In fact, it is the positive voltage drop over the capacitor which keeps the diode reverse-biased. In this mode, no current flows through the diode and the circuit equation is exactly the same as in the earlier paper.

$$D_1 \text{ reverse-biased:} \quad V_0 - \frac{1}{C_1}\int i_C dt = R_1 i_C + L_1 \frac{di_C}{dt} \quad (1A)$$

When the diode is forward-biased, the voltage drop over the capacitor will be zero. The current prefers to flow through the diode, which offers no resistance, than to flow "uphill" over the energy barrier which stored charge in the capacitor would represent. (A real diode does not have zero resistance, of course, but the resistance is small enough that the total voltage drop is only a volt or two.) Since there is no voltage drop over the diode or the capacitor, the inductance and resistance are the only meaningful components. The circuit equation reduces to

$$D_1 \text{ forward-biased:} \quad 0 = R_1 i_D + L_1 \frac{di_D}{dt} \quad (1B)$$

The two circuit equations apply over different periods of time, with Equation $(1A)$ governing the behaviour when the circuit first starts up. As was done in the earlier papers, a two-pronged attack can be made on the first differential equation. First a variable substitution using the definition of current i_C as the negative rate-of-change of the charge q stored in the capacitor. And then a differentiation. After a bit of re-arrangement, it becomes:

$$D_1 \text{ reverse-biased:} \quad L_1 \frac{d^2q}{dt^2} + R_1 \frac{dq}{dt} + \frac{q}{C_1} = 0$$

$$\rightarrow \quad \tau_{R_1L_1} \frac{d^2q}{dt^2} + \frac{dq}{dt} + \frac{q}{\tau_{R_1C_1}} = 0 \quad (2)$$

I have taken the liberty of substituting the two time constants: $\tau_{R_1L_1} = L_1/R_1$ and $\tau_{R_1C_1} = R_1C_1$. The characteristic equation and its two roots are:

$$\tau_{R_1L_1}\alpha^2 + \alpha + \frac{q}{\tau_{R_1C_1}} = 0$$

$$\rightarrow \quad \alpha = \frac{-1 \pm \sqrt{1 - \dfrac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}}}{2\tau_{R_1L_1}} \qquad (3)$$

If the argument of the square root is negative, the roots will be complex conjugates. Physically, the current will include sinusoidal components. This is the case we are interested in. (If the argument of the square root is positive, there will be two real roots, a single current pulse and no need for the diode.) In the sinusoidal case, the roots of the characteristic equation can be used to write down the expression for the charge stored in the capacitor as follows:

$$q(t) = e^{-\frac{t}{2\tau_{R_1L_1}}}\left[A\sin\left(\frac{t}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) + B\cos\left(\frac{t}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right)\right] \qquad (4)$$

We can use the definition of current in terms of the charge to work out an expression for i_C.

$$\begin{aligned} i_C(t) \quad &= -\frac{dq}{dt} \\ &= \left\{ \begin{array}{l} -\dfrac{1}{2\tau_{R_1L_1}} e^{-\frac{t}{2\tau_{R_1L_1}}}\left[A\sin\left(\dfrac{t}{2\tau_{R_1L_1}}\sqrt{\dfrac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) + B\cos\left(\dfrac{t}{2\tau_{R_1L_1}}\sqrt{\dfrac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right)\right] + \cdots \\ \cdots + \dfrac{1}{2\tau_{R_1L_1}}\sqrt{\dfrac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\, e^{-\frac{t}{2\tau_{R_1L_1}}}\left[A\cos\left(\dfrac{t}{2\tau_{R_1L_1}}\sqrt{\dfrac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) - B\sin\left(\dfrac{t}{2\tau_{R_1L_1}}\sqrt{\dfrac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right)\right] \end{array} \right\} \end{aligned} \qquad (5)$$

I am not going to bother doing the algebra. It is clear from Equation (5) that the waveform of the current has the same structure as the waveform of the charge, namely, a combination of sinusoids at a certain frequency, multiplied by an exponential decay. If we were to re-arrange Equation (5) we would come up with the following expression, in which constant coefficients P and Q would be complicated combinations of A and B:

$$i_C(t) = e^{-\frac{t}{2\tau_{R_1L_1}}}\left[P\sin\left(\frac{t}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) + Q\cos\left(\frac{t}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right)\right] \qquad (6)$$

The reason I did not do the algebra to nail down the expressions for P and Q is that I am going to evaluate them directly from the circuit's initial conditions. We know the circuit starts up from rest, with $i_C = 0$ at time $t = 0$. Substituting $t = 0$ into both sides of Equation (6) tells us that:

$$\text{IC\#1:} \quad 0 = e^{-\frac{0}{2\tau_{R_1L_1}}}\left[P\sin\left(\frac{0}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) + Q\cos\left(\frac{0}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right)\right]$$

$$\rightarrow \qquad 0 = 1[P \cdot 0 + Q \cdot 1]$$

$$\rightarrow \qquad 0 = Q \qquad (7)$$

That was easy enough. The second initial condition involves the first derivative of the current. Setting $Q = 0$ in Equation (6) and differentiating, we get:

$$\begin{aligned}\frac{di_C}{dt} &= \frac{d}{dt}\left[Pe^{-\frac{t}{2\tau_{R_1L_1}}}\sin\left(\frac{t}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)\right] \\ &= \frac{1}{2\tau_{R_1L_1}}Pe^{-\frac{t}{2\tau_{R_1L_1}}}\left[-\sin\left(\frac{t}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)+\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\cos\left(\frac{t}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)\right] \quad (8)\end{aligned}$$

The second initial condition, which is also obtained from the conditions which exist at time $t = 0$, is based on the observation that the voltage drop over the coil at that instant is equal to the voltage drop over the capacitor. Since no current is flowing at time $t = 0$, there cannot be any voltage drop over the resistor. However, we know the relationship which exists between the voltage drop over the coil and the rate-of-change of the current through it. It is this:

$$v_{coil} = L_1\frac{di_C}{dt} \quad (9)$$

and at time $t = 0$, this becomes:

$$V_0 = L_1\frac{di_C}{dt}\bigg|_{t=0} \quad (10)$$

We can substitute Equation (8) for the derivative, and evaluate it at time $t = 0$, to get:

$$\text{IC\#2:}\quad V_0 = L_1\left\{\frac{1}{2\tau_{R_1L_1}}Pe^{-\frac{0}{2\tau_{R_1L_1}}}\left[-\sin\left(\frac{0}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)+\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\cos\left(\frac{0}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)\right]\right\}$$

$$\rightarrow \qquad V_0 = L_1\left\{\frac{1}{2\tau_{R_1L_1}}P\left[\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right]\right\}$$

$$\rightarrow \qquad P = \frac{2V_0}{L_1}\frac{\tau_{R_1L_1}}{\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}} \quad (11)$$

Note that we can divide by the square root only if it is not zero. If it was zero, we would have to do something else. Subject to that limitation, the capacitor-driven current can be written as:

$$i_C(t) = \frac{2V_0}{L_1}\frac{\tau_{R_1L_1}}{\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}}e^{-\frac{t}{2\tau_{R_1L_1}}}\sin\left(\frac{t}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right) \quad (12)$$

The important question is <u>not</u>: when does this current become zero? Rather, the important question is: when will the voltage drop over the capacitor become zero? That marks the point when the diode becomes forward-biased. When that happens, differential equation (2) will stop being the one that

describes the circuit's behaviour. The other differential equation, Equation $(1B)$, will take over control at that point. And, at that point, current i_C will still be flowing; it will not be zero. In fact, the transition point could very well happen when current i_C is at or near its peak value.

Look at the schematic. When the voltage drop over the capacitor is zero, the voltage drop over the resistor (which Ohm's Law tells us is $R_1 i_C$) must be exactly equal and opposite to the voltage drop over the inductor (which is equal to $L_1\, di_C/dt$). Let's call this transition time the "stopping time" t_S. We already have expressions for i_C and di_C/dt and can substitute them into the required equality. Using the coefficient P once again just for notational ease, the equality at the stopping time is:

$$R_1 i_C(t_S) = -L_1 \left.\frac{di_C}{dt}\right|_{t=t_S}$$

$$R_1 P e^{-\frac{t_S}{2\tau_{R_1L_1}}} \sin\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) = \frac{-L_1}{2\tau_{R_1L_1}} P e^{-\frac{t_S}{2\tau_{R_1L_1}}} \left[\begin{array}{c} -\sin\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) + \cdots \\ \cdots + \sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\cos\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) \end{array}\right]$$

Collecting terms and simplifying gives:

$$\sin\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) = \frac{\tau_{R_1L_1}}{2\tau_{R_1L_1}} \left[\begin{array}{c} \sin\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) + \cdots \\ \cdots - \sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\cos\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) \end{array}\right]$$

$$\rightarrow \quad \frac{1}{2}\sin\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) = -\frac{1}{2}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\cos\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right)$$

$$\rightarrow \quad \sin\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) = -\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\cos\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right) \qquad (13)$$

I have not taken the next logical step (yet), which is to divide both sides by the cosine term to yield a tangent term on the left-hand side. I do not want to lose sight of the quadrant in which the argument (an angle) lies. I also do not want to get fouled up by cases when the argument happens to be zero.

This whole section (the oscillatory response) is based on the assumption that $4\tau_{R_1L_1} > \tau_{R_1C_1}$, which ensures that the argument of the trigonometric terms is strictly positive. Equation (13) will hold true only if the sine term on the left-hand side has the opposite algebraic sign as the cosine term on the right-hand side. The argument must therefore be in either Quadrant 2 or Quadrant 4 of the complete circle. Subject to more information becoming available, both alternatives are equally valid.

In fact, Equation (13) generates lots more possible values for the stopping time. The trigonometric terms are periodic, and each period generates two possibilities. If a certain value of the argument in Equation (13) gives one stopping time, then that value of the argument, plus π, will give another, and so on.

Fortunately, we can exclude all of these possibilities except for the smallest value of the stopping time. The smallest value of stopping time t_S is the one that will occur first in time. Once it occurs, the circuit will transition into diode-control and the higher values of t_S will simply not be relevant.

The way to proceed with Equation (13) is to use the following identities for sine and cosine:

$$\left.\begin{aligned}\sin\left(\pi-\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right) &= +\sin\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)\\ \cos\left(\pi-\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right) &= -\cos\left(\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)\end{aligned}\right\} \quad (14)$$

Substituting these into Equation (13) gives:

$$\sin\left(\pi-\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)=+\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\cos\left(\pi-\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)$$

$$\rightarrow \quad \tan\left(\pi-\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)=\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}$$

$$\rightarrow \quad \pi-\frac{t_S}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}=\tan^{-1}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}$$

$$\rightarrow \quad t_S=\frac{2\tau_{R_1L_1}}{\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}}\left[\pi-\tan^{-1}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right] \quad (15)$$

This expression will pick out the lowest value of the stopping time.

Comparing the stopping time to the time of peak current.

I am going to use the symbol t_P for the instant in time when the current peaks. At this juncture, it is not clear whether the peak time will occur before or after the stopping time. To allow for either outcome, I will mentally remove the diode. Equation (8) is the derivative of the current. Since we now know the value of coefficient *P*, we could if we wanted substitute it into Equation (8). I won't do that; it is more convenient for the algebra to leave it in place. The current will peak at the moment when its time-derivative is zero. Evaluating Equation (8) at this moment gives:

$$\left.\frac{di_C}{dt}\right|_{t=t_P}=0$$

$$\rightarrow \quad \frac{1}{2\tau_{R_1L_1}}Pe^{-\frac{t_P}{2\tau_{R_1L_1}}}\left[-\sin\left(\frac{t_P}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)+\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\cos\left(\frac{t_P}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)\right]=0$$

and, continuing:

$$\rightarrow \quad \sin\left(\frac{t_P}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)=\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\cos\left(\frac{t_P}{2\tau_{R_1L_1}}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right)$$

$$\rightarrow \qquad t_P=\frac{2\tau_{R_1L_1}}{\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}}\tan^{-1}\left(\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}\right) \qquad (16)$$

The expressions for the stopping time t_S and the peak time t_P are very similar. They can be combined to give:

$$t_S=\frac{2\tau_{R_1L_1}\pi}{\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}}-t_P \quad (17)$$

The stopping time t_S can occur before or after the peak time. The two times can also be equal, in which case:

$$t_S=t_P=\frac{\tau_{R_1L_1}\pi}{\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}}-1}} \quad (18)$$

The behaviour of the circuit once the diode takes control

After the stopping time, the diode clamps the voltage drop over the capacitor to zero. The differential equation which describes the circuit thereafter is Equation $(1B)$, which can be written using the *R*-*L* time constant as:

$$\tau_{R_1L_1}\frac{di_D}{dt}+i_D=0 \quad (19)$$

This is a straight-forward exponential decay, with the solution:

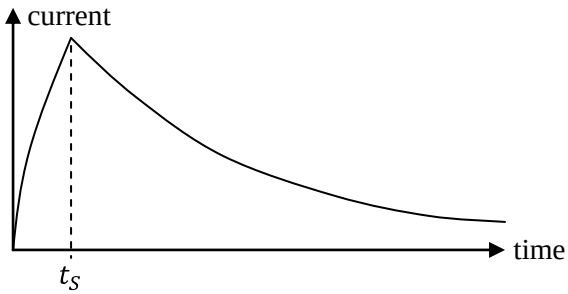
$$i_D(t)=Xe^{-\frac{t}{\tau_{R_1L_1}}} \quad (20)$$

The initial condition which we can use to compute the unknown coefficient X is easy to state. The value of the current at which the diode-driven phase of the response begins must be equal to the value of the current at which the capacitor-driven phase ends. We can calculate using Equation (15) the stopping time t_S at which the handover occurs. Substituting that stopping time into Equation (12) will give use the current flowing through the resistor and the inductor at the instant of transition. Substituting that value for the current into Equation (20) will allow us to solve for constant coefficient X. The algebra to do this is pretty messy, and I will not do it.

