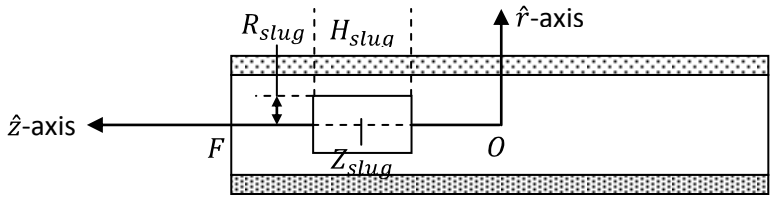
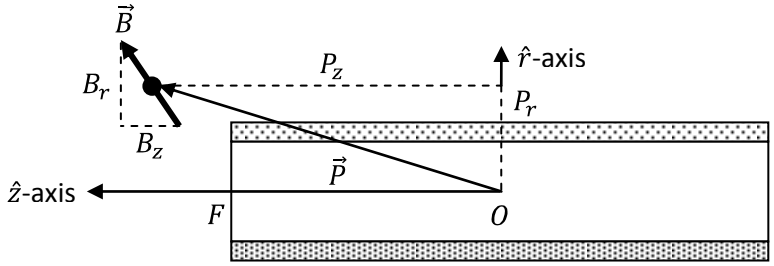
Physics of an axial coil gun

Part III – Spatial distribution of the force field

In this part of the paper, we will look at the spatial distribution of the force exerted by the solenoid on the slug. This is a topic I have already examined, in another paper titled *The force on a cylindrical steel slug inside a finite solenoid*. I will give only a brief explanation of things here. For more detail, see the earlier paper. The following figure shows a longitudinal cross-section of our coil, or solenoid. The origin O is located at the geometric center of the coil and the $\hat{z}$-axis is coincident with the coil's central axis. Point F is the point of intersection of the $\hat{z}$-axis and the end face of the coil on the side from which the slug starts its run. The slug is located somewhere along the central axis. In the figure, it is shown inside the coil. We will state the slug's position by referring to its geometric center. In this part of the paper, I will use the symbol Z_{slug} for the instantaneous position of the slug, to distinguish it from other varieties of z used for other purposes. The slug is assumed to be homogeneous and to have a radius and length of R_{slug} and H_{slug}, respectively.



The whole configuration is symmetric around the $\hat{z}$-axis. Therefore, the radial axis shown, and labeled as the $\hat{r}$-axis, is typical of any ray which passes through the origin and is perpendicular to the $\hat{z}$-axis. When the coil is powered, a magnetic field is set up inside and around the coil. The magnetic field has both a magnitude and a direction, so it is represented as a vector $\vec{B}$. Since the apparatus is radially symmetric, the vector $\vec{B}$ has components B_z and B_r in the $\hat{z}$- and $\hat{r}$-directions, respectively, but does not have any component which would be into or out of this page. So long as the current which powers the coil is constant, the magnetic field does not vary with time. But, it does vary with position. If $\vec{P}$ is a vector pointing from the origin to some point of interest, and if $\vec{P}$ has the components P_z and P_r, then we can write the magnetic field strength at the point as $\vec{B}(I, P_z, P_r)$, where its dependence on the current and the two spatial variables has been explicitly called out.



In the earlier paper, we saw how to calculate the components of the magnetic field at any point $\vec{P}$, either inside or outside of the coil, but not in the winding itself. The components are found by summation, as follows:

$$B_r = \frac{\mu_0 I}{4\pi} \sum_{m=1}^{m=N_{layers}} \sum_{n=1}^{n=N_{turns}} \sum_{\theta=0}^{\theta=2\pi} \frac{r_{Mth\ layer}(P_z - z_{Nth\ turn})\Delta\theta \cos\theta}{\left[P_r^2 + (P_z - z_{Nth\ turn})^2 + r_{Mth\ layer}^2 - 2P_r r_{Mth\ layer}\cos\theta\right]^{3/2}} \quad (1A)$$

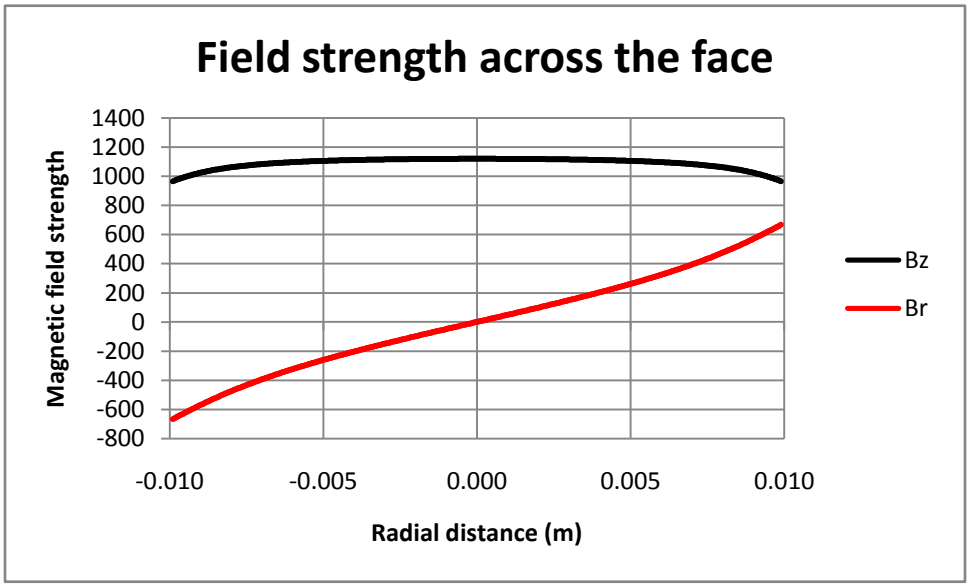
and

$$B_z = \frac{\mu_0 I}{4\pi} \sum_{m=1}^{m=N_{layers}} \sum_{n=1}^{n=N_{turns}} \sum_{\theta=0}^{\theta=2\pi} \frac{-r_{Mth\ layer}\left(P_r \cos\theta - r_{Mth\ layer}\right)\Delta\theta}{\left[P_r^2 + (P_z - z_{Nth\ turn})^2 + r_{Mth\ layer}^2 - 2P_r r_{Mth\ layer}\cos\theta\right]^{3/2}} \qquad (1B)$$

The outer summation is a summation over all of the layers in the winding, where $r_{Mth\ layer}$ is the radius to the centerline of the M^{th} layer in the winding. The middle summation is a summation over all of the turns in each layer, where $z_{Nth\ turn}$ is the axial displacement from the origin to the center of the plane which contains the N^{th} turn in each layer. The inner summation is a summation around the circumference of the N^{th} turn in the M^{th} layer, where progress around the circumference is measured by angle θ. In all instances, distances are measured to the center of the wire. The current is assumed to flow along the centerline of the wire or, alternatively, to flow uniformly across the area of the wire. The symbol μ_0 represents the permeability of free space, equal to $4\pi \times 10^{-7}$ H/m.

Note that both components of the magnetic field are directly proportional to the current I.

As an example, I have calculated and plotted the components of the magnetic field across the face of the basic coil we used in Part II of this paper. It has 96 turns of #4 gauge enameled wire wound on a core with a diameter of 2 cm. Since the wire has a diameter of 5.189 mm, the coil is 49.81 cm long. The following graph shows the components across the face where the horizontal axis measures the distance from the solenoid's centerline. The values plotted do not include the factor $\mu_0 I\ /4\pi$, so I have not labeled the units of B_r and B_z in the graphs. When the values shown are multiplied by the absent factor, the S.I. units of the magnetic field strength will be Tesla.

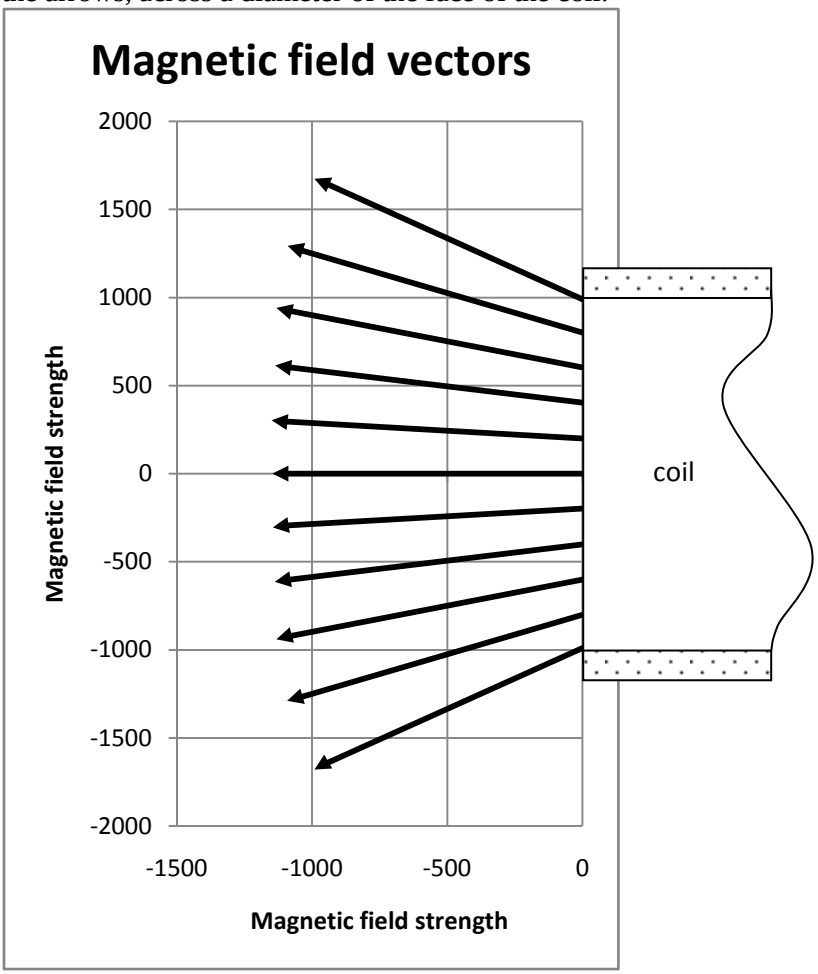


The "face" referred to is, of course, the left-hand face of the solenoid as drawn in the figure above. The axial component of the magnetic field B_z has positive values. That means that it points in the direction of the positive $\hat{z}$-axis – towards the left in the diagram. The strength of the axial component (incidentally, "axial" means that this component points along the axis of the coil) is almost constant across the face. In fact, uniformity across the face is one of the characteristics of a coil which is long in comparison to its diameter. When that is the case, the coil is usually called a solenoid.

The radial component B_r is not constant across the face. It points outwards from the coil's axis. This is the reason why its plot is asymmetric around $r = 0$. $r = 0$ defines the $\hat{z}$-axis. For points which lie below the $\hat{z}$-axis in the diagram above ($r < 0$), the radial component is negative, meaning that it points in the direction of the negative $\hat{r}$-axis, that is, downwards.

Note that the coil has an inner, or core, radius of 1 cm. The range of radii plotted in the graph is apparently from -1 cm to +1 cm. But, if you look closely, you will see that the curves stop just before they reach the edge of the coil. The curves shown are not plotted all the way from -1 cm to +1 cm. Instead, they are plotted for the range of radii -0.99 cm to +0.99 cm. Things change very quickly in that last tenth of a millimeter. As one approaches the winding, the radial component of the field skyrockets. From a physical point of view, that is not very important. We would never be able fire a real coil gun whose slug fit so snuggly within the coil.

To aid in visualizing the magnetic field, the following graph shows the two components plotted as vectors. The relative strength and direction are apparent. Note that the points to which the vectors apply are the points at the tail of the arrows, across a diameter of the face of the coil.



As a quick check, let us calculate the magnitude of the magnetic field strength at the center of the face. At that point, there is no radial component, so the field is entirely axial, pointing out of the coil. The plotted value at the center of the face is 1119, which corresponds to a physical value of:

$$
\begin{aligned}
B_z|_F &= \frac{\mu_0 I}{4\pi} \times 1119 \\
&= \frac{4\pi \times 10^{-7} \times 11119}{4\pi} I \\
&= 0.0001119 \times I \text{ Tesla} \quad (2)
\end{aligned}
$$

The magnetic field strength at the center of a solenoid's face is commonly estimated to be one-half its value at the geometric center of the coil. (Imagine cutting the coil in two at its center, and separating the two halves.) It is also well-known that the magnetic field strength at the center of a very long coil can be approximated by:

$$
B_{very\ long\ solenoid} = \frac{\mu_0 I N_{total}}{H_{coil}} \quad (3)
$$

where N_{total} is the total number of turns in the solenoid. (It is usual to express this relationship with the fraction N/H_{coil} said to be the turns per unit length along the solenoid.) Using our number of turns (96) and our coil length (49.81 cm), the field strength at the center of the face can be estimated as:

$$
\begin{aligned}
\tfrac{1}{2}B_{very\ long\ solenoid} &= \frac{4\pi \times 10^{-7} \times 96}{2 \times 0.4981} I \\
&= 0.000121 \times I \text{ Tesla} \quad (4)
\end{aligned}
$$

This is in good agreement with our calculation. Since there are not very many turns, and since we have accounted for them individually, the value from the summation in Equation (2) is probably better than the common estimate.

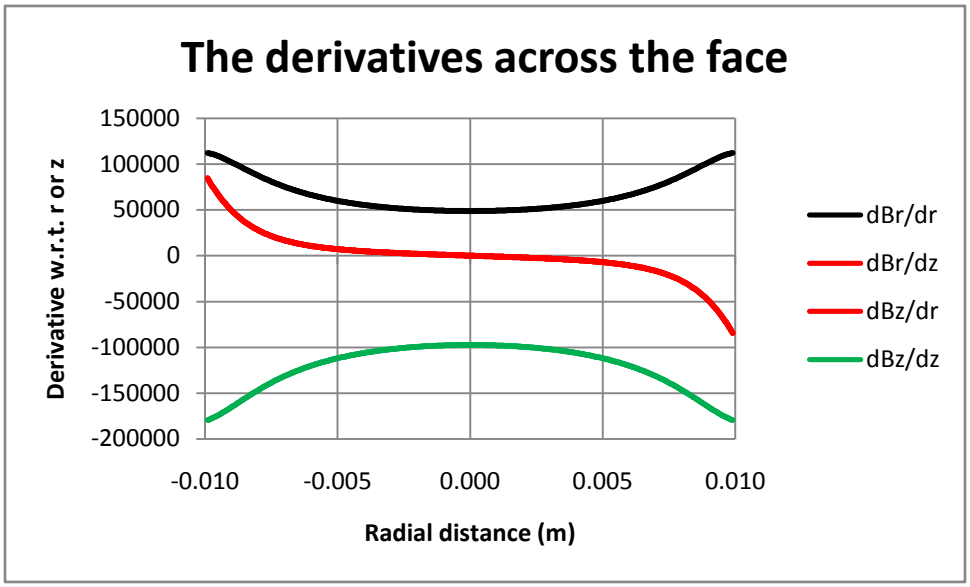
The expressions in Equation (1) only apply when the point of interest $\vec{P}$ lies in air or "free space". It does not apply if point $\vec{P}$ happens to be inside the slug. If the slug is made of steel or some other ferromagnetic material, then the strength of the magnetic field is greatly increased at points inside the slug. The free electrons inside a ferromagnetic material respond to the externally applied field. One can think of their response as orbiting in small circles. In addition, the directions of the spins of individual electrons tend to line up with the external field. Both of these responses act to increase the local strength of the magnetic field, but they do not change its direction. The increase can be enormous, by factors of hundreds or thousands for "good" ferromagnetic materials. The multiplicative factor of a material is described by its relative permeability μ_r. For a mild steel, the relative permeability usually exceeds 200 and can range up to the thousands, depending on the treatment the steel has received. Note that the relative permeability is a property of the material. It does not depend on the shape of the object which encloses the subject point. It is worth understanding that magnetic properties like these are macroscopic properties. They are not useful for describing the behavior of individual electrons or atoms, but are useful when there are enough of them in a volume so that their behavior can be described statistically.

The motion of the free electrons in the steel in response to the external field does more than just increase the local field strength. Electrons are charged particles, so their collective motion can be described as a current, another macroscopic quantity. The interaction of this current and the applied magnetic field, as augmented, is such that a net force is exerted on the slug. The force is attractive in that it will tend to pull the slug towards the center of the coil. Since the force will vary from place to place within the slug, it is useful to think of the force as being a "force acting per unit volume" at one particular spot or another inside the slug. The physics sort themselves out in such a way that we can write the force per unit volume at any particular point inside the slug as follows:

$$\vec{f}(P_r, P_z) = \frac{\mu_r^2}{\mu_0}\left(B_r \frac{\partial B_r}{\partial r} + B_r \frac{\partial B_z}{\partial z} + B_z \frac{\partial B_r}{\partial z}\right)\hat{r} + \frac{\mu_r^2}{\mu_0}\left(B_z \frac{\partial B_z}{\partial z} + B_r \frac{\partial B_z}{\partial r} + B_z \frac{\partial B_r}{\partial r}\right)\hat{z} \quad (5)$$

where the force per unit volume at point (P_r, P_z) is found by evaluating the right-hand side of Equation (5) at a radial displacement of $r = P_r$ and an axial displacement of $z = P_z$. The right-hand side of Equation (5) involves, not just the two components B_r and B_z of the magnetic field strength, but their spatial derivatives as well. For example, $\partial B_r / \partial z$ is the rate with which the radial component of the magnetic field B_r changes as one moves in the axial $\hat{z}$-direction. The four partial derivatives can be derived from Equation (1), but the expressions are complicated and I will not repeat them here.

Instead, I have set out here a graph showing the four derivatives across a diameter of the face of the coil, at the same points which were the horizontal axis in the graphs above.



As before, the values shown for the derivatives do not include the factor $\mu_0 I / 4\pi$. The derivatives, which are changes in the field strength divided by the corresponding changes in distance, have relatively large values because the field strength changes rapidly over relatively small distances.

The first thing to note is that two of the derivatives – $\partial B_r / \partial z$ and $\partial B_z / \partial r$ – are the same. The second thing to note is that each derivative is either symmetric or asymmetric around the coil's centerline $r = 0$. Both of these facts will simplify the numerical computations by avoiding certain redundant calculations. The third thing to note is that the shape of two of the derivatives – $\partial B_r / \partial r$ and $\partial B_z / \partial r$ – can be confirmed intuitively by looking at the graph above of B_r and B_z with respect to the radius r. These two derivatives are the rates at which B_r and B_z change as one progresses from left to right in the previous graph.

Let us take advantage of the symmetries in our physical situation. The force per unit volume in Equation (5) is the sum of six terms, each of which is the product of one of the two components of the magnetic field strength and one of the four derivatives. When we add up the forces per unit volume at all the little bits of volume which make up the slug, some of these terms will add up to zero.

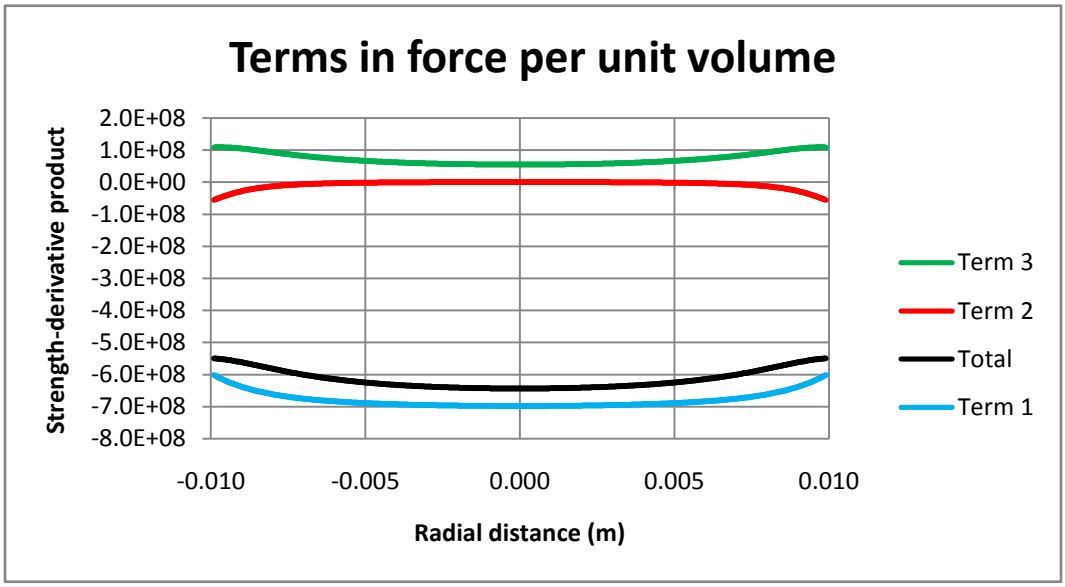
Whether a cylindrical slug is fat or skinny, or long or short, it will be the case that every particular little bit of volume will have a corresponding bit of volume with the same size and shape, but diametrically opposed to it through the $\hat{z}$-axis. The forces on these two corresponding bits of volume will either add to

each other or cancel each other out. The following table sets out how the forces interact for two such corresponding bits of volume. For the purpose of the table, the centers of the two corresponding bits of volume are assumed to be located at an axial displacement of z_* and at radii of $\pm|r_*|$. The straight brackets | | represent the absolute value. Here we go.

	Point at $(z_*, +\|r_*\|)$	Point at $(z_*, -\|r_*\|)$	Sum of terms
Value of B_z	$+\|B_z\|$	$+\|B_z\|$	
Value of B_r	$+\|B_r\|$	$-\|B_r\|$	
Value of $\partial B_r/\partial r$	$+\|\partial B_r/\partial r\|$	$+\|\partial B_r/\partial r\|$	
Value of $\partial B_r/\partial z$	$+\|\partial B_r/\partial z\|$	$-\|\partial B_r/\partial z\|$	
Value of $\partial B_z/\partial r$	$+\|\partial B_z/\partial r\|$	$-\|\partial B_z/\partial r\|$	
Value of $\partial B_z/\partial z$	$+\|\partial B_z/\partial z\|$	$+\|\partial B_z/\partial z\|$	
Term 1: $B_r\ (\partial B_r/\partial r)$	$+\|B_r\|\|\partial B_r/\partial r\|$	$-\|B_r\|\|\partial B_r/\partial r\|$	0
Term 2: $B_r\ (\partial B_z/\partial z)$	$+\|B_r\|\|\partial B_z/\partial z\|$	$-\|B_r\|\|\partial B_z/\partial z\|$	0
Term 3: $B_z\ (\partial B_r/\partial z)$	$+\|B_z\|\|\partial B_r/\partial z\|$	$-\|B_z\|\|\partial B_r/\partial z\|$	0
Term 4: $B_z\ (\partial B_z/\partial z)$	$+\|B_z\|\|\partial B_z/\partial z\|$	$+\|B_z\|\|\partial B_z/\partial z\|$	$+2\|B_z\|\|\partial B_z/\partial z\|$
Term 5: $B_r\ (\partial B_z/\partial r)$	$+\|B_r\|\|\partial B_z/\partial r\|$	$+\|B_r\|\|\partial B_z/\partial r\|$	$+2\|B_r\|\|\partial B_z/\partial r\|$
Term 6: $B_z\ (\partial B_r/\partial r)$	$+\|B_z\|\|\partial B_r/\partial r\|$	$+\|B_z\|\|\partial B_r/\partial r\|$	$+2\|B_z\|\|\partial B_r/\partial r\|$

So, three of the six terms will cancel out, including all three of the terms which make up the radial component f_r of the force per unit volume. This makes sense. It is difficult to see how a physical configuration in which everything is symmetric around an axis could give rise to a non-axial force. On the other hand, all three terms which make up the axial component f_z have non-zero contributions.

