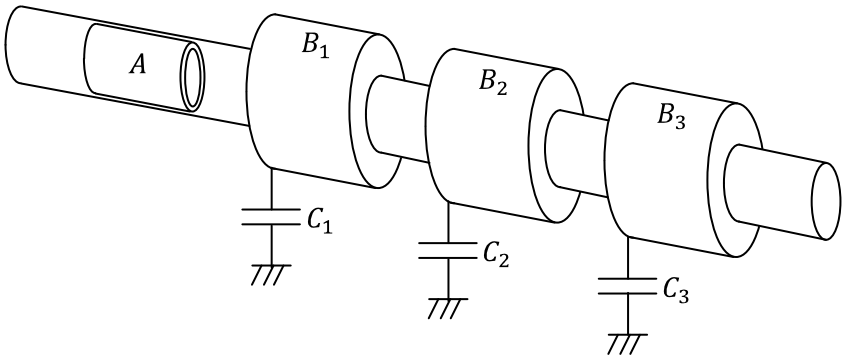
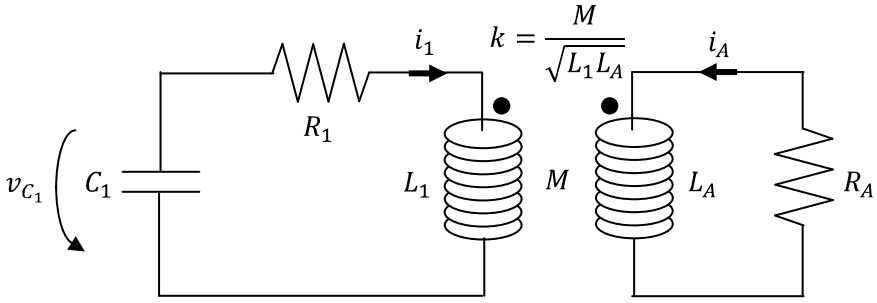
Using FEMM to design an induction coil gun

In this paper, I will look into induction-based coil guns. An induction-based coil gun should not be confused with a reluctance-based coil gun. In a reluctance gun, the field coils not only provide the magnetic field which accelerates the projectile, they also generate the magnetic field inside the projectile itself. It is not an easy task to set things up so the field coils can perform this double duty efficiently. In an induction coil gun, the projectile carries a current and generates its own magnetic field.

The following figure shows the essential components. The projectile is a cylindrical air-core solenoid. By convention, it is called the armature ("A"). The figure shows three field coils (also air-core solenoids) wound around the barrel. I will generally call the field coils "barrel coils" and have labeled them in the figure as B_1, B_2 and B_3. I have also given each coil its own driving capacitor. They are labeled C_1, C_2 and C_3. The coils will be powered up sequentially as the armature travels down the barrel.



Let's begin by looking at a simplified case involving only one barrel coil. I will set time $t = 0$ as the instant when the switch controlling the barrel coil's circuit is closed. I will assume that its capacitor is initially charged up to some voltage V_o and that both circuits are initially at rest. The following schematic diagram shows the two circuits, one for the barrel coil and one for the armature.



The components in the barrel coil's circuit all have subscript "1" to denote that they constitute barrel coil #1. The components in the armature's coil bear subscript A. I will use the convention that currents flowing into the dotted end of an inductor produce positive magnetic flux on the dotted ends of coupled coils. Here, the mutual inductance between the barrel coil and the armature is M. There are two separate circuits. The sum of the voltage drops around each circuit must be zero. The two circuit equations are:

$$\text{Barrel coil:} \quad V_o - \frac{1}{C_1}\int i_1 dt = R_1 i_1 + \frac{d}{dt}(L_1 i_1 + M i_A) \qquad (1A)$$

$$\text{Armature:} \quad \frac{d}{dt}(L_A i_A + M i_1) + R_A i_A = 0 \qquad (1B)$$

Let's overcome the integral in the first equation by making reference to the charge q_1 stored in capacitor C_1 at any instant in time. The very definition of the current flowing through the barrel coil's circuit is that:

$$i_1 = -\frac{dq_1}{dt} \quad (2)$$

where the minus sign reflects the fact that positive current flows when the charge in the capacitor is decreasing. We can substitute this definition into both circuit equations to get:

$$\text{Barrel coil:} \quad \frac{q_1}{C_1} = -R_1\frac{dq_1}{dt} + \frac{d}{dt}\left(-L_1\frac{dq_1}{dt} + Mi_A\right) \quad (3A)$$

$$\text{Armature:} \quad \frac{d}{dt}\left(L_A i_A - M\frac{dq_1}{dt}\right) + R_A i_A = 0 \quad (3B)$$

Note that the starting voltage V_o drops out. q_1 is the instantaneous charge stored in the capacitor. It takes into account both the starting voltage and the history of the current since the starting time. Apply the product rule to expand the derivatives in these equations. I will assume that the self-inductances L_1 and L_A are constant in time so their derivatives vanish. But the mutual inductance *M* is not constant.

$$\frac{q_1}{C_1} = -R_1\frac{dq_1}{dt} - L_1\frac{d^2q_1}{dt^2} + M\frac{di_A}{dt} + i_A\frac{dM}{dt} \quad (4A)$$

$$L_A\frac{di_A}{dt} - M\frac{d^2q_1}{dt^2} - \frac{dM}{dt}\frac{dq_1}{dt} + R_A i_A = 0 \quad (4B)$$

Two independent variables appear in these two equations: (i) the charge q_1 in the capacitor and (ii) the current i_A in the armature circuit. Rather than just combine the equations directly, I am going to write them in matrix format. One way to get there is to first re-arrange the two equations as follows.

$$L_1\frac{d^2q_1}{dt^2} - M\frac{di_A}{dt} = -R_1\frac{dq_1}{dt} - \frac{q_1}{C_1} + i_A\frac{dM}{dt} \quad (5A)$$

$$-M\frac{d^2q_1}{dt^2} + L_A\frac{di_A}{dt} = -R_A i_A + \frac{dq_1}{dt}\frac{dM}{dt} \quad (5B)$$

In matrix form, then:

$$\begin{bmatrix} L_1 & -M \\ -M & L_A \end{bmatrix} \cdot \begin{bmatrix} \frac{d^2q_1}{dt^2} \\ \frac{di_A}{dt} \end{bmatrix} = \begin{bmatrix} -R_1\frac{dq_1}{dt} - \frac{q_1}{C_1} + i_A\frac{dM}{dt} \\ -R_A i_A + \frac{dq_1}{dt}\frac{dM}{dt} \end{bmatrix} \quad (6)$$

If we could somehow invert the leading matrix of inductances, the solution could be written down as:

$$\begin{bmatrix} \frac{d^2q_1}{dt^2} \\ \frac{di_A}{dt} \end{bmatrix} = \begin{bmatrix} L_1 & -M \\ -M & L_A \end{bmatrix}^{-1} \cdot \begin{bmatrix} -R_1\frac{dq_1}{dt} - \frac{q_1}{C_1} + i_A\frac{dM}{dt} \\ -R_A i_A + \frac{dq_1}{dt}\frac{dM}{dt} \end{bmatrix} \quad (7)$$

In fact, the inverse of this 2×2 inductance matrix can be written down by inspection, as follows:

$$\begin{bmatrix} L_1 & -M \\ -M & L_A \end{bmatrix}^{-1} = \frac{1}{L_1 L_A - M^2} \begin{bmatrix} L_A & M \\ M & L_1 \end{bmatrix} \quad (8)$$

When we extend the physical configuration to include more barrel coils, we will find the matrix format a very convenient way to keep track of things. It will be useful to expand the vector on the right-hand side of Equation (7) to separate out the rate-of-change of the mutual inductance.

$$\begin{bmatrix} \frac{d^2 q_1}{dt^2} \\ \frac{di_A}{dt} \end{bmatrix} = \frac{1}{L_1 L_A - M^2} \begin{bmatrix} L_A & M \\ M & L_1 \end{bmatrix} \cdot \left\{ \begin{bmatrix} -R_1 \frac{dq_1}{dt} - \frac{q_1}{C_1} \\ -R_A i_A \end{bmatrix} + \frac{dM}{dt} \begin{bmatrix} i_A \\ \frac{dq_1}{dt} \end{bmatrix} \right\} \quad (9)$$

If we knew dM/dt, or knew how to get it, then Equation (9) would be an ideal basis for numerical integration. At the start of every time step, all of the values on the right-hand side of the equation would be known. The charge q_1 and its derivative dq_1/dt, and the armature current i_A, would all have been calculated at the end of the previous time step. During the previous time step, we would also have calculated how far the armature moved. We could now relocate it to its new location and use FEMM to calculate the new mutual inductance M.

All that prevents us from doing that is the rate-of-change of the mutual inductance dM/dt. If the armature was prevented from moving, then the mutual inductance would not change with time and this derivative would be identically equal to zero. That the mutual inductance changes with time is due solely to the fact that the armature does move. In fact, the independent variable which governs the change of the mutual inductance is not time *per se*, but rather the location of the armature.

In our FEMM model, the location of the armature is represented by variable z. This is the location of a reference point on the armature (say, its leading edge) along the longitudinal axis of the barrel coil(s). We can express the rate-of-change of the mutual inductance directly in terms of its location as follows:

$$\begin{aligned} \frac{dM}{dt} &= \frac{\partial M}{\partial z} \frac{dz}{dt} \\ &= \frac{\partial M}{\partial z} \times Speed \end{aligned} \quad (10)$$

Since M depends on only one independent variable (location variable z), only one partial derivative is needed. Furthermore, the rate-of-change of the location variable is simply the derivative which represents the speed of the armature.