If we were integrating numerically, the transition is even easier to manage. We simply make a note of the current at the end of the time step in which the capacitor's voltage goes negative. That becomes the starting current for the exponential decay described by Equation (20).

Maximizing the transfer of energy from the capacitor to the inductor

In general, the current flowing through the coil will look something like this. Prior to the stopping time, there will be some type of rising waveform. The current may or may not reach a peak before the stopping time t_S. After the stopping time, the current will decay to zero.



After the stopping time, the curve is an exponential decay, as the inductor "discharges" through the diode and resistor combination. One can think of the stopping time as the moment when the capacitor has finished transferring out its initial energy. Some will have burned off in the resistor as heat, but the rest is now stored in the coil's magnetic field. In a sense, the capacitor "charges up" the inductor during the first phase of the response.

It might make sense to set as a goal the task of maximizing the amount of energy which is transferred into the inductor by the stopping time. Once the inductor has all the energy, or as much as it can get, it can bleed it out as it accelerates the armature. Maximizing the energy the inductor gets is tantamount to minimizing the amount that is burned off as heat. If the transfer of energy happens too fast, the current will be higher than necessary and the resistor, which burns off energy at the rate Ri^2, burns off too much. If the tranfer happens too slowly, the current will be less but it will flow for longer than necessary.

Since we are ignoring the armature altogether, we know that the amount of magnetic energy that will be stored in the inductor at the stopping time is equal to:

$$\begin{aligned} E(t_s) &= \tfrac{1}{2} \times L_1 \times i_C^2(t_s) \\ &= \tfrac{1}{2} L_1 \left[\frac{2V_0}{L_1} \frac{\tau_{R_1L_1}}{\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}} e^{-\frac{t_s}{2\tau_{R_1L_1}}} \sin\left(\frac{t_s}{2\tau_{R_1L_1}} \sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1} \right) \right]^2 \\ &= \frac{2V_0^2}{L_1} \frac{\tau_{R_1L_1}^2}{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1} e^{-\frac{t_s}{\tau_{R_1L_1}}} \sin^2\left(\frac{t_s}{2\tau_{R_1L_1}} \sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1} \right) \end{aligned} \quad (21)$$

I am not happy that the inductance L_1 appears unaccompanied in the expression. We should be able to remove the explicit dependence if we re-state the leading coefficient in terms of the initial energy stored inside the capacitor. To do this, we are going to use the basic energy relationship for capacitors. The capacitor's initial energy E_0 can be written as:

$$E_0 = \tfrac{1}{2} \times C_1 \times V_0^2$$
$$\rightarrow \quad 2V_0^2 = \frac{4E_0}{C_1} = \frac{4E_0R_1}{\tau_{R_1C_1}} \quad (22)$$

and Equation (21) reduces to:

$$E(t_s) = \frac{4E_0}{\tau_{R_1C_1}} \frac{R_1}{L_1} \frac{\tau_{R_1L_1}^2}{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1} e^{-\frac{t_s}{\tau_{R_1L_1}}} \sin^2\left(\frac{t_s}{2\tau_{R_1L_1}} \sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1} \right)$$

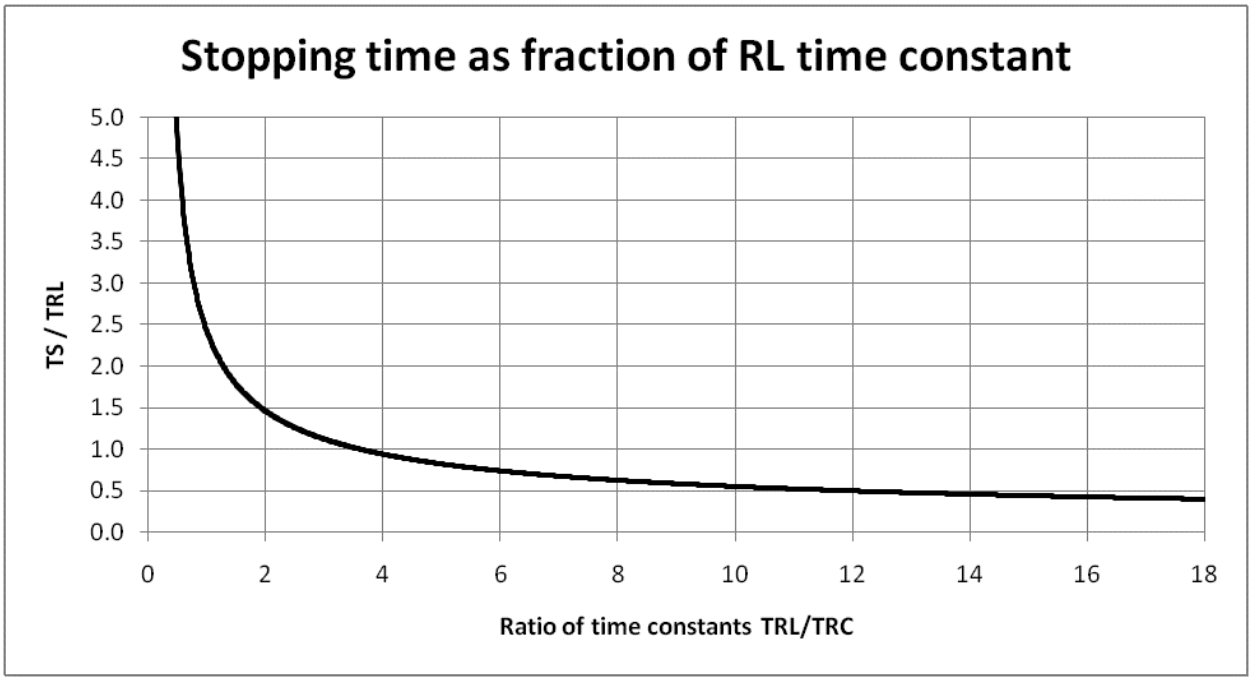
$$\rightarrow \qquad \frac{E(t_s)}{E_0} = \frac{1}{1 - \frac{\tau_{R_1C_1}}{4\tau_{R_1L_1}}} e^{-\frac{t_s}{\tau_{R_1L_1}}} \sin^2\left(\frac{t_s}{2\tau_{R_1L_1}} \sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1} \right) \qquad (23)$$

This is the fraction of the capacitor's initial energy E_0 which resides in the inductor's magnetic field $E(t_s)$ at the stopping time t_s. We want to maximize this fraction. I am going to do so graphically rather than analytically.

For this purpose, it is convenient to re-write Equation (15) so that the stopping time t_S is expressed as a fraction of the *R*-*L* time constant, thus:

$$\frac{t_S}{\tau_{R_1L_1}} = \frac{2}{\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}} \left[\pi - \tan^{-1} \sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1} \right] \qquad (15')$$

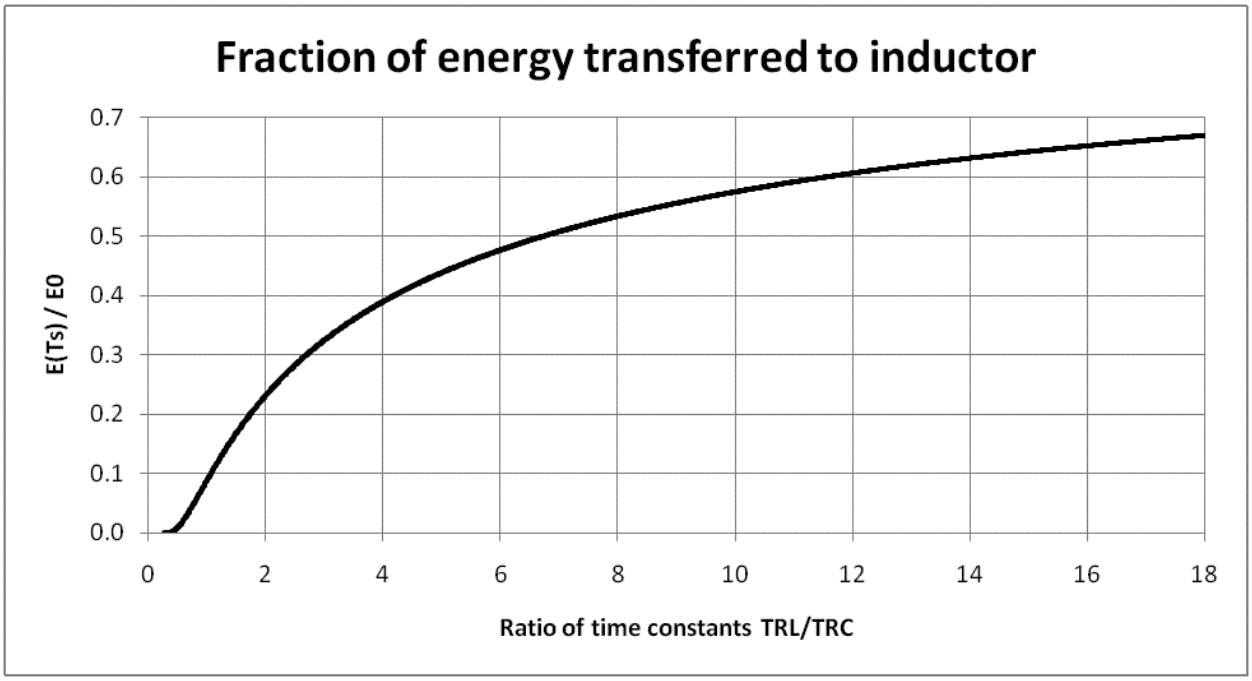
This way, the stopping time fraction depends on only a single dimensionless number, the ratio of the two time constants, $\tau_{R_1L_1}/\tau_{R_1C_1}$. The following graph shows Equation (15′).



Generally speaking, we are going to want the stopping time t_S to be a small fraction of the *R*-*L* time constant $\tau_{R_1L_1}$. The current will decay with time constant $\tau_{R_1L_1}$ and, if things are well-matched, the armature will remain near the barrel coil for just a few of these $\tau_{R_1L_1}$ time constants. We will want to "charge up" the barrel coil in a fraction of that time, perhaps one-tenth or less. So, we are going to be

looking for $\tau_S/\tau_{R_1C_1}$ in the range of 0.05 to 0.1 or thereabouts. From the graph, it looks like this ratio $\tau_S/\tau_{R_1C_1}$ can be made as small as one wants, by picking an arbitrarily big time constant ratio. That's true.

With the stopping time ratio $\tau_S/\tau_{R_1C_1}$ now determined, we can substitute it into Equation (23) to find the fraction of initial energy transferred into the inductor. Here is the corresponding graph of Equation (23).



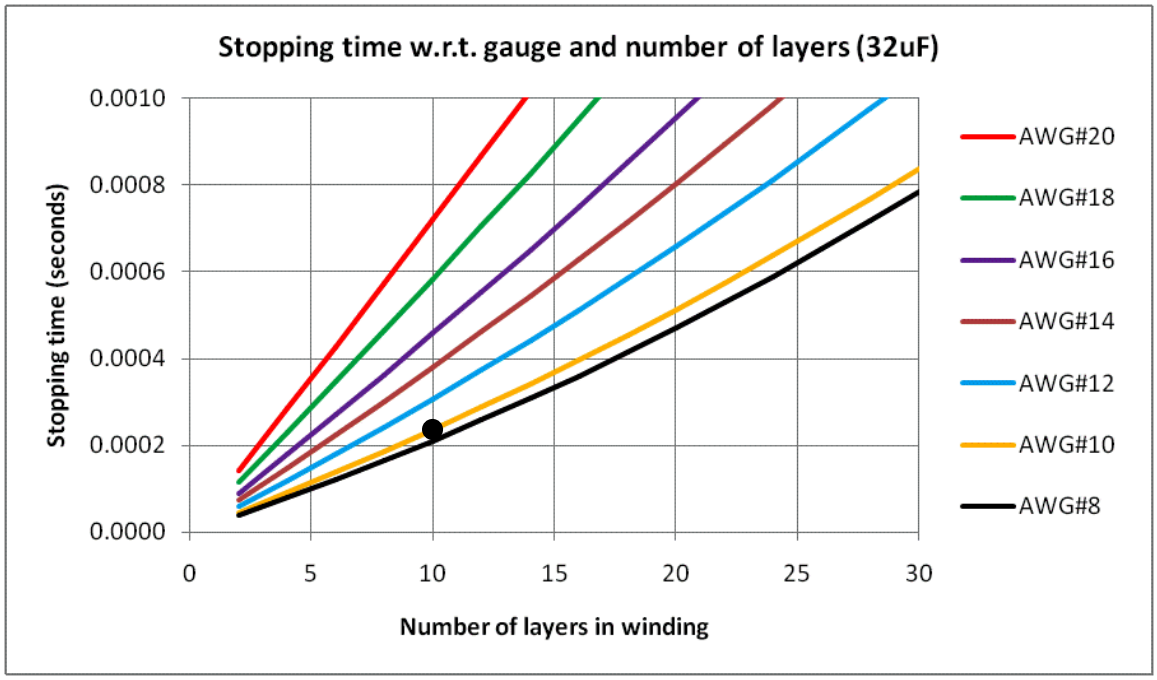
From this graph, it looks like the energy transfer ratio can be made as large as one wants (up to one, or 100%), also by picking an arbitrarily big time constant ratio. That's also true.