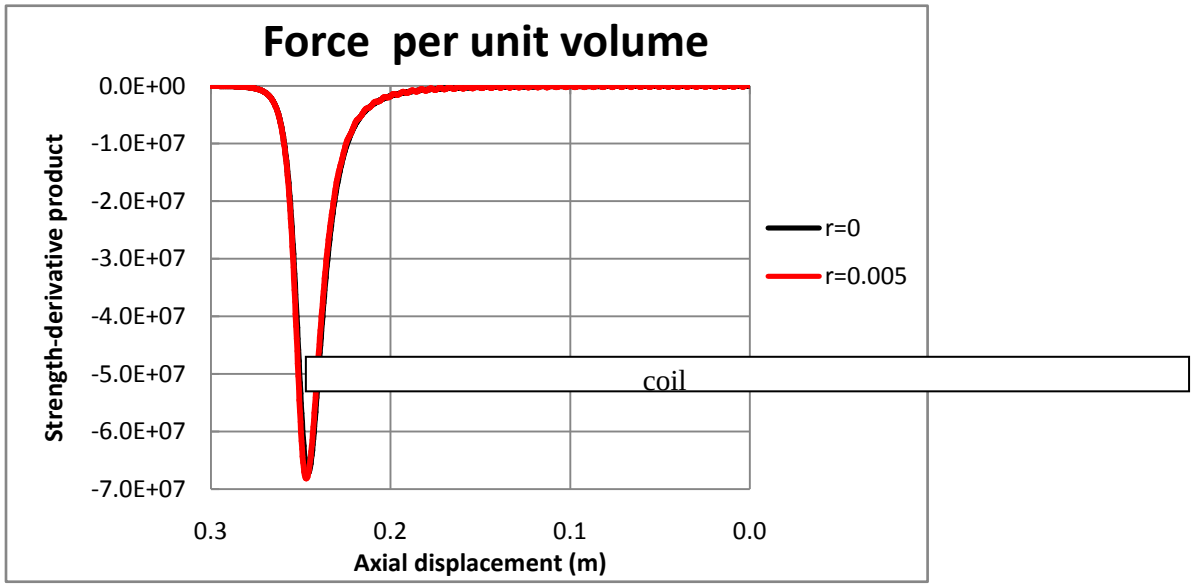
The following graph shows the relative strength of the three terms which add up to the total axial force f_z per unit volume. Once again, the values are shown across a diameter of the face of the coil, at the same points which were the basis of the graphs above. The values do not include the factor $(\mu_r^2/\mu_0)(\mu_0 I\ /4\pi)^2$. The first factor (μ_r^2/μ_0) arises in Equation (5) itself. The strength of the field contributes one of the two $(\mu_0 I\ /4\pi)$ factors and the derivative contributes the other.



It is clear that the first term in the $\hat{z}$-component of Equation (5) is the dominant term, that is, the one with the greatest magnitude. This is the blue line in the graph. Some of its effect is cancelled by the third term, shown in green, which has the same general shape but acts in the opposite, repulsive, direction. The

second term, shown in red, does not contribute much except for a small blip near the winding. The sum of the three terms, which is shown in black, is fairly constant across the face of the coil. It has a negative algebraic value, meaning that the total force acts in the direction of the negative $\hat{z}$-axis. This is towards the center of the coil so it is an attractive force.

We have been looking at the magnitude of the force per unit volume across the face of the coil. How does the magnitude change along the axis of the coil? This is something we called the "spatial distribution of the force field" in earlier parts of this paper.



Two curves are shown in the graph. The black one is the force per unit volume along the $\hat{z}$-axis. The red one is the force per unit volume along a line which is one-half centimeter from the $\hat{z}$-axis, or half of the core radius. The two curves are almost coincident, which confirms again that the force is pretty much uniform across the cross-section of the coil. The force per unit volume, being attractive, is algebraically negative.

For reference, I have also shown an outline of the coil with its length and diameter consistent with the distance used for the horizontal axis. The coil is 49.81 cm long and its center is located at the axial displacement $z = 0$. The force peaks very close to, but just inside, the face of the coil. The force per unit volume decreases to zero very quickly outside the coil. It decreases almost as quickly inside, too. For a long and skinny coil, like this one, the slug is going to be accelerated only when it is quite close to the face. Most of the coil is going to be useless. The force per unit volume is non-zero over a distance of about 10 cm, or about four inches, which is only a modest fraction of the coil's total length.

In fact, the uniformity of the magnetic field inside the coil – which is a principal characteristic of a solenoid – makes it a poor choice for a coil gun. Uniformity means that the strength of the magnetic field does not change much with distance. However, the derivatives measure exactly this – how quickly the field changes with distance. The terms in the force per unit volume are proportional to the derivatives – more uniformity means smaller derivatives and less force. In fact, we may have to change the coil so that it has a bigger diameter compared to its length. This will cause the magnetic field to diverge more quickly with distance, particularly near the face, which will in turn increase the force per unit volume and the effectiveness of the coil gun.

Let us pursue this line of thought. Let us see how the spatial extent of the force field changes as we change the dimensions of the coil. I would like to accomplish more in this analysis than simply

calculating the spatial extent for different lengths and diameters. I would like the coils being compared to be somehow "equal" so that we get some idea of their relative suitability for a coil gun. What I propose to do, therefore, is to compare coils wound using the same length of wire. The coils will then have the same electrical resistance. Their inductances will be different, of course, but we will at least have one parameter which is the same for all the coils. The base case coil above had 96 turns wound on a core with a 2 cm diameter. The length of wire used is:

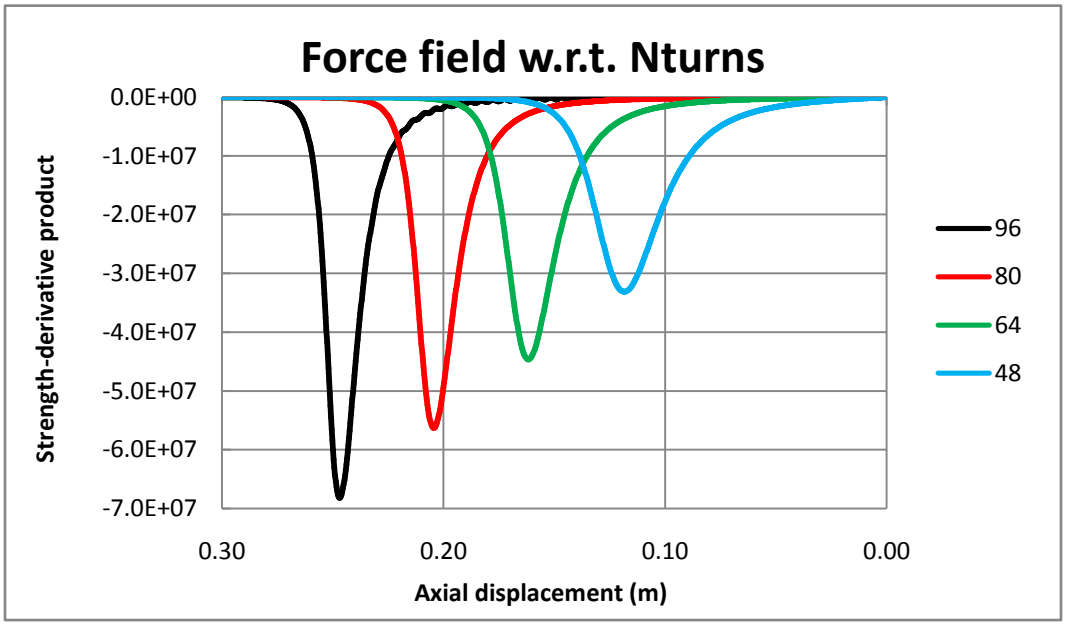
$$\begin{aligned} \text{Length} &= N_{turns} \times 2\pi \times (R_{core} + \tfrac{1}{2}D_{wire}) \\ &= 96 \times 2\pi \times (0.01 + \frac{0.005189}{2}) \\ &= 7.597 \text{ meters} \end{aligned} \qquad (6)$$

Let us wind some other coils using the same length of wire. The following table shows certain other suitable combinations. All of them have a single layer of turns.

N_{turns}	$R_{core}\ (m)$	$H_{coil}\ (m)$
96	0.01000	0.4981
80	0.01252	0.4151
64	0.01630	0.3321
48	0.02260	0.2491

Do not worry about the precision with which these core radii and lengths are shown. This will be a mathematical comparison only – I am not suggesting that any particular coil could be built with this accuracy.

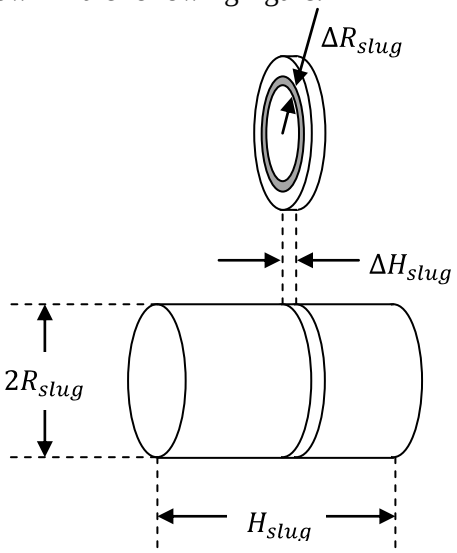
The following graph shows the spatial distribution of the force per unit volume for each of these coils. The force is shown along an axial line which is one-half the radius of the core away from the $\hat{z}$-axis, in other words, one quarter of the core's diameter away from the centerline.



As the number of turns decreases, and the core radius increases, the horizontal extent of the force field – the axial distance over which it acts – increases. That is a "good" thing so far as accelerating a slug is

concerned. But the magnitude of the force decreases, which is a "bad" thing. What is important, though, is the area enclosed by the curves. Remember that the energy imparted to the slug is the applied force multiplied by the distance through which the slug travels. The total of this product for a complete run is the area enclosed by the curves above. At least to the naked eye, the areas enclosed by the curves are pretty much the same, which means that a unit volume of the slug receives the same energy in each case. But something else is quite different between the curves – the cross-sectional area of the core. The area of the core increases with the square of the radius. The 48-turn coil, for example, has a radius 2.260 times that of the basic 96-turn coil, so its area is 5.1 times greater. The cross-sectional area of a slug which the 48-turn coil could accommodate is 5.1 times greater than that the basic coil can accommodate. This may be something of interest. However, we are getting ahead of ourselves. The purpose of this part of the paper is to look at the spatial distribution of the force.

Let us return to our original train of though. The forces we have been talking about are forces acting per unit volume. The next logical step is to consider the whole volume of the slug. We will, of course, want to add up the forces acting on each little bit of volume of the slug in order to obtain the total force. We will do this in two steps. We will first divide the slug up into a number of thin slices, each being a disk. We will calculate the force acting on each thin disk and then add up the contributions from all of the disks which make up a slug. This is shown in the following figure.



A number of thin disks, each with thickness ΔH_{slug}, are placed face-to-face to build up the slug. Each thin disk can be considered to be built up from a series of annuli, each with a radial thickness of ΔR_{slug}. A typical annulus is shaded in gray in the figure. We will imagine that the thicknesses ΔH_{slug} and ΔR_{slug} have been chosen to be small enough that the force acting per unit volume can be considered to be a constant throughout the volume of each little annulus. The force per unit volume will be different for adjacent annuli, both radially and axially, but will be constant inside each annulus.

The volume of each annulus is equal to the product of the thickness of the annulus and its circular area. The easy way to calculate its circular area is to take the area of its "outer" circle and subtract the area of its "inner" circle, which is to say:

$$\begin{aligned} \text{Volume} &= \Delta H_{slug}(\text{outer area} - \text{inner area}) \\ &= \Delta H_{slug}(\pi \times \text{outer radius}^2 - \pi \times \text{inner radius}^2) \end{aligned} \quad (7)$$

This formulation applies even to the innermost annulus. Its inner radius is zero, so it degenerates into a circle. The inner radius of each annulus in the sequence is equal to the outer radius of the annulus which lies just inside it. The outermost annulus is the one whose outer radius is equal to the radius of the slug.

The force acting on each annulus is its Volume multiplied by the force per unit volume at this axial distance and radial distance from the $\hat{z}$-axis. In our numerical procedure, we take the force per unit volume acting on an annulus as being the force per unit volume along its geometric centerline, which is the circle whose circumference coincides with the middle of its $\Delta H_{slug} \times \Delta R_{slug}$ cross-section.
It is a simple matter to add up the forces acting on all the annuli which make up one of the thin disks. With a little more work, one can then add up the forces acting on all the thin disks which make up the slug. What comes out of the calculation is the net force acting on the slug. There are three points to note.

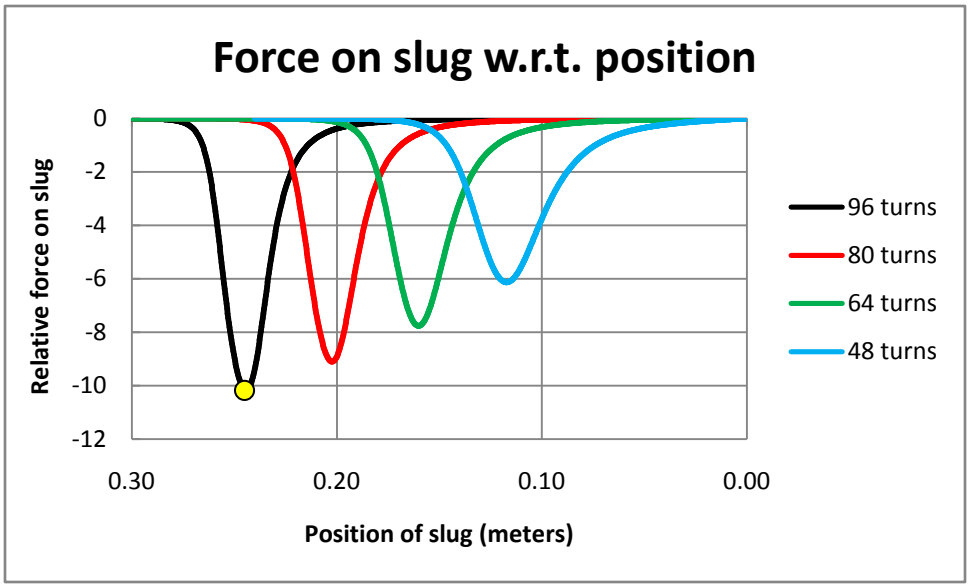
1. Remember that the forces per unit volume were computed without the factor $(\mu_r^2/\mu_0)(\mu_0 I\ /4\pi)^2$, which will need to be brought back to convert values into physical units of force, such as Newtons.

2. The net force acting on the slug is the (vector) sum of all the forces acting on the slug. As we saw above, some of the forces acting at a given point in the slug are balanced out by equal but opposite forces acting on other points in the slug. These are internal forces. The internal forces which cancel each other out do not contribute to the net force which would cause the slug to move as a rigid body. But they still exist. They tend to pull the slug apart. Indeed, in the presence of a strong enough external magnetic field, the slug will explode.

3. Let us choose physical dimensions for the slug. Wikipedia tells me that the 9 x 19mm Parabellum cartridge is a common standard. It was designed in 1902 by George Luger, of German pistol fame. There are many varieties, but "bullet weights ranging from 115 to 147 grains (7.5 to 9.5 grams) are common". There are 454 grams, and 16 ounces, in one pound. Therefore, a 9.5 gram cartridge weighs almost exactly one-third of an ounce. Our slug will differ from the 9 x 19mm Parabellum cartridge in two respects: it will be a cylinder, without the aerodynamic shape of the cartridge, and it will be made from steel. Wikipedia also tells me that steel has a mass "density of mild steel is approximately 7.85 grams per cubic centimeter". Mild steel makes good magnets. If we make our slug from this kind of steel, then a 10 gram piece must have a volume of 10 g / 7.85 g/cm^3 = 1.27 cm^3. A cylinder 9 mm in diameter and 20 mm long has a volume very close to this. Let us use these dimensions for a "basic" slug.

- $R_{slug} = 4.5$ mm
- $H_{slug} = 20$ mm
- $m = 10$ grams

Let us use the same slug in all four coils described above. When we add up the force per unit volume over the volume of the slug, we get the results shown in the graph to the right.



Now, for the first time, we can calculate the force acting on a real slug. The vertical axis in the graph above does not include the factor $(\mu_r^2/\mu_0)(\mu_0 I\ /4\pi)^2$. Let us now include this factor. For the point highlighted in yellow in the graph above, which is the position where the force field is a maximum for the basic 96-turn coil, the plotted value of -10.2 corresponds to a real value of:

$$\begin{aligned} F_{slug} &= -10.2\frac{\mu_r^2\mu_0}{16\pi^2}I^2 \\ &= -10.2\frac{200^2\times 4\pi\times 10^{-7}}{16\pi^2}20^2 \\ &= -1.30 \text{ Newtons} \end{aligned} \qquad (8)$$

In making the calculation, I have assumed that:

- the steel has a relative permeability of 200 and
- the current is 20 Amperes.

As always, the minus sign indicates that the force is directed towards the negative $\hat{z}$-axis. Since one Newton of force is about the weight of one-quarter pound, the force on the slug is about one-third of a pound.

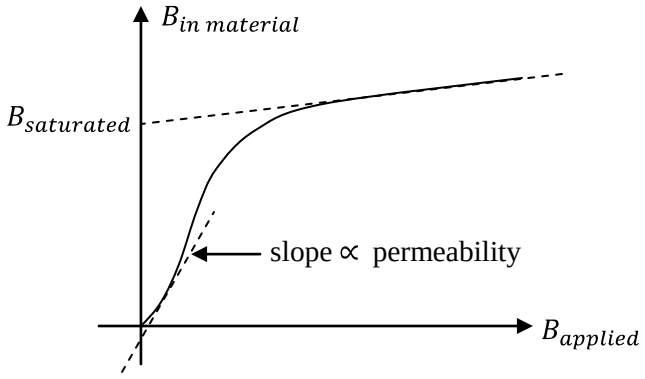
The relative permeability of steel

Up until this point, we have been assuming that the relative permeability μ_r of the slug, whatever it is made of, is some physical constant. Unfortunately, that is not so.

Like any ferromagnetic material, steel responds to an externally applied magnetic field in such a way as to increase it. The motions and spins of the electrons in the steel respond to the external field, and their altered motions and spins actually increase the strength, but do not change the direction, of the applied magnetic field. The effectiveness of any material in increasing an applied magnetic field is measured by its relative permeability, which is represented by the symbol μ_r. The permeability of a material is usually expressed as the product $\mu = \mu_r\mu_0$, where $\mu_0 = 4\pi\times 10^{-7}\text{H/m}$ is the permeability of free space.

Wikipedia states that quenched 0.9% carbon steel has a relative permeability of 100. "Electrical steel" is said to have a relative permeability of 4000. Annealed 99.8% pure iron is said to have a relative permeability of 5000. Clearly, the various materials from which we can make the slug have a wide range of relative permeability.

All of these reported relative permeabilities are a little bit misleading. It would be best to think of them as the multiplicative factor only when the applied magnetic field is quite weak. All materials have a finite ability to respond to an applied magnetic field. There comes a time when the electrons in the material are exhibiting the greatest "motions and spins" that they can. The electrons are simply unable to do more, even if the applied field strength is increased. When the electrons are doing all they can do, the material is said to be saturated. This is not a phenomenon which occurs at a specific strength of applied field. Rather, it is a condition which becomes more and more important as the applied field gets stronger and stronger. This is usually shown graphically as follows.



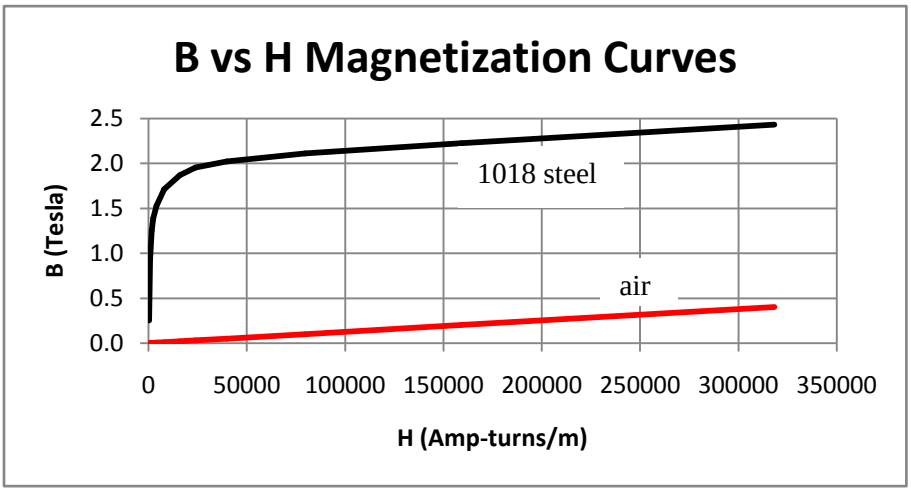
The strength of the total magnetic field inside the material ($B_{in\ material}$) gets less and less responsive to increases in the strength of the applied magnetic field ($B_{applied}$). The permeability of the material is proportional to the slope of the curve, and varies as the strength of the applied magnetic field varies. (There are two definitions of "slope": the instantaneous slope at any point on the curve or the average slope of a line drawn from the origin to the point.) The slope of the curve decreases as the applied field strength increases, until such time as the material cannot respond further. At this field strength, the material is said to be "saturated".

Since the total magnetic field inside the material is the sum of the external field and the extra field due to the material's response, the curve becomes linear at high magnetic field strengths. The extra field due to the material's response becomes a constant, being $B_{saturated}$, but the total field continues to increase due to increases in the applied field.

If a single relative permeability is given for a material, it is usually the slope at the origin of this kind of curve, where the applied field is very weak. For this reason, that value is called the "initial" relative permeability of the material.

Even when the applied field is weak, the response of the material is not necessarily linear. Often, the curve exhibits a bit of an "S" near zero. This is usually how a material behaves when it starts from a completely de-magnetizied state. It therefore depends on the history of the particular sample's exposure to magnetic fields. All materials retain a small amount of residual magnetization even after an applied field has been removed. (It can be a big amount of residual magnetization if the sample is intended to be a "permanent" magnet.) In any event, the residual magnetization will prevent the material from returning to the origin of the curve after one has played with it.

The graph at the right was prepared using data from the web site www.eng-tips.com, which gives data from the "FEA magnetic program for1018 low carbon alloy".

