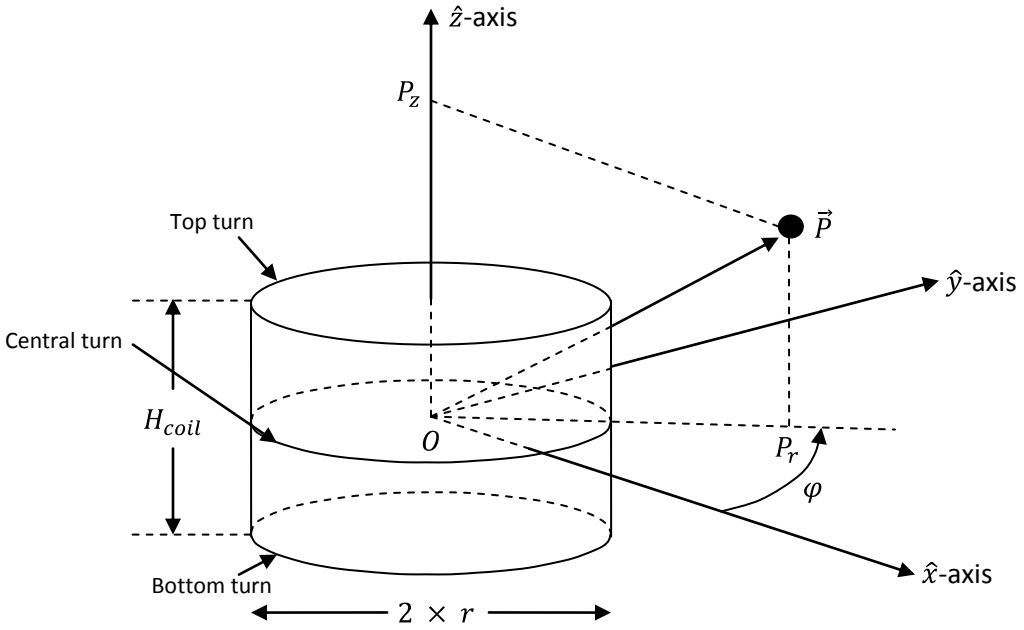
The magnetic field in and around a finite cylindrical air-core solenoid

Consider a solenoid in the shape of a circular cylinder. Let its length and inner radius be represented by H_{coil} and R_{core}, respectively. Let the number of turns of wire per layer and the number of layers be represented by N_{turns} and N_{layers}, respectively. A steady current of I amperes flows through the winding. I will often refer to the solenoid as the "coil".

We will make only two material idealizations. Firstly, we will assume that the direct current flows uniformly through the cross-sectional area of the wire. Secondly, we will assume that each turn of the winding lies entirely in a plane perpendicular to the longitudinal axis of the coil. In other words, the helical "tilt" of practical windings will be ignored. In practice, winding a coil with an even number of layers from one end of the coil and back again will cancel out most of the effect of the tilt.

Let us calculate the magnetic field at some point P which is an axial distance P_z from the center of the coil and a radial distance P_r away from the longitudinal axis. Point P can be in the interior of the coil, or outside of it, but not in the winding itself.

It is helpful to impose a Euclidian co-ordinate frame with its origin O located at the geometric center of the coil. The $\hat{z}$-axis is set coincident with the longitudinal axis of the coil and taken to be "positive" when pointing more or less in the direction of point P. The $\hat{x}$-axis can always be chosen so that the plane formed by the $\hat{z}$- and $\hat{x}$-axes includes point P. But, let us not make that choice. It is sometimes handy to be able to calculate the magnetic field at any point using a single co-ordinate frame. Instead, let us assume that the ray $\vec{P}$ from the origin to point P falls in a plane rotated around the $\hat{z}$-axis by angle φ in the right-hand direction. The situation is therefore as shown in the following diagram. Note that I have shown the coil with its longitudinal axis vertical. I have also labeled the central turn of wire and the turns at both ends.



The radius of the coil is shown in the figure using the symbol r and not the symbol R_{core}. R_{core} was described above as the inner radius of the coil, like the outside of the cylindrical form on whigh the wire was wound. But, the wire itself has non-zero thickness so the exact radius r of any point <u>in</u> the winding will depend on which layer of wire is being examined. To explore this matter further, let us look at one

particular turn of the winding. Let us look at the N^{th} turn from the bottom of the coil in the M^{th} layer of the winding.

Let D_{wire} be the outside diameter of the wire with which the coil is wound. Since there are N_{turns} of wire along the coil's length, which we will assume to be wound close one-to-the-next, we can relate the length of the coil to the diameter of the wire by $H_{coil} = N_{turns} \times D_{wire}$. Therefore, the center of the wires in the N^{th} turn from the bottom of the coil has a z-co-ordinate equal to:

$$z_{Nth\ turn} = \left[N - \tfrac{1}{2} - \left({}^{N_{turns}}/_{2}\right)\right] \times D_{wire} \quad (1)$$

Just to be certain, the z-co-ordinates of the bottom and top turns are equal to:

$$\begin{aligned} z_{bottom\ turn} &= \left[1 - \tfrac{1}{2} - \left({}^{N_{turns}}/_{2}\right)\right] \times D_{wire} = -\tfrac{1}{2}(N_{turns} - 1) \times D_{wire} \\ z_{top\ turn} &= \left[N - \tfrac{1}{2} - \left({}^{N_{turns}}/_{2}\right)\right] \times D_{wire} = +\tfrac{1}{2}(N_{turns} - 1) \times D_{wire} \end{aligned}$$

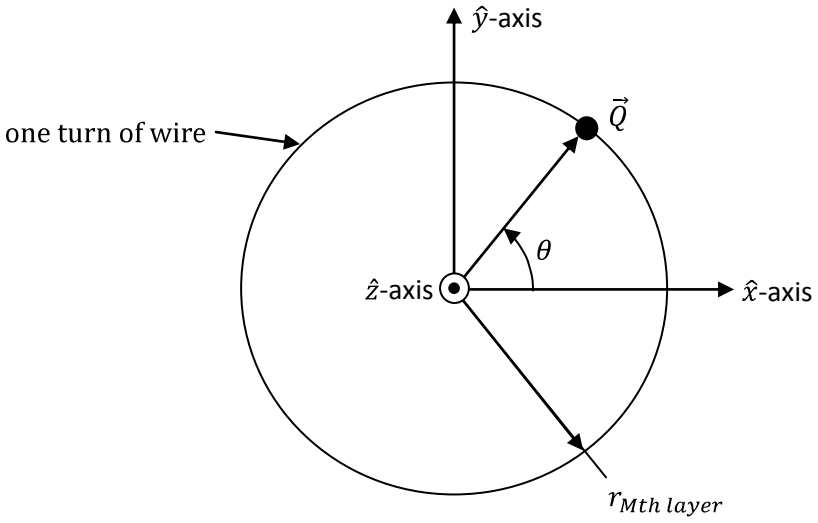
Some quick mental arithmetic will confirm that these are the correct distances whether N_{turns} is odd or even.

In a similar way, one can see that the distance from the longitudinal axis of the coil to the center of the wires in the M^{th} layer is equal to:

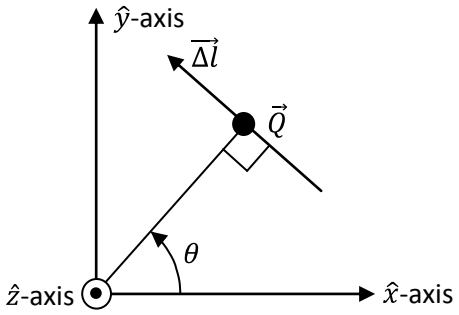
$$r_{Mth\ layer} = R_{coil} + \left[\left(M - \frac{1}{2}\right) \times D_{wire}\right] \quad (2)$$

The vertical distance $z_{Nth\ turn}$ is the same for the N^{th} turn in each layer. Similarly, the radial distance $r_{Mth\ layer}$ is the same for all turns in the M^{th} layer.

Now, let us turn our attention to some point Q located on the N^{th} turn of wire (the descriptor "from the bottom of the coil" will be dropped from now on) in the M^{th} layer of the winding. Point Q can be located anywhere "around" this turn of wire. We can specify exactly where point Q is located using the azimuth angle θ by which the ray $\vec{Q}$ from the origin to point Q is rotated away from the $\hat{z}$-$\hat{x}$ plane. The view of this turn, as seen from above, is shown in the following figure.



Consider now a very small piece of the wire at point Q, short enough to be considered straight. If the piece you have in mind still has noticeable curvature, then pick a piece one-hundredth as big. Let the length of this small piece of wire be represented by Δl. Of course, it will be tangent to the circle which passes through point Q. Accordingly, we can use a vector $\overrightarrow{\Delta l}$ to capture both the magnitude and direction of the piece of wire. The "positive" direction of $\overrightarrow{\Delta l}$ will be established using the right-hand-rule applied to the $\hat{z}$-axis, which points up out of the page in the figure above. The following figure shows the geometry of $\overrightarrow{\Delta l}$, as seen from above.



Because it is tangent to the winding, $\overrightarrow{\Delta l}$ will be perpendicular to the ray from the longitudinal axis (the $\hat{z}$-axis) to point Q. For convenience, we will select $\overrightarrow{\Delta l}$ so that point Q lies at its geometric midpoint. The components of $\overrightarrow{\Delta l}$ along the $\hat{x}$- and $\hat{y}$-axes are given by $\sin\theta$ and $\cos\theta$, respectively, so the vector $\overrightarrow{\Delta l}$ can be written as:

$$\overrightarrow{\Delta l} = -\Delta l \sin\theta\, \hat{x} + \Delta l \cos\theta\, \hat{y}$$

We can relate length Δl to angle θ, or rather to a change in the angle $\Delta\theta$, in the following way. As seen from the longitudinal axis of the coil, line segment $\overline{\Delta l}$ subtends a small angle, which we will represent by $\Delta\theta$. We know that the radius from the longitudinal axis to point Q is $r_{Mth\ layer}$ so, using the definition of the tangent function, we have:

$$\tan\left(\frac{\Delta\theta}{2}\right) = \frac{\text{opposite side}}{\text{adjacent side}} = \frac{\left(\Delta l/2\right)}{r_{Mth\ layer}}$$

In the limit, as Δl gets very small, the tangent function approaches its argument, so this equation simplifies to:

$$\Delta\theta = \frac{\Delta l}{r_{Mth\ layer}}$$

This lets us write vector $\overrightarrow{\Delta l}$ as:

$$\overrightarrow{\Delta l} = -r_{Mth\ layer}\Delta\theta \sin\theta\, \hat{x} + r_{Mth\ layer}\Delta\theta \cos\theta\, \hat{y} \quad (3)$$

Now, let us look again at this very small piece of wire $\overrightarrow{\Delta l}$ and imagine that a direct current of I amperes is flowing along its length, in the direction of the vector. The tiny vector $I\overrightarrow{\Delta l}$ is called a "current element". Its units are ampere-meters.

We now have all the parts which we need to apply Biot-Savart's Law. The experiments first done by Messrs. Biot and Savart, and confirmed by many others since, showed that a current element such as our $I\overrightarrow{\Delta l}$ produces a magnetic field, which we will represent by $\overrightarrow{\Delta B}$, at some point removed by vector $\vec{S}$ from the center of the current element, which is given by:

$$\overrightarrow{\Delta B} = \frac{\mu\, I\overrightarrow{\Delta l} \overrightarrow{\times} \vec{S}}{4\pi S^3} \quad (4)$$

There are six things to note about this expression.

1. Do not confuse $\vec{P}$ and $\vec{S}$. $\vec{P}$ is the ray from the geometric center of the solenoid to the spot at which we want to calculate the magnetic field. $\vec{S}$ points to the same spot, but it starts at the center of current element $I\overrightarrow{\Delta l}$.
2. The multiplication $\overrightarrow{\times}$ in the numerator is a vector cross-product. The cross-product of two vectors, say, $\overrightarrow{\Delta l} \overrightarrow{\times} \vec{S}$ is a third vector, which is perpendicular to both $\overrightarrow{\Delta l}$ and $\vec{S}$ and whose magnitude is the scalar product of their two lengths and the sine of the angle between them.
3. $\overrightarrow{\Delta B}$ is the entire magnetic field produced by this current element at point $\vec{S}$. The symbol $\overrightarrow{\Delta B}$, which represents that field, is shown with the difference "Δ" because we will (below) add up the fields from a lot of such current elements, of which this is only one.
4. Distance S is the length of vector $\vec{S}$.
5. Biot-Savart's equation is often written using the ratio $\hat{S}/S^2$ instead of $\vec{S}/S^3$, where $\hat{S}$ is a vector with unit length pointing in the direction of $\vec{S}$. The two representations are identical.
6. μ is the permeability of the medium at the point of interest $\vec{P}$. In our case, the medium is air. Free space is a good approximation for air (it works both ways, right?) and the permeability of free space, which is usually represented by the symbol μ_0, has the value $4\pi 10^{-7}$ T/Am.

We need to re-arrange a couple of things before we can substitute our symbols into Biot-Savart's equation. In their equation, vector $\vec{S}$ points from the mid-point of the current element (which is our point Q) to the point of interest (which is our point P). We can calculate vector $\vec{S}$ using straightforward vector subtraction, as follows:

$$\vec{S} = \vec{P} - \vec{Q}$$

In our co-ordinate frame,

$$\vec{P} = P_r \cos\varphi\, \hat{x} + P_r \sin\varphi\, \hat{y} + P_z \hat{z}$$

and

$$\vec{Q} = r_{Mth\ layer} \cos\theta\, \hat{x} + r_{Mth\ layer} \sin\theta\, \hat{y} + z_{Nth\ turn} \hat{z}$$

so

$$\vec{S} = \left(P_r \cos\varphi - r_{Mth\ layer} \cos\theta\right)\hat{x} + \left(P_r \sin\varphi - r_{Mth\ layer} \sin\theta\right)\hat{y} + (P_z - z_{Nth\ turn})\hat{z} \quad (5)$$

We have already expressed $\overrightarrow{\Delta l}$ in a suitable form in Equation (3), as:

$$\overrightarrow{\Delta l} = -r_{Mth\ layer}\Delta\theta \sin\theta\, \hat{x} + r_{Mth\ layer}\Delta\theta \cos\theta\, \hat{y} \quad (3)$$

The vector cross-product of $\overrightarrow{\Delta l}$ and $\vec{S}$ can be represented as the following determinant:

$$\overrightarrow{\Delta l} \times \vec{S} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ -r_{Mth\ layer}\Delta\theta \sin\theta & r_{Mth\ layer}\Delta\theta \cos\theta & 0 \\ P_r \cos\varphi - r_{Mth\ layer}\cos\theta & P_r \sin\varphi - r_{Mth\ layer}\sin\theta & P_z - z_{Nth\ turn} \end{vmatrix}$$

which expands out to:

$$\overrightarrow{\Delta l} \times \vec{S} = r_{Mth\ layer}\Delta\theta \begin{Bmatrix} (P_z - z_{Nth\ turn})\cos\theta\, \hat{x} + \cdots \\ (P_z - z_{Nth\ turn})\sin\theta\, \hat{y} + \cdots \\ \left[-\sin\theta\left(P_r \sin\varphi - r_{Mth\ layer}\sin\theta\right) - \cos\theta\left(P_r\cos\varphi - r_{Mth\ layer}\cos\theta\right)\right]\hat{z} \end{Bmatrix}$$

The z-component can be expanded and then simplified using two trigonometric identities. Using the shorthand s_x for $\sin x$ and c_x for $\cos x$, we can write the term in square brackets as:

$$\begin{aligned} -s_\theta\left(P_r s_\varphi - r_{Mth\ layer}s_\theta\right) - c_\theta\left(P_r c_\varphi - r_{Mth\ layer}c_\theta\right) &= \cdots \\ = -P_r s_\theta s_\varphi + r_{Mth\ layer}s_\theta^2 - P_r c_\theta c_\varphi + r_{Mth\ layer}c_\theta^2 & \\ = -P_r\left(s_\theta s_\varphi + c_\theta c_\varphi\right) + r_{Mth\ layer}\left(s_\theta^2 + c_\theta^2\right) & \end{aligned}$$

The two trigonometric identities are $s_\theta^2 + c_\theta^2 = 1$ and $s_\theta s_\varphi + c_\theta c_\varphi = \cos(\theta - \varphi)$. Remember that cosine is a symmetric function, so it does not matter if we use $\cos(\theta - \varphi)$ or $\cos(\varphi - \theta)$. Substituting these identities, we get:

$$\overrightarrow{\Delta l} \times \vec{S} = r_{Mth\ layer}\Delta\theta \begin{Bmatrix} (P_z - z_{Nth\ turn})\cos\theta\, \hat{x} + \cdots \\ (P_z - z_{Nth\ turn})\sin\theta\, \hat{y} + \cdots \\ \left[-P_r\cos(\theta - \varphi) + r_{Mth\ layer}\right]\hat{z} \end{Bmatrix} \quad (6)$$

We also need to calculate S^3 in order to use Biot-Savart's equation. Since the square of the length of a vector is the sum of the squares of its components, $S^2 = S_x^2 + S_y^2 + S_z^2$, we can write down by inspection:

$$S^2 = \begin{Bmatrix} P_r^2\cos^2\varphi - 2P_r r_{Mth\ layer}\cos\varphi\cos\theta + r_{Mth\ layer}^2\cos^2\theta + \cdots \\ P_r^2\sin^2\varphi - 2P_r r_{Mth\ layer}\sin\varphi\sin\theta + r_{Mth\ layer}^2\sin^2\theta + \cdots \\ P_z^2 - 2P_z z_{Nth\ turn} + z_{Nth\ turn}^2 \end{Bmatrix}$$

This can be simplified using the two trigonometric identities once more, to:

$$S^2 = P_r^2 + P_z^2 + r_{Mth\ layer}^2 + z_{Nth\ turn}^2 - 2\left[P_r r_{Mth\ layer}\cos(\theta - \varphi) + P_z z_{Nth\ turn}\right]$$

In the case of S^2, there is one further simplification we can make. Three of its terms are the square of a sum. Sorting that out leaves:

$$S^2 = P_r^2 + (P_z - z_{Nth\ turn})^2 + r_{Mth\ layer}^2 - 2P_r r_{Mth\ layer}\cos(\theta - \varphi) \quad (7)$$

Now, we are ready to use the Biot-Savart equation. Substitution Equations (6) and (7) gives:

$$\overrightarrow{\Delta B} = \left(\frac{\mu\, I r_{Mth\ layer}\Delta\theta}{4\pi}\right)\frac{(P_z - z_{Nth\ turn})c_\theta\hat{x} + (P_z - z_{Nth\ turn})s_\theta\hat{y} - \left[P_r\cos(\theta-\varphi) - r_{Mth\ layer}\right]\hat{z}}{\left[P_r^2 + (P_z - z_{Nth\ turn})^2 + r_{Mth\ layer}^2 - 2P_r r_{Mth\ layer}\cos(\theta-\varphi)\right]^{3/2}}$$

As explained above, $\overrightarrow{\Delta B}$ is the total magnetic field at point P generated by the direct current flowing through the small piece of wire whose geometric center is at point Q and whose centerline is $\overrightarrow{\Delta l}$. The total magnetic field set up by the magnet as a whole at point P can be calculated by adding up the contributions from all the small pieces of wire which comprise the N^{th} turn of the M^{th} layer, and then adding up the similar contributions from all of the $N_{turns} \times M_{layers}$ turns. Doing these sums gives the three components of the magnetic field at point P as:

Equations to calculate the magnetic field at point *P*

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

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

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

There are three things to note about these equations.

