Finding the Characteristics of Arbitrary Transmission Lines

A transmission-line analysis isn't always possible using the "book" formulas. Here's a way of handling that situation.

By Dave Kirkby, G8WRB

Transmission lines, in various forms, are used in great numbers in radio frequency (RF) equipment. Fig 1 shows five common transmission lines, as well as a totally arbitrary transmission line. Although one may believe that the important properties of transmission lines, such as the characteristic impedance are easily obtainable from readily available formulas, it is shown that this is not always the case. This article describes my efforts to overcome this when designing a high-power 144- to 146-MHz valve amplifier, although the technique has much wider use in the design of microwave circuits, RF relays, directional couplers, etc.

Stokes Hall Lodge
Burnham Road
Althorne, Essex CM3 6DT, England
e-mail **davek@medphys.ucl.ac.uk**

Transmission lines such as coaxial cable, twin-wire feeder, microstrip and stripline have distributed inductance along the conductors and distributed capacitance between the conductors, so a section of line can be represented by Fig 2.

The values of the distributed capacitance and inductance determine such properties as the characteristic impedance Z_0, which is calculated using Eq 1.

$$Z_0 = \sqrt{\frac{L}{C}}\ \Omega \qquad \text{Eq 1}$$

For example, a transmission line having a capacitance of 100 pF/m and an inductance of 250 nH/m would have a characteristic impedance of 50 Ω. They also determine the velocity, v, at which a radio wave propagates in the transmission line, according to Eq 2:

$$v = \frac{1}{\sqrt{LC}}\ \text{m/s} \qquad \text{Eq 2}$$

Therefore, a transmission line with C=100 pF/m and L=250 nH/m would propagate a radio wave at 2×10^8 m/s. Since radio waves propagate at 3×10^8 m/s in free space, the velocity factor of the transmission line would be $2\times10^8/3\times10^8 = 0.66$. These are common values for coaxial cables.

Finding the Properties of Common Types of Transmission Lines

Given knowledge of the physical dimensions of coaxial cable and the permittivity of the dielectric, its properties can easily be found with a few simple formulas. For example, the characteristic impedance Z_0 is given by Eq 3:

$$Z_0 = \frac{60}{\sqrt{\varepsilon_0 \varepsilon_r}} \log_e\left(\frac{D}{d}\right) \qquad \text{Eq 3}$$

where ε_0 is the permittivity of free space (ε_0=8.854x10^{-12} F/m), ε_r is the relative permittivity of the dielectric (ε_r=1.0 for air, 2.1 for PTFE, 2.3 for polyethylene etc), D is the inner diameter of the outer conductor and d is the outer diameter of the inner conductor. This formula is exact and easy to use with a scientific calculator.

Microstrip line is more complex to analyze, but an approximate formula, good enough for engineering purposes, can be found in the amateur press.[1, 2] An exact formula does exist for microstrip, but it is too complex for general use. An especially convenient calculation method is a small, free, Microsoft Windows program called *Txline* written by AWR which calculates the properties of microstrip and stripline, as well as three less widely used transmission line types. (Applied Wave Research Inc, USA, email **pekarek@appwave.com**. Web page: **http://www.appwave.com/**.)

Finding the Characteristics of an Arbitrary Transmission Line

Now consider a transmission line such as that on the right side of Fig 1. If you wish to find the characteristics of a transmission line such as this, there will definitely be no formula in a book! While such a transmission line is not likely to be encountered in practice, on occasion one does require characteristics of transmission lines that can not easily (if at all) be found in the literature. This happened to me when trying to determine the characteristic impedance of an air-spaced microstrip to be used as an anode resonator in a 144- to 146-MHz grounded-grid valve amplifier using an Eimac 3CX5000A7 triode. (The amplifier has not been completed yet, nor will it be for at least a year. The details may be published at a later date, assuming it works well.) Using transmission lines as resonators is a common practice in VHF valve amplifiers,[3,4] although contrary to popular belief, L-C tuned circuits can be successfully used.[5] When using transmission lines, theoretically any impedance line can be made to resonate with the valve's output capacitance if cut to the correct length,[6] but there is an optimum value for Z_0 for maximum amplifier bandwidth.[7] My aim was to design a transmission line with the optimal Z_0, which for my particular choice of valve (3CX5000A7) and resonator configuration (half-wave line with the tube at the center), was a line impedance of at least 81 Ω. Unfortunately, the presence of the metal amplifier case around the microstrip, essential for safety, could not be ignored. Making the case sufficiently large, so it was well away from the microstrip and could therefore be ignored, was not practical—I did not want the amplifier to fill half the shack! Hence my amplifier's enclosure was to become an integral part of an unsymmetrical shielded stripline transmission line, as shown in Fig 3, with a metal stripline of width w and thickness t, placed centrally on the horizontal axis at a height h above the bottom of a metal case of internal width W and internal height H. (The distinction between microstrip and shielded stripline is rather vague here. Any microstrip line enclosed in a metal box really becomes a shielded stripline. If, however, the enclosure is large compared with the microstrip, which is usually the case with low-power circuits, then for all practical purposes, the transmission line remains microstrip. Although my amplifier was intended to have an air-spaced microstrip resonator, the relative size of the enclosure to the anode line made the enclosure part of a shielded stripline resonator.) The term *unsymmetrical* is used to indicate that the center conductor is not in the center of the lower and upper conductors. To the best of my knowledge, following a computer search of the Science Citation Index, listing virtually all professional science and engineering publications since 1981, there is no analytical expression for the impedance of the transmission line in Fig 3. Robrish has, however, found one for an unsymmetrical stripline that is not shielded at the sides (equivalent to W being infinite).[8]