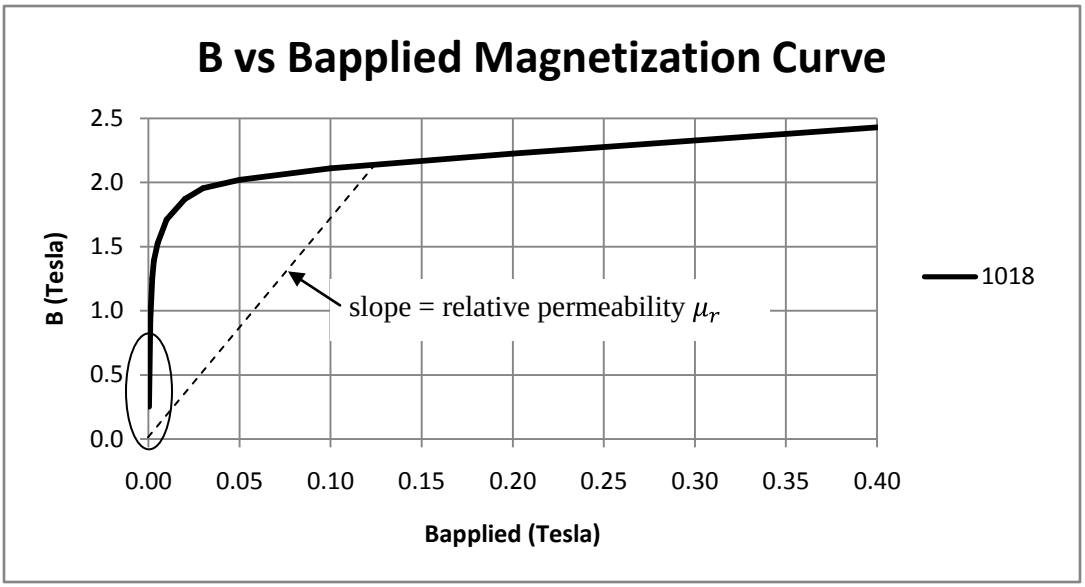


Let me describe the material whose data is shown. From among many grades of steel, I selected 1018 steel because it is so common. If you have a steel retailer near you, they will sell rods and bars made from 1018 steel.

Now, the curve itself is called a "magnetization curve". The vertical axis shows the total magnetic flux density B, measured in Tesla, which is set up inside the material when it is subjected to a magnetomotive force, or "coercive" force, H. This relationship between magnetic quantities corresponds directly to the following relationship between electrostatic quantities: the total electric flux density V, measured in Volts, which is set up inside a material when it is subjected to an electromotive force, or electrostatic field, E. Just as E represents a change in a field strength with respect to distance, and is measured in Volts per meter, so the coercive force represents a change in a field strength with respect to distance, and is measured in Ampere-turns per meter.

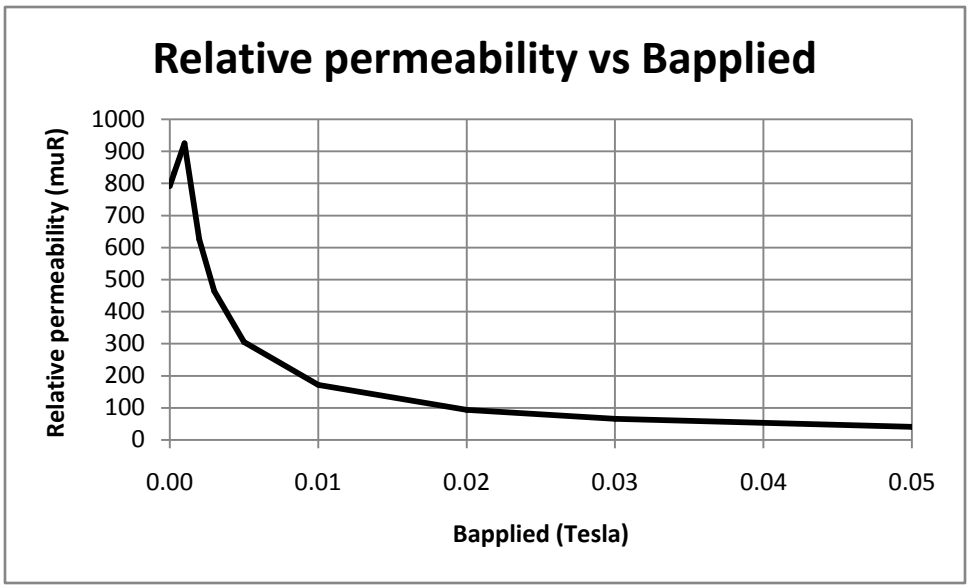
Now, Ampere-turns per meter is not a very convenient unit in which to think. I prefer to think in terms of the strength of the applied flux density, which is usually represented as B_0 or $B_{applied}$. It is the magnetic flux density which a given coercive force would set up in free space or in air, which has magnetic properties very close to those of free space. I have plotted the magnetization curve for air in red in the graph above. For free space or air, the relationship is simple: $B_0 = \mu_0 H$. So, the magnetization curve for air is a straight line with slope $\mu_0 = 4\pi \times 10^{-7}$H/m, where the H in these units is Henries. For example, a coercive force of 300,000 Ampere-turns/meter generates a magnetic flux density in air of about 0.4 Tesla.

In fact, it makes sense to re-state the horizontal axis of the graph above so that reads in terms of the applied magnetic flux density B_0 instead of the coercive force. The graph then looks like this:



Because both axes of this graph are magnetic flux densities, the ratio of the ordinate to the abscissa – which is the average slope to a point on the curve – is the relative permeability for the applied magnetic flux density which is the abscissa. The data from which this curve was prepared look as if it was intended for use with relatively strong fields. Where the data seems to be limited is for relatively weak fields. I have circled this region with an ellipse in the graph above.

To look into the behavior of 1018 steel under the influence of a weak applied field, I used data from another web site, www.fieldp.com. Their data is shown in the following graph.



Note that the entire horizontal axis for this graph is only one major division in the previous graph. As one takes this graph further out to the right, to applied field strengths of ten Tesla or more, the relative permeability approaches 1. (Remember that, for high field strengths, the total field strength in the material can be written as $B_{total} = B_{applied} + B_{sat}$, so that $\mu_r = 1 + \left(B_{sat}/B_{applied}\right) \rightarrow 1$.) Also note the small bump in this graph, which places the peak relative permeability at about 0.001 Tesla. This is evidence of the "S" behavior I described above.

Before we proceed much further, it is useful to take a moment to see what strengths of applied magnetic flux densities our basic 96-turn coil generates. Equation (2) above was an expression for the strength of the magnetic field at the center of a face of the coil, expressed as a function of the current, and in the absence of a slug. This is the spot where the force exerted on the slug would be greatest. The following table shows the applied field strength at that spot for currents of 10, 20 and 100 Amperes.

Current I	10 Amperes	20 Amperes	100 Amperes
Applied strength $B_z\|_F$	0.0011 Tesla	0.0022 Tesla	0.011 Tesla

If we add multiple layers to the coil, the applied field strength will increase in proportion. To pick a rough range, it would seem that we are going to be interested in applied field strengths up to, say, 0.05 Tesla. This is not particularly strong, and is about 1000 times the Earth's magnetic field.

When we execute the numerical integration to accelerate the slug through a run, we will need to estimate its relative permeability under a wide range of conditions. We will follow the following procedure.

Step 1: For any given coil, and any given current, we can estimate the strength of the applied magnetic flux density at the center of the coil's face using Equations (3) and (4), as:

$$B_{face\ center} \cong \frac{\mu_0 I N_{total}}{2H_{coil}} \quad (9)$$

Step 2: For any given coil, and prior to beginning the run, we will have stored in the array Ftable(*,*) the "force" acting on the slug at various axial displacements along the axis of the coil. As described above, these values of "force" will not include the factor $\mu_r^2 \mu_0 I^2/16\pi^2$. The stored values are

the relative spatial distribution of the force. The maximum of these values will occur very near to, but just inside, the face of the coil. As a proxy for the maximum, we will look up the value at the face of the coil. For the purposes of describing these steps, let us say that:

$$\mathcal{F}_{max} = \text{Ftable}|_{\text{face center}} \quad (10)$$

Step 3: The graphs above show that the spatial distribution of the magnetic flux density along the axis of the coil is very close to the spatial distribution of the force. We will, therefore, estimate the strength of the magnetic field at any axial position using the spatial distribution of the force as a proxy. If the slug at any time is located at any axial position of z_{slug}, then we will look up in Ftable(*,*) the value of the spatial distribution at that point (called $\mathcal{F}_{z_{slug}}$), and then estimate the magnetic flux density at that point as:

$$\begin{aligned} B_{z_{slug}} &\approx \frac{\mathcal{F}_{z_{slug}}}{\mathcal{F}_{max}} \times B_{face\ center} \\ &= \frac{\mathcal{F}_{z_{slug}}}{\mathcal{F}_{max}} \frac{\mu_0 N_{total}}{2H_{coil}} \times I \end{aligned} \quad (11)$$

Step 4: The value $B_{z_{slug}}$ is our estimate of the applied magnetic flux density. We can look up in the graph above the relative permeability of the steel at this applied field. In practice, we will interpolate the relative permeability from the following table. These are some of the values from the www.fieldp.com web site.

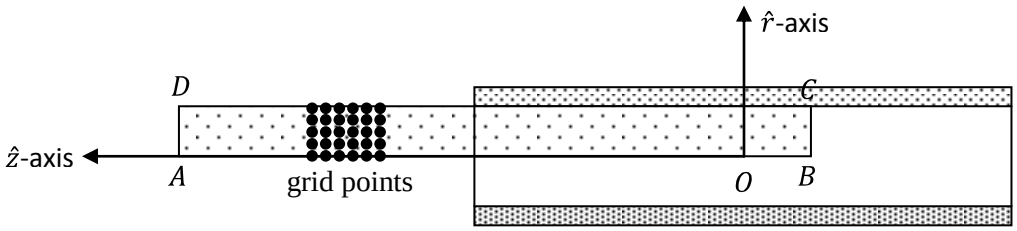
Applied flux density $B_{z_{slug}}$	Relative permeability μ_r
0.0 Tesla	790.6271
0.0003 Tesla	833.4465
0.001 Tesla	924.9707
0.002 Tesla	624.9802
0.003 Tesla	463.3380
0.005 Tesla	304.9980
0.01 Tesla	170.9989
0.02 Tesla	93.5000
0.03 Tesla	65.1665
0.05 Tesla	40.3999

Step 5: We will now make a rather heroic approximation. We will assume that the relative permeability which we obtain from interpolating the table applies throughout the volume of the slug. This is tantamount to assuming that the applied magnetic field strength is uniform throughout the slug. It is not, but a more realistic assumption would be very difficult to implement. In any event, we will use this relative permeability for the whole of the slug when it is at position z_{slug} and the current is as specified by the value substituted into Equation (11).

Program storage of the spatial distribution of the force

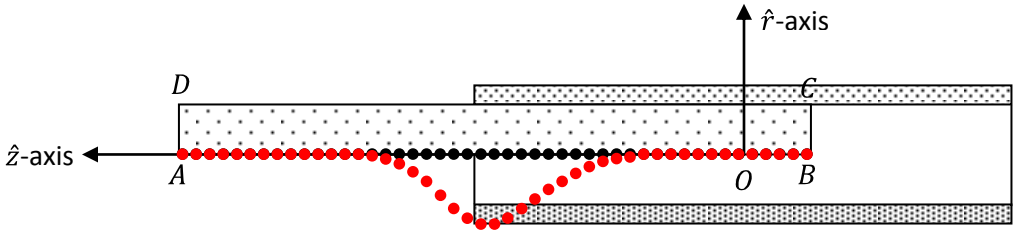
The quantitative results graphed above were produced using one of the numerical procedures in *Integration6*, a Visual Basic program whose listing is set out in Appendix "A" attached hereto. This program contains three modules, the second one of which – *Module2* – deals with the spatial distribution of the force field. The procedure in *Module2* is a cosmetic modification of the one used in the paper titled *The force on a cylindrical steel slug inside a finite solenoid*. This procedure carries out a number of

calculations in a rectangular region such as the rectangle $ABCD$ shown in the following figure. This rectangle is the region through which the slug will travel during the course of a run. Actually, the rectangle is only one-half of that – because of the symmetry around the $\hat{z}$-axis, calculations need only be done for $r \geq 0$. The region should extend from left to right a sufficient distance to accommodate the slug (not just its center, but including its horizontal extent as well) during all of the runs one plans to carry out. The region $ABCD$ need not extend all the way to the core radius of the coil – it only needs to extend to a radius large enough to accommodate the slug or slugs contemplated. However, the region can extend all the way to the core radius itself. The procedure takes into account the thickness of the wire in the winding, so the magnetic field exactly at the core radius will be large, but it will not be infinite.



The procedure sets up a rectangular grid of points throughout the region. A few sample grid points are shown in the figure. The horizontal spacing of the columns of points need not be equal to the vertical spacing of the rows but, for convenience, the grid must be rectangular. There would typically be several hundred grid points in both directions.

At every grid point, *Integration6* calculates the two components of the magnetic field strength – B_r and B_z – and the four spatial derivatives – $\partial B_r/\partial r$, $\partial B_r/\partial z$, $\partial B_z/\partial r$ and $\partial B_z/\partial z$. It is then a simple matter of multiplication of the appropriate pairs to calculate the force per unit volume at each grid point. After having done that, *Integration6* then calculates the force acting on a thin disk at each value of z where there is a column of grid points. To integrate the force on each annulus of a thin disk, it works its way up the column of grid points. When it needs a force per unit volume at a radius which is not a grid point, *Integration6* interpolates between the two adjacent grid points in the column. By the time it has finished this step, *Integration6* has calculated the force on a thin disk at each horizontal position in the grid. *Integration6* then proceeds to calculate the force on the slug at each horizontal position. To do this, the procedure interpolates horizontally to find the force on a thin disk at horizontal positions which are not grid points. When it has finished this step, *Integration6* has calculated the force on the slug at each horizontal position in the grid. A typical result is shown in the following figure.



The magnitude of the force (the red dots) are known at a number of horizontal positions (the black dots). This representation of the spatial distribution of the force is in exactly the same form as it was in the previous programs *Integration1* through *Integration4* in this paper, as a two-dimensional array Ftable(*,*). In other words, we can take the spatial distribution produced by *Integration6* and use it without change in the earlier programs.

The time-consuming work of *Integration6* (and it can be very time-consuming) is the calculation of the field strength and spatial derivatives. For this reason, *Integration6* saves the results of the basic field strength calculations in a text file *FieldTextFile.txt*. When the procedure moves on to the next steps in the

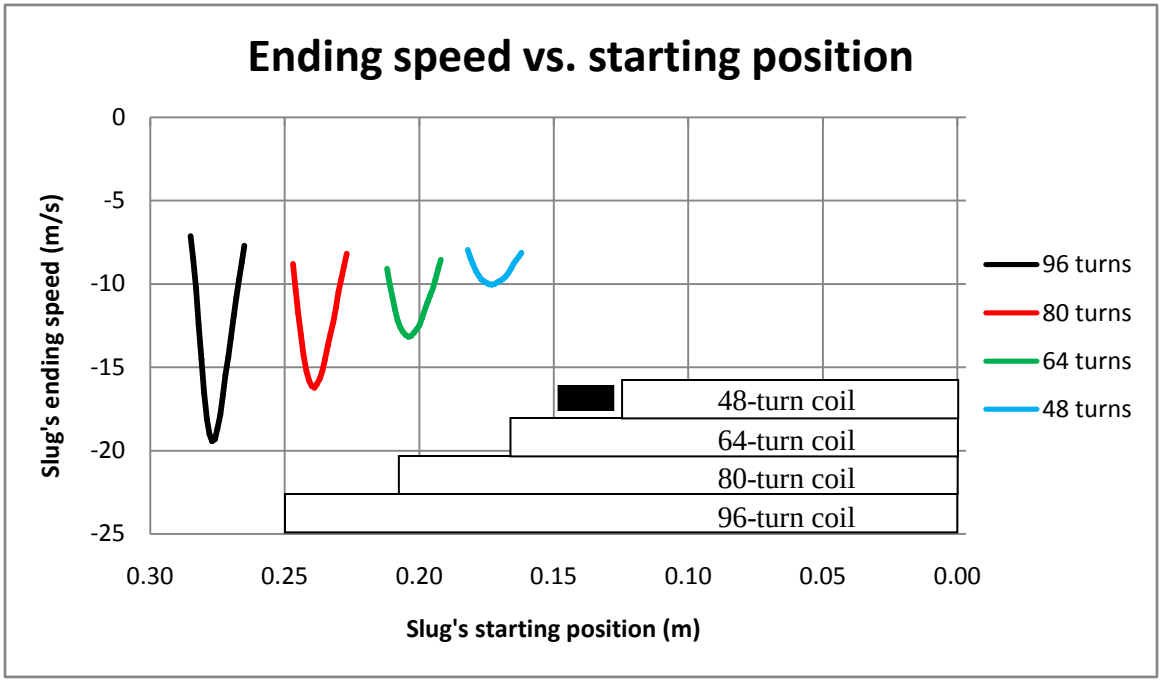
calculations, it retrieves the raw data from this text file. For convenience in plotting, the procedure can also produce Excel files listing the forces on the thin disks and the forces on the slug.

Our first real runs

We are now in a position to bring together everything we need to simulate the acceleration of a real slug out through a real coil. We will deal here with only four cases. We will use the same slug as before, which is 9 mm in diameter and 20 mm long, and fire it through each of the four coils we described above. The four coils have 96, 80, 64 and 48 turns, respectively, each of which is a single layer. Remember that we used the same length of wire to wind each coil. Therefore, the shorter the coil, the greater is diameter. The Ohmic resistance of each coil will be the same, but the inductance will not. The following table sets out the inductances of the four coils, calculated in accordance with Equations (1) and (44) of Part II of this paper.

N_{turns}	$R_{core}\ (m)$	$H_{coil}\ (m)$	$Resistance\ (\Omega)$	$Inductance\ (H)$
96	0.01000	0.4981	0.01566	11.6μ
80	0.01252	0.4151	0.01566	13.9μ
64	0.01630	0.3321	0.01566	17.4μ
48	0.02260	0.2491	0.01566	23.2μ

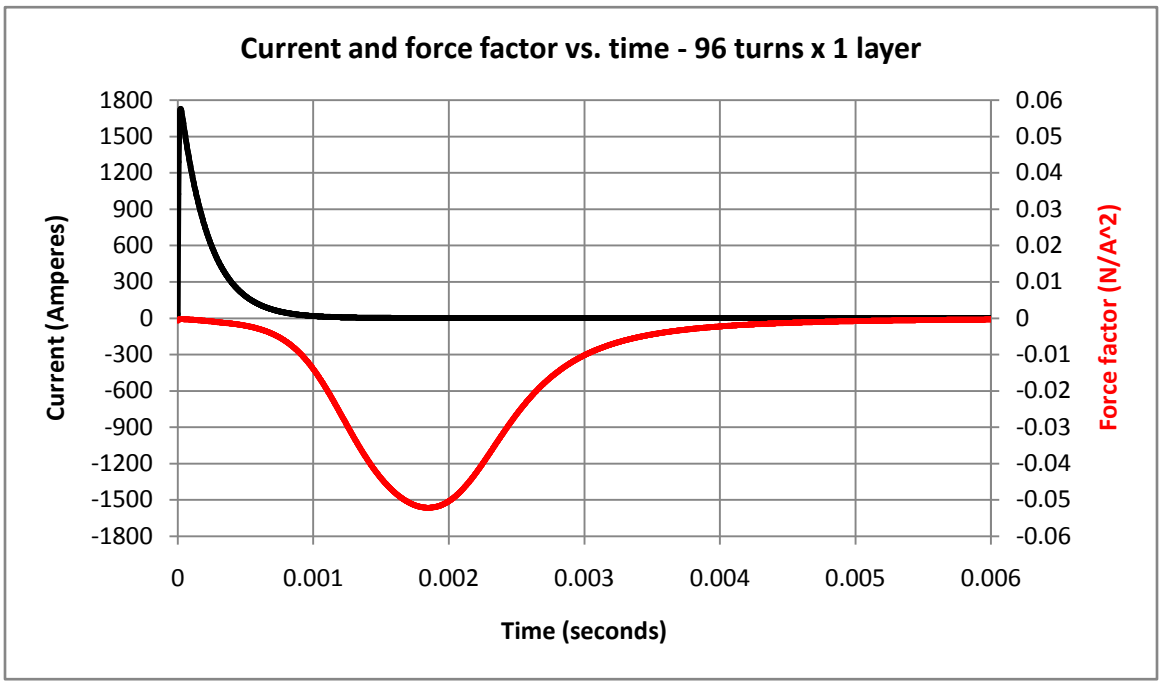
The following graph shows the results for the four coils. The values plotted on the vertical axis are the speeds of the slug at the end of the runs. As always, the speeds are negative since the slug is accelerated towards the right. The horizontal axis shows the starting position of the slug.



To give some idea for the physical scale, I have placed in the lower right a scale outline of the active side of the four coils and the slug. It can be seen that the best launching position for the slug is between three and five centimeters outside the face of each coil. The launching position is very important. The segment of the curve for each coil is two centimeters from left to right. In the case of the 96-turn coil, the curve is shown in black. The best ending speed of the slug is just less than 20 meters per second. Starting the slug one centimeter to either side of the best position cuts the ending speed by more than half.

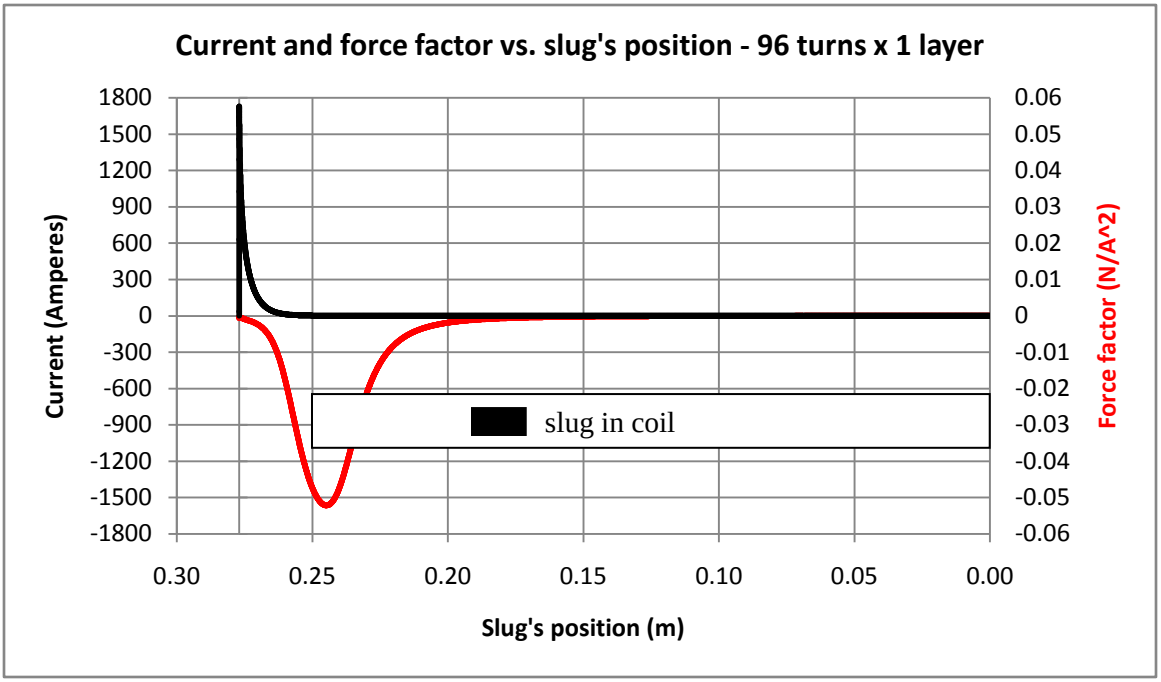
These results are, to pick a word, dreadful. At a speed of 20 meters per second, the 10 gram slug has kinetic energy of two Joules. This is only one-quarter of one percent of the 800 Joules of energy initially stored in the capacitor. As things stand, these coils are virtually useless as coil guns.