The partial derivative $\partial M/\partial z$ is a parameter which depends solely on the geometry of the problem. Assuming there is no ferro-magnetic material in the vicinity, the mutual inductance will not depend on the magnitude of the currents, the armature's speed, or anything else that we have included in this physical model (we have not included temperature). In fact, the mutual inductance and its spatial derivative are values that could be pre-computed before the numerical integration procedure even begins. If we used FEMM to calculate $\partial M/\partial z$ with the armature located at a large number of possible locations along its route, we could interpolate within the table to find the value which obtains at any particular point during the simulation. Or, if we are prepared to wait, we can let FEMM determine the derivative at the start of every time step. Whichever way we choose to find $\partial M/\partial z$, it can be found. Since we will also have calculated the armature's speed at the end of the previous time step, we can use as our basic equation the following form of Equation (9).

$$\begin{bmatrix} \frac{d^2q_1}{dt^2} \\ \frac{di_A}{dt} \end{bmatrix} = \frac{1}{L_1L_A - M^2} \begin{bmatrix} L_A & M \\ M & L_1 \end{bmatrix} \cdot \left\{ \begin{bmatrix} -R_1 \frac{dq_1}{dt} - \frac{q_1}{C_1} \\ -R_A i_A \end{bmatrix} + Speed \frac{\partial M}{\partial z} \begin{bmatrix} i_A \\ \frac{dq_1}{dt} \end{bmatrix} \right\} \quad (11)$$

With the right-hand side fully quantified, we can multiply through to calculate the two derivatives d^2q_1/dt^2 and di_A/dt on the left-hand side. We will make the duration of the time steps ΔT short enough that these derivatives can be assumed to remain constant throughout the time step. If so, then three simple linear integrations can be used to integrate through to the end of the time step, namely:

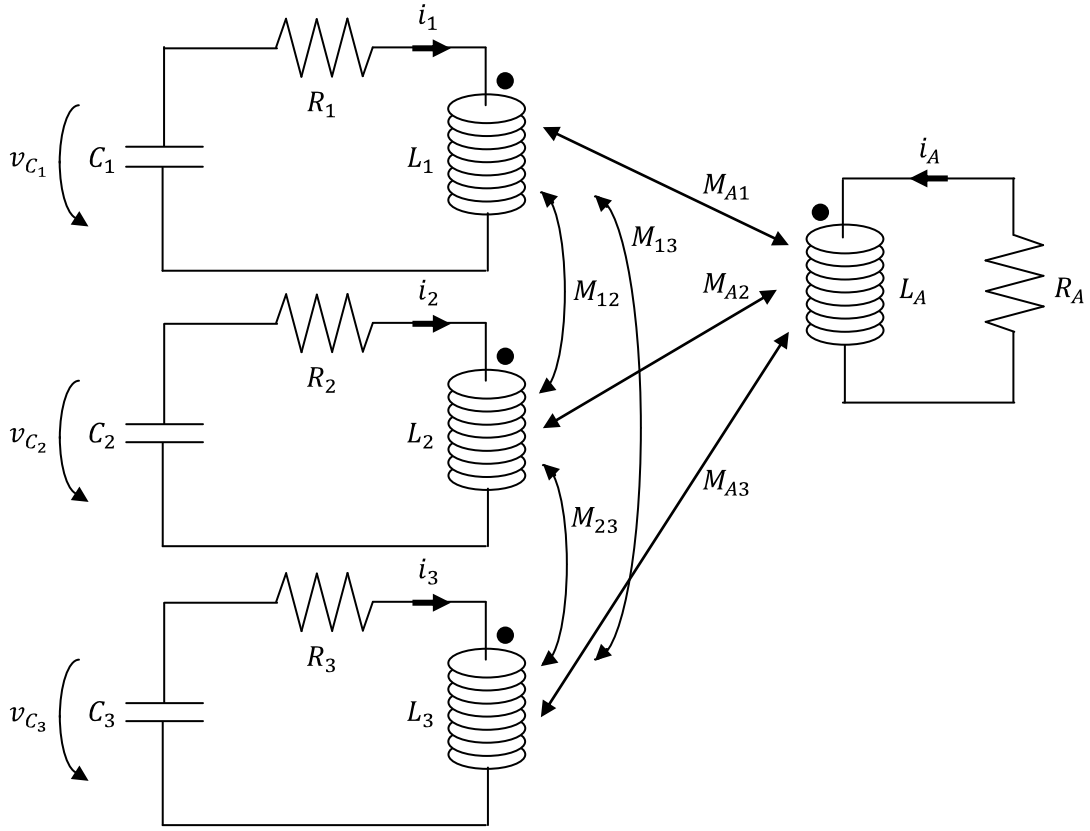
$$\left.\frac{dq_1}{dt}\right|_{end} = \left.\frac{dq_1}{dt}\right|_{start} + \Delta T \left.\frac{d^2q_1}{dt^2}\right|_{const} \quad (12A)$$

$$q_1|_{end} = q_1|_{start} + \Delta T \left.\frac{dq_1}{dt}\right|_{start} + \tfrac{1}{2}\Delta T^2 \left.\frac{d^2q_1}{dt^2}\right|_{const} \quad (12B)$$

$$i_A|_{end} = i_A|_{start} + \Delta T \left.\frac{di_A}{dt}\right|_{const} \quad (12C)$$

What if there are multiple barrel coils?

To see what happens, let's assume there are three barrel coils, all conducting current at the same time. (If the circuit of any particular barrel coil is not closed, that coil plays no role in what is going on.) Each barrel coil will have its own circuit in the schematic, which circuit will look exactly the same as the circuit for barrel coil #1. I will identify the components of circuit #j by using subscript j. The armature's circuit will remain unchanged. To avoid possible pitfalls, I want to take care with the conventions, so I will draw the complete schematic.



There are several issues which need to be clarified before we write down the circuit equations.

One is the number of mutual inductances (should it be "inducti"?). There are four coils, and there will be an interaction between each pair of coils, including interactions between the barrel coils themselves. I will define the mutual inductance between the armature and barrel coil #j as M_{Aj} and the mutual inductance between barrel coils #j and #k as M_{jk}. The interaction between any particular pair of coils affects each coil in the same way, so $M_{kj} = M_{jk}$.

Another issue relates to the timing of the individual circuits. They may all be conducting at this moment, but that does not mean they all began conducting at the same time. In fact, we will "fire" the barrel coils individually only when the armature nears them. The moment at which the switch controlling a particular circuit is closed has a significant impact on the acceleration that coil will impart to the armature. What the different start times mean to the circuit equations is that the integrals which keep track of the charge which has flown out of a particular capacitor do not all have the same starting times.

Another issue is the way we add up the total amount of magnetic flux linking each coil. This is where the conventions regarding phase dots and current directions come into play. Look back at the schematic for the case with only one barrel coil and, in particular, the direction which was assigned to positive currents. The current i_1 was assumed to be algebraically positive when it flows into the dotted end of barrel coil #1. The armature's current i_A was assumed to be algebraically positive when it flows into the dotted end of its coil. The directions assumed for algebraically positive currents in the two new circuits are the same, being positive when current flows into the dotted ends.

Because of this consistency, we can add up the amounts of flux linking each coil using a direct extension of what we did in the two-coil case. No minus signs are needed. We can say that the total amount of flux linking each of the four coils is:

$$\left.\begin{aligned} \varphi_A &= L_A i_A + M_{A1} i_1 + M_{A2} i_2 + M_{A3} i_3 \\ \varphi_1 &= L_1 i_1 + M_{1A} i_A + M_{12} i_2 + M_{13} i_3 \\ \varphi_2 &= L_2 i_2 + M_{2A} i_A + M_{21} i_1 + M_{23} i_3 \\ \varphi_3 &= L_3 i_3 + M_{3A} i_A + M_{31} i_1 + M_{32} i_2 \end{aligned}\right\} \quad (13)$$

Each coil gets a flux contribution through its own self-inductance and a further contribution from each other coil through their mutual inductance. Things will be more readable if we write this in matrix format. Taking account of the symmetries $M_{kj} = M_{jk}$, we get:

$$\begin{bmatrix} \varphi_A \\ \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{bmatrix} = \begin{bmatrix} L_A & M_{A1} & M_{A2} & M_{A3} \\ M_{A1} & L_1 & M_{12} & M_{13} \\ M_{A2} & M_{12} & L_2 & M_{23} \\ M_{A3} & M_{13} & M_{23} & L_3 \end{bmatrix} \cdot \begin{bmatrix} i_A \\ i_1 \\ i_2 \\ i_3 \end{bmatrix} \quad (14)$$

This is a convenient place to describe the time-derivative of the fluxes. Applying the product rule matrix-wide gives:

$$\frac{d}{dt}\begin{bmatrix} \varphi_A \\ \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{bmatrix} = \frac{d}{dt}\begin{bmatrix} L_A & M_{A1} & M_{A2} & M_{A3} \\ M_{A1} & L_1 & M_{12} & M_{13} \\ M_{A2} & M_{12} & L_2 & M_{23} \\ M_{A3} & M_{13} & M_{23} & L_3 \end{bmatrix} \cdot \begin{bmatrix} i_A \\ i_1 \\ i_2 \\ i_3 \end{bmatrix} + \begin{bmatrix} L_A & M_{A1} & M_{A2} & M_{A3} \\ M_{A1} & L_1 & M_{12} & M_{13} \\ M_{A2} & M_{12} & L_2 & M_{23} \\ M_{A3} & M_{13} & M_{23} & L_3 \end{bmatrix} \cdot \frac{d}{dt}\begin{bmatrix} i_A \\ i_1 \\ i_2 \\ i_3 \end{bmatrix} \quad (15)$$

Taking the derivative operation inside the matrix and vector, and noting that: (i) the self-inductances are constant with respect to time and (ii) the barrel coil-to-barrel coil mutual inductances are also constant with respect to one another), we get:

$$\frac{d}{dt}\begin{bmatrix}\varphi_A\\ \varphi_1\\ \varphi_2\\ \varphi_3\end{bmatrix}=\begin{bmatrix}0 & \frac{dM_{A1}}{dt} & \frac{dM_{A2}}{dt} & \frac{dM_{A3}}{dt}\\ \frac{dM_{A1}}{dt} & 0 & 0 & 0\\ \frac{dM_{A2}}{dt} & 0 & 0 & 0\\ \frac{dM_{A3}}{dt} & 0 & 0 & 0\end{bmatrix}\cdot\begin{bmatrix}i_A\\ i_1\\ i_2\\ i_3\end{bmatrix}+\begin{bmatrix}L_A & M_{A1} & M_{A2} & M_{A3}\\ M_{A1} & L_1 & M_{12} & M_{13}\\ M_{A2} & M_{12} & L_2 & M_{23}\\ M_{A3} & M_{13} & M_{23} & L_3\end{bmatrix}\cdot\begin{bmatrix}\frac{di_A}{dt}\\ \frac{di_1}{dt}\\ \frac{di_2}{dt}\\ \frac{di_3}{dt}\end{bmatrix} \quad (16)$$

The presence of multiple barrel coils does not change the fact that the mutual inductances depend on only one independent variable, being the *z*-location of the armature. The time-derivative of each mutual inductance can be expressed as the product of the spatial partial derivative and the speed of the armature. If S is the instanraneous speed of the armature, this can be written as:

$$\frac{d}{dt}\begin{bmatrix}\varphi_A\\ \varphi_1\\ \varphi_2\\ \varphi_3\end{bmatrix}=\begin{bmatrix}0 & S\frac{\partial M_{A1}}{\partial z} & S\frac{\partial M_{A2}}{\partial z} & S\frac{\partial M_{A3}}{\partial z}\\ S\frac{\partial M_{A1}}{\partial z} & 0 & 0 & 0\\ S\frac{\partial M_{A2}}{\partial z} & 0 & 0 & 0\\ S\frac{\partial M_{A3}}{\partial z} & 0 & 0 & 0\end{bmatrix}\cdot\begin{bmatrix}i_A\\ i_1\\ i_2\\ i_3\end{bmatrix}+\begin{bmatrix}L_A & M_{A1} & M_{A2} & M_{A3}\\ M_{A1} & L_1 & M_{12} & M_{13}\\ M_{A2} & M_{12} & L_2 & M_{23}\\ M_{A3} & M_{13} & M_{23} & L_3\end{bmatrix}\cdot\begin{bmatrix}\frac{di_A}{dt}\\ \frac{di_1}{dt}\\ \frac{di_2}{dt}\\ \frac{di_3}{dt}\end{bmatrix} \quad (17)$$