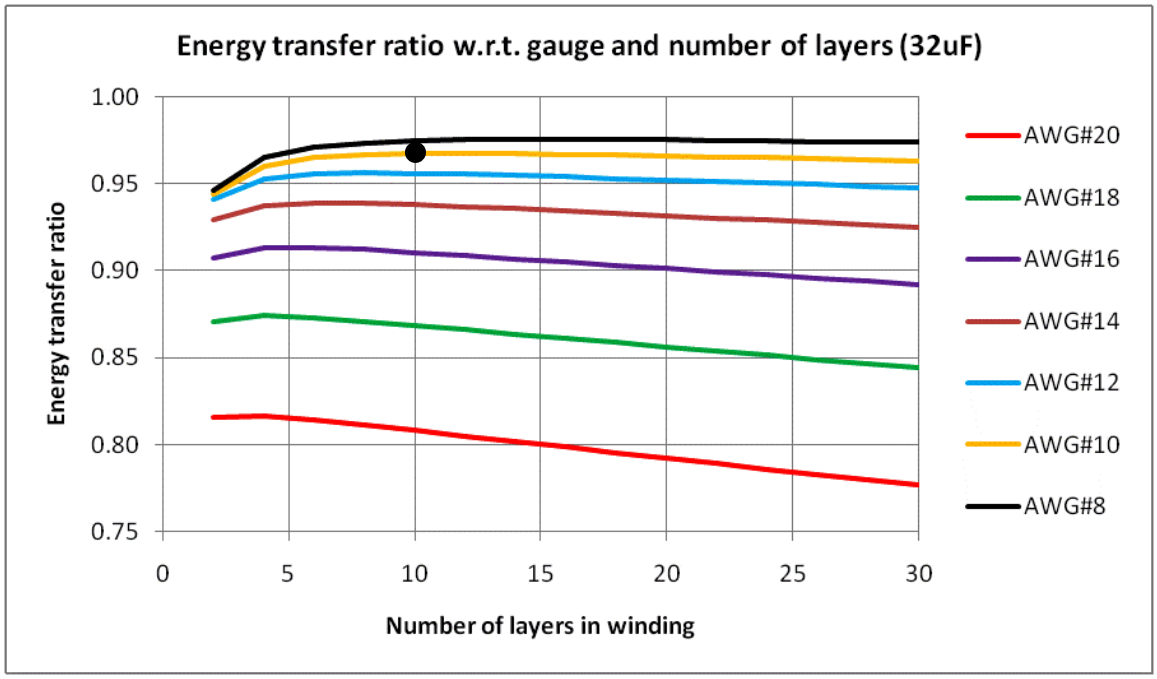
First numerical example

I am going to pause here to work through a numerical example. I am going to begin with the $32\mu F$ capacitor I used in the earlier papers. With the capacitance fixed at this value, I am going to use FEMM to build (virtually) a few coils using the characteristics of standard enameled copper wire. FEMM can calculate the Ohmic resistance and self-inductance of each coil. As before, I will assume that the Ohmic resistance of the winding comprises most of the resistance around the circuit, and that the SCR which controls when the circuit turns on adds an extra 0.02Ω to the total resistance in the circuit. An implicit assumption here is that the equivalent series resistance of the capacitor is negligible compared with that of the winding.

The first of the following two graphs is a plot of the stopping time, in seconds. The horizontal axis is the number of layers in the winding, which ranges from two to 30. There is a separate curve for each different wire gauge. I have cut the top of the graph at stopping times of one millisecond, assuming that we will be more interested in "charging up" the inductor in times shorter than that. Generally, wires with a larger diameter (and smaller AWG gauge numbers) make the charging quicker. Fewer layers in the winding also makes the charging quicker.

The second of the following two graphs is the energy transfer ratio for these same gauges and numbers of layers. Note that the legend for the curve colors is now upside down – larger diameter wires exhibit better energy transfer.

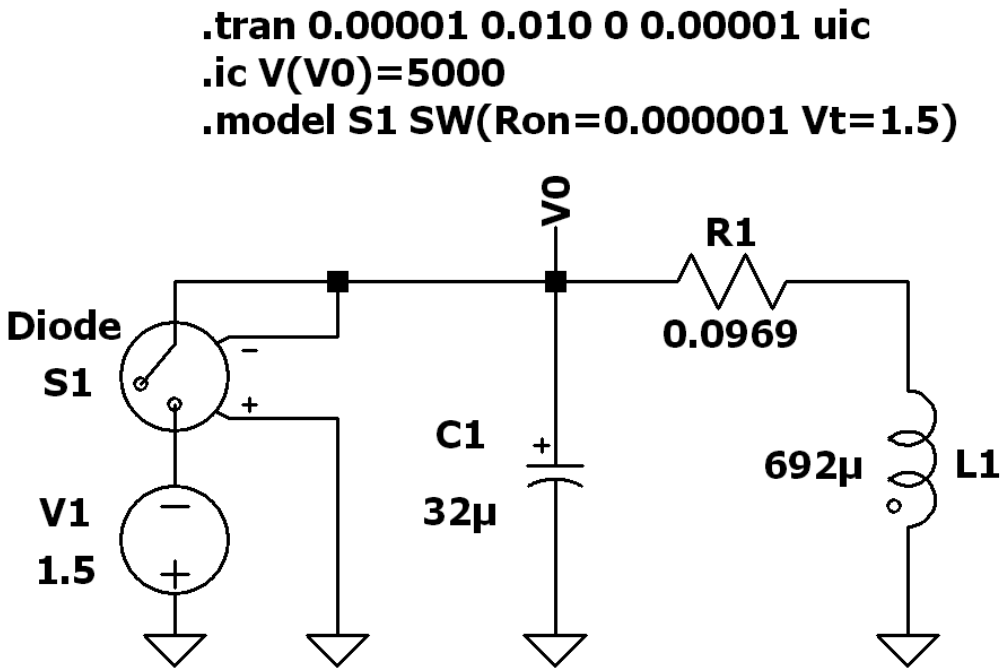




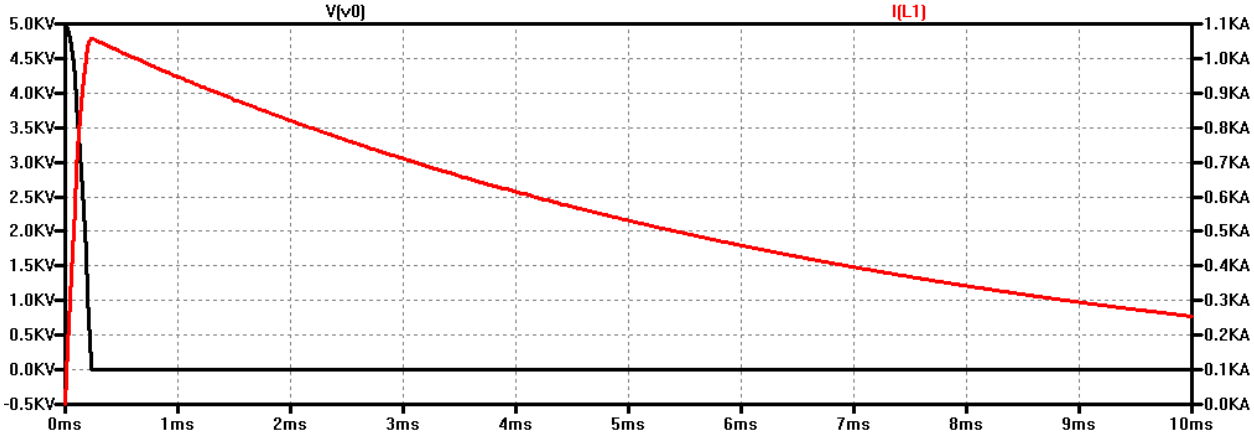
I am going to validate these conclusions by preparing a small LTSpice model. For this purpose, I am going to use as an example a barrel coil with ten layers of AWG #10 wire. This configuration is marked by a black dot in the two previous graphs. The FEMM analysis showed that this coil has an Ohmic resistance of 0.0769Ω (so, a circuit resistance of 0.0969Ω) and a self-inductance of $692\mu H$. The following LTSpice schematic shows the circuit to be simulated.

I should say a couple of words about how the diode is represented. My library of LTSpice components does not include an ideal diode. Nor do I want to get sidetracked selecting a real diode. Instead, I have modeled the diode as a voltage-controlled switch. Switch $S1$ will close when the voltage over its $+$ and $-$ terminals exceeds 1.5 Volts. Note that this threshold voltage drop is opposite in sense to the initial

voltage drop over the capacitor. In other words, the capacitor will have to have a small reverse voltage (1.5 Volts) before the diode is goaded into its conduction mode. Since LTSpice does not permit switches to have exactly zero resistance, I have set it to something very small, 0.000001Ω. The 1.5 Volt voltage source $V1$ represents the voltage drop over the diode once it does begin to conduct.



The simulation resulted in the following waveforms. The black curve is the current flowing through the resistor and inductor. It has the expected shape. The red curve is the voltage drop over the capacitor, which falls to zero and stays there. To be more precise, the voltage drop over the capacitor falls to -1.5 Volts and stays there.



There are a couple of more things that we can say from this graph. The stopping time is about 0.23 milliseconds, just what we would expect from the position of the black dot in the top graph on page 11. We can also confirm the energy transfer ratio of this configuration. The current flowing at the stopping time is about 1060 Amperes, so the energy contained in the inductor's magnetic field is about:

$$E(t_s) = \tfrac{1}{2} \times L_1 \times i_C(t_s)^2 = \tfrac{1}{2} \times 692\mu \times 1060^2 = 388.8 \text{ Joules} \quad (24)$$

This is $388.8/400 = 97.2\%$ of the 400 Joules initially stored in the capacitor, just what we would expect from the position of the black dot in the bottom graph on page 11.

What changes are needed to extend this concept to multiple barrel coils?

The earlier paper titled *Using FEMM to design an inductance coil gun* set out the basic analysis of a multiple-coil induction coil gun. The Lua script attached to that paper simulated the firing of a five-coil gun. Both the analysis and the Lua script were based on barrel coils which were simple L-R-C circuits. In this section, I want to highlight the changes which are necessary to accommodate bypass diodes in the barrel coils' circuits.

In the earlier paper, the circuit equations were organized into matrix form, with one row in the matrix for each coil in the gun, including the armature. A coil gun with five barrel coils (all operating), will have a matrix with six rows. Each row corresponds to the circuit equation governing the operation of that row's corresponding circuit. Each circuit also contributed one independent variable to the system.

The use of bypass diodes is restricted to barrel coils. The armature does not have a bypass diode. The armature's circuit will operate in exactly the same way it did before. There will be no change to the armature's circuit equation or to its row in the matrix equation.

Next, consider any barrel coil whose circuit is operating in the reverse-biased mode. The circuit equation for that mode is the same as it was in the earlier paper. It follows that there will be no change to the row in the matrix equation which corresponds to a barrel coil operating in its reverse-biased mode.

There will only be changes to the rows in the matrix equation which correspond to coil circuits in which the diode is forward-biased and conducting. To see what happens, I am going to re-state here the matrix equation as it stands if barrel coils #1, #2 and #3 are operating in their reverse-biased modes. The following equation is Equation (27) from the earlier paper. Recall that S is the speed of the armature and that $\partial M_{Aj}/\partial z$ is the partial derivative which measures the rate at which the mutual inductance between the armature and barrel coil #j changes with changes in the axial position z of the armature.

$$\begin{bmatrix} L_A & M_{A1} & M_{A2} & M_{A3} \\ M_{A1} & L_1 & M_{12} & M_{13} \\ M_{A2} & M_{12} & L_2 & M_{23} \\ M_{A3} & M_{13} & M_{23} & L_3 \end{bmatrix} \cdot \begin{bmatrix} \frac{di_A}{dt} \\ \frac{di_1}{dt} \\ \frac{di_2}{dt} \\ \frac{di_3}{dt} \end{bmatrix} = \cdots$$

$$\cdots = -\begin{bmatrix} R_A & S\frac{\partial M_{A1}}{\partial z} & S\frac{\partial M_{A2}}{\partial z} & S\frac{\partial M_{A3}}{\partial z} \\ S\frac{\partial M_{A1}}{\partial z} & R_1 & 0 & 0 \\ S\frac{\partial M_{A2}}{\partial z} & 0 & R_2 & 0 \\ S\frac{\partial M_{A3}}{\partial z} & 0 & 0 & R_3 \end{bmatrix} \cdot \begin{bmatrix} i_A \\ i_1 \\ i_2 \\ i_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{C_1} & 0 & 0 \\ 0 & 0 & \frac{1}{C_2} & 0 \\ 0 & 0 & 0 & \frac{1}{C_3} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (25)$$

Take barrel coil #2 as an example. The third row in the matrix equation is the circuit equation for this coil. The third row by itself is:

$$M_{A2}\frac{di_A}{dt} + M_{12}\frac{di_1}{dt} + L_2\frac{di_2}{dt} + M_{23}\frac{di_3}{dt} = -S\frac{\partial M_{A2}}{\partial z}i_A - R_2 i_2 + \frac{q_2}{C_2} \quad (26)$$

The very last term q_2/C_2 is equal to the voltage drop over capacitor C_2 in the circuit of barrel coil #2. This equation is the multiple-coil counterpart of circuit equation (2) above. (Recall that current i_2 is equal by definition to $-\,dq_2/dt$.)