The following figure shows what is going wrong. This graph applies to the best run with the 96-turn coil, where the slug starts at a position of 27.7 cm from the center of the coil. The black curve shows the current in the circuit as a function of time. The red curve is the spatial force factor $\mathcal{F}$, including the coefficient factor $\mu_r^2 \mu_0 I^2 / 16\pi^2$. The force acting on the slug at any time is equal to the spatial force factor multiplied by the square of the current. The spatial force factor is negative because the force pulls the slug in the direction of the negative $\hat{z}$-axis.



The problem is the current spike. It is not just that a current of 1800 Amperes is far beyond the capacity of the wire from which the coil is wound. The real problem is that the resistor burns off almost all of the energy in the capacitor. Remember that the power consumed by the resistor is proportional to the square of the current and, with such a large current, the resistor takes all of the power. In addition, the spike in the current occurs far too early – the energy imparted to the slug would be greater if the maximum current occurred closer to the point where the spatial force factor is greatest.

The graph above shows the progress of the run as a function of time. It is useful to re-plot the same values showing them as functions of the slug's position. This is done in the following graph.



The horizontal axis in this graph is 30 cm long. As before, I have placed on the right a scale outline of the active side of the 96-turn coil and a scale outline of the slug. The spatial force factor (in red) has the size and shape expected – approximately centered on the face of the coil, but a little inside. But, the current flowed for only about one-half millisecond. Furthermore, the current flowed where the spatial force factor had little magnitude.

This is important. Firstly, the force acting on the slug is equal to the spatial force factor multiplied by the square of the current. Secondly, the energy imparted to the slug is equal to the force multiplied by the distance through which it acts. A much better result would obtain if the current peaked at the same time / location as the spatial force factor peaked.

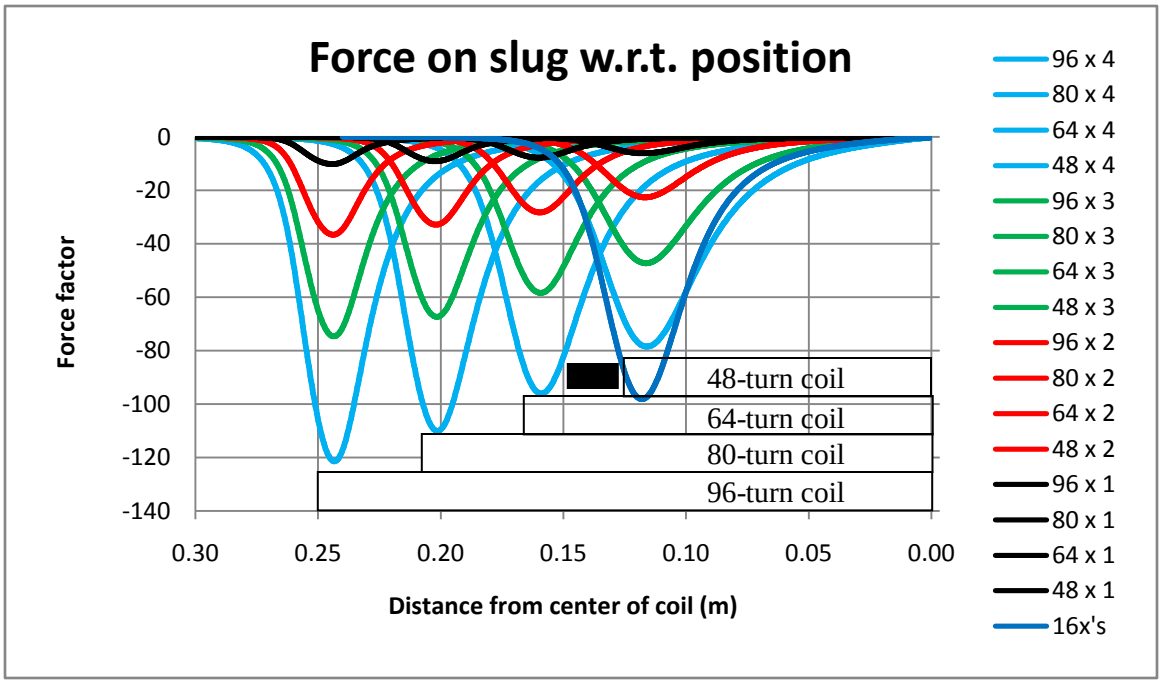
What is happening is that the components in the electrical circuit (that is, the magnitudes of the inductance, resistor and capacitance), on the one hand, and the mechanism by which the slug draws its energy from the magnetic field, on the other hand, are combining in such a way that they are not getting the maximum benefit from each other.

Adding layers to the coils

We have already seen that adding turns to the coil helps to deal with this problem. Adding turns to the coil increases its inductance more than its resistance, and the relatively greater inductance tends to "slow down" the current, both reducing its peak value and delaying the time at which the peak current occurs. Furthermore, adding turns to the coil increases the spatial force factor. And, in addition, the lower current means that less energy is wasted in the resistor. This is a win-win-win modification, up until the point where the exchange of energy between the inductance and capacitance continues even after the slug has departed from the force field.

The first step down this path is to look at the spatial force factor of the coils as we add additional layers of turns. This is shown in the following graph. This is similar to the graph set out above titled *Force on slug w.r.t. position*. In fact, the values for the four coils with only one layer was shown in that graph. In

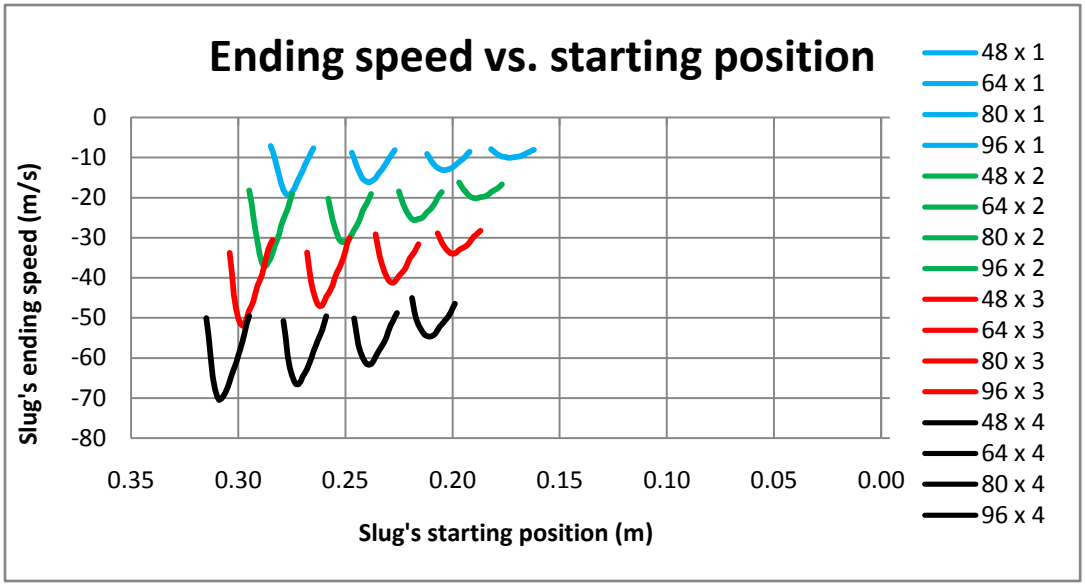
the following graph, the force on the slug is shown for the four coils, where each coil has up to four layers. The spatial force factor shown here does not include the coefficient $\mu_r^2\mu_0 I^2/16\pi^2$.



The original four coils, with one layer each, are shown in black. Coils with two layers are shown in red, coils with three layers are shown in green and coils with four layers are shown in light blue. For reference, I have placed in the lower right scale outlines of the four coils. In the ideal case, the force factor would increase with the square of the number of turns. For example, a coil with four layers should have a force factor $4^2 = 16$ times as great as the same coil with only one layer. That assumes, of course, that the wires are infinitely thin. When the wire has thickness, each layer lies further away from the core and its effect on the magnetic field inside the coil is less than the first layer's. In the graph above, I have shown in dark blue the curve which is 16 times the magnitude of the one-layer 48-turn coil. This ideal curve exceeds the calculated curve by about 20%.

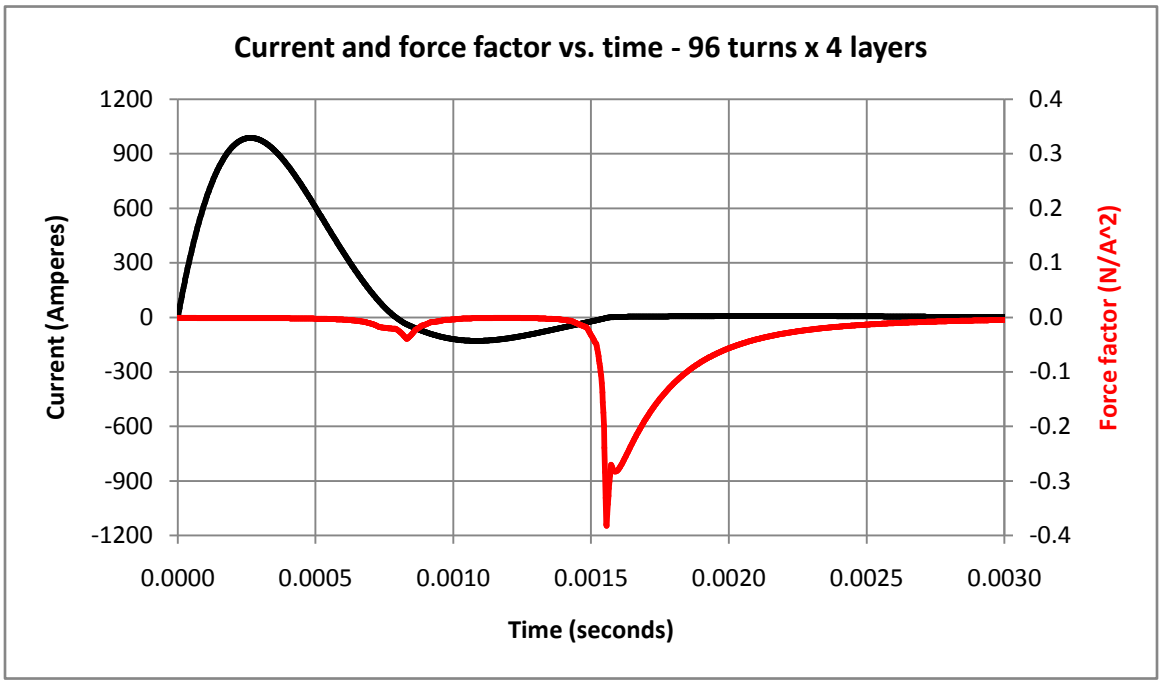
The graph at the right shows the ending speed of the slug using the sixteen coils, for different starting positions.

The four one-layer coils are shown in blue; the



four four-layer coils are shown in black. As expected, more layers result in higher ending speeds. It should be noted, too, that the optimal starting displacement gets further from the end of the coil as the number of layers increases. For example, the optimal starting displacement for the one-layer 96-turn coil is about 28 cm from the center of the coil. With four layers, the optimal starting displacement is about 31 cm from the center of the coil. As before, the ending speed of the slug is highly sensitive to changes in the starting displacement.

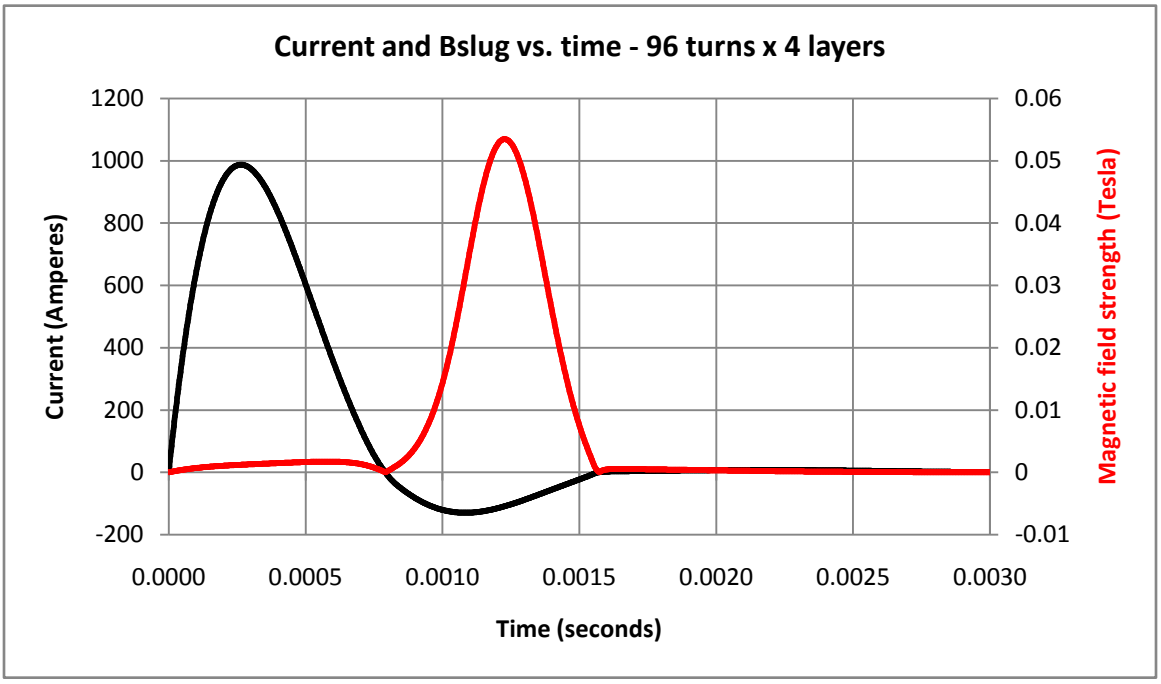
The following graphs show the details during the run through the four-layer 96-turn coil. The first graph shows the current and the force factor (that is, the spatial distribution of the force field) with respect to time.



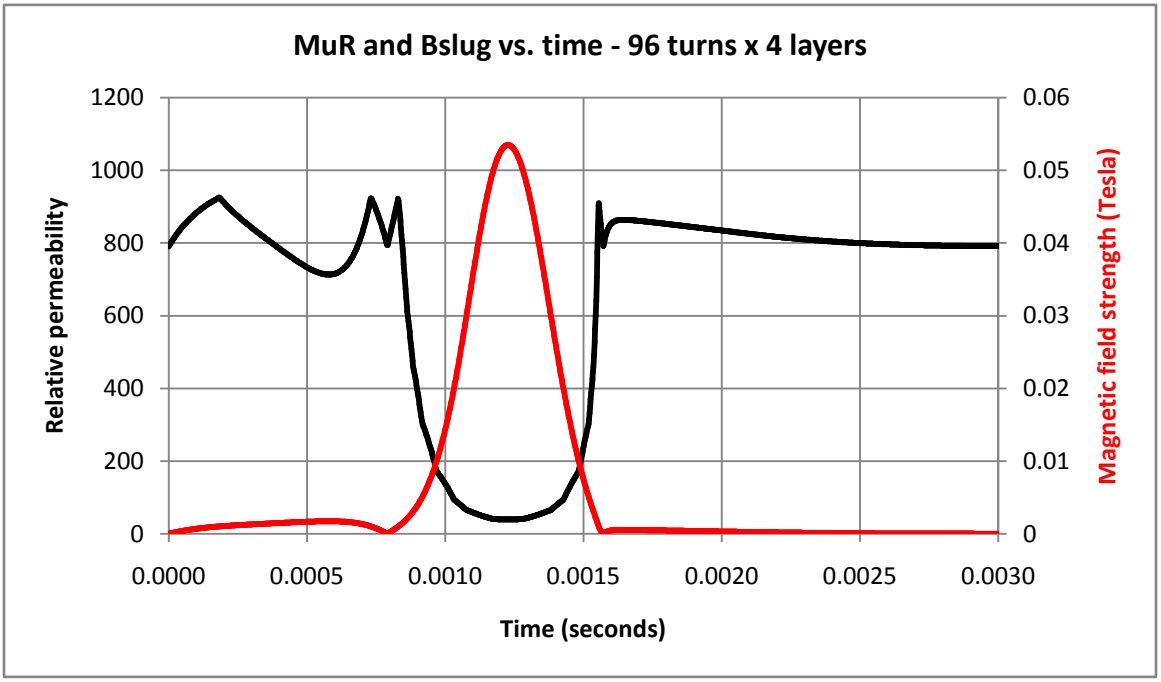
Let us consider the current waveform – the black curve – first. There are several important changes from the waveform with a single layer, and all of them can be attributed to the increased inductance from the additional layers. The increased inductance has changed the waveform from an R-C exponential decay, with its maximum value at time $t = 0$, to a "pulse" whose peak is noticeably delayed from the starting time. The peak current occurs about one-quarter millisecond into the run. Secondly, there is oscillation. The current becomes negative when the capacitor starts to accept charge from the inductor. Thirdly, the peak current is less than it was before.

Now, let us look at the force factor, whose waveform is entirely unexpected. Previously, the waveforms for the force factor peaked at the time the slug passes through (approximately) the face of the coil, and decreased smoothly on either side of that event. What we see here is a result of changes in the slug's relative permeability. Previously, we treated the force factor as depending on the slug's axial displacement only and, in particular, as not depending on the current. The total force acting on the slug was then the product of the spatial force factor and the square of the current. However, the spatial force factor depends on the relative permeability of the slug. Since the relative permeability of the slug depends on the strength of the magnetic field, there is an indirect dependence on the current. And, this indirect dependence seems to be very important.

The following graph shows the current, once again, and the magnetic field intensity at the center of the slug, both with respect to the time during the run.

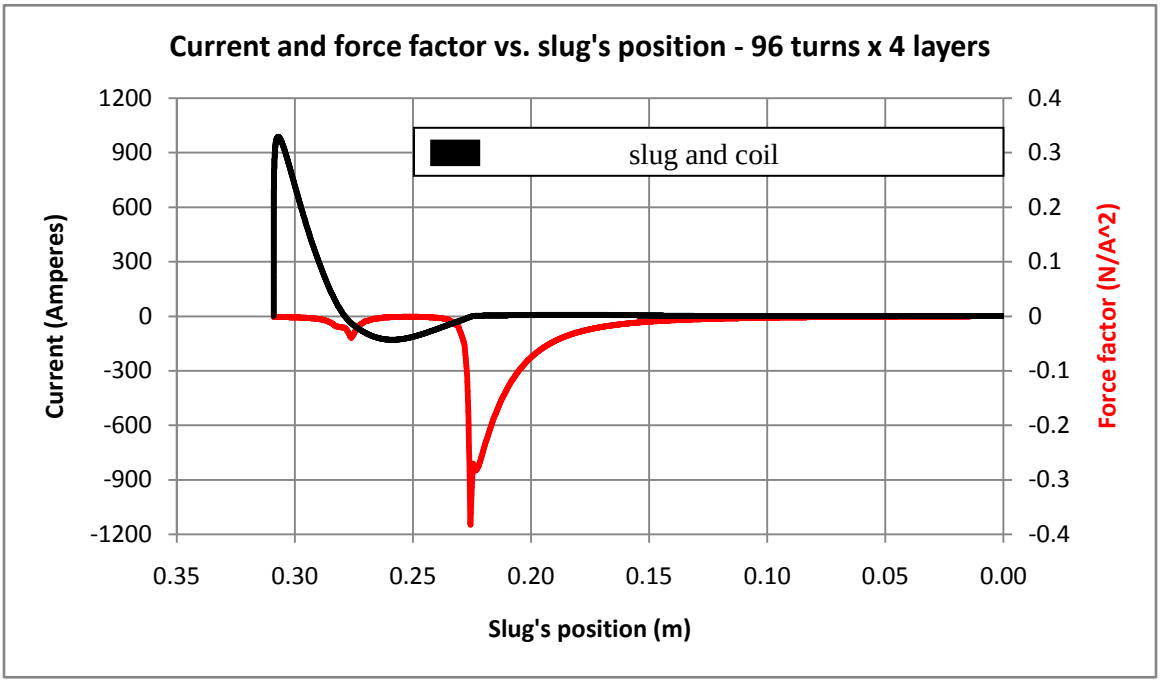


The strength of the magnetic field is shown in red. It peaks at about 0.05 Tesla at a time, interestingly, when the current is far below its maximum value. We normally think of the magnetic field strength as being proportional to the current, which is true, but it also depends on the slug's displacement. It happens that the slug is located near the face of the coil just as the current is passing through its second, smaller, pulse. That the current is negative during the second pulse does not change the fact that the force on the slug is attractive. The following graph takes us to the next step. It shows the magnetic field intensity, once again, and the relative permeability of the slug.

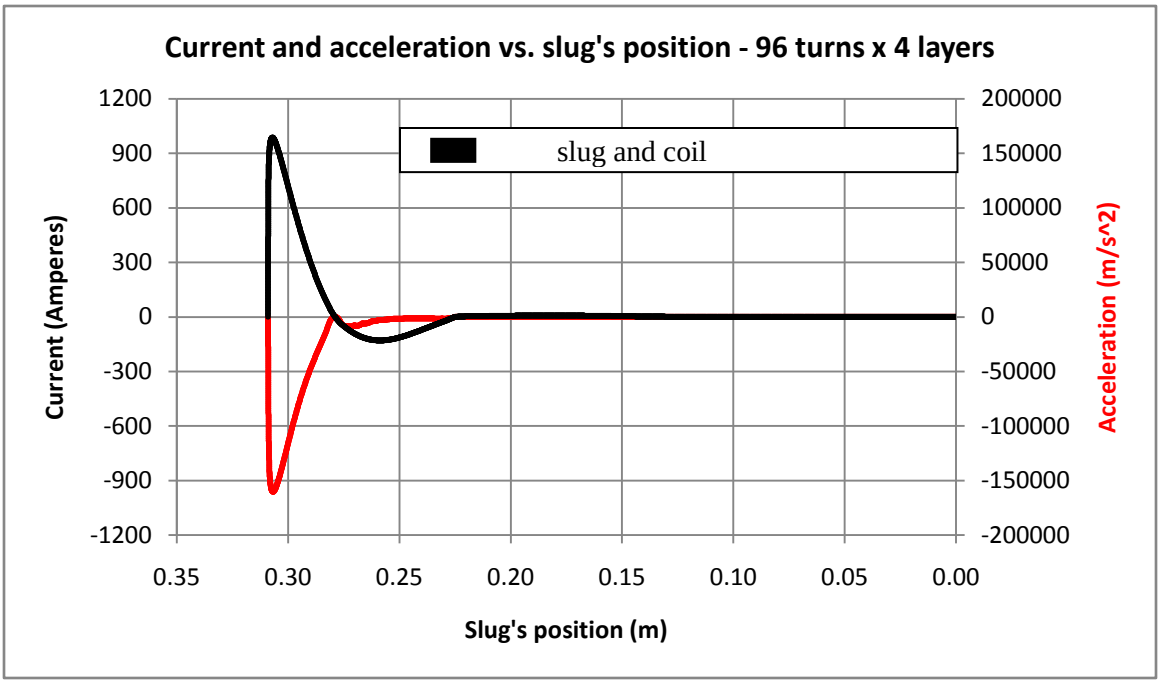


The slug's relatively permeability decreases by a factor of twenty when the magnetic field intensity spikes at 0.05 Tesla. It decreases to about $\mu_r = 40$ from its initial value of about $\mu_r = 800$. A lot of the benefit one would hope to gain from a higher current is lost due to decreasing relative permeability.