With a real need to solve a problem, but with no formula available to me, a *finite difference* computer program was written to solve numerically the problem of the shielded stripline in Fig 3. This was based on a method in a book by Dworsky where the interested reader can find full proofs of all the mathematics, which is quoted here without proof.[9] The program calculates the characteristic impedance (in Ω), the capacitance per meter (in pF/m) and the inductance per meter (in nH/m) of the transmission line. In its most basic form, the complete code is only 66 lines long and is reproduced in full in Appendix 1. The program expects the five parameters W, H, w, h and t to follow the program name on the command line. Once you understand how the program functions, modifying it to handle a weird transmission line like that on the right of Fig 1 is not difficult.

The program can also be used in

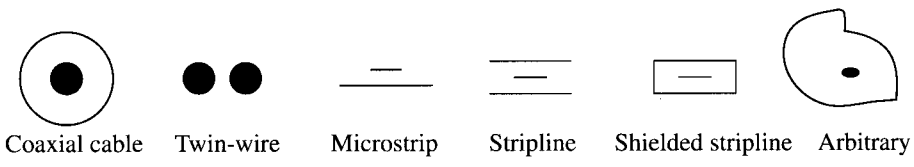

Fig 1—Diagram showing a number of common transmission lines. All, except the twin-wire, are just special cases of the last one and can be analysed by making minor modifications to the program described here.

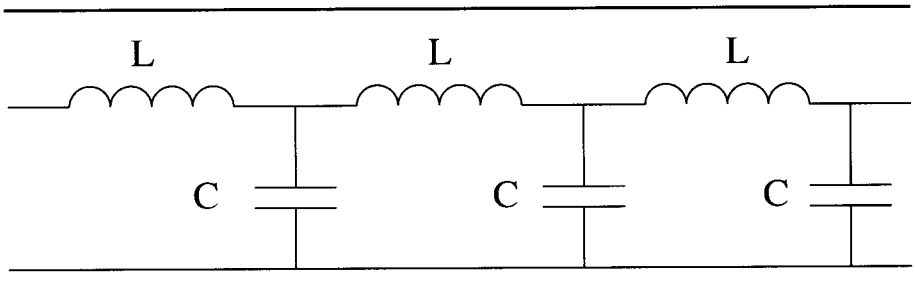

Fig 2—A transmission line has a distributed shunt capacitance and distributed series inductance. Loss terms of series resistance and shunt resistance are ignored.

[1]Notes appear on page 8.

designing directional couplers, RF relays or any other item that contains a transmission line.[6] By being able to calculate the impedance for any position of a center conductor relative to the outer, you can usually design a transmission line for exactly 50-Ω using whatever sizes of inner and outer conductors are readily available. There is no need to machine the inner to have the right size for a 50-Ω line—you simply place the inner conductor the correct amount off-center.

Theory

The complete theory of how the program works is quite complex. Fortunately, neither using the program nor modifying it to handle virtually any transmission line requires a detailed understanding. Here's a simplified description that assumes only one dielectric, although we later extend this to more than one. The transmission line is assumed to have constant dimensions along its length. We assume the outer conductor is earthed, with 0 V on it. We set the inner conductor to a dc voltage of V_0 V. V_0 will be set equal to 1 V here, although it does not matter what voltage is chosen as long as it is nonzero, since its value gets cancelled. Theoretically, if we knew the voltage at every single location over the cross section of the line, we could determine the capacitance per meter of the transmission line. However, there are an infinite number of different locations, and as no computer has an infinite amount of memory, that approach is impossible.