Next, just like we did in the two-coil case, we are going to relate the current flowing in each barrel coil's circuit to the absolute electrical charge, in Coulombs, stored in the associated capacitor.

$$\left.\begin{aligned}&\text{barrel coil \#1:}\quad i_1=-\frac{dq_1}{dt}\\ &\text{barrel coil \#2:}\quad i_2=-\frac{dq_2}{dt}\\ &\text{barrel coil \#3:}\quad i_3=-\frac{dq_3}{dt}\end{aligned}\right\} \quad (18)$$

The rate-of-change of the fluxes can be expressed in terms of these charges, with the result that:

$$\frac{d}{dt}\begin{bmatrix}\varphi_A\\ \varphi_1\\ \varphi_2\\ \varphi_3\end{bmatrix}=\begin{bmatrix}0 & S\frac{\partial M_{A1}}{\partial z} & S\frac{\partial M_{A2}}{\partial z} & S\frac{\partial M_{A3}}{\partial z}\\ S\frac{\partial M_{A1}}{\partial z} & 0 & 0 & 0\\ S\frac{\partial M_{A2}}{\partial z} & 0 & 0 & 0\\ S\frac{\partial M_{A3}}{\partial z} & 0 & 0 & 0\end{bmatrix}\cdot\begin{bmatrix}i_A\\ -\frac{dq_1}{dt}\\ -\frac{dq_2}{dt}\\ -\frac{dq_3}{dt}\end{bmatrix}+\begin{bmatrix}L_A & M_{A1} & M_{A2} & M_{A3}\\ M_{A1} & L_1 & M_{12} & M_{13}\\ M_{A2} & M_{12} & L_2 & M_{23}\\ M_{A3} & M_{13} & M_{23} & L_3\end{bmatrix}\cdot\begin{bmatrix}\frac{di_A}{dt}\\ -\frac{d^2q_1}{dt^2}\\ -\frac{d^2q_2}{dt^2}\\ -\frac{d^2q_3}{dt^2}\end{bmatrix} \quad (19)$$

Having thus taken care of a couple of preliminaries, let's now turn to the circuit equations. There are four separate circuits and there will be a separate circuit equation for each one. Each of the circuits is a series circuit with either two or three components. The circuit equation for each circuit is a statement that the voltage drops over the components in the circuit, taken in sequence around the loop, must add up to zero. The circuit equation for the armature is just like Equation $(1B)$ above, the only difference being a more complicated expression for the total flux linking the armature coil. The armature's circuit equation is:

$$\frac{d}{dt}\varphi_A + R_A i_A = 0 \quad (20)$$

The circuit equations for the three barrel coils are just like the one for barrel coil #1 alone, in Equation $(1A)$, the only difference being that they too have more complicated expressions for their flux linkages. Their circuit equations are:

$$\left.\begin{aligned}
&\text{barrel coil \#1:} \quad V_o|_1 - \frac{1}{C_1}\int_{t_1\,start} i_1 dt = R_1 i_1 + \frac{d}{dt}\varphi_1 \\
&\text{barrel coil \#2:} \quad V_o|_2 - \frac{1}{C_2}\int_{t_2\,start} i_2 dt = R_2 i_2 + \frac{d}{dt}\varphi_2 \\
&\text{barrel coil \#3:} \quad V_o|_3 - \frac{1}{C_3}\int_{t_3\,start} i_3 dt = R_3 i_3 + \frac{d}{dt}\varphi_3
\end{aligned}\right\} \quad (21)$$

Since there is now more than one capacitor, and since each capacitor might have a different starting voltage V_o, I have added subscripts to relate the starting voltages to their respective circuits. Next, by replacing the occurrences of currents i_1, i_2 and i_3 with their charge-equivalent derivatives, the circuit equations can be written as:

$$\left.\begin{aligned}
&\text{barrel coil \#1:} \quad \frac{q_1}{C_1} = -R_1\frac{dq_1}{dt} + \frac{d}{dt}\varphi_1 \quad \text{for } t > t_{1start} \\
&\text{barrel coil \#2:} \quad \frac{q_2}{C_2} = -R_2\frac{dq_2}{dt} + \frac{d}{dt}\varphi_2 \quad \text{for } t > t_{2start} \\
&\text{barrel coil \#3:} \quad \frac{q_3}{C_3} = -R_3\frac{dq_3}{dt} + \frac{d}{dt}\varphi_3 \quad \text{for } t > t_{3start}
\end{aligned}\right\} \quad (22)$$

I have tried to be precise in describing the time frames. Each of these three equations only applies after its switch has been closed. For all times before the switch closures, the stored charges are directly related to the capacitors' initial voltages, thus:

$$\left.\begin{aligned}
&\text{barrel coil \#1:} \quad \frac{q_1}{C_1} = V_o|_1 \quad \text{for } t \le t_{1start} \\
&\text{barrel coil \#2:} \quad \frac{q_2}{C_2} = V_o|_2 \quad \text{for } t \le t_{2start} \\
&\text{barrel coil \#3:} \quad \frac{q_3}{C_3} = V_o|_3 \quad \text{for } t \le t_{3start}
\end{aligned}\right\} \quad (23)$$

The circuit equations can be re-arranged into the following forms:

$$\left.\begin{aligned} \frac{d}{dt}\varphi_A + R_A i_A &= 0 \\ \frac{d}{dt}\varphi_1 - R_1\frac{dq_1}{dt} - \frac{q_1}{C_1} &= 0 \\ \frac{d}{dt}\varphi_2 - R_2\frac{dq_2}{dt} - \frac{q_2}{C_2} &= 0 \\ \frac{d}{dt}\varphi_3 - R_3\frac{dq_3}{dt} - \frac{q_3}{C_3} &= 0 \end{aligned}\right\} \quad (24)$$

which can be stated in matrix form as follows:

$$\frac{d}{dt}\begin{bmatrix}\varphi_A\\ \varphi_1\\ \varphi_2\\ \varphi_3\end{bmatrix} + \begin{bmatrix}R_A & 0 & 0 & 0\\ 0 & -R_1 & 0 & 0\\ 0 & 0 & -R_2 & 0\\ 0 & 0 & 0 & -R_3\end{bmatrix}\cdot\begin{bmatrix}i_A\\ \frac{dq_1}{dt}\\ \frac{dq_2}{dt}\\ \frac{dq_3}{dt}\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & -\frac{1}{C_1} & 0 & 0\\ 0 & 0 & -\frac{1}{C_2} & 0\\ 0 & 0 & 0 & -\frac{1}{C_3}\end{bmatrix}\cdot\begin{bmatrix}0\\ q_1\\ q_2\\ q_3\end{bmatrix} = \begin{bmatrix}0\\ 0\\ 0\\ 0\end{bmatrix} \quad (25)$$

Equations (19) and (25) can be combined to give the set of differential equations which describes the whole system.

$$\begin{bmatrix}0 & S\frac{\partial M_{A1}}{\partial z} & S\frac{\partial M_{A2}}{\partial z} & S\frac{\partial M_{A3}}{\partial z}\\ S\frac{\partial M_{A1}}{\partial z} & 0 & 0 & 0\\ S\frac{\partial M_{A2}}{\partial z} & 0 & 0 & 0\\ S\frac{\partial M_{A3}}{\partial z} & 0 & 0 & 0\end{bmatrix}\cdot\begin{bmatrix}i_A\\ -\frac{dq_1}{dt}\\ -\frac{dq_2}{dt}\\ -\frac{dq_3}{dt}\end{bmatrix} + \begin{bmatrix}L_A & M_{A1} & M_{A2} & M_{A3}\\ M_{A1} & L_1 & M_{12} & M_{13}\\ M_{A2} & M_{12} & L_2 & M_{23}\\ M_{A3} & M_{13} & M_{23} & L_3\end{bmatrix}\cdot\begin{bmatrix}\frac{di_A}{dt}\\ -\frac{d^2q_1}{dt^2}\\ -\frac{d^2q_2}{dt^2}\\ -\frac{d^2q_3}{dt^2}\end{bmatrix} + \cdots$$

$$\cdots + \begin{bmatrix}R_A & 0 & 0 & 0\\ 0 & -R_1 & 0 & 0\\ 0 & 0 & -R_2 & 0\\ 0 & 0 & 0 & -R_3\end{bmatrix}\cdot\begin{bmatrix}i_A\\ \frac{dq_1}{dt}\\ \frac{dq_2}{dt}\\ \frac{dq_3}{dt}\end{bmatrix} + \begin{bmatrix}0 & 0 & 0 & 0\\ 0 & -\frac{1}{C_1} & 0 & 0\\ 0 & 0 & -\frac{1}{C_2} & 0\\ 0 & 0 & 0 & -\frac{1}{C_3}\end{bmatrix}\cdot\begin{bmatrix}0\\ q_1\\ q_2\\ q_3\end{bmatrix} = \begin{bmatrix}0\\ 0\\ 0\\ 0\end{bmatrix} \quad (26)$$

I am going to collect terms which have the same dependence on the independent variables. But, I am going to do one other thing as well. I am going to replace the charge-derivatives with their equivalent currents. That will put matters back into terms of current, and make the result a little easier to use. We get:

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

$$\cdots = -\begin{bmatrix} R_A & S\frac{\partial M_{A1}}{\partial z} & S\frac{\partial M_{A2}}{\partial z} & S\frac{\partial M_{A3}}{\partial z} \\ S\frac{\partial M_{A1}}{\partial z} & R_1 & 0 & 0 \\ S\frac{\partial M_{A2}}{\partial z} & 0 & R_2 & 0 \\ S\frac{\partial M_{A3}}{\partial z} & 0 & 0 & R_3 \end{bmatrix} \cdot \begin{bmatrix} i_A \\ i_1 \\ i_2 \\ i_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{C_1} & 0 & 0 \\ 0 & 0 & \frac{1}{C_2} & 0 \\ 0 & 0 & 0 & \frac{1}{C_3} \end{bmatrix} \cdot \begin{bmatrix} 0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (27)$$

The way ahead is to recognize that, at the start of any time step, all of the variables on the right-hand side of Equation (27) will be known. So will all the inductances which appear on the left-hand side. Gathering up all the constants on the right-hand side into vector B, Equation (27) has the form:

$$\begin{bmatrix} L_A & M_{A1} & M_{A2} & M_{A3} \\ M_{A1} & L_1 & M_{12} & M_{13} \\ M_{A2} & M_{12} & L_2 & M_{23} \\ M_{A3} & M_{13} & M_{23} & L_3 \end{bmatrix} \cdot \begin{bmatrix} \frac{di_A}{dt} \\ \frac{di_1}{dt} \\ \frac{di_2}{dt} \\ \frac{di_3}{dt} \end{bmatrix} = \begin{bmatrix} b_A \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (28)$$

where everything is numeric (at the start of the time step) except for the four unknown currents. In principle, we can invert the inductance matrix and multiply through to write the solution as:

$$\begin{bmatrix} \frac{di_A}{dt} \\ \frac{di_1}{dt} \\ \frac{di_2}{dt} \\ \frac{di_3}{dt} \end{bmatrix} = \begin{bmatrix} L_A & M_{A1} & M_{A2} & M_{A3} \\ M_{A1} & L_1 & M_{12} & M_{13} \\ M_{A2} & M_{12} & L_2 & M_{23} \\ M_{A3} & M_{13} & M_{23} & L_3 \end{bmatrix}^{-1} \cdot \begin{bmatrix} b_A \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (29)$$

This will give us the derivatives we need to integrate through the time step in the same manner as above. As an example, for barrel coil #1 (note the negative signs in the expression for charge q_1), we would use:

$$i_1|_{end} = i_1|_{start} + \Delta T \left.\frac{di_1}{dt}\right|_{const} \quad (30A)$$

$$q_1|_{end} = q_1|_{start} - \Delta T i_1|_{start} - \tfrac{1}{2}\Delta T^2 \left.\frac{di_1}{dt}\right|_{const} \quad (30B)$$

The heat burned off by the resistors

In order to validate the procedure, I am going to keep track of the various stores of energy as the armature travels along the barrel. One of the more troublesome types of energy to look after is the heat energy burned off by the resistors in the circuits.