Equation (26) will govern the behaviour of barrel coil #2's circuit from the time the circuit is closed, during the time capacitor C_2 discharges, all the way down through the point when the charge and voltage are zero $q_2 = v_2 = 0$, and then just a little bit more. When the voltage drop over the capacitor reaches the threshold voltage of the diode (-1.5 Volts in the LTSpice schematic above), the diode will take over. It will clamp the voltage drop to -1.5 Volts, where it will stay. Now, 1.5 Volts is representative of the real diodes we will use, but just to keep all quantities in parametric form I will use the symbol V_{diode2} for the forward-biased voltage drop over the diode in barrel coil #2's circuit. V_{diode2} will always be algebraically positive.

Once the diode kicks in, the circuit equation will be:

$$M_{A2}\frac{di_A}{dt} + M_{12}\frac{di_1}{dt} + L_2\frac{di_2}{dt} + M_{23}\frac{di_3}{dt} = -S\frac{\partial M_{A2}}{\partial z}i_A - R_2 i_2 - V_{diode2} \quad (27)$$

The transition from Equation (26) to Equation (27) will occur at the instant in time (or during the time step) when the charge stored in capacitor C_2 reaches the following negative value:

$$q_2 = -C_2 V_{diode2} \quad (28)$$

That moment in time (or time step) when the transition occurs is the stopping time, as described above.

The changes to the matrix equation can be summarized quite simply:

1. As long as $q_2 > -C_2 V_{diode2}$, there is no change at all in how the third row is handled.
2. Once q_2 falls to $-C_2 V_{diode2}$, it will stay there. (The operation of the diode keeps it there.)

The Lua script which is listed in Appendix "A" below includes this modification for all five barrel coils.

Second numerical example

In this section, I want to do a time simulation using the new Lua script. I will only fire up barrel coil #1. It goes without saying that I want an efficient transfer of energy from the capacitor to the inductor. But, I want to impose another restriction as well.

Take a look at the waveforms of current and voltage in the middle of page 12. They are the result of an LTSpice simulation for a barrel coil chosen pretty much at random. The coil current peaks at about 1060 Amperes, which is good. What's not so good is the length of time it takes the current to decay thereafter. Even after ten milliseconds, the current is still about 250 Amperes. I am hoping the armature will be long gone from barrel coil #1 ten milliseconds after firing.

Suppose we set a goal to have the current substantially reduced within three milliseconds. We want barrel coil #1 to have completed its work within three milliseconds. Equation (20) above is the shape of the current's waveform after the diode begins to conduct. It is a straight-forward exponential decay with a

time constant of $\tau_{R_1L_1}$. One of the rules-of-thumb for exponential decay is that the decay is almost complete after five time constants. To achieve this goal, the *R-L* time constant needs to be approximately $\tau_{R_1L_1} = 0.003/5 = 0.0006$ seconds.

Suppose we set a second goal, that the stopping time should be one-tenth of the overall time during which barrel coil #1 is "active". Then, the stopping time should be about $t_S = 0.003/10 = 0.0003$ seconds.

The ratio of the stopping time and the *R-L* time constant is then:

$$\frac{t_S}{\tau_{R_1L_1}} = \frac{0.0003}{0.0006} = 0.5 \quad (29)$$

There is a unique value of time constant ratio which corresponds to this stopping time ratio. We can find it by looking at the graph at the bottom of page 9. Or, we can find it analytically by solving Equation $(15')$. Doing the work analytically, we get:

$$(15'): \quad 0.5 = \frac{2}{\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}} \left[\pi - \tan^{-1}\sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1}\right]$$

$$\rightarrow \quad \sqrt{\frac{4\tau_{R_1L_1}}{\tau_{R_1C_1}} - 1} = 6.862$$

$$\rightarrow \quad \frac{\tau_{R_1L_1}}{\tau_{R_1C_1}} = 12.02 \quad (30)$$

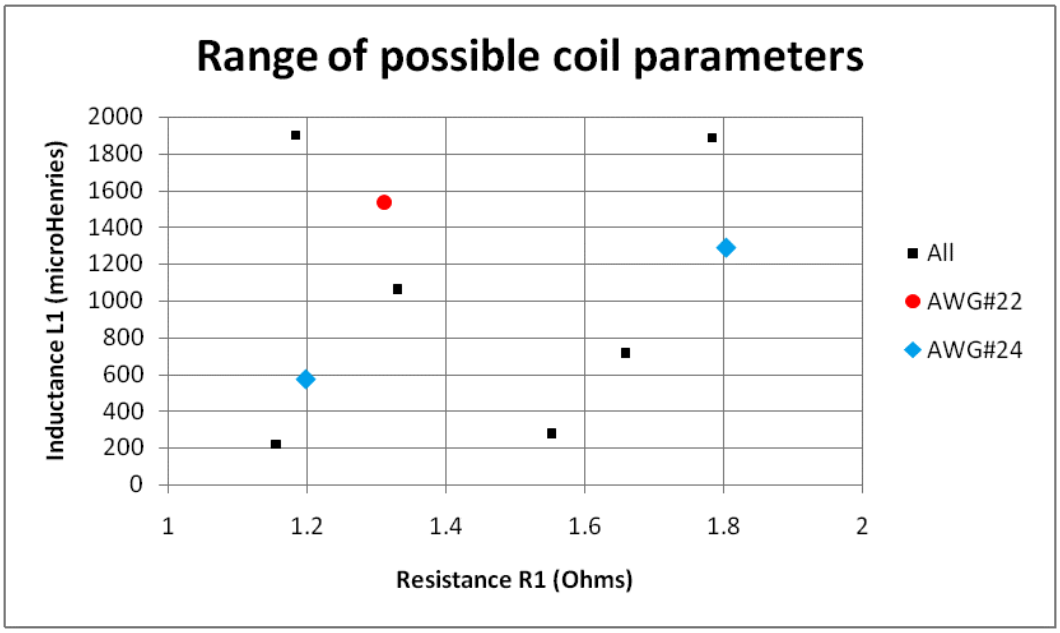
With this time constant ratio, we can see from the energy transfer graph above (page 10) that just over 60% of the capacitor's initial energy will be transferred into the inductor by the stopping time.

Since the *R-L* time constant is 0.0006 seconds, Equation (30) tells us that the *R-C* time constant must be about $0.0006/12.02 = 0.0000499$ seconds, or 50 microseconds.

If we use a 32uF capacitor, then a $50\mu\text{s}$ *R-C* time constant requires a circuit resistance of $R_1 = \tau_{R_1C_1}/C_1 = 50\mu\text{s}/32\mu\text{F} = 1.56\Omega$. With this resistance, a $600\mu\text{s}$ *R-L* time constant requires a self-inductance of $L_1 = \tau_{R_1L_1} \times R_1 = 600\mu\text{s} \times 1.56\Omega = 936\mu\text{H}$. The coil we need must have an Ohmic resistance of 1.54Ω (don't forget the SCR) and a self-inductance of $936\mu\text{H}$. I have looked through the output from the FEMM program which produced the data for the graphs on page 11, which print the resistance and inductance, too, and do not see a suitable candidate. The following table sets out the characteristics of the coils which are closest to the one required.

AWG gauge	**Number of layers**	$\boldsymbol{R_1}$	$\boldsymbol{L_1}$
#21	6	1.433Ω	$2740\mu\text{H}$
#22	4	1.313Ω	$1530\mu\text{H}$
#23	4	1.783Ω	$1890\mu\text{H}$
#24	2	1.199Ω	$572\mu\text{H}$
#25	2	1.658Ω	$710\mu\text{H}$
#26	2	2.310Ω	$892\mu\text{H}$

Up to this point, I have only tested even numbers of layers, and I have not considered (and will not consider) partial layers in the winding. Nor do I intend to consider mixing wire sizes in a winding. To see the additional possibilities, I re-ran the FEMM analysis for odd numbers of layers. The following graph shows the range of possibilities which exists. This is a scatter graph, with a coil's inductance on the vertical axis and the corresponding circuit resistance along the horizontal. This is the complete set of possibilities, which is to say, that these (few) points are the only choices available for wire gauges from AWG #18 to AWG #32 and between one and eight layers in the winding. All the other coils have resistances or inductances outside of the ranges shown in the graph.



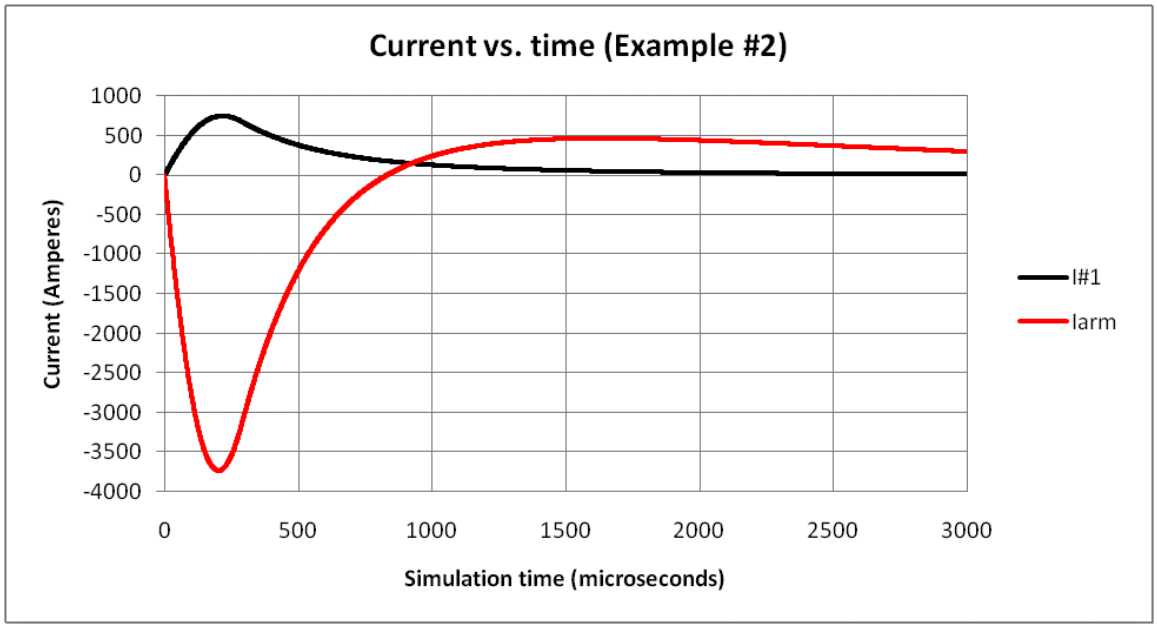
The square black dots correspond to coils wound using odd AWG gauges. I have elected to exclude those points from further consideration (hard to buy, and therefore expensive). I am going to focus only on the three possibilities which remain, which can be wound from the more common sizes AWG #22 and AWG #24. For these three possibilities, I am going to reverse the earlier analysis and determine the time constants to which they give rise.

AWG	Layers	$R_1(\Omega)$	$L_1(\mu H)$	$\tau_{R_1L_1}(\mu s)$	$5\tau_{R_1L_1}(s)$	$\tau_{R_1C_1}(\mu s)$	$\tau_{R_1L_1}/\tau_{R_1C_1}$	$t_S(\mu s)$
#22	4	1.313	1530	1170	0.0059	42.0	27.9	371
#24	2	1.199	572	477	0.0024	38.4	12.4	234
#24	3	1.804	1287	713	0.0035	57.7	12.4	350

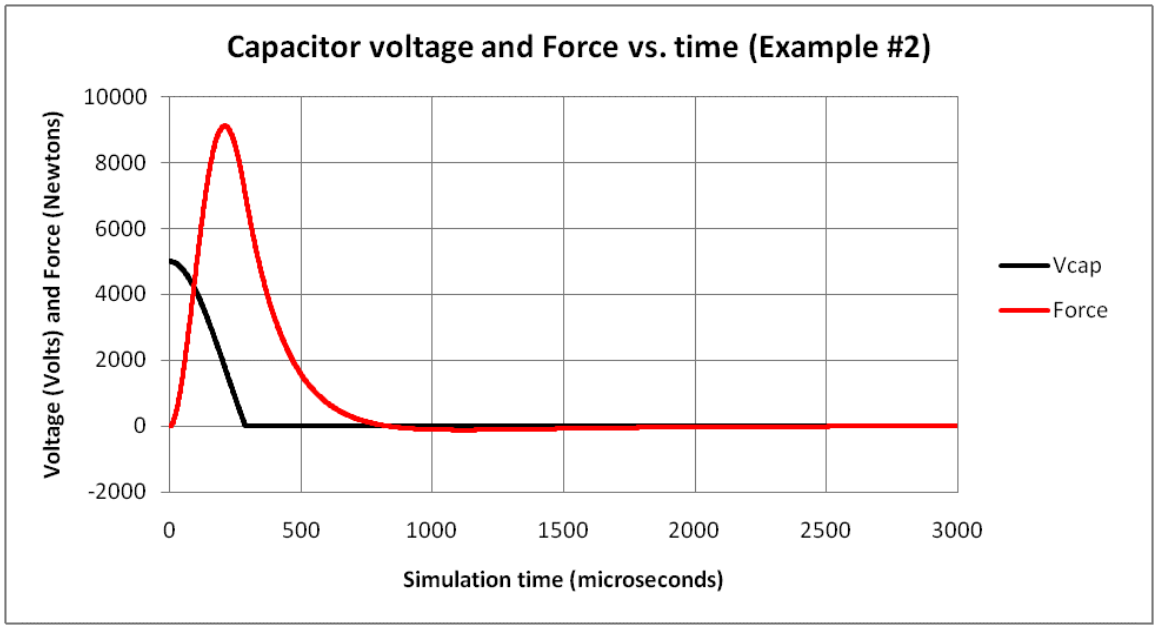
It looks like the coil in the last row is closest to our goals. I have shaded the most important cells in yellow. The parameter $5\tau_{R_1L_1}$ is five times the R-L time constant, and is the time during which this barrel coil will be active: 3.5 milliseconds. The stopping time t_S is 350 microseconds, not far off the 300 microseconds we were looking for.

