How to Start Modeling Antennas using EZNEC

Greg Ordy, W8WWV
CTU, Contest University
Dayton, May 19, 2011
Version 1.01

This modeling tutorial is the companion document to a slide presentation given at a Contest University session on May 19th, 2011.

Table of Contents

Introduction

The target audience for this material is the antenna enthusiast who has antenna knowledge and experience, but has thus far avoided integrating modeling into their design and thinking process.

Modeling has its own terminology, jargon, rules, quirks, tricks, guidelines, restrictions, limitations, and frustrations. Some aspects of antenna modeling are immediately familiar to a person with antenna design experience. Other aspects take time to understand, appreciate, and master.

A number of very good modeling tutorials and practical model examples exist and the ones that I am aware of will be listed in the next section. The approach taken here will be to go into painful detail in the running of the EZNEC program while assuming a moderate level of general antenna knowledge.

As with so many activities in life, it often comes down to working through a few examples, and getting past those pesky hurdles. My hope is that some of these examples and background information will provide illumination on that initial journey. If you have any questions or comments about the material in this document, I can be reached by email at ordy@seed-solutions.com.

Although I've never made a single contact on an antenna model, I would not want to design or install an antenna without at least a quick check of a model to help frame my expectations and crosscheck my measurements.

References and Links

The EZNEC®[1] software, created and marketed by Roy Lewallen, W7EL, can be obtained from www.eznec.com. The package comes with a number of example models, and there is an extensive indexed *Help©*[2] facility with information on all aspects of the program. The version used in this document is EZNEC+ V5.0.

The name that many people associate with modeling is L.B. Cebik, W4RNL (SK). L.B. wrote extensively about modeling and modeling tools. Although now a silent key, L.B.'s personal web site remains at www.cebik.com. His work at that site is now maintained by AntenneX, www.antennex.com. It is necessary to register to access the site, but there is no fee involved. It is well worth it to register and read through the many pages.

Antennex, at www.antennex.com, maintains a large amount of antenna and antenna modeling information. Although they have an online journal that has a subscription cost, there is also a guest section that is free. It does appear as if tutorial or teaching material created by W4RNL has been placed in products with fees. If you register as a guest at Antennex.com you can access a large number of articles, including models and descriptions of models.

From time to time the ARRL appears to offer an online course on antenna modeling that was written by L.B. Cebik. Over the years, many articles have appeared in QST and QEX that refer to antenna modeling. Some links can be found on this page: http://www.arrl.org/antenna-modeling.

The *ARRL Antenna Book* makes frequent reference to modeling, and many of the models presented in the book are included on the CD that comes with the book. The most recent 21st edition includes a free version of EZNEC, called EZNEC-ARRL. Although that might sound appealing, it appears to be nothing more than a frozen-in-time version of the free version of standard EZNEC, and version 3 at that. For normal modeling, the 20 segment limit applies. That limit can be exceeded for the model files provided with the book. As best as I can tell, it's probably better to just download the most recent free EZNEC, unless models included with the book are of particular interest.

Chapter 4 of the ARRL Antenna Book is titled: *Antenna Modeling & System Planning*, and contains very useful background information including a modeling package comparison as well as a good tutorial.

In the most recent and last (5th) edition of *ON4UN's Low-Band DXing*, John Develdere, ON4UN, has supplied most all of the models used in the book on the CD that is included with the book. They are EZNEC models. Chapter 4 of the book is on the topic of *Antenna Design Software*, and includes a history of modeling as well as additional references and links.

[1] EZNEC is a registered trademark of Roy W. Lewallen.
[2] The EZNEC programs and Help manual are copyright © 2000-2007, Roy W. Lewallen

Dan Maguire, AC6LA, has an interesting web site (www.ac6la.com) with a number of programs that work in conjunction with modeling software. A lot of Dan's work deals with transmission lines, and he is a true expert in that area. If you have transmission lines issues, from theory through practice, talk to Dan.

Rudy Severns, N6LF, (http://www.antennasbyn6lf.com) is a frequent author and contributor to amateur radio publications. Rudy's work is highly regarded, and he backs up his work with models and/or measurements. He often uses modeling in his work, and his site covers a number of interesting and contemporary topics.

The modeling package **4nec2** (http://home.ict.nl/~arivoors) is free, and has a number of users and followers. One interesting feature that it has is the ability to read EZNEC files and run them, as well as create NEC input decks from the EZNEC model. From time to time, this can be quite handy, especially if you find yourself starting to use several different modeling packages. Some features introduced with EZNEC version 5 are not currently supported and can not be translated and simulated. These include features such as Virtual Wires, transformers, and lossy transmission lines.

The Wikipedia has an article about the history of the NEC engines at: http://en.wikipedia.org/wiki/Numerical_Electromagnetics_Code.

Although more about the engine than modeling with the engine, the NEC-2 unofficial home page is located at: www.nec2.org.

Finally, it never hurts to simply search the Internet for models related to a particular antenna design. Although there are no guarantees of quality, there is a lot of information out in cyberspace, much of it very good.

There are many authors who have written on the topic of modeling, and they should be included on any useful list. I apologize to those I have omitted, and please feel free to contact me and I will update this list.

Why EZNEC?

The various forms of the EZNEC[3] product family are probably the most popular modeling packages used by the amateur radio community. At the entry level, there is a free package with all of the features of the commercial version, but limited in the complexity of the model.

The examples in this document were simulated with the EZNEC+ V5.0 package.

[3] www.eznec.com

In some cases, the information will be very specific to EZNEC, and you should not expect to see the identical behavior in other antenna simulators. After all, this is a tutorial about modeling with EZNEC, not general modeling.

EZNEC and other similar antenna modeling packages, with few exceptions, are *wrappers* for the NEC (Numerical Electromagnetics Code) engines. These engines date back to the 1970's, and were written in Fortran to run on mainframe computers. Going along with that era, these programs accept input in the form of a punched card deck, and the outputs are a large text file full of numbers suitable for printing on a *line printer*, with fan fold 14" wide tractor feed paper.

EZNEC does a lot of work on our behalf to hide all of that mess, and let us work with a contemporary GUI with many powerful features.

Many of the concepts, terminology, and jargon used in EZNEC and similar programs come from NEC. This is good news because this means that once you learn one NEC-based program, you can usually pick up others quite quickly.

Each NEC shell or wrapper has its own strengths and weaknesses, and don't be surprised if you collect several as the years go by.

Since the NEC engines are *ancient* in terms of computer evolution, they might seem to be static, out of data, and maybe even *falling behind*. In some ways, they are! I don't believe that the core engines have been officially enhanced in decades. Fortunately, modeling tools can continue to advance by building upon features long present in NEC but unused or at least underused.

Just because NEC is effectively frozen in time, don't think that NEC-based tools are – they are not.

If you tried modeling several years ago, and were frustrated by not finding some capability, it may be time to check again.

Putting Modeling in Context

I think of modeling as one part of a three part approach to successful antenna design and implementation. The three parts are:

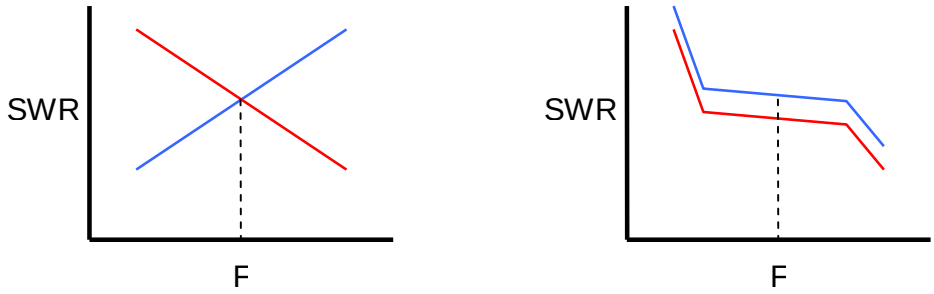
- Theory (Math)
- Modeling
- Measurements

Each of these can be used to drive a design process, but they also serve to crosscheck each other. The idea is to create a solid and broad base that provides results we can be confident in. I sometimes think of "*theory"* as "*math*", so that I can remember the *3M's*.

If traditional antenna theory says that the impedance of a half wavelength dipole in free space is around 72 Ohms at resonance, and a model taking into account the wire diameter and the height about ground reports 65 Ohms, and the measured value is 61 Ohms, we can be more confident that we have built a form of what we modeled, and what is close to a textbook half wavelength dipole. If the model reports 150 Ohms, we might want to double check the model. If we measure 40 Ohms, then we need to check the measurement device and the model, and even wonder if we are applying the correct textbook expectations.

As part of comparing models and measurements, I always try to check the *tracking across the band*, as well as data at a single frequency. The problem with data at a single frequency is that it can accidentally create a false sense of security. As the saying goes: *even a stopped clock is right twice a day*.

Consider the following two example SWR comparisons between a model and a measurement.



If all we measured was the SWR at a single frequency, **F**, then we might think that the antenna on the left (blue) matches the model (red) much more closely than the antenna on the right. If we step back and look at a wider sweep, it's clear that the left antenna measurements look nothing like the model; except for the accident that the SWR is the same at the one frequency we happened to check. Although I used SWR in this example, the parameter might be impedance, gain, front to back ratio, or any useful characteristic of an antenna.

As I model an antenna, I'm always on the lookout for *signatures* and *trends* in characteristics that can also be measured or observed in the field. In the above example on the right, we see that there are two frequencies where the SWR curve changes slope. If I can find those in my measurements, and they are close to where they are expected, then I'm more confident in my implementation. In some cases, these sorts of signatures are found outside of the formal band edges.

Modeling Expectations

Optimizers

Modeling with EZNEC tends to be more *what if* as opposed to *find me the answer*. As an example, let's say that we want to determine the length of a 40 meter dipole made from #12 copper wire with the ends at 35 feet high and the center at 45 feet. The target frequency is 7.150 MHz, where we want the reactance to be zero Ohms.

With EZNEC, you answer that question by modeling a set of lengths that converge on an acceptable answer. You could call that *cut and try*, or even *trial and error*.

Some folks seem to think that modeling software should provide only the final answer, and do it in a single step after the question is appropriately asked.

There are indeed programs that provide that approach to modeling. They are generally called *optimizers*. As far as I know, EZNEC is not one of them. It simply provides results for the current model, and if you want to alter to model to come closer to some design goal, you'll have to make the changes manually and then run the model again.

My experience has been that except for the most aggressive and complex designs, the *cut and try* approach is more than acceptable. Convergence to the design goals usually happens in a few quick steps, and during that process valuable insight is gained about the design. In the previous example, it might be a sense of how many KHz resonance shifts for each inch of antenna length. That's certainly good to know if you are actually tuning the antenna out in the field.

If you do anticipate that you might need to run many models (hundreds to thousands) to find *the answer*, then you might want to consider an optimizer. Even when using an optimizer, some *hand modeling* is desirable for insight and developing a sense of how the antenna performs in the overall design space and frequency range.

One of the concerns about optimizers is that they converge and stop at a *local* design goal that is not the best in the overall *global* design space. Having knowledge about the space is insurance against falling into those traps. I think it's reasonable to say that an experienced modeler will get more out of an optimizer than a beginning modeling. Even if you want to or need to rely upon optimizers, it is probably a good idea to become a proficient modeler on your own.

Trends and Directions

I often view modeling as providing answers about trends and directions as opposed to absolute numbers. To highlight this idea, let's say that I model two variations of a design, and one reports a maximum gain of 4.0 dBi, and the other reports 5.2 dBi.

What I would take from these results is that the second design has *about* 1 dB more gain than the first. To expect the 4.0 and 5.2 dBi gain values to be spot on accurate in the actual antenna is probably going to lead to disappointment. It's not because modeling has some flaw or inaccuracy, but rather that models almost never take into account all of the environmental variables that surround real antennas in the real world. This is especially true as the antenna becomes closer to ground.

In comparing designs, I usually think in terms of words like *more* or *less*, *higher* or *lower*, or *narrower* or *wider*. In other words, I'm looking at the relative differences between antennas, not absolute gain, front to back ratio, RDF, bandwidth, or other antenna characteristics.

An exception would be impedance, where we might be targeting specific values such as 50 Ohms. Since impedance is becoming easier and easier to measure with accuracy, we can double check our results when it matters. Because most transmission lines are 50 Ohms, it is important that we come acceptably close.

A Few Words on NEC

As mentioned earlier in this document, EZNEC, and most all antenna modeling packages, are based upon a simulation engine family broadly named *NEC* – the Numerical Electromagnetics Code. This is usually pronounced as *neck*, and it is not the same as the National Electrical Code, which is the code that most think of as being about safe house wiring. It is also not related to the N.E.C. which is the *Nippon Electric Company*.

The references provided earlier, especially the Wikipedia page and chapter 4 of the ARRL Antenna Book provide detailed historic information.

As best as I can determine, NEC-1 does not exist. As you might guess, NEC-2, NEC-3, and NEC-4 are more recent versions, each an update to the previous. NEC-4, the most recent, released in 1992, is now almost 20 years old. NEC-3, although released and used in some products, did not survive.

When talking about NEC, there are only two practical choices – NEC-2 and NEC-4. As far as I know, NEC-4, in all areas of accuracy, is considered to be superior to NEC-2. That begs the question, *why is NEC-2 still being used?*

NEC-2 has the desirable property that it is free, in the public domain, can be distributed and used without a software license, and can be used around the world.

NEC-4 still requires a specific software license, and the cost is around $300 at this time (2011). In addition, there are export restrictions on the engine, so it cannot be sent to certain countries no matter how much money is available.

Since NEC-4 is considered to be more accurate, it is often viewed as the *professional* form of NEC, and that means that the wrappers that surround it might offer more features, and also cost more money. Note that there are two costs here – the cost of the NEC-4 engine, **and** the cost of the wrapper software.

So, the bottom line is that NEC-2 tools are several hundred dollars less expensive than NEC-4 tools, if not free. EZNEC, using NEC-2 for example, has a free version, a version around $90, and a version around $140 (2011 prices). The EZNEC+ V5.0 that I use is the $140 package. But, if you were to move up to the versions that can use NEC-4, you are stepping up to a little less than $1000. Other modeling packages with both NEC-2 and NEC-4 price points have a similar two-tier structure.

I think we can now appreciate that unless the NEC-4 features and accuracy are absolutely needed, or, money is no object, the vast majority of folks doing antenna modeling around the world use NEC-2.

When the personal computer arrived on the scene, a new engine was created for the PC, and it was initially written in Basic as opposed to Fortran. It was called MININEC. Because it could run on hardware that we could all afford, it was initially quite popular. It has been developed and enhanced over the years, and there are commercial tools that use the MININEC engine.

There are areas where MININEC is considered more accurate than NEC-2, and areas where NEC-2 is considered better. So, there are good reasons why both engines should still exist.

Over time, the PC grew into a more capable computing platform, and it was possible to move the original NEC-2 and NEC-4 mainframe code to the PC.

EZNEC, as far as I know, is exclusively based upon NEC-2 and NEC-4. But, the term *MININEC* still seems to come up from time to time. That is not a reference to the MININEC engine, but to a type of ground model that is associated with the MININEC engine, and was then adopted by versions of NEC-2.

This is a good time to mention what are considered to be the primary weaknesses in NEC-2. NEC-2 is considered to have two main *issues*[4]:

1. Modeling tapered elements (stepped diameter), such as made out of telescoping aluminum tubing.
2. Modeling wires that are very close to ground, lying on the ground, or buried in the ground.

At first blush, these look like serious problems, since in the case of the first problem it is hard to model most Yagi and beam antennas, or any antenna made from telescoping aluminum tubing. In the case of the second, it seems hard to model a vertical antenna with a ground radial system.

Because these are both very common and important types of antennas, NEC-2, at least in EZNEC, has been augmented and enhanced to work around these problems.

In the case of the telescoping tubing, an algorithm called *stepped diameter correction* has been added to EZNEC. This approach converts a set of stepped lengths into a single length that is considered to be equivalent. The actual model uses the equivalent dimensions, not the original. The details of this approach are described in the EZNEC Help documentation[5]. I should mention that I recently had the opportunity to work on some very large Yagi designs, and I was able to see differences between NEC-2 and NEC-4 results for the same model, and to confirm via impedance measurements that NEC-4 was closer to the measured values. Note that these are feed point impedance differences, which appear to be the place where the engines mainly diverge. For the casual and even serious designer, NEC-2 with the stepped diameter correction in EZNEC is probably more than good enough. If you want to be a world class designer of antennas with stepped diameters, you probably need to spend the money and move up to NEC-4.

The issues with modeling ground radials, including buried radials, can also be solved by using NEC-4. If you want to do serious work with ground radial systems, then using NEC-4 is strongly suggested. If you don't want to spend the money, you can still achieve good results using EZNEC/NEC-2. One approach, the one that I use most all of the time, is to select the *MININEC-type* ground model, and then use a series resistor at the base of the antenna to represent expected ground loss. This popular approach that is described in the Help documentation avoids the problem of having to model the ground system at all, and a vertical is not much more than a wire touching the ground with a resistor at the base. This is especially convenient for modeling vertical arrays, where part of

[4] There are some secondary issues too. For example, NEC-2 has a difficult time modeling very small loop structures. The issues presented here tend to come up most often.
[5] And see *Physical Design of Yagi Antennas*, David Leeson, W6QHS, page 8-18 (available from the ARRL)

the design task is to locate the verticals. Who wants to drag around radial wires in addition to the verticals in their models?

Note, however, that this is the *MININEC-type* ground in EZNEC, **not** the MININEC engine.

There is at least one more issue that lingers from the days of the early personal computers. The NEC engines obviously perform a large number of computations on real or floating point data (as opposed to integer data). At the computer hardware level, there are two popular data formats for representing floating point data. One is the 32-bit format that is often called *single precision*, and the other is the 64-bit or *double precision* format. In the early days of PCs (remember the optional *floating point coprocessor chip*?), there was a performance and program size cost to using the double precision format. Double precision programs were larger and slower.

These differences led to the creation of NEC engines that were compiled for the single or double precision data representation. While there still might be situations where using single precision makes sense, the advances in computer speed and capacity largely remove this as an issue. In the case of many antennas, the difference in precision is not going to be a factor in the results. Still, unless you are forced at gun point to use single precision, use double precision to gain the higher accuracy potential.

If you want to move a model from one package to another, you may have some choices. One approach that will always work is to simply manually transfer the model into the new package while looking at the old package model. If there are a lot of wires and complexity, this can be tedious and error prone, although it always works.

Because the wire definitions are usually the longest part of a model, some packages have facilities to import and export just the wire portion of the model. This can also be useful if you are generating your wire coordinates through a design program.

Another approach is to see if you can exchange NEC *card decks*. As mentioned before, the NEC engines maintain the abstraction of a deck of punched cards for models. Some tools let you import or export NEC data. Although EZNEC versions other than EZNEC/Pro do not support this, there are still some options. One *trick* I've used from time to time is to work in EZNEC, then import that ***.EZ** model file into 4nec2, and then have 4nec2 create a NEC-format card deck image file. From that point, I can move to any tool that consumes the low level NEC format.

Modeling and Computer Performance

When the NEC engines escaped the Fortran mainframe computer and arrived on the lowly PC, performance and capacity were valid concerns. Do you remember the days of magical memory numbers such as 640K, and *extended* and *expanded* memory?

The practical impact on models was to encourage the use of the minimum number of Wire segments, and to analyze antenna patterns as slices through single azimuth or elevation planes. If you dig out antenna articles[6] from that time that used antenna models, you'll find references to taking many minutes to run a single model.

Fortunately the *dark ages* of personal computing did not last too long, and we now sit in front of demand paged virtual memory machines with multiple core CPUs running at several GHz with several gigabytes of physical memory[7]. There is very little performance penalty[8] in using double precision (64-bit) floating point calculations as compared to single precision (32-bit) calculations. File and memory sizes are rarely an issue.

The good news today is that most all antenna models simulated on EZNEC can produce a full 3 dimensional pattern analysis in just a second or two. Capacity is usually limited by the modeling tool product definition, not the time it takes to sit and wait for a result.

While there are times when using too many Wire segments is a bad idea, it is no longer necessary to waste a lot of time trying to minimize segment count, potentially lowering model accuracy. It's also possible to quickly produce 3D patterns that fully reveal the total pattern, as opposed to single slices that might hide important details.

An aspect of this computer performance bonanza is that it's now easy to quickly model antenna performance across an entire amateur band, not just a single frequency. This can reveal nasty performance surprises that can be easily missed if you are just looking at a single target frequency. Amateur radio is somewhat unique in that antennas, especially transmitting antennas, usually function across an entire band, not just a single assigned frequency.

---

[6] *Modeling HF Antennas with MININEC – Guidelines and Tips from a Code User's Notebook*: Dr. John Belrose, VE2CY, ARRL Antenna Compendium #3, page 156.

[7] The performance improvement in modeling is at least partly due to improved compiler technology and some hand optimization of the NEC programs. To the best of my knowledge, folks engaged in these efforts have always tried to preserve the NEC semantics.

[8] I measured a 6% increase in execution time with the double precision as opposed to single precision engine for a model sweep that took 3 minutes of wall clock time.

Designers used to have to settle for modeling an antenna at a single frequency looking at a few carefully selected azimuth or elevation pattern slices. Now, inspecting and evaluating the full 3D antenna pattern across an entire band is not a big deal.

If you tried modeling several years ago, and were frustrated by the practical capacity or complexity limits of a model, it may be time to check again.

Examples

It's finally time to get into the EZNEC program and some examples. EZNEC is a Windows®[9]-based program that attempts to follow the common GUI *style guidelines* that make it possible to quickly learn how to operate a new program because it generally operates like other Windows programs. Because of differences in the underlying applications, deviations from the most common guidelines may make sense.

I would like to cover two of these areas where I have found EZNEC to be a little uncommon, and perhaps for that reason, confusing (at least for me).

File Open/Save As

With many Windows programs, the *File* menu includes the commands *New*, *Open*, *Save*, and *Save As*. Combined with these commands is a recently used file list. By the way EZNEC model files have a .EZ suffix. When the program is installed, a file association is registered with Wndows, so if you Open a *.EZ file, you will end up in EZNEC. EZNEC does accept or create several other file extensions for data related to models.

EZNEC uses what I would call the *Open*, *Save As*, and *LAST* paradigm. When EZNEC starts, it automatically opens a model file named LAST.EZ. As the name suggests, this is the state of the model when EZNEC last terminated. So, whenever you execute EZNEC, you will start off exactly where you left off in terms of the model contents. EZNEC apparently updates LAST.EZ continuously, with every model change, or when it encounters a fatal error. What this means is that it is nearly impossible to lose a model due to a program or system crash. I have had EZNEC crash on me from time to time, but I've **never** lost one keystroke of work because the next time I start the program, my model will be where it was at the time of the crash.

The only file that EZNEC automatically opens or saves is the special name LAST.EZ. It opens it on start up, and it saves it at the end, and after all model changes.

[9] Windows is a registered trademark of Microsoft Corporation.

There is not a *Save* command, usually the *File->Save* menu command, or, the Control-S (*Ctrl-S*) keyboard command. Because the current model state is always saved to LAST.EZ upon program exit, the program does not prompt you to save your work if there is unsaved work when you terminate the program.

There is no *New* command. This means that the concept of the *empty* model, or model without content does not exist. Said another way, EZNEC will not allow you to save into a file a model that cannot be executed. An example is a model without a Source. During the running of EZNEC you can delete all Sources, but you can't save that model until you add at least one valid Source. EZNEC will not let you run a model or save it to a file unless it meets some minimum level of completeness and consistency. This is a common sense restriction that prevents chaos from breaking out.

If you want to save a model into a file with a particular name, you must execute the *Save As* command explicitly. If you are updating an existing model, you will be prompted for permission to overwrite the contents – even if is the file that the model came from.

Let me describe a scenario where this can be a little confusing.

You are working on a file named Dipole.EZ. When you opened that file, the single Wire describing the dipole was 33.43 feet long. This is a 20 meter dipole example provided in the EZNEC package. You change the length to be 35 feet as you are experimenting with various permutations.

After some time, you decide to exit the program and actually turn on your radio. The model has changed, because the Wire is now 35 feet long, as opposed to 33.43 feet long. You will not be prompted to save your changes, obviously related to Dipole.EZ. Instead, EZNEC will automatically save the updated model in LAST.EZ.

The next day, you decide to pick up where you left off. When you start EZNEC, it will automatically load LAST.EZ, and, without doing an explicit *Open* command, or picking from a recent history list, you will indeed start off exactly where you left off.

If you are uncertain or forget that you are where you left off, you might be tempted to *Open* Dipole.EZ, since that's what you thought you were last working on. The problem is that the changed length of 35 feet was saved in LAST.EZ, **not** Dipole.EZ. In fact, Dipole.EZ has the original 33.43 foot length! So, if you *Open* Dipole.EZ, you will actually lose all of your changes from the previous session. Now if you want to reload the model with the 35 foot Wire, you need to open LAST.EZ, since that is where it was actually saved when you exited the program the day before. You need to open LAST.EZ and recover your changes before you make any new ones.

The only time that EZNEC asks if you want to save changes is when you try to *Open* a file, and have made changes that were not saved to an explicitly named file.

Program Options

Programs usually have options and settings in addition to the contents of a file that the application manages. These are also called *Preferences*. In the old days of DOS, these might be saved in the **.ini* file. In the world of Windows, these are usually saved in the *Registry*.

These sorts of options and preferences are usually automatically saved and reused once the user changes them. They persist across runs of the program. In EZNEC, making these options the true *default*, that will be used next time the program starts, requires executing an explicit *save as default* command.

Here's an example. The power level value is not part of a model. This is because the gain and pattern of an antenna are never a function of the power level. The power level specification serves to scale voltage or current levels at Sources so that they are set as if that power level was being supplied.

I usually set the level to 1500 watts, since I'm often interested in making sure that I know the maximum voltage and current levels needed for capacitors and inductors in matching networks. This also causes voltage and current levels to be values such as 4.5 amps, as opposed to 0.00045 amps. I just find it easier to work with.

The power level is set with a menu command on the Option menu. The Power Level dialog is:

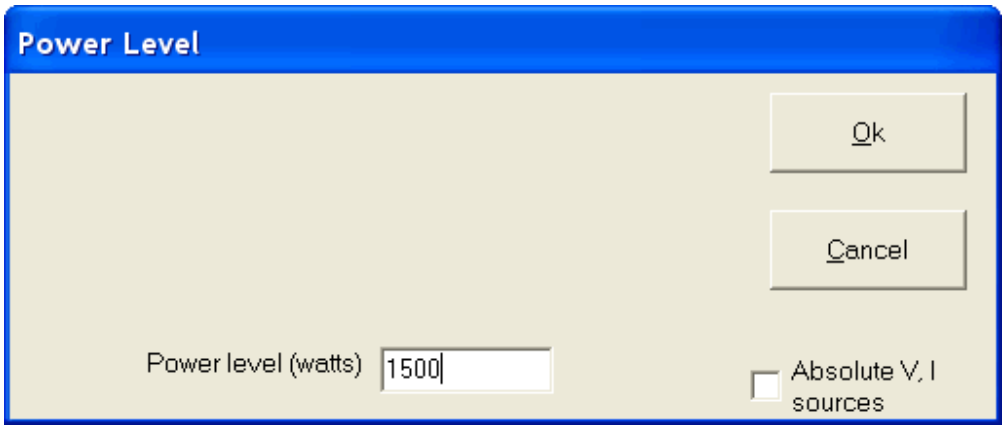


Figure 1 – Power Level Dialog

If you uncheck the *Absolute V, I sources* control, insert a power level value, and click OK, the program will adopt that power level. It will remain in use for all models open and run until the EZNEC program terminates.

When the program is restarted, however, the power level will revert to the default value.

If you want the power level to always be set to 1500 watts, then you must save the options, creating a new set of defaults. This is done with a command on the *Options* menu, *Options->Save as Default*. It is the bottom command on the menu. Here is a capture of the menu where the command is selected.

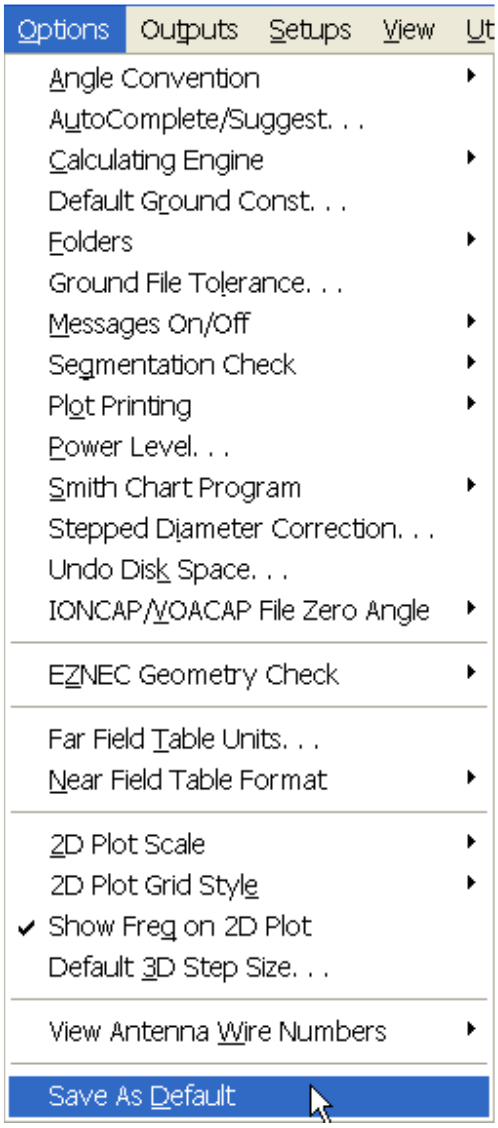


Figure 2 – Saving Options as the Defaults

If you change any program option, and there are many, they will have their changed values only until the EZNEC program terminates, unless you explicitly save them all as the new defaults.

There are other preferences in the program that require a similar explicit save to make them the persistent default values. If you change something that you expect to survive across future program executions, you probably should consult the Help file to make sure that you are following the correct procedure.

Wires

I believe it's fair to say that the heart of modeling is the *Wire*. Although we all have a practical understanding of what a wire is, in EZNEC/NEC-2 it is a specific term of art. A Wire is a straight conductor of a certain length and diameter, and optional insulation. All Wires in a model are made of the same material, such as copper or aluminum. The material serves to determine a resistive loss and permeability.