The instantaneous power consumed by resistance R_1 in the first barrel coil's circuit is $R_1 i_1^2$. The power consumed by this resistor during the course of a whole time step is obtained by integrating the instantaneous power by a small differential in time dt and and then adding up (by integrating) the small energy differentials. Equation $(30A)$ is the calculation of current i_1 at the end of a time step, in terms of the conditions at the start of the time step. Because the current is assumed to change linearly during the time step (since di_1/dt is assumed to be constant during the time step), the instantaneous current at any arbitrary time t during the time step can be written in an analogous way:

$$i_1(\tau) = i_1|_{start} + \tau \left.\frac{di_1}{dt}\right|_{const} \qquad (31)$$

where time τ in this instance is the time elapsed from the start of the time step. Then, the instantaneous power being consumed by resistance R_1 at time τ can be expressed as:

$$\begin{aligned} P_1(\tau) &= R_1 i_1^2 \\ &= R_1 \left[(i_1|_{start})^2 + \left(2 i_1|_{start} \left.\frac{di_1}{dt}\right|_{const} \right) \tau + \left(\left.\frac{di_1}{dt}\right|_{const} \right)^2 \tau^2 \right] \end{aligned} \qquad (32)$$

When $P_1(\tau)d\tau$ is integrated over one complete time step, the result is the energy consumed by resistance R_1 during the whole time step. It is:

$$\begin{aligned} E_1 &= \int_{\tau=start}^{end} P_1(\tau) d\tau \\ &= R_1 \left[(i_1|_{start})^2 \tau + \left(2 i_1|_{start} \left.\frac{di_1}{dt}\right|_{const} \right) \tfrac{1}{2} \tau^2 + \left(\left.\frac{di_1}{dt}\right|_{const} \right)^2 \tfrac{1}{3} \tau^3 \right]_{t=start}^{end} \\ &= R_1 \left[(i_1|_{start})^2 \Delta T + \left(i_1|_{start} \left.\frac{di_1}{dt}\right|_{const} \right) \Delta T^2 + \left(\left.\frac{di_1}{dt}\right|_{const} \right)^2 \tfrac{1}{3} \Delta T^3 \right] \end{aligned} \qquad (33)$$

In a similar way, the power consumed by any other resistance R_* during the time step is:

$$E_* = R_* \Delta T \left[(i_*|_{start})^2 + \left(i_*|_{start} \left.\frac{di_*}{dt}\right|_{const} \right) \Delta T + \frac{1}{3} \left(\left.\frac{di_*}{dt}\right|_{const} \right)^2 \Delta T^2 \right] \qquad (34)$$

The energy stored in the magnetic fields

Another of the troublesome energies to keep track of is the energy stored in the magnetic fields. For a single fixed coil with self-inductance L, the energy stored in its magnetic field when the instantaneous current is I is well-known;. It is $\frac{1}{2}LI^2$, with no ifs ands or buts. In our case, though, we have mutual inductance, which changes with time. What is the instantaneous energy stored in the magnetic field of this more complicated configuration? To find the answer, I will apply the principle of superposition.

This means that I will notionally create the instantaneous configuration through a series of steps, in each of which the additional energy stored can be calculated quite easily.

Let's examine the case when there are three barrel coils, in which the instantaneous currents are I_1, I_2 and I_3. (There is no good reason for my using capital letters for instantaneous currents, but I will do so anyway.) I am going to assume they are algebraically positive, meaning that the currents are flowing into the dotted ends of their respective coils in the schematic. The instantaneous current in the armature circuit at this moment is I_A. I will also assume that it is algebraically positive as well, meaning that I_A is flowing into the dotted end of the armature coil.

Let's create this configuration in a series of steps. At the start, assume that all currents are zero, so there is no stored magnetic energy at all. In the first step, we will increase the current in the circuit of barrel coil #1 from $i_1 = 0$ to $i_1 = I_1$. Since there is no current flowing in any of the other circuits, they can be treated as if they simply do not exist. That means that the voltage drop over the coil in barrel circuit #1 is determined by its self-inductance alone. There is not any mutual inductance during this first step. We can write the instantaneous voltage drop over barrel coil #1 as:

$$v_1 = \frac{d(L_1 i_1)}{dt} \quad (35)$$

since the only flux linkage is the quantity $L_1 i_1$. The instantaneous power being consumed by this voltage drop is, as usual, the product of the voltage and the current, namely:

$$p_1 = v_1 i_1 = i_1 \frac{d(L_1 i_1)}{dt} \quad (36)$$

I am going to take advantage of the fact that the self-inductance of barrel coil #1 is constant with respect to time, so it can be brought outside the derivative. At the same time, I am going to restate the differential $i_1 di_1 = ½ di_1^2$, which is a mathematical identity. Then, Equation (36) for the instantaneous power usage can be written as:

$$p_1 = ½ L_1 \frac{d(i_1^2)}{dt} \quad (37)$$

The energy consumed by barrel coil #1 during a very short period of time dt can be written as $p_1 dt$. This assumes that the interval of time dt is so short that the power level p_1 can be assumed to be constant during the interval. Remember that, for constant power, the work done, or energy invested, is simply the product of the power level multiplied by the period of time during which it acts. If we let dE_1 be the amount of work, or energy, invested in barrel coil #1 during time interval dt, then:

$$dE_1 = p_1 dt = ½ L_1 \frac{d(i_1^2)}{dt} dt \quad (38)$$

In the case of an inductor like barrel coil #1, the work which is done is invested in building up a magnetic field. The cumulative work which is done, equal to the energy invested in the magnetic field, from some time $t = 0$ when current $i_1 = 0$ to some later time $t = \tau$ when the current is $i_1 = I_1$ is the sum (or integral) of the little increments dE_1 which were added to the field's energy during that time. We can write the cumulative energy stored during this first step as:

$$
\begin{aligned}
E_{step\,\#1} &= \int_{t=0}^{t=\tau} dE_1 \\
&= \int_{i_1=0}^{i_1=I_1} ½L_1 d(i_1^2) \\
&= ½L_1 I_1^2 \qquad (39)
\end{aligned}
$$

Now that barrel coil #1 is powered up, let's bring barrel coil #2 on line. In this second step, the only thing that will change is the current flowing through barrel coil #2. There still will not be any current flowing through the armature or barrel coil #3. Significantly, though, there will be a constant current of I_1 flowing through barrel coil #1. It was there at the end of the first step, and we will keep it there.

As the current in barrel coil #2 ramps up, the instantaneous voltage drop which develops will be:

$$v_2 = \frac{d(L_2 i_2)}{dt} \qquad (40)$$

Even though there is a mutual coupling between barrel coil #2 and barrel coil #1, the current flowing through barrel coil #1 is constant and will not induce any voltage drop over barrel coil #2. The reverse is not true. Through their mutual inductance, the changing current in barrel coil #2 will generate an instantaneous voltage drop over barrel coil #1, in the amount of:

$$v_1 = \frac{d(M_{12} i_2)}{dt} \qquad (41)$$

So, in this second step, there are two voltage drops through which current flows. That means there are two uses of power. The two voltage-times-current products are the following:

$$\text{In barrel coil \#2:} \quad p_2 = i_2 \frac{d(L_2 i_2)}{dt} \qquad (42A)$$

$$\text{In barrel coil \#1:} \quad p_1 = I_1 \frac{d(M_{12} i_2)}{dt} \qquad (42B)$$

Just to be clear about the power being used in barrel coil #1. Its voltage drop may well be generated by the goings-on in barrel coil #2, but the flow of current is nevertheless the constant current I_1. The self-inductance of barrel coil #2 L_2 is a constant. Because these are two stationary barrel coils, their mutual inductance M_{12} is also constant. We can add up Equations $(42A)$ and $(42B)$ and say that the total power being consumed during this second step is:

$$p = L_2 i_2 \frac{di_2}{dt} + M_{12} I_1 \frac{di_2}{dt} = ½L_2 \frac{d(i_2^2)}{dt} + M_{12} I_1 \frac{di_2}{dt} \qquad (43)$$

We will consider the small bits of work done, or energy stored, in little dt intervals of time and then add them up during the period while current i_2 climbs from zero to I_2. The energy stored during this second step is:

$$
\begin{aligned}
E_{step\ \#2} &= \int_{t=0}^{t=\tau} p\,dt \\
&= \int_{i_2=0}^{i_2=I_2} \tfrac{1}{2}L_2\,d(i_2^2) + M_{12}I_1\,di_2 \\
&= \tfrac{1}{2}L_2I_2^2 + M_{12}I_1I_2 \qquad (44)
\end{aligned}
$$

Note that the integral over time can be expressed as an integral over current values (as they rise), and that the value of the integral does not depend on <u>how</u> current i_2 makes its way from zero to I_2. The change could be fast, it could be slow, it could be a saw tooth, it may exceed I_2 somewhere along the way. When the current gets to I_2, the energy stored during this step #2 is given by Equation (44). When we add in the energy from Equation (39), we have the energy stored in this pair of barrel coils at the end of step #2.