1. The outer summation over the layers in the winding (using counter m) and the middle summation over the turns in the layers (using counter n) have the physical meanings already described. The third, inner, summation has not yet been completely described. The intention, of course, is to divide the circle representing one turn of wire into small arcs, each arc having a subtended angle of $\Delta\theta$, and then to add up the contributions to the magnetic field from all the arcs around the circle, from angle $\theta = 0$ to angle $\theta = 2\pi$ radians = 360^{o}. How many arcs the circle should be divided into is up to you and the speed of your computer. My experience has been that 1000 arcs is sufficient – the increased accuracy from using more arcs is minor.

2. There is a symmetry between the first two components, B_x and B_y, there being a $\cos\theta$ in the former and a $\sin\theta$ in the latter. The symmetry has a physical implication and a computational implication. If each term in the summation for B_x is compared with the corresponding term in the summation for B_y, the B_y term divided by the B_x term will always be equal to $\sin\theta/\cos\theta$, or $\tan\theta$. Since the azimuth angle for that particular term is θ, it follows that the resultant of the B_x and B_y components has the same direction as the ray from the solenoid's centerline (the $\hat{z}$-axis) to the current element. Physically, this means that the magnetic field will be radially symmetric around the centerline of the coil. In each plane containing the $\hat{z}$-axis, the magnetic field will appear the same. The computational implication is this. In some applications, it may be enough to do the summations for a single value of φ, say, $\varphi = 0$. $\varphi = 0$ describes a radial plane (that is,

it contains the $\hat{z}$-axis) and B_y will be zero everywhere in it. The B_x components calculated for this plane are, in fact, the radial components of the field. B_x and B_y can then be found for other values of φ by multiplying the appropriate radial component by the appropriate $\sin\varphi$ and $\cos\varphi$ factors.

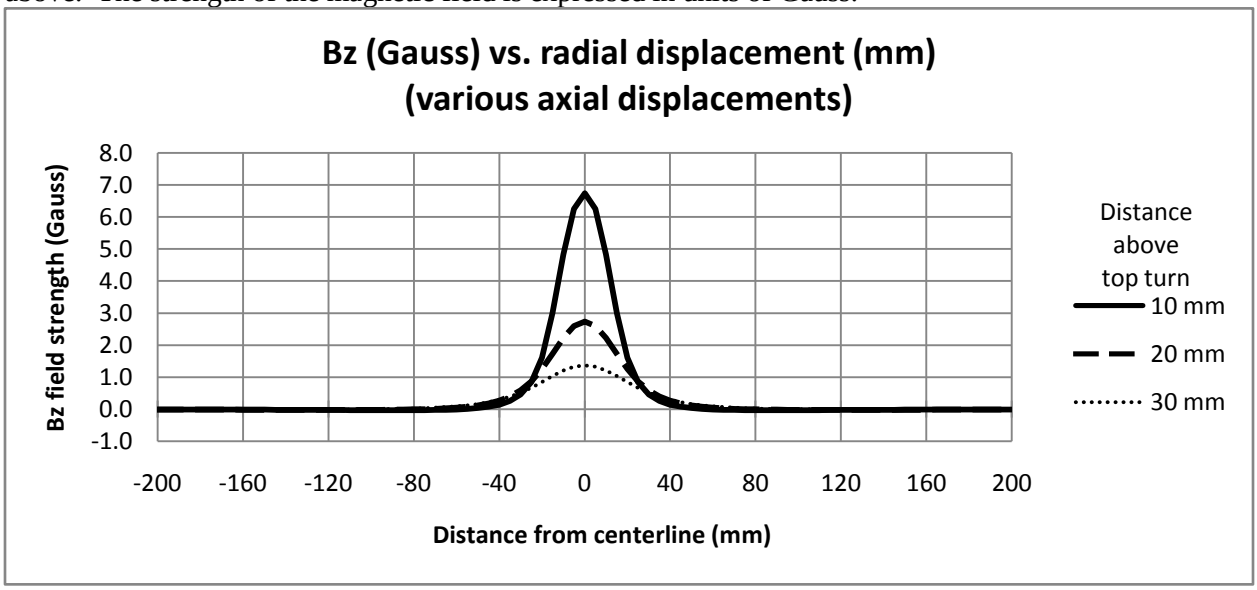
3. In the days before computers, much effort was spent trying to obtain closed-form solutions for these equations. Imagining thinner and thinner wires, so that N_{layers} becomes extremely large, enables the middle summation to be approximated by an integral over the axial length of the coil. Imagining thinner wires also enables either of two assumptions: (i) that the winding can be replaced by a thin "sheet" of current, removing the outermost summation entirely, or (ii) that the winding can be modeled by a torus with a rectangular cross-section and a uniform current density, so the outermost summation can be approximated by an integral operating radially from the inner radius to the outer radius of the coil. Similarly, the innermost summation can be approximated by an integral around the circle. Only a very few special geometries surrendered to this effort. As always, though, it is useful to benchmark your numerical procedure against one or more of those special geometries.

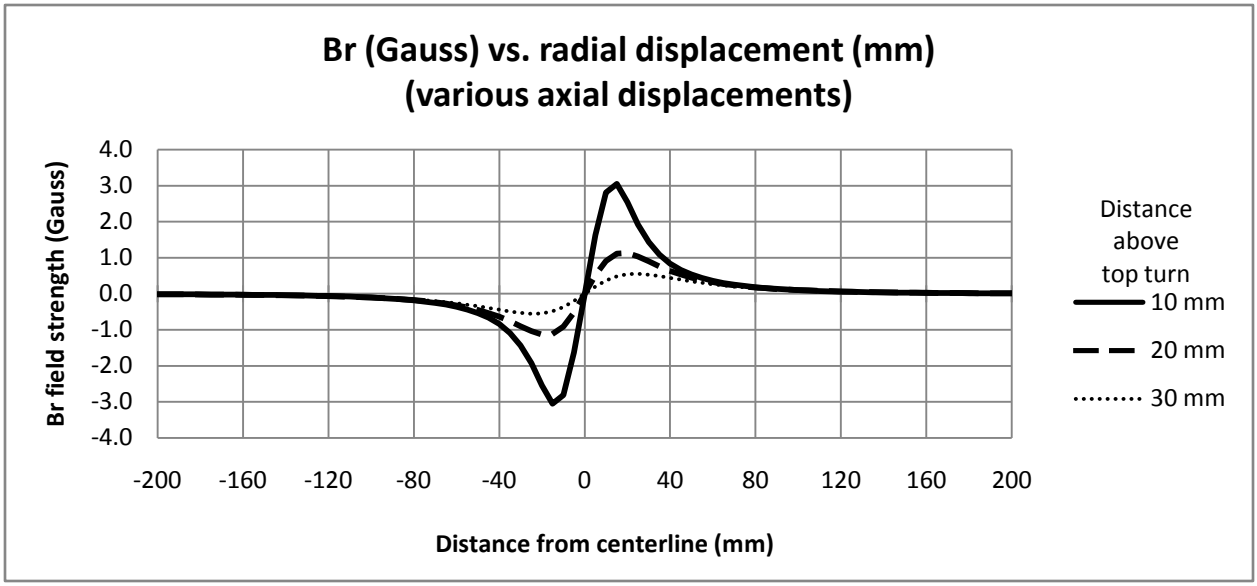
(Sample calculation)

Let us wind a coil using #28 gauge enameled copper wire. This wire has an outside diameter, including insulation, of 0.349mm and a resistance of $0.217\Omega/\text{m}$. A coil 100mm long will just accommodate 287 turns. Let the inner radius of the winding be 10mm and wind 16 layers. Finally, let us power the coil with 5V dc.

The winding is 16 layers, or 5.584mm thick, so the average radius of all the turns is 12.792mm. The length of wire in the coil is the total number of turns multiplied by the average circumference, or $287 \times 16 \times 2 \times \pi \times r_{average}$, or 369.1m. The resistance of the coil as a whole is $369.1\text{m} \times 0.217\Omega/\text{m}$, or 80.1Ω. Ohm's Law gives the current through the coil as $I = V/R = 5\text{V}/80.1\,\Omega = 62.4\text{mA}$.

I have calculated the B_r and B_z components of the magnetic field at certain points above the top turn of the solenoid. These components are shown in the following two graphs. Each component is shown for points from 200mm on one side of the centerline to 200mm on the other side of the centerline. The field strength is shown for three elevations: 5mm above the top turn of the magnet, 15mm above and 25mm above. The strength of the magnetic field is expressed in units of Gauss.





The algebraic signs of the components of the magnetic field are as expected and are consistent with the right-hand rule. The B_z component is positive, meaning that it "points" upwards in the region above the solenoid. The B_r component is positive at points to the right of the centerline, where it "points" away from the centerline, and it is negative to the left of the centerline, where it also "points" away from the centerline. Of course, both components show symmetry about the centerline.

To get some feeling for the horizontal dimension, note that the inner diameter of the coil is 20mm, one-half of the spacing of the vertical gridlines in the graphs. It is apparent that the B_r component has its maximum values right above the wires in the winding. The B_z component has its maximum value right on the centerline.

To get some feeling for the vertical scale, note that the strength of the Earth's magnetic field is about one-half Gauss at mid-latitudes.

The subroutine used to calculate these values is listed below.

(An aside for a benchmark solution)

I mentioned above the importance of benchmarking your numerical procedure against a known solution. That is so important that it is useful to have ready at hand such a known solution. One obvious point to use for this purpose is the geometric center of the coil. That is the point where we placed the origin of our co-ordinate system. Let us calculate the magnetic field here. At this handy spot, both P_r and P_z are zero so the three summations reduce to:

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

$$B_y = \frac{\mu}{4\pi} \sum_{m=1}^{m=N_{layers}} \left[\sum_{n=1}^{n=N_{turns}} \left\{ \sum_{\theta=0}^{\theta=2\pi} \frac{-I r_{Mth\ layer} z_{Nth\ turn} \Delta\theta \sin\theta}{\left(z_{Nth\ turn}^2 + r_{Mth\ layer}^2\right)^{3/2}} \right\} \right]$$

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

Before swimming into deeper waters, look at the terms in the innermost summation. The innermost summation is a summation over angles, during which $r_{Mth\ layer}$, $z_{Nth\ turn}$ and I are constant. These constant factors can be taken outside the summation. The B_x term, for example, can be re-written as:

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

It can be seen by inspection that the innermost summation vanishes. For every value of θ, there is a corresponding value on the "other" side of the circle, having a negative cosine with the same magnitude. Take small enough arcs, and the innermost summation becomes the integral of $\cos\theta$ around a circle. In a similar way, B_y also vanishes. From a physical point-of-view, this means that the magnetic field has no radial component at the center of the coil. This is true whether the coil is long or short compared with its radius. It is true whether the wire is fat or skinny. (The only assumption still in effect is that each turn of wire falls wholly in a plane perpendicular to the longitudinal axis.)

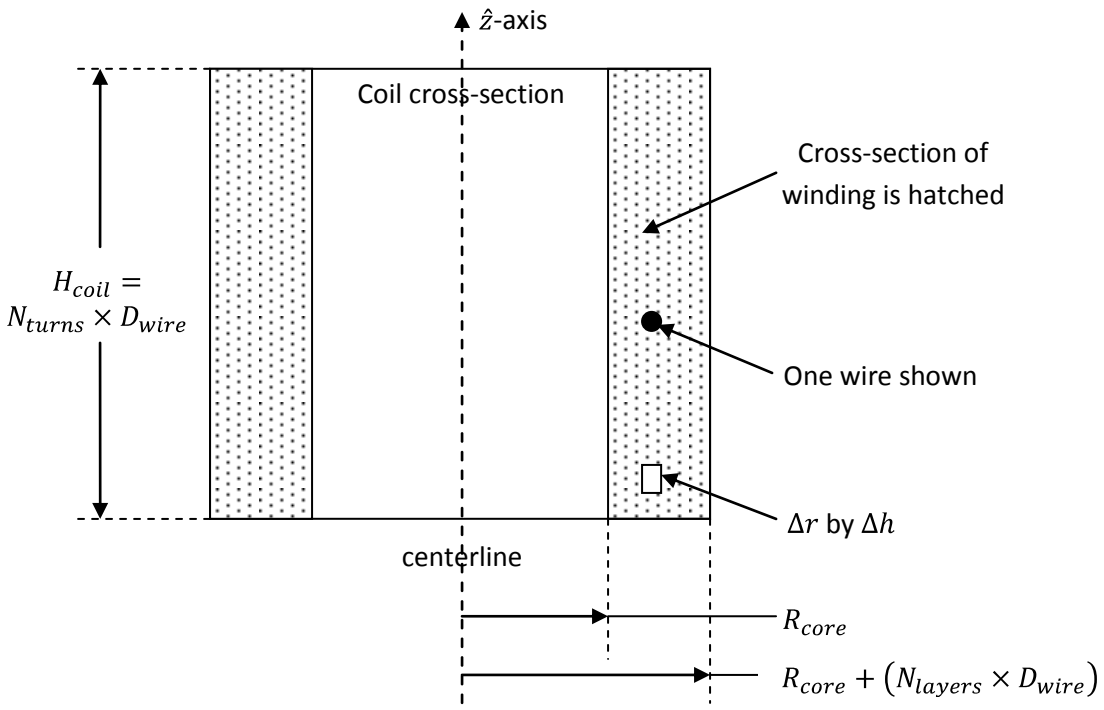
The B_z term does not vanish, but does become easier to deal with. Removing constant factors from the innermost summation for B_z gives:

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

The sum of the angles of arc around a circle is 2π, so this simplifies immediately to:

$$B_z = \frac{\mu}{2} \sum_{m=1}^{m=N_{layers}} \left[\sum_{n=1}^{n=N_{turns}} \frac{I r_{Mth\ layer}^2}{\left(z_{Nth\ turn}^2 + r_{Mth\ layer}^2\right)^{3/2}} \right]$$

Up until now, we have assumed that the current I flows along the centerlines of $N_{turns} \times N_{layers}$ discrete circular turns. What we are going to do now is to "smooth out" that total current uniformly over the cross-sectional area of the winding. A cross-section of the coil is shown in the following figure.



The cross-sectional area of the winding is $A_{winding} = (N_{turns} \times D_{wire}) \times (N_{layers} \times D_{wire})$. The total current flowing through this area (into the page on one side and out of the page on the other) is $I \times N_{turns} \times N_{layers}$. We can define the current density i as the current flowing per unit area of the cross-section. Then:

$$i = \frac{I \times N_{turns} \times N_{layers}}{(N_{turns} \times D_{wire}) \times (N_{layers} \times D_{wire})} = \frac{I}{D_{wire}^2}$$

So, the current flowing through a small rectangle Δr wide and Δh high, as shown in the figure, would be equal to $i\Delta r\Delta h = I\Delta r\Delta h / D_{wire}^2$.