The following graph shows the current and the force factor during the run, this time plotted against the slug's position. For reference, I have once again shown a scale outline of the coil and slug.



Finally, the following graph shows the current and the slug's acceleration during the run, both plotted against the slug's position.



All of the slug's acceleration takes place outside the coil, and by a considerable distance.

Let me list the most important problems we have encountered:

1. The current is peaking too soon, before the slug reaches positions where the spatial force distribution is most favourable, near the face of the coil.
2. The current is still far too high. 1000 Amperes will destroy the coil.
3. The relative permeability of the slug is awfully low. It would be useful if we could find a material which saturates at higher values.

We will address some of these issues in Part IV of this paper.

Jim Hawley
September 2012

An e-mail setting out errors and omissions would be appreciated.

Appendix "A" - Listing of the Visual Basic program *Integration6*

Integration6 has three main modules and a control form. *Module1* integrates the circuit and dynamic equations to simulate a run. *Module2* calculates the spatial distribution of the force field. *Module3* calculates the relative permeability of the slug. *Form1* is the form which controls execution of the program and display on the monitor. The listing below shows three typical sub-programs which execute when the Start button is clicked.

Listing of *Module1*

```
Option Strict On
Option Explicit On

Module Module1

    ' Module 1 runs the integration for a single run.

    ' Output control
    Private delTdisplay As Double = 0.00001   ' Display results every 10 us
    Private Tlastdisplay As Double            ' The time of the last display event
    Private delTsave As Double = 0.000002     ' Save results every 2 us
    Private Tlastsave As Double               ' The time of the last save event

    Public Sub IntegrateOneRun( _
        ByVal SaveResults As Boolean, _
        ByRef objExcelWS As Microsoft.Office.Interop.Excel.Worksheet, _
        ByVal RowNum As Int32)
        ' If SaveResults is True, then the results are saved to the Excel
        ' file which is passed as the object.  In that case, argument RowNum
        ' specifies the row in the spreadsheet at which these interim results
        ' are to be saved.
        ' If SaveResults is False, then the integration is ended when the
        ' slug's position at the end of a time step is less 0, that is, when
        ' the slug has passed through the center of the coil.
        ' If SaveResults is True, then the integration is ended when the
        ' slug's position at the end of a time step is less than -Hcoil/2,
        ' that is, when the slug leaves the coil entirely.  This allows the
        ' output to show what happens after the slug is gone.
        '////////////////////////////////////////////////
        '// Write the header for the output, if requested.
        If (SaveResults = True) Then
            With objExcelWS
                .Cells(RowNum + 1, 1) = "Ending"
                .Cells(RowNum + 2, 1) = "Time"
                .Cells(RowNum + 3, 1) = "(s)"
                .Cells(RowNum, 2) = "Ending"
                .Cells(RowNum + 1, 2) = "Slug's"
                .Cells(RowNum + 2, 2) = "Position"
                .Cells(RowNum + 3, 2) = "(m)"
                .Cells(RowNum, 3) = "Ending"
                .Cells(RowNum + 1, 3) = "Slug's"
                .Cells(RowNum + 2, 3) = "Speed"
                .Cells(RowNum + 3, 3) = "(m/s)"
                .Cells(RowNum, 4) = "Starting"
                .Cells(RowNum + 1, 4) = "Slug's"
                .Cells(RowNum + 2, 4) = "Acceleration"
```

```
.Cells(RowNum + 3, 4) = "(m/s^2)"
.Cells(RowNum + 1, 5) = "Ending"
.Cells(RowNum + 2, 5) = "Charge"
.Cells(RowNum + 3, 5) = "(C)"
.Cells(RowNum, 6) = "Ending"
.Cells(RowNum + 1, 6) = "Total"
.Cells(RowNum + 2, 6) = "Current"
.Cells(RowNum + 3, 6) = "(A)"
.Cells(RowNum, 7) = "Spatial"
.Cells(RowNum + 1, 7) = "Force"
.Cells(RowNum + 2, 7) = "Factor"
.Cells(RowNum + 3, 7) = "(N/A^2)"
.Cells(RowNum, 8) = "Ending"
.Cells(RowNum + 1, 8) = "Capacitor's"
.Cells(RowNum + 2, 8) = "Energy"
.Cells(RowNum + 3, 8) = "(J)"
.Cells(RowNum, 9) = "Ending"
.Cells(RowNum + 1, 9) = "Inductor's"
.Cells(RowNum + 2, 9) = "Energy"
.Cells(RowNum + 3, 9) = "(J)"
.Cells(RowNum, 10) = "Ending"
.Cells(RowNum + 1, 10) = "Cumulative"
.Cells(RowNum + 2, 10) = "Heat"
.Cells(RowNum + 3, 10) = "(J)"
.Cells(RowNum, 11) = "Ending"
.Cells(RowNum + 1, 11) = "Slug's"
.Cells(RowNum + 2, 11) = "Energy"
.Cells(RowNum + 3, 11) = "(J)"
.Cells(RowNum, 12) = "Ending"
.Cells(RowNum + 1, 12) = "Total"
.Cells(RowNum + 2, 12) = "Energy"
.Cells(RowNum + 3, 12) = "(J)"
.Cells(RowNum, 13) = "Starting"
.Cells(RowNum + 1, 13) = "Capacitor's"
.Cells(RowNum + 2, 13) = "Voltage"
.Cells(RowNum + 3, 13) = "(C)"
.Cells(RowNum, 14) = "Starting"
.Cells(RowNum + 1, 14) = "Inductor's"
.Cells(RowNum + 2, 14) = "Voltage"
.Cells(RowNum + 3, 14) = "(V)"
.Cells(RowNum, 15) = "Starting"
.Cells(RowNum + 1, 15) = "Resistors's"
.Cells(RowNum + 2, 15) = "Voltage"
.Cells(RowNum + 3, 15) = "(V)"
.Cells(RowNum, 16) = "Starting"
.Cells(RowNum + 1, 16) = "Slug's"
.Cells(RowNum + 2, 16) = "Voltage"
.Cells(RowNum + 3, 16) = "(V)"
.Cells(RowNum, 17) = "Starting"
.Cells(RowNum + 1, 17) = "Voltage"
.Cells(RowNum + 2, 17) = "Error"
.Cells(RowNum + 3, 17) = "(V)"
.Cells(RowNum, 18) = "Midpoint"
.Cells(RowNum + 1, 18) = "Slug's"
.Cells(RowNum + 2, 18) = "Relative"
.Cells(RowNum + 3, 18) = "Permeability"
.Cells(RowNum + 2, 19) = "BFaceCenter"
.Cells(RowNum + 3, 19) = "(Tesla)"
```

```
        .Cells(RowNum + 2, 20) = "F_FaceCenter"
        .Cells(RowNum + 3, 20) = "(N/A^2)"
        .Cells(RowNum + 2, 21) = "F_Zslug"
        .Cells(RowNum + 3, 21) = "(N/A^2)"
        .Cells(RowNum + 2, 22) = "BZSlug"
        .Cells(RowNum + 3, 22) = "(Tesla)"
    End With
End If
'/////////////////////////////////////////////////
'// Write the starting data to the file, if requested.
If (SaveResults = True) Then
    RowNum = RowNum + 4
    With objExcelWS
        .Cells(RowNum, 1) = 0           ' Time
        .Cells(RowNum, 2) = Z0          ' Ending slug's position
        .Cells(RowNum, 3) = S0          ' Ending slug's speed
        .Cells(RowNum, 4) = 0           ' Starting slug's acceleration
        .Cells(RowNum, 5) = QC0         ' Ending charge
        .Cells(RowNum, 6) = 0           ' Ending current
        .Cells(RowNum, 7) = ""          ' Spatial force factor
        .Cells(RowNum, 8) = 0.5 * C * VC0 * VC0 ' Ending capacitor's energy
        .Cells(RowNum, 9) = 0           ' Ending inductor's energy
        .Cells(RowNum, 10) = 0          ' Ending cumulative heat
        .Cells(RowNum, 11) = 0          ' Ending slug's energy
        .Cells(RowNum, 12) = 0.5 * C * VC0 * VC0 ' Ending total energy
        .Cells(RowNum, 13) = VC0        ' Starting capacitor's voltage
        .Cells(RowNum, 14) = VC0        ' Starting inductor's voltage
        .Cells(RowNum, 15) = 0          ' Starting resistor's voltage
        .Cells(RowNum, 16) = 0          ' Starting slug's voltage
        .Cells(RowNum, 17) = 0          ' Starting voltage error
    End With
End If
'/////////////////////////////////////////
'// Set things up for the first iteration.
T = 0
Tlastdisplay = -1000 ' Dummy previous time for displaying results
Tlastsave = -1000    ' Dummy previous time for saving results
' For the slug:
Zstart = Z0
Sstart = S0
' For the energies:
QCstart = QC0
VCstart = VC0
ECstart = 0.5 * C * VC0 * VC0
ERstart = 0
ELstart = 0
ESstart = 0
ETstart = ECstart + ERstart + ELstart + ESstart
' For the current:
Istart = 0
' For the termination of the run:
If (SaveResults = True) Then
    Zfinished = -0.5 * Hcoil
Else
    Zfinished = 0
End If
'/////////////////////////////////
'// Main loop for the integration.
```

```
MaxCurrent = 0
Do
    ' Increment the time to the end of the next time step.
    T = T + delT
    ' Estimate the ending position of the slug - Equation (30).
    ZendEst = Zstart + (Sstart * delT)
    ' Find the spatial force factor at the beginning of the time step.
    F_start = InterpolateForceField(Zstart)
    ' Estimate the spatial force factor at the end of the time step.
    F_endEst = InterpolateForceField(ZendEst)
    ' Calculate the average force factor - Equation (31).
    F_avg = 0.5 * (F_start + F_endEst)
    ' Estimate the slug's relative permeability in the middle of the step.
    Dim BFaceCenter As Double
    Dim F_FaceCenter As Double
    Dim F_Zslug As Double
    Dim BZSlug As Double
    MuRmid = GetRelativePermeability( _
        0.5 * (Zstart + ZendEst), Istart, Nturns, Nlayers, Hcoil, _
        BFaceCenter, F_FaceCenter, F_Zslug, BZSlug)
    ' Convert the force to physcial units.
    F_avg = F_avg * MuRmid * MuRmid * Mu0 / (16 * Math.PI * Math.PI)
    ' Calculate the slug's acceleration at the start - Equation (28'Nlayers).
    Astart = Istart * Istart * F_avg / M
    ' Calculate the slug's speed at the end - Equation (29').
    Send = Sstart + (Astart * delT)
    ' Calculate the slug's position at the end - Equation (32).
    Zend = Zstart + (Sstart * delT) + (0.5 * Astart * delT * delT)
    ' Calculate the second derivative of the charge - Equation (27'Nlayers).
    ' Note the slug always consumes positive power.
    d2Qdt2start = (Istart * R) + (Istart * (F_avg * Sstart))
    d2Qdt2start = d2Qdt2start - (QCstart / C)
    d2Qdt2start = d2Qdt2start / L
    ' Calculate the current at the end - Equation (33).
    Iend = Istart - (d2Qdt2start * delT)
    If (Math.Abs(Iend) > MaxCurrent) Then
        MaxCurrent = Math.Abs(Iend)
    End If
    ' Calculate the charge at the end - Equation (34).
    QCend = QCstart - (Istart * delT) + (0.5 * d2Qdt2start * delT * delT)
    '//////////////////////
    '// Energy calculations
    ' Calculate the ending energy in the capacitor - Equation (35).
    ECend = 0.5 * QCend * QCend / C
    ' Calculate the heat dissipated during the time step - Equation (36).
    delER = (Istart * Istart) - (Istart * d2Qdt2start * delT) + _
        (d2Qdt2start * d2Qdt2start * delT * delT / 3)
    delER = R * delER * delT
    ' Calculate the cumulative heat at the end - Equation (37).
    ERend = ERstart + delER
    ' Calculate the ending energy in the inductor - Equation (38).
    ELend = 0.5 * L * Iend * Iend
    ' Calculate the ending energy of the slug - Equation (39).
    ESend = 0.5 * M * Send * Send
    ' Calculate the total ending energy.
    ETend = ECend + ERend + ELend + ESend
    '//////////////////////
    '// Voltage calculations
```

```
' Calculate the starting voltage over the capacitor.
VCstart = QCstart / C
' Calculate the starting voltage over the resistor.
VRstart = R * Istart
' Calculate the starting voltage over the inductor.
VLstart = -L * d2Qdt2start
' Calculate the starting voltage over the slug.
VSstart = Istart * Sstart * F_avg
' Calculate the starting voltage error.
ErrVstart = VCstart - (VRstart + VLstart + VSstart)
'/////////////////////////////////////////////////
'// If it is time, then save the data for this step.
If (SaveResults = True) Then
    If ((T - Tlastsave) >= delTsave) Then
        RowNum = RowNum + 1
        With objExcelWS
            .Cells(RowNum, 1) = T           ' Time
            .Cells(RowNum, 2) = Zend        ' Ending slug's position
            .Cells(RowNum, 3) = Send        ' Ending slug's speed
            .Cells(RowNum, 4) = Astart      ' Starting slug's acceleration
            .Cells(RowNum, 5) = QCend       ' Ending charge
            .Cells(RowNum, 6) = Iend        ' Ending current
            .Cells(RowNum, 7) = F_avg       ' Spatial force factor
            .Cells(RowNum, 8) = ECend       ' Ending capacitor's energy
            .Cells(RowNum, 9) = ELend       ' Ending inductor's energy
            .Cells(RowNum, 10) = ERend      ' Ending cumulative heat
            .Cells(RowNum, 11) = ESend      ' Ending slug's energy
            .Cells(RowNum, 12) = ETend      ' Ending total energy
            .Cells(RowNum, 13) = VCstart    ' Starting capacitor's voltage
            .Cells(RowNum, 14) = VLstart    ' Starting inductor's voltage
            .Cells(RowNum, 15) = VRstart    ' Starting resistor's voltage
            .Cells(RowNum, 16) = VSstart    ' Starting slug's voltage
            .Cells(RowNum, 17) = ErrVstart  ' Starting voltage error
            .Cells(RowNum, 18) = MuRmid     ' Midpoint relative permeability
            .Cells(RowNum, 19) = BFaceCenter   ' For calculating MuR
            .Cells(RowNum, 20) = F_FaceCenter  ' For calculating MuR
            .Cells(RowNum, 21) = F_Zslug       ' For calculating MuR
            .Cells(RowNum, 22) = BZSlug        ' For calculating MuR
        End With
        Tlastsave = T
    End If
End If
'/////////////////////////////////////////////////
'// Check if we should stop integrating.
'// Case A: Not saving results.
If (SaveResults = False) Then
    If (Zend <= Zfinished) Then
        Exit Do
    End If
Else
    '// Case B: Interim results are being saved.
    If (Zend <= (-0.5 * Hcoil)) Then
        RowNum = RowNum + 1
        With objExcelWS
            .Cells(RowNum, 1) = T           ' Time
            .Cells(RowNum, 2) = Zend        ' Ending slug's position
            .Cells(RowNum, 3) = Send        ' Ending slug's speed
            .Cells(RowNum, 4) = Astart      ' Starting slug's acceleration
```

```
            .Cells(RowNum, 5) = QCend       ' Ending charge
            .Cells(RowNum, 6) = Iend        ' Ending current
            .Cells(RowNum, 7) = F_avg       ' Spatial force factor
            .Cells(RowNum, 8) = ECend       ' Ending capacitor's energy
            .Cells(RowNum, 9) = ELend       ' Ending inductor's energy
            .Cells(RowNum, 10) = ERend      ' Ending cumulative heat
            .Cells(RowNum, 11) = ESend      ' Ending slug's energy
            .Cells(RowNum, 12) = ETend      ' Ending total energy
            .Cells(RowNum, 13) = VCstart    ' Starting capacitor's voltage
            .Cells(RowNum, 14) = VLstart    ' Starting inductor's voltage
            .Cells(RowNum, 15) = VRstart    ' Starting resistor's voltage
            .Cells(RowNum, 16) = VSstart    ' Starting slug's voltage
            .Cells(RowNum, 17) = ErrVstart  ' Starting voltage error
            .Cells(RowNum, 18) = MuRmid     ' Midpoint relative permeability
        End With
        Exit Do
    End If
End If
'////////////////////////////////////////////////////////
'// If it is time, then display interim results to the user.
If ((T - Tlastdisplay) >= delTdisplay) Then
    Form1.labelDisplay.Text = _
            "Number of turns = " & Str(Nturns) & vbCrLf & _
            "Numbers of layers = " & Str(Nlayers) & vbCrLf & _
            "Starting position = " & Str(Z0) & vbCrLf & _
            "Capacitance = " & Str(C) & vbCrLf & vbCrLf & _
            "Time = " & Str(T) & vbCrLf & _
            "ZendEst = " & Str(ZendEst) & vbCrLf & _
            "F_avg = " & Str(F_avg) & vbCrLf & _
            "Astart = " & Str(Astart) & vbCrLf & _
            "Send = " & Str(Send) & vbCrLf & _
            "Zend = " & Str(Zend) & vbCrLf & _
            "d2Qdt2start = " & Str(d2Qdt2start) & vbCrLf & _
            "Istart = " & Str(Istart) & vbCrLf & _
            "QCstart = " & Str(QCstart) & vbCrLf & _
            "Iend = " & Str(Iend) & vbCrLf & vbCrLf & _
            "ECend = " & Str(ECend) & vbCrLf & _
            "ERend = " & Str(ERend) & vbCrLf & _
            "ELend = " & Str(ELend) & vbCrLf & _
            "ESend = " & Str(ESend) & vbCrLf & _
            "ETend = " & Str(ETend) & vbCrLf & vbCrLf & _
            "VCstart = " & Str(VCstart) & vbCrLf & _
            "VRstart = " & Str(VRstart) & vbCrLf & _
            "VLstart = " & Str(VLstart) & vbCrLf & _
            "VSstart = " & Str(VSstart) & vbCrLf & _
            "ErrVstart = " & Str(ErrVstart) & vbCrLf & vbCrLf & _
            "MuRmid = " & Str(MuRmid)
    Form1.labelDisplay.Refresh()
    Tlastdisplay = T
End If
'////////////////////////////////
'// Update for the next iteration.
Zstart = Zend
Sstart = Send
Istart = Iend
QCstart = QCend
ECstart = ECend
ELstart = ELend
```

```
        ERstart = ERend
        ESstart = ESend
        ETstart = ETend
    Loop
End Sub

'////////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////////
'//
'//                                  Build the coil
'//
'////////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////////

Public Sub BuildTheCoil()
    ' This function takes the physical parameters of the coil and calculates
    ' its resistance and inductance.  All of the variables except the number
    ' of layers are global variables.
    ' Step #1: Calculate the resistance.
    LenWire = 0
    For I As Int32 = 1 To Nlayers Step 1
        LenWire = LenWire + _
            (Nturns * 2 * Math.PI * (Rcore + ((Nlayers - 0.5) * Dwire)))
    Next I
    Rcoil = RPerM * LenWire
    ' Step #2: Calculate the inductance.
    RADavg = Rcore + (0.5 * Nlayers * Dwire)
    AREAavg = Math.PI * RADavg * RADavg
    Lcoil = 4 * Math.PI * 0.0000001 * _
        (Nturns * Nlayers) * (Nturns * Nlayers) * AREAavg / Hcoil
End Sub

'////////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////////
'//
'//                       Interpolation in the force field table
'//
'////////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////////

Public Function InterpolateForceField(ByVal Z As Double) As Double
    ' This function returns the interpolated magnitude of the force field at
    ' for the abscissa passed as the argument.  The returned value will be
    ' positive unless the given abscissa lies outside the force field, in
    ' which case the function returns zero.
    Dim lIndexToLeft As Int32
    Dim lSlope As Double
    ' Check if the slug is positioned outside the force field, to the left.
    If (Z > Ftable(1, 1)) Then
        InterpolateForceField = 0
        Exit Function
    End If
    ' Check if the slug is positioned outside the force field, to the right.
    If (Z < Ftable(FnumZ, 1)) Then
        InterpolateForceField = 0
        Exit Function
    End If
    ' Find the index of the abscissa in the force field table which lies just
```

```
        ' to the left of the given abscissa.  Note that this routine does not
        ' assume that the points in the table are equally-spaced.  It assumes only
        ' that they are in order of decreasing values, that is, each point is
        ' closer to the origin than the preceding point.
        For I As Int32 = 1 To (FnumZ - 1) Step 1
            If ((Z <= Ftable(I, 1)) And (Z > Ftable(I + 1, 1))) Then
                lIndexToLeft = I
                Exit For
            End If
        Next I
        ' Let us check for the trivial case, where we are exactly at a data point.
        If (Z = Ftable(lIndexToLeft, 1)) Then
            ' By chance, we are exactly at a data point.
            InterpolateForceField = Ftable(lIndexToLeft, 2)
            Exit Function
        End If
        ' We deal here with the general case, where we are between two data points.
        Try
            lSlope = (Ftable(lIndexToLeft + 1, 2) - Ftable(lIndexToLeft, 2)) / _
                (Ftable(lIndexToLeft + 1, 1) - Ftable(lIndexToLeft, 1))
        Catch ex As Exception
            MsgBox("Error in the force field table.  Two abscissas are the same.")
            InterpolateForceField = 0
            Exit Function
        End Try
        InterpolateForceField = Ftable(lIndexToLeft, 2) + _
            (lSlope * (Z - Ftable(lIndexToLeft, 1)))
    End Function

End Module
```