The position of a Wire is defined by the 3-dimensional (3D) coordinates of its two end points.

Wires are also characterized by the number of *segments* on the Wire.

Each Wire is assigned an identifying number, which is its position on the Wire window list.

Wires are *connected* when they share end point coordinates. It is also possible to connect Wires at segment boundaries, although I rarely see this done, and for several reasons it can be difficult to *get right*.

There are a number of rules and guidelines that influence how Wires should be oriented and connected. These rules and guidelines are designed to improve the accuracy of the results. EZNEC performs a geometry check every time you execute a model. Should you receive a warning or error, you need to adjust your model to live within the rules.

Wires are the places where all other modeling objects attach. EZNEC uses the term *insertion object* to imply a wide range of parts that literally insert themselves into a Wire. Insertion objects are inserted into a specific segment on the Wire. When more than one insertion object occupies the same segment, they directly interact. Insertion objects include Sources, Loads (inductors, capacitors, resistors, traps, etc), transmission lines, transformers, and L Networks.

There are a number of rules and options for how insertion objects interact on a Wire segment. These provide the richness we need to model various antenna

and circuit topologies. Insertion objects connect to each other through Wires. Let's say that you want to connect a Source to a transmission line. That is accomplished by connecting the Source and the line to the same segment on the same Wire.

All Wires radiate. Wires are also *tie points*. This duality presents a certain conflict, since we don't usually think of tie points as antennas. When Wires need to be used to create small circuit topologies, such as a matching network with several inductors and/or capacitors, the traditional approach is to use very short Wires with few segments. It is also common practice to locate those types of Wires far enough away from the actual antenna Wires that there is little interaction through coupling.

Recent versions of EZNEC have added the concept of a *Virtual Segment*. I tend to call it a *Virtual Wire* since it behaves like a special one segment Wire that does not require the specification of end point coordinates. These are a powerful addition to modeling since they provide useful tie points that don't have to be manually located away from the antenna and assigned coordinates. I believe that EZNEC converts Virtual Segments or Wires into actual short one segment wires located off in the distance. That's the sort of detail management that computers are good at, and it simplifies the modeling effort and results in a cleaner model abstraction.

Although we usually think of a *wire* as a flexible piece of metal not thicker than a car battery jumper cable, an EZNEC/NEC-2 Wire can have any diameter. If you want to model a 2" diameter aluminum tube that is part of a large 40 meter Yagi, you use a 2" diameter Wire and set the Wire Loss to mimic aluminum. Truss structures such as antenna towers can be modeled as a large number of Wires. Usually, however, unless you are making a detailed study of the tower, it's much easier to estimate equivalent cylinders of Wire and use them in place of a complex tower model. The details of that transformation are too long to discuss here, but if you have avoided modeling because you believe your structure is too complex, you can probably model simplified versions that will produce very useful results.

Many antennas, especially at higher frequencies, consist of solid metal surfaces. An example would be a parabolic dish reflector. With NEC-2, those are modeled as wire frames or a grid or mesh of Wires. Let's say that you wanted to model a vertical antenna mounted on a car body. The approach to that design would be to create a wire frame model of the car. This type of modeling is beyond the scope of this document.

Segments

Each Wire has a segment count that is part of the Wire specification. Segments break up a Wire into smaller pieces used to drive the computations behind the simulation of an antenna. As you might guess, more segments generally implies more accuracy.

The time it takes to compute results is a function of the segment count in a model. The time rises approximately with the square of the number of segments. With the faster computers we enjoy today, a high segment count rarely results in a performance problem, unless you have a very large design based upon thousands of segments. Most common antennas can be modeled with less than a few dozen segments.

Segment count is usually limited by the software package. I believe that this restriction started because of the use of fixed table sizes within the Fortran source code of NEC. Entry level packages, including the free EZNEC version, support a limited number of segments. The idea is to allow basic modeling and the opportunity to determine if modeling is right for you.

The bottom line is that specifying and managing segments can be a major part of modeling. There are accuracy issues, simulation time issues, and product capacity issues. This is one of those areas that can be more of an art than science.

EZNEC has an *automatic segmentation* command that directs the program to choose segment counts for your Wires. That is probably not a bad way to get started. I personally believe that it tends to use fewer segments than I would do manually, but it does result in the fastest models with the highest capacity, since it is very frugal with the segment count.

From my experiences with modeling, there are two tips I would like to pass along.

When I need a very short Wire that will act as a tie point, I tend to use 3 segments on those Wires. One segment will usually work, but short 3 segment Wires seem to crop up all of the time. This is especially true for short wires that include a Source. Please see the *Source Placement Precautions* section of the Help documentation.

Second, for Wires in general, I almost always use an odd number of segments. Consider the cases of two and three segment Wires:

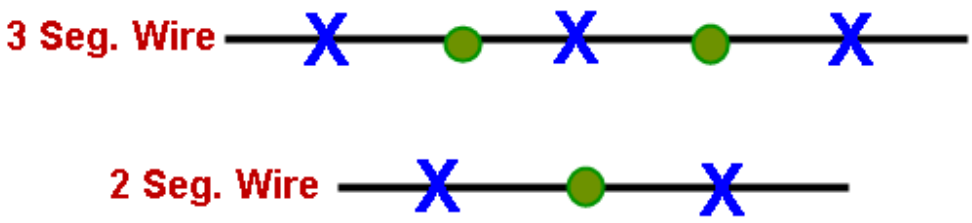


Figure 3 – Odd and Even Segment Counts

The green circles are segment boundaries, and define the segments. As mentioned in the Wires topic, *insertion objects* are conceptually inserted into the center of a segment. The potential location of insertion objects are indicated with the blue X's. When you have an odd number of segments, an insertion object can be located exactly in the lengthwise center of a Wire, as the top example shows. With an even number of segments, you can't get to the precise middle of a Wire. For many common antenna configurations, we usually are symmetric around the center, and connect or attach something at the center.

As the number of segments grows, the potential problem goes away, since the segments get shorter and shorter, and the deviation from being at the exact center drops. As an example, I took a 40 meter dipole, around 66 feet long, and modeled it with 11, 12, 60, and 61 segments. The Source was specified as being 50% down the wire – the center of the dipole.

The gain and pattern reported for all 4 cases were effectively identical. The variations occurred around the impedance values. The impedance values were:

Segment Count Influence	
# Segments	**Impedance**
11	88.55 + j 11.31
12	**90.03** + j 11.42
60	88.80 + j 11.87
61	88.75 + j 11.87

The odd man out is the 12 segment case, where it's an even number of fewer segments. The minimum number of segments, 11, was suggested by the EZNEC *Auto Segmentation* feature.

Determining Wire End Coordinates

I would be willing to bet that the #1 reason why some folks shy away from modeling is because of the need to calculate the coordinates of the end points of

Wires. This can involve that dreaded subject – *math*. Fortunately, the math is nothing more than trigonometry, which we all should have seen go whizzing by in junior or senior high school.

There was a time when computing Wire coordinates beyond something trivial required working with a calculator and a pad of paper and then entering the values into the program. That still happens from time to time. EZNEC has evolved a number of commands that will do a lot of computation for you. The trade off is that you have to understand how to use the commands and how to direct them to achieve a desired result.

A Wire end coordinate consists of 3 values, the X, Y, and Z values. The X and Y axes lie on the ground, and the Z-Axis is the height in the air. The value on the Z-Axis is always a positive number, unless the Ground Type is *Free Space*, in which case a Wire end can have a negative Z value. In general, I never model in *Free Space*, so I always think of my model as being somewhere above the ground, with positive Z-Axis values.

A coordinate is a number and a unit. The units support both the English and Metic systems. All of my examples will be in units of feet. The modeling of Yagi's, or other antennas made out of stepped diameter tubing, tends to be done in units of inches (if you are from the English system).

Following the normal mathematical conventions, the positive X-Axis points to the right, and the positive Y-Axis points to the top of the page. The Z-Axis comes out of the X, Y plane, towards us. Positive azimuth angles are drawn from the X-Axis in the counter clockwise direction. Folks more comfortable with mapping tend to locate the zero degree azimuth angle pointing straight up, or to the north, with positive angles in the clockwise direction. EZNEC supports both orientations when drawing patterns.

Trigonometry Refresher and Tips

So long as Wires are either horizontal or vertical, getting the end coordinates right is not a big deal. The problem comes in when Wires run at an angle, and we need to determine the projection of the end point onto the coordinate axes. Given the length of the Wire and the angle, the sine or cosine of the angle times the length gives the projection values. The trick is picking the right one.

Let's say that you wanted to determine the Wire coordinates for a 40 meter inverted vee that was 33 feet on a side, with a 22 degree drop angle. The top center is at 50 feet. Here is a diagram of the antenna.

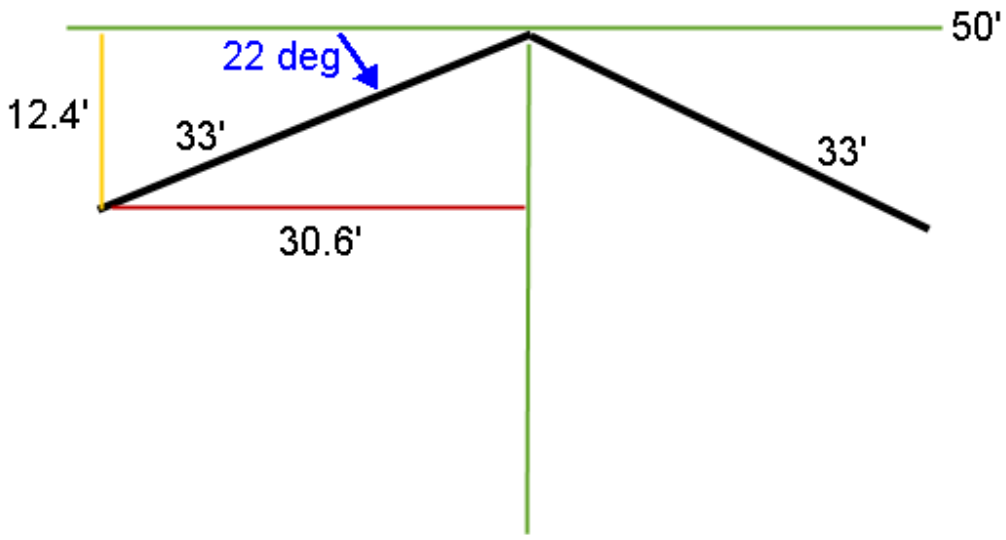


Figure 4 – Sine and Cosine Example

Using a calculator, the sine of 22 degrees is 0.375, and the cosine is 0.927. Then, multiple each of those values by 33, since that's the length of the wire. The results are 12.4' and 30.6'. It should be clear that the longer dimension is away from the center line, and the shorter dimension is the drop down. Since the Wire is dropping down, we need to subtract the 12.4' from the 50' height, resulting in a 37.6' height off of the ground for the Wire ends.

This can be translated into an EZNEC Wire list as follows:

Wires

Wire Create Edit Other

Coord Entry Mode Preserve Connections Show Wire Insulati

Wires

No.	End 1				End 2				Diameter	Segs
	X (ft)	Y (ft)	Z (ft)	Conn	X (ft)	Y (ft)	Z (ft)	Conn	(in)	
1	0	0	50	W2E1	0	30.6	37.6		#12	25
2	0	0	50	W1E1	0	-30.6	37.6		#12	25
*										

Figure 5 – Wires List from Example Vee

We created two Wires. The common center point is at 0, 0, 50. They share that coordinate. The two ends will be 37.6' high. I've arbitrarily chosen to locate the antenna long the Y-Axis, so, the 30.6' goes in the positive direction for one, and the negative for the other.

By clicking the **View Ant** button on the main EZNEC window, we can get a 3D view of the antenna.

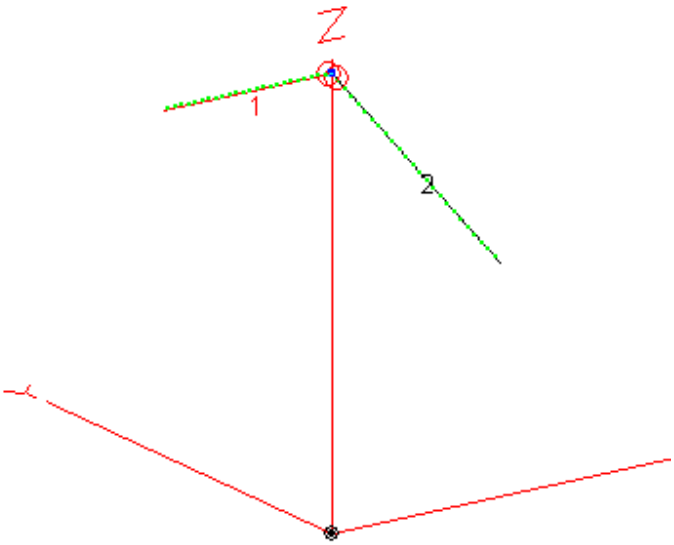


Figure 6 – Example Vee Antenna View

This looks pretty good. One last check is to verify that each Wire is 33' long. We can let EZNEC tell us the length. To do that, I moved the mouse over the arrow to the left of the Wire number, and right clicked. This created a popup window that reported the Wire length.

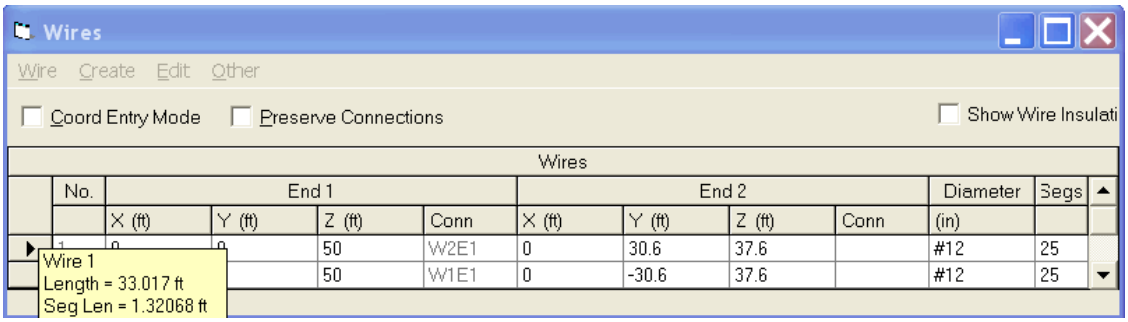


Figure 7 – Wire Length Check

The reported length is 33.017 feet, which is more than close enough.

Another way to display Wire information, including the length, is to hover the mouse cursor over the Wire in the View Ant window. A small window containing a number of pieces of Wire information will popup on the screen.

Using Program Shortcuts and Commands

Recent versions of EZNEC have been adding a number of Wire shortcuts and commands that make it easy to create complex Wire structures with arbitrary angles. Please check the Wires window and its menu commands. Some commands are available by right clicking over the actual coordinates in the Wire table.

Let's redo the previous example of the 40 meter inverted vee with the 22 degree drop angle. This time, we are going to let EZNEC do the work.

I'll start with a dipole at 50 feet off of the ground. Since I'll be letting EZNEC drop the ends of the Wires, I do need to create two Wires to begin with, so that I can drop two ends without dropping the whole single Wire.

The Wire table is:

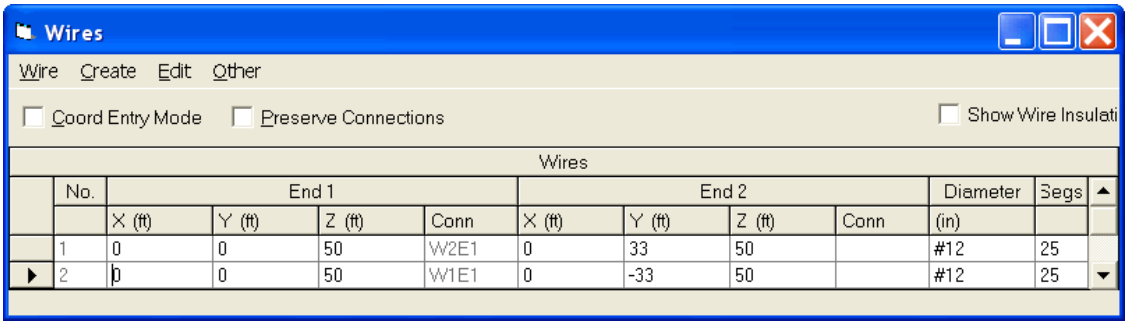


Figure 8 – Flat Dipole Composed of Two Wires

The antenna view is:

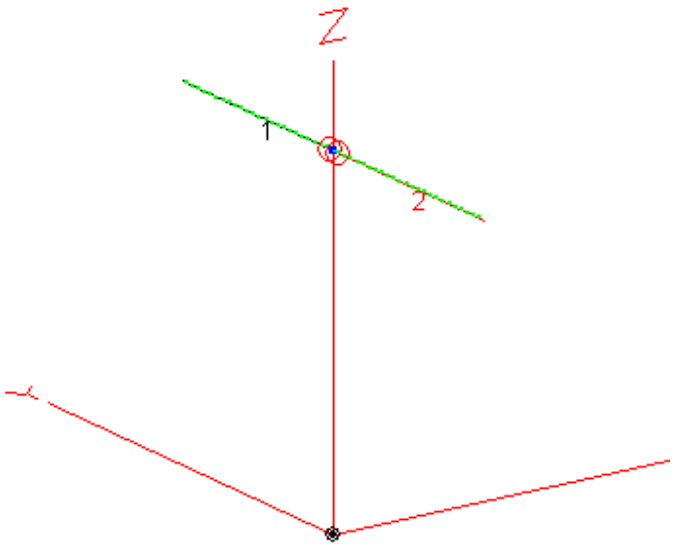


Figure 9 – Flat Dipole View

It's just a flat horizontal dipole made from two Wires, each 33 feet long. All we need to do is to take End 2 of each dipole and rotate it down by 22 degrees.

To do that, I right clicked over the Wire 1 End 2 cells in the table, and that displayed a context menu, as follows:

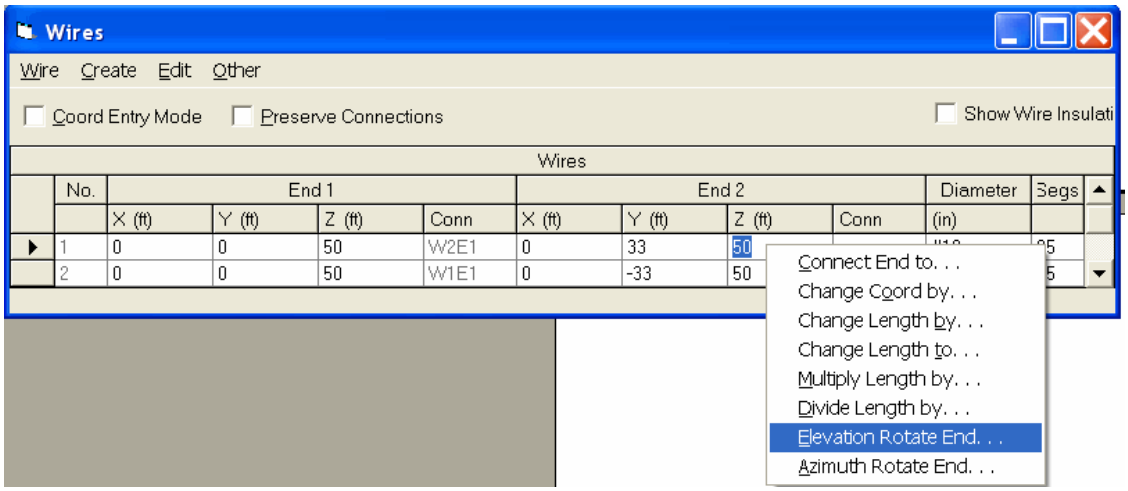


Figure 10 – Wire 1, End 2 Context Menu

From that menu, I selected Elevation Rotate End, since that's the change I need. That command brought up the following dialog:

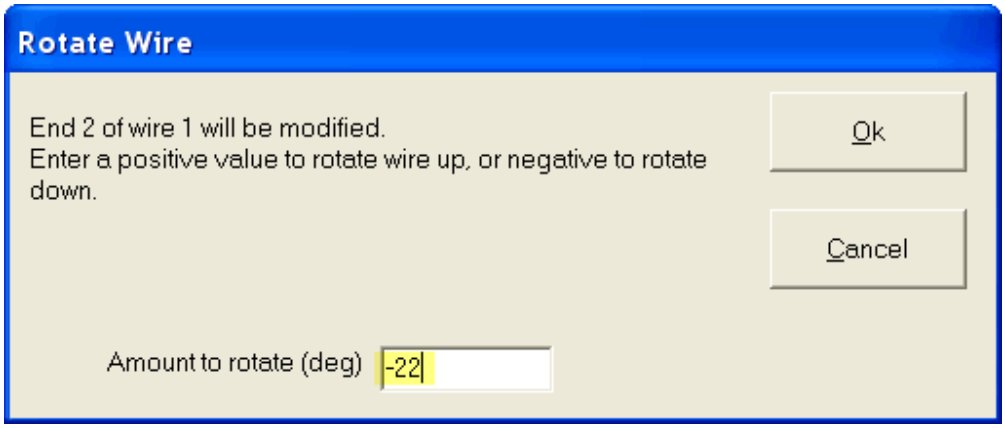


Figure 11 – Rotate Wire Dialog

I filled in the value of negative or -22 degrees, since I want the end rotated down by 22 degrees. The resulting antenna view was:

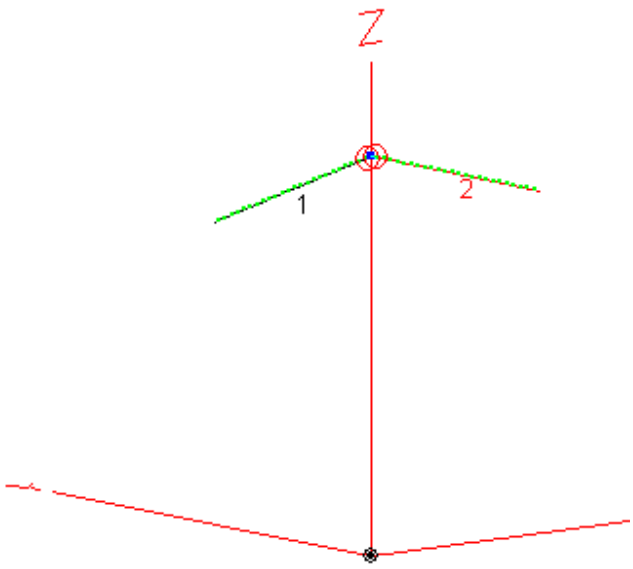


Figure 12 – Wire 1 End 2 Rotated Down

The outboard end of Wire 1 has been rotated down 22 degrees. I did the same operation to End 2 of Wire 2, and the vee was complete. The final Wire list is:

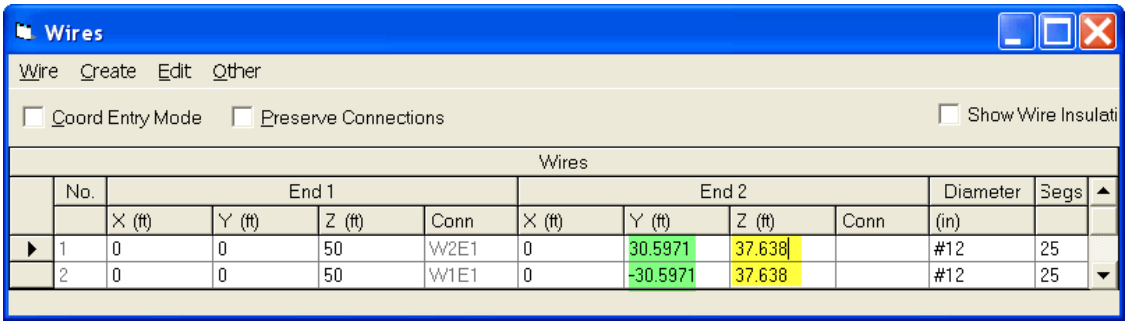


Figure 13 – Final Wire List

If you compare to the previous manually computed results, it's the same values, except for rounding.

EZNEC has a number of commands for modifying, copying, and moving Wires. Most all of the pain of pushing around coordinate values has been removed. It is a good idea to keep a constant eye on the antenna view, so that you know you are creating what you want.

Sources

After Wires and their segments, the next object that is encountered in modeling is the *Source*. At a minimum, a model must contain at least one Wire and at least one Source.

Although it is tempting to think of a Source as a transmitter, that really is not the best metaphor. In NEC-speak, a Source is specified with the EX card, short for *excitation.* Perhaps a more accurate description of a Source is that it follows the *stimulus-response model*. We stimulate or excite the antenna at the location of the Source with a voltage, and part of what the NEC engine computes in response to the stimulus at that point is the current flow. By dividing the complex voltage by the complex current, we have the impedance of the antenna load at that point.

While a Source might not be identical to a transmitter, they usually connect at the point where a transmission line or transmitter would be connected.

Sources are one of a group of EZNEC/NEC objects that are called *insertion objects* in the EZNEC documentation. They are inserted into Wires. To be specific, they are conceptually inserted into the middle of a segment on a Wire.

Here are a few examples worthy of discussion.

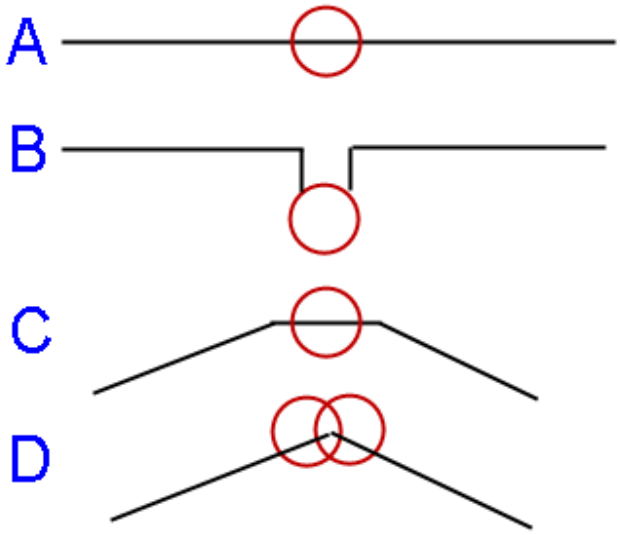


Figure 14 – Source Examples

If you ask most people how many wires are needed to build a real dipole, they would answer *two*. Assuming that the dipole is a straight wire, the answer in NEC is one. That's because the Source effectively breaks the Wire at the segment of insertion. The A antenna shows how that is indicated in EZNEC. A is a dipole. You can consider it to be equivalent to what's drawn in B, if that picture is more comforting.

There are times when it's necessary to locate a Source at the end of a Wire, usually because it's at a point where the real wire bends at an angle and the modeled Wire attaches to another Wire (since modeled Wires cannot bend). Examples include the Inverted Vee dipole, and corner fed loops.

One of the traditional solutions to create that model in NEC is shown in C. Here, a short Wire is inserted between the ends of the angled Wires. Since this need comes up frequently, EZNEC has a special Source type called a *Split Source*. This Source is located at the end (last segment) of a Wire, and is automatically arranged to split itself across the two Wires at the junction. The picture of this sort of Split Source is shown in D. The EZNEC documentation prefers the Split Source in D over the added Wire in C. Although the D example shows overlapping circles, that is not always the visual appearance, especially if you zoom in on the antenna view.

So far, we have talked about Voltage Sources (V), and Split Voltage Sources (SV). EZNEC has two other Source types. They are the Current Source (I), and the Split Current Source (SI). With these Sources, you specify the complex

(magnitude and phase) current at a Source, and the NEC engine will compute the voltage at that point, again resulting in the determination of impedance.

For many models, whether you pick Voltage or Current Sources will not make a difference. Current Sources most often come up in the context of phased array design, where more than one Wire is driven. Since current as opposed to voltage directly leads to radiation, it's common to talk about the driving points in terms of current ratios. For example, the classic two-element vertical array spaced 90 degrees apart on the ground is often driven with two Sources, each with the same **current** magnitude and a 90 degree electrical phase shift between them. If you were modeling this sort of array with two Sources, you would need to specify Current Sources to achieve the expected results. If you are designing phased arrays with more than one Source, and you are getting unexpected results, check that you are using the correct Source type.

All four types of Sources are specified with relative Amplitude and Phase values. These voltage or current values interact with the Power Level option. You can choose to either work with the absolute amplitudes specified, or, have the specified voltage or current values automatically scaled by EZNEC to correspond to the specified power level. I prefer the second option, and usually model with a power level of 1500 watts. That presents me with realistic voltage and current levels for the maximum power level allowed in the United States.

Needless to say, increasing the voltage (or current) amplitude on a Source does not change the gain or pattern of the antenna. It certainly does, however, change the voltage and current levels! This assumes a single Source, and not an array with multiple Sources. A real antenna has the same gain and pattern if you drive it with 1 watt, or 1,000,000 watts. The same is true with models.

Amplitude and phase can sometimes be confusing due to their relative relationship. This mainly applies to multiple Source models. Let's say that you are talking to a friend about a phased array with two Sources. In your mind, there is a 135 degree phase shift between the Sources that you consider to be the values 0 and 135 degrees. Your friend sends you a model of the antenna, and when you look at the Source list, you find the respective phases are -135 and 0 degrees. This is the same antenna, with the same shift, but expressed with different values. The values might be -35 and 100, it's the same phasing. It could even be -251 and -116 degrees. All of these pairs and an infinite combination more, represent the same phasing (the right number is 135 degrees greater than the left number). You can play the same game with the amplitudes. When looking at a model created by another designer, you might have to take some time to understand the Source relationships. Although there are an infinite number of equivalent permutations, there are some conventions worth following.

Sources and other insertion objects attach to the middle of a segment on a Wire. That might suggest that these objects are placed on the Wire by specifying a segment number. In EZNEC, they are specified by the percentage of the distance from end 1 of the Wire to end 2. That percentage is used to determine the closest segment center. In most cases, that's a useful way to locate the object. One desirable attribute that it has is that if you change the number of segments, the object tends to stay in the same place, which seems very reasonable and intuitive.