If, however, we cover the transmission line's cross section with an imaginary square grid, there is now a finite number of nodes (corners of the squares). This is shown in Fig 4. We can now store the voltage at every node in a computer. If the grid size is sufficiently small, the voltage will not change much from one node to the next, and the error in not knowing the voltage everywhere will be small. The voltages will be stored in a two-dimensional matrix $V_{i,j}$, where i ranges from 0 to I_{max} and j ranges from 0 to J_{max}. The transmission line's boundary must lie on the grid, so odd shaped conductors will have to be approximated. This is shown in Fig 5 (for the strange transmission line to the right of Fig 1).

We already know the voltage at some points on the grid since the outer is at 0 V dc and the inner at V_0, which is 1-V dc, but the other voltages can be found easily if we accept Laplace's equation.[10] We need not bother ourselves with the details of Laplace's equation, but must accept that to satisfy a discrete version of it, and so calculate the voltage at every unknown node, we just apply Eq 4 over every node, except on the transmission line conductors, where the voltages are fixed at either 0 or 1 V.

$$V_{i,j} = \frac{V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1}}{4} \quad \text{Eq 4}$$

We do this once at every node, then repeat the process again and again. We keep doing this since each time we get a new voltage for the point i,j, it is closer to the true value, but will probably never exactly get there. Hence if we have 1,000 nodes, we might typically apply Eq 4 100,000 times—100 times per node. Eq 4 ensures that the voltage at a node is the average of the voltages at the nodes around it, which ensures that Laplace's equation is satisfied.

If you inspect the program you will note that the two lines that update voltage are equivalent to:

$$V_{i,j}(\text{new}) = r\left[\frac{V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1}}{4}\right] + (1-r)V_{i,j}(\text{old}) \quad \text{Eq 5}$$

where r=1.5. The reason for using Eq 5 instead of Eq 4 is that the latter speeds the program's convergence to the correct value of voltage at every point—it does not affect the ultimate result.

Having found the voltage $V_{i,j}$ at every node, we can find the capacitance per meter of the line at dc. The theoretical basis of this is not trivial, so it is best to accept that the capacitance per meter is related to the voltage summed over the rectangle enclosing the transmissions line's cross section.

$$C_0 = \frac{\varepsilon_0}{2Vo^2} \sum_{i=0}^{I_{max}-1} \sum_{j=0}^{J_{max}-1} (V_{i,j} - V_{i+1,j+1})^2 + (V_{i+1,j} - V_{i,j+1})^2 \text{ F/m} \quad \text{Eq 6}$$

This looks complex, but in fact takes only 4 lines of code to complete.

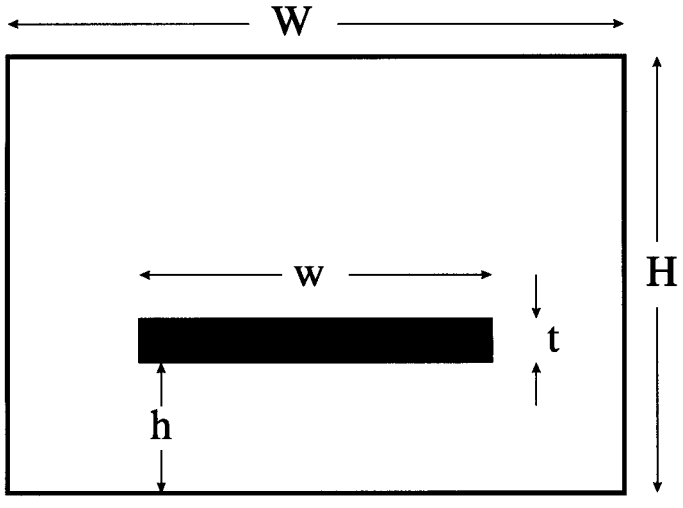

Fig 3—The amplifier whose stripline characteristics were wanted. Unfortunately, the presence of the amplifier case can not be ignored when determining the stripline impedance.

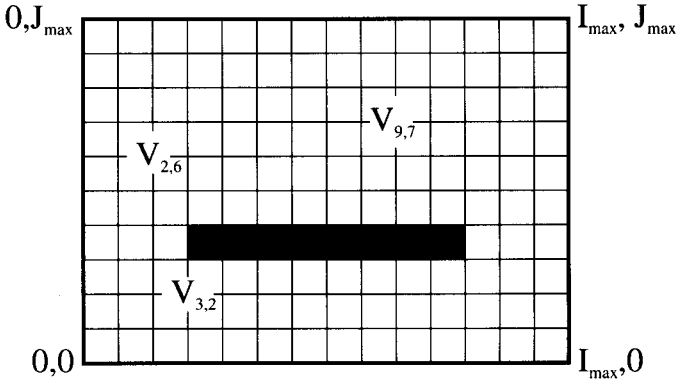

Fig 4—The transmission line cross section is covered with a grid I_{max} by J_{max} (in this case 12×10), and the voltage V_{ij} is evaluated at every node.