Now, we can replace the summation over layers with a summation over bits of Δr from the inner radius of the core R_{core} to the outer radius of the winding $R_{core} + (N_{layers} \times D_{wire})$. We can replace the summation over turns with a summation over bits of Δh from the bottom turn of the coil, at $z = -H_{coil}/2$, to the top turn of the coil, at $z = +H_{coil}/2$. And, of course, the current flowing through each rectangular bit is $I\Delta r\Delta h / D_{wire}^2$. Making these substitutions gives the following expression for B_z:

$$B_z = \frac{\mu}{2} \sum_{r=R_{core}}^{r=R_{core}+(N_{layers} \times D_{wire})} \left[\sum_{h=-H_{coil}/2}^{h=+H_{coil}/2} \frac{(I\Delta r\Delta h / D_{wire}^2) r^2}{(h^2 + r^2)^{3/2}} \right]$$

In the limit, as we pick Δr and Δh extremely small, we can replace the summations with integrals.

$$B_z = \frac{\mu}{2} \int_{r=R_{core}}^{r=R_{core}+(N_{layers} \times D_{wire})} \left[\int_{h=-H_{coil}/2}^{h=+H_{coil}/2} \frac{(I/D_{wire}^2) r^2}{(h^2 + r^2)^{3/2}} dr dh \right]$$

Withdrawing constants, and dividing both the numerator and the denominator by r^3, we can re-write this expression as:

$$B_z = \frac{\mu I}{2D_{wire}^2} \int_{r=R_{core}}^{r=R_{core}+(N_{layers}\times D_{wire})} \left[\int_{h=-H_{coil}/2}^{h=+H_{coil}/2} \frac{1}{\left(1+\left(\frac{h}{r}\right)^2\right)^{3/2}} d\left(\frac{h}{r}\right) \right] dr$$

We divided through by r^3/r^3 so that the integral in the square brackets would appear in a more digestible form. In indefinite form, it is:

$$\int \frac{1}{(1+x^2)^{3/2}} dx$$

This can be integrated easily, starting by adding and subtracting x^2 to the numerator, and then using the product rule:

$$\begin{aligned}
\int \frac{1}{(1+x^2)^{3/2}} dx &= \int \frac{1+x^2-x^2}{(1+x^2)^{3/2}} dx + C \\
&= \int \left[\frac{1+x^2}{(1+x^2)^{3/2}} - \frac{x^2}{(1+x^2)^{3/2}} \right] dx + C \\
&= \int \left[\frac{1}{(1+x^2)^{1/2}} + \frac{x\left(-\frac{1}{2}\right)(2x)}{(1+x^2)^{3/2}} \right] dx + C \\
&= \int \left[\frac{dx}{(1+x^2)^{1/2}} + \frac{x\left(-\frac{1}{2}\right)d(1+x^2)}{(1+x^2)^{3/2}} \right] + C \\
&= \int \left\{ \frac{dx}{(1+x^2)^{1/2}} + xd\left[\frac{1}{(1+x^2)^{1/2}} \right] \right\} + C \\
&= \int d\left[\frac{x}{(1+x^2)^{1/2}} \right] + C \\
&= \frac{x}{(1+x^2)^{1/2}} + C
\end{aligned}$$

Applying this result to the expression for B_z, we get:

$$B_z = \frac{\mu I}{2D_{wire}^2} \int_{r=R_{core}}^{r=R_{core}+(N_{layers}\times D_{wire})} \left[\frac{\left(\frac{h}{r}\right)}{\left(1+\left(\frac{h}{r}\right)^2\right)^{1/2}} \Bigg|_{h=-H_{coil}/2}^{h=+H_{coil}/2} \right] dr$$

The integrand is symmetric about $h = 0$, so we can evaluate at the endpoints to obtain:

$$B_z = \frac{\mu I}{2D_{wire}^2} \int_{r=R_{core}}^{r=R_{core}+\left(N_{layers}\times D_{wire}\right)} \frac{2\left(\frac{H_{coil}}{2r}\right)}{\left[1+\left(\frac{H_{coil}}{2r}\right)^2\right]^{1/2}} dr$$

Now, making a few adjustments to clarify the dependence on the remaining variable r, we get:

$$B_z = \frac{\mu I H_{coil}}{2D_{wire}^2} \int_{r=R_{core}}^{r=R_{core}+\left(N_{layers}\times D_{wire}\right)} \frac{1}{\sqrt{r^2+\frac{1}{4}H_{coil}^2}} dr$$

If, for the moment, we introduce the constant $a = \frac{1}{2}H_{coil}$, then the interesting part can be processed as follows:

$$\begin{aligned}
\int \frac{1}{\sqrt{r^2+a^2}} dr &= \int \left(\frac{r+\sqrt{r^2+a^2}}{r+\sqrt{r^2+a^2}}\right)\left(\frac{1}{\sqrt{r^2+a^2}}\right) dr + C \\
&= \int \left(\frac{1}{r+\sqrt{r^2+a^2}}\right)\left(\frac{r+\sqrt{r^2+a^2}}{\sqrt{r^2+a^2}}\right) dr + C \\
&= \int \left(\frac{1}{r+\sqrt{r^2+a^2}}\right)\left(1+\frac{r}{\sqrt{r^2+a^2}}\right) dr + C \\
&= \int \left(\frac{1}{r+\sqrt{r^2+a^2}}\right)\left(dr+\frac{\left(\frac{1}{2}\right)2rdr}{\sqrt{r^2+a^2}}\right) + C \\
&= \int \left(\frac{1}{r+\sqrt{r^2+a^2}}\right)\left(dr+\frac{\left(\frac{1}{2}\right)dr^2}{\sqrt{r^2+a^2}}\right) + C \\
&= \int \left(\frac{1}{r+\sqrt{r^2+a^2}}\right)\left(dr+d\sqrt{r^2+a^2}\right) + C \\
&= \int \left(\frac{1}{r+\sqrt{r^2+a^2}}\right) d\left(r+\sqrt{r^2+a^2}\right) + C \\
&= \ln\left(r+\sqrt{r^2+a^2}\right) + C
\end{aligned}$$

Applying this result to the equation for B_z, we get:

$$B_z = \frac{\mu I H_{coil}}{2D_{wire}^2}\left[\ln\left(r+\sqrt{r^2+\frac{1}{4}H_{coil}^2}\right)\Bigg|_{r=R_{core}}^{r=R_{core}+\left(N_{layers}\times D_{wire}\right)}\right]$$

Recalling that the difference between two logarithms is the logarithm of their ratio, this can be evaluated as:

$$B_z = \frac{\mu I H_{coil}}{2D_{wire}^2} \ln\left\{\frac{\left[R_{core}+\left(N_{layers}\times D_{wire}\right)\right]+\sqrt{\left[R_{core}+\left(N_{layers}\times D_{wire}\right)\right]^2+\frac{1}{4}H_{coil}^2}}{R_{core}+\sqrt{R_{core}^2+\frac{1}{4}H_{coil}^2}}\right\}$$

This may be unwieldy, but it is a closed-form result. Most often, simplifying assumptions are now made. It is here that the traditional "coil is much longer than its radius" assumption is made. If the coil is much longer than its radius, then the square root terms can be expanded into their Taylor series and the terms greater than the first order ignored. For example:

$$\begin{aligned}\sqrt{R_{coil}^2 + \tfrac{1}{4}H_{coil}^2} &= \frac{1}{2}H_{coil}\sqrt{1 + \frac{4R_{coil}^2}{H_{coil}^2}} \\ &= \frac{1}{2}H_{coil}\left[1 + \frac{1}{2}\left(\frac{4R_{coil}^2}{H_{coil}^2}\right)^1 - \frac{1}{8}\left(\frac{4R_{coil}^2}{H_{coil}^2}\right)^2 + \frac{1}{16}\left(\frac{4R_{coil}^2}{H_{coil}^2}\right)^3 + \cdots\right] \\ &\cong \frac{1}{2}H_{coil}\left[1 + \frac{1}{2}\left(\frac{4R_{coil}^2}{H_{coil}^2}\right)\right]\end{aligned}$$

Applying this approximation to the expression for B_z gives:

$$B_z \cong \frac{\mu I H_{coil}}{2D_{wire}^2}\ln\left\{\frac{\left[R_{core} + \left(N_{layers} \times D_{wire}\right)\right] + \frac{1}{2}H_{coil}\left[1 + \frac{2\left[R_{core} + \left(N_{layers} \times D_{wire}\right)\right]^2}{H_{coil}^2}\right]}{R_{core} + \frac{1}{2}H_{coil}\left[1 + \frac{2R_{core}^2}{H_{coil}^2}\right]}\right\}$$

Extracting $\frac{1}{2}H_{coil}$ from both the numerator and denominator of the logarithm's argument leaves:

$$B_z \cong \frac{\mu I H_{coil}}{2D_{wire}^2}\ln\left\{\frac{1 + \frac{2\left[R_{core} + \left(N_{layers} \times D_{wire}\right)\right]}{H_{coil}} + \frac{2\left[R_{core} + \left(N_{layers} \times D_{wire}\right)\right]^2}{H_{coil}^2}}{1 + \frac{2R_{core}}{H_{coil}} + \frac{2R_{core}^2}{H_{coil}^2}}\right\}$$

The numerator and denominator each contain three terms: unity, a first-order term in R_{core}/H_{coil} and a second-order term in R_{core}/H_{coil}. Having already assumed that the coil is much longer than its radius, the second-order term will be much smaller than the first-order term. If we ignore the second-order terms, the expression reduces to:

$$B_z \cong \frac{\mu I H_{coil}}{2D_{wire}^2}\ln\left\{\frac{1 + \frac{2\left[R_{core} + \left(N_{layers} \times D_{wire}\right)\right]}{H_{coil}}}{1 + \frac{2R_{core}}{H_{coil}}}\right\}$$

Since $R_{core} \ll H_{coil}$, we can make good use of another Taylor series expansion and truncation:

$$\begin{aligned}
\frac{1}{1+\frac{2R_{core}}{H_{coil}}} &= \left(\frac{1}{1+\frac{2R_{core}}{H_{coil}}}\right)\left(\frac{1-\frac{2R_{core}}{H_{coil}}}{1-\frac{2R_{core}}{H_{coil}}}\right) \\
&= \frac{1-\frac{2R_{core}}{H_{coil}}}{1-\frac{4R_{core}^2}{H_{coil}^2}} \\
&= \left(1-\frac{2R_{core}}{H_{coil}}\right)\left[1+\left(\frac{4R_{core}^2}{H_{coil}^2}\right)^1+\left(\frac{4R_{core}^2}{H_{coil}^2}\right)^2+\left(\frac{4R_{core}^2}{H_{coil}^2}\right)^3+\cdots\right] \\
&= 1-\frac{2R_{core}}{H_{coil}}+\frac{4R_{core}^2}{H_{coil}^2}-8\left(\frac{R_{core}}{H_{coil}}\right)^3+\cdots \\
&\cong 1-\frac{2R_{core}}{H_{coil}}
\end{aligned}$$

Applying this approximation to the expression for B_z gives:

$$\begin{aligned}
B_z &\cong \frac{\mu I H_{coil}}{2D_{wire}^2}\ln\left[\left(1-\frac{2R_{core}}{H_{coil}}\right)\left\{1+\frac{2\left[R_{core}+\left(N_{layers}\times D_{wire}\right)\right]}{H_{coil}}\right\}\right] \\
&= \frac{\mu I H_{coil}}{2D_{wire}^2}\ln\left[1-\frac{2R_{core}}{H_{coil}}+\frac{2\left[R_{core}+\left(N_{layers}\times D_{wire}\right)\right]}{H_{coil}}-\frac{4R_{core}\left[R_{core}+\left(N_{layers}\times D_{wire}\right)\right]}{H_{coil}^2}\right] \\
&= \frac{\mu I H_{coil}}{2D_{wire}^2}\ln\left[1+\frac{2\left(N_{layers}\times D_{wire}\right)}{H_{coil}}-\frac{4R_{core}^2+4R_{core}\left(N_{layers}\times D_{wire}\right)}{H_{coil}^2}\right]
\end{aligned}$$

If the coil is much longer than its radius, it should be the case the coil is also much longer than the thickness of its winding, $N_{layers}\times D_{wire}$. Therefore, we can use a third Taylor series expansion, that $\ln(1+x)=x-\frac{1}{2}x^2+\frac{1}{3}x^3+\cdots$, and approximate the series with its first term, x, with the following result:

$$\begin{aligned}
B_z &\cong \frac{\mu I H_{coil}}{2D_{wire}^2}\left[\frac{2\left(N_{layers}\times D_{wire}\right)}{H_{coil}}\right] \\
&= \frac{\mu I N_{layers}}{D_{wire}}
\end{aligned}$$

Remember that the height of the coil H_{coil} is related to the wire diameter D_{wire} by the number of turns per layer N_{turns} through the relationship $H_{coil}=N_{turns}\times D_{wire}$. Eliminating D_{wire} using this relationship gives the classical result for the magnetic field at the center of a long solenoid:

$$\vec{B}\Big|_{center} \cong \frac{\mu I N_{turns}N_{layers}}{H_{coil}}\hat{z} \quad (8)$$

The factor $IN_{turns}N_{layers}$ is called the "ampere-turns" of the coil. It is one measure of the coil's strength. Note that the magnetic field strength does not depend on the radius of the coil, so long as it is much less than the length.

As a numerical example, consider a coil 100mm long (about four inches) wound with 1000 total turns carrying *1* ampere. The magnetic field at this coil's center is axial and, if it has an air-core, has a magnitude equal to:

$$B = \frac{(4\pi \times 10^{-7})(1)(1000)}{0.1} = 0.0126\text{T}$$

T is the Tesla, the SI unit of magnetic field strength. One Tesla equals 10,000 Gauss, so this field strength can also be expressed as 126 Gauss. I have already pointed out that the Earth's magnetic field has a strength of about one-half Gauss. By way of comparison, the magnetic field strength at the face of a rare earth magnet can be 10,000 Gauss or more.

(A second aside for a second benchmark solution)

The magnetic field at the geometric center of a solenoid is sometimes less useful to know than the magnetic field strength at the center of one "face" of the coil. Fortunately, this is very easy to calculate. Like many scalar and vector fields in nature, magnetic fields can be added by superposition. Two identical cylindrical air-core solenoids brought together end-to-end generate exactly the same magnetic field as one similar solenoid twice as long. In reverse, imagine a cylindrical solenoid being cut at its middle and the two halves separated. The magnetic field at the center of the two faces which were in proximity must add up to the magnetic field at the geometric center of the original solenoid. Since the two sub-solenoids are identical, each of the two faces must have contributed the same amount to the original field. One face is now a "north" pole and its counterpart is now a "south" pole. When added, the field lines add constructively. The magnetic field at the center of a face must be one-half of the strength of a solenoid twice as long:

$$\vec{B}\big|_{face\ center} \cong \frac{1}{2}\frac{\mu I N_{turns} N_{layers}}{H_{coil}}\hat{z} \quad (9)$$

(Computational procedure)

The following listing is a Visual Basic subroutine which calculates the components of the magnetic field at any given point in or around a cylindrical air-core solenoid. The co-ordinate frame is as used above – centered at the coil's center with the $\hat{z}$-axis axial. The variable names used in the subroutine match those used above. The ByVal arguments passed to the subroutine are the input parameters; the ByRef arguments are the magnetic field components returned by the subroutine. The subroutine also returns a string, which is either "SUCCESS" or an error message.

```
'////////////////////////////////////////////////////////////////////////////////
'// Generalized subroutine to calculate the components of the magnetic field at any
'// arbitrary point in or around a cylindrical air-core solenoid.
'// Nturns = number of turns of wire in each layer
'// Nlayers = number of layers of wire in the winding
'// I = current flow, in amperes
'// Rcore = inner radius of the winding, in meters
'// Hcoil = height of winding, in meters
'// *** Note that Dwire, the diameter of the wire, is calculated from Hcoil / Nturns ***
'// U = permeability of the medium, in T/mA
'// Ntheta = number of arcs into which each turn is divided (1000 is good)
'// Pz = axial displacement from center of coil to point P, in meters
'// Pr = radial displacement from centerline of coil to point P, in meters
'// Pphi = angular displacement of point P from the x-axis, in radians
```

```
'// Bx, By, Bz = components of the magnetic field at vector P, in Telsa
'// ReturnString = "SUCCESS" or an appropriate error message
Public Sub MagneticFieldAroundCoil( _
    ByVal Nturns As Int32, _
    ByVal Nlayers As Int32, _
    ByVal I As Double, _
    ByVal Rcore As Double, _
    ByVal Hcoil As Double, _
    ByVal U As Double, _
    ByVal Ntheta As Int32, _
    ByVal Pz As Double, _
    ByVal Pr As Double, _
    ByVal Pphi As Double, _
    ByRef Bx As Double, _
    ByRef By As Double, _
    ByRef Bz As Double, _
    ByRef ReturnString As String)
    Dim PI As Double = 3.141592654
    Dim Dwire As Double         ' Diameter of the wire, in meters
    Dim zNthturn As Double      ' Height of the Nth turn from center of coil, in meters
    Dim rMthlayer As Double     ' Radius of Mth layer from centerline of coil, in meters
    Dim Theta As Double         ' Azimuth of current element from x-axis
    Dim CosTheta As Double      ' Cosine of angle Theta
    Dim SinTheta As Double      ' Sine of angle Theta
    Dim CosThetaPhi As Double   ' Cosine of differential angle Theta - Pphi
    Dim Denominator1 As Double  ' Temporary variable
    Dim Denominator2 As Double  ' Temporary variable
    Dim dBx As Double
    Dim dBy As Double
    Dim dBz As Double
    ' Validate the incoming arguments.
    If (Nturns < 1) Then
        ReturnString = "Error -- number of turns must be positive"
        Exit Sub
    End If
    If (Nlayers < 1) Then
        ReturnString = "Error -- number of layers must be positive"
        Exit Sub
    End If
    If (Ntheta <= 100) Then
        ReturnString = "Error -- number of elements per turn must exceed 100"
        Exit Sub
    End If
    If (Rcore < 0) Then
        ReturnString = "Error -- inner radius of coil must be positive"
        Exit Sub
    End If
    If (Hcoil < 0) Then
        ReturnString = "Error -- height of coil must be positive"
    End If
    Try
        Bx = 0 : By = 0 : Bz = 0
        Dwire = Hcoil / Nturns
        ' Loop which steps through the layers of the winding
         For M As Int32 = 1 To Nlayers Step 1
            rMthlayer = Rcore + (Dwire * (M - 0.5))
            ' Loop which steps through the turns in each layer
            For N As Int32 = 1 To Nturns Step 1
```

```
                zNthturn = Dwire * (N - 0.5 - (Nturns / 2))
                Denominator1 = (Pr * Pr) + _
                    ((Pz - zNthturn) * (Pz - zNthturn)) + _
                    (rMthlayer * rMthlayer)
                ' Loop which steps around a single turn
                For Itheta As Int32 = 1 To Ntheta Step 1
                    Theta = 2 * PI * (Itheta - 0.5) / Ntheta
                    CosTheta = Math.Cos(Theta)
                    SinTheta = Math.Sin(Theta)
                    CosThetaPhi = Math.Cos(Theta - Pphi)
                    Denominator2 = Denominator1 - (2 * Pr * rMthlayer * CosThetaPhi)
                    Denominator2 = Denominator2 * Math.Sqrt(Denominator2)
                    Denominator2 = rMthlayer * (2 * PI / Ntheta) / Denominator2
                    dBx = (Pz - zNthturn) * CosTheta * Denominator2
                    dBy = (Pz - zNthturn) * SinTheta * Denominator2
                    dBz = (rMthlayer - (Pr * CosThetaPhi)) * Denominator2
                    Bx = Bx + dBx
                    By = By + dBy
                    Bz = Bz + dBz
                Next Itheta
            Next N
        Next M
        Bx = I * U * Bx / (4 * PI)
        By = I * U * By / (4 * PI)
        Bz = I * U * Bz / (4 * PI)
        ReturnString = "SUCCESS"
    Catch e As Exception
        ReturnString = "Error -- " & e.ToString
    End Try
End Sub
```