Conclusion: We will wind the barrel coil using three turns of AWG #24 wire.

The Lua script listed in Appendix "A" ran the time simulation which produced the following results. The first graph is a plot of the current flowing through barrel coil #1 and the armature. The waveforms have the shapes we want, but they are too fast.



The current flowing through barrel coil #1 (in black) is a single pulse. The current through the armature (in red) changes direction after about 90μs. The reversal of the direction of the current flowing through the armature causes the force to reverse direction. The following graph shows this reversal. The following graph also shows the voltage drop over the capacitor (in black), which is exactly what we want. The capacitor's voltage goes to zero (more precisely, −1.5 Volts), but stops there.



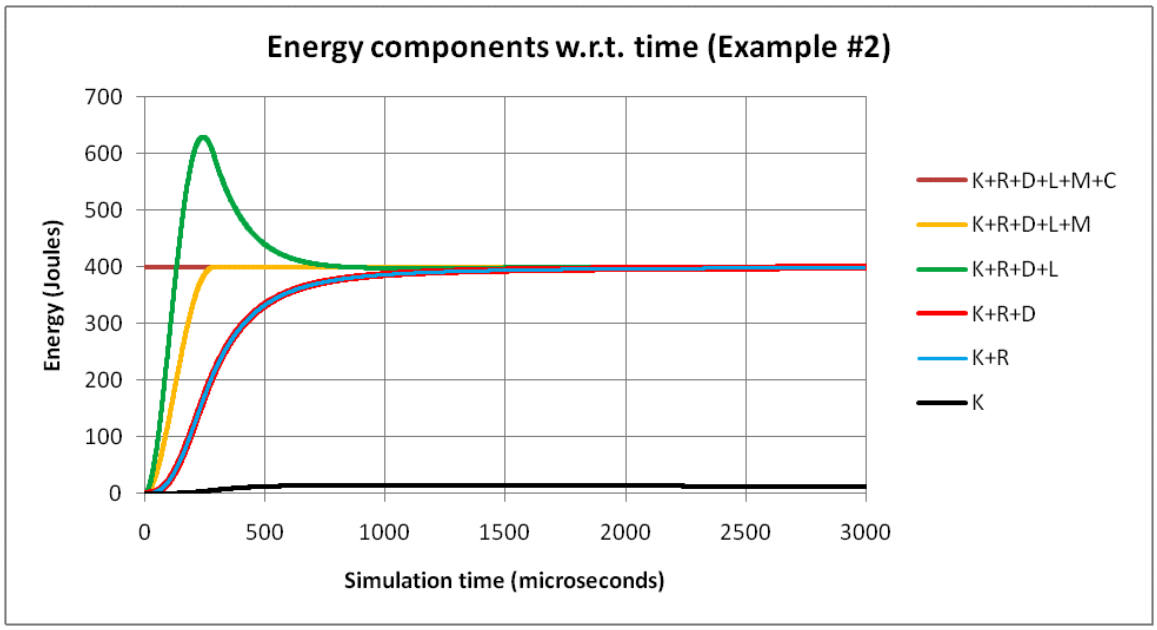
Although the direction of the force reverses, its magnitude in the wrong direction is minor, and we could probably ignore it.

What we cannot ignore is the speed of the decay. My calculations above were designed to produce an exponential decay which would be "finished" after about three milliseconds. The decay we are seeing here is finished in less than one millisecond.

Bear in mind that the calculations were based on a circuit model which excluded the armature entirely. The armature represents a use of energy which is not accounted for in the formulae, and has the effect of speeding up the decay.

In the next paper, I will bring the armature back into the model (at least partly) and, by accounting for its use of energy, re-calibrate the parameters to slow the decay down to three milliseconds.

Before ending this paper, I want to show one more result of the time simulation. The following graph shows the various components of energy during the simulation. The diode burns off heat as well – like all physical components, its power consumption is voltage times current or, in this case, $V_{diode} \times i_1$. The following graph shows the "pools" of energy during the simulation. As before, K is the kinetic energy of the armature. $K + R$ (the blue curve) is that kinetic energy plus the cumulative energy burned off by the resistances as heat. $K + R + D$ (the red curve) is that energy plus the cumulative energy burned off by the diode as heat. The red curve is very slightly above the blue curve. There is not much difference between the red curve and the blue curve because the diode does not burn off very much heat. The curve $K + R + D + L$ adds the magnetostatic energy stored in the self-inductance fields. The next layer, $K + R + D + L + M$, includes the magnetostatic energy stored in the mutual inductance fields. The final layer, $K_R + D + L + M + C$, also includes the electrostatic energy stored inside the capacitor. It remains at 400 Joules throughout. (In this simulation, I used a time step of 0.2 microseconds. The error in the energy reconciliation at the end of the run was less than 0.07 Joules.)



Jim Hawley
© June 2015

If you found this description helpful, please let me know. If you spot any errors or omissions, please send an e-mail. Thank you.

Appendix "A"