Having found the capacitance per meter for an air-spaced line C_0, the inductance per meter L is given by Eq 7:

$$L = \frac{\mu_0 \varepsilon_0}{C_0} \text{ H/m} \qquad \text{Eq 7}$$

where μ_0 is the permeability of free space = 12.57×10^{-7} H/m. If the line is not air-spaced, but is filled with a single dielectric with relative permittivity ε_r (where $\varepsilon_r > 1$), the true capacitance per meter of the line is:

$$C = \varepsilon_r C_0 \text{ F/m} \qquad \text{Eq 8}$$

Having found the distributed inductance and capacitance, the characteristic impedance Z_0 is simply found from Eq 1. If the line is not air-spaced, the velocity of propagation can be found from Eq 2.

A Simple Computer Program

Appendix 1 lists a *finite difference* computer program, written in C, which I called *ATLC—Arbitrary Transmission Line Calculator*. The program as listed solves the problem of Fig 3 for arbitrary values of W, H, w, h and t. It could, however, be modified to solve for any transmission line, such as that in Fig 5, by altering a few lines that set different parts of the transmission line to 0 or 1 V. The geometry of the transmission line would need to be defined in the source code. The program compiles with no problem on Microsoft's *Quick C* version 2.0 for a PC, using a compact memory model, *gcc* version 2.7.2 for a Sun Ultra 1 workstation and *gcc* 2.7.2 for Linux. Since the program uses no special functions, it should compile and run without any hassle on any modern computer. The program's source code (ATLC.C), along with a precompiled executable (ATLC.EXE) is available by anonymous FTP from **ftp.arrl.org/pub/qex**.

Where possible, the program was written in such a way that programmers unfamiliar with the C programming language should be able to convert it to any language they like—FORTRAN, BASIC, Pascal, etc. There are just a few bits that might puzzle such a programmer. First, a C array of dimensions declared as *float arraynname[x][y]*, has locations [0..x–1][0..y–1], which is different from say FORTRAN, where they wound be (1..x)(1..y). The function *atoi()*, converts a string of characters to an integer, *atof()* a string of characters to a floating point number and the function *fabs()* finds the absolute value of a floating point number. The *main()* function essentially reads the values of W, H, w, h and t from the command line, does a few quick checks, then passes them to the calculation routine, which is something I often find convenient for numerical programs. However, you could read W, H, w, h and t in from the keyboard or disk if you prefer.

The program expects the input dimensions to be in units of the width (or height) of a grid square—not millimeters or inches. You must decide the size of grid to use. Generally, if the largest dimension in your problem is y, then you want to allocate 40 or more grid squares to y. So for example, if a stripline is enclosed in a box 400-mm wide × 200-mm high, use 10 mm = 1 grid square, and hence the problem uses 40 grid squares (W=40) by 20 grid squares (H=20). Then recheck the results at double the resolution (5 mm = 1 grid square in this case) and ensure the results are not significantly different. The examples and results later demonstrate this. If you wish to use in excess of a 127 by 127 grid, the array in the program in Appendix 1 will need to be enlarged.

Mixed Dielectric

If the transmission line contains two or more different dielectrics with different relative permittivities, such as a microstrip on a printed circuit board as shown in Fig 6, then the problem becomes more complex. First, any calculation of C performed with a program like this is the dc value. While this does not vary with frequency when there is only one dielectric, it does when there are more than one. Hence even if we find C, L and Z_0 for the dc case, these will be frequency dependant. However, if the frequency is not too high (and this is difficult to quantify), the approximations will be acceptable.

The procedure for finding the characteristic impedance of a multiple-dielectric microstrip line and a FORTRAN IV program to perform this are contained in Dworsky's book. No program will be given, but the basic method is outlined here. The method is:

1) Calculate the voltage distribution across the transmission line, assuming it's air-spaced, as before.

2) Calculate the capacitance per meter C_0, assuming the transmission line has just an air-dielectric, as described before.