The third step is to bring barrel coil #3 up to speed. Constant currents I_1 and I_2 will flow through the first two barrel coils, respectively, while current i_3 in the new coil is raised from zero to I_3. The rising current i_3 will generate a self-induced voltage across barrel coil #3 and, through mutual interaction, across the other two barrel coils as well. The three instantaneous voltage drops are the following.

$$
\left.
\begin{aligned}
&\text{Across barrel coil \#3:} \quad v_3 = \frac{d(L_3i_3)}{dt} \\
&\text{Across barrel coil \#2:} \quad v_2 = \frac{d(M_{32}i_3)}{dt} \\
&\text{Across barrel coil \#1:} \quad v_1 = \frac{d(M_{31}i_3)}{dt}
\end{aligned}
\right\} \qquad (45)
$$

Different currents flow through the three coils, so the contributions each makes to the total power consumption during this step are:

$$
\left.
\begin{aligned}
&\text{By barrel coil \#3:} \quad p_3 = i_3\frac{d(L_3i_3)}{dt} = \tfrac{1}{2}L_3\frac{d(i_3^2)}{dt} \\
&\text{By barrel coil \#2:} \quad p_2 = I_2\frac{d(M_{32}i_3)}{dt} = M_{32}I_2\frac{di_3}{dt} \\
&\text{By barrel coil \#1:} \quad p_1 = I_1\frac{d(M_{31}i_3)}{dt} = M_{31}I_1\frac{di_3}{dt}
\end{aligned}
\right\} \qquad (46)
$$

As before, we can rely on the fact that the mutual inductances between barrel coils are constant, and can therefore be extracted from within the derivatives. We add the energy contributions $p\,dt$ to calculate the energy consumed (from the power supplies as voltage-times-current) and stored in the magnetic field.

$$
\begin{aligned}
E_{step\ \#3} &= \int_{t=0}^{t=\tau} p\,dt \\
&= \int_{i_3=0}^{i_3=I_3} \tfrac{1}{2}L_3\,d(i_3^2) + M_{32}I_2\,di_3 + M_{31}I_1\,di_3 \\
&= \tfrac{1}{2}L_3I_3^2 + M_{32}I_2I_3 + M_{31}I_1I_3 \qquad (47)
\end{aligned}
$$

In the fourth step , we are going bring the current flowing through the armature up to I_A. Once we have completed this fourth step, the configuration will be exactly the one we have been aiming for, with different currents flowing in all four coils. During this last step, we will assume that constant currents I_1, I_2 and I_3 keep flowing through the barrel coils, while current i_A in the armature climbs from zero to I_A. This rising current i_A will generate a self-induced voltage across the armature, and mutually-induced voltages across the three barrel coils. The four instantaneous voltage drops are:

$$\left.\begin{array}{ll} \text{Across the armature:} & v_A = \dfrac{d(L_A i_A)}{dt} \\ \text{Across barrel coil \#3:} & v_3 = \dfrac{d(M_{A3} i_A)}{dt} \\ \text{Across barrel coil \#2:} & v_2 = \dfrac{d(M_{A2} i_A)}{dt} \\ \text{Across barrel coil \#1:} & v_1 = \dfrac{d(M_{A1} i_A)}{dt} \end{array}\right\} \quad (48)$$

Different currents flow through all four coils. As always, we can calculate the instantaneous power consumptions as the voltage-times-current products:

$$\left.\begin{array}{ll} \text{By the armature:} & p_A = i_A \dfrac{d(L_A i_A)}{dt} \\ \text{By barrel coil \#3:} & p_3 = I_3 \dfrac{d(M_{A3} i_A)}{dt} \\ \text{By barrel coil \#2:} & p_2 = I_2 \dfrac{d(M_{A2} i_A)}{dt} \\ \text{By barrel coil \#1:} & p_1 = I_1 \dfrac{d(M_{A1} i_A)}{dt} \end{array}\right\} \quad (49)$$

As before, we can add up the pdt contributions to the stored magnetic energy. The integral is:

$$\begin{aligned} E_{step\ \#4} &= \int_{t=0}^{t=\tau} p\,dt \\ &= \int_{i_A=0}^{i_A=I_A} \left\{ i_A \frac{d(L_A i_A)}{dt} + I_3 \frac{d(M_{A3} i_A)}{dt} + I_2 \frac{d(M_{A2} i_A)}{dt} + I_1 \frac{d(M_{A1} i_A)}{dt} \right\} dt \end{aligned} \quad (50)$$

But, things are a little different this time around. The self-inductance of the armature L_A is a constant, but the three mutual inductances are not. They have their own dependence on time and cannot simply be removed from the derivatives as constants. The first term in the curly brackets, for the self-inductance, can be integrated to give the sum $\frac{1}{2}L_A I_A^2$, which has exactly the same form as the self-induced energy of the barrel coils. But, the mutual inductance terms have to be handled differently. As an example, let's expand the derivative of the middle mutual inductance term using the product rule.

$$I_2 \frac{d(M_{A2} i_A)}{dt} = I_2 \left[M_{A2} \frac{di_A}{dt} + i_A \frac{dM_{A2}}{dt} \right] \quad (51)$$

As we did before, we can express the time-derivative of the mutual inductance as the product of its spatial derivative and the speed of the armature, to get:

$$I_2 \frac{d(M_{A2} i_A)}{dt} = I_2 \left[M_{A2} \frac{di_A}{dt} + i_A Speed \frac{\partial M_{A2}}{\partial z} \right] \quad (52)$$

We can integrate the first half in closed form, but not the second half. We get:

$$\begin{aligned} \int_{i_A=0}^{i_A=I_A} \left\{ I_2 \frac{d(M_{A2} i_A)}{dt} \right\} dt &= \int_{i_A=0}^{i_A=I_A} I_2 M_{A2} di_A + \int_{t=0}^{t=\tau} I_2 i_A Speed \frac{\partial M_{A2}}{\partial z} dt \\ &= M_{A2} I_A I_2 + \int_{z_A=start}^{z_A=end} I_2 i_A Speed \frac{\partial M_{A2}}{\partial z} dz \end{aligned} \quad (53)$$

I just want to point one thing out. It's important to understanding why we were able to carry out the first integral. The mutual inductance M_{A2} is not constant; it changes. But M_{A2} is constant with respect to current. Change in its value are not caused by changes in current but, rather, by changes in location. Since the mutual inductance does not depend on current, it can be considered to be constant for the purpose of integrating over current, as is done in the first integral. For the first integral (only), the mutual inductance can be taken outside as a constant coefficient.

Not so in the second integral. Since the fundamental dependence of the mutual inductance is on the axial location variable z, I have taken the liberty of re-stating the integral from one over variable time t to one over variable distance z.

When one applies this treatment of the middle mutual inductance term to its counterparts, the expression in Equation (50) for the energy consumed during the fourth step can be re-written as follows:

$$E_{step\ \#4} = \left[\begin{array}{c} \tfrac{1}{2} L_A I_A^2 + M_{A3} I_A I_3 + M_{A2} I_A I_2 + M_{A1} I_A I_1 + \cdots \\ \cdots + \int_{z_A=start}^{z_A=end} i_A Speed \left\{ I_3 \frac{\partial M_{A3}}{\partial z} + I_2 \frac{\partial M_{A2}}{\partial z} + I_1 \frac{\partial M_{A1}}{\partial z} \right\} dz \end{array} \right] \quad (54)$$

Before I do something about the last term, I want to regroup. I want to add up the energies which were consumed in all four steps. That will give us the total energy consumed as the magnetic fields which link the four coils were constructed. The total is:

$$\begin{aligned} E_{total} &= \left[\begin{array}{rl} \tfrac{1}{2} L_1 I_1^2 + \cdots & \leftarrow Step\ \#1 \\ \cdots + \tfrac{1}{2} L_2 I_2^2 + M_{12} I_1 I_2 + \cdots & \leftarrow Step\ \#2 \\ \cdots + \tfrac{1}{2} L_3 I_3^2 + M_{32} I_2 I_3 + M_{31} I_1 I_3 + \cdots & \leftarrow Step\ \#3 \\ \cdots + \tfrac{1}{2} L_A I_A^2 + M_{A3} I_A I_3 + M_{A2} I_A I_2 + M_{A1} I_A I_1 + \cdots & \leftarrow Step\ \#4 \\ \cdots + \int_{z_A=start}^{z_A=end} i_A Speed \left\{ I_3 \frac{\partial M_{A3}}{\partial z} + I_2 \frac{\partial M_{A2}}{\partial z} + I_1 \frac{\partial M_{A1}}{\partial z} \right\} dz & \leftarrow Step\ \#4 \end{array} \right] \\ &= \left[\begin{array}{c} \tfrac{1}{2} L_1 I_1^2 + \tfrac{1}{2} L_2 I_2^2 + \tfrac{1}{2} L_3 I_3^2 + \tfrac{1}{2} L_A I_A^2 + \cdots \\ \cdots + M_{12} I_1 I_2 + M_{32} I_2 I_3 + M_{31} I_1 I_3 + M_{A3} I_A I_3 + M_{A2} I_A I_2 + M_{A1} I_A I_1 + \cdots \\ \cdots + \int_{z_A=start}^{z_A=end} i_A Speed \left\{ I_3 \frac{\partial M_{A3}}{\partial z} + I_2 \frac{\partial M_{A2}}{\partial z} + I_1 \frac{\partial M_{A1}}{\partial z} \right\} dz \end{array} \right] \end{aligned} \quad (55)$$

The first line in Equation (55) is a self-inductance energy term of form $\frac{1}{2}LI^2$, with one such term for each individual coil.

The second line in Equation (55) is a mutual inductance energy term of form $M_{xy}I_xI_y$, with one such term for each different pair of coils[1].

And, now we come to the last line in Equation (55). Here's how we deal with it.

Equation (55) is the sum of the energy that was consumed during the process of bringing the coils to a given configuration. The process we used to derive all of this was based on the product of voltage drop and current, which measures the instantaneous power which the power supplies are delivering to the coils. In our universe, energy does not disappear. If energy was supplied to these coils, they must have done something with it. I said casually in the discussion above that such-and-such an energy was stored in such-and-such a magnetic field. And, that is so. Coils are used precisely because of their ability to take electrical energy and convert it into a magnetic field, which holds energy. The first two lines of Equation (55) are the amounts of energy which are stored in the magnetic field. Of course, there is only one magnetic field which links all four coils, but it is the sum of contributions which arise from ten different interactions[2].

The last line in Equation (55) is merely a different type of energy. It is the kinetic energy of the armature. The armature is the only component in the system which is able to move. All of the location-dependent effects in this system are rooted in the movement of the armature.

I am not going to have to carry out the integration of the last line in Equation (55) in the numerical procedure, although one could. It would not be too difficult. We have already made arrangements to calculate the spatial partial derivatives of the mutual inductance (see the discussion at the bottom of page 3 above), so it would be only a matter of a few multiplications. Instead, I am going to kill two birds, or perhaps even three, with one stone.

In the numerical integration, we are going to use FEMM to calculate the force acting on the armature at the start of every time step. If the time steps are kept short enough, it will be a good assumption that this level of force remains constant during the time step. Using the simple integration results for dynamics driven by constant force, we will be able to calculate the speed and position of the armature at the end of the time step. The kinetic energy of the armature can be written as:

$$KE = \frac{1}{2}Mass \times Speed^2 \quad (56)$$

We will calculate the energy stored in the "self-inductance magnetic fields" using the first line of Equation (55). Similarly, we will calculate the energy stored in the "mutual inductance magnetic fields" using the second line of Equation (55). Separately, of course, we will keep track of the voltage drops over the capacitors, and the electrostatic energy stored inside them. And, we will keep track of the energy burned off as heat.