On the other hand, don't expect the object to be located exactly at the precise percentage that you have specified. After all, what matters are segments, not percentages. This can become an issue if the segment count is relatively low, or, if you are attempting to locate multiple insertion objects on the same segment. In those cases, you may need to pay attention to the percentage and segment details to be sure the model matches your expectations.

Consider the example of a 3 segment Wire.

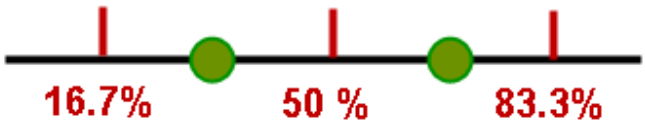


Figure 15 – Object Placement Choices on a 3 Segment Wire

The green circles represent segment boundaries. Because this is a 3 segment Wire, there are only 3 possible positions for an insertion object, indicated with the red lines. These points are going to be at 16.7, 50, and 83.3% down the Wire from the left. Here is a Source located on that Wire.

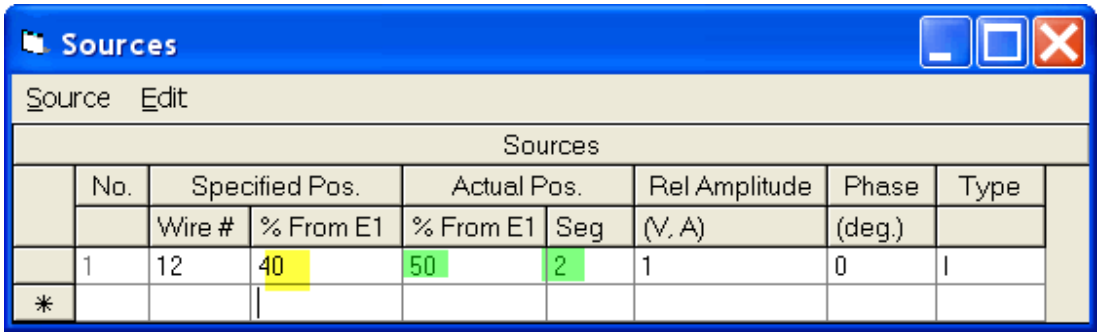


Figure 16 – Source on a 3 Segment Wire

I specified that the Source should be located 40% of way down Wire 12 from end 1. The actual position is reported as 50%, and that is segment 2. EZNEC picks the closest segment.

In some models, following the details of the percentages and segments will matter. An alternative in some cases would be to use multiple Wires on the same straight line, and manage location issues through Wire selection.

There can be no doubt that there is an art to building a model, especially in terms of laying out Wires and their segment counts.

Antenna View (View Ant button)

A very important window is displayed by clicking the **View Ant** button on the main EZNEC window. Although this window does indeed display a 3-dimensional view of the antenna, it is really a *Swiss Army Knife* with a large number of functions.

The window is highly interactive, and is cross linked and synchronized with many other windows. It is worth it to take the time to investigate all of the features and functions of this important tool. Only a few will be highlighted here.

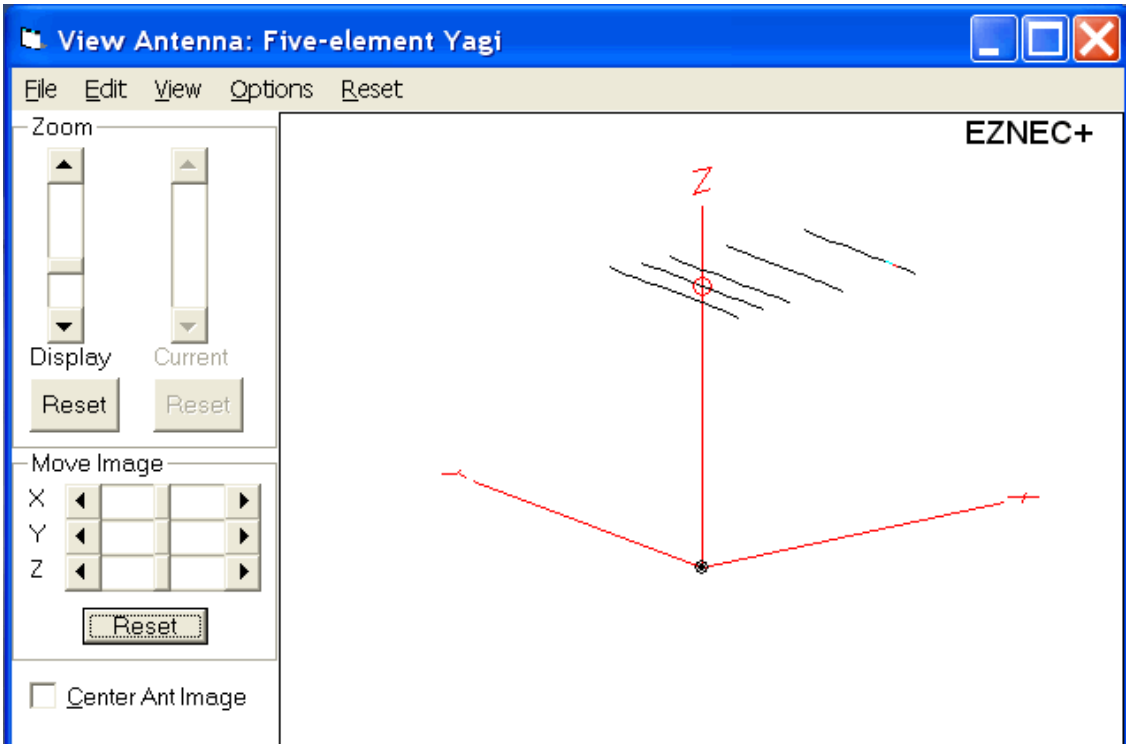


Figure 17 – View Ant Window

This example shows a view of a 5-element Yagi. The view can be changed by clicking in the window and dragging the mouse with the button down. You can view the antenna from any orientation.

Insertion objects are drawn on their Wires. In this case, a Source on the center of the driven element is depicted as a red circle. There are different shapes for different object types.

A strong linkage exists between the drawn Wires and the Wire window. If you select a Wire on either window, it is highlighted in the other window.

The antenna view can be annotated with a number of data items. The data display is controlled by commands on the *View* menu. The *Options* dialog on the *View* menu contains the following controls:

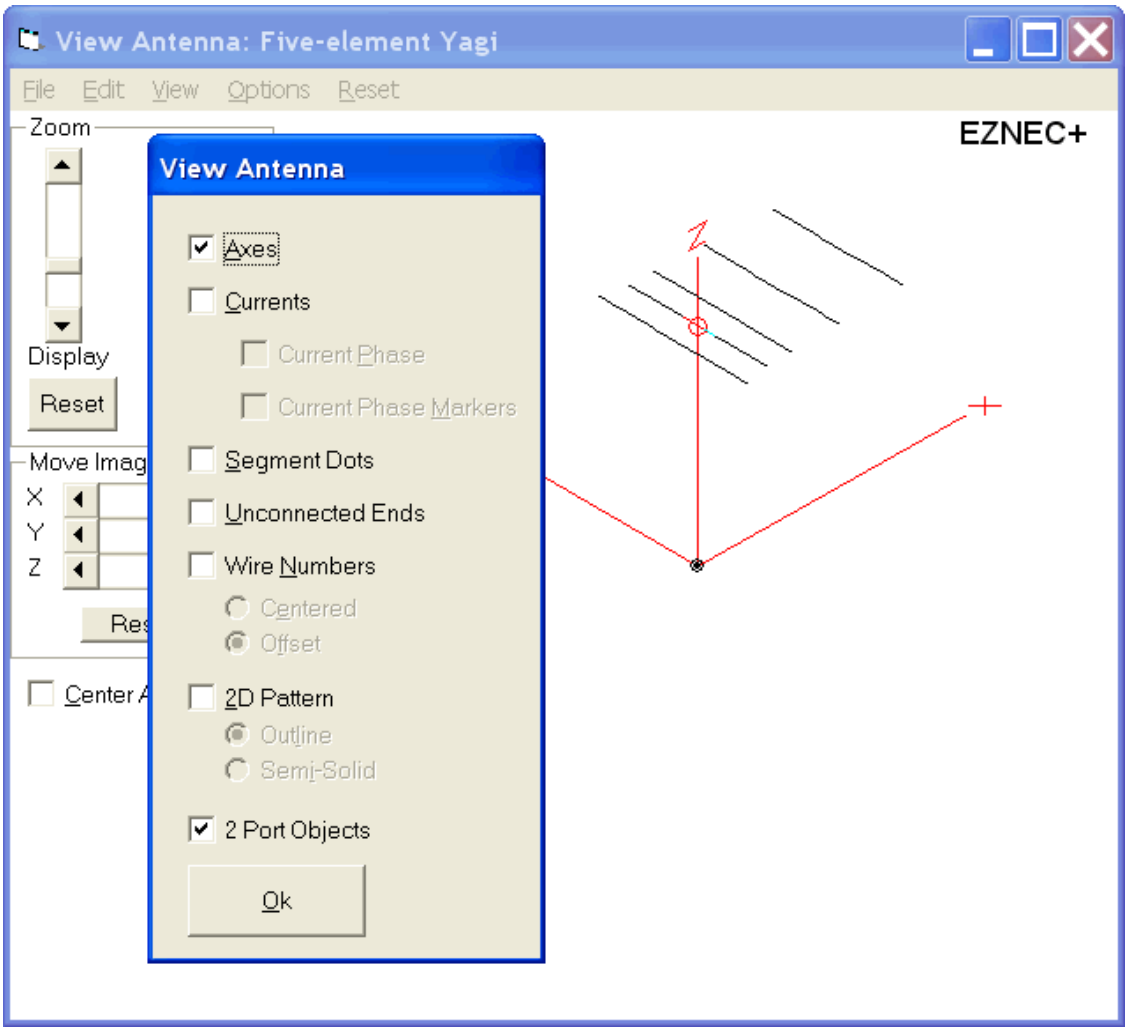


Figure 18 – *View->Options* Dialog

If **Wire Numbers** is checked, the antenna picture becomes:

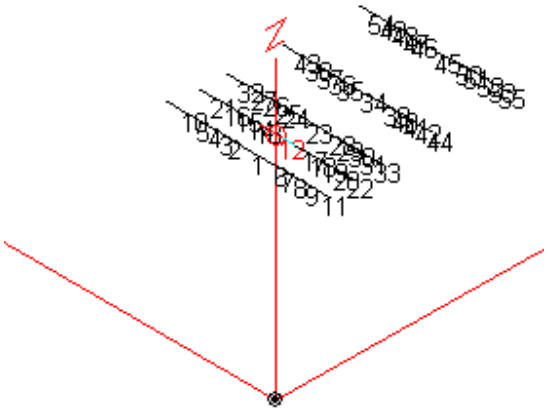


Figure 19 – View Wire Numbers

Each Wire is marked with its number. Since there are many Wires in this Yagi model, the view becomes rather crowded. One solution is to use the zoom controls on the left side of the window, and zoom in on the antenna to spread the Wires over a larger portion of the screen.

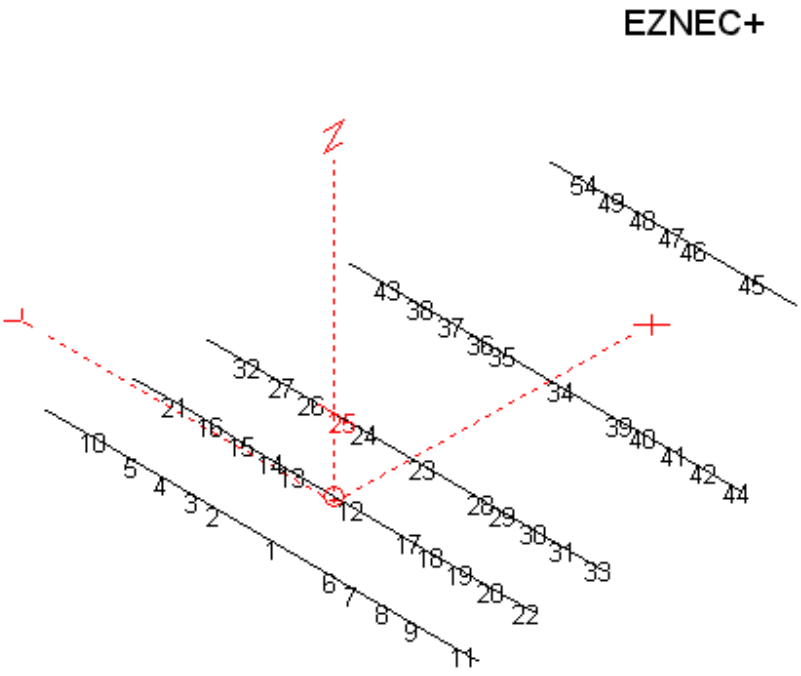


Figure 20 – View Wire Numbers (zoomed)

Another common viewing option is to display a 2 dimensional pattern slice in the correct alignment with the 3D antenna view. For this example, an elevation slice added to the antenna would show:

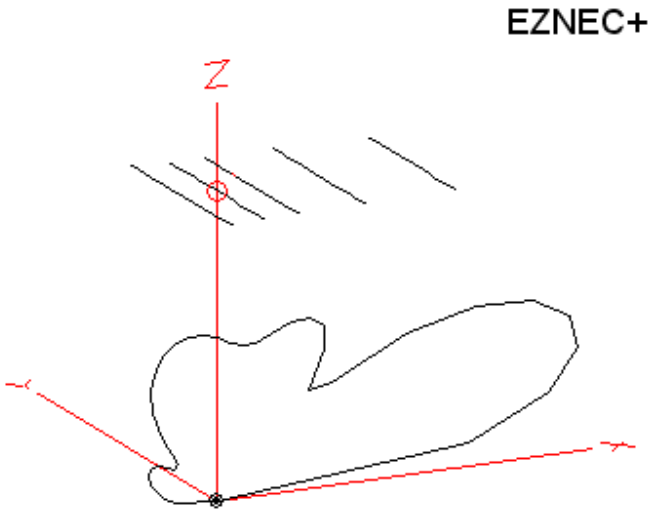


Figure 21 – Elevation Slice Added to Antenna View

Other common viewing options including Wire segment dots, and the current flow on the Wires.

40 Meter Vee

Since we just created a 40 meter inverted vee, we might as well continue on with it.

All antenna models need a *Source*. This is a point where we drive the antenna with either a voltage or current signal. There can be more than one Source in a model. If we drive with voltage, the modeling engine computes the current at that Source. If we drive with current, the modeling engine computes the voltage at that point. Once we have both voltage and current values, then the impedance of that point is simple Ohms Law, which is the voltage divided by the current.

We are most always interested in the impedance of Sources, since eventually they make their way back to our radios, and impedance matching is of some importance.

Sources are insertion objects, and as discussed earlier, are normally inserted into the middle of a segment. In the case of the inverted vee antenna, we want the Source to be located at the junction of two Wires, not the middle of a segment.

Since this situation comes up with any corner fed antenna, EZNEC has a special type of Source called a *split Source* that allows us to locate the source at the vee junction. The Source window for this antenna is:

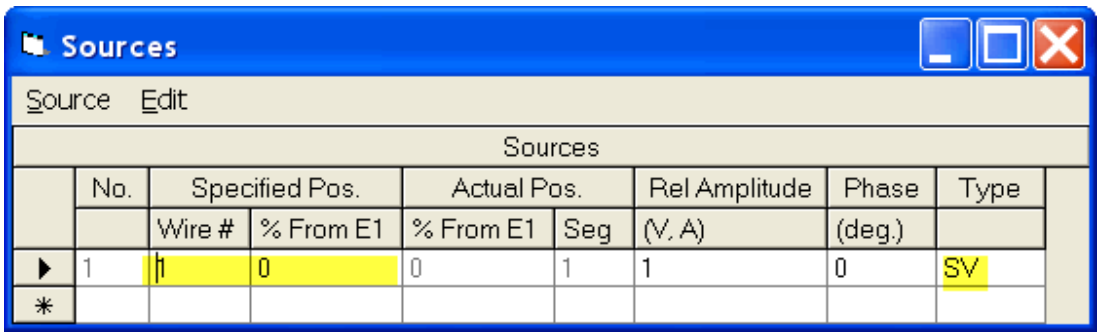


Figure 22 – Inverted Vee Source

The important settings are that we have placed the Source on End 1, Wire 1, and the type is **SV**, which is the *split Voltage Source*.

After starting EZNEC, and creating the Wires and Source, we have the following screen shot:

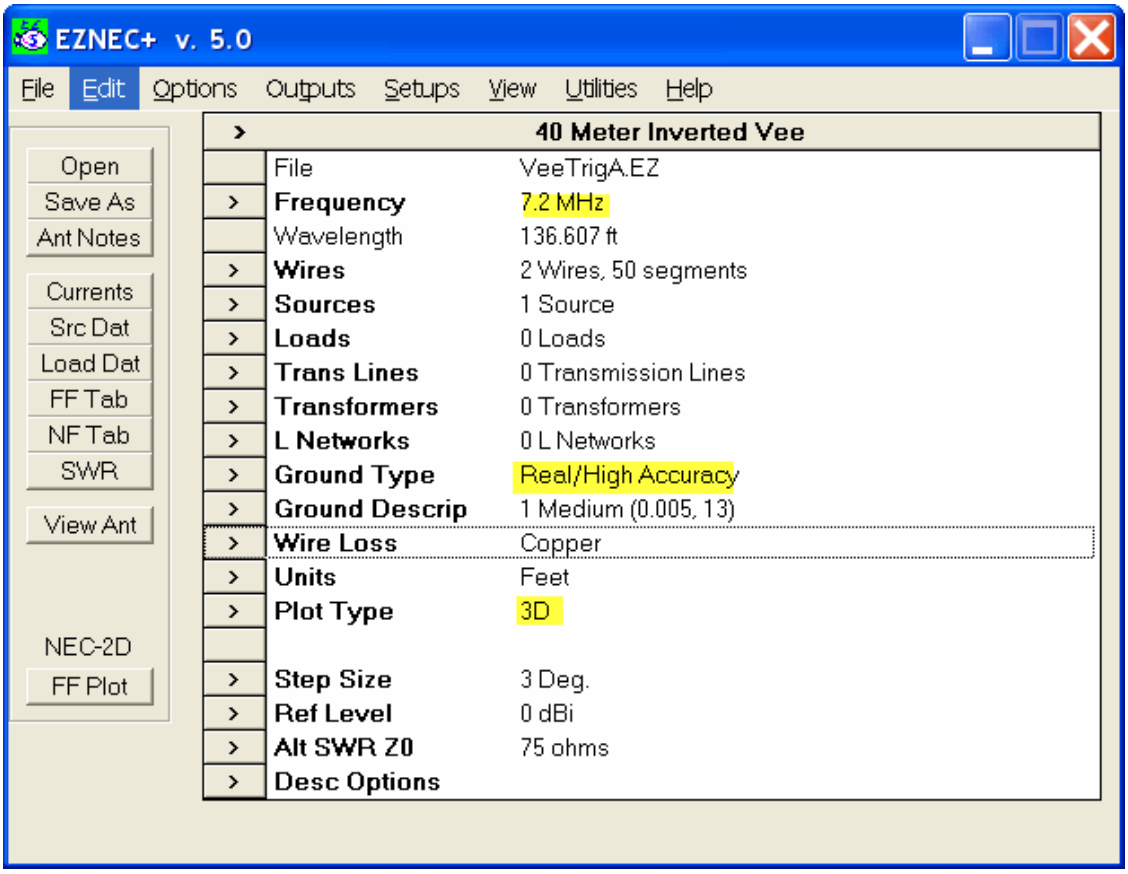


Figure 23 – EZNEC Main Window, 40 Meter Vee

This main window is always present when the EZNEC application is open and not minimized. A number of companion windows are usually open to display aspects of the model and computed results. It's not uncommon to have 5 to 10 windows on the screen that are all part of the same model and its results.

Of note on the main window is the selection of the test frequency and the **Ground Type**. For horizontal antennas that are not too close to the ground, the Real/High Accuracy ground type is most appropriate. If you know something about your ground conditions, you can use the **Ground Descrip** command to change the characteristics of the ground model.

The Plot Type is set to 3D, requesting a full 3-dimensional pattern plot. To run or execute the model and generate results, click the **FF Plot button**. The result is a 3D pattern plot window that looks like:

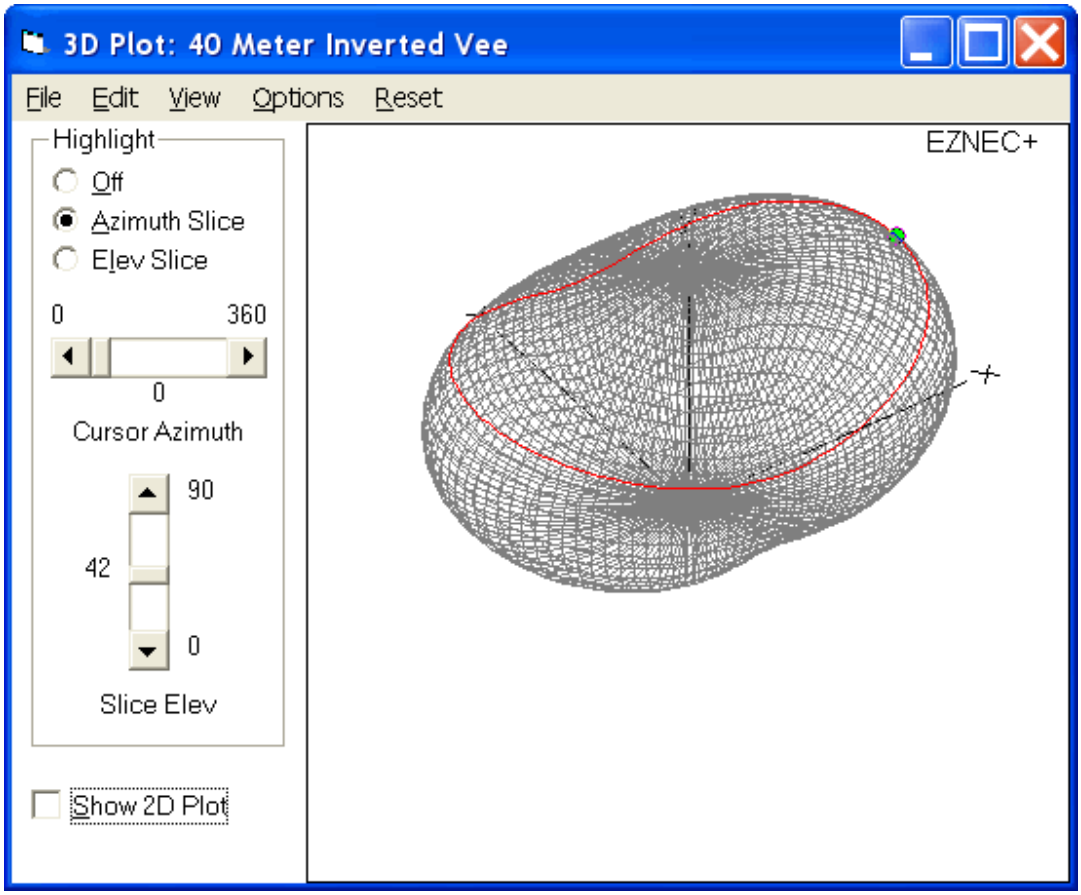


Figure 24 – Inverted Vee 3D Pattern

By left clicking and holding down the mouse button, and then dragging, the orientation of the pattern will change. This makes it possible to inspect the pattern from all angles.

Traditionally, patterns have been analyzed by taking slices through the 3D shape, and rendering them as either azimuth or elevation patterns. We can enable those views by clicking the **Show 2D Plot** control in the lower left corner of the window. The azimuth view is:

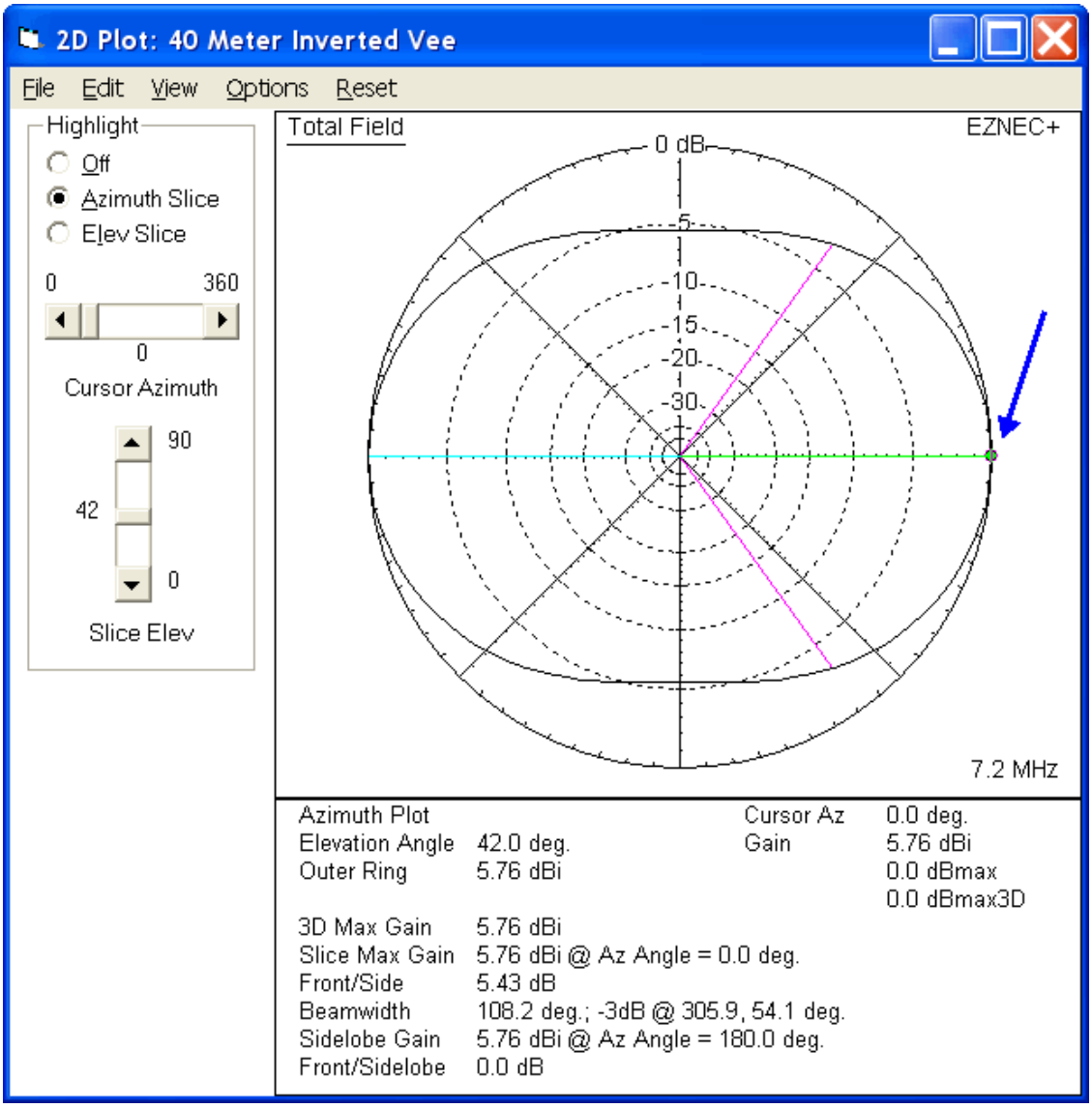


Figure 25 – Inverted Vee Azimuth Pattern

This slice is taken at a take off or elevation angle of 42 degrees. An important feature of this window is the round cursor that follows the pattern shape. I added a blue arrow to call it out. It can be moved around the pattern by either clicking over the plot, or, adjusting the slider control on the left. All of the data in the lower portion of the window will follow the cursor position on the pattern.

If we are interested in the Source impedance, click on the **Src Dat** button to display the Source data report.

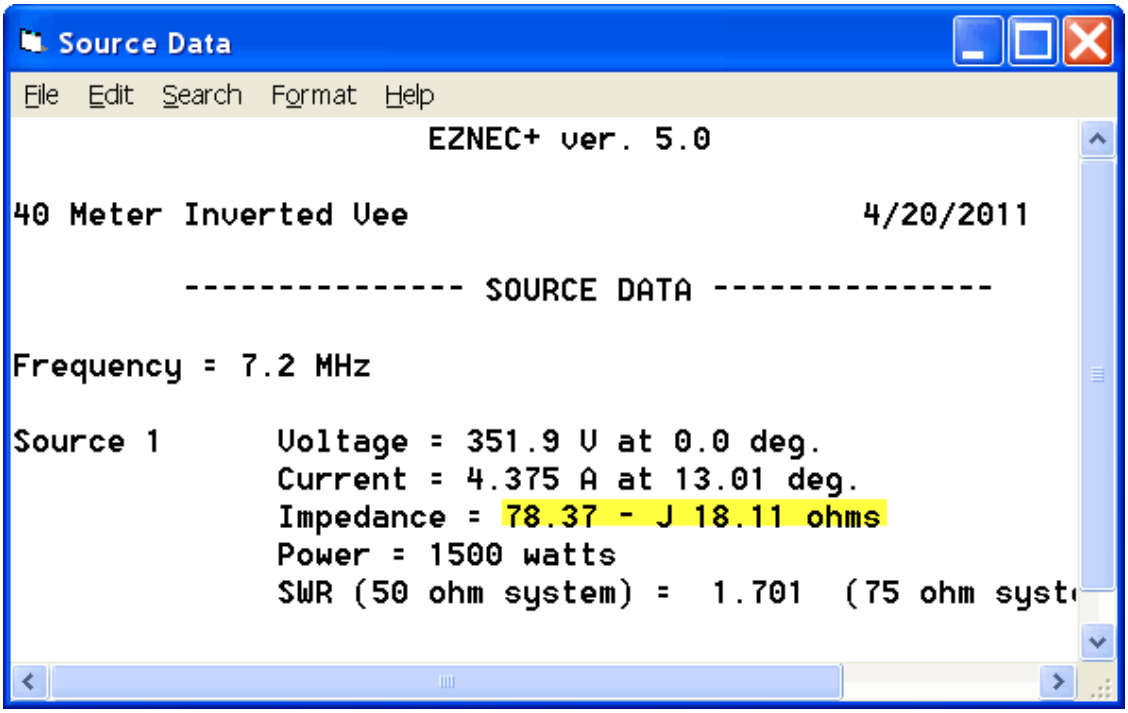


Figure 26 – Source Data Report

The impedance of this vee, which is a bit off resonance, is 78.37 – j 18.11 Ohms.