One further bit of information will be needed by the practitioner of coil winding: the diameters of standard enameled copper wire. The following table sets out the diameter, in millimeters, for a range of common AWG wire gauges. To be useful, the table also gives the resistance per unit length. And, finally, the table is expressed as a Visual Basic subroutine, with which these properties can be looked up programmatically.

```
'////////////////////////////////////////////////////////////////////////////////
'// Generalized subroutine which returns the diameter and lineal resistance of common
'// enameled copper wire.
'// Gauge = AWG solid enameled copper wire gauge
'// Diameter = wire diameter, including coating, in millimeters
'// Resistance = resistance per unit length, in Ohms per meters
'// ReturnString = "SUCCESS" or an appropriate error message
Public Sub GetWireProperties( _
    ByVal Gauge As Int32, _
    ByRef Diameter As Double, _
    ByRef Resistance As Double, _
    ByRef ReturnString As String)
    ReturnString = "SUCCESS"
    Select Case Gauge
        Case 14 : Diameter = 1.69 : Resistance = 0.00844
        Case 15 : Diameter = 1.51 : Resistance = 0.0106
        Case 16 : Diameter = 1.34 : Resistance = 0.0134
        Case 17 : Diameter = 1.2 : Resistance = 0.0169
        Case 18 : Diameter = 1.08 : Resistance = 0.0213
        Case 19 : Diameter = 0.962 : Resistance = 0.027
```

```
        Case 20 : Diameter = 0.864 : Resistance = 0.0339
        Case 21 : Diameter = 0.767 : Resistance = 0.0428
        Case 22 : Diameter = 0.686 : Resistance = 0.054
        Case 23 : Diameter = 0.615 : Resistance = 0.0681
        Case 24 : Diameter = 0.549 : Resistance = 0.0858
        Case 25 : Diameter = 0.491 : Resistance = 0.108
        Case 26 : Diameter = 0.438 : Resistance = 0.136
        Case 27 : Diameter = 0.391 : Resistance = 0.172
        Case 28 : Diameter = 0.349 : Resistance = 0.217
        Case 29 : Diameter = 0.311 : Resistance = 0.274
        Case 30 : Diameter = 0.281 : Resistance = 0.345
        Case 31 : Diameter = 0.251 : Resistance = 0.435
        Case 32 : Diameter = 0.225 : Resistance = 0.549
        Case 33 : Diameter = 0.2 : Resistance = 0.692
        Case 34 : Diameter = 0.178 : Resistance = 0.872
        Case 35 : Diameter = 0.161 : Resistance = 1.1
        Case 36 : Diameter = 0.145 : Resistance = 1.39
        Case 37 : Diameter = 0.128 : Resistance = 1.75
        Case 38 : Diameter = 0.113 : Resistance = 2.21
        Case 39 : Diameter = 0.102 : Resistance = 2.78
        Case 40 : Diameter = 0.09 : Resistance = 3.51
        Case Else : ReturnString = "Error -- unknown wire gauge"
    End Select
End Sub
```

(A procedure to approximate the magnetic field strength by interpolation)

Above, we developed general equations (summations, but not closed-form) which can be used to calculate the magnetic field vector at any point inside or around a cylindrical air-core solenoid. The computational procedure which implements the summation is straightforward but time-consuming. If the coil has ten layers and three hundred turns per layer, and one divides the turns into 5000 discrete arcs, the procedure can take two or three seconds or more on a typical PC. This is not a problem if one only wants to calculate the magnetic field at a few dozen or hundred spots.

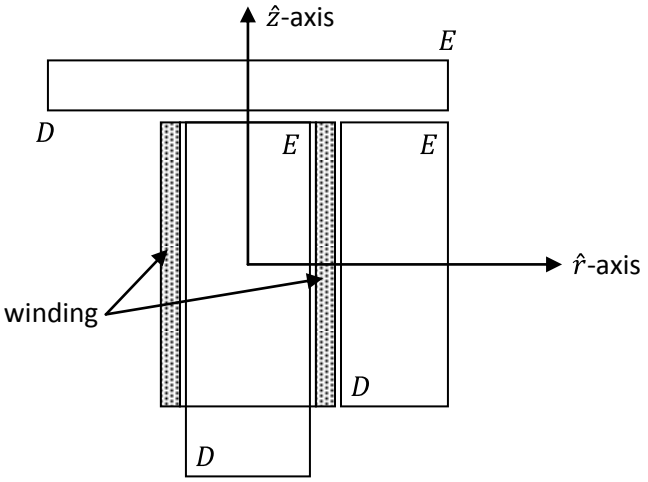
Usually, there is a greater goal. If one wanted to calculate the force this solenoid exerts on a nearby magnet, for example, the magnetic field would need to be calculated at each point in the whole region of space occupied by the magnet. Even if one took advantage of rotational symmetry, and calculated the magnetic field strength in a grid 1000 points wide and 1000 points high, the procedure would need to be repeated a million times. For ready reference, the three million seconds the calculation would take is equal to 35 days. Something else needs to be done if calculating the magnetic field is part of a bigger procedure.

We will start by taking advantage of the radial nature of the magnetic field inside and around the cylindrical solenoid, which has already been described. Using the notation from above, we can focus our attention on a single radial plane, which is the same as all other radial planes. For convenience, we will pick the radial plane which is defined by $\varphi = 0$ in our general equations. When the co-ordinate frame is specified in this way, we need only concern ourselves with the axial z-co-ordinate and the radial r-co-ordinate. The general equations reduce to:

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

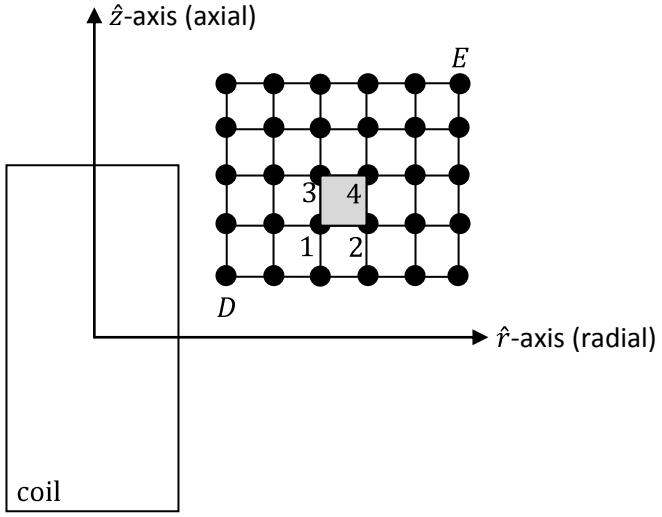
$$B_z = \frac{\mu}{4\pi} \sum_{m=1}^{m=N_{layers}} \sum_{n=1}^{n=N_{turns}} \sum_{\theta=0}^{\theta=2\pi} \frac{-I r_{Mth\ layer}\left[P_r \cos(\theta - \varphi) - r_{Mth\ layer}\right]\Delta\theta}{\left[P_r^2 + (P_z - z_{Nth\ turn})^2 + r_{Mth\ layer}^2 - 2P_r r_{Mth\ layer}\cos\theta\right]^{3/2}}$$

Suppose, for our greater purpose, that we want to calculate the magnetic field in some rectangular region inside or outside the solenoid (but not including any part of the winding). The rectangles DE shown in the following diagram are samples of three regions which might be of interest to us.



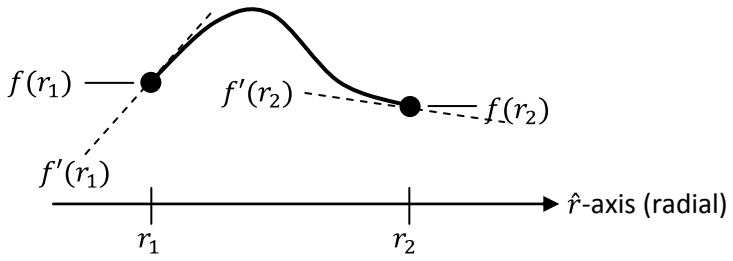
What we will do is this: we will impose a grid of modest size on region DE, perhaps a grid 100 points wide and 100 points high. We will calculate the magnetic field components B_r and B_z, and a couple of other variables, at each of these 10,000 grid points. That would take several hours, but less than a day. Then, we will devise a strategy to interpolate the values of the magnetic field components at any arbitrary location within the grid and between grid points.

Let us impose a rectangular grid on whatever region DE is of interest. The following figure shows a rudimentary grid.



I have shown region DE as being divided into a 5×4 grid, with 6×5 grid points, but a useful grid would likely be divided much more finely than this. Neither the region nor the grid need be square, but they do need to be rectangular, and the grid points must be equally-spaced in their respective directions. I have shaded in grey one of the rectangles in the grid and labeled its four corners. The order of the points will be important during the analysis – points 3 and 4 are immediately above points 1 and 2, respectively, and points 2 and 4 are immediately to the right of points 1 and 3, respectively.

We are going to try to approximate a function within this grid rectangle based on what we know at the four corners. Consider points 1 and 2, which have the same z-value. Consider also an arbitrary function of the radial distance r, $f(r)$, as shown in the following figure:



Suppose we happened to know the value of the function, $f(r_1)$ and $f(r_2)$, at the two points and the value of the first derivative, $f'(r_1)$ and $f'(r_2)$, there as well. With these four bits of information, we could find the four constants in the following cubic polynomial which approximates the function:

$$f(r) = C_0 + C_1 r + C_2 r^2 + C_3 r^3, \text{for constant } z$$

A cubic polynomial is not a bad choice – it has exactly one point of inflection, just like the magnetic field components which we are interested in.

If we now consider points 1 and 3, which have the same r-value, and repeat the process, we could find the four constants in the following cubic polynomial which approximates the function (in the z-direction):

$$f(z) = D_0 + D_1 z + D_2 z^2 + D_3 z^3, \text{for constant } r$$

We can combine the two strategies, and look for a polynomial of the following form, to approximate the function f over the r-z surface (within this grid rectangle, at least):

$$f(r,z) = C_0 + C_1 r + C_2 r^2 + C_3 r^3 + C_4 r + C_5 r^2 + C_6 r^3$$

In fact, we can aim higher than this. A more general form of cubic polynomial in two dimensions is the following:

$$f(r,z) = C_0 + C_1 r + C_2 r^2 + C_3 r^3 + C_4 z + C_5 z^2 + C_6 z^3 + C_7 rz + C_8 rz^2 + C_9 r^2 z \quad (10)$$

This more general polynomial includes all combinations of powers of the independent variables of three or less. There are ten constants. However, we seem to have twelve pieces of information from which to find them: the value of the function at the four corners, the first derivative of the function in the r-direction at the four corners and the first derivative of the function in the z-direction at the four corners. This is unusual, but let us proceed in the customary way.

The proposed polynomial has the following first partial derivatives:

$$\frac{\partial}{\partial r}f(r,z) = C_1 + 2C_2 r + 3C_3 r^2 + C_7 z + C_8 z^2 + 2C_9 rz$$
$$\frac{\partial}{\partial z}f(r,z) = C_4 + 2C_5 z + 3C_6 z^2 + C_7 r + 2C_8 rz + C_9 r^2$$

We can write down the boundary conditions at the four corners as follows:

$$Value\ at\ point\ 1 \qquad f(r,z)|_1 \qquad = \begin{cases} C_0 + C_1 r_1 + C_2 r_1^2 + C_3 r_1^3 + C_4 z_1 + C_5 z_1^2 + C_6 z_1^3 + \cdots \\ C_7 r_1 z_1 + C_8 r_1 z_1^2 + C_9 r_1^2 z_1 \end{cases}$$
$$Value\ at\ point\ 2 \qquad f(r,z)|_2 \qquad = \begin{cases} C_0 + C_1 r_2 + C_2 r_2^2 + C_3 r_2^3 + C_4 z_2 + C_5 z_2^2 + C_6 z_2^3 + \cdots \\ C_7 r_2 z_2 + C_8 r_2 z_2^2 + C_9 r_2^2 z_2 \end{cases}$$
$$Value\ at\ point\ 3 \qquad f(r,z)|_3 \qquad = \begin{cases} C_0 + C_1 r_3 + C_2 r_3^2 + C_3 r_3^3 + C_4 z_3 + C_5 z_3^2 + C_6 z_3^3 + \cdots \\ C_7 r_3 z_3 + C_8 r_3 z_3^2 + C_9 r_3^2 z_3 \end{cases}$$
$$Value\ at\ point\ 4 \qquad f(r,z)|_4 \qquad = \begin{cases} C_0 + C_1 r_4 + C_2 r_4^2 + C_3 r_4^3 + C_4 z_4 + C_5 z_4^2 + C_6 z_4^3 + \cdots \\ C_7 r_4 z_4 + C_8 r_4 z_4^2 + C_9 r_4^2 z_4 \end{cases}$$
$$r - derivative\ at\ point\ 1 \qquad \frac{\partial}{\partial r}f(r,z)\Big|_1 \qquad = C_1 + 2C_2 r_1 + 3C_3 r_1^2 + C_7 z_1 + C_8 z_1^2 + 2C_9 r_1 z_1$$
$$r - derivative\ at\ point\ 2 \qquad \frac{\partial}{\partial r}f(r,z)\Big|_2 \qquad = C_1 + 2C_2 r_2 + 3C_3 r_2^2 + C_7 z_2 + C_8 z_2^2 + 2C_9 r_2 z_2$$
$$r - derivative\ at\ point\ 3 \qquad \frac{\partial}{\partial r}f(r,z)\Big|_3 \qquad = C_1 + 2C_2 r_3 + 3C_3 r_3^2 + C_7 z_3 + C_8 z_3^2 + 2C_9 r_3 z_3$$
$$r - derivative\ at\ point\ 4 \qquad \frac{\partial}{\partial r}f(r,z)\Big|_4 \qquad = C_1 + 2C_2 r_4 + 3C_3 r_4^2 + C_7 z_4 + C_8 z_4^2 + 2C_9 r_4 z_4$$
$$z - derivative\ at\ point\ 1 \qquad \frac{\partial}{\partial z}f(r,z)\Big|_1 \qquad = C_4 + 2C_5 z_1 + 3C_6 z_1^2 + C_7 r_1 + 2C_8 r_1 z_1 + C_9 r_1^2$$
$$z - derivative\ at\ point\ 2 \qquad \frac{\partial}{\partial z}f(r,z)\Big|_2 \qquad = C_4 + 2C_5 z_2 + 3C_6 z_2^2 + C_7 r_2 + 2C_8 r_2 z_2 + C_9 r_2^2$$
$$z - derivative\ at\ point\ 3 \qquad \frac{\partial}{\partial z}f(r,z)\Big|_3 \qquad = C_4 + 2C_5 z_3 + 3C_6 z_3^2 + C_7 r_3 + 2C_8 r_3 z_3 + C_9 r_3^2$$
$$z - derivative\ at\ point\ 4 \qquad \frac{\partial}{\partial z}f(r,z)\Big|_4 \qquad = C_4 + 2C_5 z_4 + 3C_6 z_4^2 + C_7 r_4 + 2C_8 r_4 z_4 + C_9 r_4^2$$

These equations can be written in matrix form. For now, let us look at the eight derivative conditions only. In matrix form, they are:

$$\begin{bmatrix} \frac{\partial}{\partial r} f(r,z)\Big|_1 \\ \frac{\partial}{\partial r} f(r,z)\Big|_2 \\ \frac{\partial}{\partial r} f(r,z)\Big|_3 \\ \frac{\partial}{\partial r} f(r,z)\Big|_4 \\ \frac{\partial}{\partial z} f(r,z)\Big|_1 \\ \frac{\partial}{\partial z} f(r,z)\Big|_2 \\ \frac{\partial}{\partial z} f(r,z)\Big|_3 \\ \frac{\partial}{\partial z} f(r,z)\Big|_4 \end{bmatrix} \equiv \begin{bmatrix} 0 & 1 & 2r_1 & 3r_1^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_1z_1 \\ 0 & 1 & 2r_2 & 3r_2^2 & 0 & 0 & 0 & z_2 & z_2^2 & 2r_2z_2 \\ 0 & 1 & 2r_3 & 3r_3^2 & 0 & 0 & 0 & z_3 & z_3^2 & 2r_3z_3 \\ 0 & 1 & 2r_4 & 3r_4^2 & 0 & 0 & 0 & z_4 & z_4^2 & 2r_4z_4 \\ 0 & 0 & 0 & 0 & 1 & 2z_1 & 3z_1^2 & x_1 & 2r_1z_1 & r_1^2 \\ 0 & 0 & 0 & 0 & 1 & 2z_2 & 3z_2^2 & x_2 & 2r_2z_2 & r_2^2 \\ 0 & 0 & 0 & 0 & 1 & 2z_3 & 3z_3^2 & x_3 & 2r_3z_3 & r_3^2 \\ 0 & 0 & 0 & 0 & 1 & 2z_4 & 3z_4^2 & x_4 & 2r_4z_4 & r_4^2 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