---

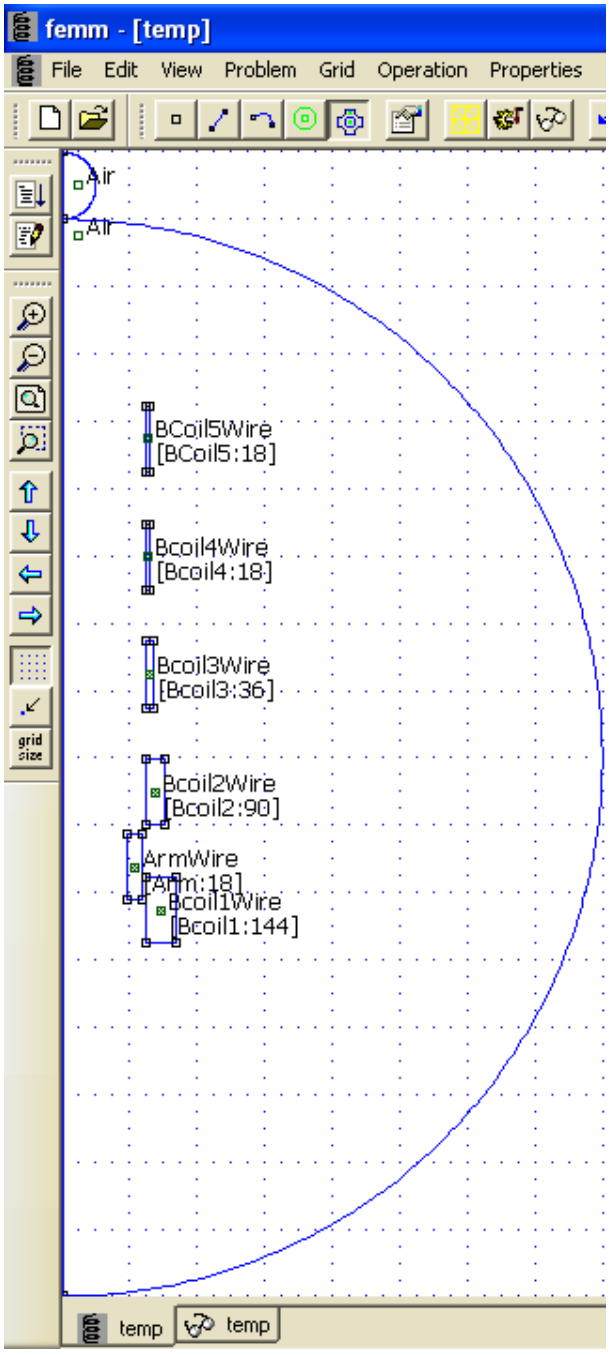
[1] The order in which I chose to power up the coils was completely arbitrary. If one reverses the order in which two particular coils are powered up, the subscripts which show up in the mutual inductances would be reversed. This is one of many reasons why mutual inductance is symmetric, so that $M_{xy} = M_{yx}$.

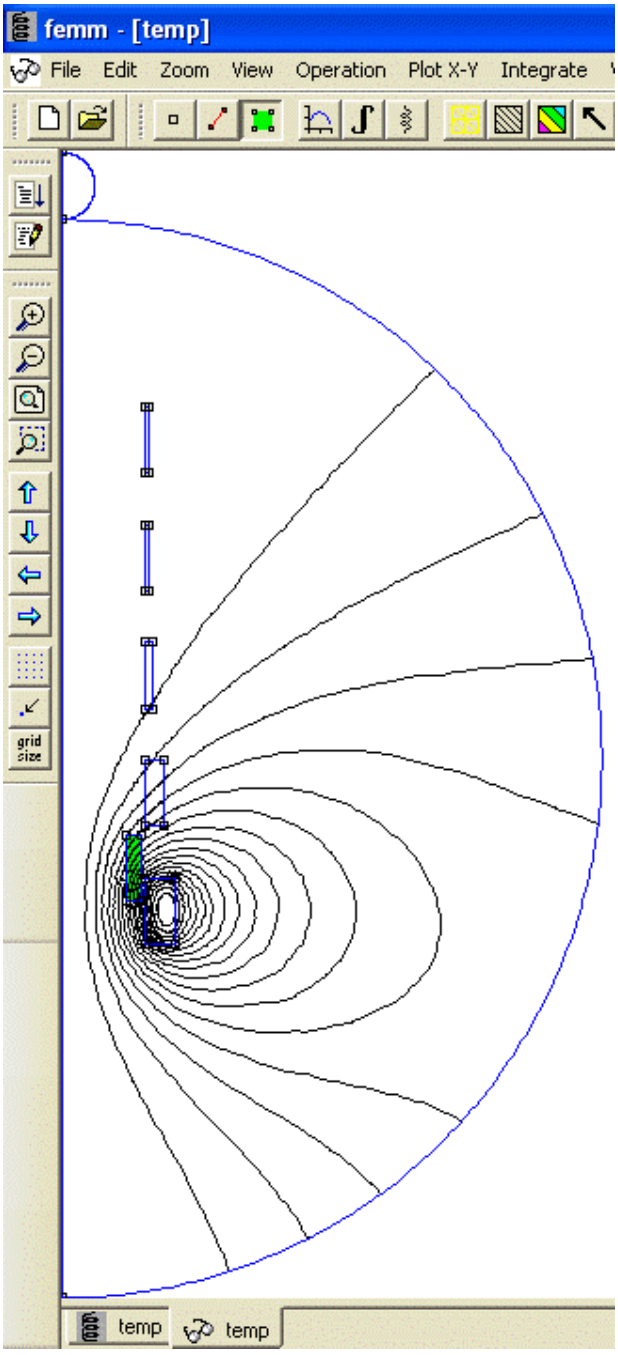
[2] It is worth remembering that each turn in each coil generates its own magnetic field. It's for our human convenience that we group similar turns together into things called coils, which have self-inductances assigned to them. One could do a similar energy analysis on a system-wide turn-by-turn basis, and thus end up with thousands of interactions.

If the sum of all these energies remains constant during the course of the simulation, we will have good confirmation that our circuit analysis is correct, that our energy analysis is correct, and that we are using FEMM in an appropriate way.

The FEMM code for a five-coil gun

I have listed in Appendix "A" the code I used to simulate a five-coil inductance gun. The program is a Lua script which controls the operation of FEMM. The program builds a physical model of the configuration and then proceeds to numerically integrate the circuit equations and the dynamic equations for the armature. The following two figures depict the physical model.





The figure on the left shows the arrangement of the armature and five barrel coils at the start of a run, when there is no current flowing through any of the coils. The barrel coils are wound directly on the outer surface of the barrel, which is a tube with an outside diameter of 2½ inches. The wall thickness of the tube is one-thirty-secondth of an inch. The diameter of the armature must clear the inside of the tube. As is usual in FEMM geometry, the axial z-direction points vertically upwards. During a simulation, the armature travels towards the top.

As a starting point, I have set the nominal length of all coils to one inch. The actual length of a particular coil will be slightly different, since it is equal to an integral number of complete turns of whatever AWG gauge of enameled copper wire is used to wind that coil. For the most part, I set the number of layers in each winding arbitrarily. The exception is the armature. I chose to make the armature using two layers. That way, the armature circuit can be closed physically by soldering the two free ends of the wire across one lip of the armature, rather than having to run a wire up the length of the coil.

The figure on the right shows the magnetic field lines not long after the start of a simulation. A couple of things are noteworthy.

1. When the current starts to ramp up in barrel coil #1, a current in the opposite direction will be induced in the armature. Look back at the schematic diagram. When barrel current i_1 is positive and increasing, the armature current i_A will be algebraically negative.
2. Since the current flows in opposite directions in the armature and barrel coil #1, their magnetic fields will have opposite orientations, and the force they exert on each other will be repulsive.
3. Since the force between the armature and barrel coil #1 is repulsive, the starting location of the armature must be on the down-barrel end (or positive z end) of the first coil, so that it is repelled further down the barrel.
4. It does not matter which way the current flows in barrel coil #1. The current induced in the armature will be in the opposite direction and the force will be repulsive.

There are a couple of features in the Lua script I should mention.

1. I make heavy use of vectors. In Lua, a vector named Q is declared using the statement $\mathrm{Q} = \{\}$. A particular element in that vector is then referenced using square brackets, like this: $\mathrm{Q}[2]$.
2. Lua permits zero-based indexing of elements in a vector. I have taken advantage of that to use index zero for the armature and indices 1 through 5 for the five barrel coils. For example, the armature's name is set using $\mathrm{CoilName}[0] = \text{"Arm"}$.
3. I defined two functions at the start of the script. One is a matrix inversion of the matrix equation $\tilde{A} \cdot \vec{X} = \vec{B}$, where the inductance matrix $\tilde{A}$ and the constant vector $\vec{B}$ are populated before calling the function. I used Euler's method to invert. The matrix is first reduced to upper-triangular form by adding and subtracting rows. Then, starting with the last row and working upwards, each row is solved for an additional unknown in the $\vec{X}$ vector.
4. I used the second function defined at the outset of the Lua script for debugging purposes. It simply writes a copy of the matrix equation to a text file, where it can be inspected for accuracy.
5. If all five barrel coils are operating, the matrix equation will consist of six line equations, each contributing one row. When the script prepares the matrix equation, it does not include barrel coils which are not operating. Because of the physical nature of self-inductance, none of the elements on the main diagonal of matrix $\tilde{A}$ will be zero. Furthermore, because of the nature of mutual inductance, none of the off-diagonal elements will be zero, either. It is for this reason that I did not include any error checking in the inversion function.

6. At the start of every time step, after the armature has been moved to its new location, FEMM calculates new values for the mutual inductances and new value for the force exerted on the armature. FEMM also estimates the spatial derivatives of the mutual inductances $\partial M_{Aj}/\partial z$ between the armature and barrel coils $j = 1, 2, 3, 4, 5$. To do this, FEMM moves the armature to a slightly different axial location and recalculates the mutual inductances. The spatial derivative is estimated using the ratio of differences $\Delta M_{Aj}/\Delta z$. In the code, this distance Δz is represented by the variable Joggle. I shall have more to say about Joggle.

7. The code re-calculates the total system energy at the start of every time step, and the error/difference between this energy and the total system energy at the start of the simulation.

8. All results are written into a text file. The text file will be created in the same directory as the Lua script. The output is written as csv (comma-separated values) so it can easily be imported into an Excel spreadsheet.

Results of a test simulation

In this section, I will describe the results of a test simulation. It is not my intention in this paper to do any "optimization" of coil parameters, capacitor values and so on. My primary purpose is to ensure that energy is conserved. There is no point doing a lot of work with details only to find later a fundamental flaw in the methodology.