L Networks – 160 Meter Vertical

L networks are an important impedance matching solution. Modeling them in NEC has always been possible, through the Load primitive. Recent versions of EZNEC have introduced an explicit L network primitive that is very easy to use.

The goal of this example is to look at design variations of a 160 meter vertical constructed out of a left over 40 meter vertical. An L network will be used for impedance matching. This example will also show that we can use modeling to keep track of the various losses in the system, which can creep in and add up if we are not paying attention.

We start off with an abandoned 40 meter vertical that is 33 feet of 1" diameter aluminum. We want to see what sort of performance we will get if we move this antenna to 160 meters.

We need an estimate of the radial system loss, no matter if it's elevated or ground mounted. If we were using NEC-4, then we would be able to model a radial system of wires on or even buried in the ground. With NEC-2, we can't go that far with acceptable accuracy. So, we are going to have to come up with an external estimate. One approach to estimating loss is to look at sources such as

the ON4UN book. Table 9-1 of the 5th edition lists ground loss estimates as a function of the number of radials and the radial length. If we construct a good radial system, we can expect values between 5 and 10 Ohms. As a starting point I'll pick 8 Ohms.

The first model will let us determine the feed point impedance at the base of the antenna. We'll need the MININEC-type ground, the vertical Wire, a Load resistor representing the 8 Ohm loss, and the Source.

Here is the Wire list:

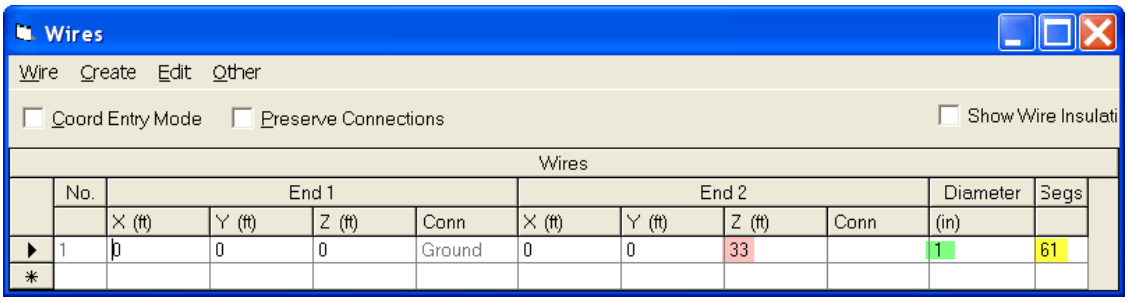


Figure 27 – Vertical Wire List

The vertical base is located at the origin, and makes a direct contact with ground (Z = 0) at End 1. This is part of the MININEC-type ground approach. The Wire is 33 feet tall, and 1 inch in diameter. The number of segments has been set to 61. That may seem like a high value, but we'll need the resolution along the Wire to experiment with some variations.

The Source list is:

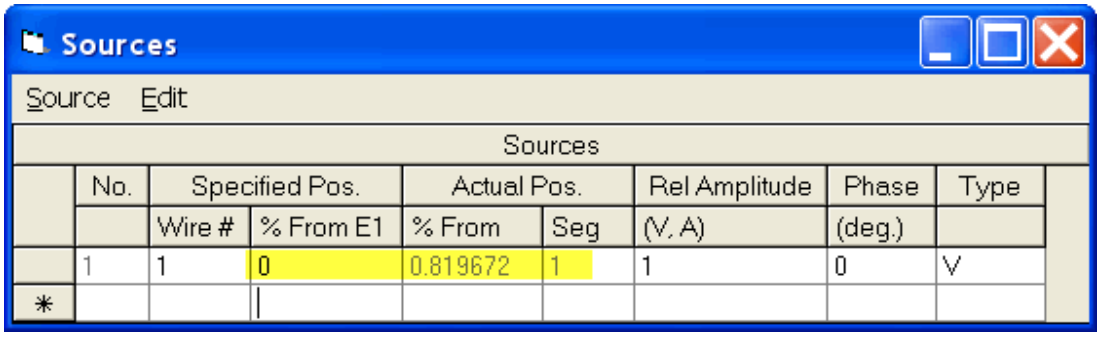


Figure 28 – Vertical Source List

The Source is placed at the bottom of the Wire, in segment 1.

The Load list is:

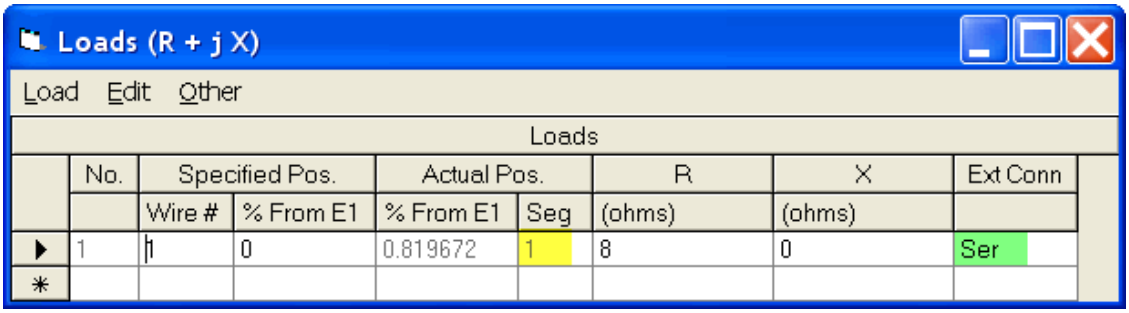


Figure 29 – Vertical Load List

The 8 Ohm resistor is located on the same segment as the Source. I also highlighted the **Ext Conn** (external connection), which is set to Serial.

It's important to understand exactly what's happening at the first or bottom segment of the Wire. Here's a diagram of what we have created so far:

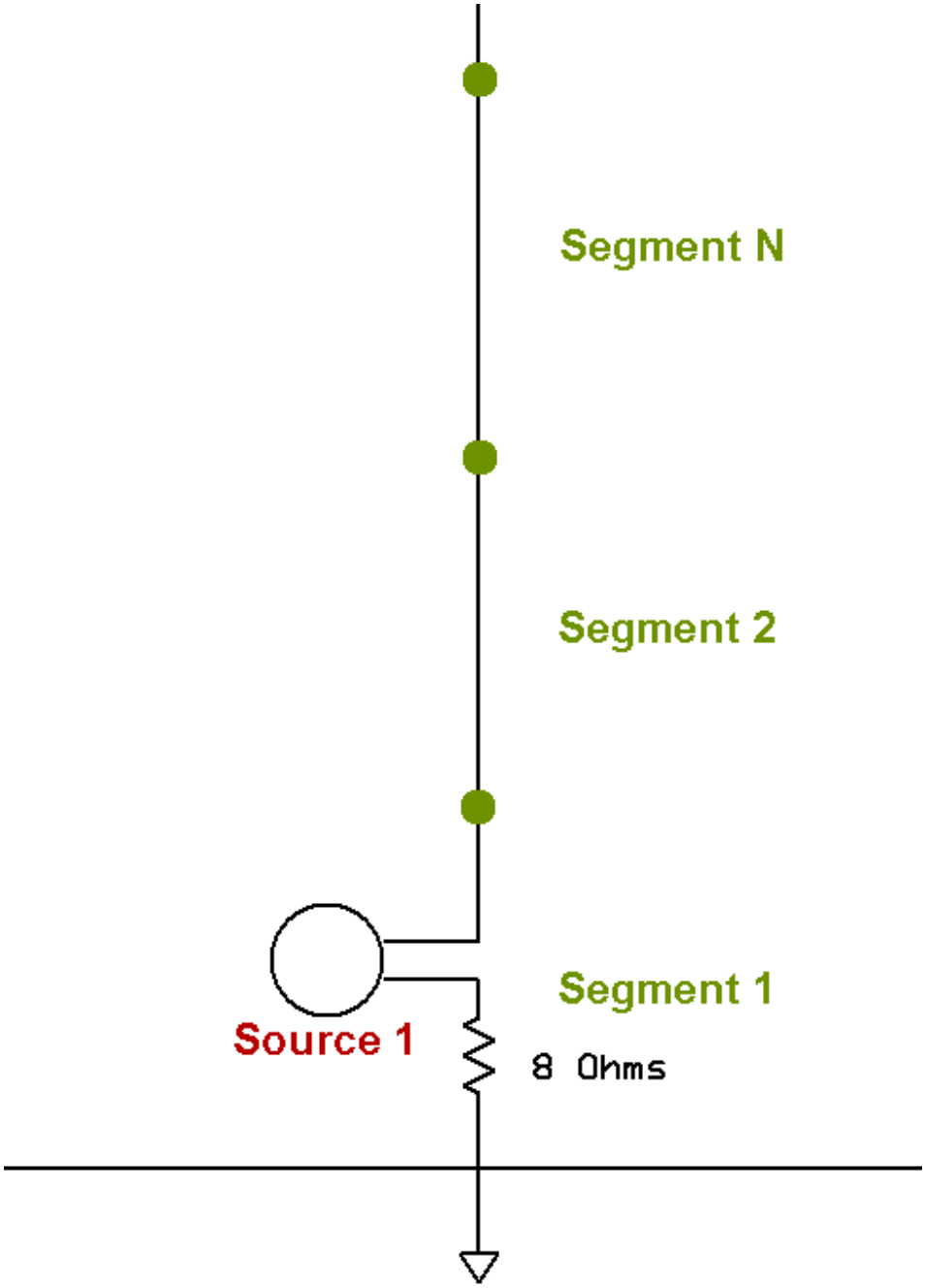


Figure 30 – Vertical Wire Segment Details

According to the Source and Load lists, we have attached a Source and an 8 Ohm resistor onto Wire 1 at segment 1, on the end that connects to the MININEC-type ground. What we need to know now are the *rules* surrounding placing multiple *insertion objects* onto a single segment. That is explained in the EZNEC Manual. The Source always inserts itself into the Wire at that segment, and the Load can be placed in series with the segment or parallel. That's why in

the last screen capture we selected the **Ext Conn** as **Series**, so that the resistor would be in series with everything else on the segment. Now I really don't know the order of the two parts, and it doesn't matter, although I drew it in the way that we are most comfortable seeing, with the resistor associated with ground, since it is acting as a proxy for the ground loss.

In summary, we took a Wire that is the length and diameter of the vertical, and connected one end to the MININEC-type ground model. At that point, we have a lossless vertical; it would be 100% efficient. We added a resistor that simulates the expected ground loss. We also added a Source so that we could drive the antenna. We put the Source and resistor in the bottom segment of a 61 segment Wire, and used the EZNEC/NEC-2 rules and options to make sure that they were both in series with the antenna.

In some sense, we are giving up a little bit of length on the antenna to attach the Source. That's why I wanted to have a generous number of segments, so that if a segment or two ends up being used as part of the feed at the base, we won't be giving up a significant amount of length. There are other ways to avoid that concern, such as using a separate wire so that the formal antenna is not used to support the feed system or other details.

Here is the main window after running the model:

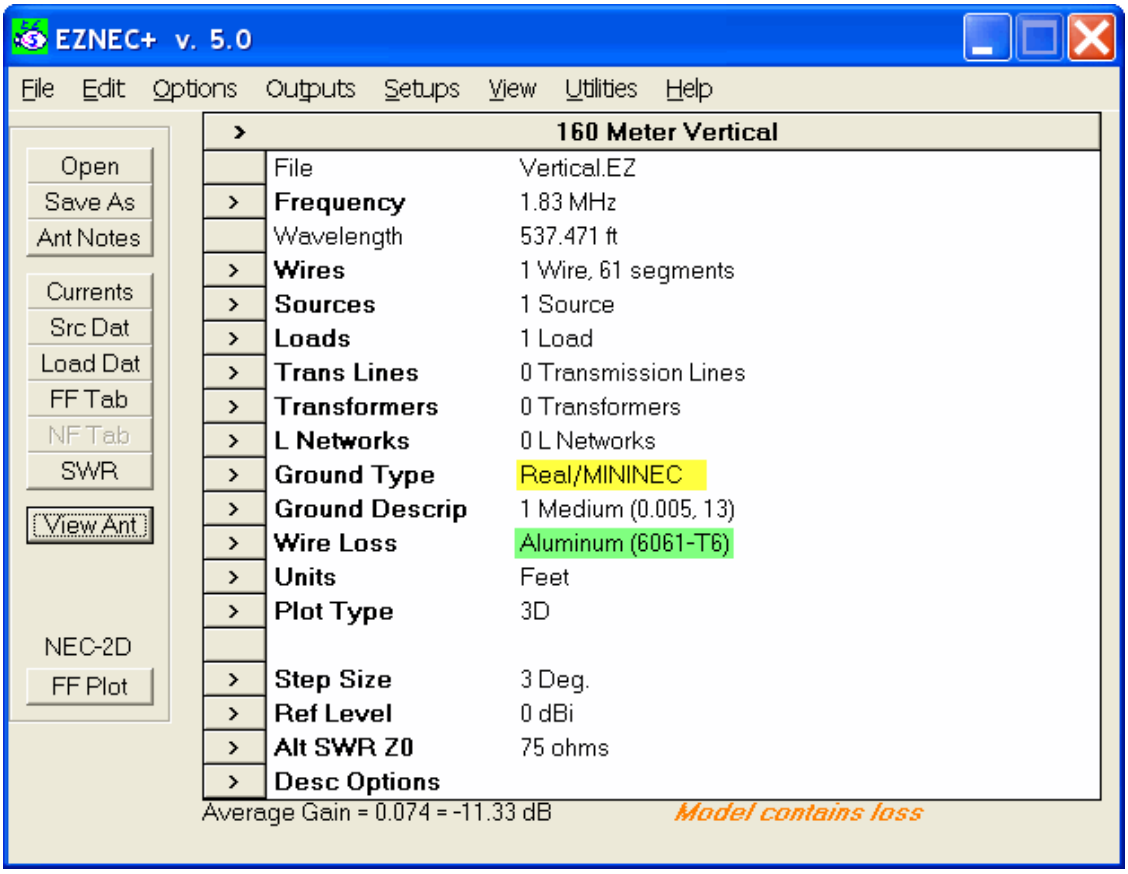


Figure 31 – Vertical Main Window

The important part of this window is the selection of the Real/MININEC Ground Type, and the selection of aluminum for the Wire material.

Clicking the **Src Dat** button runs enough of the NEC engine to generate the Source data report, which is:

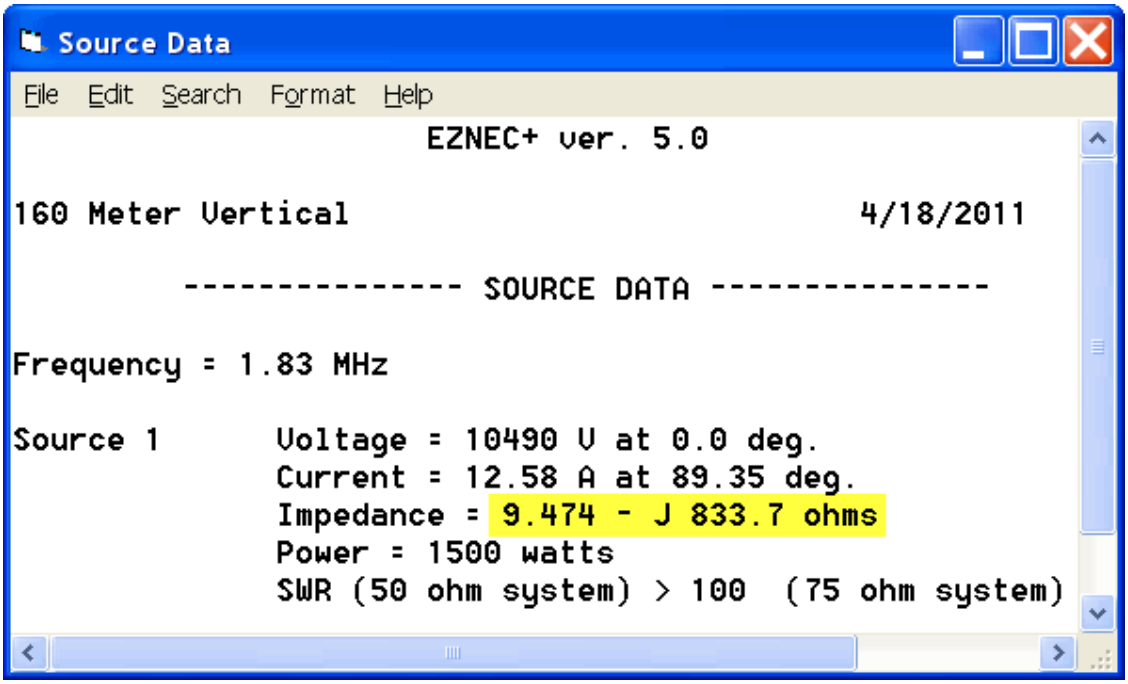


Figure 32 – Vertical Source Data

The impedance at the feed point is **9.474 – j 833.7 Ohms**. Of course 8 of those 9.474 Ohms of resistance are due to the resistor representing ground loss.

Clicking on the **Load Dat** button displays the Load Data, which is a proxy for the ground loss. The report is:

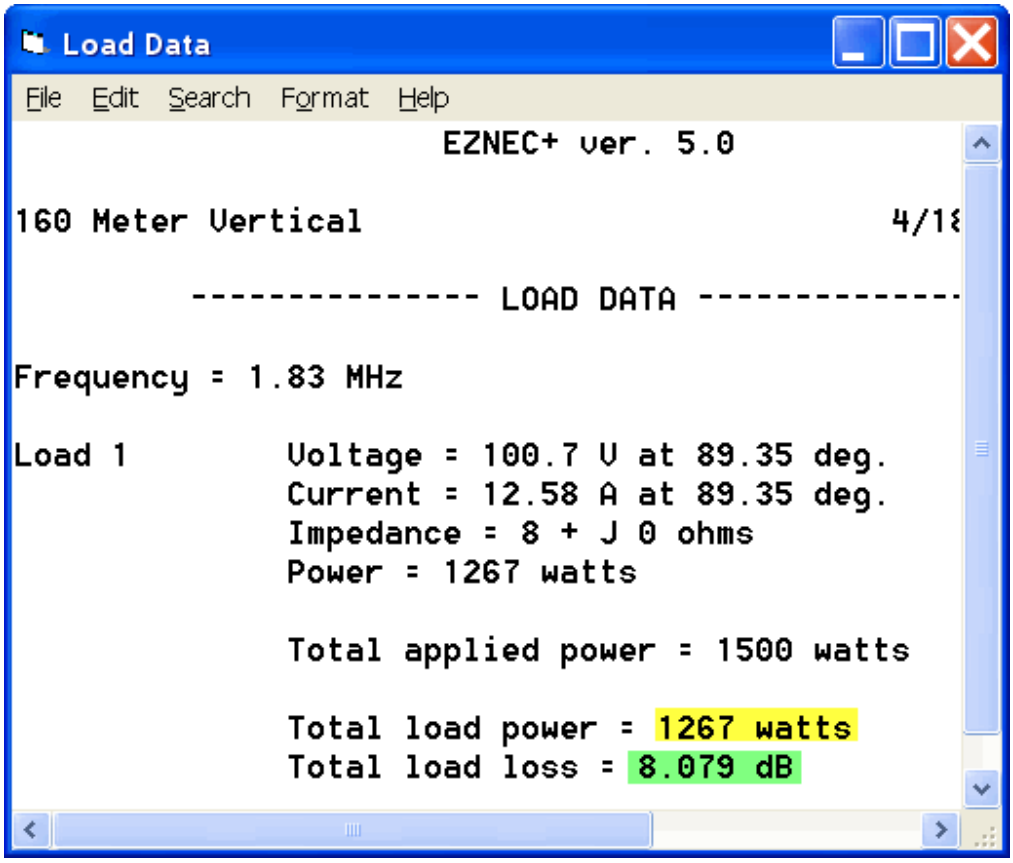


Figure 33 – Vertical Load Data

Of the 1500 watts supplied to the antenna, 1267 watts are dissipated in the ground loss. The impact of that loss on the gain is 8.079 dB. In other words, if the ground loss was 0 Ohms, we would have 8.079 dB of added gain.

This Load data provides motivation to reduce ground loss in the system. It is easy to experiment with ground loss by simply changing the value of the Load resistor. For example, if we were able to cut the loss in half, the resistance would drop to 4 Ohms. At that level, the gain would increase by 2.3 dB. For now, let's leave the value at 8 Ohms, and add the matching network.

We need an L network that matches 50 Ohms to 9.474 – j 833.7 Ohms at 1.83 MHz. Although EZNEC allows us to model an L network, it has absolutely no mechanism short of trial and error to compute the values for the network. There are many calculators and formulas that will compute solutions. One that I use is the L Network module in the ON4UN Low Band Software. After specifying the two impedance values, it produces the following solutions:

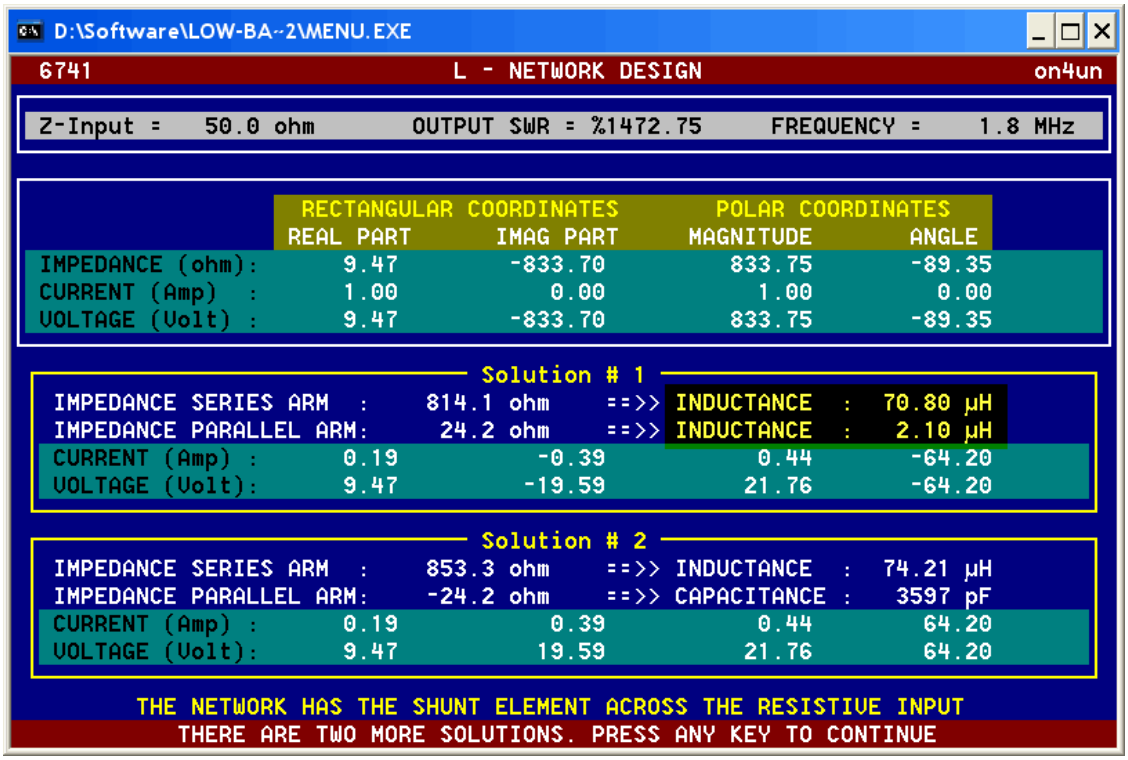


Figure 34 – L Network Solutions

I'll go with the first solution that uses two inductors, one in series, and one in shunt. The large series inductor is often called a *loading coil*, and the shunt inductor is called a *matching coil*. In fact, some folks like to approach a design such as this by first figuring out the amount of inductance needed to *tune out* the capacitive reactance of the short vertical. The second step is to match that low resistance up to 50 Ohms. I simply chose to *cut to the chase* by doing it all in one L Network step.

To add an L Network to the model, click on the **L Networks** Selection Button to display the dialog. I added one L Network, and connected it as follows:

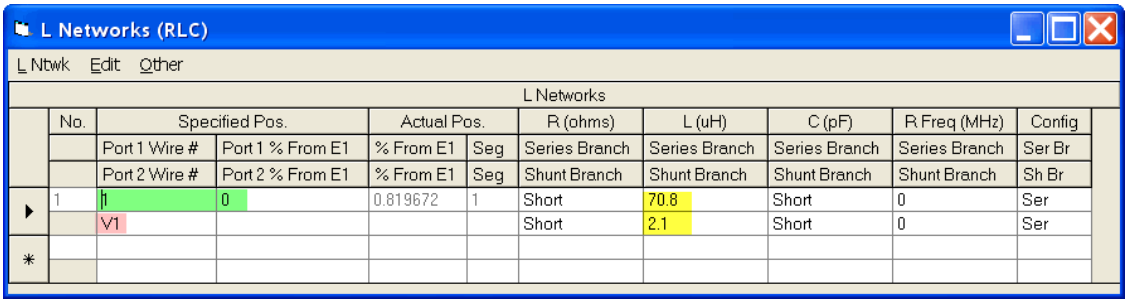


Figure 35 – L Network Model

The key values to note are that the 70.8 uH and 2.1 uH inductors from the ON4UN program have been transferred to the series and shunt arms of the L network. Port 1 connects to the antenna, which means Wire 1, segment 1, and Port 2 connects to a Virtual Wire that provides the attachment point for the Source.

We can again draw a diagram of what we have constructed in a more familiar representation.

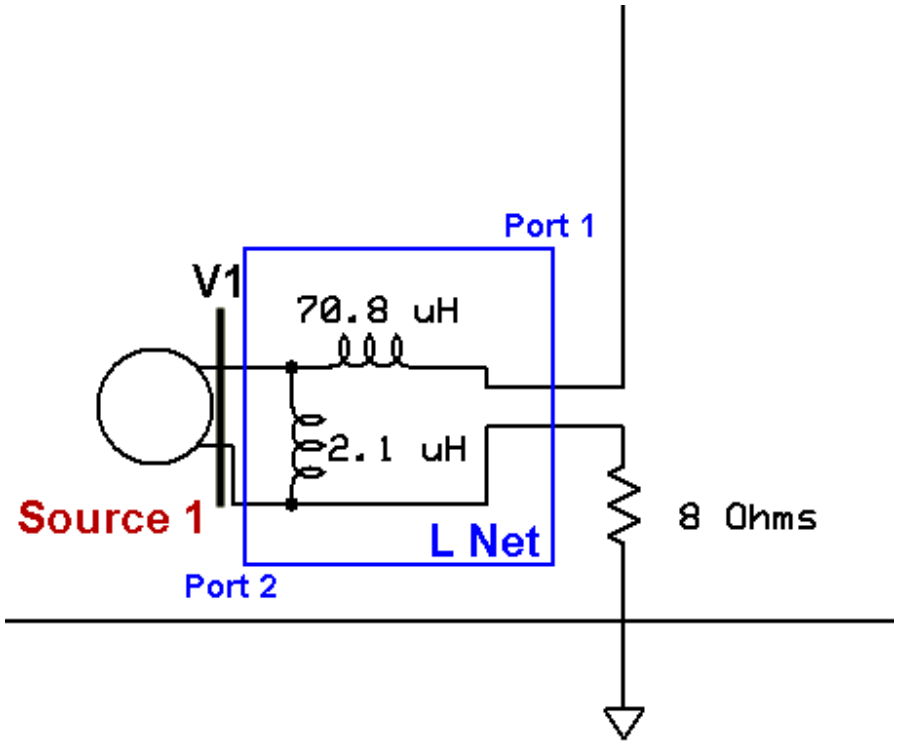


Figure 36 – L Network Primitive

The Source connects to Port 2 of the L Network – the shunt port – via Virtual Wire V1. Port 1 connects in place of the Source in the previous model. The series and shunt components in the L Network have the values taken from the L Network calculator program.