However, recall that we chose the four points which define the grid rectangle so that $z_2 = z_1$, $z_4 = z_3$, $r_3 = r_1$ and $r_4 = r_2$, in which case these equations can be re-written, with the components of the vector on the left-hand side defined by position, as:

$$\begin{bmatrix} LHS_1 \\ LHS_2 \\ LHS_3 \\ LHS_4 \\ LHS_5 \\ LHS_6 \\ LHS_7 \\ LHS_8 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2r_1 & 3r_1^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_1z_1 \\ 0 & 1 & 2r_2 & 3r_2^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_2z_1 \\ 0 & 1 & 2r_1 & 3r_1^2 & 0 & 0 & 0 & z_3 & z_3^2 & 2r_1z_3 \\ 0 & 1 & 2r_2 & 3r_2^2 & 0 & 0 & 0 & z_3 & z_3^2 & 2r_2z_3 \\ 0 & 0 & 0 & 0 & 1 & 2z_1 & 3z_1^2 & r_1 & 2r_1z_1 & r_1^2 \\ 0 & 0 & 0 & 0 & 1 & 2z_1 & 3z_1^2 & r_2 & 2r_2z_1 & r_2^2 \\ 0 & 0 & 0 & 0 & 1 & 2z_3 & 3z_3^2 & r_1 & 2r_1z_3 & r_1^2 \\ 0 & 0 & 0 & 0 & 1 & 2z_3 & 3z_3^2 & r_2 & 2r_2z_3 & r_2^2 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

We can take linear combinations of the rows without affecting the equality. Let us subtract the first row from the third row, the second row from the fourth row, the fifth row from the sixth row, and the seventh from the eighth row. The four difference equations are:

$$\begin{bmatrix} LHS_3 - LHS_1 \\ LHS_4 - LHS_2 \\ LHS_6 - LHS_5 \\ LHS_8 - LHS_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_3 - z_1 & z_3^2 - z_1^2 & 2r_1(z_3 - z_1) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & z_3 - z_1 & z_3^2 - z_1^2 & 2r_2(z_3 - z_1) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_2 - r_1 & 2z_1(r_2 - r_1) & r_2^2 - r_1^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_2 - r_1 & 2z_3(r_2 - r_1) & r_2^2 - r_1^2 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

Oh, dear! Here we have four equations which depend on only three variables: C_7, C_8 and C_9. This is telling us that these eight boundary conditions are not independent from each other. Knowing seven of the derivatives, we can calculate the eighth. In order for our boundary conditions to be independent, we must drop one of them. Which one we drop does not matter. For convenience, let us eliminate the last boundary condition, for $\frac{\partial}{\partial z} f(r,z)\Big|_4$.

In a similar way, let us now look at only the four function-value conditions. In matrix form, they are:

$$\begin{bmatrix} f(r,z)|_1 \\ f(r,z)|_2 \\ f(r,z)|_3 \\ f(r,z)|_4 \end{bmatrix} = \begin{bmatrix} 1 & r_1 & r_1^2 & r_1^3 & z_1 & z_1^2 & z_1^3 & r_1 z_1 & r_1 z_1^2 & r_1^2 z_1 \\ 1 & r_2 & r_2^2 & r_2^3 & z_2 & z_2^2 & z_2^3 & r_2 z_2 & r_2 z_2^2 & r_2^2 z_2 \\ 1 & r_3 & r_3^2 & r_3^3 & z_3 & z_3^2 & z_3^3 & r_3 z_3 & r_3 z_3^2 & r_3^2 z_3 \\ 1 & r_4 & r_4^2 & r_4^3 & z_4 & z_4^2 & z_4^3 & r_4 z_4 & r_4 z_4^2 & r_4^2 z_4 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

As we did before, we can substitute the equalities for the grid rectangle: $z_2 = z_1$, $z_4 = z_3$, $r_3 = r_1$ and $r_4 = r_2$. Focusing only on the right-hand side of the equation, we get:

$$\begin{bmatrix} LHS_1 \\ LHS_2 \\ LHS_3 \\ LHS_4 \end{bmatrix} = \begin{bmatrix} 1 & r_1 & r_1^2 & r_1^3 & z_1 & z_1^2 & z_1^3 & r_1 z_1 & r_1 z_1^2 & r_1^2 z_1 \\ 1 & r_2 & r_2^2 & r_2^3 & z_1 & z_1^2 & z_1^3 & r_2 z_1 & r_2 z_1^2 & r_2^2 z_1 \\ 1 & r_1 & r_1^2 & r_1^3 & z_3 & z_3^2 & z_3^3 & r_1 z_3 & r_1 z_3^2 & r_1^2 z_3 \\ 1 & r_2 & r_2^2 & r_2^3 & z_3 & z_3^2 & z_3^3 & r_2 z_3 & r_2 z_3^2 & r_2^2 z_3 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

Subtracting the first row from the third row, and the second row from the fourth row, we get the following two difference equations:

$$\begin{bmatrix} LHS_3 - LHS_1 \\ LHS_4 - LHS_2 \end{bmatrix} = \begin{bmatrix} z_3 - z_1 & z_3^2 - z_1^2 & z_3^3 - z_1^3 & r_1(z_3 - z_1) & r_1(z_3^2 - z_1^2) & r_1^2(z_3 - z_1) \\ z_3 - z_1 & z_3^2 - z_1^2 & z_3^3 - z_1^3 & r_2(z_3 - z_1) & r_2(z_3^2 - z_1^2) & r_2^2(z_3 - z_1) \end{bmatrix} \begin{bmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

We can repeat our new trick, and subtract the first row from the second row, giving one final difference equation:

$$\begin{bmatrix} LHS_4 - LHS_2 + \cdots \\ \cdots - (LHS_3 - LHS_1) \end{bmatrix} = \begin{bmatrix} (r_2 - r_1)(z_3 - z_1) & (r_2 - r_1)(z_3^2 - z_1^2) & (r_2^2 - r_1^2)(z_3 - z_1) \end{bmatrix} \begin{bmatrix} C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

Here, we have another equation which depends only on the same three variables: C_7, C_8 and C_9. So, we still have one more boundary condition that we need. We can drop another of the boundary conditions. For convenience, let us drop the other derivative at point 4, for $\frac{\partial}{\partial r} f(r,z)\Big|_4$, from further consideration. Now have ten equations in ten unknowns. These ten equations should be independent. Defining quantities on the left-hand side using H_i in order to simplify the appearance of the equations, we get:

$$\begin{bmatrix} f(r,z)|_1 \\ f(r,z)|_2 \\ f(r,z)|_3 \\ f(r,z)|_4 \\ \frac{\partial}{\partial r} f(r,z)\Big|_1 \\ \frac{\partial}{\partial r} f(r,z)\Big|_2 \\ \frac{\partial}{\partial r} f(r,z)\Big|_3 \\ \frac{\partial}{\partial z} f(r,z)\Big|_1 \\ \frac{\partial}{\partial z} f(r,z)\Big|_2 \\ \frac{\partial}{\partial z} f(r,z)\Big|_3 \end{bmatrix} \equiv \begin{bmatrix} H_1 \\ H_2 \\ H_3 \\ H_4 \\ H_5 \\ H_6 \\ H_7 \\ H_8 \\ H_9 \\ H_{10} \end{bmatrix} = \begin{bmatrix} 1 & r_1 & r_1^2 & r_1^3 & z_1 & z_1^2 & z_1^3 & r_1 z_1 & r_1 z_1^2 & r_1^2 z_1 \\ 1 & r_2 & r_2^2 & r_2^3 & z_1 & z_1^2 & z_1^3 & r_2 z_1 & r_2 z_1^2 & r_2^2 z_1 \\ 1 & r_1 & r_1^2 & r_1^3 & z_3 & z_3^2 & z_3^3 & r_1 z_3 & r_1 z_3^2 & r_1^2 z_3 \\ 1 & r_2 & r_2^2 & r_2^3 & z_3 & z_3^2 & z_3^3 & r_2 z_3 & r_2 z_3^2 & r_2^2 z_3 \\ 0 & 1 & 2r_1 & 3r_1^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_1 z_1 \\ 0 & 1 & 2r_2 & 3r_2^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_2 z_1 \\ 0 & 1 & 2r_1 & 3r_1^2 & 0 & 0 & 0 & z_3 & z_3^2 & 2r_1 z_3 \\ 0 & 0 & 0 & 0 & 1 & 2z_1 & 3z_1^2 & r_1 & 2r_1 z_1 & r_1^2 \\ 0 & 0 & 0 & 0 & 1 & 2z_1 & 3z_1^2 & r_2 & 2r_2 z_1 & r_2^2 \\ 0 & 0 & 0 & 0 & 1 & 2z_3 & 3z_3^2 & r_1 & 2r_1 z_3 & r_1^2 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

I have already substituted the equalities of the grid rectangle, that $z_2 = z_1$ and so on. We now take linear combinations of rows which look similar. Defining $\Delta r = r_2 - r_1$ and $\Delta z = z_2 - z_1$, we get:

$$\begin{bmatrix} H_1 \\ H_2 - H_1 \\ H_3 - H_1 \\ H_4 - H_2 - (H_3 - H_1) \\ H_5 \\ H_6 - H_5 \\ H_7 - H_5 \\ H_8 \\ H_9 - H_8 \\ H_{10} - H_8 \end{bmatrix} = \cdots$$

$$= \begin{bmatrix} 1 & r_1 & r_1^2 & r_1^3 & z_1 & z_1^2 & z_1^3 & r_1 z_1 & r_1 z_1^2 & r_1^2 z_1 \\ 0 & \Delta r & r_2^2 - r_1^2 & r_2^3 - r_1^3 & 0 & 0 & 0 & z_1 \Delta r & z_1^2 \Delta r & z_1 (r_2^2 - r_1^2) \\ 0 & 0 & 0 & 0 & \Delta z & z_3^2 - z_1^2 & z_3^3 - z_1^3 & r_1 \Delta z & r_1 (z_3^2 - z_1^2) & r_1^2 \Delta z \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta r \Delta z & \Delta r (z_3^2 - z_1^2) & \Delta z (r_2^2 - r_1^2) \\ 0 & 1 & 2r_1 & 3r_1^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_1 z_1 \\ 0 & 0 & 2\Delta r & 3(r_2^2 - r_1^2) & 0 & 0 & 0 & 0 & 0 & 2z_1 \Delta r \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta z & z_3^2 - z_1^2 & 2r_1 \Delta z \\ 0 & 0 & 0 & 0 & 1 & 2z_1 & 3z_1^2 & r_1 & 2r_1 z_1 & r_1^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \Delta r & 2z_1 \Delta r & r_2^2 - r_1^2 \\ 0 & 0 & 0 & 0 & 0 & 2\Delta z & 3(z_3^2 - z_1^2) & 0 & 2r_1 \Delta z & 0 \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

If we re-order the equations, then the matrix becomes block-upper-triangular with three blocks.

Block #1 consists of the fourth, seventh and ninth equations, for which:

$$\begin{bmatrix} H_4 - H_2 - (H_3 - H_1) \\ H_7 - H_5 \\ H_9 - H_8 \end{bmatrix} = \begin{bmatrix} \Delta r \Delta z & \Delta r(z_3^2 - z_1^2) & \Delta z(r_2^2 - r_1^2) \\ \Delta z & z_3^2 - z_1^2 & 2r_1 \Delta z \\ \Delta r & 2z_1 \Delta r & r_2^2 - r_1^2 \end{bmatrix} \begin{bmatrix} C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

Block #2 consists of the third, eighth and tenth equations, for which:

$$\begin{bmatrix} H_3 - H_1 \\ H_8 \\ H_{10} - H_8 \end{bmatrix} = \begin{bmatrix} \Delta z & z_3^2 - z_1^2 & z_3^3 - z_1^3 & r_1 \Delta z & r_1(z_3^2 - z_1^2) & r_1^2 \Delta z \\ 1 & 2z_1 & 3z_1^2 & r_1 & 2r_1 z_1 & r_1^2 \\ 0 & 2\Delta z & 3(z_3^2 - z_1^2) & 0 & 2r_1 \Delta z & 0 \end{bmatrix} \begin{bmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

Block #3 consists of the first, second, fifth and sixth equations, for which:

$$\begin{bmatrix} H_1 \\ H_2 - H_1 \\ H_5 \\ H_6 - H_5 \end{bmatrix} = \begin{bmatrix} 1 & r_1 & r_1^2 & r_1^3 & z_1 & z_1^2 & z_1^3 & r_1 z_1 & r_1 z_1^2 & r_1^2 z_1 \\ 0 & \Delta r & r_2^2 - r_1^2 & r_2^3 - r_1^3 & 0 & 0 & 0 & z_1 \Delta r & z_1^2 \Delta r & z_1(r_2^2 - r_1^2) \\ 0 & 1 & 2x_1 & 3r_1^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_1 z_1 \\ 0 & 0 & 2\Delta x & 3(r_2^2 - r_1^2) & 0 & 0 & 0 & 0 & 0 & 2z_1 \Delta r \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

We can solve the three blocks separately. We can solve Block #1 for C_7, C_8 and C_9 as follows:

$$\begin{bmatrix} H_4 - H_2 - (H_3 - H_1) \\ H_7 - H_5 \\ H_9 - H_8 \end{bmatrix} \equiv \begin{bmatrix} J_8 \\ J_9 \\ J_{10} \end{bmatrix} = \begin{bmatrix} \Delta r \Delta z & \Delta r(z_3^2 - z_1^2) & \Delta z(r_2^2 - r_1^2) \\ \Delta z & z_3^2 - z_1^2 & 2r_1 \Delta z \\ \Delta r & 2z_1 \Delta r & r_2^2 - r_1^2 \end{bmatrix} \begin{bmatrix} C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} J_8 - \Delta z J_{10} \\ J_9 \\ J_{10} \end{bmatrix} = \begin{bmatrix} 0 & \Delta r(z_3^2 - z_1^2) - 2z_1 \Delta r \Delta z & 0 \\ \Delta z & z_3^2 - z_1^2 & 2r_1 \Delta z \\ \Delta r & 2z_1 \Delta r & r_2^2 - r_1^2 \end{bmatrix} \begin{bmatrix} C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

Noting that $r_2^2 - r_1^2 = (r_1 + \Delta r)^2 - r_1^2 = 2r_1 \Delta r + \Delta r^2$ and similarly that $z_3^2 - z_1^2 = 2z_1 \Delta z + \Delta z^2$, this simplifies to:

$$\begin{bmatrix} J_8 - \Delta z J_{10} \\ J_9 \\ J_{10} \end{bmatrix} = \begin{bmatrix} 0 & \Delta r \Delta z^2 & 0 \\ \Delta z & 2z_1 \Delta z + \Delta z^2 & 2r_1 \Delta z \\ \Delta r & 2z_1 \Delta r & 2r_1 \Delta r + \Delta r^2 \end{bmatrix} \begin{bmatrix} C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} J_8 - \Delta z J_{10} \\ J_9 \\ \Delta z J_{10} - \Delta x J_9 \end{bmatrix} = \begin{bmatrix} 0 & \Delta r \Delta z^2 & 0 \\ \Delta z & 2z_1 \Delta z + \Delta z^2 & 2r_1 \Delta z \\ 0 & -\Delta r \Delta z^2 & \Delta z \Delta r^2 \end{bmatrix} \begin{bmatrix} C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

Withdrawing the common factor Δz leaves:

$$\begin{bmatrix} J_8 - \Delta z J_{10} \\ J_9 \\ \Delta z J_{10} - \Delta x J_9 \end{bmatrix} = \Delta z \begin{bmatrix} 0 & \Delta r \Delta z & 0 \\ 1 & 2z_1 + \Delta z & 2r_1 \\ 0 & -\Delta r \Delta z & \Delta r^2 \end{bmatrix} \begin{bmatrix} C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

C_8 can be found directly from the first equation in this block. Knowing C_8, then C_9 can be found from the last equation. Finally, knowing both C_8 and C_9, C_7 can be found from the middle equation.