Listing of revised Lua script for time simulations with bypass diodes

```
-- This Lua script is the base case for simulating multiple coils with
-- bypass diodes wired across all barrel coil capacitors.

--////////////////////////////////////////////////////////////////////////
--// Matrix inversion function
--// Warning: Does not check for singularities
--// Warning: The function overwrites A[][] and B[]
--// Local argument N is the total number of coils, including the armature
--// Global variables A[][] and B[] must be populated before a call
--// Global variable X[] holds the solution
A = {}       -- Inductance matrix, maximum size N=10
for I= 1,10 do
    A[I] = {}
end
B = {}       -- Vector of constants
X = {}       -- Solution vector
for I= 1,10 do
    X[I] = 0
end
function INVERT(N)
    -- Part #1: Reduce the leading element in row #1 to one
    temp = A[1][1]
    for Icol = 1,N do
        A[1][Icol] = A[1][Icol] / temp
    end
    B[1] = B[1] / temp
    -- Part #2: On a row-by-row basis, reduce the matrix to upper-triangular
    for Irow = 2,N do
        for Icol = 1,(Irow-1) do
            temp = A[Irow][Icol]
                for Irun = 1,N do
                    A[Irow][Irun] = A[Irow][Irun] - (temp*A[Icol][Irun])
              end
            B[Irow] = B[Irow] - (temp*B[Icol])
        end
        temp = A[Irow][Irow]
        for Irun = Irow,N do
            A[Irow][Irun] = A[Irow][Irun] / temp
        end
        B[Irow] = B[Irow] / temp
    end
    -- Part #3: Set the value of the last variable
    X[N] = B[N]
    -- Part #4: Use Euler elimination to work back up the equations
    for Itemp = 1,N-1 do
        Irow = N - Itemp
        X[Irow] = 0
        for Icol = (Irow+1),N do
            X[Irow] = X[Irow] + (A[Irow][Icol]*X[Icol])
        end
        X[Irow] = B[Irow] - X[Irow]
```

```
    end
    return
end
--// End of matrix inversion function
--////////////////////////////////////////////////////////////////////////////

--////////////////////////////////////////////////////////////////////////////
--// Save-matrix-equation-to-file function, for debugging purposes only
function WRITEMATRIXEQUATION(MO)
    for Ir = 1,MO do
        for Ic = 1,MO do
            write(handle,"A[",Ir,"][",Ic,"]=",A[Ir][Ic],"   ")
        end
        write(handle,"X[",Ir,"]=",X[Ir],"   ")
        write(handle,"B[",Ir,"]=",B[Ir],"\n")
    end
    write(handle,"\n")
end
--// End of save-matrix-equation-to-file function
--////////////////////////////////////////////////////////////////////////////

-- Initial location and speed of the armature
ArmPosEndZ_0 = -1.25      -- Location of armature's leading edge, inches
Speed_0 = 0               -- Armature's initial speed, meters per second
Mass = 0.25               -- Armature's mass, kilograms

-- Other important variables related to the simulation process
MaxMeshSize = 0.01        -- Maximum FEMM triangle size
MaxResidual = 1E-8        -- Maximum FEMM residual for convergence test
deltaT = 0.0000002        -- Duration of one time step, in seconds
SaveInterval = 5          -- Number of time steps between save events
SaveCounter = 0           -- Counter of time steps between save events
MaxNumTimeSteps = 100000  -- Maximum number of time steps permitted

-- Definition of Groups
-- Group(9) is the air
-- Group(0) is the armature
-- Group(1) is barrel coil #1
-- Group(2) is barrel coil #2
-- Group(3) is barrel coil #3
-- Group(4) is barrel coil #4
-- Group(5) is barrel coil #5

-- ALL DIMENSIONS ARE IN INCHES UNLESS A SUFFIX STATES OTHERWISE

-- Define string names for all coils
CoilName = {}
CoilName[0] = "Arm"
CoilName[1] = "Bcoil1"
CoilName[2] = "Bcoil2"
CoilName[3] = "Bcoil3"
CoilName[4] = "Bcoil4"
CoilName[5] = "BCoil5"

-- Specify wire gauges for all coils
```

```
CoilAWG = {}
CoilAWG[0] = 10
CoilAWG[1] = 24
CoilAWG[2] = 16
CoilAWG[3] = 16
CoilAWG[4] = 16
CoilAWG[5] = 16

-- Define vectors for the diameters of the wires used to wind all coils
CoilWireExtDiamMM = {}
CoilWireBareDiamMM = {}
CoilWireExtDiam = {}
CoilWireBareDiam = {}

-- Specify lengths for all coils.  The numbers of turns will be calculated.
CoilLength = {}
CoilLength[0] = 1
CoilLength[1] = 1
CoilLength[2] = 1
CoilLength[3] = 1
CoilLength[4] = 1
CoilLength[5] = 1

-- Define vectors for the numbers of turns which make up the lengths
CoilNumTurns = {}

-- Specify the number of layers in all coils
CoilNumLayers = {}
CoilNumLayers[0] = 2
CoilNumLayers[1] = 3
CoilNumLayers[2] = 5
CoilNumLayers[3] = 2
CoilNumLayers[4] = 1
CoilNumLayers[5] = 1

-- Specify the z-locations of the negative ends of all coils
CoilNegEndZ = {}
CoilNegEndZ[0] = ArmPosEndZ_0 - CoilLength[1]
CoilNegEndZ[1] = -2.75
CoilNegEndZ[2] = -1
CoilNegEndZ[3] = 0.75
CoilNegEndZ[4] = 2.5
CoilNegEndZ[5] = 4.25

-- Define a vector for the z-locations of the positive ends of all coils
CoilPosEndZ = {}

-- Specify the inner and outer diameters of the barrel tube
TubeOuterDiam = 2.5
TubeInnerDiam = TubeOuterDiam - (1/16)
TubeOuterRad = TubeOuterDiam / 2
TubeInnerRad = TubeInnerDiam / 2

-- Define vectors to hold the resistances and self-inductances of all coils.
-- CoilRfemm is the resistance calculated by FEMM, but there will be other
```

```
-- resistance in each coil circuit.
CoilRfemm = {}
CoilLfemm = {}

-- Define an array to hold the instantaneous mutual inductances
CoilMfemm = {}
for I = 0,5 do
    CoilMfemm[I] = {}
end

-- Specify the dimensions of the bounding air spheres
AirLocalRad = 8
AirExtDiam = 1

--////////////////////////////////////////////////////////////////////////
--// Enough parameters have been defined to allow us to describe the
--// physical geometry to FEMM.
--////////////////////////////////////////////////////////////////////////
newdocument(0)                                  -- New magnetics problem
mi_probdef(0,'inches','axi',MaxResidual,0,30) -- Axisymmetric, in inches
mi_grid_snap('off')                     -- Do not snap to a grid

-- Set the parameters of the enameled copper wire used to wind each coil
for I = 0,5 do
    AWGGauge = CoilAWG[I]
    if (AWGGauge == 1) then
        WireExtDiamMM = 7.496
        WireBareDiamMM = 7.348
    end
    if (AWGGauge == 2) then
        WireExtDiamMM = 6.690
        WireBareDiamMM = 6.543
    end
    if (AWGGauge == 3) then
        WireExtDiamMM = 5.971
        WireBareDiamMM = 5.827
    end
    if (AWGGauge == 4) then
        WireExtDiamMM = 5.330
        WireBareDiamMM = 5.189
    end
    if (AWGGauge == 5) then
        WireExtDiamMM = 4.757
        WireBareDiamMM = 4.620
    end
    if (AWGGauge == 6) then
        WireExtDiamMM = 4.246
        WireBareDiamMM = 4.115
    end
    if (AWGGauge == 7) then
        WireExtDiamMM = 3.790
        WireBareDiamMM = 3.665
    end
    if (AWGGauge == 8) then
        WireExtDiamMM = 3.383
```

```
    WireBareDiamMM = 3.264
end
if (AWGGauge == 9) then
    WireExtDiamMM = 3.023
    WireBareDiamMM = 2.906
end
if (AWGGauge == 10) then
    WireExtDiamMM = 2.703
    WireBareDiamMM = 2.588
end
if (AWGGauge == 11) then
    WireExtDiamMM = 2.418
    WireBareDiamMM = 2.304
end
if (AWGGauge == 12) then
    WireExtDiamMM = 2.163
    WireBareDiamMM = 2.052
end
if (AWGGauge == 13) then
    WireExtDiamMM = 1.934
    WireBareDiamMM = 1.829
end
if (AWGGauge == 14) then
    WireExtDiamMM = 1.732
    WireBareDiamMM = 1.628
end
if (AWGGauge == 15) then
    WireExtDiamMM = 1.549
    WireBareDiamMM = 1.450
end
if (AWGGauge == 16) then
    WireExtDiamMM = 1.384
    WireBareDiamMM = 1.290
end
if (AWGGauge == 17) then
    WireExtDiamMM = 1.240
    WireBareDiamMM = 1.151
end
if (AWGGauge == 18) then
    WireExtDiamMM = 1.110
    WireBareDiamMM = 1.024
end
if (AWGGauge == 19) then
    WireExtDiamMM = 0.993
    WireBareDiamMM = 0.912
end
if (AWGGauge == 20) then
    WireExtDiamMM = 0.892
    WireBareDiamMM = 0.813
end
if (AWGGauge == 21) then
    WireExtDiamMM = 0.800
    WireBareDiamMM = 0.724
end
if (AWGGauge == 22) then
```

```
        WireExtDiamMM = 0.714
        WireBareDiamMM = 0.643
    end
    if (AWGGauge == 23) then
        WireExtDiamMM = 0.643
        WireBareDiamMM = 0.574
    end
    if (AWGGauge == 24) then
        WireExtDiamMM = 0.577
        WireBareDiamMM = 0.511
    end
    if (AWGGauge == 25) then
        WireExtDiamMM = 0.516
        WireBareDiamMM = 0.455
    end
    if (AWGGauge == 26) then
        WireExtDiamMM = 0.462
        WireBareDiamMM = 0.404
    end
    if (AWGGauge == 27) then
        WireExtDiamMM = 0.419
        WireBareDiamMM = 0.361
    end
    if (AWGGauge == 28) then
        WireExtDiamMM = 0.373
        WireBareDiamMM = 0.320
    end
    if (AWGGauge == 29) then
        WireExtDiamMM = 0.338
        WireBareDiamMM = 0.287
    end
    if (AWGGauge == 30) then
        WireExtDiamMM = 0.307
        WireBareDiamMM = 0.254
    end
    if (AWGGauge == 31) then
        WireExtDiamMM = 0.275
        WireBareDiamMM = 0.226
    end
    if (AWGGauge == 32) then
        WireExtDiamMM = 0.247
        WireBareDiamMM = 0.203
    end
    CoilWireExtDiamMM[I] = WireExtDiamMM
    CoilWireBareDiamMM[I] = WireBareDiamMM
    CoilWireExtDiam[I] = CoilWireExtDiamMM[I] / 25.4
    CoilWireBareDiam[I] = CoilWireBareDiamMM[I] / 25.4
end

-- Calculate the length of all coils.  This will be equal to the integral
-- number of turns of the wire used which best approximates the target length.
-- Note that the target lengths in vector CoilLength[] are replaced by the
-- computed lengths.
for I = 0,5 do
    TargetLength = CoilLength[I]
```

```
    WireDiam = CoilWireExtDiam[I]
    CoilNumTurns[I] = floor(0.5 + (TargetLength/WireDiam))
    CoilLength[I] = CoilNumTurns[I] * WireDiam
end

-- Define all nodes for the surrounding air, in Group(9)
mi_addnode(0,AirLocalRad)              -- Top of local sphere
mi_addnode(0,-AirLocalRad)             -- Bottom of local sphere
mi_addnode(0,AirLocalRad + AirExtDiam)        -- Top of external sphere
mi_clearselected()
mi_selectnode(0,AirLocalRad)
mi_selectnode(0,-AirLocalRad)
mi_selectnode(0,AirLocalRad + AirExtDiam)
mi_setnodeprop('',9)

-- Clarify the (r,z) co-ordinates of the four corners of the windings of
-- all coils.  The armature needs to be handled separately from the others
-- because its outer diameter is limited by the tube size.  The other coils
-- are assumed to be wound directly on the tube's outer surface.
CoilInnerRad = {}
CoilOuterRad = {}
-- For the armature
TotalTubeClearance = 1/16
CoilOuterRad[0] = TubeInnerRad - (0.5*TotalTubeClearance)
CoilInnerRad[0] = CoilOuterRad[0] - (CoilNumLayers[0]*CoilWireExtDiam[0])
CoilPosEndZ[0] = CoilNegEndZ[0] + CoilLength[0]
-- For the other coils
for I = 1,5 do
    CoilInnerRad[I] = TubeOuterRad
    CoilOuterRad[I] = CoilInnerRad[I] + (CoilNumLayers[I]*CoilWireExtDiam[I])
    CoilPosEndZ[I] = CoilNegEndZ[I] + CoilLength[I]
end

-- Define all nodes for all coils.  Note that the Group number = index I.
for I = 0,5 do
    mi_addnode(CoilInnerRad[I],CoilPosEndZ[I])      -- Top inside corner
    mi_addnode(CoilOuterRad[I],CoilPosEndZ[I])      -- Top outside corner
    mi_addnode(CoilOuterRad[I],CoilNegEndZ[I])      -- Bottom outside corner
    mi_addnode(CoilInnerRad[I],CoilNegEndZ[I])      -- Bottom inside corner
    mi_clearselected()
    mi_selectnode(CoilInnerRad[I],CoilPosEndZ[I])
    mi_selectnode(CoilOuterRad[I],CoilPosEndZ[I])
    mi_selectnode(CoilOuterRad[I],CoilNegEndZ[I])
    mi_selectnode(CoilInnerRad[I],CoilNegEndZ[I])
    mi_setnodeprop('',I)
end

-- Define all line segments for the surrounding air, in Group(9)
-- There are two line segments:
-- 1. From the bottom of the local sphere to the top of the local sphere
-- 2. The diameter line across the external sphere
mi_addsegment(0,-AirLocalRad,0,AirLocalRad)
mi_addsegment(0,AirLocalRad,0,AirLocalRad+AirExtDiam)
mi_clearselected()
mi_selectsegment(0,AirLocalRad-0.1)
```

```
mi_selectsegment(0,AirLocalRad+0.1)
mi_setsegmentprop('',0,1,0,9)

-- Define a periodic boundary condition
mi_addboundprop('PeriodicBC',0,0,0,0,0,0,0,0,4)

-- Define all arcs for the surrounding air, in Group(0)
-- There are two arcs:
-- 1. Enclosing the local sphere
-- 2. Enclosing the external sphere
mi_addarc(0,-AirLocalRad,0,AirLocalRad,180,1)
mi_addarc(0,AirLocalRad,0,AirLocalRad+AirExtDiam,180,1)
mi_clearselected()
mi_selectarcsegment(AirLocalRad,0)
mi_selectarcsegment(AirExtDiam/2,AirLocalRad+(AirExtDiam/2))
mi_setarcsegmentprop(1,'PeriodicBC',0,9)

-- Define all line segments for all coils
for I= 0,5 do
    mi_addsegment(CoilInnerRad[I],CoilPosEndZ[I],
        CoilOuterRad[I],CoilPosEndZ[I])
    mi_addsegment(CoilOuterRad[I],CoilPosEndZ[I],
        CoilOuterRad[I],CoilNegEndZ[I])
    mi_addsegment(CoilOuterRad[I],CoilNegEndZ[I],
        CoilInnerRad[I],CoilNegEndZ[I])
    mi_addsegment(CoilInnerRad[I],CoilNegEndZ[I],
        CoilInnerRad[I],CoilPosEndZ[I])
    mi_clearselected()
    mi_selectsegment(CoilInnerRad[I]+0.001,CoilPosEndZ[I])
    mi_selectsegment(CoilOuterRad[I],CoilPosEndZ[I]-0.001)
    mi_selectsegment(CoilOuterRad[I]-0.001,CoilNegEndZ[I])
    mi_selectsegment(CoilInnerRad[I],CoilNegEndZ[I]+0.001)
    mi_setsegmentprop('',0,1,0,I)
end

-- Define a block label for the air, in Group(9)
CentroidRLocal = 0.25
CentroidZLocal = AirLocalRad - 0.25
mi_addblocklabel(CentroidRLocal,CentroidZLocal)
CentroidRExt = 0.25
CentroidZExt = AirLocalRad + (AirExtDiam/2)
mi_addblocklabel(CentroidRExt,CentroidZExt)
mi_clearselected()
mi_selectlabel(CentroidRLocal,CentroidZLocal)
mi_selectlabel(CentroidRExt,CentroidZExt)
mi_getmaterial('Air')
mi_setblockprop('Air',0,0,0,0,9,0)

-- Describe the external region as a Kelvin transformation
mi_defineouterspace(AirLocalRad+(AirExtDiam/2),AirExtDiam/2,AirLocalRad)

-- Define names for the materials of the wires being used
CoilMaterial = {}
for I= 0,5 do
    CoilMaterial[I] = CoilName[I] .. "Wire"
```

```
end

-- Define new materials for the enameled copper wire being used
-- Relative permeability in r-direction = 1
-- Relative permeability in z-direction = 1
-- Permanent magnet coercivity = 0
-- Applied source current density = 0
-- Electrical conductivity = 58 MS/m for copper wire
-- Lamination thickness = 0
-- Hysteresis lag angle = 0
-- Lamination fill fraction = 1
-- Lamination type = 3 (This code identifies magnet wire)