The confidence test for developing the correct component values and model is the Source impedance, which should be 50 + j 0 Ohms. The reported value at 1.83 MHz is:

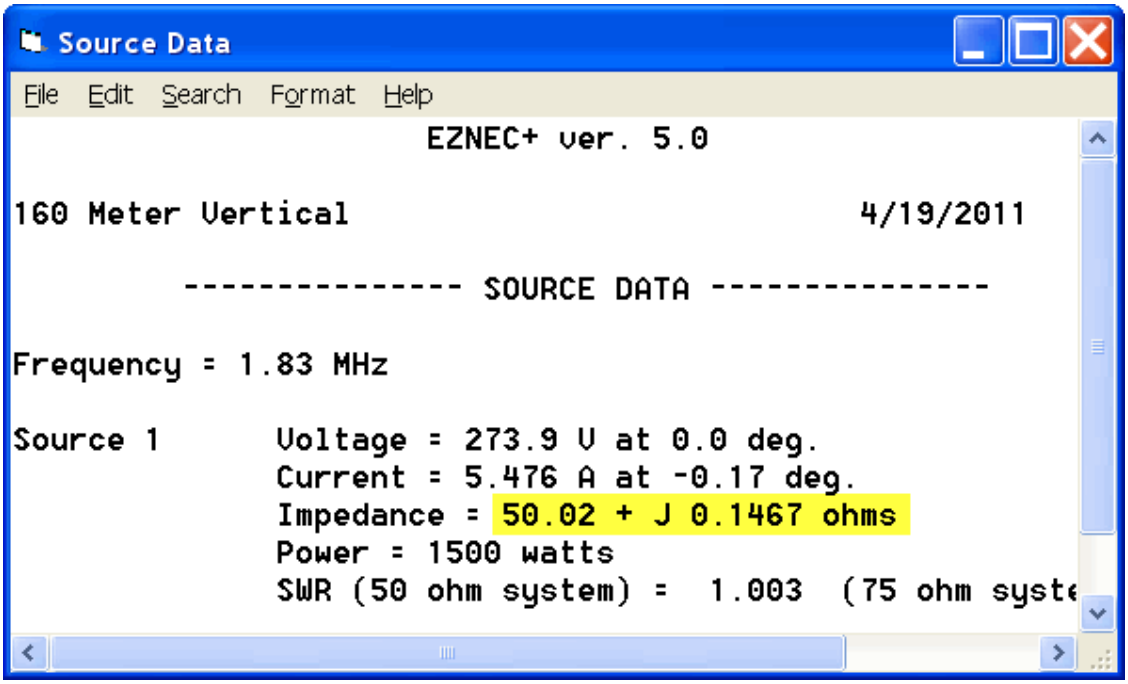


Figure 37

The impedance is darn close to 50, so things look good. We can also perform an SWR sweep at this point, since we have modeled the antenna **and** the companion feed system. A sweep over the first 100 KHz of the band reveals:

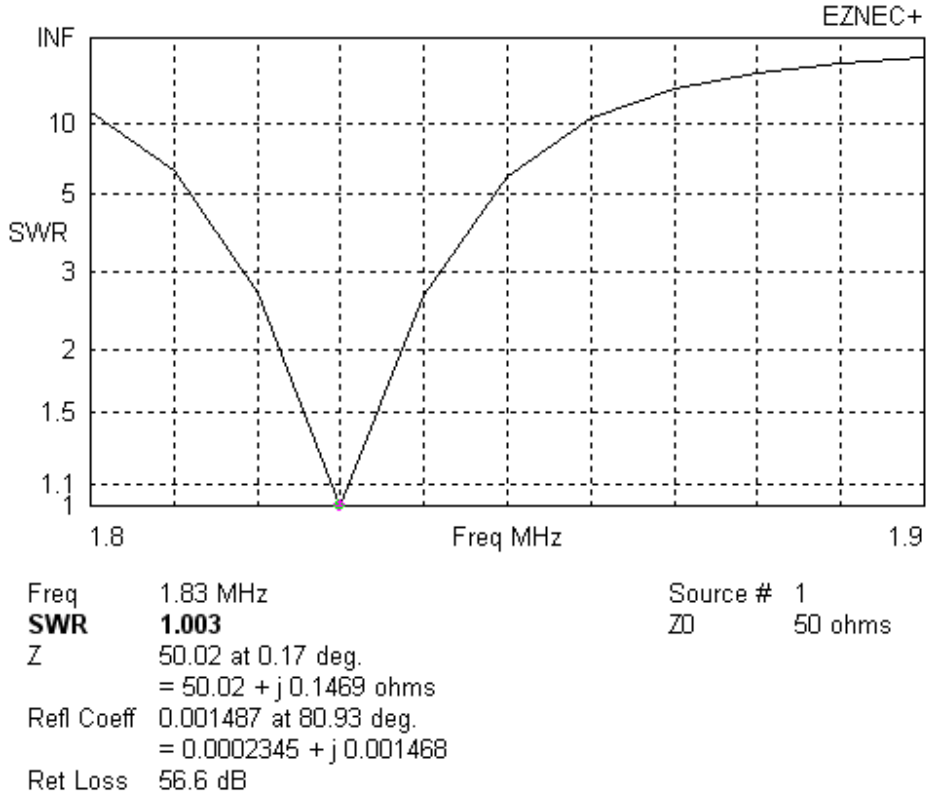


Figure 38 – Vertical SWR Sweep

We hit the target frequency dead on, but, as expected for a very short vertical, we have very little bandwidth.

This model is still a bit artificial in that we are not trying to model the loss in the two inductors in the L network. Fortunately, that's not hard to add. If we were to take reasonable care in winding the two inductors, we might expect a Q of 200. That means that the ratio of reactance to resistance is 200. We would have to be very careful with the 70 uH inductor, since that's a big whopping value that will probably be self-resonant in a few MHz above the band. Still, if we take the reactance values from the L network calculator, and divide by 200, we get resistance values for the inductors. They are 4.07 and 0.12 Ohms. Those are added to the L network as follows:

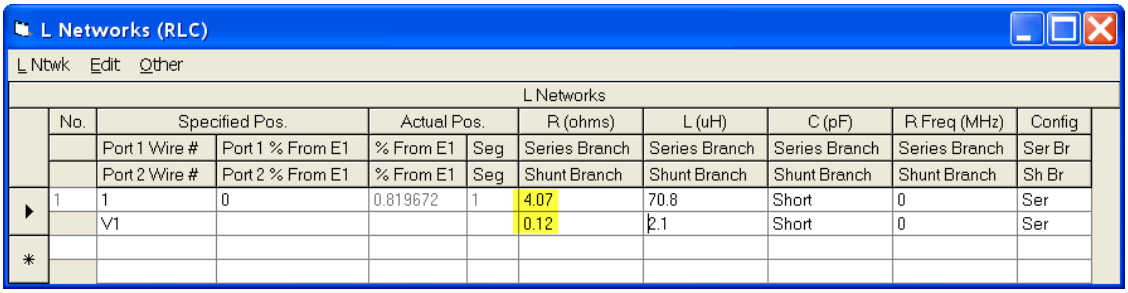


Figure 39 – L Network with Loss

The *Other* menu command is used to select the component value representation type. In this model, we want to use RLC values. The other choice is to measure resistance and reactance in Ohms.

When this model was executed, the Source Data window with the SWR value reported:

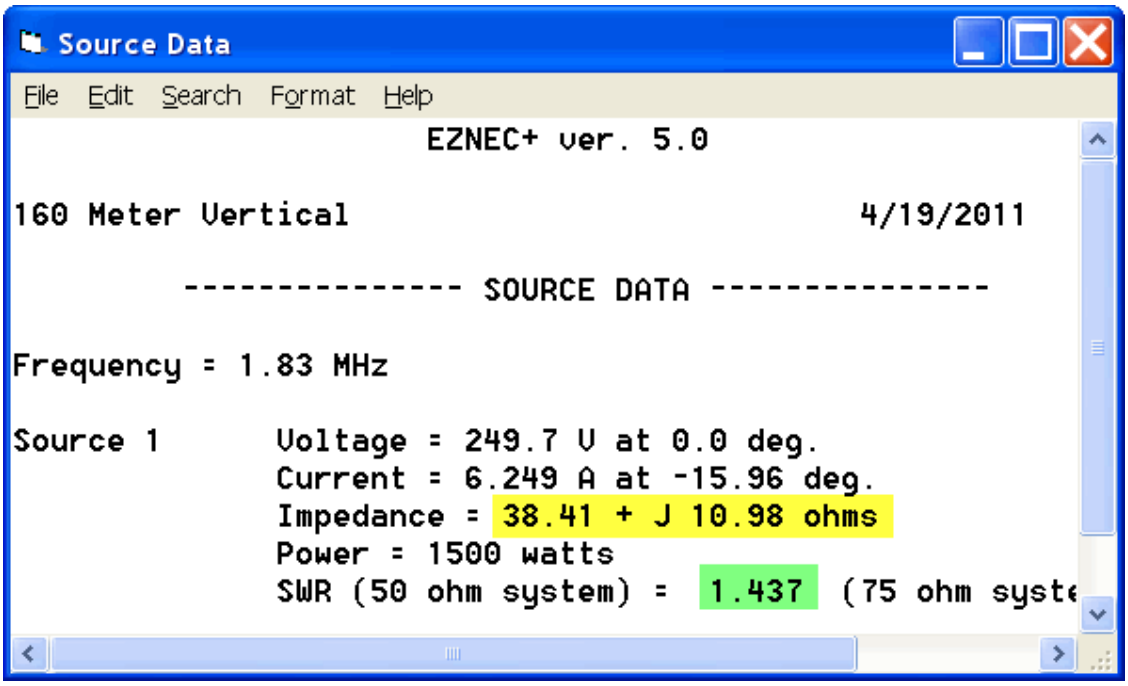


Figure 40 – Source Data with Lossy L Network

These results are a bit odd, since we clearly lost our nearly perfect 1.0 SWR and 50 Ohm impedance. The problem is that the 4 Ohms of loss resistance we added in the series inductor is relatively close to the original 9.47 Ohms that we asked to match. So, in adding the loss, we threw off the calculations. This is a good example of why modeling without loss, which a lot of folks like to do, can be a little dangerous. In this case, the quickest way to get close to a new solution is to understand that the 4 Ohms of extra inductor loss is effectively in series with the

impedance we asked to match. So, we need to add that resistance to the original 9.47 Ohms.

I reran the ON4UN matching program, this time with an impedance of 13.47 – j 833.7 Ohms. The updated L network values were 70.58 uH and 2.64 uH. I modified the L Network description in the model, and reran the model. Now, the Source Data reveals:

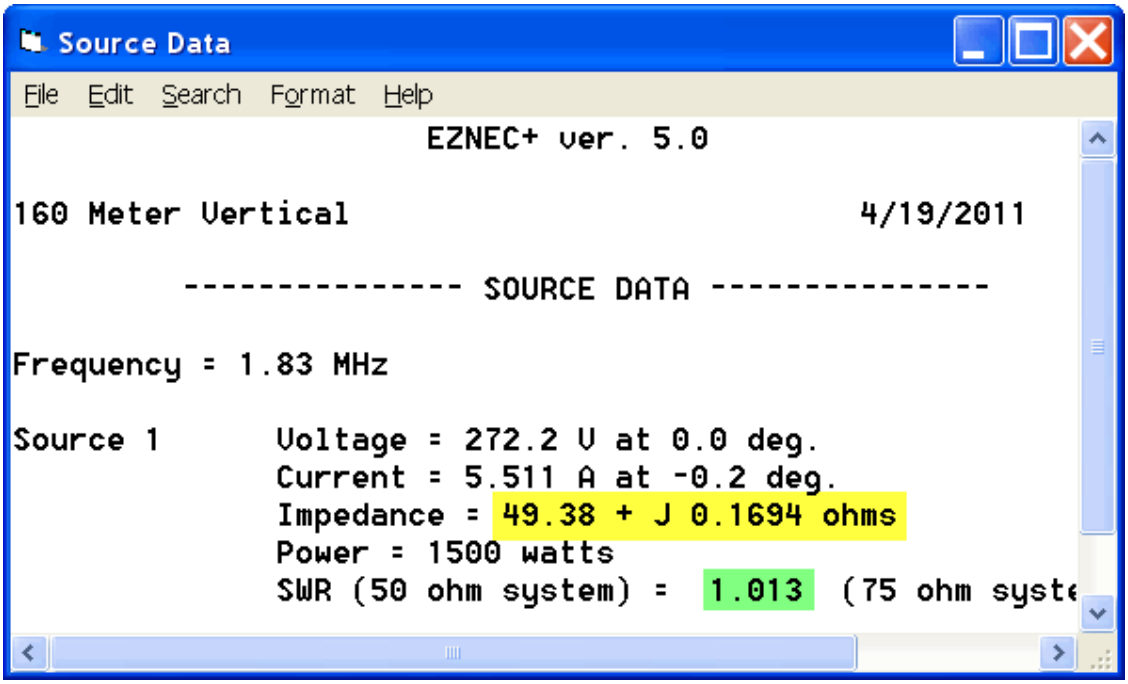


Figure 41 – Source Data with Lossy L Network and Updated Component Values

At this point, we have a very good match, and, loss inserted into the ground and matching components. What did this do to our antenna gain? I saved an elevation plot from the lossless version, and we can compare that to the version with loss in the L network.

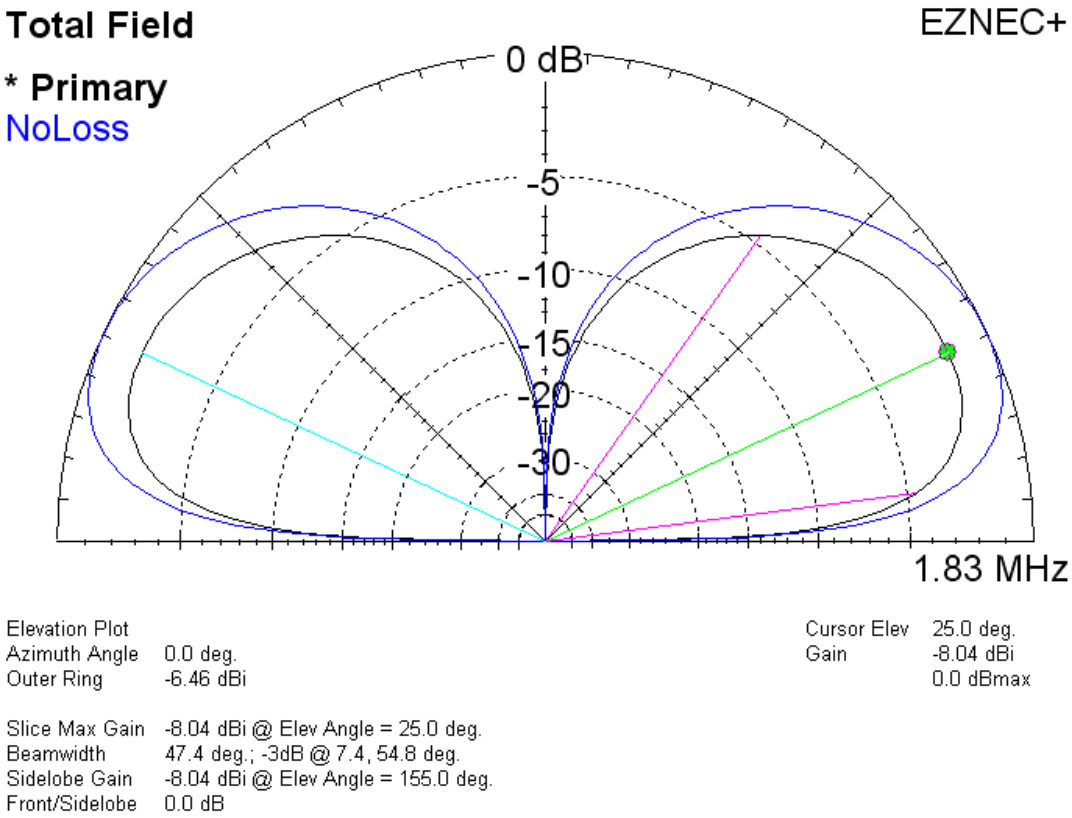


Figure 42 – Lossy and Lossless L Network Gain Comparison

By adding reasonable loss values to the L network components, the maximum gain fell an additional 1.58 dB.

We might as well take a look at the Load Data report. It is:

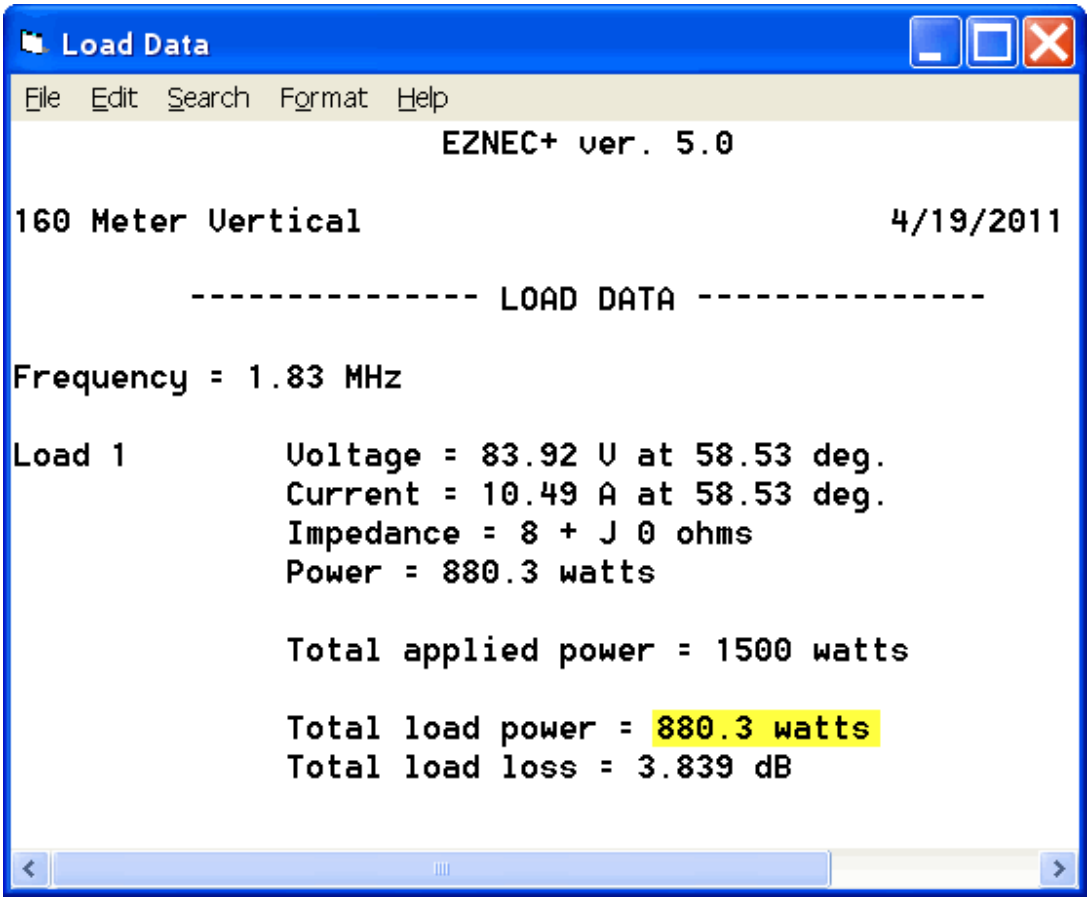


Figure 43 – Load Data Report with Lossy L Network

We see that the power in the resistor is 880.3 watts, which is a reduction from the value of 1267 watts in the case of the first model that only had ground loss represented by the resistor. It makes sense that the power in the resistor would drop, because we know that we are dissipating some power in the lossy inductors. But, can we get a report on their loss values, as well as current through them and voltages across them? As far as I know, the answer is **no**. This is because we modeled the L network with an EZNEC L Network primitive and only Loads provide all of the data reporting. My belief is that this is the case because the L Network primitive provided by EZNEC is actually constructed from a NEC-2 *Network*, which is simply a different abstraction than a NEC-2 *Load*, and they provide different information coming out of the engine. Said another way, an L Network is not a fundamental primitive in the NEC-2 command set, but EZNEC has been programmed to provide it as one, and EZNEC internally converts it into what NEC-2 allows. This is an example of how the modeling software can advance and grow, while being built on top of a static NEC-2 engine.

Here's where we can go back to the days before the L Network primitive, and construct the L network out of 2 Loads as opposed to one L Network. Although

the process is more complicated, and far less intuitive, we can get the Load Data report to provide some insight into the power dissipation and component ratings.

Here is my diagram of the proposed solution. It integrates the L network back onto the body of the single Wire antenna.

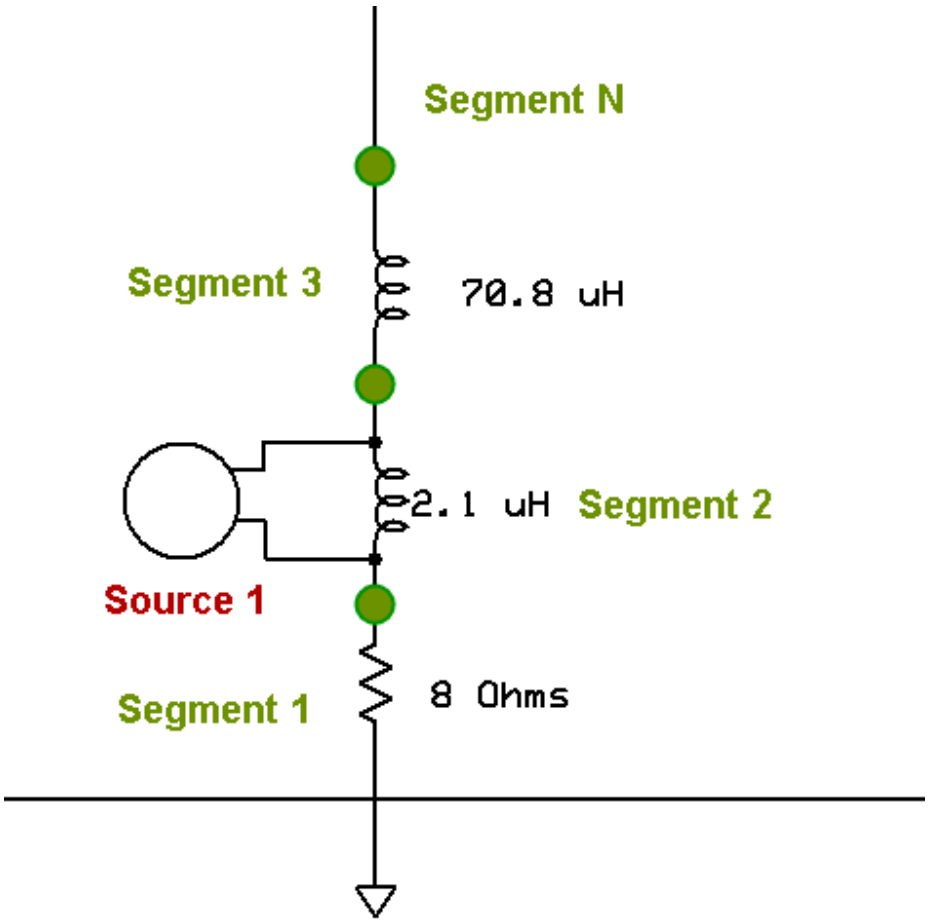


Figure 44 – L Network Composed of Loads

If you look at this long enough, you can convince yourself that the L network is now merged into the antenna Wire. The 8 Ohms of ground loss are in the first segment. The second segment has the shunt inductor connected in parallel with the Source. The third segment includes the series inductor.

The shunt inductor is placed in parallel with the Source by changing its external connection specification. Here is the new Load list, now with 3 Loads.

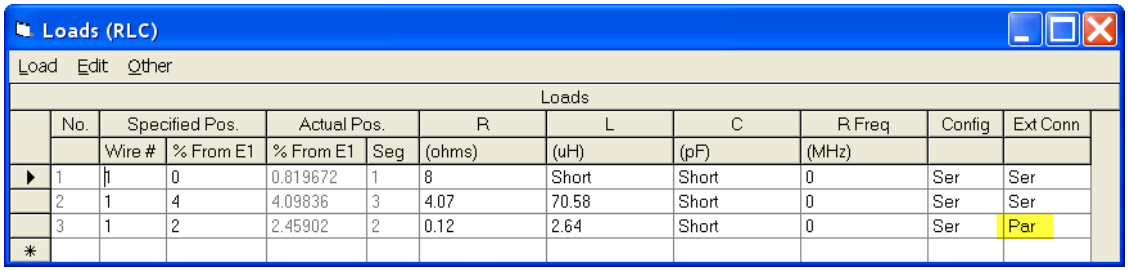


Figure 45 – L Network Loads

Load 2 is the series inductor, and Load 3 is the shunt inductor. The parallel external connection on the shunt inductor is highlighted.

We have one more detail to take care of. In looking at the diagram of the feed system, it's clear that the first 3 antenna segments are now devoted to the feed system, and are effectively removed from the antenna. Since the antenna has 61 segments, and we are now using 3, this implies that the antenna is now 3/61 or 5% shorter So, to compensate, we need to make the antenna 5% longer. With a starting length of 33 feet, the final length is 34.6 feet. This is why I created the Wire with 61 segments to begin with. More segments means less adjustment. Since this antenna is so short relative to the frequency, the rates of change are so high that even a 5% difference is significant.

There are other ways to approach creating the L network, some using additional wires. I don't know if there is a *best* way. Since getting a lot of short Wires close together in a model can cause problems, I tend to prefer fewer and straighter Wires. This is another area where modeling becomes an art, not a science. The L Network primitive is very convenient, and I suspect even more accurate. On the other hand, I wish that it provided data as if it were a Load.

With the longer antenna and added Load components, the SWR sweep is:

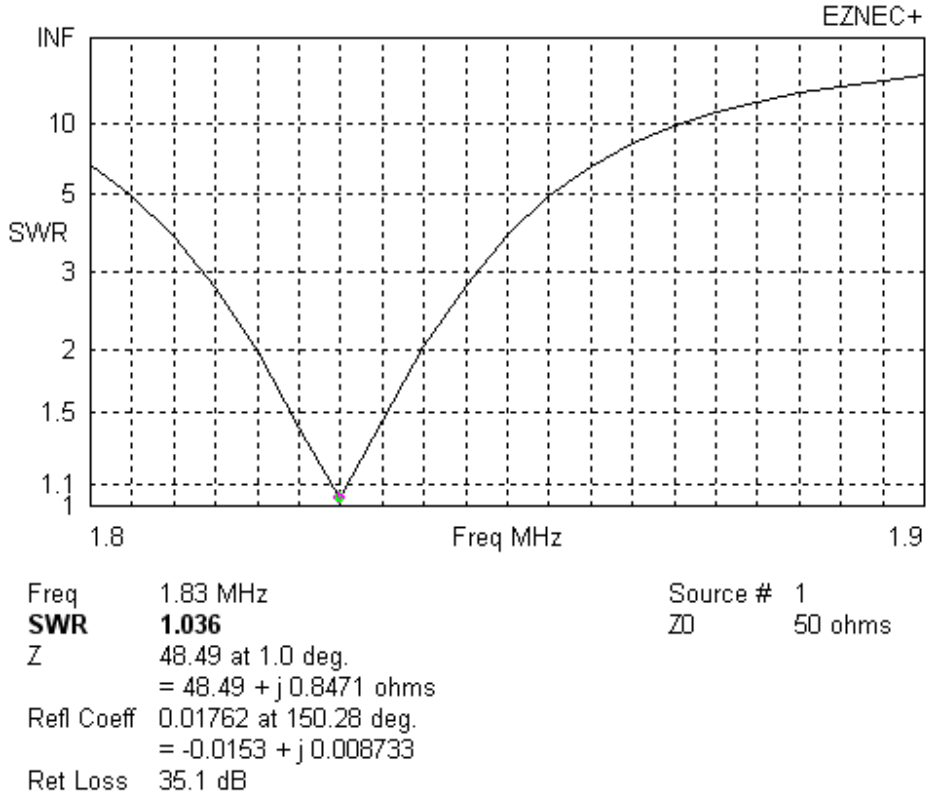


Figure 46 – Load L Network SWR Sweep

The SWR performance tracks the performance with the separate L Network.

As far as gain goes, we expect the gain to be close to the last model, since all we did was recast the L Network in a different form. The gain comparison is:

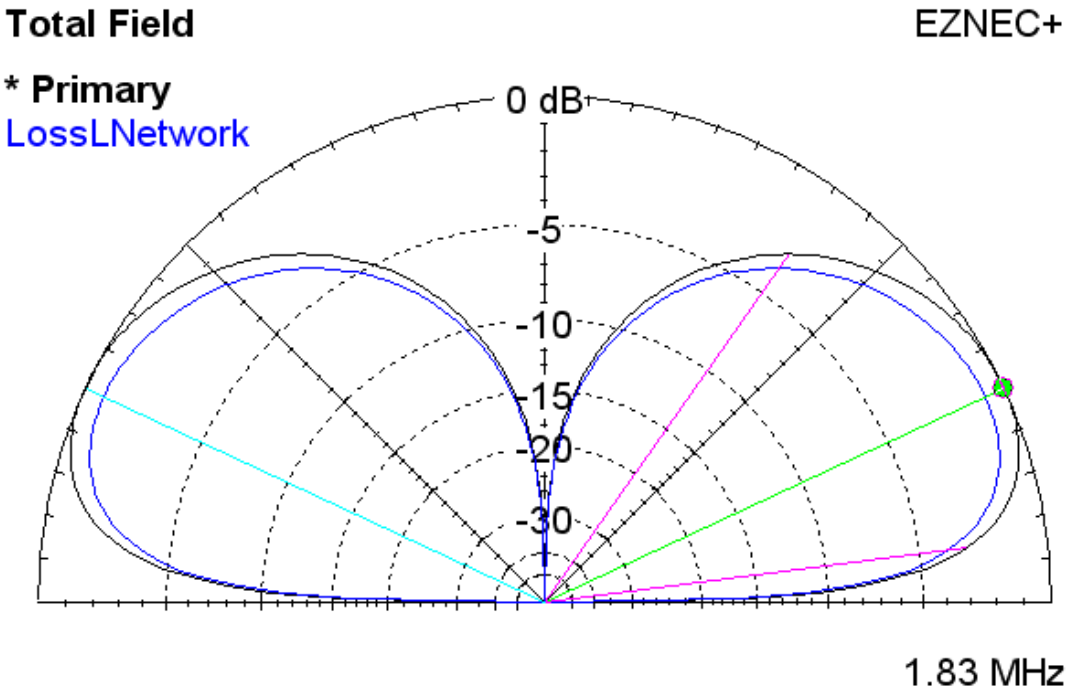


Figure 47 – L Network Gain Comparison

The gain with the L network constructed from Loads is 0.73 dB greater than the version using the L Network primitive.