Having now calculated C_7, C_8 and C_9, we turn to Block #2. We can solve this block for three more constants, as follows:

$$\begin{bmatrix} H_3 - H_1 \\ H_8 \\ H_{10} - H_8 \end{bmatrix} \equiv \begin{bmatrix} J_5 \\ J_6 \\ J_7 \end{bmatrix} = \begin{bmatrix} \Delta z & z_3^2 - z_1^2 & z_3^3 - z_1^3 & r_1 \Delta z & r_1(z_3^2 - z_1^2) & r_1^2 \Delta z \\ 1 & 2z_1 & 3z_1^2 & r_1 & 2r_1 z_1 & r_1^2 \\ 0 & 2\Delta z & 3(z_3^2 - z_1^2) & 0 & 2r_1 \Delta z & 0 \end{bmatrix} \begin{bmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} J_5 - \Delta z J_6 \\ J_6 \\ J_7 \end{bmatrix} = \begin{bmatrix} 0 & z_3^2 - z_1^2 - 2z_1 \Delta z & z_3^3 - z_1^3 - 3z_1^2 \Delta z & 0 & r_1(z_3^2 - z_1^2) - 2r_1 z_1 \Delta z & 0 \\ 1 & 2z_1 & 3z_1^2 & r_1 & 2r_1 z_1 & r_1^2 \\ 0 & 2\Delta z & 3(z_3^2 - z_1^2) & 0 & 2r_1 \Delta z & 0 \end{bmatrix} \begin{bmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

Substituting $z_3^2 - z_1^2 = 2z_1 \Delta z + \Delta z^2$ and $z_3^3 - z_1^3 = 3z_1^2 \Delta z + 3z_1 \Delta z^2 + \Delta z^3$, this simplifies to:

$$\begin{bmatrix} J_5 - \Delta z J_6 \\ J_6 \\ J_7 \end{bmatrix} = \begin{bmatrix} 0 & \Delta z^2 & 3z_1 \Delta z^2 + \Delta z^3 & 0 & r_1 \Delta z^2 & 0 \\ 1 & 2z_1 & 3z_1^2 & r_1 & 2r_1 z_1 & r_1^2 \\ 0 & 2\Delta z & 6z_1 \Delta z + 3\Delta z^2 & 0 & 2r_1 \Delta z & 0 \end{bmatrix} \begin{bmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} J_5 - \Delta z J_6 - \frac{1}{2} \Delta z J_7 \\ J_6 \\ J_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -\frac{1}{2} \Delta z^3 & 0 & 0 & 0 \\ 1 & 2z_1 & 3z_1^2 & r_1 & 2r_1 z_1 & r_1^2 \\ 0 & 2\Delta z & 6z_1 \Delta z + 3\Delta z^2 & 0 & 2r_1 \Delta z & 0 \end{bmatrix} \begin{bmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

C_6 can be found directly from the first equation in this block. Knowing C_6, then C_5 can be found from the last equation. Finally, knowing both C_5 and C_6, C_4 can be found from the middle equation.

Having now calculated six of the ten constants, we can attack Block #1 to calculate the remaining four.

$$\begin{bmatrix} H_1 \\ H_2 - H_1 \\ H_5 \\ H_6 - H_5 \end{bmatrix} \equiv \begin{bmatrix} J_1 \\ J_2 \\ J_3 \\ J_4 \end{bmatrix} = \begin{bmatrix} 1 & r_1 & r_1^2 & r_1^3 & z_1 & z_1^2 & z_1^3 & r_1 z_1 & r_1 z_1^2 & r_1^2 z_1 \\ 0 & \Delta r & r_2^2 - r_1^2 & r_2^3 - r_1^3 & 0 & 0 & 0 & z_1 \Delta r & z_1^2 \Delta r & z_1(r_2^2 - r_1^2) \\ 0 & 1 & 2r_1 & 3r_1^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_1 z_1 \\ 0 & 0 & 2\Delta r & 3(r_2^2 - r_1^2) & 0 & 0 & 0 & 0 & 0 & 2z_1 \Delta r \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

Subtracting Δr multiplied by the third equation from the second equation, and substituting expressions for the difference of squares and cubes, this becomes:

$$\begin{bmatrix} J_1 \\ J_2 - \Delta x J_3 \\ J_3 \\ J_4 \end{bmatrix} = \begin{bmatrix} 1 & r_1 & r_1^2 & r_1^3 & z_1 & z_1^2 & z_1^3 & r_1 z_1 & r_1 z_1^2 & r_1^2 z_1 \\ 0 & 0 & \Delta r^2 & 3r_1 \Delta r^2 + \Delta r^3 & 0 & 0 & 0 & 0 & 0 & z_1 \Delta r^2 \\ 0 & 1 & 2r_1 & 3r_1^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_1 z_1 \\ 0 & 0 & 2\Delta r & 6r_1 \Delta r + 3\Delta r^2 & 0 & 0 & 0 & 0 & 0 & 2z_1 \Delta r \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

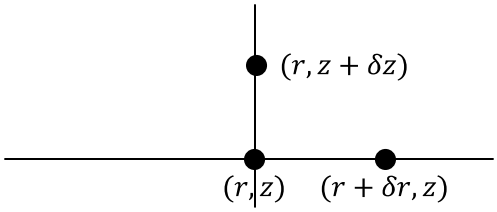
$$\rightarrow \begin{bmatrix} J_1 \\ J_2 - \Delta x J_3 - \frac{1}{2}\Delta x J_4 \\ J_3 \\ J_4 \end{bmatrix} = \begin{bmatrix} 1 & r_1 & r_1^2 & r_1^3 & z_1 & z_1^2 & z_1^3 & r_1 z_1 & r_1 z_1^2 & r_1^2 z_1 \\ 0 & 0 & 0 & -\frac{1}{2}\Delta r^3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2r_1 & 3r_1^2 & 0 & 0 & 0 & z_1 & z_1^2 & 2r_1 z_1 \\ 0 & 0 & 2\Delta r & 6r_1 \Delta r + 3\Delta r^2 & 0 & 0 & 0 & 0 & 0 & 2z_1 \Delta r \end{bmatrix} \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \end{bmatrix}$$

C_3 can be found directly from the second equation in this block. Knowing C_3, then C_2 can be found from the last equation. Then, knowing both C_2 and C_3, C_1 can be found directly from the third equation. Finally, C_0 can be found from the first equation.

There are three things to note.

1. When reducing the ten equations above, no division was done.
2. All of the leading coefficients in the ten reduced equations are either 1, powers of Δx or powers of Δz. So long as the grid is not degenerate, and $\Delta x \neq 0$ and $\Delta z \neq 0$, none of the leading coefficients will be zero.
3. The implication of the first two observations is that there will always be a solution to the ten equations as long as the grid is not degenerate.

We still have not talked about calculating the partial derivatives at each point. The method I have found successful is based on evaluating the function at two other points around and very near each point in the grid, as shown in the following diagram:



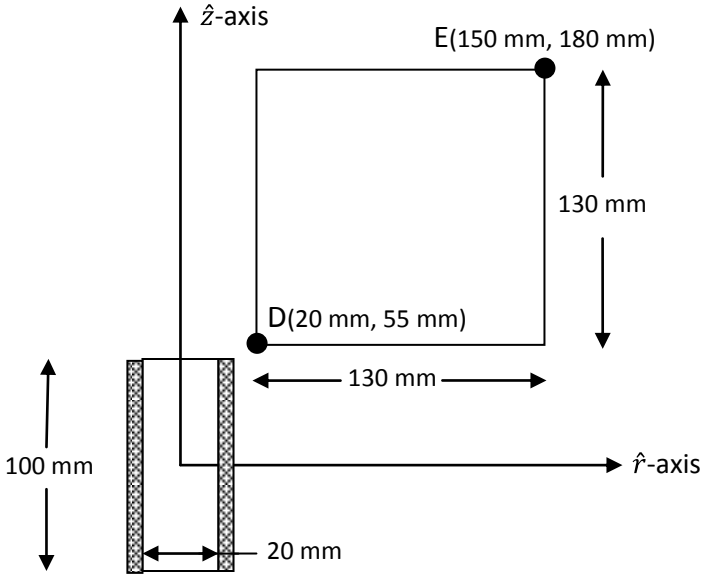
One can pick δr and δz small, but not so small that dividing by small numbers causes computational errors. It should be understood that δr and δz are not the same as the grid spacing Δr and Δz. Picking them to be 1% of the size of their respective grid sizes would be a good starting point. The approximations for the partial derivatives at point (r, z) are then given by:

$$\frac{\partial}{\partial r} f(r,y)\Big|_{(r,y)} \cong \frac{f(r+\delta r, z) - f(r,z)}{\delta r}$$
$$\frac{\partial}{\partial z} f(r,y)\Big|_{(r,y)} \cong \frac{f(r, z+\delta z) - f(r,z)}{\delta z}$$

This procedure requires that the function be calculated at three points – remember that the value at point (r, z) itself is also needed – for each grid point in region DE. If the region of interest was divided into 25 elements across and 100 down, then $25 \times 100 \times 3 = 7{,}500$ calculations of the function would be needed. (In our Visual Basic subroutine listed above, both components of the magnetic field are calculated at the same time.)

(Sample calculation #1)

Let us consider again the coil described above, with height 100mm and core radius 10mm. Let us calculate the constants which will enable us to approximate B_z and B_r in a rectangular region to the upper right of the coil. The following figure shows the region DE we will investigate.



Region DE happens to be a 130 mm square. (I selected this region for a purpose. In a following paper, I want to calculate the forces this solenoid exerts on a similar one. I plan to put a solenoid of a similar size at various points in region DE to see what happens. Region DE is big enough to allow the new solenoid to be located at various relative positions.)

Region DE is divided into 2 mm squares, with 65 squares bordering each edge. Stated differently, 66 points were equally-spaced along each edge of region DE and a grid drawn between opposing points. The subroutine listed below, named PolynomialApproximationInRegion(), was executed. $65 \times 65 = 4{,}225$ pairs of sets of constants were calculated. Each set consists of the ten constants for use in Equation (10). Note that a pair of sets is needed for each grid point, one for component B_r and another for component B_z. This subroutine needed to call subroutine MagneticFieldAroundCoil() three times for each point in the grid, for a total of $3 \times 66 \times 66 = 13{,}068$ times. On my 2 GHz laptop, the calculation took a few minutes less than 24 hours.

Let us look at the result in a sample grid square, say, the square whose lower left corner is at $r = 42\text{mm}$ and $z = 67\text{mm}$. The constants for this grid square, in all their glorious precision, are:

Constants	B_r	B_z
C_0	3830252.84372085	3421200.92541774
C_1	54027.8426726818	−45479.3246763996
C_2	1167.54534457527	1006.8938634627
C_3	−8.9319027917693	−7.87819997862122
C_4	−134867.359251351	−122232.891448302
C_5	1948.49487770538	1778.28303179761
C_6	−9.4090608678326	−8.60224885558234
C_7	112.141186204333	63.5526136652289
C_8	−0.686356184619402	−0.561258655192558
C_9	−0.218551074911239	0.148586314269379

These constants must be used with the same units of measurement in which they were calculated. The units used in this case were Gauss and millimeters.

Subroutine PolynomialApproximationInRegion() does not scale lengths (as it could be revised to do) to, say, the r and z values at the principal corner. In exchange for this simplicity, one must be vigilant to ensure that the constants do not represent an exploding function. The following table shows the contribution of each term in the polynomials to the total magnetic field strengths.

Contribution	B_r	B_z
C_0	3830252.84372085	3421200.92541774
$C_1 r$	−2269169.39225264	−1910131.63640878
$C_2 r^2$	2059549.98783078	1776160.7751482
$C_3 r^3$	−661746.814036604	−583680.080016089
$C_4 z$	−9036113.06984052	−8189603.72703623
$C_5 z^2$	8746793.50601945	17982712.52973947
$C_6 z^3$	−2829897.37379194	−2587238.17255151
$C_7 rz$	315565.297978993	178837.054853954
$C_8 rz^2$	−129404.222335773	−105818.584332695
$C_9 r^2 z$	−25830.1144416095	17561.1193108694
Total	0.648851008 Gauss	0.204124898 Gauss

The totals, $B_r = 0.648851008$ Gauss and $B_z = 0.204124898$ Gauss, are the components of the magnetic field at the point $r = 42\text{mm}$ and $z = 67\text{mm}$. Because this point is a grid point, these components are exactly equal to the exact values. For any other point within this grid square, which occupies $-42\text{mm} \le r < 44\text{mm}$ and $67\text{mm} \le z < 69\text{mm}$, the ten components of B_r and B_z would being calculated and added up in the same way.

It would seem that all ten components of the polynomials contribute in roughly equal measures to the totals. That is good. On the other hand, the totals are the sums and differences of some very big numbers. That is bad. In addition, the contributions from the r^1, r^2 and r^3 terms do not diminish with increasing order as well as could be hoped. Neither do the terms from increasing orders of z. This means that the lower-order terms in the polynomials are not natural representations of the underlying functions. But, we already knew that. What saves us from the problems of a poorly-matched polynomial is double precision arithmetic and a small grid size.

Before indiscriminately using the polynomial approximations, it is wise to get some understanding for the errors involved. Since the approximation is derived using values at the four corners, intuition tells us that the maximum error will be near the center of each grid square, equidistant from the corners. Furthermore, recall that less information was used for the upper-right corner of each grid cell (both derivatives at point 4 were dropped), so the location of maximum error will probably lie slightly off center, in the direction towards point 4.

As a start towards understanding the errors, I calculated the exact values of B_r and B_z at the center of each grid square in region DE, and compared these with the values given by the polynomials. Both the absolute error and the percentage error are of interest.

The five worst absolute errors in B_r are between 0.019 Gauss and 0.034 Gauss. The five worst percentage errors in B_r are between 0.76% and 1.05%. The locations at which these errors occur are all located in the lower-left of region DE, where r is less than 23mm and z is less than 60mm. This is the area in region DE in which B_r changes most quickly.

The five worst absolute errors in B_z are between 0.008 Gauss and 0.010 Gauss. These absolute errors also occur in the lower-left of region DE.

The five worst percentage errors in B_z are between 31% and 127%. The locations at which these percentage errors occur fall roughly along a line passing diagonally through region DE, from the lower left to the upper right. At these points, B_z is very close to zero and the concept of percentage error begins to fail. For example, the worst percentage error in B_z occurs where B_z is only 0.000013 Gauss.

Errors of this magnitude were acceptable for my purpose. If your work requires more precision, then the easiest change would be to reduce the grid size. Generally, reducing the grid size by a factor of two should reduce the absolute errors by a factor of four. If there are areas in your region where B_r or B_z are zero or very close to zero, the percentage errors should be interpreted accordingly.

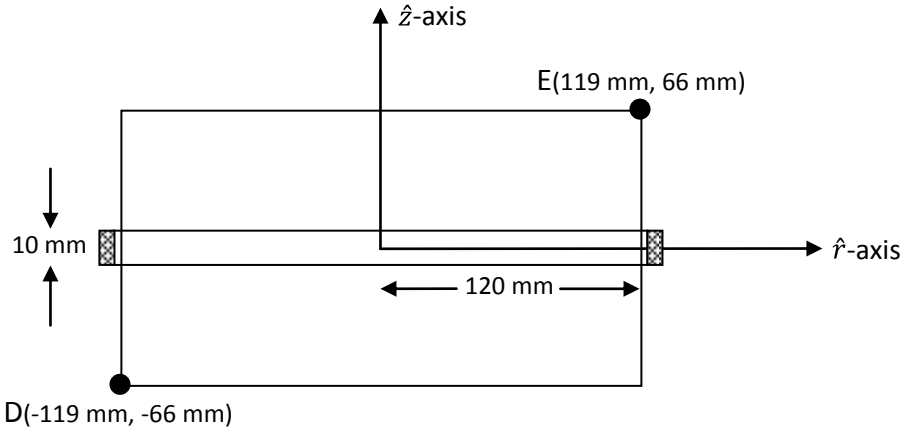
(Sample calculation #2)

Sample calculation #1 looked at a region outside of a solenoid. The methodology works just as well for a region inside a solenoid. (The methodology fails for locations inside the winding.) Sample calculation #2 looks at a very wide solenoid. It has a core radius of 120mm (about 4¾ inches) and a height of only 10mm (about ⅜ inch). It is shaped more like a loop than a traditional coil. Like the solenoid above, it is wound with 16 layers of #28 enameled wire. This winding uses 358 meters of wire. When powered by

a 5V dc supply, it draws 64.4mA. This is a 29.9 Amp − turn magnet. (Comparison: the solenoid in Sample calculation #1 used 370 meters of wire, which drew 62.4mA at 5V. It had 286.7 Amp − turns.)

This solenoid, for Sample calculation #2, was chosen to be as different as possible from the one for Sample calculation #1. The solenoid in Sample calculation #1 may be useful for moving things around outside itself; the solenoid in Sample calculation #2 would be more useful for moving things around inside its bounds.

The following figure shows the region DE in which the constants for the polynomials were calculated.