Listing of *Module2*

```
Option Strict On
Option Explicit On

Module Module2

    '*************** Spatial distribution of the force field ***************

    ' Module 2 calculates the spatial distribution of the force of a slug.  This module
    ' allows the user to calculate the force from scratch or, if the user so directs, to
    ' retrieve the raw data from a text file.  There are two principal subroutines.
    '
    ' 1.  CalculateAndSaveFields(...) calculates the force field and saves it in
    '     the specified text file.
    '
    ' 2.  ReadAndCalculateSpatialDistribution(...) reades the specified text file
    '     and calculates ll field variables.
    '
    ' The second-order sibroutines are:
    '
    ' 1.  ReadFieldFile(...) reads the specified text file.
    ' 2.  CalculateForceComponents(...) calculates the three force components and
    '     the force per unit volume.
    ' 3.  CalculateForceOnThinDisk(...) does the thin disk integration.
    ' 4.  SaveForceOnThinDiskFileForPlotting(...) saves the Excel file for plotting
    '     the force on the thin disks.
    ' 5.  CalculateForceOnSlug(...) does the slug integration.
    ' 6.  SaveForceOnSlugFileForPlotting(...) saves the Excel file for plotting
    '     the force on a slug.
    '
    ' The public variables which are made available to Module 2 are:
    '
    ' Physical parameters:
    '   Rcore      Core radius in meters
    '   Dwire      Wire diameter in meters
    '   Nlayers    Number of layers in the winding
    '   Nturns     Number of turns in each layer of the winding
    '   Rslug      Radius of the slug in meters
    '   Hslug      Length of the slug in meters
    '
    ' Integration parameters:
    '   FnumZ      The number of horizontal points at which to calculate the force
    '   FleftZ     The left-most point at which to calculate the force
    '   FrightZ    The right-most point at which to calculate the force
    '   FnumR      The number of vertical points at which to calculate the force
    '   FinnerR    The inner-most radius at which to calculate the force
    '   FouterR    The outer-most radius at which to calculate the force
    '
    ' The spatial distribution calculated by Module 2:
    '   Ftable(1001, 2)   (*,1) is an axial position
    '                     (*,2) is the magnitude of the force
    '                     (0,*) is the index which corresponds to FleftZ
    '                     (FnumZ,*) is the index which corresponds to FrightZ
    '
    ' Note that:
    ' 1. The values do not include the factor MuR^2 * Mu0 * Current^2 / 4Pi
```

```
' 2. The force is returned algebraically positive when attractive.
' 3. The routine places the forces on the slug in private variable
'    ForceOnSLug(FnumZ).  Before returning, the appropriate values are
'    copied into the public variable Ftable(FnumZ, 2).  The values are
'    negated so that Ftable(,) has algebraically necgative values.
'
'////////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////////
'//
'//                                Private variables
'//
'////////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////////

Private Title As String = "Force on a slug"

' File handling.
' (Add COM file "Microsoft Excel 12.0 Object library" to the project's References.)
Private Filewriter As System.IO.StreamWriter
Private Filereader As System.IO.StreamReader

' Six key summations are two field components and four derivatives.
' First dimension is FnumZ different values of the axial distance z.
' Second dimension is FnumR different values of the radial distance r.
' Both dimensions are zero-based.
Private Br(FnumZ, FnumR) As Double
Private Bz(FnumZ, FnumR) As Double
Private dBrdr(FnumZ, FnumR) As Double
Private dBrdz(FnumZ, FnumR) As Double
Private dBzdr(FnumZ, FnumR) As Double
Private dBzdz(FnumZ, FnumR) As Double

' Three terms in the integral.
' First dimension is FnumZ different values of the axial distance z.
' Second dimension is FnumR different values of the radial distance r.
' Both dimensions are zero-based.
Private BzdBzdz(FnumZ, FnumR) As Double
Private BzdBrdr(FnumZ, FnumR) As Double
Private BrdBzdr(FnumZ, FnumR) As Double
Private TotalBxdBxdx(FnumZ, FnumR) As Double ' Sum of the three components

' Force on a thin disk and slug.
' Dimension is FnumZ different values of the axial distance z.
' Dimension is zero-based.
Private ForceOnDisk(FnumZ) As Double
Private ForceOnSlug(FnumZ) As Double

' Two spatial dimensions z (axial) and r (radial)
Private Z As Double
Private R As Double
Private DelZ As Double
Private DelR As Double

' Summation variables.
Private Theta As Double         ' Angle around a coil turn
Private CosTheta As Double      ' Cosine of Theta
Private DelTheta As Double      ' Differental of Theta
Private Zprime As Double        ' Axial distance z
```

```
Private DelZprime As Double     ' Differential of axial distance z

' String handlers.
Private TempString As String
Private LenString As Int32

' Miscellaneous variables.
Dim objExcel As Microsoft.Office.Interop.Excel.Application
Dim objExcelWB As Microsoft.Office.Interop.Excel.Workbook
Dim objExcelWS As Microsoft.Office.Interop.Excel.Worksheet

'////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////
' CalculateAndSaveFields() is the principal numerical subroutine.  It
' calculates six summations -- the two components of the magnetic field
' and the four derivaties -- at FnumZ * FnumR different grid points.
'
' The grid is FnumZ points wide and FnumR points high.  Where the grid
' starts and stops is determined by FleftZ, FrightZ, FinnerR and FouterR.
' The grid points are spaced horizontally by (FleftZ-FrightZ)/(FnumZ-1)
' and vertically by (FouterR-FinnerR)/(FnumR-1) vertically.  For all
' matrices, the Z-component is the first dimension and the R-component is
' the second dimension.  The Z-indices count from left to right and the
' R-indices count from bottom to top.  The zero-index is the left-most
' Z-value (being FleftZ) and the bottom-most R-value (being FinnerR). The
' maximum index is the right-most Z-value (being FrightZ) and the top-most
' R-value (being FtopR).
'
' Note carefully that the coil's length is determined as Hcoil = Dwire *
' Nturns.  This will ensure that all three quantities are consistent
' physically, but will likely mean that the coil's length is not a round
' number.
'
' Traverses around turns of the coil are done by dividing the circle into
' 2000 segments.
'
' The results do not include the factor: permeability of free space
' multiplied by the current and divided by 4Pi.  Calculation are carried
' out in the units supplied.  They must be consistent, but need not be SI.
'
' Results are saved in the text file with the name FieldTextFileName.
'
Public Sub CalculateAndSaveFields( _
    ByVal Rcore As Double, ByVal Dwire As Double, _
    ByVal Nturns As Int32, ByVal Nlayers As Int32, _
    ByVal FnumZ As Int32, ByVal FnumR As Int32, _
    ByVal FleftZ As Double, ByVal FrightZ As Double, _
    ByVal FinnerR As Double, ByVal FouterR As Double, _
    ByVal FileName As String)
    Dim Hcoil As Double
    Dim Rlayer As Double
    Dim Zturn As Double
    Dim Numerator0 As Double        ' Temporary Rlayer * DelTheta
    Dim Numerator1 As Double        ' Temporary Z - Zturn
    Dim Numerator2 As Double        ' Temporary (Z - Zturn) * CosTheta
    Dim Numerator3 As Double        ' Temporary (R * CosTheta) - Rlayer
    Dim Numerator4 As Double        ' Temporary R - (Rlayer * CosTheta)
    Dim Denominator As Double       ' Temporary variable
```

```
Dim Denominator1 As Double      ' Temporary variable
Dim Denominator2 As Double      ' Temporary variable
' Prepare the output file.
Filewriter = New System.IO.StreamWriter(FileName)
' Write header information to the text file.
Filewriter.Write("FleftZ= " & Trim(Str(FleftZ)) & vbCrLf)
Filewriter.Write("FrightZ= " & Trim(Str(FrightZ)) & vbCrLf)
Filewriter.Write("FinnerR= " & Trim(Str(FinnerR)) & vbCrLf)
Filewriter.Write("FouterR= " & Trim(Str(FouterR)) & vbCrLf)
Filewriter.Write("FnumZ= " & Trim(Str(FnumZ)) & vbCrLf)
Filewriter.Write("FnumR= " & Trim(Str(FnumR)) & vbCrLf)
Filewriter.Write("Rcore= " & Trim(Str(Rcore)) & vbCrLf)
Filewriter.Write("Dwire= " & Trim(Str(Dwire)) & vbCrLf)
Filewriter.Write("Nturns= " & Trim(Str(Nturns)) & vbCrLf)
Filewriter.Write("Nlayers= " & Trim(Str(Nlayers)) & vbCrLf)
' Set up constants.
Hcoil = Dwire * Nturns
DelZ = -(FleftZ - FrightZ) / (FnumZ - 1)
DelR = (FouterR - FinnerR) / (FnumR - 1)
DelTheta = 2 * Math.PI / 2000
' Loop over grid points in the axial z-direction.
For Iz As Int32 = 0 To (FnumZ - 1) Step 1
    Z = FleftZ + (Iz * DelZ)
    Filewriter.Write("Iz= " & Trim(Str(Iz)) & " Z= " & Trim(Str(Z)) & vbCrLf)
    ' Loop over grid points in the radial r-direction.
    For Ir As Int32 = 0 To (FnumR - 1) Step 1
        R = FinnerR + (Ir * DelR)
        ' Initialize the six summations for this grid point.
        Br(Iz, Ir) = 0
        Bz(Iz, Ir) = 0
        dBrdr(Iz, Ir) = 0
        dBrdz(Iz, Ir) = 0
        dBzdr(Iz, Ir) = 0
        dBzdz(Iz, Ir) = 0
        ' Loop over layers in the solenoid.
        For Ilayer As Int32 = 1 To Nlayers Step 1
            Rlayer = Rcore + ((Ilayer - 0.5) * Dwire)
            Numerator0 = Rlayer * DelTheta
            ' Loop over the turns in the layer.
            For Iturn As Int32 = 1 To Nturns Step 1
                Zturn = -(Hcoil / 2) + ((Iturn - 0.5) * Dwire)
                Numerator1 = Z - Zturn
                ' Loop over segments around a turn.
                For Itheta As Int32 = 1 To 2000 Step 1
                    Theta = (Itheta - 0.5) * DelTheta
                    CosTheta = Math.Cos(Theta)
                    Numerator2 = Numerator1 * CosTheta
                    Numerator3 = (R * CosTheta) - Rlayer
                    Numerator4 = R - (Rlayer * CosTheta)
                    Denominator = (R * R) + _
                        (Numerator1 * Numerator1) + _
                        (Rlayer * Rlayer) - _
                        (2 * R * Rlayer * CosTheta)
                    Denominator1 = 1 / (Denominator * Math.Sqrt(Denominator))
                    Denominator2 = Denominator1 / Denominator
                    ' Br
                    Br(Iz, Ir) = Br(Iz, Ir) + _
                        (Numerator0 * Numerator2 * Denominator1)
```

```
                        ' Bz
                        Bz(Iz, Ir) = Bz(Iz, Ir) - _
                            (Numerator0 * Numerator3 * Denominator1)
                        ' dBr/dr
                        dBrdr(Iz, Ir) = dBrdr(Iz, Ir) - _
                            (3 * Numerator0 * Numerator2 * Numerator4 * Denominator2)
                        ' dBr/dz
                        dBrdz(Iz, Ir) = dBrdz(Iz, Ir) + _
                            (Numerator0 * CosTheta * Denominator1)
                        dBrdz(Iz, Ir) = dBrdz(Iz, Ir) - _
                            (3 * Numerator0 * Numerator1 * Numerator2 * Denominator2)
                        ' dBz/dr
                        dBzdr(Iz, Ir) = dBzdr(Iz, Ir) - _
                            (Numerator0 * CosTheta * Denominator1)
                        dBzdr(Iz, Ir) = dBzdr(Iz, Ir) + _
                            (3 * Numerator0 * Numerator3 * Numerator4 * Denominator2)
                        ' dBz/dz
                        dBzdz(Iz, Ir) = dBzdz(Iz, Ir) + _
                            (3 * Numerator0 * Numerator1 * Numerator3 * Denominator2)
                    Next Itheta
                Next Iturn
            Next Ilayer
            ' Write the six summations to the text file.
            Filewriter.Write("Ir= " & Trim(Str(Ir)) & " R= " & Trim(Str(R)) & vbCrLf)
            Filewriter.Write("Br= " & Trim(Str(Br(Iz, Ir))) & vbCrLf)
            Filewriter.Write("Bz= " & Trim(Str(Bz(Iz, Ir))) & vbCrLf)
            Filewriter.Write("dBrdr= " & Trim(Str(dBrdr(Iz, Ir))) & vbCrLf)
            Filewriter.Write("dBrdz= " & Trim(Str(dBrdz(Iz, Ir))) & vbCrLf)
            Filewriter.Write("dBzdr= " & Trim(Str(dBzdr(Iz, Ir))) & vbCrLf)
            Filewriter.Write("dBzdz= " & Trim(Str(dBzdz(Iz, Ir))) & vbCrLf)
        Next Ir
        ' Display progress to user
        Form1.labelDisplay.Text =
            "Nturns=" & Str(Nturns) & vbCrLf & _
            "Nlayers=" & Str(Nlayers) & vbCrLf & _
            "Iz=" & Str(Iz)
        Form1.labelDisplay.Refresh()
        Application.DoEvents()
    Next Iz
    Filewriter.Close()
End Sub

'////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////
'
Public Sub ReadAndCalculateSpatialDistribution(ByVal FileName As String)
    Form1.labelDisplay.Text = "Now reading force file."
    Form1.labelDisplay.Refresh()
    ReadFieldFile(Rcore, Dwire, Nturns, Nlayers, _
        FnumZ, FnumR, FleftZ, FrightZ, FinnerR, FouterR, _
        FileName)
    Form1.labelDisplay.Text = "Now calculating ..."
    Form1.labelDisplay.Refresh()
    CalculateForceComponents(FnumZ, FnumR)
    CalculateForceOnThinDisk(FnumZ, FnumR, FinnerR, FouterR, Rslug)
    CalculateForceOnSlug(FnumZ, FnumR, FleftZ, FrightZ, Rslug, Hslug)
    ' Transfer results into Ftable(,).
    ' The field file results are already negative (attractive).
```

```
    Dim DelZgrid As Double = (FleftZ - FrightZ) / (FnumZ - 1)
    For Iz As Int32 = 0 To FnumZ Step 1
        Ftable(Iz, 1) = FleftZ - (Iz * DelZgrid)
        Ftable(Iz, 2) = ForceOnSlug(Iz)
    Next Iz
    Form1.labelDisplay.Text = "All finished calculating forces."
    Form1.labelDisplay.Refresh()
End Sub

'////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////
' ReadFieldFile() reads the data from the text file created by the
' subroutine CalculateAndSaveFields().  All data is saved in the same
' variables from whence it originally came.
'
Public Sub ReadFieldFile( _
    ByRef Rcore As Double, ByRef Dwire As Double, _
    ByRef Nturns As Int32, ByRef Nlayers As Int32, _
    ByRef FnumZ As Int32, ByRef FnumR As Int32, _
    ByRef FleftZ As Double, ByRef FrightZ As Double, _
    ByRef FinnerR As Double, ByRef FouterR As Double, _
    ByVal FileName As String)
    Filereader = New System.IO.StreamReader(FileName)
    Try
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 7) <> "FleftZ=") Then
            MsgBox("Error in line FleftZ=")
            Exit Sub
        Else
            FleftZ = Val(Strings.Right(TempString, Len(TempString) - 7))
        End If
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 8) <> "FrightZ=") Then
            MsgBox("Error in line FrightZ=")
            Exit Sub
        Else
            FrightZ = Val(Strings.Right(TempString, Len(TempString) - 8))
        End If
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 8) <> "FinnerR=") Then
            MsgBox("Error in line FinnerR=")
            Exit Sub
        Else
            FinnerR = Val(Strings.Right(TempString, Len(TempString) - 8))
        End If
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 8) <> "FouterR=") Then
            MsgBox("Error in line FouterR=")
            Exit Sub
        Else
            FouterR = Val(Strings.Right(TempString, Len(TempString) - 8))
        End If
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 6) <> "FnumZ=") Then
            MsgBox("Error in line FnumZ=")
            Exit Sub
        Else
            FnumZ = CInt(Val(Strings.Right(TempString, Len(TempString) - 6)))
```

```
End If
TempString = Filereader.ReadLine()
If (Strings.Left(TempString, 6) <> "FnumR=") Then
    MsgBox("Error in line FnumR=")
    Exit Sub
Else
    FnumR = CInt(Val(Strings.Right(TempString, Len(TempString) - 6)))
End If
TempString = Filereader.ReadLine()
If (Strings.Left(TempString, 6) <> "Rcore=") Then
    MsgBox("Error in line Rcore=")
    Exit Sub
Else
    Rcore = Val(Strings.Right(TempString, Len(TempString) - 6))
End If
TempString = Filereader.ReadLine()
If (Strings.Left(TempString, 6) <> "Dwire=") Then
    MsgBox("Error in line Dwire=")
    Exit Sub
Else
    Dwire = Val(Strings.Right(TempString, Len(TempString) - 6))
End If
TempString = Filereader.ReadLine()
If (Strings.Left(TempString, 7) <> "Nturns=") Then
    MsgBox("Error in line Nturns=")
    Exit Sub
Else
    Nturns = CInt(Val(Strings.Right(TempString, Len(TempString) - 7)))
End If
TempString = Filereader.ReadLine()
If (Strings.Left(TempString, 8) <> "Nlayers=") Then
    MsgBox("Error in line Nlayers=")
    Exit Sub
Else
    Nlayers = CInt(Val(Strings.Right(TempString, Len(TempString) - 8)))
End If
For Iz = 0 To (FnumZ - 1) Step 1
    ' Display progress to user
    Form1.labelDisplay.Text = "Reading Iz=" & Str(Iz)
    Form1.labelDisplay.Refresh()
    TempString = Filereader.ReadLine()
    If (Strings.Left(TempString, 3) <> "Iz=") Then
        MsgBox("Expected Iz= and read " & TempString)
        Exit Sub
    End If
    TempString = Strings.Right(TempString, Len(TempString) - 3)
    If (Val(TempString) <> Iz) Then
        MsgBox("Error reading line Iz= " & TempString)
        Exit Sub
    End If
    For Ir = 0 To (FnumR - 1) Step 1
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 3) <> "Ir=") Then
            MsgBox("Expected Ir= and read " & TempString)
            Exit Sub
        End If
        TempString = Strings.Right(TempString, Len(TempString) - 3)
        If (Val(TempString) <> Ir) Then
```

```
            MsgBox("Error reading line Ir= " & TempString)
            Exit Sub
        End If
        ' Read Br
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 3) <> "Br=") Then
            MsgBox("Error reading Br when Iz= " & Str(Iz) & _
                " and Ir= " & Str(Ir))
            Exit Sub
        End If
        TempString = Strings.Right(TempString, Len(TempString) - 3)
        Br(Iz, Ir) = Val(TempString)
        ' Read Bz
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 3) <> "Bz=") Then
            MsgBox("Error reading Bz when Iz= " & Str(Iz) & _
                " and Ir= " & Str(Ir))
            Exit Sub
        End If
        TempString = Strings.Right(TempString, Len(TempString) - 3)
        Bz(Iz, Ir) = Val(TempString)
        ' Read dBr/dr
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 6) <> "dBrdr=") Then
            MsgBox("Error reading dBrdr when Iz= " & Str(Iz) & _
                 " and Ir= " & Str(Ir))
            Exit Sub
        End If
        TempString = Strings.Right(TempString, Len(TempString) - 6)
        dBrdr(Iz, Ir) = Val(TempString)
        ' Read dBr/dz
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 6) <> "dBrdz=") Then
            MsgBox("Error in dBrdz when Iz= " & Str(Iz) & _
                " and Ir= " & Str(Ir))
            Exit Sub
        End If
        TempString = Strings.Right(TempString, Len(TempString) - 6)
        dBrdz(Iz, Ir) = Val(TempString)
        ' Read dBz/dr
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 6) <> "dBzdr=") Then
            MsgBox("Error reading dBzdr when Iz= " & Str(Iz) & _
                " and Ir= " & Str(Ir))
            Exit Sub
        End If
        TempString = Strings.Right(TempString, Len(TempString) - 6)
        dBzdr(Iz, Ir) = Val(TempString)
        ' Read dBz/dz
        TempString = Filereader.ReadLine()
        If (Strings.Left(TempString, 6) <> "dBzdz=") Then
            MsgBox("Error reading dBzdz when Iz= " & Str(Iz) & _
                " and Ir= " & Str(Ir))
            Exit Sub
        End If
        TempString = Strings.Right(TempString, Len(TempString) - 6)
        dBzdz(Iz, Ir) = Val(TempString)
    Next Ir
```

```
        Next Iz
        Filereader.Close()
    Catch e As Exception
        MsgBox("Error reading field file " & e.ToString)
        Exit Sub
    End Try
End Sub

'////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////
' CalculateForceComponents() calculates the three axial terms of the
' force at each grid point where the two magnetic field components and
' the four derivatives are known.  Do not forget that all variables must
' exist before calling this subroutine.
'
Public Sub CalculateForceComponents( _
    ByVal FnumZ As Int32, ByVal FnumR As Int32)
    For Iz As Int32 = 0 To (FnumZ - 1) Step 1
        For Ir As Int32 = 0 To (FnumR - 1) Step 1
            BzdBzdz(Iz, Ir) = Bz(Iz, Ir) * dBzdz(Iz, Ir)
            BrdBzdr(Iz, Ir) = Br(Iz, Ir) * dBzdr(Iz, Ir)
            BzdBrdr(Iz, Ir) = Bz(Iz, Ir) * dBrdr(Iz, Ir)
            TotalBxdBxdx(Iz, Ir) = _
                BzdBzdz(Iz, Ir) + BrdBzdr(Iz, Ir) + BzdBrdr(Iz, Ir)
        Next Ir
    Next Iz
End Sub

'////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////
' CalculateForceOnThinDisk() calculates the force acting on a thin disk.
' Enough radius information needs to be available so that this
' subroutine can integrate across the face of the disk.  The integration
' is carried out in 1000 steps and values of the force components are
' interpolated linearly from those at the known grid points.
'
Public Sub CalculateForceOnThinDisk( _
    ByVal FnumZ As Int32, ByVal FnumR As Int32, _
    ByVal FinnerR As Double, ByVal FouterR As Double, _
    ByVal Rslug As Double)
    Dim DelRgrid As Double
    Dim DelRdisk As Double
    Dim IgridLow As Int32    ' Lower bound for interpolation
    Dim IgridHigh As Int32   ' Lower bound for interpolation
    Dim RdiskMid As Double   ' Mid-point of radial segment on disk
    Dim IntBzdBzdz As Double ' Interpolated value
    Dim IntBzdBrdr As Double ' Interpolated value
    Dim IntBrdBzdr As Double ' Interpolated value
    DelRgrid = (FouterR - FinnerR) / (FnumR - 1)
    DelRdisk = Rslug / 1000
    For Iz As Int32 = 0 To (FnumZ - 1) Step 1
        ' Initialize the cumulative total.
        ForceOnDisk(Iz) = 0
        For Ir As Int32 = 1 To 1000 Step 1
            RdiskMid = (Ir - 0.5) * DelRdisk
            ' Provisionally calculate the nearest surrounding grid points.
            IgridLow = CInt(RdiskMid / DelRgrid)
            IgridHigh = IgridLow + 1
```

```
            ' Ensure that RgridLow is less than or equal to RdiskMid.
            If ((IgridLow * DelRgrid) > RdiskMid) Then
                IgridLow = IgridLow - 1
                IgridHigh = IgridLow + 1
            End If
            ' Ensure that RgridHigh is greater than or equal to RdiskMid.
            If ((IgridHigh * DelRgrid) < RdiskMid) Then
                IgridLow = IgridLow - 1
                IgridHigh = IgridLow + 1
            End If
            ' Interpolate the force components from the surrounding grid points.
            IntBzdBzdz = BzdBzdz(Iz, IgridLow) + _
                ((RdiskMid - (IgridLow * DelRgrid)) * _
                (BzdBzdz(Iz, IgridHigh) - BzdBzdz(Iz, IgridLow)) / DelRgrid)
            IntBzdBrdr = BzdBrdr(Iz, IgridLow) + _
                ((RdiskMid - (IgridLow * DelRgrid)) * _
                (BzdBrdr(Iz, IgridHigh) - BzdBrdr(Iz, IgridLow)) / DelRgrid)
            IntBrdBzdr = BrdBzdr(Iz, IgridLow) + _
                ((RdiskMid - (IgridLow * DelRgrid)) * _
                (BrdBzdr(Iz, IgridHigh) - BrdBzdr(Iz, IgridLow)) / DelRgrid)
            ' Add the increment to the integral.
            ForceOnDisk(Iz) = ForceOnDisk(Iz) + _
                (RdiskMid * IntBzdBzdz * DelRdisk) + _
                (RdiskMid * IntBzdBrdr * DelRdisk) + _
                (RdiskMid * IntBrdBzdr * DelRdisk)
        Next Ir
    Next Iz
End Sub

'////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////
' SaveForceOnThinDiskFileForPlotting() saves the force on a thin disk in
' an Excel file for plotting purposes.  The complete path name of the
' Excel file is passed in as an argument.  Do not forget that all
' variables must exist before calling this subroutine.
'
Public Sub SaveForceOnThinDiskFileForPlotting( _
    ByVal Rcore As Double, ByVal Dwire As Double, _
    ByVal Nturns As Int32, ByVal Nlayers As Int32, _
    ByVal FnumZ As Int32, ByVal FnumR As Double, _
    ByVal FleftZ As Double, ByVal FrightZ As Double, _
    ByVal FinnerR As Double, ByVal FouterR As Double, _
    ByVal FileName As String)
    Dim DelZ As Double
    Dim DelR As Double
    Dim RowNum As Int32
    Dim TempNum As Double
    DelZ = (FleftZ - FrightZ) / (FnumZ - 1)
    DelR = (FouterR - FinnerR) / (FnumR - 1)
    ' Open the Excel file.
    Try
        objExcel = CType(CreateObject("Excel.Application"),  _
            Microsoft.Office.Interop.Excel.Application)
        objExcel.Visible = False
        objExcelWB = CType(objExcel.Workbooks.Open(FileName),  _
            Microsoft.Office.Interop.Excel.Workbook)
        objExcelWS = CType(objExcelWB.Sheets("Sheet1"),  _
            Microsoft.Office.Interop.Excel.Worksheet)
```

```
    Catch ex As Exception
        Cursor.Current = Cursors.Default
        MsgBox("Could not open the Excel file.", vbOKOnly)
        Exit Sub
    End Try
    ' Write header information into Excel file.
    With objExcelWS
        objExcelWS.Cells(1, 1) = "Force on thin disk"
        objExcelWS.Cells(2, 1) = "FleftZ=" & Trim(Str(FleftZ))
        objExcelWS.Cells(3, 1) = "FrightZ=" & Trim(Str(FrightZ))
        objExcelWS.Cells(4, 1) = "FnumZ=" & Trim(Str(FnumZ))
        objExcelWS.Cells(5, 1) = "FinnerR=" & Trim(Str(FinnerR))
        objExcelWS.Cells(6, 1) = "FouterR=" & Trim(Str(FouterR))
        objExcelWS.Cells(7, 1) = "FnumR=" & Trim(Str(FnumR))
        objExcelWS.Cells(8, 1) = "Rcore=" & Trim(Str(Rcore))
        objExcelWS.Cells(9, 1) = "Dwire=" & Trim(Str(Dwire))
        objExcelWS.Cells(10, 1) = "Nturns=" & Trim(Str(Nturns))
        objExcelWS.Cells(11, 1) = "Nlayers=" & Trim(Str(Nlayers))
        Hcoil = Dwire * Nturns
        objExcelWS.Cells(12, 1) = "Hcoil=" & Trim(Str(Hcoil))
        objExcelWS.Cells(13, 1) = "DelZ=" & Trim(Str(DelZ))
        objExcelWS.Cells(14, 1) = "DelR=" & Trim(Str(DelR))
        ' Write headers.
        objExcelWS.Cells(16, 1) = "Iz"
        objExcelWS.Cells(16, 2) = "Z"
        objExcelWS.Cells(16, 3) = "Force"
        ' Write the ForceOnDisk data.
        RowNum = 17
        For Iz As Int32 = 0 To (FnumZ - 1) Step 1
            RowNum = RowNum + 1
            objExcelWS.Cells(RowNum, 1) = Trim(Str(Iz))
            TempNum = FleftZ - (Iz * DelZ)
            objExcelWS.Cells(RowNum, 2) = Trim(Str(TempNum))
            objExcelWS.Cells(RowNum, 3) = Trim(Str(ForceOnDisk(Iz)))
        Next Iz
    End With
    objExcelWB.Save()
    objExcelWB.Close()
End Sub

'////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////
' CalculateForceOnSlug() calculates the force acting on a slug.  The
' results are saved in ForceOnSlug(), where the dimension is the axial
' displacement of the geometric center of the slug.
'
' This subroutine has been modified from the template.  Previously, the
' force on the slug was set to zero if ANY PART of the slug lies outside
' of the grid at which the magnetic data has been calculated.  Now, the
' force on the disk is calculated where the force on a component thin
' disk is set to zero if the thin disk lies outside of the grid at which
' the magnetic data has been calculate.
'
' The slug is divided axially into 500 thin disks.  Before calling this
' subroutine, the subroutine CalculateForceOnThinDisk() must be called
' and vector ForceonDisk(,1) loaded at all axial displacements on the
' grid with the force on a disk with the same radius as the slug.  The
' values of the force on the thin disk at axial displacements which are
```

```
' not on the grid are interpolated linearly from those at the known grid
' points.  Do not forget that all variables must exist before calling
' this subroutine.
'
' Note that the index Iz = 0 corresponds to the left-hand side of the curve,
' where Z = FleftZ.  The index Iz = NumZ corresponds to the right-hand side
' of the curve, where Z = FrightZ, where FrightZ < FleftZ.  In this subroutine,
' delZ is defined to be algebraically negative.  ZslugMax is the left-hand
' end of the slug, with the greater Z-value.
'
Public Sub CalculateForceOnSlug( _
    ByVal FNumZ As Int32, ByVal FnumR As Int32, _
    ByVal FleftZ As Double, ByVal FrightZ As Double, _
    ByVal Rslug As Double, ByVal Hslug As Double)
    Dim DelZgrid As Double
    Dim IgridLeft As Int32   ' Left grid index for interpolation
    Dim IgridRight As Int32  ' Right grid index for interpolation
    Dim ZgridLeft As Double  ' Left grid Z-value for interpolation
    Dim ZgridRight As Double ' Right grid Z-value for interpolation
    Dim ZslugMid As Double   ' Axial mid-point of the slug
    Dim ZslugMax As Double   ' Axial displacement of positive face of the slug
    Dim ZslugMin As Double   ' Axial displacement of negative face of the slug
    Dim ZdiskMid As Double   ' Axial mid-point of a thin disk in the slug
    Dim IntForceOnDisk As Double ' Interpolated value of force on a thin disk
    DelZgrid = (FleftZ - FrightZ) / (FNumZ - 1)
    For Iz As Int32 = 0 To (FNumZ - 1) Step 1
        ZslugMid = FleftZ - (Iz * DelZgrid)
        ZslugMax = ZslugMid + (0.5 * Hslug)
        ZslugMin = ZslugMid - (0.5 * Hslug)
        ' Initialize the cumulative total.
        ForceOnSlug(Iz) = 0
        ' Step through the slug from right to left.
        For Idisk As Int32 = 1 To 500 Step 1
            ZdiskMid = ZslugMin + ((Idisk - 0.5) * Hslug / 500)
            ' If ZdiskMid is outside of the grid on the right, then set
            ' the force acting on it to zero.
            If (ZdiskMid < 0) Then
                IntForceOnDisk = 0
            Else
                ' If ZdiskMid is outside the grid on the left, then set
                ' the force acting on it to zero.
                If (ZdiskMid > ((FNumZ - 1) * DelZgrid)) Then
                    IntForceOnDisk = 0
                Else
                    ' We are here if ZdiskMid is in the grid.
                    ' Provisionally calculate the nearest surrounding grid points.
                    ' The Z-value corresponding to IgridLeft should be to the left
                    ' of ZdiskMid and the Z-value corresponding to IgridRight should
                    ' be to the right of ZdiskMid.
                    IgridLeft = CInt((FleftZ - ZdiskMid) / DelZgrid)
                    IgridRight = IgridLeft + 1
                    ZgridLeft = FleftZ - (IgridLeft * DelZgrid)
                    ZgridRight = FleftZ - (IgridRight * DelZgrid)
                    ' Ensure that ZgridRight is less than or equal to ZdiskMid.
                    If (ZgridRight > ZdiskMid) Then
                        IgridLeft = IgridLeft + 1
                        IgridRight = IgridRight + 1
                    End If
```

```
                    ' Ensure that ZgridLeft is greater than or equal to ZdiskMid.
                    If (ZgridLeft < ZdiskMid) Then
                        IgridLeft = IgridLeft - 1
                        IgridRight = IgridRight - 1
                    End If
                    ZgridLeft = FleftZ - (IgridLeft * DelZgrid)
                    ' Interpolate the force components from surrounding grid points.
                    IntForceOnDisk = ForceOnDisk(IgridLeft) + _
                        ((ZdiskMid - ZgridLeft) * _
                        (ForceOnDisk(IgridRight) - ForceOnDisk(IgridLeft)) _
                        / DelZgrid)
                End If
            End If
            ' Add the increment to the integral.
            ForceOnSlug(Iz) = ForceOnSlug(Iz) + (IntForceOnDisk * Hslug / 500)
        Next Idisk
        ' Display progress to user.
        Form1.labelDisplay.Text = "Calculating force at Iz = " & Trim(Str(Iz))
        Form1.labelDisplay.Refresh()
    Next Iz
End Sub

'////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////
' SaveForceOnSlugForPlotting() saves the force on a slug in an Excel file
' for plotting purposes.  The complete path name of the Excel file is
' passed in as an argument.  Do not forget that all variables must exist
' before calling this subroutine.
'
Public Sub SaveForceOnSlugFileForPlotting( _
    ByVal Rcore As Double, ByVal Dwire As Double, _
    ByVal Nturns As Int32, ByVal Nlayers As Int32, _
    ByVal FnumZ As Int32, ByVal FnumR As Int32, _
    ByVal FleftZ As Double, ByVal FrightZ As Double, _
    ByVal FinnerR As Double, ByVal FouterR As Double, _
    ByVal Rslug As Double, ByVal Hslug As Double, _
    ByVal FileName As String)
    Dim DelZ As Double
    Dim DelR As Double
    Dim RowNum As Int32
    Dim TempNum As Double
    DelZ = (FleftZ - FrightZ) / (FnumZ - 1)
    DelR = (FinnerR - FouterR) / (FnumR - 1)
    ' Open Excel file.
    Try
        objExcel = CType(CreateObject("Excel.Application"),  _
            Microsoft.Office.Interop.Excel.Application)
        objExcel.Visible = False
        objExcelWB = CType(objExcel.Workbooks.Open(FileName),  _
            Microsoft.Office.Interop.Excel.Workbook)
        objExcelWS = CType(objExcelWB.Sheets("Sheet1"),  _
            Microsoft.Office.Interop.Excel.Worksheet)
    Catch ex As Exception
        Cursor.Current = Cursors.Default
        MsgBox("Could not open the Excel file.", vbOKOnly)
        Exit Sub
    End Try
    ' Write header information into Excel file.
```

```
        With objExcelWS
            objExcelWS.Cells(1, 1) = "Force on slug"
            objExcelWS.Cells(2, 1) = "FleftZ=" & Trim(Str(FleftZ))
            objExcelWS.Cells(3, 1) = "FrightZ=" & Trim(Str(FrightZ))
            objExcelWS.Cells(4, 1) = "FnumZ=" & Trim(Str(FnumZ))
            objExcelWS.Cells(5, 1) = "FinnerR=" & Trim(Str(FinnerR))
            objExcelWS.Cells(6, 1) = "FouterR=" & Trim(Str(FouterR))
            objExcelWS.Cells(7, 1) = "FnumR=" & Trim(Str(FnumR))
            objExcelWS.Cells(8, 1) = "Rcore=" & Trim(Str(Rcore))
            objExcelWS.Cells(9, 1) = "Dwire=" & Trim(Str(Dwire))
            objExcelWS.Cells(10, 1) = "Nturns=" & Trim(Str(Nturns))
            objExcelWS.Cells(11, 1) = "Nlayers=" & Trim(Str(Nlayers))
            Hcoil = Dwire * Nturns
            objExcelWS.Cells(12, 1) = "Hcoil=" & Trim(Str(Hcoil))
            objExcelWS.Cells(13, 1) = "DelZ=" & Trim(Str(DelZ))
            objExcelWS.Cells(14, 1) = "DelR=" & Trim(Str(DelR))
            objExcelWS.Cells(15, 1) = "Rslug=" & Trim(Str(Rslug))
            objExcelWS.Cells(16, 1) = "Hslug=" & Trim(Str(Hslug))
            ' Write headers across the page.
            objExcelWS.Cells(18, 1) = "Iz"
            objExcelWS.Cells(18, 2) = "Z"
            objExcelWS.Cells(18, 3) = "Force"
            ' Write the ForceOnSlug data.
            RowNum = 19
            For Iz As Int32 = 0 To (FnumZ - 1) Step 1
                RowNum = RowNum + 1
                objExcelWS.Cells(RowNum, 1) = Trim(Str(Iz))
                TempNum = FleftZ - (Iz * DelZ)
                objExcelWS.Cells(RowNum, 2) = Trim(Str(TempNum))
                objExcelWS.Cells(RowNum, 3) = Trim(Str(ForceOnSlug(Iz)))
            Next Iz
        End With
        objExcelWB.Save()
        objExcelWB.Close()
    End Sub

End Module
```