The reason why we created the L network out of Loads is so that we can look at the Load Data report. Those results are:

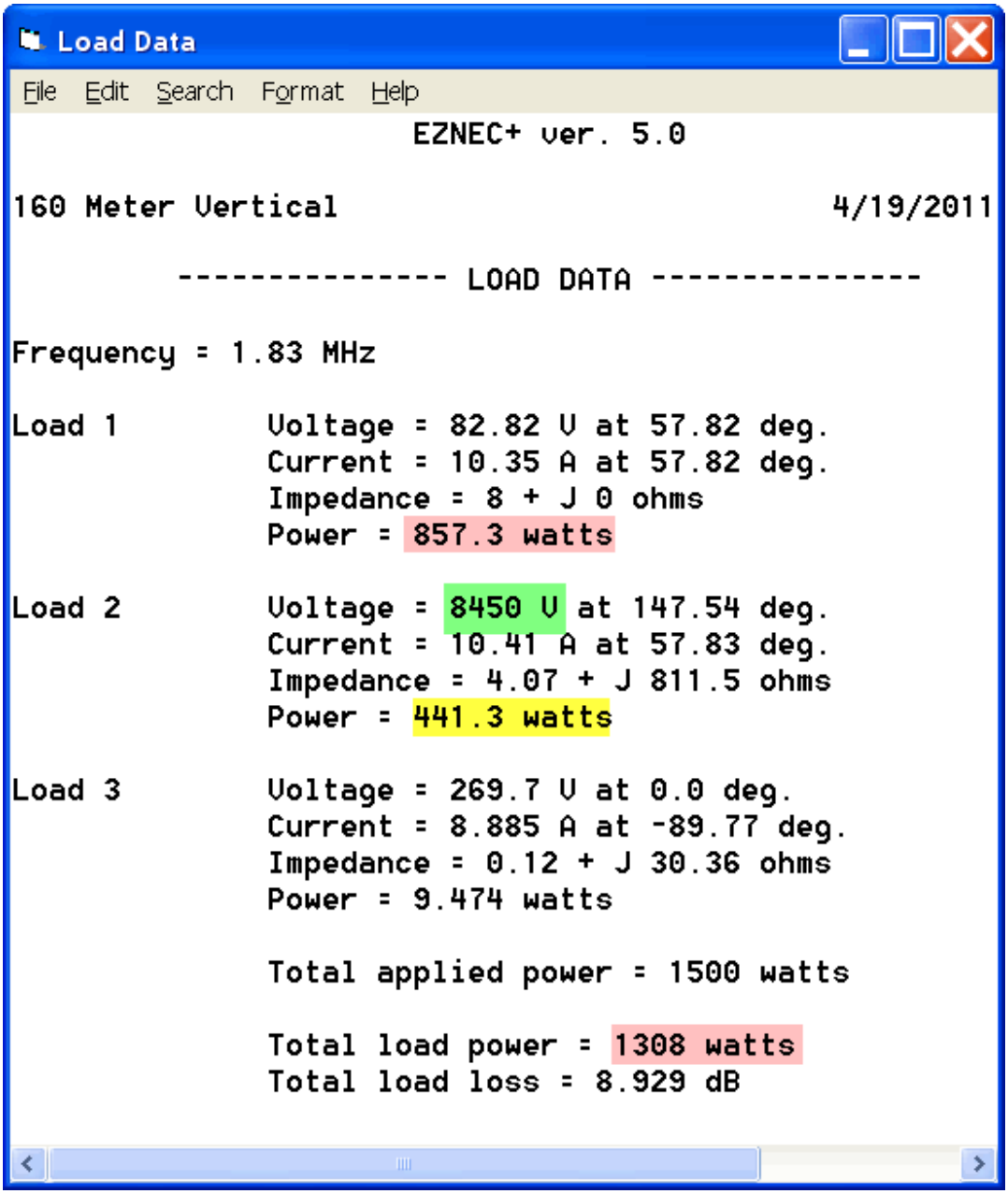


Figure 48 – L Network Load Report

When we last looked at the 8 Ohm resistor, the power dissipation was 880.3 watts, and this model has it at 857.3 watts, which is a 2.7% difference. So, there are some differences as you look at alternative models of the same antenna.

Of note are the values for the large series inductor, Load 2. This inductor, with a Q of 200, will dissipate 441.3 watts out of the 1500 that are supplied. If you build this antenna, be sure that your series inductor can handle the power. Perhaps more noteworthy is that the voltage across the series inductor is a whopping

8450 volts. This is not really a shock (no pun intended), since we have a low impedance that implies a high current, and a large inductor with a high reactance. Ohms Law, V = IZ, says that a lot of voltage will be across that part. This value is high enough that special wiring insulation is probably needed, and, if you wanted to switch in the 160 operation and keep the 40 meter operation, a vacuum relay would probably be needed.

The last Load Data report shows that due to the losses in the ground, the loading coil and the matching coil, we are down about 10 dB from a 100% efficient antenna. Although it's not the job of this document to design a higher gain antenna, we can use that obvious goal to show another modeling example.

Antenna *theory* states that the efficiency of an antenna is equal to the radiation resistance divided by the sum of the radiation resistance and loss resistance.

$$Efficiency = \frac{Rrad}{(Rrad + Rloss)}$$

If we want to improve the efficiency, the two obvious choices are to increase the radiation resistance and decrease the loss. The radiation resistance of the vertical is under 2 Ohms. We learned that from the first Source Data report, where the Source resistance was listed as 9.474 Ohms, and we knew that 8 Ohms of that came from the ground loss resistance Load. That's a result of being very short relative to the textbook 36 Ohm ¼ wavelength vertical. Let's say that our goal is to increase the radiation resistance since it appears as if we have a lot of room for improvement.

One way to increase the radiation resistance is to simply make the antenna taller. If that's not possible, then we can change the location of the loading device. In the initial design, we have what amounts to *base loading*, with the large 70 uH inductor in the L network. One of the ways to increase the radiation resistance is to move the loading inductor further up the antenna. This is often called *center loading* or *top loading*. It's all really a continuum, where the loading inductor can be located at any point along the antenna. On one hand, the further the inductor is away from the feed point, the higher the radiation resistance. On the other hand, as it moves further away, the inductance needed to maintain resonance at some frequency increases.

Several sources discuss the merits of base versus center versus top loading. Two that come to mind are the ON4UN book, and a series of pages by L.B. Cebik on his web site.

I had the opportunity this last winter to supply some modeling data for a 160 meter vertical erected by Josh, 6Y5WJ, in Jamaica. Josh had a vertical similar to

this example, although it was slightly taller, at the 40 foot level. To improve the efficiency, he wanted to use top loading. The idea was to move the inductor all the way to the top of the vertical, right under a horizontal fan of wires often called a *top hat* or a *capacity hat*. Based upon available materials and wind loading considerations, the idea was to construct the top hat from 40 wires enclosed in a 10' X 10' square created by wooden spreader arms on the diagonals.

Here is a picture of Josh's top hat while still on the ground.

Figure 49 – 6Y5WJ Top Hat

A number of the capacity hat wires can be seen reflecting the sunlight.

Josh was very clever and inventive in using available materials while keeping the wind loading as low as possible. Erected on top of the vertical, the antenna looked like:

Figure 50 – 6Y5WJ Vertical with Top Hat

Off in the distance is the Caribbean Sea, viewed from this beautiful island location.

It would have been somewhat tedious to model all 40 top hat wires by hand. Being in a square frame, the Wires do not have a single common length. We can tell that the shortest wire runs from the middle of a side to the antenna center, and the longest wire runs from the end of a side (at a corner) to the center.

From the square perimeter in the design, there should be 8-way symmetry in the top hat. Still, it would be necessary to compute the end points of the 5 wires on ½ of a side, and then reflect and rotate those around to fill in all 8 sections.

The solution I used, which was more than good enough for estimating the impact on the antenna, was to use the EZNEC *Create->Radials* menu command on the Wires window. Now you might think that a radial needs to be on the ground. But, EZNEC does not require that, and, it simply takes a prototype Wire or group of Wires, and makes copies, attaching their ends together and spacing them equally around the X, Y plane. So, by specifying a single Wire manually, and they using the radial command with a count of 40, I was able to create the following model.

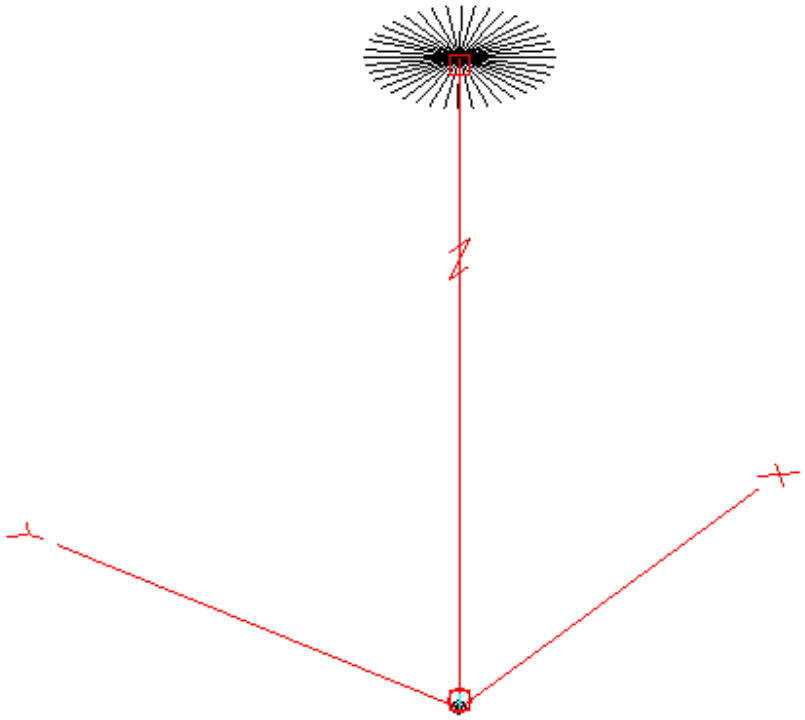


Figure 51 – Top Hat Radial Wires

If you wanted to improve the fidelity of the model, a second approach would be to calculate one side of the square capacity hat, and then make copies of that set of Wires and then rotate the copies around the center point.

My point in this section is to show that the days of having to manually compute all Wire end point coordinates are largely over, and that the command set present in the program allows the creation of complex wire topologies quickly and easily.

At the risk of engaging in self promotion, I wanted to mention the LBDXView program that is a postprocessor for EZNEC output, as well as several other

sources of antenna data. It is a free program that I wrote that is available on a web site, as well as on the CD included with the 5th edition of the ON4UN book.

It shows far more data in a sweep format than EZNEC, although the data comes from EZNEC (through a sweep data file that EZNEC generates). Although EZNEC executes an SWR sweep, it does it for the current model, and comparisons are not possible. Here is the LBDXView sweep for the SWR data for the lossless and lossy L networks.

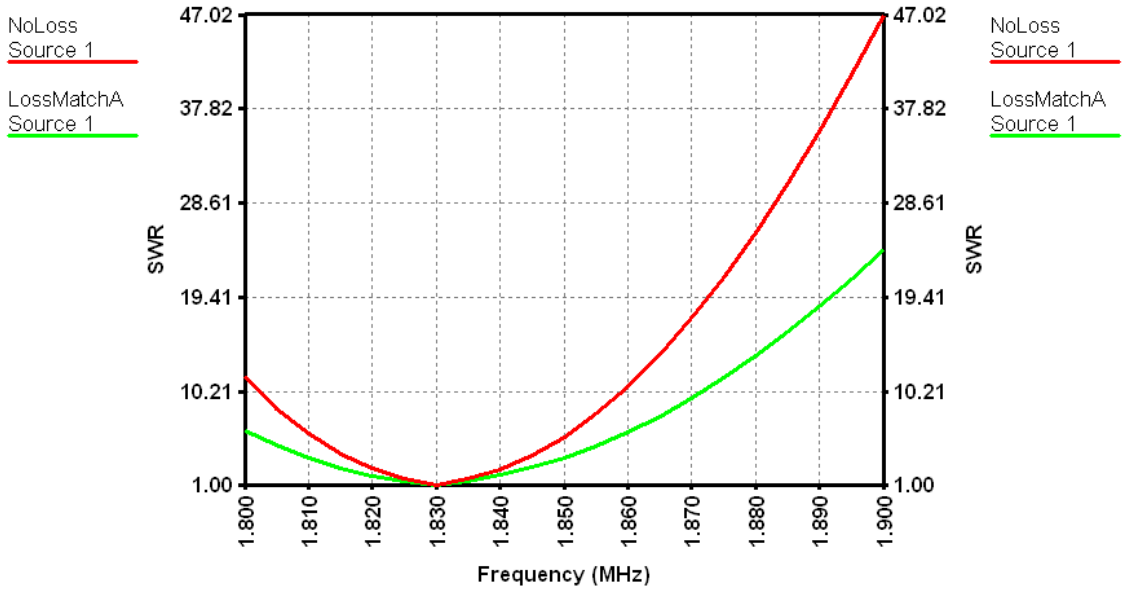


Figure 52 – LBDXView Output

If you find yourself wanting to compare model data as graphic or tabular sweeps, please check out the program.

Transformers – Flag Antenna and Array

The EZNEC transformer *insertion object* will be illustrated using the *Flag* receiving antenna. The Flag antenna, associated with Earl, K6SE (SK), is one of the last decade's new directional receiving antennas that is relatively compact, and also broadbanded (non resonant). A single antenna provides good performance from below the 160 meter band to above the 40 meter band. The antenna is described in the ON4UN book, as well as many web pages. I have built a number of them over the years, both as single antennas and in arrays of up to three antennas. Although I used models to help my designs, modeling was most useful in helping me reveal my errors! I'll explain this a little later.

The basic form of the antenna is:

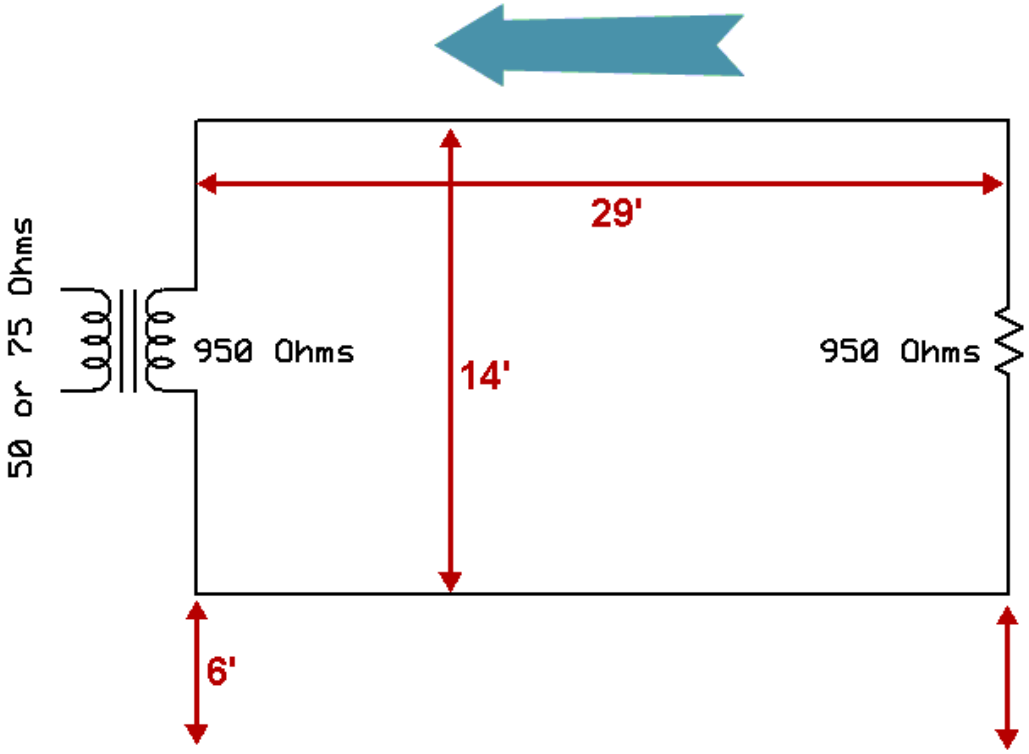


Figure 53 – Flag Antenna Schematic

The direction of maximum signal strength is towards the transformer.

The EZNEC model consists of 4 Wires, with a Load on one representing the resistor, and a transformer on the opposite side. The transformer has a pair of connections, much like a transmission line. One port is connected to the antenna Wire. The other port will be connected to a *virtual wire* which will be a junction point for the Source and the transformer.

The Wire table for the single Flag model is:

Wires

Wire Create Edit Other

Coord Entry Mode Preserve Connections Show Wire Insulati

Wires

	No.	End 1				End 2				Diameter	Segs
		X (ft)	Y (ft)	Z (ft)	Conn	X (ft)	Y (ft)	Z (ft)	Conn	(in)	
	1	0	0	6	W4E2	0	0	20	W2E1	#12	21
	2	0	0	20	W1E2	29	0	20	W3E1	#12	21
	3	29	0	20	W2E2	29	0	6	W4E1	#12	21
▶	4	29	0	6	W3E2	0	0	6	W1E1	#12	21
*											

Figure 54 – Single Flag Wire Table

The antenna view is:

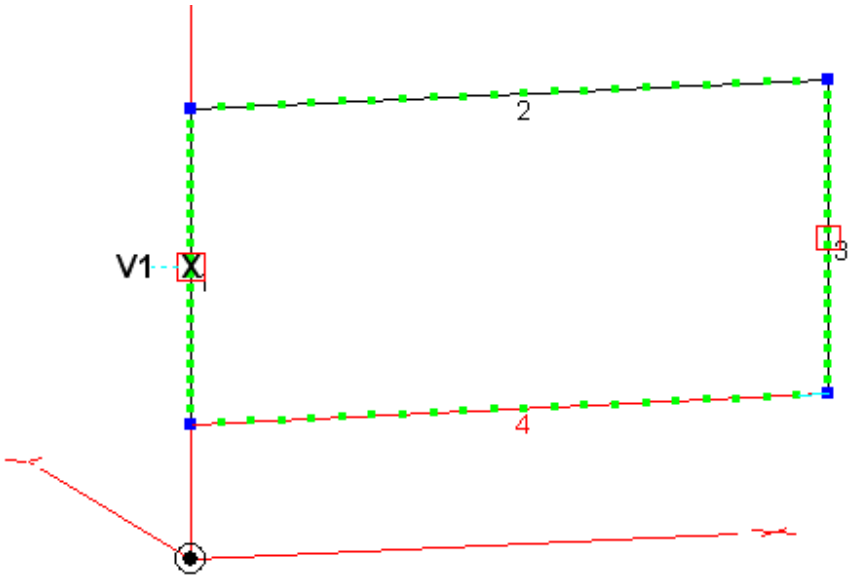


Figure 55 – Single Flag View

Wire 1, on the left, is the connection point for the transformer. The resistor Load connects to the middle of Wire 3 on the right.

To add a transformer to the model, click the **Transformers** Selection Button on the main window. This will display the Transformers dialog box. Use the menu to add one transformer. I filled in the transformer specification with the following data:

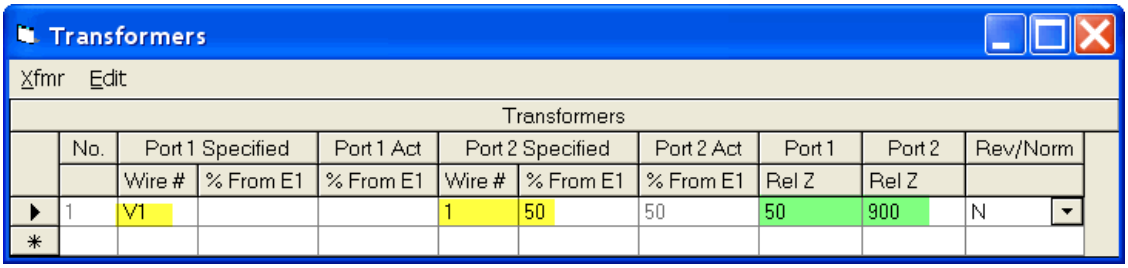


Figure 56 – Single Flag Transformer

The two yellow highlights show the connections to the two transformer ports. Port 1 is connected to virtual Wire **V1**, and Port 2 connects 50% of the way down Wire 1. The green highlight shows the impedance values of the two ports. I set the V1 side to 50 Ohms, and the antenna side to 900 Ohms. Although the actual Source impedance may be closer to 950 Ohms (you can determine that value by connecting the antenna directly to the Source without the transformer), in practice, 900 Ohms represents an 18:1 ratio, which might be a more practical ratio to wind or at least come close to.

The resistor is created as a Load. The specification is:

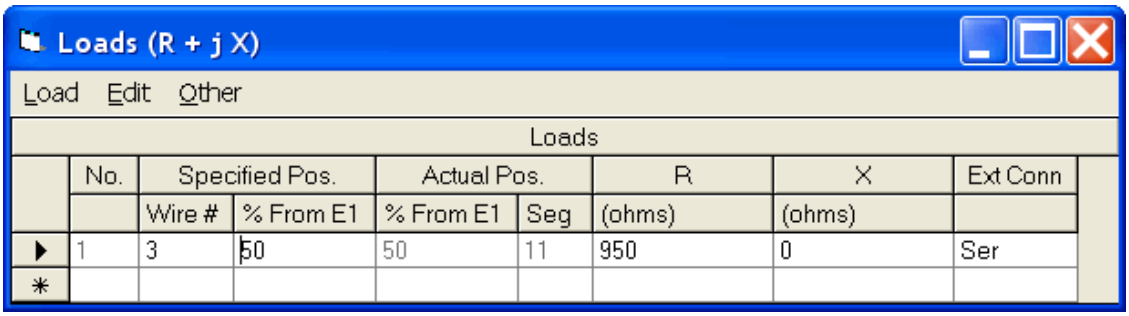


Figure 57 – Single Flag Resistor Load

The single Source connects to V1, which implies that it is connected to the transformer. The ground model is Real/High Accuracy. This completes the model.

I set the Frequency to 1.83 MHz. The 3D pattern is:

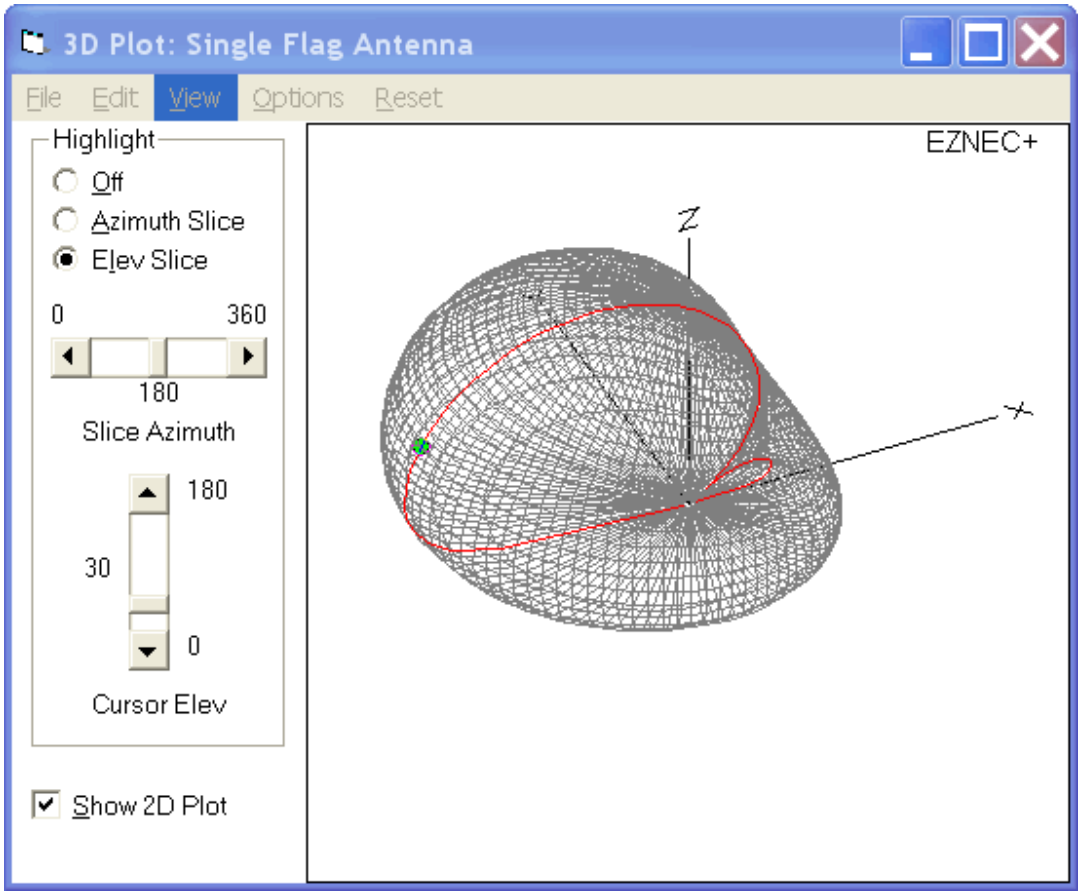


Figure 58 – Single Flag 3D Pattern

The main lobe is pointing down the –X-Axis, which is consistent with the location of the transformer and the resistor. I checked the **Show 2D Plot** control to display 2D slices. The azimuth slice is:

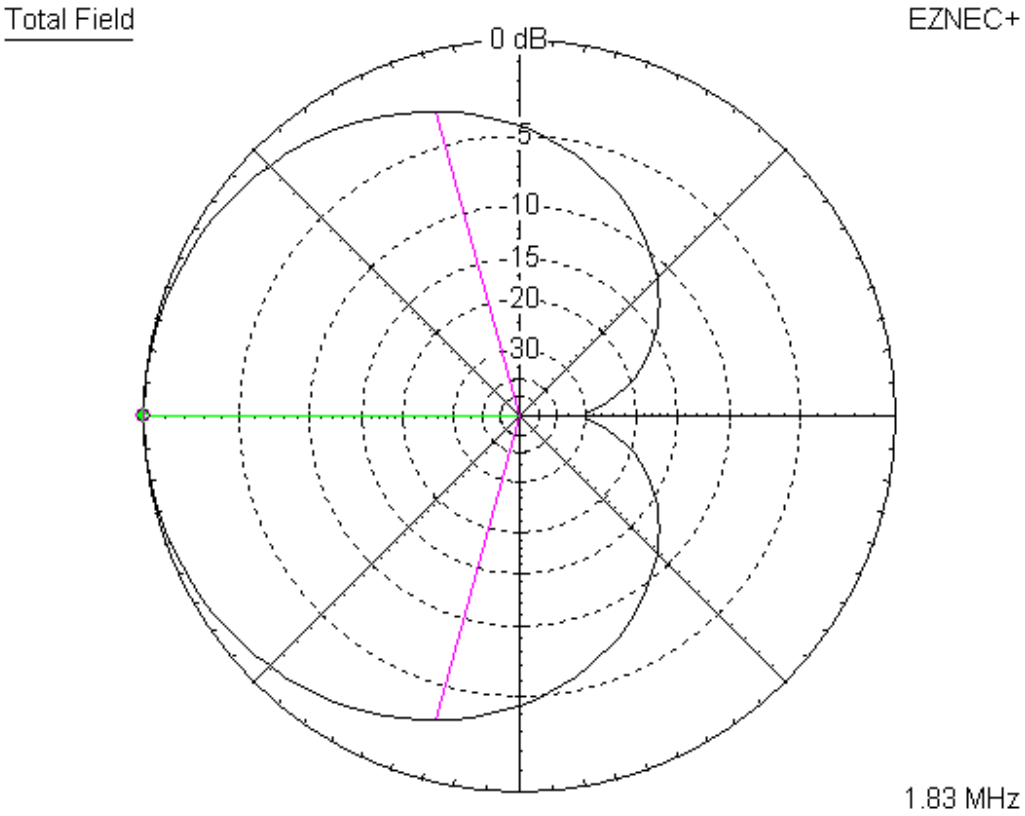


Figure 59 – Single Flag Azimuth Pattern

The slice elevation angle is 30 degrees, which is the take off angle of maximum gain. I highlighted the gain, which is -29.97 dBi. As with most lower HF receiving antennas, they have a low output signal level. The front to back ratio is a respectable 30.44 dB. The elevation slice, through the center of the main lobe, is:

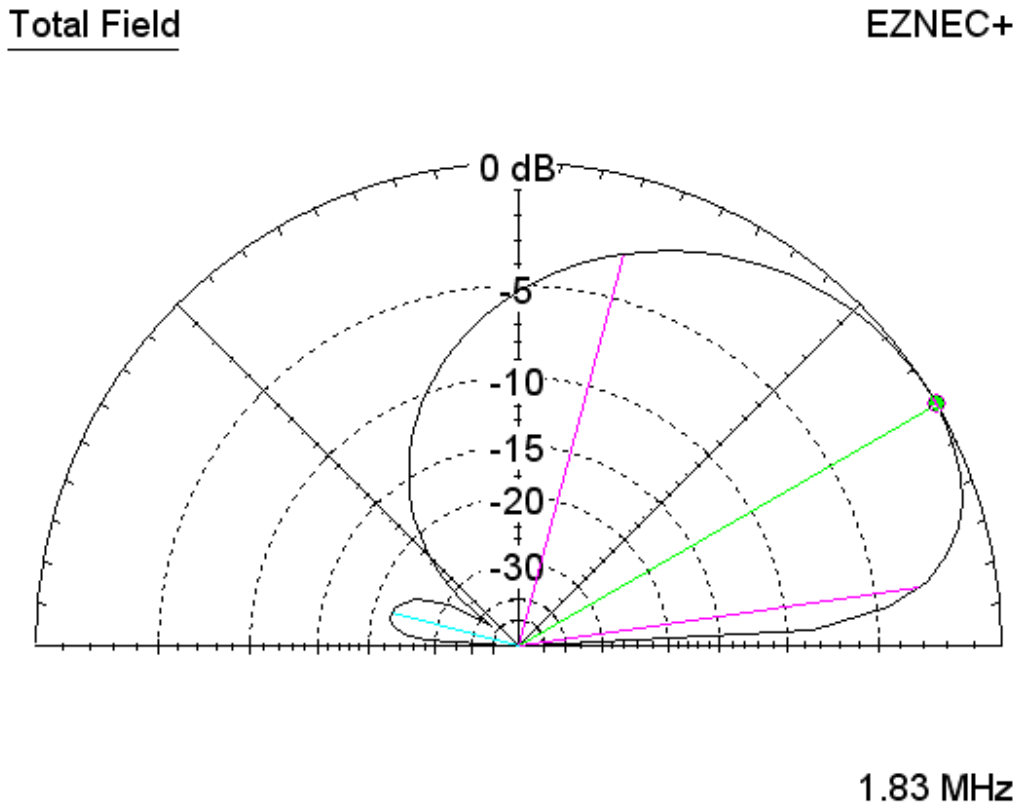


Figure 60 – Single Flag Elevation Pattern

This antenna is a good broadband design because these pattern characteristics largely remain the same as the frequency changes.