The test simulation used only the first two barrel coils. The principal parameters were:

1. The armature is two layers of AWG #10 wire with nine turns in each layer.

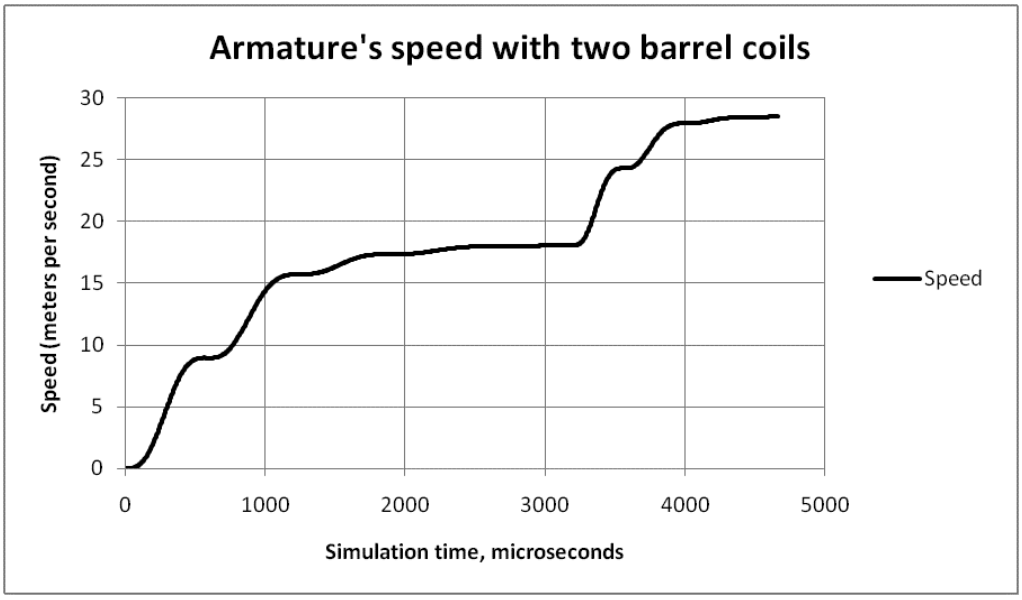
2. Barrel coil #1 is eight layers of AWG #16 wire with 18 turns in each layer.

3. Barrel coil #2 is five layers of AWG #16 wire with 18 turns in each layer.

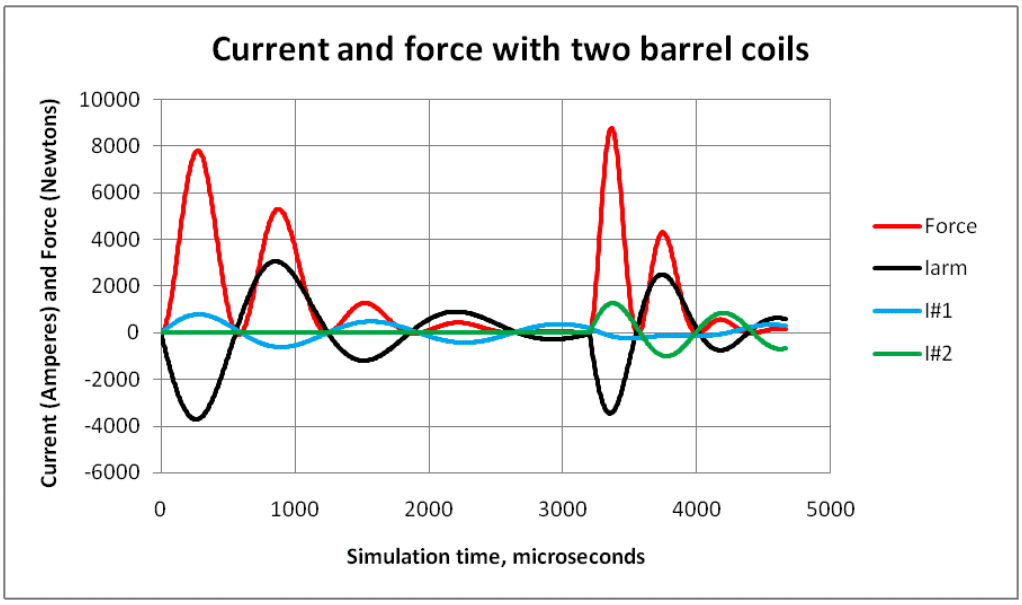
4. 32μF capacitors power both barrel coil circuits. They are initially charged to 5,000 Volts.

5. At the start of the simulation, the armature is placed with its leading edge one-half inch past the positive end of barrel coil #1. The circuit for barrel coil #2 is closed when the armature's leading edge is one-inch inch past the positive end of barrel coil #2.

The following graph shows the speed of the armature for about four and one-half milliseconds.

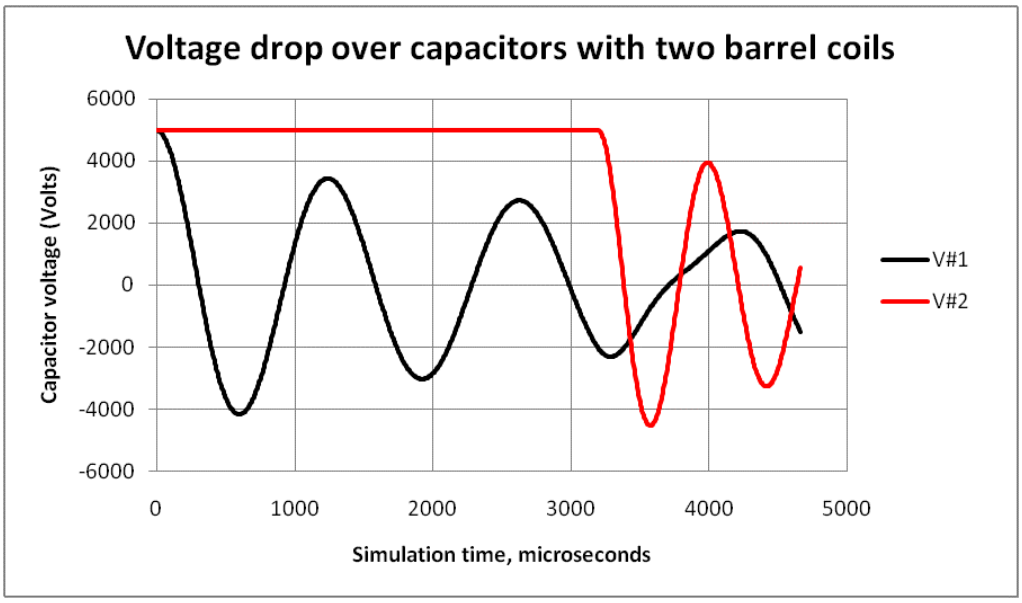


The big acceleration which starts at around 3¼ milliseconds is caused when barrel coil #2 comes on line. The many bumps in the speed curve represent acceleration peaks which, in accordance with Newton's Laws, result from peaks in the force. Whatever is going on is certainly not a smooth and uniform process. That is clear in the following graph, which shows the current flowing in the armature (black), the currents flowing in barrel coils #1 and #2 (blue and green, respectively) and the force exerted on the armature (red).



The inductances and capacitances are such that the circuits' responses are very oscillatory. The energy sloshes between the capacitors and the coils several times. In barrel coil #1's circuit, the period of these oscillations is about 1¼ milliseconds. Since barrel coil #2 has fewer layers, and therefore less inductance, its oscillation period is less, approximately three-quarters of a millisecond. Fortunately, the interaction between the armature and the barrel coils is such that the induced currents are in opposition so the force is almost always favourable for accelerating the armature down the barrel.

The following graph shows the voltages on the two capacitors.



The voltage waveforms are decaying sinusoids. But, the degree of oscillation is so great that the energy in the capacitors' electrostatic fields is not being transferred into armature kinetic energy as quickly and efficiently as one might wish. A lot more attention must be given to the component values to improve the energy transfer. This will be the subject of subsequent papers.

Another possible problem is apparent from the foregoing graph of the voltage drops over the capacitors. Because of the oscillation, the capacitors do far more than simply discharge. They recharge in the opposite direction, almost up to the original starting voltage. For most polarized capacitors, this kind of reverse charge would be fatal. This, too, will have to be examined in more detail.

Conservation of energy during the simulation

In this section, I will discuss the topic which I think is the most important one at this early stage of the analysis. When the circuits start up from rest, as they will do when the gun is fired, the only energy in the system is contained in the two capacitors. I have said that the capacitors are $32\mu F$ and that they are initially charged up to $5{,}000$ Volts. Each capacitor contains energy of:

$$E_{cap} = \tfrac{1}{2}CV^2 = \tfrac{1}{2} \times 0.000032 \times 5000^2 = 400 \text{ Joules} \quad (56)$$

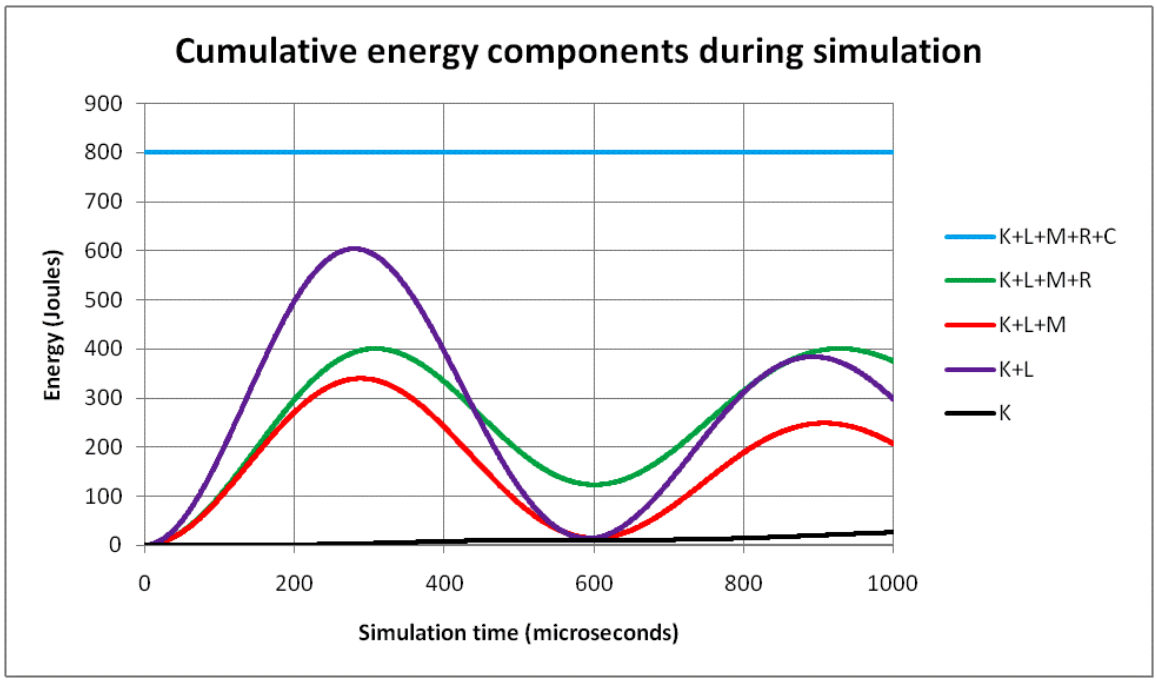
and the total energy in the system is 800 Joules.

Let's compare that to the kinetic energy of the armature at the end of the simulation. From the speed graph above, we can see that the armature reaches a final speed of about 28 meters per second. I have assumed the armature has a mass of 0.25 kg, which is about half a pound. Its kinetic energy is:

$$KE_{arm} = \tfrac{1}{2}Mass \times Speed^2 = \tfrac{1}{2} \times 0.25 \times 28^2 = 98 \text{ Joules} \quad (57)$$

One can think of the "efficiency" of this gun as the percentage of the original electrostatic energy which is converted into kinetic energy. Here, the efficiency is $98/800 = 12.3\%$.