-- Hysteresis lag angle in the x-direction = 0
-- Hysteresis lag angle in the y-direction = 0
-- Number of strands in wire = 1
-- Diameter of wire (in millimeters) has already been specified above.
-- For the wire-wound coils
for I= 0,5 do
    mi_addmaterial(CoilMaterial[I],1,1,0,0,58,
        0,0,1,3,0,0,1,CoilWireExtDiamMM[I])
end

-- Define block labels for all coils.  Set the current to 1A for
-- initialization purposes only.
Current = 1
for I= 0,5 do
    CentroidR = (CoilInnerRad[I] + CoilOuterRad[I]) / 2
    CentroidZ = (CoilPosEndZ[I] + CoilNegEndZ[I]) / 2
    mi_addcircprop(CoilName[I],Current,1)
    mi_addblocklabel(CentroidR,CentroidZ)
    mi_clearselected()
    mi_selectlabel(CentroidR,CentroidZ)
    TotalNumTurns = CoilNumLayers[I] * CoilNumTurns[I]
    mi_setblockprop(CoilMaterial[I],0,MaxMeshSize,CoilName[I],0,I,TotalNumTurns)
end

--////////////////////////////////////////////////////////////////////////
--// Now that the geometry has been specified, we can save the construction
--// in a temporary file which will be a sister file to this Lua script.
--// Then, we can get ready to do some analysis.
--////////////////////////////////////////////////////////////////////////
mi_saveas("./temp.fem")          -- Save the geometry
main_maximize()                  -- Maximize the main FEMM window
mi_zoomnatural()                 -- Zoom the display for the best fit
showconsole()                    -- Show the Lua output window
clearconsole()                   -- Clear the Lua output window
mi_seteditmode("group")          -- Make edits such as translation by Group

--////////////////////////////////////////////////////////////////////////
--// Analyze the coils independently
--// We will power up each coil separately to calculate its Ohmic resistance
--// and self-inductance.  While we are at it, we will calculate the mutual
--// inductance between this coil and all the other coils.  We will do this
--// for all coils, even though there will be (or should be) redundancy
--// between corresponding pairs and even though not all coils may be
```

```
--// included in the simulation.
--////////////////////////////////////////////////////////////////////////////
for I= 0,5 do
    -- Zero out all currents, for all coils, including the powered coil
    for J= 0,5 do
        mi_modifycircprop(CoilName[J],1,0)
    end
    -- Set the current in the powered coil to 100A
    mi_modifycircprop(CoilName[I],1,100)
    -- Analyze the problem
    mi_analyze()
    mi_loadsolution()
    -- For the powered coil, val1=current, val2=voltage and val3=flux
    val1,val2,val3 = mo_getcircuitproperties(CoilName[I])
    CoilRfemm[I] = val2 / val1          -- Ohmic resistance
    CoilLfemm[I] = val3 / val1          -- Self-inductance
    -- For each other coil, val4=current, val5=voltage and val6=flux
    for J= 0,5 do
        if (J ~= I) then
            val4,val5,val6 = mo_getcircuitproperties(CoilName[J])
            CoilMfemm[I][J] = val6 / val1
        end
    end
end

-- Display the interim results in the Lua window
for I= 0,5 do
    print(CoilName[I])
    print("R=",CoilRfemm[I])
    print("L(uH)=",CoilLfemm[I]*1000000)
    for J= 0,5 do
        if (J ~= I) then
            print("CoilM(uH) ",I," to ",J," = ",CoilMfemm[I][J]*1000000)
        end
    end
end

--////////////////////////////////////////////////////////////////////////////
--// There are a large number of variables which are best suited for
--// representation as vectors.  Some of these are fixed parameters, some
--// are initial conditions and some are temporary variables which will be
--// needed during the simulation.  I will try to list these new vectors
--// in that order.
--////////////////////////////////////////////////////////////////////////////

-- Additional resistance in each coil circuit.  No harm is done if we
-- define some extra resistance even for coils which are not used during
-- a particular run.
CoilRextra = {}
CoilRextra[0] = 0.0001    -- Solder joint of two layers in armature
CoilRextra[1] = 0.02      -- SCR
CoilRextra[2] = 0.02      -- SCR
CoilRextra[3] = 0.02      -- SCR
CoilRextra[4] = 0.02      -- SCR
CoilRextra[5] = 0.02      -- SCR
```

```
-- Total resistance of each coil circuit.  The total resistance is the Ohmic
-- resistance computed by FEMM plus the extra resistance just defined.
CoilR = {}
for I= 0,5 do
    CoilR[I] = CoilRfemm[I] + CoilRextra[I]
end

-- Capacitance in each coil circuit.  There will never be any capacitance in
-- the armature circuit, but no harm is done if we define a zero-value
-- capacitance anyway.
Cap = {}
Cap[0] = 0
Cap[1] = 0.000032
Cap[2] = 0.000032
Cap[3] = 0.000032
Cap[4] = 0.000032
Cap[5] = 0.000032

-- These "offsets" control the times at which the switches (SCRs) are closed
-- for each coil circuit.  The offsets are distances measured along the z-
-- axis.  The switch which controls Coil #I will close when the leading edge
-- of the armature reaches StartOffset[I].  Since the armature circuit is
-- always closed, its StartOffset[0] is irrelevant, and is set to a big
-- negative distance just so its usage is consistent with the others.  The
-- offset for barrel coils #2 and later is set by default to one-half inch
-- past the positive end of the coil.  Since the armature has a length of
-- about one inch, this will start these later coils when the armature is
-- just about centered on the downstream lip of the coil.  The offset for
-- the first barrel can be used in two different ways, depending on the
-- setting of the CoilConnect[1] variable, which see.  If CoilConnect[1] is
-- set to one, then the switch for barrel coil #1 will be closed at time t=0
-- when the simulation starts.  If that is the case, the starting offset is
-- simply not relevant.  On the other hand, if CoilConnect[1] is set to zero,
-- then barrel coil #1 will start up on the same condition as the others,
-- when the leading edge of the armature reaches the given z-value.  That
-- only makes sense, however, if the armature has some positive speed at the
-- start, which will carry it forward to the trigger point.
StartOffset = {}
StartOffset[0] = -1000                  -- Armature circuit is always closed
StartOffset[1] = CoilPosEndZ[1] + 0.5   -- Start barrel coil #1 here
StartOffset[2] = CoilPosEndZ[2] + 0.5   -- Start barrel coil #2 here
StartOffset[3] = CoilPosEndZ[3] + 0.5   -- Start barrel coil #3 here
StartOffset[4] = CoilPosEndZ[4] + 0.5   -- Start barrel coil #4 here
StartOffset[5] = CoilPosEndZ[5] + 0.5   -- Start barrel coil #5 here

-- Declare which coil circuits are closed, i.e., "connected", at the start of
-- the simulation.  These must not conflict with the distances in the
-- StartOffset[] vector since these variables are used as Boolean flags to
-- tell the procedure which coils are in operation at the start of any
-- particular time step.  The default settings assume the gun starts up from
-- rest, with the switch for barrel coil #1 closed at the commencement of the
-- simulation.
CoilConnect = {}
CoilConnect[0] = 1
```

```
CoilConnect[1] = 1
CoilConnect[2] = 0
CoilConnect[3] = 0
CoilConnect[4] = 0
CoilConnect[5] = 0

-- The CircuitMode[] states whether the diode in a particular barrel coil
-- circuit is reverse-biased or forward-biased.  This variable is only
-- used for barrel coils which are in operation, i.e., CoilConnect[I] = 1.
-- The flags are "R" for reverse-bias and "F" for forward-bias.  Since all
-- barrel coils start of with a charged capacitor, the CircuitMode is
-- initialized to "R".
CircuitMode = {}
CircuitMode[0] = "X"      -- Not relevant for the armature
CircuitMode[1] = "R"
CircuitMode[2] = "R"
CircuitMode[3] = "R"
CircuitMode[4] = "R"
CircuitMode[5] = "R"

-- Vdiode[] is the forward voltage drop over the diode when it is conducting
Vdiode = {}
Vdiode[0] = 0             -- Not relevant for the armature
Vdiode[1] = 1.5
Vdiode[2] = 1.5
Vdiode[3] = 1.5
Vdiode[4] = 1.5
Vdiode[5] = 1.5

-- Initial voltages on the capacitors.  Only non-zero capacitors should be given
-- non-zero initial voltages.
Vcap_0 = {}
Vcap_0[0] = 0
Vcap_0[1] = 5000
Vcap_0[2] = 0
Vcap_0[3] = 0
Vcap_0[4] = 0
Vcap_0[5] = 0

-- Instantaneous voltage drops over capacitors.  The following line only
-- declares the vector.  We do not need to insert specific values at this
-- time.
Vcap = {}

-- Initial electrical charge stored in the capacitors
Q_0 = {}
for I= 1,5 do
    Q_0[I] = Cap[I] * Vcap_0[I]
end

-- Instantaneous electrical charge stored in the capacitors
Q_start = {}
Q_end = {}

-- Initial currents flowing in the coils.  These will be zero for all coils if
```

```
-- the simulation starts up from rest.  However, we can use a non-zero initial
-- current to "start" the armature without having to go through a complete
-- start-up cycle.
I_0 = {}
I_0[0] = 0
I_0[1] = 0
I_0[2] = 0
I_0[3] = 0
I_0[4] = 0
I_0[5] = 0

-- Instantaneous currents flowing through the coils.  The following lines only
-- declare the vectors.  We do not need to insert specific values at this time.
I_start = {}
I_end = {}

-- Instantaneous first derivatives of the currents flowing through the coils.
-- The following line only declares the vectors.  The numerical integration
-- procedure assumes these derivatives remain constant throughout each time
-- step, so they are given the subscript "constant".
dIdt_const = {}

-- Spatial derivative of the mutual inductances.  Since this is only relevant
-- when one of each pair of coils is the armature, we can get away with using
-- a vector instead of an array.
dMdz = {}

-- deltaER[] is the energy burned off as heat by Ohmic resistance during one
-- time step.  This is recorded for each separate coil circuit at the end of
-- each time step.  ERCum[] is the cumulative energy burned off by the
-- resistances since the start of the simulation.  A separate total is kept
-- for each resistance.  ERCum[] is not printed out in version of the script,
-- but can be if it is useful to know the resistance loss in different barrel
-- coils, or the armature.
deltaER = {}
ERCum = {}

-- deltaED[] is the energy burned off as heat by the diodes during one time
-- step.  This is recorded for each separate diode at the end of each time
-- step.  EDCum[] is the cumulative energy burned off by the diodes since
-- the start of the simulation.  A separate total is kept for each diode.
-- EDCum[] is not printed out in this version of the script, but can be if it
-- is useful to track the diode loss in different barrel coils.
deltaED = {}
EDCum = {}

-- Instantaneous stores of energy -- kinetic, electrostatic, self-inductive,
-- mutual inductive and cumulative Ohmic heat.  These are the values at the
-- START of each time step.
EK = 0
ECTotal = 0
ELTotal = 0
EMTotal = 0
ERTotal = 0
EDTotal = 0
```

```
ETotal = 0
EError = 0

-- Initial stores of energy.  These will be calculated at the start of the
-- first iteration in the simulation.
EK_0 = 0
ECTotal_0 = 0
ELTotal_0 = 0
EMTotal_0 = 0
ERTotal_0 = 0
EDTotal_0 = 0
ETotal_0 = 0

-- Define certain other scalar parameters and variables (for completeness only)
-- Force_const            -- Force on armature, constant during time step
-- Speed_start            -- Speed of armature at start of time step
-- ArmPosEndZ_start       -- Location of armature's positive end at start of ts
-- Distance_start         -- Cumulative distance travelled at start of time step

--////////////////////////////////////////////////////////////////////////////
--// Initialize working variables for the simulation.  This involves setting
--// the working variables to their proper values for time Time=0.
--////////////////////////////////////////////////////////////////////////////
ArmPosEndZ_end = CoilPosEndZ[0]
Speed_end = Speed_0 -- Speed of armature at end of time step
Distance_end = 0    -- Cumulative distance travelled at end of time step
for I= 0,5 do
    Q_end[I] = Q_0[I]
    I_end[I] = I_0[I]
end

--////////////////////////////////////////////////////////////////////////////
--// Open a text file for output and write a short header
--////////////////////////////////////////////////////////////////////////////
handle = openfile("./Induction_Gun_Results.txt","a")
write(handle,"\nFiring an induction gun with:\n")
write(handle,"  TimeStep = ",deltaT*1000000," micro-seconds\n")
write(handle,"  Armature starting conditions:\n")
write(handle,"    Positive end location = ",ArmPosEndZ_0," inches\n")
write(handle,"    Speed = ",Speed_0," m/s\n")
write(handle,"    Current = ",I_0[0]," Amperes\n")
write(handle,"  FEMM maximum mesh size = ",MaxMeshSize,"\n")
write(handle,"  FEMM maximum residual = ",MaxResidual,"\n")

-- Write the self-inductances and the resistances
for I= 0,5 do
    write(handle,"NumLayers[",I,"] = ",CoilNumLayers[I],"\n")
    write(handle,"NumTurns[",I,"] = ",CoilNumTurns[I],"\n")
    write(handle,"CoilTrueLength[",I,"](inch) = ",CoilLength[I],"\n")
    write(handle,"L[",I,"](uH) = ",CoilLfemm[I]*1000000,"\n")
    write(handle,"R[",I,"](Ohms) = ",CoilR[I],"\n")
end

-- Write the mutual inductances (upper triangle only)
for I= 0,4 do
```

```
    for J= (I+1),5 do
        write(handle,"M[",I,"][",J,"] = ",CoilMfemm[I][J],"\n")
    end
end

-- Write the starting values for Time=0 in the same order that they will
-- be written at the end of every time step.  This will be the first row
-- in the listing of the simulation results.  Note that only the mutual
-- inductances with respect to the armature are written.  Also note that
-- zero results will be written for all coils, even if they do not take
-- part in the simulation.  This keeps all columns aligned when coil
-- circuits happen to start up or shut down during the simulation.
write(handle,"Time(us)=,",0,",")
for I= 0,5 do
    write(handle,"I_0[",I,"](A)=,",I_0[I],",")
    write(handle,"CircuitMode[",I,"]=,",CircuitMode[I],",")
end
for I= 1,5 do
    write(handle,"MA_start[",I,"](uH)=,",CoilMfemm[0][I]*1000000,",")
end
for I= 1,5 do
    write(handle,"Q_0[",I,"](C)=,",Q_0[I],",")
    write(handle,"Vcap_0[",I,"](V)=,",Vcap_0[I],",")
end
write(handle,"Force_const(N)=,",0,",")
write(handle,"ArmPosEndZ_end(inch)=,",ArmPosEndZ_end,",")