Listing of *Module3*

```
Option Strict On
Option Explicit On

Module Module3

    '*************** Interpolation of the relative permeability ***************

    ' Module 3 is a function which returns the relative permeability for the
    ' geometric center of the slug at certain given parameters.

    Private MuTable(20, 2) As Double

    Public Sub InitializeMuTable()
        MuTable(1, 1) = 0 : MuTable(1, 2) = 790.6271
        MuTable(2, 1) = 0.0003 : MuTable(2, 2) = 833.4465
        MuTable(3, 1) = 0.001 : MuTable(3, 2) = 924.9707
        MuTable(4, 1) = 0.002 : MuTable(4, 2) = 624.9802
        MuTable(5, 1) = 0.003 : MuTable(5, 2) = 463.338
        MuTable(6, 1) = 0.005 : MuTable(6, 2) = 304.998
        MuTable(7, 1) = 0.01 : MuTable(7, 2) = 170.9989
        MuTable(8, 1) = 0.02 : MuTable(8, 2) = 93.5
        MuTable(9, 1) = 0.03 : MuTable(9, 2) = 65.1665
        MuTable(10, 1) = 0.05 : MuTable(10, 2) = 40.3999
        MuTable(11, 1) = 0.1 : MuTable(11, 2) = 21.1
        MuTable(12, 1) = 0.2 : MuTable(12, 2) = 11.125
        MuTable(13, 1) = 0.4 : MuTable(13, 2) = 6.075
        MuTable(14, 1) = 0.47 : MuTable(14, 2) = 5.3192
        MuTable(15, 1) = 0.96999 : MuTable(15, 2) = 3.0928
        MuTable(16, 1) = 1.96996 : MuTable(16, 2) = 2.0305
        MuTable(17, 1) = 2.97 : MuTable(17, 2) = 1.6835
        MuTable(18, 1) = 7.97003 : MuTable(18, 2) = 1.2547
        MuTable(19, 1) = 17.96945 : MuTable(19, 2) = 1.113
        MuTable(20, 1) = 1000000 : MuTable(20, 2) = 1
    End Sub

    Public Function GetRelativePermeability( _
        ByVal Zslug As Double, ByVal Current As Double, _
        ByVal Nturns As Int32, ByVal Nlayers As Int32, _
        ByVal Hcoil As Double, _
        ByRef BFaceCenter As Double, _
        ByRef F_FaceCenter As Double, _
        ByRef F_Zslug As Double, _
        ByRef BZSlug As Double) As Double
        Dim mu0 As Double = 4 * Math.PI * 0.0000001
        Dim IndexL As Int32
        Dim IndexR As Int32
        ' Calculate the applied flux density at the face center.
        BFaceCenter = 0.5 * mu0 * Nturns * Nlayers * Current / Hcoil
        ' Field strength will always be positive.
        BFaceCenter = Math.Abs(BFaceCenter)
        ' Look up the spatial distribution of the force at the face center.
        F_FaceCenter = InterpolateForceField(0.5 * Hcoil)
        ' Look up the spatial distribution of the force at Zslug.
        F_Zslug = InterpolateForceField(Zslug)
        ' Calculate the applied flux density at the center of the slug.
```

```
        BZSlug = (F_Zslug / F_FaceCenter) * BFaceCenter
        ' Ensure that the applied flux density does not exceed 0.05.
        If (BZSlug > 100) Then
            MsgBox("Error while calculating the relative permeability." & vbCrLf & _
                "The applied field strength is greater than 100 Tesla.")
            GetRelativePermeability = 0
            Exit Function
        End If
        ' Ensure that the applied flux density is greater than zero.
        If (BZSlug < 0) Then
            If (BZSlug > -0.0000001) Then
                BZSlug = 0
            Else
                MsgBox("Logic error #1 when calculating the relative permeability.")
                MsgBox("BZslug = " & Trim(Str(BZSlug)) & vbCrLf & _
                    "F_Zslug = " & Trim(Str(F_Zslug)) & vbCrLf & _
                    "F_FaceCenter = " & Trim(Str(F_FaceCenter)) & vbCrLf & _
                    "BFaceCenter = " & Trim(Str(BFaceCenter)))
                GetRelativePermeability = 0
                Exit Function
            End If
        End If
        ' Find the bounding indices.
        IndexL = -1
        For I As Int32 = 1 To 19 Step 1
            If ((MuTable(I, 1) <= BZSlug) And (MuTable(I + 1, 1) > BZSlug)) Then
                IndexL = I
                IndexR = I + 1
                Exit For
            End If
        Next I
        If (IndexL < 0) Then
            MsgBox("Logic error #2 when calculating the relative permeability.")
            GetRelativePermeability = 0
            Exit Function
        End If
        ' Interpolate the relative permeability.
        GetRelativePermeability = MuTable(IndexL, 2) + _
            ((MuTable(IndexR, 2) - MuTable(IndexL, 2)) * _
            (BZSlug - MuTable(IndexL, 1)) / _
            (MuTable(IndexR, 1) - MuTable(IndexL, 1)))
    End Function

End Module
```

Listing of *Variables*, a module which defines global variables

```
Module Variables

    ' Timing variables
    Public T As Double                  ' The current time in the integration
    Public delT As Double = 0.0000001   ' The time step for the integration is 100 ns
    Public Zfinished As Double          ' End the integration when the slug gets here

    ' Physical parameters of the coil, with base case values.
    Public Rcore As Double = 0.01       ' The core is 2 cm in diameter
    Public Hcoil As Double = 0.4981     ' The coil is 49.81 cm long
    Public Dwire As Double = 0.005189   ' The wire has a diameter of 5.189 mm
    Public Nturns As Int32 = 96         ' Turns in each layer
    Public Nlayers As Int32 = 1         ' Number of layers of wire in the winding
    Public LenWire As Double            ' The length of wire used in the winding
    Public RPerM As Double = 0.00081543 ' Resistance of the wire in Ohms per meter
    Public RADavg As Double             ' Average radius of a turn in the winding
    Public AREAavg As Double            ' Average cross-sectional area of the winding
    Public Rcoil As Double              ' Resistance of the coil, in Ohms
    Public Lcoil As Double              ' Inductance of the coil, in Henries

    ' Variables for the spatial force field, expressed as points in a table.
    Public Ftable(10000, 2) As Double   ' (*,1) is the position; (*,2) is the magnitude
    '                                   ' Force values are stored negative
    Public FnumZ As Int32 = 1201        ' The number of grid columns in the region
    Public FleftZ As Double = 0.36      ' Left edge of region of interest
    Public FrightZ As Double = 0        ' Right edge of region of interest
    Public FnumR As Int32 = 399         ' The number of grid rows in the region
    Public FinnerR As Double = 0        ' Bottom edge of region of interest
    Public FouterR As Double = Rcore * 0.99 ' Top edge of region of interest

    ' Characteristics of other components in the circuit, with base case values.
    Public Resr As Double = 2           ' Equivalent series resistance of the capacitor
    Public Rsw As Double = 0.15         ' Static resistance of the switch
    Public C As Double = 0.0001         ' Capacitance = 100uF
    Public R As Double                  ' Total series resistance in the circuit
    Public L As Double                  ' Total series inductance in the circuit

    ' Initial conditions for a run, with base case values.
    Public VC0 As Double = 4000         ' Initial voltage over the capacitor
    Public QC0 As Double = 0.4          ' Initial charge stored in the capacitor
    Public Z0 As Double                 ' The slug's starting position
    Public S0 As Double = 0             ' The slug's starting speed

    ' Characteristics of the slug
    Public Rslug As Double = 0.0045     ' Slug has 9 mm diameter
    Public Hslug As Double = 0.02       ' Length of the slug
    Public RHOslug As Double            ' Mass density of the slug
    Public M As Double                  ' Mass of the slug
    Public Mu0 As Double = 4 * Math.PI * 0.0000001 ' Permeability of free space

    ' Voltages
    Public VCstart As Double            ' Voltage over the capacitor at start of step
    Public VCend As Double              ' Voltage over the capacitor at end of step
    Public VRstart As Double            ' Voltage over the resistor at start of step
    Public VRend As Double              ' Voltage over the resistor at end of step
```

```
Public VLstart As Double            ' Voltage over the inductor at start of step
Public VLend As Double              ' Voltage over the inductor at end of step
Public VSstart As Double            ' Voltage over the slug at start of step
Public VSend As Double              ' Voltage over the slug at end of step
Public ErrVstart As Double          ' Error in the total starting voltages
Public ErrVend As Double            ' Error in the total ending voltages

' Currents
Public QCstart As Double            ' Charge on capacitor at start of step
Public QCend As Double              ' Charge on capacitor at end of step
Public Istart As Double             ' Current at start of step
Public Iend As Double               ' Current at end of step
Public d2Qdt2start As Double        ' d2Q/dt2 at start of step

' Energy stocks
Public ECstart As Double            ' Energy in the capacitor at start of step
Public ECend As Double              ' Energy in the capacitor at end of step
Public delEC As Double              ' Change in capacitor's energy during step
Public ERstart As Double            ' Cumulative heat at start of step
Public ERend As Double              ' Cumulative heat at end of step
Public delER As Double              ' Change in heat during step
Public ELstart As Double            ' Energy in the inductor at start of step
Public ELend As Double              ' Energy in the inductor at end of step
Public delEL As Double              ' Change in inductor's energy during step
Public ESstart As Double            ' Energy in the slug at start of step
Public ESend As Double              ' Energy in the slug at end of step
Public delES As Double              ' Change in slug's energy during step
Public ETstart As Double            ' Total system energy at start of step
Public ETend As Double              ' Total system energy at end of step
Public delET As Double              ' Change in total energy during step

' Variables for the slug during a single time step
Public Zstart As Double             ' The slug's position at start of a step
Public Zend As Double               ' The slug's position at end of a step
Public Sstart As Double             ' The slug's speed at start of a step
Public Send As Double               ' The slug's speed at end of a step
Public Astart As Double             ' The slug's acceleration at start of step
Public MuRmid As Double             ' Relative permeability at middle of step
Public MuRend As Double             ' Relative permeability at end of step

' Variables for the force during a single step
' Variable names F_ refer to the spatial distribution factor only.
Public ZendEst As Double            ' Estimate of the slug's position at the end
Public F_start As Double            ' Force at the slug's starting position
Public F_endEst As Double           ' Force at the slug's estimated ending position
Public F_avg As Double              ' Average force during the step

' Default file names
Public FieldTextFileName_Default As String = _
    "C:\FieldTextFile.txt"
Public ForceOnThinDiskPlotFileName_Default As String = _
    "C:\ForceOnThinDiskPlotFile.xlsx"
Public ForceOnSlugPlotFileName_Default As String = _
    "C:\ForceOnSlugPlotFile.xlsx"

' Variables needed to save the results
' (Add COM file "Microsoft Excel 12.0 Object library" to the project's References.)
Public objExcel As Microsoft.Office.Interop.Excel.Application
```

```
    Public objExcelWB As Microsoft.Office.Interop.Excel.Workbook
    Public objExcelWS As Microsoft.Office.Interop.Excel.Worksheet

    ' Variable passed globally to track the maximum current during a run
    Public MaxCurrent As Double

End Module
```