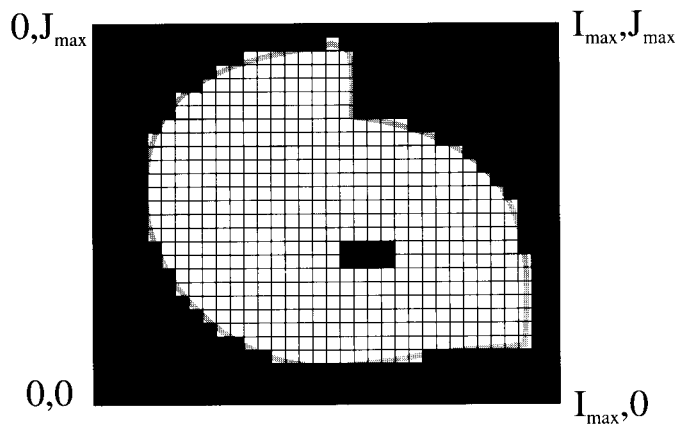

Fig 5—Diagram showing how the odd shaped transmission line on the right of Fig 1 would be approximated on the grid.

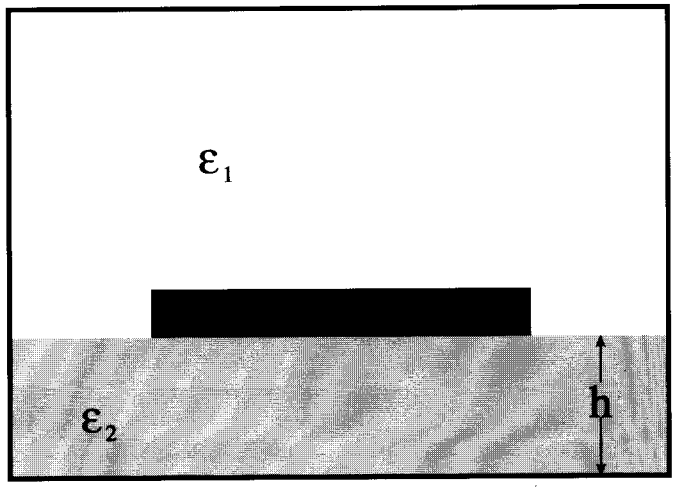

Fig 6—Shielded microstrip with a mixed dielectric. Microstrip line etched on a PC board and enclosed in a case would resemble this.

3) Find the inductance per meter, by using C_0 in Eq 7.

4) Recalculate the voltage distribution across the transmission line using Eq 4, *except* when you are at the interface between the two dielectrics. When at the interface, in order to satisfy Laplace's equation, we must use Eq 9 instead:

$$V_{i,j} = \frac{\varepsilon_1 V_{i+1,j} + \varepsilon_2 V_{i-1,j} + \frac{(\varepsilon_1+\varepsilon_2)}{2}(V_{i,j+1}+V_{i,j-1})}{2(\varepsilon_1+\varepsilon_2)} \quad \text{(when j = h)}$$

Eq 9

where ε_1 is the relative permittivity of the dielectric above the interface—this is probably air, so ε_1=1.0. ε_2 is the relative permittivity of the dielectric below the interface—usually a PC board material.

5) Recalculate the true line capacitance per meter, C, using the fact that C is:

$$C = \frac{\varepsilon_0}{2Vo^2}\sum_{i=0}^{I_{max}-1}\sum_{j=0}^{J_{max}-1}\varepsilon_j\left[(V_{i,j}-V_{i+1,j+1})^2+(V_{i+1,j}-V_{i,j+1})^2\right] \text{ F/m}$$

Eq 10

where ε_j is the permittivity of the dielectric immediately below row j—that is ε_2 where $j < h$ and ε_1 where $j > h$.

6) Calculate Z_0, using L calculated in step 3 and C calculated in step 5.

Since the voltage and capacitance distribution are calculated twice, the program will take twice as long as a single dielectric case.

Effects of Boundaries

This finite difference method fully encloses the inner conductor in a shield, which need not be rectangular. However, it must exist. If the physical problem being modeled is not like this, there is a small problem. For example, to model an air-spaced microstrip (see Fig 1), with no surrounding metal, we have a problem. One way to solve this is to put the boundary at a large distance. Try it at a distance of x, then repeat the calculation at a distance of $2x$. If the results are significantly different, x was not large enough.