We can easily evaluate the SWR across the band with an SWR sweep. Clicking the **SWR** *Action Button* brings up the SWR sweep dialog:

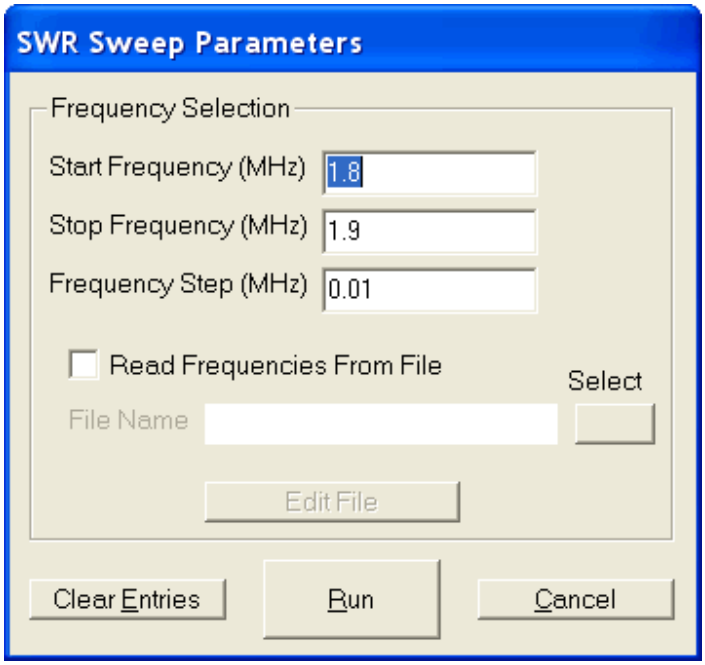


Figure 61 – SWR Sweep Parameters Dialog

I specified a sweep from 1.8 to 1.9 MHz in steps of 10 KHz. After clicking Run, the resulting graph is:

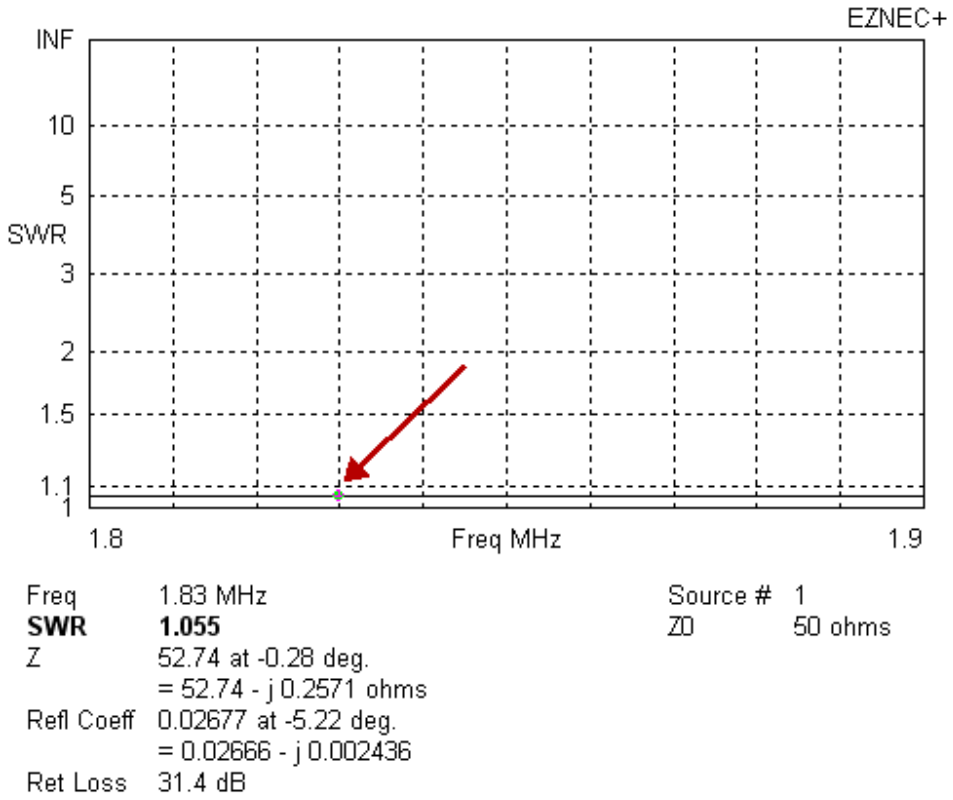


Figure 62 – Single Flag SWR Sweep, 1.8 to 1.9 MHz

The SWR is a flat line with a value of 1.055, clearly an excellent match to a 50 Ohm system. With receiving antennas the quality of the match often times is not too important. But, in the case of low gain antennas, it would be shame to waste gain in additional transmission line loss due to large mismatches.

Of course it's just as easy to measure the SWR across any span. Here's a sweep from 1.8 through 7.3 MHz in steps of 50 KHz.

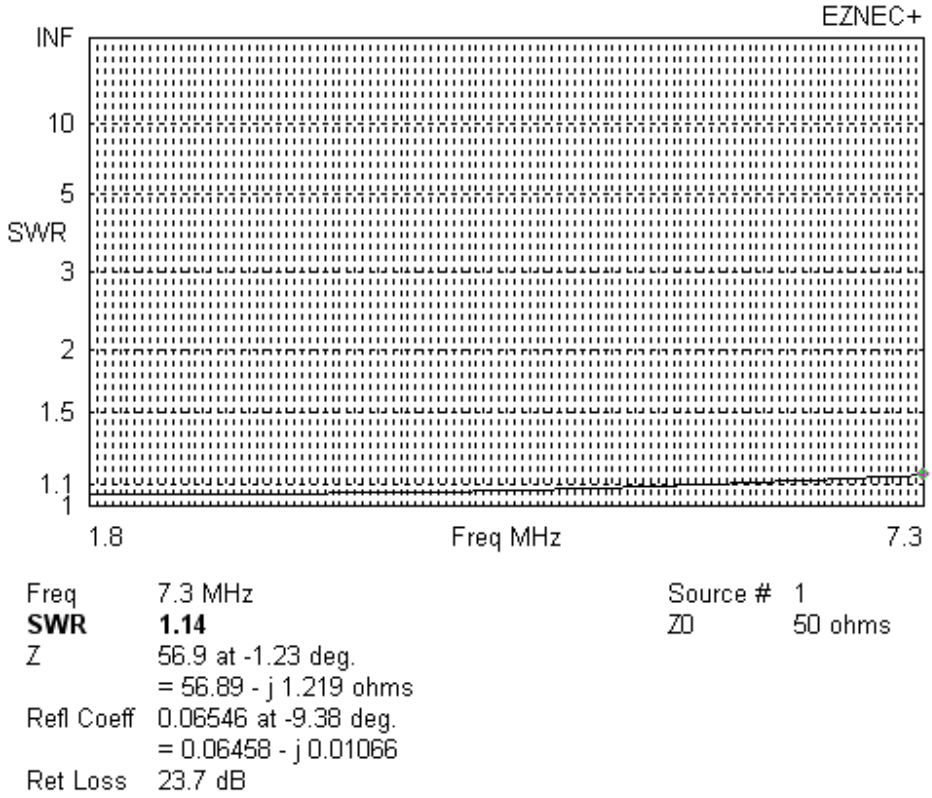


Figure 63 - Single Flag SWR Sweep, 1.8 to 7.3 MHz

At 7.3 MHz, the SWR has risen to 1.14. I should mention that the transformer in EZNEC is an ideal broadband transformer, an exact mathematical model. Real transformers are usually far from ideal. If you build one and combine it with this antenna design, you will not see such a flat SWR, especially across a wide frequency range.

Of course SWR says nothing about gain and pattern. We can take a comparative look at that within EZNEC by saving trace data at one frequency, and then comparing that to data at other frequencies after changing the model.

I started by saving azimuth pattern data at 1.83 and 3.6 MHz, values within 160 and 80 meters. I saved that trace data with a menu command on the azimuth pattern window. The command is:

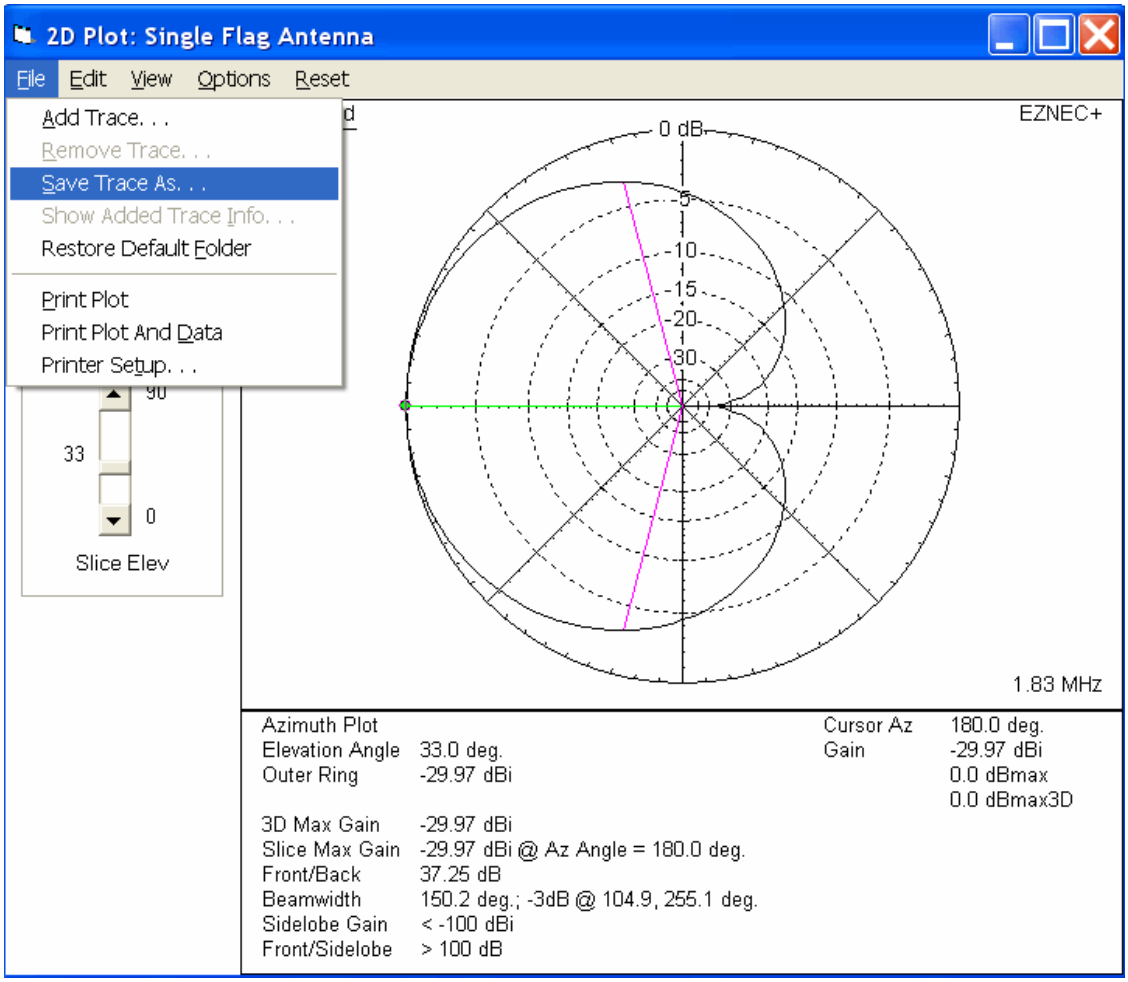


Figure 64 – Saving 160 m Azimuth Trace Data

I created a file named **160mAZ.PF** for the 160 meter data. I changed the frequency to 3.6 MHz, ran the model, and then saved the azimuth data to a file named **80mAZ.PF**.

The next step is to change the frequency to 7.15 MHz, and again run the model. This time, while viewing the azimuth pattern, I used the same menu as in the last example, but selected *Add Trace…*, which allowed me to load the 160 and 80 meter data into the same plot. The comparison is:

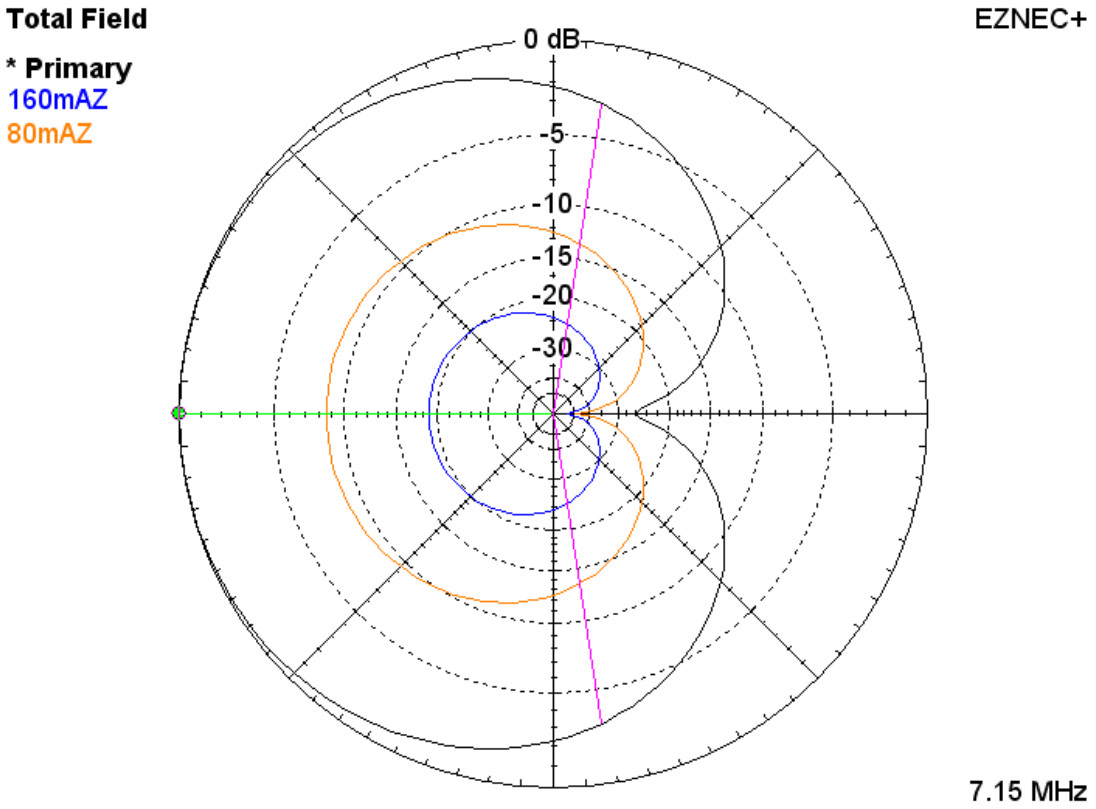


Figure 65 – Single Flag 160, 80, and 40 m Azimuth Pattern Comparison

This comparison shows that as you go from 160 to 80 meters you pick up about 10 dB of gain, and you pick up about 10 dB of additional gain going to 40 meters. The pattern shape, however, is very similar across all three bands. I should point out that I saved the patterns at the elevation angles of maximum gain, which moved from 33 degrees on 160 meters to 39 degrees on 40 meters. This may or may not be appropriate for the desired analysis – that's up to you as the designer. Since the main lobe is not very narrow in the elevation plane, it's a distinction without a difference.

Flag Antenna Array

To improve the gain of the antenna system, and to narrow the front lobe, I decided to explore creating an array of multiple Flag antennas. This example will look at 2 Flags operated in phase as a broadside array.

The only variable is the spacing between the flags. Values between ¼ and 5/8 wavelength should provide useful results, although since we have a model we can try anything we want and evaluate the results.

The first step is to create the array. Although we could type in another set of 4 wires and add another resistor Load and transformer, we can simplify the process by copying the first 4 Wires. That single step will do a lot of the work for us.

To copy the Wires, use the *Wire->Copy Wires…* menu command from the Wires window. That command displays the following dialog:

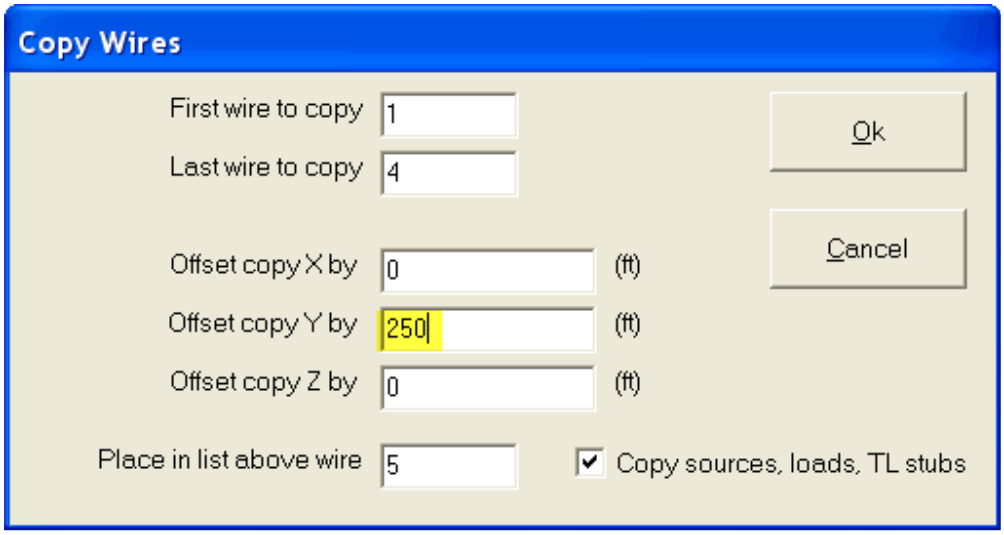
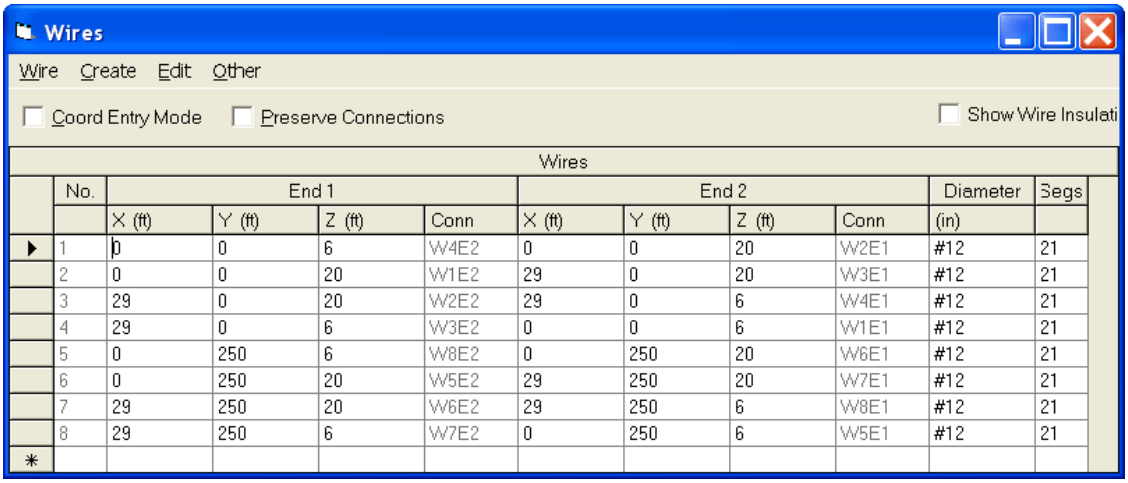


Figure 66 – Copy Wires to Create an Array

All of the existing Wires (1 through 4) are specified as the copy set by default. Since I wanted to copy all of the Wires, I did not need to make a change. What I did need to do was to specify the offset along the Y-Axis that I wanted between the original Wires and the copy. For that, I picked 250 feet, which is highlighted in yellow. I picked that number because it's very close to ½ wavelength on 160 meters which is typical broadside spacing.

The control in the lower right corner, **Copy sources, loads, TL stubs**, is checked by default. We want to use this feature since it will create a copy of the Load resistor. Unfortunately transformers are not part of that list. So, after the new Wires are created, it will be necessary to create a second transformer and a second Source, and connect them to the second Flag antenna.

The Wire list after the copy is:



Wires

Wire Create Edit Other

Coord Entry Mode Preserve Connections Show Wire Insulati

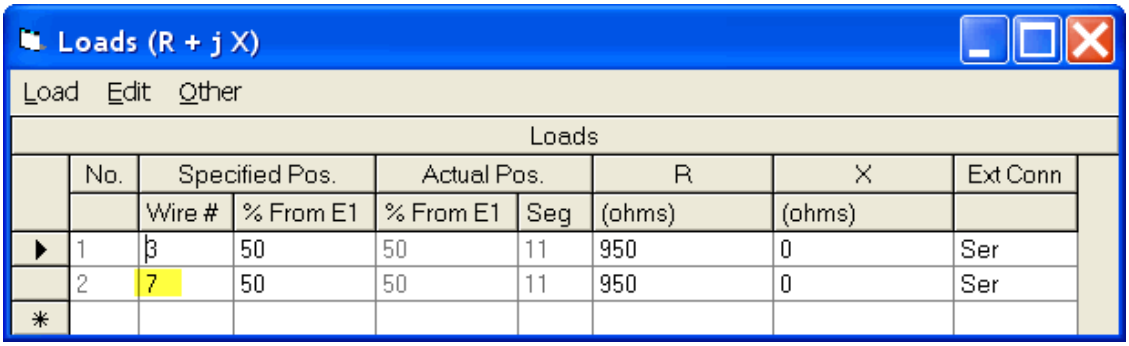
Wires

	No.	End 1				End 2				Diameter	Segs
		X (ft)	Y (ft)	Z (ft)	Conn	X (ft)	Y (ft)	Z (ft)	Conn	(in)	
▶	1	0	0	6	W4E2	0	0	20	W2E1	#12	21
	2	0	0	20	W1E2	29	0	20	W3E1	#12	21
	3	29	0	20	W2E2	29	0	6	W4E1	#12	21
	4	29	0	6	W3E2	0	0	6	W1E1	#12	21
	5	0	250	6	W8E2	0	250	20	W6E1	#12	21
	6	0	250	20	W5E2	29	250	20	W7E1	#12	21
	7	29	250	20	W6E2	29	250	6	W8E1	#12	21
	8	29	250	6	W7E2	0	250	6	W5E1	#12	21
*											

Figure 67 – Wire List for Two Flag Antennas

Wires 5 through 8 are a copy of 1 through 4, except that they are 250 feet down the Y-Axis.

The copy automatically created a second Load resistor.



Loads (R + j X)

Load Edit Other

Loads

	No.	Specified Pos.		Actual Pos.		R	X	Ext Conn
		Wire #	% From E1	% From E1	Seg	(ohms)	(ohms)	
▶	1	3	50	50	11	950	0	Ser
	2	7	50	50	11	950	0	Ser
*								

Figure 68 – Automatically Copied Resistor

The second 950 Ohm resistor was placed on Wire 7, which is the corresponding Wire to Wire 3 in the original array.

A second Transformer was manually added to the Transformer list.

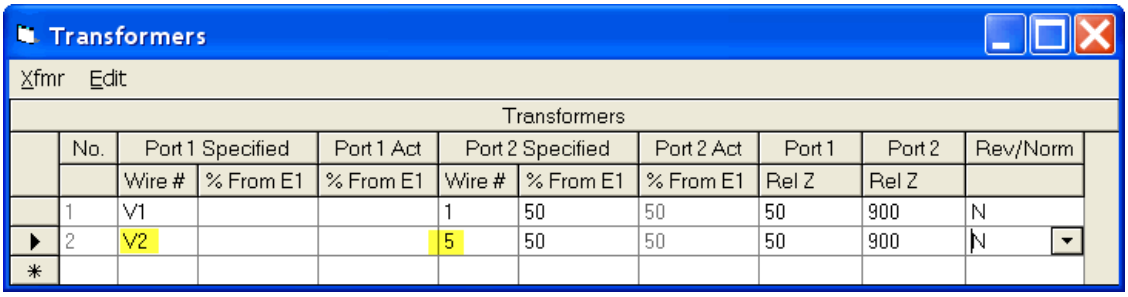


Figure 69 – Manually Added Transformer

The second transformer was connected to a second Virtual Wire, V2, and Wire 5, which is the appropriate Wire for the second Flag. I added a second Source that connects to the second Flag through V2.

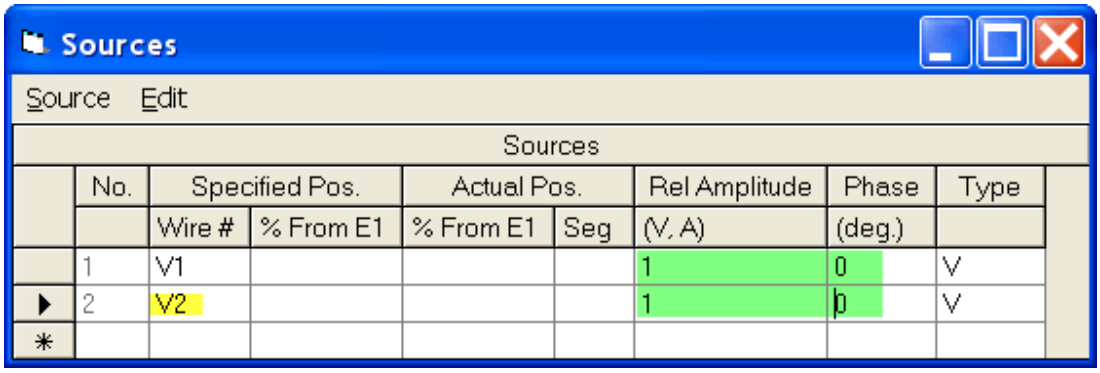


Figure 70 – Array Sources

Since the idea was to drive the two Flags with the same signal level and in phase, it's important that the amplitude and phase values match on the two sources. The absolute values don't matter, but it does matter that they are the same for both Sources. Practically, this is driving the pair in a *broadside* configuration.

The View Antenna window shows us what we have created:

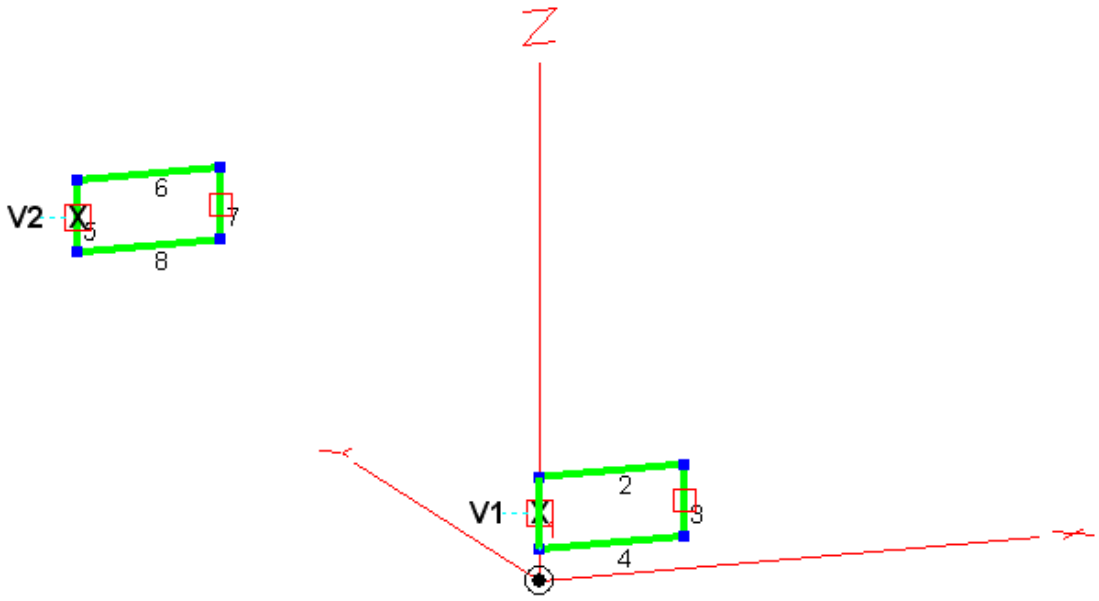


Figure 71 – Array View

This looks like what we wanted – two Flags pointing in the same direction with 250 foot spacing between them.

All that's left is to collect some results. Running the model at 1.83 MHz produces the following azimuth pattern plot.