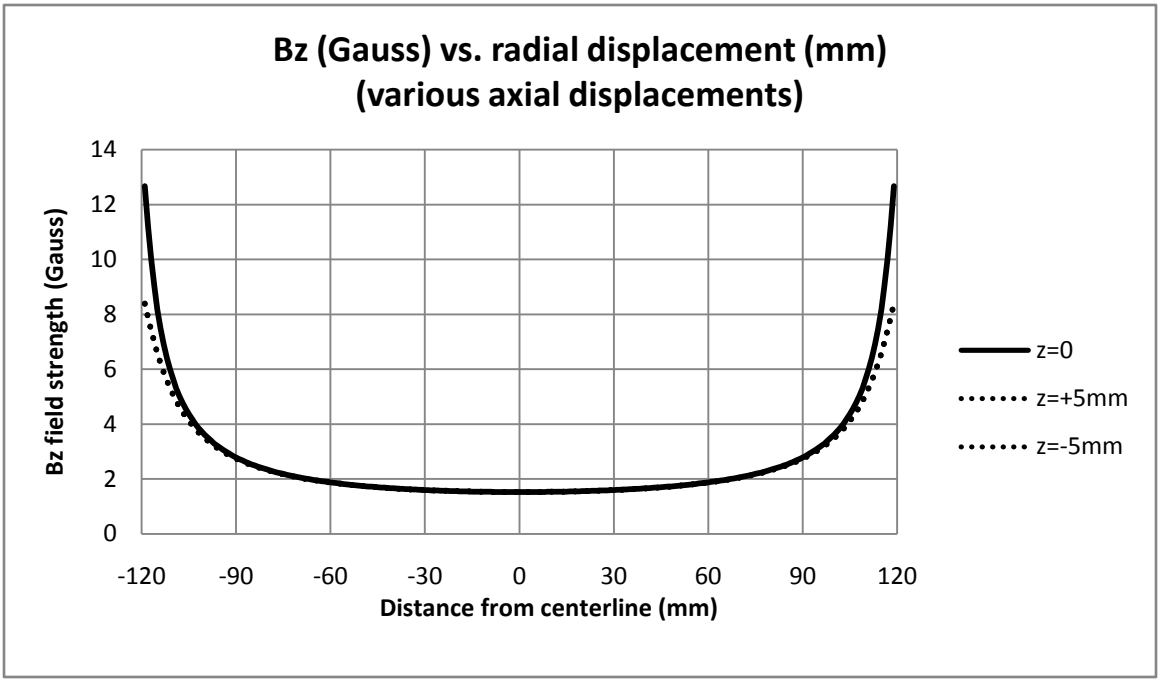


In this case, the region was divided into 1mm squares. There are $239 \times 133 = 31{,}787$ points in the grid. Although there are more grid points in this sample calculation than in Sample calculation #1, there are also fewer turns in the winding. As a result, the calculations of the magnetic field at the grid points run more quickly. Overall, calculating the constants took about seven hours. Of course, virtually all of this time was spent calculating the exact values of B_r and B_y, and their derivatives, at the grid points.

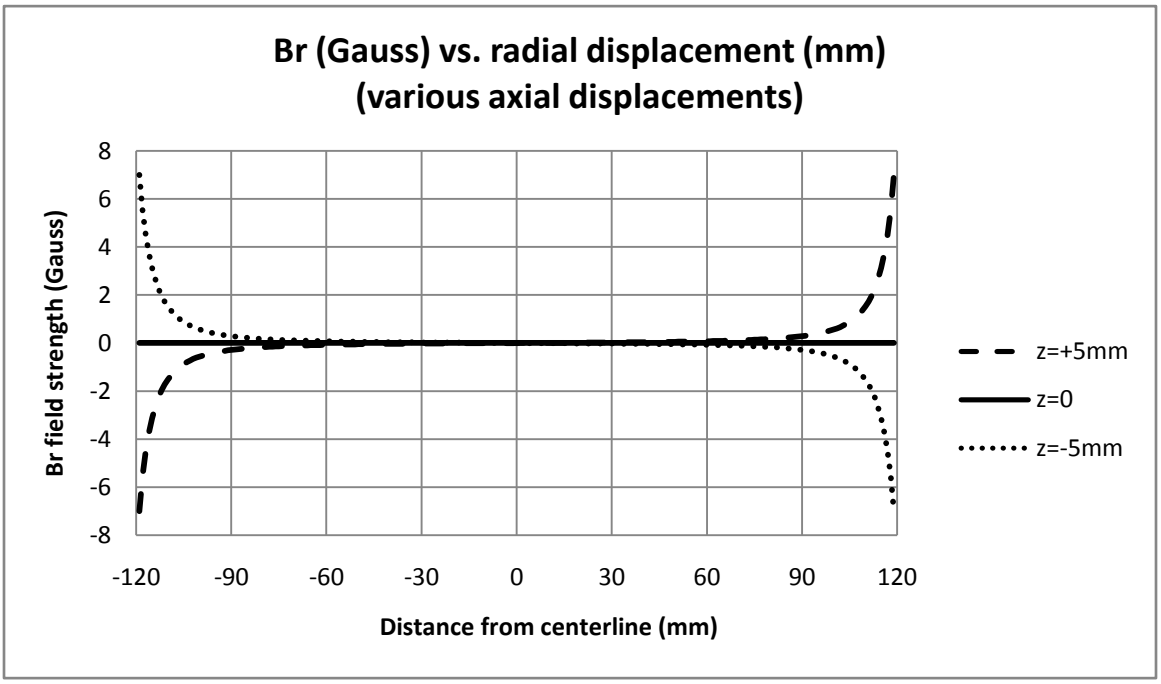
Calculating the constants took four times longer than it could have taken. Region DE is centered in the $\hat{r}$-$\hat{z}$ plane so there is symmetry among the four quadrants of the region. I could have calculated the constants only in the first quadrant, for example. I did not do this. Not knowing what references would be made to this region in subsequent work, I did not want to complicate all subsequent references by requiring that they sort out the relevant symmetries.

It should be noted that, although the magnetic field is symmetric among the quadrants, the constants themselves are not symmetric. The constants are derived from squares in which the order of the corners is important. In Quadrant 1, the lower-left point of each grid point (point 1) is the one closest to the origin. In Quadrant 3, point 1 is the one farthest from the origin. This asymmetry means that the constants for any particular grid square are not necessarily related to the constants for the corresponding grid square in the other quadrants.

For interest's sake, the following figure shows the axial component (B_z) of the magnetic field across a diameter of this solenoid. B_z is shown for an equatorial diameter ($z = 0$) and at the top ($z = 5$ mm) and bottom ($z = -5$ mm) of the winding. B_z takes the shape of a well, with a minimum at the center and rising towards the winding. The rise at the edges is quite steep. The curves across the top and bottom of the coil are identical. Because this magnet is very wide compared to its height, the shape of the field is very much like that of a single circular loop.



The following figure shows the radial component (B_r) of the magnetic field across the diameters at the same three axial displacements.



The radial component is zero or very small throughout most of the central area, but rises very steeply near the winding. The radial component is asymmetric.

As before, it is necessary to take a look at the errors involved in the polynomial approximations before putting them to work. I calculated the absolute and percentage errors at the centers of all the grid points

in the upper-right quadrant of region DE. The errors vary across the region and are best understood in "bands" of the radial displacement r.

Band (mm)	B_r absolute error	B_r percentage error	B_z absolute error	B_z percentage error
$0 \le r < 1$	0.00018 Gauss	6.8%	0.000031 Gauss	0.003%
$1 \le r < 2$	0.00018 Gauss	2.3%	0.000022 Gauss	0.002%
$2 \le r < 5$	0.00018 Gauss	1.4%	0.000015 Gauss	0.001%
$5 \le r < 100$	0.0023 Gauss	4.1%	0.0017 Gauss	0.057%
$100 \le r < 110$	0.011 Gauss	6.6%	0.0082 Gauss	0.18%
$110 \le r < 115$	0.031 Gauss	8.9%	0.027 Gauss	0.44%
$115 \le r < 119$	0.094 Gauss	9.5%	0.14 Gauss	1.8%

The errors in the first three bands (close to the centerline) occur far above and below the solenoid (at $z = \pm 65.5$ mm), where the magnetic field strength is lowest. The errors in the last three bands (nearing the winding) occur near the central plane of the solenoid, where the change in field strength with distance is greatest.

(Computational procedure)