Testing the Program

Any numerical program requires very thorough testing as, unlike many programs such as games and word-processors, data can look believable even though it is total rubbish. Some people may be skeptical since the theoretical basis of the program has not been rigorously justified here—although it is in Dworsky's book. To test the program, a number of transmission lines were analysed under conditions where the results could be checked by other means. For example:

Test 1) The characteristic impedance Z_0 of a thin (t=0), air-spaced (ε_r=1) microstrip of width 5 mm and height 5 mm was calculated using an empirical formula (Note 1) and found to be 126.15 Ω. The finite difference program was then set to find the impedance of a shielded stripline, using 1 grid point = 0.5 mm. Therefore the line was 5/0.5=10 grid points wide (w=10) and 5/0.5=10 grid points high (h=10). This was enclosed in a shield 63-mm wide (therefore W= 63/0.5=126) × 63-mm high (therefore H=63/0.5=126). The shield being so large compared to the stripline, its effect should be minimal and so the results should be similar. The program terminated after performing 1140 iterations in 954 seconds on a 25-MHz 486 PC and returned a Z_0 of 123.2 Ω, which is acceptably close, with a difference of –2.3%. Of course, the shield, even at this distance, may affect Z_0 by a few percent. To check this for certain, the program was run with a 0.25 mm = 1-grid-point resolution (therefore w=20, h=20, t=0) and a screen 300-mm wide (W=300/0.25=1200) by 300-mm high (H=300/0.25=1200) on a fast Sun Ultra 1 computer. Z_0 was then 124.8 Ω, a difference of just –1.1% from the expected value. This could be due to errors in the empirical formula, which is not 100% accurate.

Test 2) The characteristic impedance Z_0 of an air-spaced shielded stripline with a width of 19 grid points (w=19), 1 grid point thick (t=1) placed centrally along both the x and y axes, in a shield of 99 (W=99) × 49 (H=49) grid points was calculated. Since it was placed centrally, h=24. According to Dworsky, this has a theoretical impedance of 109 Ω. The program took 104 seconds on the 25-MHz 486 to perform 400 iterations and return Z_0 =108.7 Ω, an error of only –0.28%.

Results

For the amplifier being designed, the anode line needed to be at least 90 mm above the ground for practical reasons—to clear the valve chimney. According to the *Txline* program mentioned earlier, choosing a stripline 160-mm wide and 1-mm thick, and assuming no surrounding metal case, the line impedance would be 94.5 Ω. This would have been acceptable as it exceeded the 81 Ω minimum needed. However, this ignored the presence of the metal case, which was 200-mm wide by 290-mm high.

For a finite difference simulation of the problem, a scale of 5 mm = 1 grid point was used, as this gave convenient numbers while also executing quickly. Hence the case width (W) was set at 290/5=58 grid points, the case height (H) to 200/5=40 grid points, the stripline width (w) to 160/5=32 and the stripline height (h) to 90/5=18. The stripline thickness should have been 1/5 = 0.2 grid points, but this is clearly impossible, so this was rounded to the nearest integer: 0. Below is the output of the computer simulation. Note that Z_0 slowly converges upwards, as the program makes better and better estimates of the true voltage distribution across the transmission lines cross section.

```
C:\2D_ATLC>atlc 58 40 32 18 0
 10 c=77.91pF/m l=142.85nH/m Zo=42.820003 Ohms
 20 c=60.06pF/m l=185.30nH/m Zo=55.544252 Ohms
 30 c=53.14pF/m l=209.45nH/m Zo=62.782017 Ohms
 40 c=50.13pF/m l=222.03nH/m Zo=66.553549 Ohms
 50 c=48.81pF/m l=228.01nH/m Zo=68.347540 Ohms
 60 c=48.23pF/m l=230.76nH/m Zo=69.170590 Ohms
 70 c=47.97pF/m l=232.01nH/m Zo=69.544214 Ohms
 80 c=47.85pF/m l=232.57nH/m Zo=69.713328 Ohms
 90 c=47.80pF/m l=232.82nH/m Zo=69.789547 Ohms
100 c=47.78pF/m l=232.94nH/m Zo=69.823501 Ohms
110 c=47.77pF/m l=232.99nH/m Zo=69.838244 Ohms
120 c=47.76pF/m l=233.01nH/m Zo=69.844314 Ohms
130 c=47.76pF/m l=233.01nH/m Zo=69.846544 Ohms
140 c=47.76pF/m l=233.02nH/m Zo=69.847136 Ohms
```

The 140 iterations took 17 seconds on the 25-MHz 486 PC. To check that the data was reasonable, the program was rerun using double the resolution (2.5 mm between grid points). This took 272 seconds, but the result, Z_0 = 70.1 Ω, differed by less than 0.4% from the previous, much faster run. Hence there was no appreciable difference except a 16-fold increase in execution time. A much more accurate calculation, using a 280 × 200 grid, where the thickness of the stripline could be taken into account properly (t did not have to be set to zero), showed the correct result to be nearer 68.6 Ω but took 9111 s (just over 2.5

hours). Hence a calculation performed in 17 seconds has an error of probably less than 2%, which should be of adequate accuracy.