write(handle,"Distance_end(mm)=,",Distance_end,",")
write(handle,"Speed_end(m/s)=,",Speed_end,",")
write(handle,"EK(J)=,,")
write(handle,"ECTotal(J)=,,")
write(handle,"ELTotal(J)=,,")
write(handle,"EMTotal(J)=,,")
write(handle,"ERTotal(J)=,,")
write(handle,"EDTotal(J)=,,")
write(handle,"ETotal(J)=,,")
write(handle,"EError(J)=,\n")
SaveCounter = 0

--////////////////////////////////////////////////////////////////////////
--// This is the simulation's main loop through time
--////////////////////////////////////////////////////////////////////////
for TimeStep = 1,MaxNumTimeSteps do
    Time = TimeStep * deltaT

    -- Step #1: Bring forward the values from the end of the previous time step
    Speed_start = Speed_end
    Distance_start = Distance_end
    ArmPosEndZ_start = ArmPosEndZ_end
    for I= 0,5 do
        Q_start[I] = Q_end[I]
        I_start[I] = I_end[I]
    end

    -- Step #2: Use FEMM to calculate the mutual inductances in the new location.
    -- Note that the armature will already have been moved to its new location.
```

```
-- Set the current in the armature to 100A
mi_modifycircprop(CoilName[0],1,100)
-- Zero out the currents in all coils other than the armature
for J= 1,5 do
    mi_modifycircprop(CoilName[J],1,0)
end
-- Analyze the problem
mi_analyze()
mi_loadsolution()
-- For the armature, val1=current, val2=voltage and val3=flux
val1,val2,val3 = mo_getcircuitproperties(CoilName[0])
-- For each other coil, val4=current, val5=voltage and val6=flux
-- No harm is done if the mutual inductance is calculated for all coils,
-- whether they are in operation or not.
for I= 1,5 do
    val4,val5,val6 = mo_getcircuitproperties(CoilName[I])
    CoilMfemm[0][I] = val6 / val1
    CoilMfemm[I][0] = CoilMfemm[0][I]
end

-- Step #3: Move the armature to a slightly different location and
-- calculate the mutual inductances again.  Use the difference to
-- calculate the spatial derivative of the mutual inductances.  The
-- spatial derivatives will be in units of Henries per meter.
-- Step #3A: Translate the armature to its joggled location
Joggle = 0.02       -- Joggle distance in inches
mi_seteditmode("group")
mi_clearselected()
mi_selectgroup(0)
mi_movetranslate(0,Joggle)
-- Step #3B: Calculate the new mutual inductances.  Keep the same currents
-- as used in Step #2.
mi_analyze()
mi_loadsolution()
-- For the armature, val1=current, val2=voltage and val3=flux
val1,val2,val3 = mo_getcircuitproperties(CoilName[0])
-- For each other coil, val4=current, val5=voltage and val6=flux
for I= 1,5 do
    val4,val5,val6 = mo_getcircuitproperties(CoilName[I])
    NewMfemm = val6 / val1
    dMdz[I] = (NewMfemm - CoilMfemm[0][I]) / (Joggle * 2.54 / 100)
end

-- Step #4: Use FEMM to calculate the force in the new configuration
-- Step #4A: Move the armature back to its location before the joggle
mi_seteditmode("group")
mi_clearselected()
mi_selectgroup(0)
mi_movetranslate(0,-Joggle)
-- Step #4B: Re-set the currents to their values at start of time step
for I= 0,5 do
    mi_modifycircprop(CoilName[I],1,I_start[I])
end
-- Calculate the force on the armature,  which is Group(0)
mi_analyze()
```

```
mi_loadsolution()
mo_groupselectblock(0)
Force_const = mo_blockintegral(19)

-- Step #5: This is a convenient place to calculate the stored magnetic energy
-- at the START of the time step.  Only include coils in operation.  If this
-- happens to be the first time step, then make a note of the initial self-
-- inductive energy.
ELTotal = 0
for I= 0,5 do
    if (CoilConnect[I] == 1) then
        ELTotal = ELTotal + (0.5 * CoilLfemm[I] * I_start[I] * I_start[I])
    end
end
if (TimeStep == 1) then
    ELTotal_0 = ELTotal
end

-- Step #6: Calculate the magnetic energy stored in the mutual inductances at
-- the START of the time step
EMTotal = 0
-- Deal with interactions with the armature first
for I= 1,5 do
    if (CoilConnect[I] == 1) then
        EMTotal = EMTotal + (CoilMfemm[0][I] * I_start[0] * I_start[I])
    end
end
-- Deal with all other interactions
for I= 1,4 do
    if (CoilConnect[I] == 1) then
        for J= (I+1),5 do
            if (CoilConnect[J] == 1) then
                EMTotal = EMTotal +
                    (CoilMfemm[I][J] * I_start[I] * I_start[J])
            end
        end
    end
end
if (TimeStep == 1) then
    ELTotal_0 = ELTotal
end

-- Step #7: Calculate the armature's kinetic energy at the START of the time step
EK = 0.5 * Mass * Speed_start * Speed_start
if (TimeStep == 1) then
    EK_0 = EK
end

-- Step #8: Calculate the electrostatic energy stored in all capacitors at the
-- START of the time step.
ECTotal = 0
for I= 1,5 do
    if (Cap[I] == 0) then
        Vcap[I] = 0
    else
```

```
        Vcap[I] = Q_end[I] / Cap[I]
        ECTotal = ECTotal + (0.5 * Cap[I] * Vcap[I] * Vcap[I])
    end
end
if (TimeStep == 1) then
    ECTotal_0 = ECTotal
end

-- Step #9: Calculate the total heat energy by the START of the time step.
-- This is the awkward calculation because it uses the heat burned off during
-- the previous time step, which was stored at that time in the variable
-- deltaER[], with one vector element per coil circuit.  It is necessary to do
-- an energy balance at the start of each time step, or at the end.  Since the
-- magnetic energy is much easier to calculate at the start of each time step,
-- when the mutual inductances have just been re-calculated, that is the point
-- in time when we will do the energy balance calculation.
if (TimeStep == 1) then
    for I= 0,5 do
        deltaER[I] = 0
        deltaED[I] = 0
        ERCum[I] = 0
        EDCum[I] = 0
    end
    ERTotal = 0
    ERTotal_0 = 0
    EDTotal = 0
    EDTotal_0 = 0
else
    for I= 0,5 do
        ERTotal = ERTotal + deltaER[I]
        EDTotal = EDTotal + deltaED[I]
    end
end

-- Step #10: Calculate the total system energy, and the energy error, at the
-- START of the time step.
ETotal = EK + ECTotal + ELTotal + EMTotal + ERTotal + EDTotal
if (TimeStep == 1) then
    ETotal_0 = ETotal
end
EError = ETotal - ETotal_0

-- Step #11: Construct the inductance matrix A[][].  Only circuits which
-- are connected at this time are included in the matrix.  Note that all
-- elements are algebraically positive.
MatrixRow = 0
for I= 0,5 do
    if (CoilConnect[I] == 1) then
        MatrixRow = MatrixRow + 1
        MatrixColumn = 0
        for J= 0,5 do
         if (J == I) then
             MatrixColumn = MatrixColumn + 1
             A[MatrixRow][MatrixColumn] = CoilLfemm[I]
         else
```

```
                if (CoilConnect[J] == 1) then
                 MatrixColumn = MatrixColumn + 1
                    A[MatrixRow][MatrixColumn] = CoilMfemm[I][J]
                end
            end
        end
    end
end
MatrixOrder = MatrixRow

-- Step #12: Load the constant vector B[].  Note that the armature's
-- circuit is a little different from the others.  The matrix equation
-- is the same for circuits in reverse-biased and forward-biased modes.
-- Step #12A: Load B[0] for the armature
B[1] = -CoilR[0] * I_start[0]
for I= 1,5 do
    if (CoilConnect[I] == 1) then
        B[1] = B[1] - (Speed_start * dMdz[I] * I_start[I])
    end
end
-- Step #12B: Load B[] for the other coils
MatrixRow = 1
for I= 1,5 do
    if (CoilConnect[I] == 1) then
        MatrixRow = MatrixRow + 1
        B[MatrixRow] =
            (-Speed_start * dMdz[I] * I_start[0]) +
            (-CoilR[I] * I_start[I]) +
         (Q_start[I] / Cap[I])
    end
end
-- Step #12C: If desired for debugging purposes, uncomment the following
-- two lines to write the matrix equation to the output text file.
--write(handle,"Matrix equation before inversion\n")
--WRITEMATRIXEQUATION(MatrixOrder)

-- Step #13: Solve the matrix equation
INVERT(MatrixOrder)

-- Step #14: Transfer the solution to the appropriate "constant-slope"
-- variable
dIdt_const[0] = X[1]
SolutionIndex = 1
for I= 1,5 do
    if (CoilConnect[I] == 1) then
        SolutionIndex = SolutionIndex + 1
        dIdt_const[I] = X[SolutionIndex]
    end
end

-- Step #15: Do the first-order integrations, but only for the circuits
-- which are in operation
for I= 0,5 do
    if (CoilConnect[I] == 1) then
        I_end[I] = I_start[I] + (dIdt_const[I] * deltaT)
```

```
    else
        -- No current will flow through unconnected coils
        I_end[I] = 0
    end
end

-- Step #16: Do the second-order integration, but only for those
-- circuits which have capacitors, are connected and have not yet
-- entered their forward-biased modes.
for I= 1,5 do
    if ((CoilConnect[I] == 1) and (CircuitMode[I] == "R")) then
        Q_end[I] = Q_start[I] +
            (-I_start[I] * deltaT) +
            (-0.5 * dIdt_const[I] * deltaT * deltaT)
    else
        -- Hold over the charge on all other capacitors, including
        -- those in forward-biased mode.
        Q_end[I] = Q_start[I]
    end
end

-- Step #17: Calculate the armature's kinematics
Accel_const = Force_const / Mass
Speed_end = Speed_start  + (Accel_const * deltaT)
Distance_end = Distance_start +
    (Speed_start * deltaT) +
    (0.5 * Accel_const * deltaT * deltaT)

-- Step #18: Convert the change in distance (meters) to location (inches)
ArmPosEndZ_end = ArmPosEndZ_start +
    ((Distance_end - Distance_start) * 100 / 2.54)

-- Step #19: Translate the armature to its new location
mi_seteditmode("group")
DesiredLELocation = ArmPosEndZ_end
RequiredTranslation = DesiredLELocation - ArmPosEndZ_start
mi_clearselected()
mi_selectgroup(0)
mi_movetranslate(0,RequiredTranslation)
CurrentLELocation = ArmPosEndZ_end

-- Step #20: Calculate the energy consumed by the resistors during this time
-- step, but only for circuits which are operating.  These values will be
-- held over until the start of the next time step, when the energy balances
-- are reconciled.
for I= 0,5 do
    if (CoilConnect[I] == 1) then
        deltaER[I] = CoilR[I] * deltaT * (
            (I_start[I] * I_start[I]) +
            (I_start[I] * dIdt_const[I] * deltaT) +
            (dIdt_const[I] * dIdt_const[I] * deltaT * deltaT / 3))
    else
       -- No current will flow through unconnected coils
        deltaER[I] = 0
    end
```

```
    ERCum[I] = ERCum[I] + deltaER[I]
end

-- Step #21: Calculate the energy consumed by the diodes during this time
-- step, but only for circuits which are operating and in forward-biased
-- mode.  These energies will be held over until the start of the next time
-- step, when the energy balances are reconciled.
for I= 1,5 do
    if ((CoilConnect[I] == 1) and (CircuitMode == "F")) then
        deltaED[I] = Vdiode[I] * deltaT * (
            (Istart[I] + (0.5 * dIdt_const[I] * deltaT)))
    else
        -- No current will flow through reverse-biased diodes
        deltaED[I] = 0
    end
    EDCum[I] = EDCum[I] + deltaED[I]
end

-- Step #22: Calculate the voltage drops over all capacitors.  If a particular
-- capacitance is zero, set its voltage drop to zero.
for I= 0,5 do
    if (Cap[I] == 0) then
        Vcap[I] = 0
    else
        Vcap[I] = Q_end[I] / Cap[I]
   end
end

-- Step #23: Write the interim results to the output text file
SaveCounter = SaveCounter + 1
if (SaveCounter >= SaveInterval) then
    SaveCounter = 0
    write(handle,"Time(us)=,",Time * 1000000,",")
    for I= 0,5 do
        write(handle,"I_end[",I,"](A)=,",I_end[I],",")
        write(handle,"CircuitMode[",I,"]=,",CircuitMode[I],",")
   end
    for I= 1,5 do
        write(handle,"MA_start[",I,"](uH)=,",CoilMfemm[0][I]*1000000,",")
    end
    for I= 1,5 do
        write(handle,"Q_end[",I,"](C)=,",Q_end[I],",")
        write(handle,"Vcap_end[",I,"](V)=,",Vcap[I],",")
    end
    write(handle,"Force_const(N)=,",Force_const,",")
    write(handle,"ArmPosEndZ_end(inch)=,",ArmPosEndZ_end,",")
    write(handle,"Distance_end(mm)=,",Distance_end,",")
    write(handle,"Speed_end(m/s)=,",Speed_end,",")
    write(handle,"KE_start(J)=,",EK,",")
    write(handle,"ECTotal_start(J)=,",ECTotal,",")
    write(handle,"ELTotal_start(J)=,",ELTotal,",")
    write(handle,"EMTotal_start(J)=,",EMTotal,",")
    write(handle,"ERTotal_start(J)=,",ERTotal,",")
    write(handle,"EDTotal_start(J)=,",EDTotal,",")
    write(handle,"ETotal_start(J)=,",ETotal,",")
```

```
        write(handle,"EError_start(J)=,",EError,"\n")
    end

    -- Step #24: Terminate the simulation if the armature has passed through
    -- the last barrel coil by at least two inches.  Obviously, this
    -- termination condition can be adjusted to suit before beginning any
    -- particular simulation.
    TerminalLELocation = CoilPosEndZ[2] + 2
    if (ArmPosEndZ_end >= TerminalLELocation) then
        TimeStep = MaxNumTimeSteps + 1
    end

    -- Step #25: Check all operating coils which are in reverse-biased mode,
    -- checking to see whether the capacitor has discharged.  If so, the
    -- forward-biased mode should begin.
    for I= 1,5 do
        if ((CoilConnect[I] == 1) and (CircuitMode[I] == "R")) then
           if (Vcap[I] <= -Vdiode[I]) then
                CircuitMode[I] = "F"
            end
        end
    end

    -- Step #26: Turn on any coil circuit whose starting time has arrived.
    -- Note that coil circuits which have completed their operation will
    -- have CoilConnect[] =-1.  Do not restart them.
    for I= 1,5 do
        if ((CoilConnect[I] == 0) and (ArmPosEndZ_end >= StartOffset[I])) then
            CoilConnect[I] = 1
            Q_end[I] = Q_0[I]
            I_end[I] = 0
        end
    end

    -- Step #27: Turn off any operating coil circuit controlled by an SCR
    -- whose current flow has become negative.  Set CoilConnect[] = -1 to
    -- avoid inadvertently turning this coil back on.
    for I= 1,5 do
        if (CoilConnect[I] == 1) then
            if (I_end[I] < 0) then
                CoilConnect[I] = -1
            end
        end
    end

    -- Step #28: Display a message to the user in the Lua window
    print("Time(us)=",Time * 1000000)
    print("IArm(A)=",I_end[0],"  Icoil#1(A)=",I_end[1])
    print("Force(N)=",Force_const,"  Mode=",CircuitMode[1])
    print("ArmPosEndZ(inch)=",ArmPosEndZ_end)
    print("Speed_end(m/s)=",Speed_end)

end

-- Close out the Lua program
```

```
closefile(handle)
mo_close()
mi_close()
messagebox("All done.")
```