The following listing is a Visual Basic subroutine which evaluates the constants of the two polynomials for B_r and B_z in a rectangular region. As always, the ByVal variables should be treated as arguments passed to the subroutine and the ByRef variables as those which the subroutine calculates and returns. The procedure saves the constants, as well as all intermediate calculations, to a text file.

```
'////////////////////////////////////////////////////////////////////////////////
'// Generalized subroutine to calculate the 20 coefficients of the two 3rd-order
'// polynomials which approximate both components of the magnetic field in a region of
'// space from D=(Rstart,Zstart) to E=(Rend,Zend) divided into a grid with N points in
'// the r-direction and M points in the z-direction.  Region DE must not intersect the
'// windings, but the subroutine does not check for this.
'//
'// Each grid square has a polynomial of form: Br(r,z) = C(0) + C(1)r + C(2)r^2 + ...
'// C(3)r^3 + C(4)z + C(5)z^2 + C(6)z^3 + C(7)rz + C(8)rz^2 + C(9)r^2z and a similar
'// polynomial for Bz(r,z).
'//
'// Results are saved to a file on disk named C:\MagFieldRegionApproximation.txt.
'// The saved file has three parts:
'// Part A: For each of N*M points, there are three lines of text:
'//         Line 1 contains r, z, Br and Bz at (r,z)
'//         Line 2 contains r+delr, z, Br and Bz at (r+delr,z)
'//         Line 3 contains r, z+delz, Br and Bz at (r,z+delz)
'// Part B: For each of N*M points, there are three lines of text:
'//         Line 1 contains r, z, Br and Bz at (r,z)
'//         Line 2 contains dBr/dr and dBr/dz at (r,z)
'//         Line 3 contains dBz/dr and dBz/dz at (r,z)
'// Part C: For each of N*M points, there are eleven lines of text:
'//         Line 1 contains r and z
'//         Line 2 contains BrC(0) and BzC(0) at (r,z)
'//         Line 3 contains BrC(1) and BzC(1) at (r,z)
'//         ...
'//         Line 11 contains BrC(9) and BzC(9) at (r,z)
'//
'// The increments delR and delZ used to estimate the partial derivatives are set
'// to 1% of the grid spacing.
'//
```

```
'// Nturns = number of turns of wire in each layer
'// Nlayers = number of layers of wire in the winding
'// Current = current flow, in amperes
'// Rcore = inner radius of the winding, in meters
'// Hcoil = height of windings, in meters
'// Rstart, Zstart = lower left corner of the region DE, in millimeters
'// Rend, Zend = upper right corner of the region DE, in millimeters
'// N = number of grid points in the r-direction
'// M = number of grid points in the z-direction
'// ReturnString = "SUCCESS" or an appropriate error message
Public Sub PolynomialApproximationInRegion( _
    ByVal Nturns As Int32, _
    ByVal Nlayers As Int32, _
    ByVal Current As Double, _
    ByVal Rcore As Double, _
    ByVal Hcoil As Double, _
    ByVal Rstart As Double, _
    ByVal Rend As Double, _
    ByVal Zstart As Double, _
    ByVal Zend As Double, _
    ByVal N As Int32, _
    ByVal M As Int32, _
    ByRef ReturnString As String)
    Dim OutStream As System.IO.StreamWriter ' Writer for text file
    Dim InStream As System.IO.StreamReader  ' Reader for text file
    Dim TempString As String
    Dim U As Double = 4 * PI * 0.0000001    ' Permeability
    Dim NumDelTheta As Int32 = 5000         ' Number of elements in a current loop
    ' Validate the incoming arguments.
    If (Nturns < 1) Then
        ReturnString = "Error -- number of turns must be positive"
        Exit Sub
    End If
    If (Nlayers < 1) Then
        ReturnString = "Error -- number of layers must be positive"
        Exit Sub
    End If
    If (Rcore < 0) Then
        ReturnString = "Error -- inner radius of coil must be positive"
        Exit Sub
    End If
    If (Hcoil < 0) Then
        ReturnString = "Error -- height of coil must be positive"
        Exit Sub
    End If
    If (Rend <= Rstart) Then
        ReturnString = "Error -- Rend must be greater than Rstart"
        Exit Sub
    End If
    If (Zend <= Zstart) Then
        ReturnString = "Error -- Zend must be greater than Zstart"
        Exit Sub
    End If
    If (N < 2) Then
        ReturnString = "Error -- horizontal grid N must be greater than 1"
        Exit Sub
    End If
    If (M < 2) Then
```

```
    ReturnString = "Error -- vertical grid M must be greater than 1"
    Exit Sub
End If
' Calculate delR and delZ for the purpose of calculating derivatives.
Dim delR As Double
Dim delZ As Double
delR = (Rend - Rstart) / (100 * (N - 1))
delZ = (Zend - Zstart) / (100 * (M - 1))
' Allow the user to skip the evaluations of the magnetic field.
If (System.IO.File.Exists("C:\MagFieldRegionApproximation.txt")) Then
    Dim MsgBoxResult As MsgBoxResult
    MsgBoxResult = MsgBox("File already exists.  Repeat function evaluations?", _
        vbYesNo, "Repeat function evaluations?")
    If (MsgBoxResult = vbNo) Then
        GoTo StartPartB
    End If
End If
' Open the file to save Part A: function evaluations.
OutStream = New System.IO.StreamWriter("C:\MagFieldRegionApproximation.txt", False)
If (OutStream Is Nothing) Then
    ReturnString = "Error -- Could not open output file"
    Exit Sub
End If
' Write coil and region information as a header for the file.
TempString = "Approximating the magnetic field components in a region:" & vbCrLf
TempString = TempString & "Rcore (mm) = " & Trim(Str(Rcore)) & vbCrLf
TempString = TempString & "Hcoil (mm) =" & Trim(Str(Hcoil)) & vbCrLf
TempString = TempString & "Nturns = " & Trim(Str(Nturns)) & vbCrLf
TempString = TempString & "Nlayers = " & Trim(Str(Nlayers)) & vbCrLf
TempString = TempString & "Current (A) =" & Trim(Str(Current)) & vbCrLf
TempString = TempString & "Rstart (mm) =" & Trim(Str(Rstart)) & vbCrLf
TempString = TempString & "Rend (mm) =" & Trim(Str(Rend)) & vbCrLf
TempString = TempString & "Zstart (mm) =" & Trim(Str(Zstart)) & vbCrLf
TempString = TempString & "Zend (mm) =" & Trim(Str(Zend)) & vbCrLf
TempString = TempString & "N = " & Trim(Str(N)) & vbCrLf
TempString = TempString & "M = " & Trim(Str(M)) & vbCrLf & vbCrLf
TempString = TempString & "Part A: Evaluations of the magnetic fields:" & vbCrLf
OutStream.Write(TempString)
'Part A
Dim R As Double
Dim Z As Double
Dim Bx As Double
Dim By As Double
Dim Bz As Double
' Main loop in the r-direction
For I As Int32 = 1 To N Step 1
    R = Rstart + ((Rend - Rstart) * (I - 1) / (N - 1))
    ' Main loop in the z-direction
    For J As Int32 = 1 To M Step 1
        Z = Zstart + ((Zend - Zstart) * (J - 1) / (M - 1))
        TempString = "r(mm)=" & Trim(FormatNumber(R, 3))
        TempString = TempString & " z(mm)=" & Trim(FormatNumber(Z, 3))
        MagneticFieldAroundCoil(Nturns, Nlayers, Current, _
            Rcore / 1000, Hcoil / 1000, _
            U, NumDelTheta, Z / 1000, R / 1000, 0, Bx, By, Bz, ReturnString)
        If (ReturnString <> "SUCCESS") Then
            ReturnString = "Error -- " & ReturnString
            Exit Sub
```

```
            End If
            TempString = TempString & " Br(G)=" & Trim(Str(Bx * 10000))
            TempString = TempString & " Bz(G)=" & Trim(Str(Bz * 10000)) & vbCrLf
            TempString = TempString & "r+delr(mm)=" & Trim(FormatNumber(R + delR, 3))
            TempString = TempString & " z(mm)=" & Trim(FormatNumber(Z, 3))
            MagneticFieldAroundCoil(Nturns, Nlayers, Current, _
                Rcore / 1000, Hcoil / 1000, _
                U, NumDelTheta, (R + delR) / 1000, Z / 1000, 0, Bx, By, Bz, ReturnString)
            If (ReturnString <> "SUCCESS") Then
                ReturnString = "Error -- " & ReturnString
                Exit Sub
            End If
            TempString = TempString & " Br(G)=" & Trim(Str(Bx * 10000))
            TempString = TempString & " Bz(G)=" & Trim(Str(Bz * 10000)) & vbCrLf
            TempString = TempString & "r(mm)=" & Trim(FormatNumber(R, 3))
            TempString = TempString & " z+delz(mm)=" & Trim(FormatNumber(Z + delZ, 3))
            MagneticFieldAroundCoil(Nturns, Nlayers, Current, _
                Rcore / 1000, Hcoil / 1000, _
                U, NumDelTheta, R / 1000, (Z + delZ) / 1000, 0, Bx, By, Bz, ReturnString)
            If (ReturnString <> "SUCCESS") Then
                ReturnString = "Error -- " & ReturnString
                Exit Sub
            End If
            TempString = TempString & " Br(G)=" & Trim(Str(Bx * 10000))
            TempString = TempString & " Bz(G)=" & Trim(Str(Bz * 10000)) & vbCrLf
            OutStream.Write(TempString)
            labelResultsC.Text = "R=" & Trim(Str(R)) & "  Z=" & Trim(Str(Z))
            Me.Refresh()
            Threading.Thread.Sleep(1000)
        Next J
    Next I
    ' Close the output file.
    OutStream.Close()
StartPartB:
    Dim Q(N, M, 12) As Double       ' Q(i,j,1)=r, Q(i,j,2)=z, ...
    '                               ' Q(i,j,3)=Br(r,z), Q(i,j,4)=Bz(r,z), ...
    '                               ' Q(i,j,5)=Br(r+delr,z), Q(i,j,6)=Bz(r+delr,z), ...
    '                               ' Q(i,j,7)=Br(r,z+delz), Q(i,j,8)=Bz(r,z+delz), ...
    '                               ' Q(i,j,9)=dBr(r,z)/dr, Q(i,j,10)=dBz(r,z)/dr, ...
    '                               ' Q(i,j,11)=dBr(r,z)/dz, Q(i,j,12)=dBz(x,y)/dz
    ' Open the file for input.
    Try
        InStream = New System.IO.StreamReader("C:\MagFieldRegionApproximation.txt")
        If (InStream Is Nothing) Then
            ReturnString = "Error -- Could not open the file"
            Exit Sub
        End If
        ' Find the last row of the header.
        Dim FoundLastLine As Boolean = False
        For I As Int32 = 1 To 100
            TempString = InStream.ReadLine
            If (Strings.Left(TempString, 6) = "Part A") Then
                FoundLastLine = True
                Exit For
            End If
        Next I
        If (FoundLastLine = False) Then
            InStream.Close()
```

```
        ReturnString = "Error -- Could not find last line of header"
        Exit Sub
    End If
Catch e As Exception
    ReturnString = "Error -- Problem opening file: " & e.ToString
    Exit Sub
End Try
' Read the rows and calculate the derivatives.
For Ir As Int32 = 1 To N Step 1
    For Iz As Int32 = 1 To M Step 1
        Dim Position As Int32
        If (InStream.EndOfStream) Then
            InStream.Close()
            ReturnString = "Error -- Data ended at (Ir, Iz) = " & _
                Trim(Str(Ir)) & ", " & Trim(Str(Iz))
            Exit Sub
        End If
        Try
            TempString = InStream.ReadLine
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, " ")
            Q(Ir, Iz, 1) = Val(Strings.Left(TempString, Position))
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, " ")
            Q(Ir, Iz, 2) = Val(Strings.Left(TempString, Position))
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, " ")
            Q(Ir, Iz, 3) = Val(Strings.Left(TempString, Position))
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Q(Ir, Iz, 4) = Val(TempString)
        Catch e As Exception
            ReturnString = "Error -- Problem reading line#1 for (Ir, Iz) = " & _
                Trim(Str(Ir)) & ", " & Trim(Str(Iz)) & ": " & e.ToString
            InStream.Close()
            Exit Sub
        End Try
        Try
            TempString = InStream.ReadLine
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, " ")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, " ")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, " ")
            Q(Ir, Iz, 5) = Val(Strings.Left(TempString, Position))
            TempString = Strings.Right(TempString, Len(TempString) - Position)
```

```
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Q(Ir, Iz, 6) = Val(TempString)
        Catch e As Exception
            ReturnString = "Error -- Problem reading line#2 for (Ir, Iz) = " & _
                Trim(Str(Ir)) & ", " & Trim(Str(Iz)) & ": " & e.ToString
            InStream.Close()
        End Try
        Try
            TempString = InStream.ReadLine
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, " ")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, " ")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, " ")
            Q(Ir, Iz, 7) = Val(Strings.Left(TempString, Position))
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Position = Strings.InStr(TempString, "=")
            TempString = Strings.Right(TempString, Len(TempString) - Position)
            Q(Ir, Iz, 8) = Val(TempString)
        Catch e As Exception
            ReturnString = "Error -- Problem reading line#3 for (Ir, Iz) = " & _
                Trim(Str(Ir)) & ", " & Trim(Str(Iz)) & ": " & e.ToString
            InStream.Close()
            Exit Sub
        End Try
    Next Iz
Next Ir
InStream.Close()
' Calculate the partial derivatives.
For Ir As Int32 = 1 To N Step 1
    For Iz As Int32 = 1 To M Step 1
        Try
            Q(Ir, Iz, 9) = (Q(Ir, Iz, 5) - Q(Ir, Iz, 3)) / delR
            Q(Ir, Iz, 10) = (Q(Ir, Iz, 6) - Q(Ir, Iz, 4)) / delR
            Q(Ir, Iz, 11) = (Q(Ir, Iz, 7) - Q(Ir, Iz, 3)) / delZ
            Q(Ir, Iz, 12) = (Q(Ir, Iz, 8) - Q(Ir, Iz, 4)) / delZ
        Catch e As Exception
            ReturnString = "Error -- Problem calculating derivatives at " & _
                vbCrLf & "(Ir, Iz)=(" & Trim(Str(Ir)) & ", " & Trim(Str(Iz)) & _
                ")" & vbCrLf & e.ToString
            Exit Sub
        End Try
    Next Iz
Next Ir
' Open the file to append Part B: partial derivatives.
OutStream = New System.IO.StreamWriter("C:\MagFieldRegionApproximation.txt", True)
If (OutStream Is Nothing) Then
    ReturnString = "Error -- Could not open output file"
    Exit Sub
End If
' Write the partial derivatives to the file for safe-keeping.
```

```
    TempString = " " & vbCrLf & "Part B: Partial derivatives" & vbCrLf
    OutStream.Write(TempString)
    For Ir As Int32 = 1 To N Step 1
        For Iz As Int32 = 1 To M Step 1
            TempString = "r(mm)=" & Trim(FormatNumber(Q(Ir, Iz, 1))) & _
                " z(mm)=" & Trim(FormatNumber(Q(Ir, Iz, 2))) & _
                " Br(G)=" & Trim(Str(Q(Ir, Iz, 3))) & _
                " Bz(G)=" & Trim(Str(Q(Ir, Iz, 4))) & vbCrLf & _
                "dBr/dr=" & Trim(Str(Q(Ir, Iz, 9))) & _
                " dBr/dz=" & Trim(Str(Q(Ir, Iz, 11))) & vbCrLf & _
                "dBz/dr=" & Trim(Str(Q(Ir, Iz, 10))) & _
                " dBz/dz=" & Trim(Str(Q(Ir, Iz, 12))) & vbCrLf
            OutStream.Write(TempString)
        Next Iz
    Next Ir
StartPartC:
    Dim BrC(N, M, 9) As Double      ' Coefficients of Br polynomial.  BrC(*,*,0) is used.
    Dim BzC(N, M, 9) As Double      ' Coefficients of Bz polynomial.  BzC(*,*,0) is used.
    For Ir As Int32 = 1 To N - 1 Step 1
        For Iz As Int32 = 1 To M - 1 Step 1
            ' Calculate delR and delZ for the purpose of solving the constants.
            delR = (Rend - Rstart) / (N - 1)
            delZ = (Zend - Zstart) / (M - 1)
            ' Calculate the coefficients of the Br component from the bottom left point
            ' of region DE to the top right point less 1 (that is, N-1 and M-1)
            ' Set up the co-ordinates
            Dim R1, R2, Z1, Z3 As Double
            R1 = Q(Ir, Iz, 1)         ' Q(*,*,1) is r
            R2 = Q(Ir + 1, Iz, 1)     ' Ir + 1 is one grid point to the right of (Ir, Iz)
            Z1 = Q(Ir, Iz, 2)         ' Q(*,*,2) is z
            Z3 = Q(Ir, Iz + 1, 2)     ' Iz + 1 is one grid point above (Ir, Iz)
            ' Set up vector H(10)
            Dim H1, H2, H3, H4, H5, H6, H7, H8, H9, H10 As Double
            H1 = Q(Ir, Iz, 3)         ' Q(*,*,3) is Br at Point 1 in the rectangle
            H2 = Q(Ir + 1, Iz, 3)     ' Point 2 in the rectangle
            H3 = Q(Ir, Iz + 1, 3)     ' Point 3 in the rectangle
            H4 = Q(Ir + 1, Iz + 1, 3) ' Point 4 in the rectangle
            H5 = Q(Ir, Iz, 9)         ' Q(*,*,9) is dBr/dr at Point 1 in the rectangle
            H6 = Q(Ir + 1, Iz, 9)     ' Point 2 in the rectangle
            H7 = Q(Ir, Iz + 1, 9)     ' Point 3 in the rectangle
            H8 = Q(Ir, Iz, 11)        ' Q(*,*,11) is dBr/dz at Point 1 in the rectangle
            H9 = Q(Ir + 1, Iz, 11)    ' Point 2 in the rectangle
            H10 = Q(Ir, Iz + 1, 11)   ' Point 3 in the rectangle
            ' Set up vector J(10)
            Dim J1, J2, J3, J4, J5, J6, J7, J8, J9, J10 As Double
            J1 = H1
            J2 = H2 - H1
            J3 = H5
            J4 = H6 - H5
            J5 = H3 - H1
            J6 = H8
            J7 = H10 - H8
            J8 = H1 - H2 - H3 + H4
            J9 = H7 - H5
            J10 = H9 - H8
            ' Modify the vector J(10) into JJ(10)
            Dim JJ1, JJ2, JJ3, JJ4, JJ5, JJ6, JJ7, JJ8, JJ9, JJ10 As Double
            JJ1 = J1
```

```
JJ2 = J2 - (delR * J3) - (0.5 * delR * J4)
JJ3 = J3
JJ4 = J4
JJ5 = J5 - (delZ * J6) - (0.5 * delZ * J7)
JJ6 = J6
JJ7 = J7
JJ8 = J8 - (delZ * J10)
JJ9 = J9
JJ10 = (delZ * J10) - (delR * J9)
' Solve for constants
Dim C0, C1, C2, C3, C4, C5, C6, C7, C8, C9 As Double
Dim RHS As Double
' First block
C8 = (JJ8 / delZ) / (delR * delZ)
RHS = -delR * delZ * C8
C9 = ((JJ10 / delZ) - RHS) / (delR * delR)
RHS = (((2 * Z1) + delZ) * C8) + (2 * R1 * C9)
C7 = (JJ9 / delZ) - RHS
' Second block
C6 = -JJ5 / (0.5 * delZ * delZ * delZ)
RHS = 2 * R1 * delZ * C8
RHS = RHS + ((6 * Z1 * delZ + 3 * delZ * delZ) * C6)
C5 = (JJ7 - RHS) / (2 * delZ)
RHS = (R1 * C7) + (2 * R1 * Z1 * C8) + (R1 * R1 * C9)
RHS = RHS + (2 * Z1 * C5) + (3 * Z1 * Z1 * C6)
C4 = JJ6 - RHS
' Third block
C3 = JJ2 / (-0.5 * delR * delR * delR)
RHS = 2 * Z1 * delR * C9
RHS = RHS + (((6 * R1 * delR) + (3 * delR * delR)) * C3)
C2 = (JJ4 - RHS) / (2 * delR)
RHS = (Z1 * C7) + (Z1 * Z1 * C8) + (2 * R1 * Z1 * C9)
RHS = RHS + (2 * R1 * C2) + (3 * R1 * R1 * C3)
C1 = JJ3 - RHS
RHS = (Z1 * C4) + (Z1 * Z1 * C5) + (Z1 * Z1 * Z1 * C6)
RHS = RHS + (R1 * Z1 * C7) + (R1 * Z1 * Z1 * C8) + (R1 * R1 * Z1 * C9)
RHS = RHS + (R1 * C1) + (R1 * R1 * C2) + (R1 * R1 * R1 * C3)
C0 = JJ1 - RHS
' Store the constants
BrC(Ir, Iz, 0) = C0
BrC(Ir, Iz, 1) = C1
BrC(Ir, Iz, 2) = C2
BrC(Ir, Iz, 3) = C3
BrC(Ir, Iz, 4) = C4
BrC(Ir, Iz, 5) = C5
BrC(Ir, Iz, 6) = C6
BrC(Ir, Iz, 7) = C7
BrC(Ir, Iz, 8) = C8
BrC(Ir, Iz, 9) = C9
'
' Calculate the coefficients of the Bz component from the bottom left point
' of region DE to the top right point less 1 (that is, N-1 and M-1)
' Set up vector H(10)
H1 = Q(Ir, Iz, 4)         ' Q(*,*,4) is Bz at Point 1 in the rectangle
H2 = Q(Ir + 1, Iz, 4)     ' Point 2 in the rectangle
H3 = Q(Ir, Iz + 1, 4)     ' Point 3 in the rectangle
H4 = Q(Ir + 1, Iz + 1, 4) ' Point 4 in the rectangle
H5 = Q(Ir, Iz, 10)        ' Q(*,*,10) is dBz/dr at Point 1 in the rectangle
```

```
H6 = Q(Ir + 1, Iz, 10)    ' Point 2 in the rectangle
H7 = Q(Ir, Iz + 1, 10)    ' Point 3 in the rectangle
H8 = Q(Ir, Iz, 12)        ' Q(*,*,12) is dBz/dz at Point 1 in the rectangle
H9 = Q(Ir + 1, Iz, 12)    ' Point 2 in the rectangle
H10 = Q(Ir, Iz + 1, 12)   ' Point 3 in the rectangle
' Set up vector J(10)
J1 = H1
J2 = H2 - H1
J3 = H5
J4 = H6 - H5
J5 = H3 - H1
J6 = H8
J7 = H10 - H8
J8 = H1 - H2 - H3 + H4
J9 = H7 - H5
J10 = H9 - H8
' Modify the vector J(10)
JJ1 = J1
JJ2 = J2 - (delR * J3) - (0.5 * delR * J4)
JJ3 = J3
JJ4 = J4
JJ5 = J5 - (delZ * J6) - (0.5 * delZ * J7)
JJ6 = J6
JJ7 = J7
JJ8 = J8 - (delZ * J10)
JJ9 = J9
JJ10 = (delZ * J10) - (delR * J9)
' Solve for constants
' First block
C8 = (JJ8 / delZ) / (delR * delZ)
RHS = -delR * delZ * C8
C9 = ((JJ10 / delZ) - RHS) / (delR * delR)
RHS = (((2 * Z1) + delZ) * C8) + (2 * R1 * C9)
C7 = (JJ9 / delZ) - RHS
' Second block
C6 = -JJ5 / (0.5 * delZ * delZ * delZ)
RHS = 2 * R1 * delZ * C8
RHS = RHS + ((6 * Z1 * delZ + 3 * delZ * delZ) * C6)
C5 = (JJ7 - RHS) / (2 * delZ)
RHS = (R1 * C7) + (2 * R1 * Z1 * C8) + (R1 * R1 * C9)
RHS = RHS + (2 * Z1 * C5) + (3 * Z1 * Z1 * C6)
C4 = JJ6 - RHS
' Third block
C3 = JJ2 / (-0.5 * delR * delR * delR)
RHS = 2 * Z1 * delR * C9
RHS = RHS + (((6 * R1 * delR) + (3 * delR * delR)) * C3)
C2 = (JJ4 - RHS) / (2 * delR)
RHS = (Z1 * C7) + (Z1 * Z1 * C8) + (2 * R1 * Z1 * C9)
RHS = RHS + (2 * R1 * C2) + (3 * R1 * R1 * C3)
C1 = JJ3 - RHS
RHS = (Z1 * C4) + (Z1 * Z1 * C5) + (Z1 * Z1 * Z1 * C6)
RHS = RHS + (R1 * Z1 * C7) + (R1 * Z1 * Z1 * C8) + (R1 * R1 * Z1 * C9)
RHS = RHS + (R1 * C1) + (R1 * R1 * C2) + (R1 * R1 * R1 * C3)
C0 = JJ1 - RHS
' Store the constants
BzC(Ir, Iz, 0) = C0
BzC(Ir, Iz, 1) = C1
BzC(Ir, Iz, 2) = C2
```

```
            BzC(Ir, Iz, 3) = C3
            BzC(Ir, Iz, 4) = C4
            BzC(Ir, Iz, 5) = C5
            BzC(Ir, Iz, 6) = C6
            BzC(Ir, Iz, 7) = C7
            BzC(Ir, Iz, 8) = C8
            BzC(Ir, Iz, 9) = C9
        Next Iz
    Next Ir
    ' Write the constants for the Br and Bz polynomials to the file for safe-keeping.
    TempString = " " & vbCrLf & "Part C: C() for Br and Bz" & vbCrLf
    OutStream.Write(TempString)
    For Ir As Int32 = 1 To N Step 1
        For Iz As Int32 = 1 To M Step 1
            TempString = "r(mm)=" & Trim(FormatNumber(Q(Ir, Iz, 1))) & _
                " z(mm)=" & Trim(FormatNumber(Q(Ir, Iz, 2))) & vbCrLf & _
                "BrC(0)=" & Trim(Str(BrC(Ir, Iz, 0))) & _
                " BzC(0)=" & Trim(Str(BzC(Ir, Iz, 0))) & vbCrLf & _
                "BrC(1)=" & Trim(Str(BrC(Ir, Iz, 1))) & _
                " BzC(1)=" & Trim(Str(BzC(Ir, Iz, 1))) & vbCrLf & _
                "BrC(2)=" & Trim(Str(BrC(Ir, Iz, 2))) & _
                " BzC(2)=" & Trim(Str(BzC(Ir, Iz, 2))) & vbCrLf & _
                "BrC(3)=" & Trim(Str(BrC(Ir, Iz, 3))) & _
                " BzC(3)=" & Trim(Str(BzC(Ir, Iz, 3))) & vbCrLf & _
                "BrC(4)=" & Trim(Str(BrC(Ir, Iz, 4))) & _
                " BzC(4)=" & Trim(Str(BzC(Ir, Iz, 4))) & vbCrLf & _
                "BrC(5)=" & Trim(Str(BrC(Ir, Iz, 5))) & _
                " BzC(5)=" & Trim(Str(BzC(Ir, Iz, 5))) & vbCrLf & _
                "BrC(6)=" & Trim(Str(BrC(Ir, Iz, 6))) & _
                " BzC(6)=" & Trim(Str(BzC(Ir, Iz, 6))) & vbCrLf & _
                "BrC(7)=" & Trim(Str(BrC(Ir, Iz, 7))) & _
                " BzC(7)=" & Trim(Str(BzC(Ir, Iz, 7))) & vbCrLf & _
                "BrC(8)=" & Trim(Str(BrC(Ir, Iz, 8))) & _
                " BzC(8)=" & Trim(Str(BzC(Ir, Iz, 8))) & vbCrLf & _
                "BrC(9)=" & Trim(Str(BrC(Ir, Iz, 9))) & _
                " BzC(9)=" & Trim(Str(BzC(Ir, Iz, 9))) & vbCrLf
            OutStream.Write(TempString)
        Next Iz
    Next Ir
    ' Close the file.
    OutStream.Close()
    ReturnString = "SUCCESS"
End Sub
```

Jim Hawley
September 2011

An e-mail describing errors and omissions would be appreciated.