Listing of *Form1*

```
Option Strict On
Option Explicit On

' Integration6

' Module 1 handles the integration of the circuit and dynamic equations.
' Module 2 handles the creation and retrieval of the spatial force distribution.
' Module 3 handles the interpolation of the relative permeability.

Public Class Form1
    Inherits System.Windows.Forms.Form

    Public Sub New()
        InitializeComponent()
        With Me
            Name = ""
            Text = "Axial Coil Gun - Integration #6"
            FormBorderStyle = Windows.Forms.FormBorderStyle.FixedSingle
            Size = New Drawing.Size(800, 800)
            CenterToScreen()
            .Visible = True
            Controls.Add(buttonCalculateNewField)
            buttonCalculateNewField.BringToFront()
            Controls.Add(gbCalculateNewField)
            gbCalculateNewField.BringToFront()
            gbCalculateNewField.Visible = False
            Controls.Add(buttonPlotField)
            buttonPlotField.BringToFront()
            Controls.Add(gbPlotField)
            gbPlotField.BringToFront()
            gbPlotField.Visible = False
            Controls.Add(buttonStartExecution)
            buttonStartExecution.BringToFront()
            Controls.Add(buttonExit)
            buttonExit.bringtofront()
            Controls.Add(labelDisplay)
            labelDisplay.BringToFront()
            labelDisplay.Visible = False
            PerformLayout()
            BringToFront()
        End With
    End Sub

    Private WithEvents buttonCalculateNewField As New Windows.Forms.Button With _
        {.Size = New Drawing.Size(200, 30), _
         .Location = New Drawing.Point(210, 5), _
         .Text = "Calculate new field", .TextAlign = ContentAlignment.MiddleCenter}

    Private Sub buttonCalculateNewField_Click() _
        Handles buttonCalculateNewField.MouseClick
        labelDisplay.Visible = False
        gbCalculateNewField.Visible = True
    End Sub

    Private gbCalculateNewField As New Windows.Forms.GroupBox With _
```

```
    {.Size = New Drawing.Size(200, 345), _
     .Location = New Drawing.Point(5, 5)}

Private Sub buttonNewFieldGo_Click() Handles buttonNewFieldGo.MouseClick
    gbCalculateNewField.Visible = False
    labelDisplay.Visible = True
    Rcore = Val(tbRcore.Text)
    Dwire = Val(tbDwire.Text)
    Nturns = CInt(Val(tbNturns.Text))
    Nlayers = CInt(Val(tbNlayers.Text))
    FnumZ = CInt(Val(tbFnumZ.Text))
    FleftZ = Val(tbFleftZ.Text)
    FrightZ = Val(tbFrightZ.Text)
    FnumR = CInt(Val(tbFnumR.Text))
    FinnerR = Val(tbFinnerR.Text)
    FouterR = Val(tbFouterR.Text)
    CalculateAndSaveFields(Rcore, Dwire, Nturns, Nlayers, _
        FnumZ, FnumR, FleftZ, FrightZ, FinnerR, FouterR, _
        tbOutputFieldFileName.Text)
    labelDisplay.Text = ""
    MsgBox("All done")
End Sub

Private WithEvents buttonPlotField As New Windows.Forms.Button With _
    {.Size = New Drawing.Size(200, 30), _
     .Location = New Drawing.Point(210, 40), _
     .Text = "Plot field", .TextAlign = ContentAlignment.MiddleCenter}

Private Sub buttonPlotField_Click() Handles buttonPlotField.MouseClick
    labelDisplay.Visible = False
    gbPlotField.Visible = True
End Sub

Public gbPlotField As New Windows.Forms.GroupBox With _
    {.Size = New Drawing.Size(200, 300), _
     .Location = New Drawing.Point(5, 5), _
     .Visible = False}

Private Sub buttonPlotFieldGo_Click() Handles buttonPlotFieldGo.MouseClick
    gbPlotField.Visible = False
    labelDisplay.Visible = True
    ReadAndCalculateSpatialDistribution(tbInputFieldFileName.Text)
    SaveForceOnThinDiskFileForPlotting(Rcore, Dwire, _
        Nturns, Nlayers, FnumZ, FnumR, FleftZ, FrightZ, FinnerR, FouterR, _
        tbThinDiskExcelFileName.Text)
    SaveForceOnSlugFileForPlotting(Rcore, Dwire, _
        Nturns, Nlayers, FnumZ, FnumR, FleftZ, FrightZ, FinnerR, FouterR, _
        Rslug, Hslug, tbSlugExcelFileName.Text)
    labelDisplay.Text = ""
    MsgBox("All done")
End Sub

Private WithEvents buttonStartExecution As New Windows.Forms.Button With _
    {.Size = New Drawing.Size(200, 30), _
     .Location = New Drawing.Point(210, 75), _
     .Text = "Start execution", .TextAlign = ContentAlignment.MiddleCenter}

Private WithEvents buttonExit As New Windows.Forms.Button With _
```

```
    {.Size = New Drawing.Size(200, 30), _
     .Location = New Drawing.Point(210, 110), _
     .Text = "Quit and exit", .TextAlign = ContentAlignment.MiddleCenter}

Private Sub buttonExit_Click() Handles buttonExit.MouseClick
    Application.Exit()
End Sub

Public labelDisplay As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(200, 600), _
     .Location = New Drawing.Point(5, 5), _
     .Text = "", .TextAlign = ContentAlignment.TopLeft, _
     .Visible = False}

Private Sub buttonStartExecution_Click() Handles buttonStartExecution.MouseClick

    '////////////////////////////////////////////////////
    '////////////////////////////////////////////////////
    '// Sub-program #1
    '// Use this block to calculate the force fields for a
    '// selection of coils.
    'labelDisplay.Visible = True
    'Dwire = 0.005189
    'FnumZ = 1201
    'FnumR = 399
    'FrightZ = 0
    'FinnerR = 0
    'Dim FileName As String
    '' Main loop to step through the number of turns.
    'For I As Int32 = 1 To 4 Step 1
    '    Select Case I
    '        Case 1
    '            Nturns = 96
    '            Rcore = 0.01
    '            Hcoil = 0.4981
    '            FleftZ = 0.36
    '            FouterR = 0.99 * Rcore
    '        Case 2
    '            Nturns = 80
    '            Rcore = 0.01252
    '            Hcoil = 0.4151
    '            FleftZ = 0.36
    '            FouterR = 0.99 * Rcore
    '        Case 3
    '            Nturns = 64
    '            Rcore = 0.0163
    '            Hcoil = 0.3321
    '            FleftZ = 0.3
    '            FouterR = 0.99 * Rcore
    '        Case 4
    '            Nturns = 48
    '            Rcore = 0.0226
    '            Hcoil = 0.2491
    '            FleftZ = 0.24
    '            FouterR = 0.99 * Rcore
    '    End Select
    '    ' Main loop to step through the number of layers.
    '    For J As Int32 = 1 To 4 Step 1
```

```
'        Nlayers = J
'        FileName = "C:\FieldTextFile-" & Trim(Str(Nlayers)) & _
'            "x" & Trim(Str(Nturns)) & "TurnsWithExtra1201x399.txt"
'        CalculateAndSaveFields(Rcore, Dwire, Nturns, Nlayers, _
'            FnumZ, FnumR, FleftZ, FrightZ, FinnerR, FouterR, _
'            FileName)
'    Next J
'Next I
'MsgBox("All done.")
'////////////////////////////////////////////////////
'////////////////////////////////////////////////////

'////////////////////////////////////////////////////
'////////////////////////////////////////////////////
'// Use this block to create plot files for a
'// selection of coils.
'// Sub-program #2
'labelDisplay.Visible = True
'Dim InputFileName As String
'Dim OutputFileName1 As String
'Dim OutputFileName2 As String
'For I As Int32 = 1 To 4 Step 1
'    Select Case I
'        Case 1
'            Nturns = 96
'        Case 2
'            Nturns = 80
'        Case 3
'            Nturns = 64
'        Case 4
'            Nturns = 48
'    End Select
'    For J As Int32 = 1 To 4 Step 1
'        Nlayers = J
'        InputFileName = "C:\FieldTextFile-" & Trim(Str(Nlayers)) & _
'            "x" & Trim(Str(Nturns)) & "TurnsWith1201x399.txt"
'        ' Files must exist before execution.
'        OutputFileName1 = "C:\ForceOnThinDiskPlotFile-" & Trim(Str(Nlayers)) & _
'            "x" & Trim(Str(Nturns)) & "TurnsWith1201x399.xlsx"
'        OutputFileName2 = "C:\ForceOnSlugPlotFile-" & Trim(Str(Nlayers)) & _
'            "x" & Trim(Str(Nturns)) & "TurnsWith1201x399.xlsx"
'        ReadAndCalculateSpatialDistribution(InputFileName)
'        SaveForceOnThinDiskFileForPlotting(Rcore, Dwire, Nturns, Nlayers, _
'            FnumZ, FnumR, FleftZ, FrightZ, FinnerR, FouterR, _
'            OutputFileName1)
'        SaveForceOnSlugFileForPlotting(Rcore, Dwire, Nturns, Nlayers, _
'            FnumZ, FnumR, FleftZ, FrightZ, FinnerR, FouterR, _
'            Rslug, Hslug, OutputFileName2)
'    Next J
'Next I
'MsgBox("All done.")
'////////////////////////////////////////////////////
'////////////////////////////////////////////////////

'////////////////////////////////////////////////////
'////////////////////////////////////////////////////
'// Use this block to integrate a selection of runs
'// for four coils.
```

```
'// Sub-program #3
labelDisplay.Visible = True
Dim InputFileName As String
Dim OutputFileName As String
For I As Int32 = 1 To 4 Step 1
    Select Case I
        Case 1
            Nturns = 96
            ' Best Z0 for one layer is 0.277
            ' Best Z0 for two layers is 0.288
            ' Best Z0 for three layers is 0.298
            ' Best Z0 for four layers is 0.309
            Z0 = 0.272
        Case 2
            Nturns = 80
            ' Best Z0 for one layer is 0.239
            ' Best Z0 for two layers is 0.251
            ' Best Z0 for three layers is 0.262
            ' Best Z0 for four layers is 0.271
            Z0 = 0.235
        Case 3
            Nturns = 64
            ' Best Z0 for one layer is 0.204
            ' Best Z0 for two layers is 0.218
            ' Best Z0 for three layers is 0.228
            ' Best Z0 for four layers is 0.239
            Z0 = 0.202
        Case 4
            Nturns = 48
            ' Best Z0 for one layer is 0.173
            ' Best Z0 for two layers is 0.189
            ' Best Z0 for three layers is 0.200
            ' Best Z0 for four layers is 0.211
            Z0 = 0.184
    End Select
    Nlayers = 4
    ' Read the file with the spatial distribution of the force.
    ' This call will set the following parameters:
    '   Rcore, Dwire, Nturns, Nlayers
    '   FnumZ, FleftZ, FrightZ
    InputFileName = "C:\FieldTextFile-" & Trim(Str(Nlayers)) & _
        "x" & Trim(Str(Nturns)) & "TurnsWith1201x399.txt"
    ReadAndCalculateSpatialDistribution(InputFileName)
    ' Initialize the relative permeability table.
    InitializeMuTable()
    ' Open the Excel file for the results.
    ' The file must exist before execution begins.
    OutputFileName = "C:\Integration6Results-" & Trim(Str(Nlayers)) & _
        "x" & Trim(Str(Nturns)) & "TurnsWith1201x399.xlsx"
    Try
        objExcel = CType(CreateObject("Excel.Application"),  _
            Microsoft.Office.Interop.Excel.Application)
        objExcel.Visible = False
        objExcelWB = CType(objExcel.Workbooks.Open(OutputFileName),  _
            Microsoft.Office.Interop.Excel.Workbook)
        objExcelWS = CType(objExcelWB.Sheets("Sheet1"),  _
            Microsoft.Office.Interop.Excel.Worksheet)
    Catch ex As Exception
```

```
    Cursor.Current = Cursors.Default
    MsgBox("Could not open the Excel file to save the output.", vbOKOnly)
    Exit Sub
End Try
' Write the configuration header.
With objExcelWS
    .Cells(1, 1) = "Integration #6 Results"
    .Cells(2, 1) = "File name"      ' Name of file with raw data
    .Cells(4, 1) = "DelT (s)"
    .Cells(5, 1) = "Rcore (m)"
    .Cells(6, 1) = "Dwire (m)"
    .Cells(7, 1) = "Nturns"
    .Cells(8, 1) = "Nlayers"
    .Cells(9, 1) = "Hcoil (m)"
    .Cells(10, 1) = "FnumZ"
    .Cells(11, 1) = "FleftZ (m)"
    .Cells(12, 1) = "FrightZ (m)"
    .Cells(13, 1) = "gridDelZ (m)"
    .Cells(14, 1) = "Rcoil (Ohms)"
    .Cells(15, 1) = "Lcoil (H)"
    .Cells(16, 1) = "RperM (O/m)"
    .Cells(17, 1) = "Rslug (m)"
    .Cells(18, 1) = "Hslug (m)"
    .Cells(19, 1) = "RHOslug (k/m3)"
    .Cells(20, 1) = "Mass (kg)"
    .Cells(21, 1) = "C (F)"
    .Cells(22, 1) = "V0 (V)"
    .Cells(23, 1) = "EC0 (J)"
    .Cells(24, 1) = "Q0 (C)"
    .Cells(25, 1) = "Resr (Ohms)"
    .Cells(26, 1) = "Rsw (Ohms)"
    .Cells(27, 1) = "Rtot (Ohms)"   ' Total series resistance
    .Cells(29, 1) = "Z0 (m)"        ' Starting position of slug
    .Cells(30, 1) = "Kend (J)"      ' Kinetic energy at end of run
    .Cells(31, 1) = "Send (m/s)"    ' Speed at end of run
    .Cells(32, 1) = "REend (J)"     ' Recoverable energy at end of run
End With
' Set the parameters which are fixed during all runs.
delT = 0.0000001    ' The time step for the integration is 100 ns
Rslug = 0.0045      ' The slug has a 9 mm diameter
Hslug = 0.02        ' The slug is 20 mm long
RHOslug = 7850      ' Steel weighs 7850 kg/m^3
M = Math.PI * Rslug * Rslug * Hslug * RHOslug ' Mass of the slug
RPerM = 0.00081543  ' Linear resistance of the wire
C = 0.0001          ' The capacitance is 100uF
VC0 = 4000          ' Initial voltage is 4000V
QC0 = C * VC0       ' Initial charge stored in the capacitor
Resr = 2            ' Equivalent series resistance of the capacitor
Rsw = 0.15          ' Static resistance of the switch
' Calculate the length of the coil.
Hcoil = Dwire * Nturns
' Calculate the resistance and inductance of the coil.
BuildTheCoil()
' Calculate the total circuit parameters.
R = Resr + Rsw + Rcoil
L = Lcoil
' Write the fixed parameters in the header.
With objExcelWS
```

```
            .Cells(2, 2) = OutputFileName
            .Cells(4, 2) = Trim(Str(delT))
            .Cells(5, 2) = Trim(Str(Rcore))
            .Cells(6, 2) = Trim(Str(Dwire))
            .Cells(7, 2) = Trim(Str(Nturns))
            .Cells(8, 2) = Trim(Str(Nlayers))
            .Cells(9, 2) = Trim(Str(Hcoil))
            .Cells(10, 2) = Trim(Str(FnumZ))
            .Cells(11, 2) = Trim(Str(FleftZ))
            .Cells(12, 2) = Trim(Str(FrightZ))
            .Cells(13, 2) = Trim(Str((FleftZ - FrightZ) / (FnumZ - 1)))
            .Cells(14, 2) = Trim(Str(Rcoil))
            .Cells(15, 2) = Trim(Str(Lcoil))
            .Cells(16, 2) = Trim(Str(RPerM))
            .Cells(17, 2) = Trim(Str(Rslug))
            .Cells(18, 2) = Trim(Str(Hslug))
            .Cells(19, 2) = Trim(Str(RHOslug))
            .Cells(20, 2) = Trim(Str(M))
            .Cells(21, 2) = Trim(Str(C))
            .Cells(22, 2) = Trim(Str(VC0))
            .Cells(23, 2) = Trim(Str(0.5 * C * VC0 * VC0))
            .Cells(24, 2) = Trim(Str(QC0))
            .Cells(25, 2) = Trim(Str(Resr))
            .Cells(26, 2) = Trim(Str(Rsw))
            .Cells(27, 2) = Trim(Str(R))
        End With
        ' Set the column index in the spreadsheet.
        Dim ColNum As Int32
        ColNum = 1
        ' Main loop to step through different starting positions.
        For J As Int32 = 1 To 25 Step 1
            ' Increment the slug's starting position 1 mm to the left.
            Z0 = Z0 + 0.001
            ' Execute the run.
            IntegrateOneRun(False, objExcelWS, 0)
            ' Write the results to the Excel file.
            ColNum = ColNum + 1
            With objExcelWS
                .Cells(29, ColNum) = Trim(Str(Z0))
                .Cells(30, ColNum) = Trim(Str(ESend))
                .Cells(31, ColNum) = Trim(Str(Send))
                .Cells(32, ColNum) = Trim(Str(ECend + ELend))
            End With
        Next J
        ' Re-integrate from the best starting position and,
        ' this time, show the details of the run.
        If (Nturns = 96) Then
            Z0 = 0.309
            IntegrateOneRun(True, objExcelWS, 35)
        End If
        ' Save and close the output file.
        objExcelWB.Save()
        objExcelWB.Close()
        objExcel.Quit()
    Next I
    MsgBox("All done.")
End Sub
```

```
'////////////////////////////////
'// Controls for groupboxNewField

Private labelRcore As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 10), _
     .Text = "Rcore", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbRcore As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 10), _
     .Text = Trim(Str(Rcore)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelDwire As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 35), _
     .Text = "Dwire", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbDwire As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 35), _
     .Text = Trim(Str(Dwire)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelNturns As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 60), _
     .Text = "Nturns", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbNturns As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 60), _
     .Text = Trim(Str(Nturns)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelNlayers As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 85), _
     .Text = "Nlayers", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbNlayers As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 85), _
     .Text = Trim(Str(Nlayers)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelFnumZ As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 110), _
     .Text = "FnumZ", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbFnumZ As New Windows.Forms.TextBox With _
```

```
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 110), _
     .Text = Trim(Str(FnumZ)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelFleftZ As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 135), _
     .Text = "FleftZ", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbFleftZ As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 135), _
     .Text = Trim(Str(FleftZ)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelFrightZ As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 160), _
     .Text = "FrightZ", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbFrightZ As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 160), _
     .Text = Trim(Str(FrightZ)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelFnumR As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 185), _
     .Text = "FnumR", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbFnumR As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 185), _
     .Text = Trim(Str(FnumR)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelFinnerR As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 210), _
     .Text = "FinnerR", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbFinnerR As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 210), _
     .Text = Trim(Str(FinnerR)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelFouterR As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(105, 20), _
     .Location = New Drawing.Point(5, 235), _
     .Text = "FouterR", .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}
```

```
Private tbFouterR As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(85, 20), _
     .Location = New Drawing.Point(110, 235), _
     .Text = Trim(Str(FouterR)), .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private labelOutputFieldFileName As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(190, 20), _
     .Location = New Drawing.Point(5, 260), _
     .Text = "File name for raw field data:", _
     .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbCalculateNewField}

Private tbOutputFieldFileName As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(190, 20), _
     .Location = New Drawing.Point(5, 285), _
     .Text = FieldTextFileName_Default, _
     .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbCalculateNewField}

Private WithEvents buttonNewFieldGo As New Windows.Forms.Button With _
    {.Size = New Drawing.Size(190, 30), _
     .Location = New Drawing.Point(5, 310), _
     .Text = "Execute", .TextAlign = ContentAlignment.MiddleCenter, _
     .Parent = gbCalculateNewField}

'////////////////////////////////
'// Controls for groupboxPlotField

Private labelInputFieldFileName As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(190, 20), _
     .Location = New Drawing.Point(5, 10), _
     .Text = "File name for raw field data:", _
     .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbPlotField}

Private tbInputFieldFileName As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(190, 20), _
     .Location = New Drawing.Point(5, 35), _
     .Text = FieldTextFileName_Default, _
     .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbPlotField}

Private labelThinDiskExcelFileName As New Windows.Forms.Label With _
    {.Size = New Drawing.Size(190, 20), _
     .Location = New Drawing.Point(5, 60), _
     .Text = "File name for thin disk forces:", _
     .TextAlign = ContentAlignment.MiddleLeft, _
     .Parent = gbPlotField}

Private tbThinDiskExcelFileName As New Windows.Forms.TextBox With _
    {.Size = New Drawing.Size(190, 20), _
     .Location = New Drawing.Point(5, 85), _
     .Text = ForceOnThinDiskPlotFileName_Default, _
     .TextAlign = HorizontalAlignment.Left, _
     .Parent = gbPlotField}
```

```
    Private labelSlugExcelFileName As New Windows.Forms.Label With _
        {.Size = New Drawing.Size(190, 20), _
         .Location = New Drawing.Point(5, 110), _
         .Text = "File name for slug forces:", _
         .TextAlign = ContentAlignment.MiddleLeft, _
         .Parent = gbPlotField}

    Private tbSlugExcelFileName As New Windows.Forms.TextBox With _
        {.Size = New Drawing.Size(190, 20), _
         .Location = New Drawing.Point(5, 135), _
         .Text = ForceOnSlugPlotFileName_Default, _
         .TextAlign = HorizontalAlignment.Left, _
         .Parent = gbPlotField}

    Private WithEvents buttonPlotFieldGo As New Windows.Forms.Button With _
        {.Size = New Drawing.Size(190, 30), _
         .Location = New Drawing.Point(5, 160), _
         .Text = "Execute", .TextAlign = ContentAlignment.MiddleCenter, _
         .Parent = gbPlotField}

End Class
```