The results show that the amplifier case reduces the impedance of the stripline from 94.5 Ω to 68.6 Ω, below the minimum acceptable value of 81 Ω. Since the line impedance is altered by the amplifier case, the resonant length of a line is also altered. The program allows many "what if?" calculations—what if the line was narrower, the case bigger, the line thicker, etc.

Computational Problems

The program is computationally intensive and will therefore be slow if the highest accuracy is desired, although such accuracy is usually not necessary. The program's execution time increases with the square of the array area used for calculation, so doubling the resolution means a 16 times increase in execution time. A computer with some built-in floating point hardware is essential for this—a fast 386 with a 387 math chip, or better still, a 486 or Pentium machine. This is not the application to demonstrate that your Commodore 64 or Sinclair Spectrum was worth hanging on to! The amount of memory used by the program is determined by the array size used for storing the voltages. A 127x127 array of single-precision floating point numbers (4 bytes each) uses just under 64 kbytes, which is the limit on some PC compilers for a single array. Using a decent compiler will allow much bigger arrays, but the execution time will slow. However, 127×127 seems adequate for most applications. The way to be sure you are using sufficiently fine resolution is to try doubling it to see if the results differ significantly. If you have the memory, it would be wise to use double-precision numbers for the array to reduce rounding errors.

Discussion

Itoh has compared a number of different numerical methods for analysing passive microwave structures and finds the finite difference method to be the slowest and use the most memory![11] However, it does have two distinct advantages over other methods. First, it is very general, and second, the problem requires no pre-processing. These advantages far outweigh its disadvantages for amateur use.

If the program was written as a Windows application, it would be possible to define the transmission line's outline with a mouse, which would be easier than the method used here for complex shapes. Different colors could be used for different dielectrics.

A three-dimensional (3-D) version of this program could be written, which would make analysing transmission lines with discontinuities along their length possible. However, this is probably not a viable option for current home computers—but it will be in a few years. The memory requirements for a 3-D model would not be excessive even by current standards—a 100×100×100 array of double-precision numbers needing only about 8 Mbytes of RAM—but would probably take about a week to execute on a fast 200-MHz Pentium processor.

Notes:

[1]Weiner, K. DJ9HO, "The UHF Compendium," Parts 1 and 2, Published by Verlag Rudolf Schmidt, Germany, pp 35-40. Note: the formula in this reference has a typographical error—it shows Z_0 being proportional to $1/\varepsilon$, when it should be proportional to $1/\sqrt{\varepsilon}$.

[2]Fisk, J. R., W1HR, "Microstrip Transmission Line," *Ham Radio*, pp 28-37, January 1978.

[3]Meade, E. L., K1AGB, "A 2-kW PEP Amplifier for 144 MHz," Part 1, *QST*, pp 34-38, Dec 1973. Part 2, *QST*, pp 26-33, Jan 1974.

[4]McMullen, T. F. and Tilton E. P., "New Ideas for the 2-Meter Kilowatt," *QST*, pp 24-30, Feb 1971.

[5]Mandelkern, M., "A Luxury Linear," *QEX*, pp 3-12, May 1996.

[6]Jessop, G. R., G6JP, *VHF UHF Manual*, 4th Edition, Radio Society of Great Britain, 1983.

[7]Sutherland, R. I., W6UOV, "Design Data for a Two-Kilowatt VHF Linear," *Ham Radio*, pp 6-13, March 1969.

[8]Robrish, P., "An Analytic Algorithm for Unbalanced Stripline Impedance," *IEEE transactions on microwave theory and techniques*, pp 1011-1016, 1990.

[9]Dworsky, L. N., *Modern Transmission Line Theory and Applications*, Chapter 9, John-Wiley, 1979.

[10]Ramo, S., Whinnery, J. R. and Van Duzer, T., *Fields and Waves in Communication Electronics*, 2nd edition, Wiley, 1984.

[11]Itoh, T., *Numerical Techniques for Microwave and Millimetre-Wave Passive Structures*, Wiley, 1989.