The following is a graph of the various "pools" of energy during the first millisecond of the simulation period.



The blue curve is the total energy – it remains constant at 800 Joules. The total energy is the sum of the armature's kinetic energy (K), the magnetic energy stored by means of the coils' self-inductances (L), the magnetic energy stored by means of the mutual inductances among the three coils (M), the cumulative heat burned off by the resistances (R) and the electrostatic energy stored in the capacitors (C). The graph adds up these pools of energy in "layers", so that each curve is the sum of the energy in all the pools below. A couple of these energies are easy to understand.

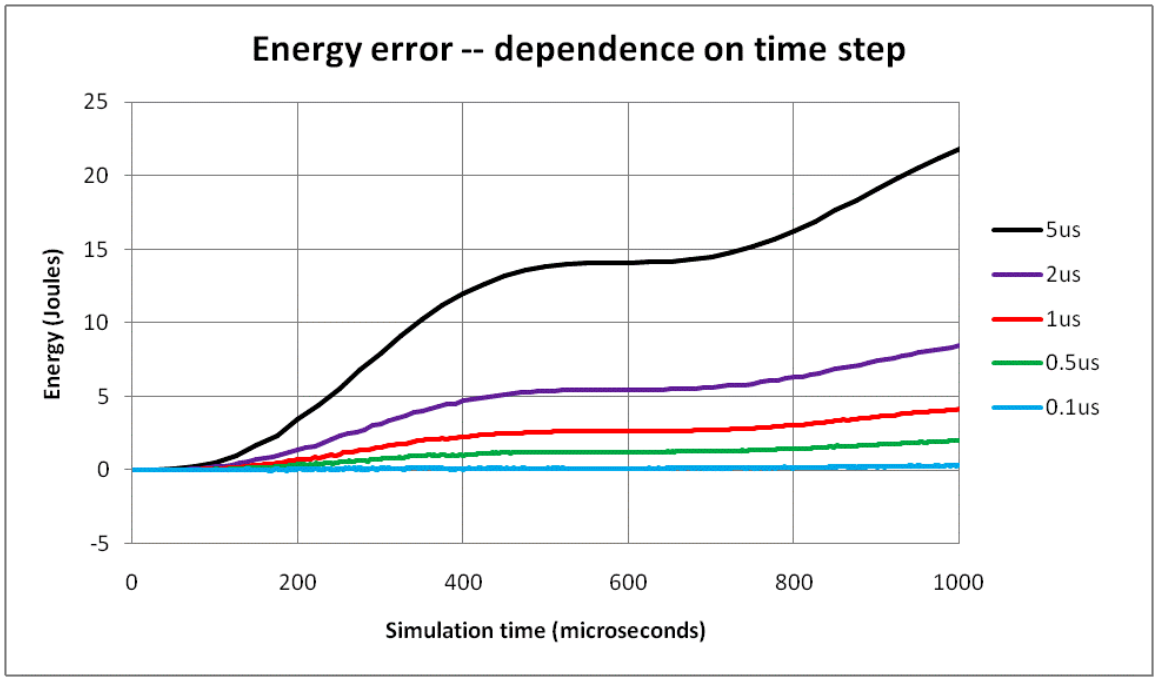
The black curve at the bottom is the kinetic energy of the armature. Roughly speaking, it increases quadratically with respect to time.

The energy burned off as heat is the space between the red curve and the green curve above it. This use of energy also grows monotonically with time, but not at a quadratic rate.

Mutual inductance (in this case) actually looks like a negative energy. Let me explain. The purple curve is the sum of the armature's kinetic energy and the magnetic energy stored in the magnetic fields generated by the three coils' self-inductances. Adding the effect of mutual inductance reduces the total energy, causing the red curve to lie below the purple curve. When two coils interact like they do in the coil gun, with the barrel coil driving the current in the armature, the effect of mutual inductance is to reduce the total amount of stored magnetic energy. The reduction in stored energy caused by the mutual inductance will never be greater than the total stored energy due to self-inductance. The mutual inductance reduces the total magnetic energy, but the reduction will never be great enough to cause "negative" magnetic energy in total.

The oscillations in the magnetic curves L and M arise as the system's energy sloshes between its electric form, when stored in the capacitors, and its magnetic form, when stored in the magnetic field. Note, agin, that this is only the first one millisecond of the simulation. Since barrel coil #2 is not activated until later in the simulation, the 400 Joules of energy stored in its capacitor is unchanged through the duration of the graph.

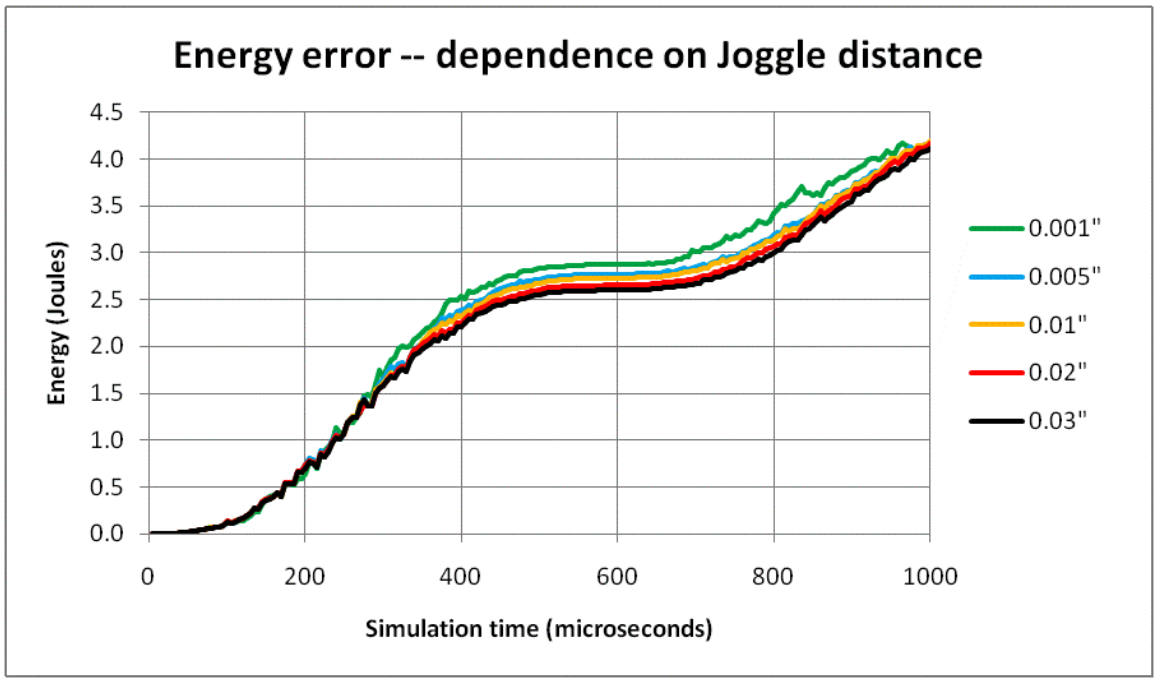
The blue curve in the graph above is the system's total energy. On the scale of the graph, it looks perfectly constant. If one looks in greater detail, it is not exactly constant. The following graph shows the error which arises during this one millisecond time period for several different sizes of time step.



The "errors" plotted in this graph are the differences between the system energy calculated at the start of each time step, and the total energy at the start of the simulation. A perfect numerical simulation would have zero error. The blue curve gets pretty close to that ideal. The blue curve is the error when the time step is set to one-tenth of one microsecond, or 100 nanoseconds. The energy error is a fraction of a Joule. The approximations relied on to justify integrating numerically get less realistic as longer and longer time steps are used. For example, the assumption that the derivative of the current remains constant during the whole of each time step gets less accurate as the time steps get longer. A time step of one millisecond gives rise to an energy error of about four Joules.

A low energy error is an excellent goal, but shorter time steps require more computer time to carry out. For preliminary coil selection, a $1\mu s$ time step would probably suffice. The amount of error that is acceptable depends on how much kinetic energy is imparted to the armature.

Although the length of the time step is arguably the most important simulation parameter, it is not the only one. Another is the Joggle distance, which is the incremental displacement given to the armature's location for the purpose of estimating the spatial derivatives $\partial M_{Aj}/\partial z$. The following graph shows the effect on the energy error of using different Joggle increments. (Incidently, all the runs made to prepare the previous graph, in which different time steps were compared, used a Joggle increment of 0.02 inches.) All of the runs made to prepare the following graph, in which Joggle increments are compared, used a time step of 1 μs.



The energy error does not behave quite as one might expect. One might think that reducing the Joggle distance would give a better estimate of the spatial derivatives $\partial M_{Aj}/\partial z$, but that is not so. Making the Joggle distance too small is actually counter-productive. The reason is this. In order to carry out its own analysis, FEMM discretizes the universe into little triangles. These little triangles (in two dimensions) represent small bits of volume. For practical reasons, they cannot be made infinitely small. The default triangles used by FEMM have side lengths of 0.01 inches. If the armature is moved by a distance which is too small compared with the triangulation, FEMM's ability to resolve the change is reduced.

Based on what I have seen, the Joggle distance should be set to 0.02 or 0.03 inches. Making the Joggle distance bigger than this introduces a different problem – the difference-based approximation of the slope of the mutual inductance curves begins to fail.

I have attached as Appendix "A" a copy of the Lua script used for these simulations.

Jim Hawley

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"

The Lua script

```
--////////////////////////////////////////////////////////////////////////
--// 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-maxtix-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 = 0.000001         -- 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] = 16
```

```
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] = 8
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 enamelled 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 enamalled 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 offests 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 deault 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 ofset is
-- simply not relevant.  On the other hand, i 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

-- 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] = 5000
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 subscricpt "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 = {}

-- 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.
deltaER = {}

-- 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
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
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],",")
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,"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
    end
    ERTotal = 0
    ERTotal_0 = 0
else
    for I= 0,5 do
       if (CoilConnect[I] == 1) then
            ERTotal = ERTotal + deltaER[I]
        end
    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
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.
-- 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 and are connected.
for I= 1,5 do
    if (CoilConnect[I] == 1) then
        Q_end[I] = Q_start[I] +
            (-I_start[I] * deltaT) +
            (-0.5 * dIdt_const[I] * deltaT * deltaT)
    else
        -- Hold over the charge on the other capacitors
        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
end

-- Step #21: 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 #22: 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],",")
    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,"ETotal_start(J)=,",ETotal,",")
    write(handle,"EError_start(J)=,",EError,"\n")
end

-- Step #23: 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 #24: 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 #25: Turn off any operating coil circuit controlled by an SCR
    -- whose current flow has become negative.  Set CoilConnect[] = -1 to
    -- avoid inadvertantly 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 #26: Write a warning, and terminate the simulation, if the voltage
    -- drop over any operating capacitor becomes negative.
--    for I= 1,5 do
--        if (CoilConnect[I] == 1) then
--            if (Vcap[I] < 0) then
--                write(handle,"ERROR -- Negative voltage on Cap ;",I,"\n")
--                TimeStep = MaxNumTimeSteps + 1
--            end
--        end
--    end

    -- Step #27: 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)
    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.")
```