Appendix 1—Source Code for ATLC.C

Note: Some compilers may need the line '#include <stdlib.h>'—others may not.

```
#include <stdio.h> /* ATLC - Arbitrary Transmission Line Calculator. ver 1.0 */
#include <math.h>        /* By D. Kirkby G8WRB.  Compiles okay with          */
#include <stdlib.h>     /* Microsoft  Quick C, version 2.0 and GNU C.       */
#define Imax 126         /* Voltage array size will be 0..Imax               */
#define Jmax 126         /* ie v[0..Imax][[0..Jmax]                          */
float v[Imax+1][Jmax+1]; /* Declare an array to hold the voltages            */
void arbitrary_transmission_line(int W, int H, int w, int h, int t);

void main(int argc, char **argv) /* Read parameters from command line here   */
{
    int W, H, w, h, t; /* integers for number of grid squares to use. */
    if((argc!=6) ) /* Check the number of command line arguments are correct */
    {
            printf("Usage: %s W(shield) H(shield) width height thickness\n", argv[0]);
```

```
            exit(1); /* Exit - program called with wrong number of arguments */
    }
    W=atoi(argv[1]); /* Read shield width (in grid points) from command line.    */
    H=atoi(argv[2]); /* Read shield height (in grid points) from command line.   */
    w=atoi(argv[3]); /* Read strip width (in grid points) from command line.     */
    h=atoi(argv[4]); /* Read strip height (in grid points) from command line.    */
    t=atoi(argv[5]); /* Read strip thickness (in grid points) from command line. */
    if((W>Imax)||(H>Jmax)||(h+t>H-1)||(w>W-2)||(h<0)||(t<0)||(W<0)||(H<0)) /* Basic checks */
    {
            printf("Sorry - one of the arguments is silly - too big, too small ?\n");
            exit(2);
    }
    arbitrary_transmission_line(W,H,w,h,t); /* Calculate L, C and Zo     */
}

void arbitrary_transmission_line(int W, int H, int w, int h, int t)
{
    double Eo=8.854e-12, Er=1.0, mu=12.57e-7, c, l, Zo, vnew,r=1.5, c_old;
    int i, j, k=0, done=0;
    for(i=0;i<=W;i=i+1)    /* Zero the voltage array. Its essential that the  */
            for(j=0;j<=H;j=j+1) /* outer is at 0V, but desirable for everywhere to */
                    v[i][j]=0.0;     /* start at 0 V. */
    for(i=(W-w)/2;i<=(W-w)/2+w;i=i+1) /* Put stripline in centre of x axis, */
            for(j=h;j<=h+t;j=j+1)          /* and between h and h+t on the y axis,*/
                    v[i][j]=1.0;                /* then set stripline there to 1 V */
    do{ /* Set up a relaxation loop, to find the voltage at every point */
            k=k+1; /* increment the counter used to count the iterations */
       for(i=1;i<=W-1;i=i+1) /* Data at i=0 must stay fixed at v=0 */
                    for(j=1;j<=H-1;j=j+1) /* as this is a 'boundary condition' */
                            if(v[i][j]!=1.0) /*ie. don't do this where the stripline is */
                            {
                       vnew=r*(v[i+1][j]+v[i-1][j]+v[i][j+1]+v[i][j-1])/4+(1-r)*v[i][j];
                                    v[i][j]=vnew; /* New voltage is calculated */
                            }
            if(k%10==0) /* Now we have v distribution we find C every 10 iterations */
            {
                    c_old=c; c=0.0;
                    for(i=0;i<=W-1;i=i+1) /* Sum v over cross-section to get C, which  */
                          for(j=0;j<=H-1;j=j+1) /* is easy for a rectangular cross section */
                               c=c+pow(v[i][j]-v[i+1][j+1],2.0)+pow(v[i+1][j]-v[i][j+1],2.0);
                    c=c*Eo/2.0; /* Find capacitance - only correct if air-spaced */
                    l=mu*Eo/c;  /* Calculate the line inductance - always correct */
                    c=c*Er;          /* Correct the capacitance if line has a dielectric */
```

```
                    Zo=sqrt(l/c);     /* Calculate the characteristic impedance */
                    printf("%5d c=%.2lfpF/m l=%.2lfnH/m Zo=%lf Ohms\n",k,c*1e12,l*1e9,Zo);
                    if(fabs(c_old-c)/c < 0.00001) /* Until they differ by < 0.001 % */
                              done=1;  /* Little change in calculated value of C - so we finish*/
                    else
                              done=0; /* Large change in calculated value of C - lets continue */
          }
    }while(done==0); /* Repeat for until the capacitance has converged */
} /* End line of program - line 66 */
```

Notes - added on 24/12/1999 (after publication)

1) There is an error in equation 3 - the ε_o should not be there.
2) An updated (Windows 95/98/NT) version of the programme may be found on my web site - http://www.medphys.ucl.ac.uk/~davek