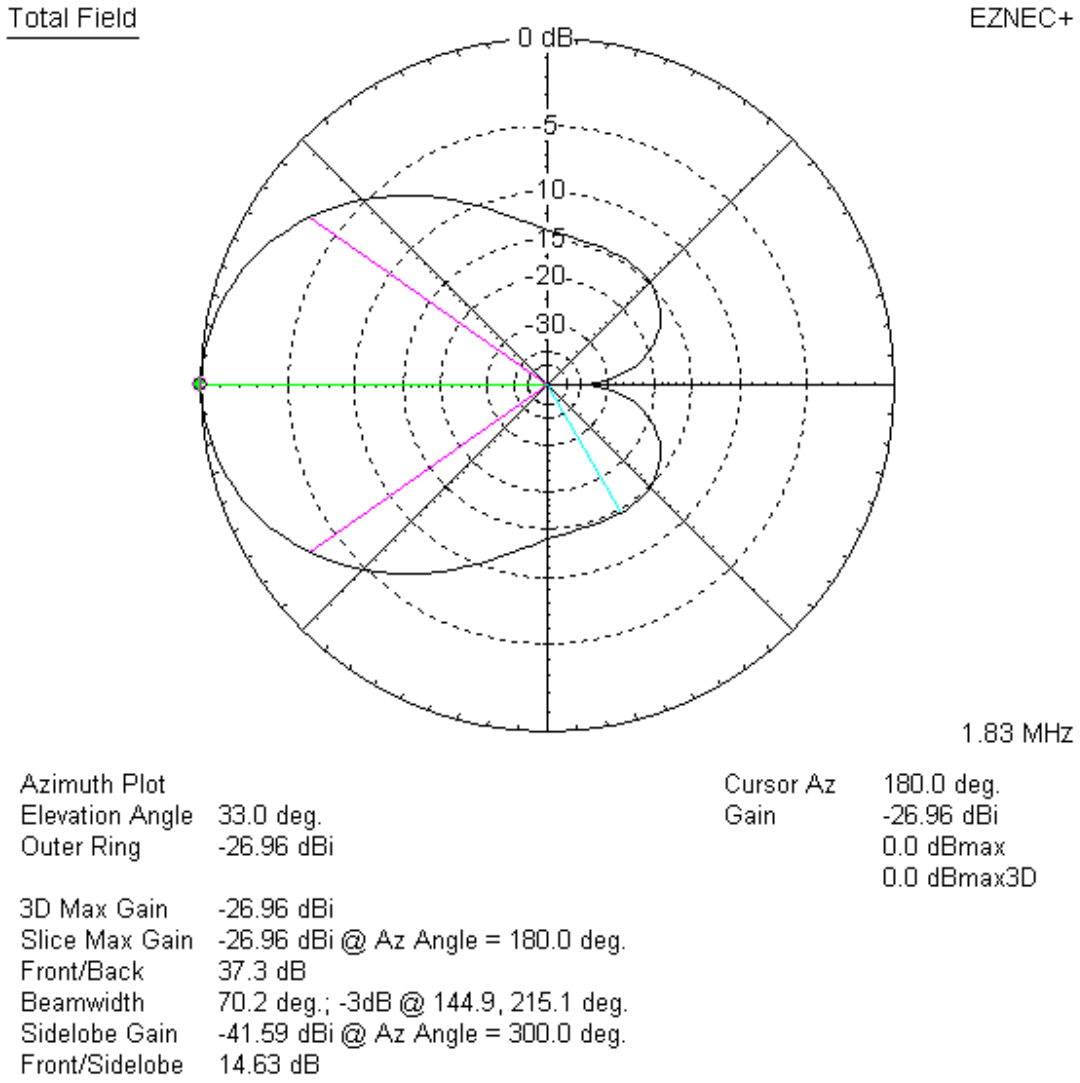
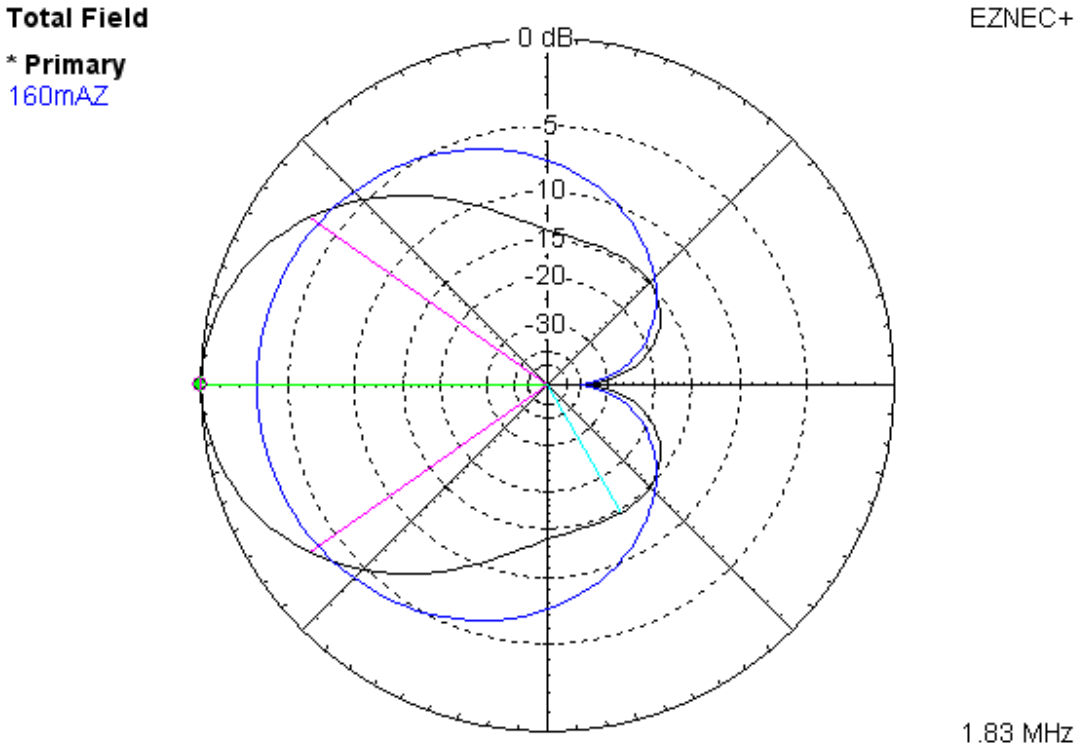


Figure 72 – Array Azimuth Response

This appears to be much narrower than the single Flag response, with some additional gain. Of course we don't have to crosscheck manually, we can add the original 160 meter trace to this plot. That produces:



Azimuth Plot		Cursor Az	180.0 deg.
Elevation Angle	33.0 deg.	Gain	-26.96 dBi
Outer Ring	-26.96 dBi		0.0 dBmax
			0.0 dBmax3D
3D Max Gain	-26.96 dBi		
Slice Max Gain	-26.96 dBi @ Az Angle = 180.0 deg.		
Front/Back	37.3 dB		
Beamwidth	70.2 deg.; -3dB @ 144.9, 215.1 deg.		
Sidelobe Gain	-41.59 dBi @ Az Angle = 300.0 deg.		
Front/Sidelobe	14.63 dB		

By clicking on the data set name in the upper left corner, either **Primary** or **160mAZ**, the data values change to the selected set. The set with the asterisk (*) is the selected set. This makes it easy to go back and forth between sets. After doing that comparison, we see that the front lobe has narrowed from a 150 degree 3 dB beamwidth to a 70 degree beamwidth. At the same time, the forward gain has increased by 3 dB, from -29.97 dBi to -26.96 dBi.

Of concern is the impact on other bands. On 80 meters, the spacing is now nearly 1 wavelength, which is wider than conventional wisdom suggests. The azimuth pattern on 3.6 MHz is:

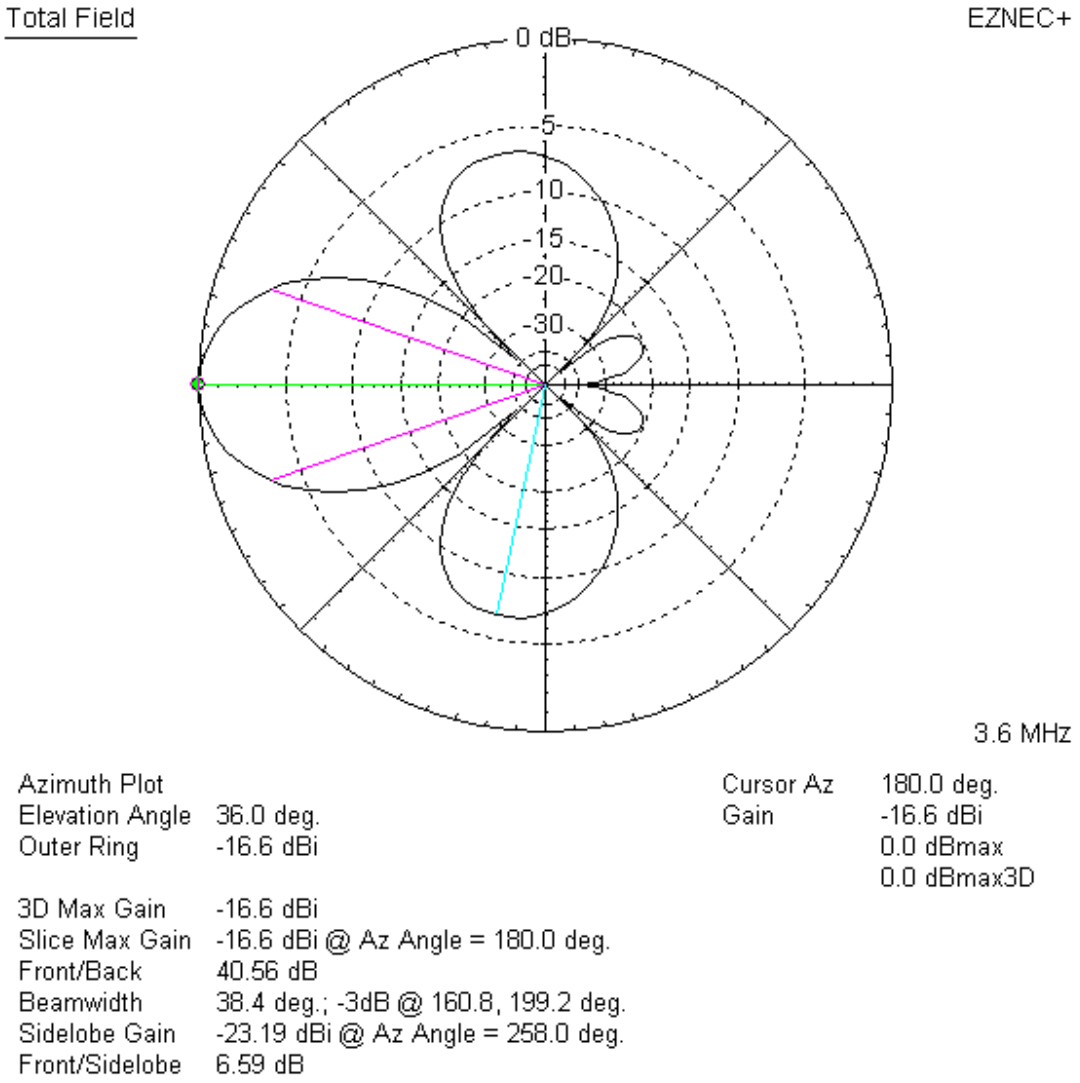


Figure 73 – Array Azimuth Pattern on 3.6 MHz

The wide spacing has created the *ears* on the pattern. This may or may not be a problem, it's all subjective in the eyes of the array designer. If we wanted to shrink the ears, and also not take up so much real estate, one approach would be to pick a spacing that would be as wide as practical for 80 meters, and then accept the results on 160 meters. Broadside spacing without creating large ears stops at about 5/8 wavelength. On 80 meters, that would be approximately 170 feet.

To evaluate that spacing, we need to change the model. One approach would be to manually edit all of the 250 foot Y-Axis values in the Wire table to 170 feet. Although that works, it seems tedious, and we can use a Wire table *Move* command to move Wires 5 through 8 with a single operation. The Move Wire dialog for this operation is:

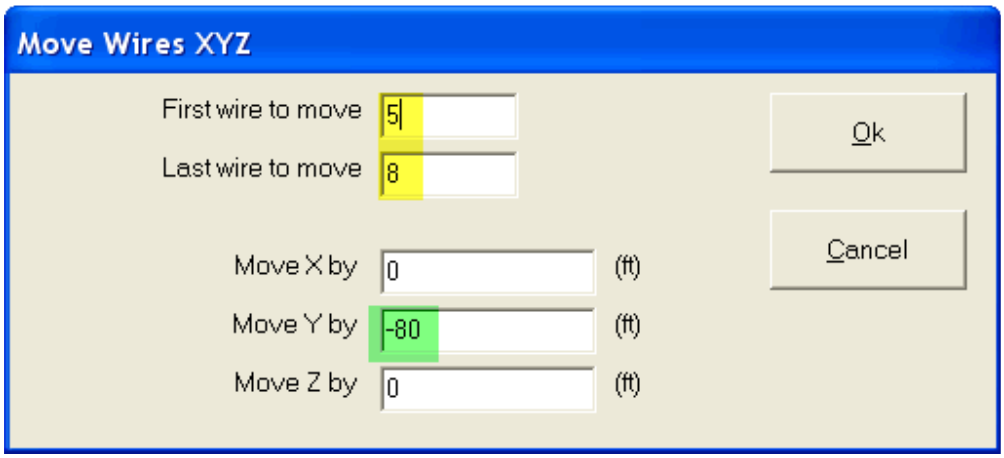


Figure 74 – Move the Second Flag Closer

The Wires we more are 5 through 8 – the Wires in the second Flag. The distance we want to move them is -80 feet down the Y-Axis. That's the difference between 250 feet and 170 feet.

After moving the second Flag, the 80 meter azimuth response is:

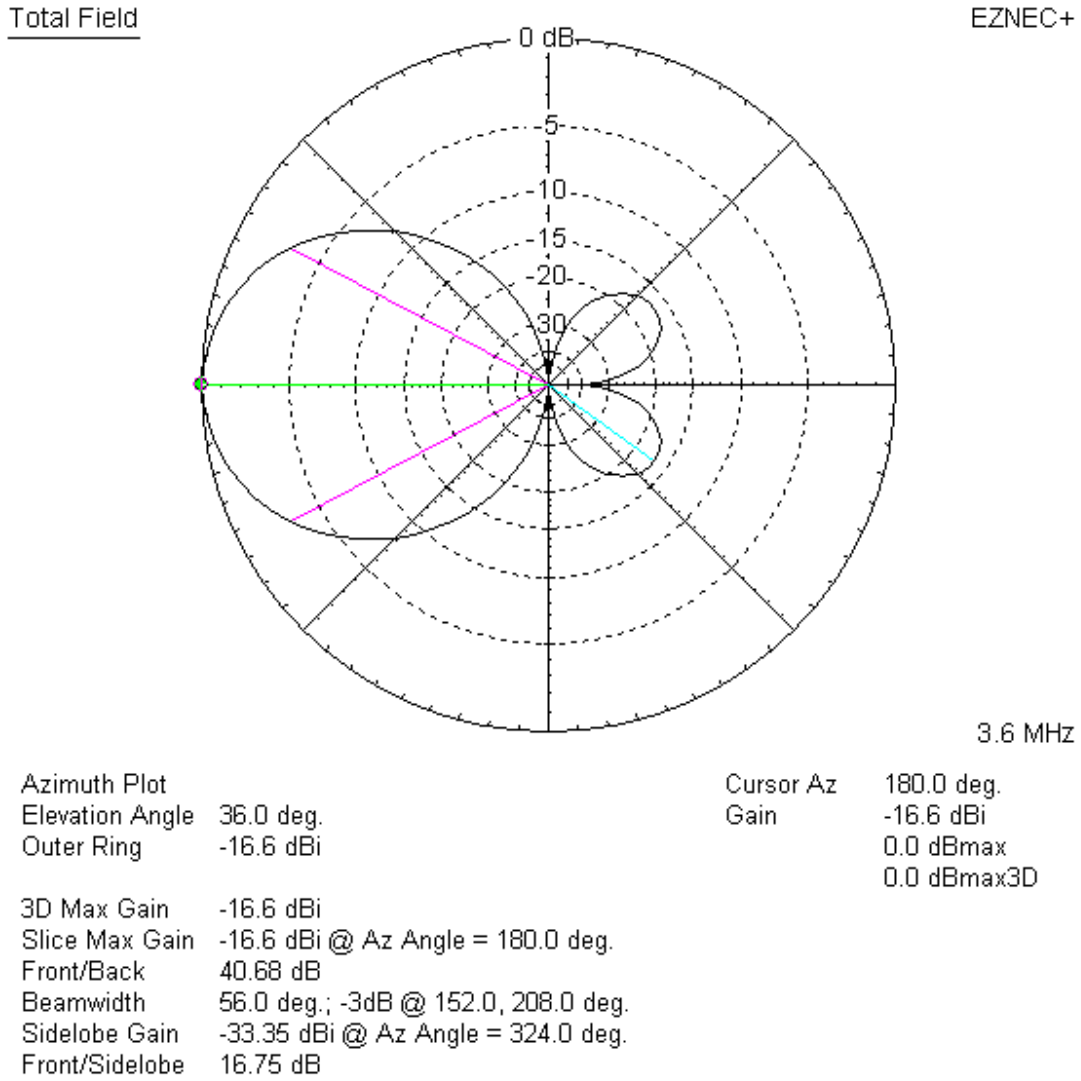


Figure 75 – 80 Meter Array Pattern with 170 Foot Spacing

Moving the two Flags closer together certainly knocked down the *ears* on 80 meters. On 160 meters, with 170' spacing, the azimuth response is:

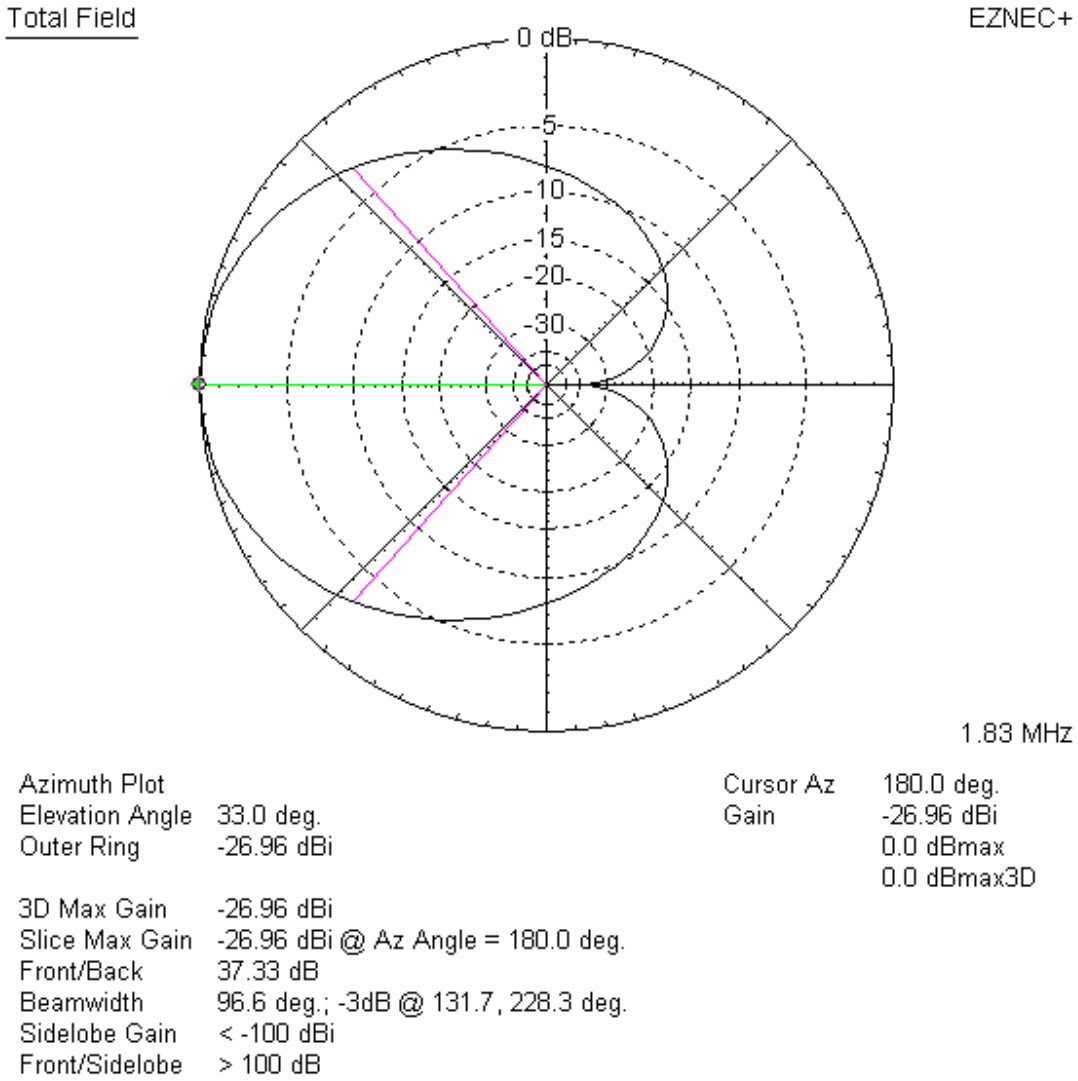


Figure 76 - 160 Meter Array Pattern with 170 Foot Spacing

The beamwidth has expanded to 97 degrees, but the added gain has remained.

The aim of this section is not to design an array, but to show typical steps in modeling the array, and how to create permutations in EZNEC. In just a few minutes it's possible to evaluate a number of permutations, and look for performance that meets your needs and can fit into available space.

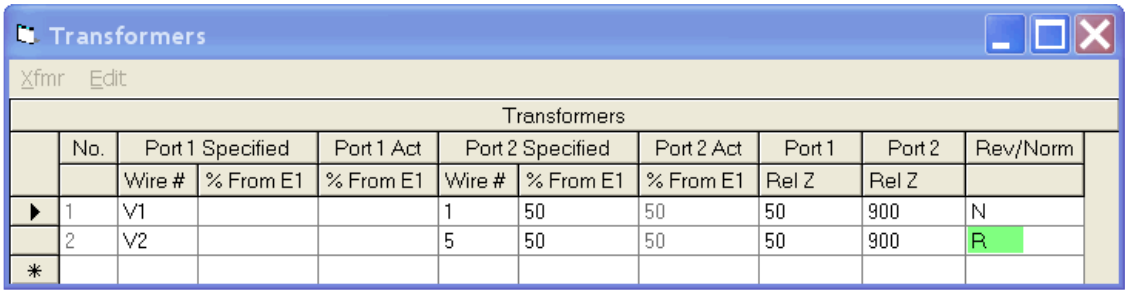
At that start of this section I mentioned how EZNEC helped me find my error in building this array. I did have this exact configuration in my yard a few years ago. The two antennas were connected in a switch box that allowed me to use either one Flag antenna or two. I knew from modeling that I should expect about 3 dB additional gain in the front direction when going from one to two Flags. After

connecting the second Flag, I ran inside to turn on the radio and check out some signals in Europe on 160 meters, which was the direction I was pointing towards. As I tuned in my first few European signals, however, it was clear that I had less gain when going from one Flag to two, not more gain.

What could I have done wrong?

After thinking about it for a few minutes, I realized that in my excitement to build the second Flag I did not pay attention to the phasing of the transformer. So long as you have a single Flag, it doesn't matter. What I could have done is add a 180 degree phase shift between the two antennas.

This is very easy to add to the model. If we go back to the Transformer window, we see:



	No.	Port 1 Specified		Port 1 Act	Port 2 Specified		Port 2 Act	Port 1	Port 2	Rev/Norm
		Wire #	% From E1	% From E1	Wire #	% From E1	% From E1	Rel Z	Rel Z	
▶	1	V1			1	50	50	50	900	N
	2	V2			5	50	50	50	900	R
*										

Figure 77 – Array Transformer with Phase Inversion

The right-most column, labeled **Rev/Norm**, determines if the connections to one side of the transformer should be reversed. This is exactly what I had a 50/50 chance of having done out in the yard. In this example I have set the value to be **R** for the second Flag to force an out of phase combination.

With this change, the array response on 1.83 MHz was:

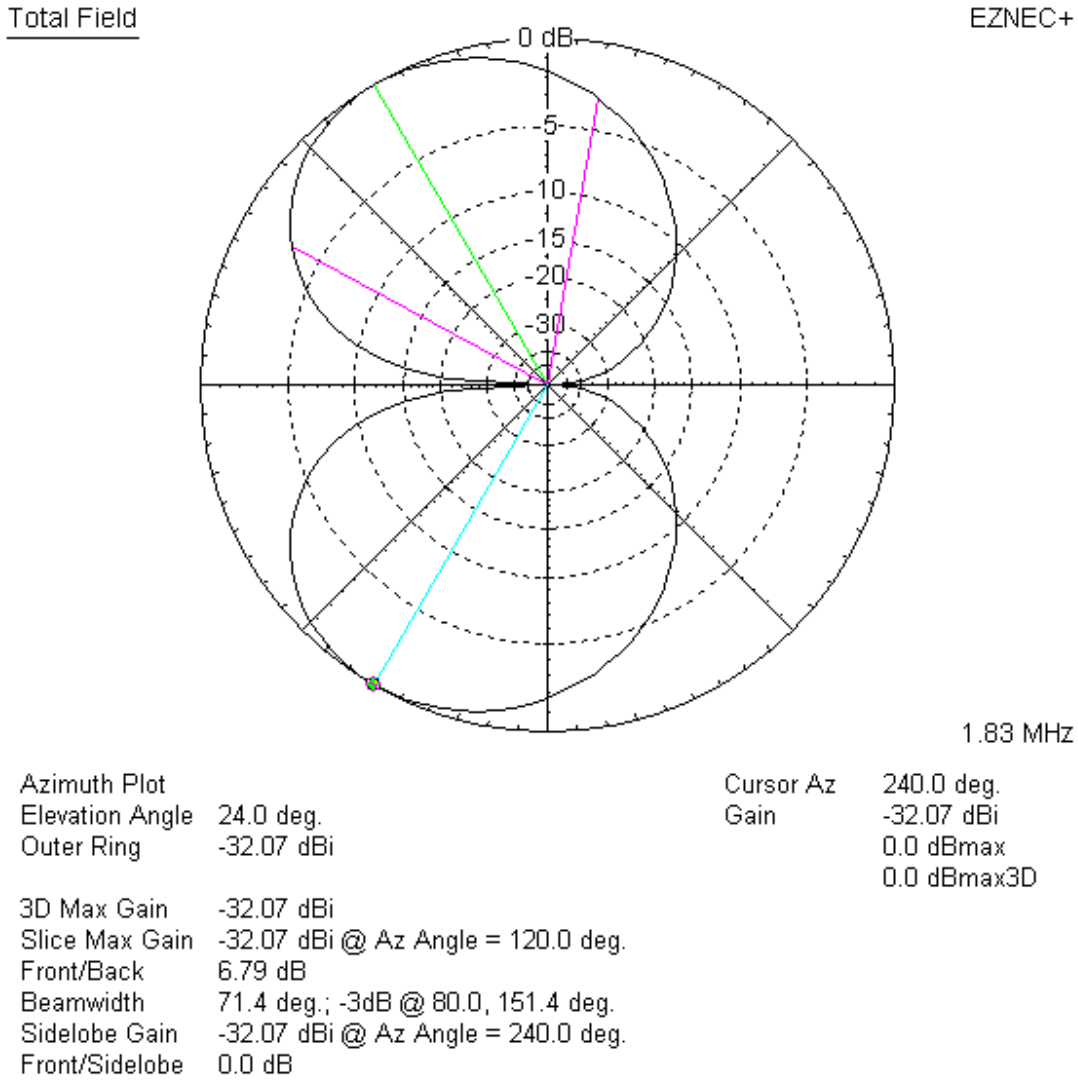


Figure 78 – Out of Phase Flag Array Azimuth Pattern

Bingo! The model showed me exactly what I was experiencing on the radio. I had put the pair out of phase. A few minutes with a soldering iron and I was enjoying 3 dB of additional gain over one Flag and a narrower main lobe.

Although we might like the results from a pair of Flags with 170 foot spacing, and decide to build it, our modeling is not really done. I say that because the model has two Sources. While that's not a problem for models, it is a problem for the real world, since we need to combine the Flags onto one cable.

The simplest way to finish up the design is to add two transmission lines that combine the two 50 Ohm feeds with a simple wire junction. That parallel combination will produce a 25 Ohm result. We can use another transformer, this

time with 25 and 50 Ohm ports, to get us back to 50 Ohms for the journey into the station.

Just to be sure we are all on the same page, here's a diagram of the enhanced model:

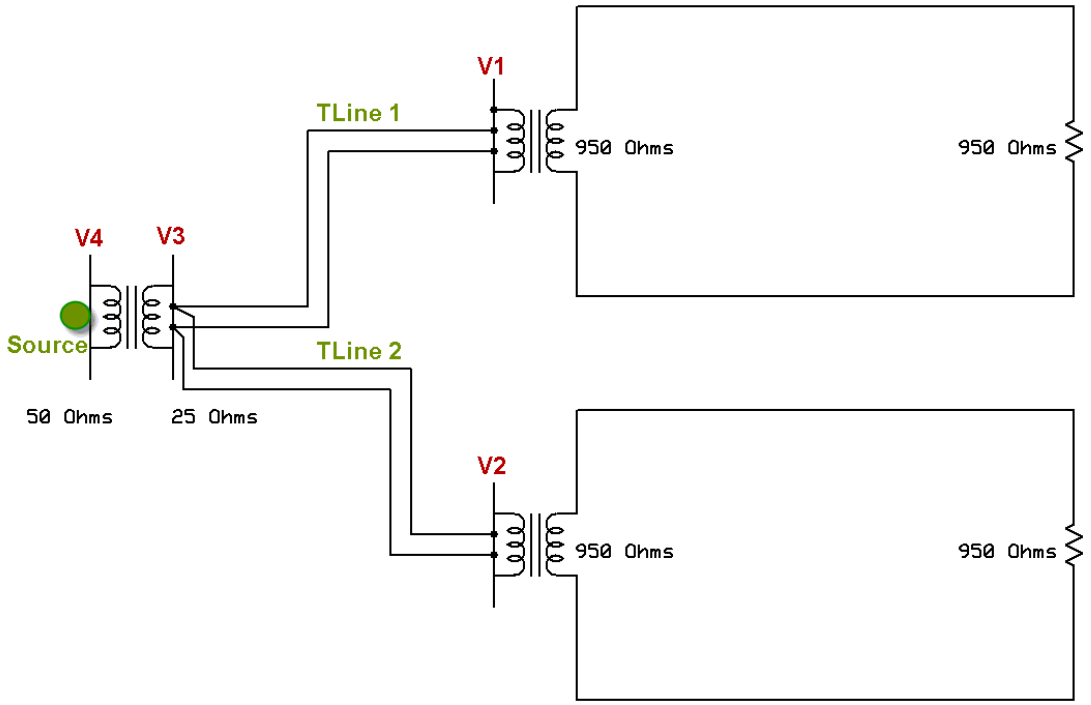


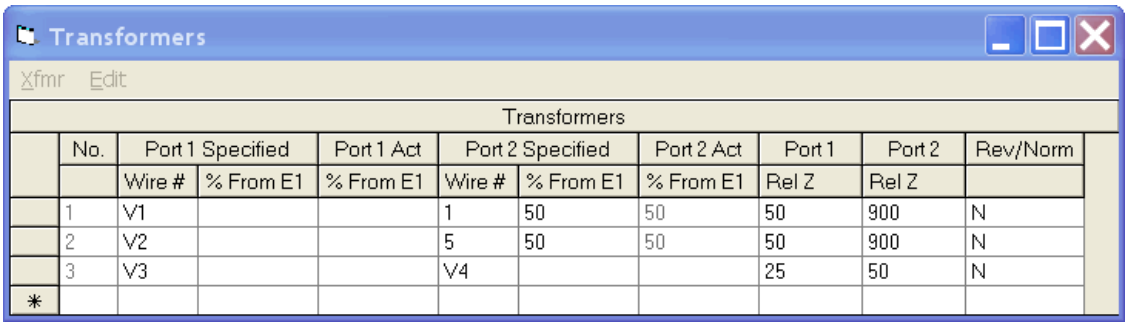
Figure 79 – Flag Array Diagram

Perhaps the most important details to understand from a modeling point of view are the four Virtual Wires, V1 through V4. These are tie points that create connections between the *insertion objects* that meet at the same segment on the Wire.

Until EZNEC added the Virtual Wire concept, each of these Wires would have been modeled as actual Wires in the model. The normal approach was to make them very short and put them very far away so that they don't take part in radiation and coupling. I suspect that's what Virtual Wires may be doing behind the curtain. Whether or not that's true, modeling with NEC requires those Wires, whether real or virtual.

We need to select a transmission line type and length for the final model. I picked a 0.66 VF 50 Ohm cable that is similar to RG-213. For a length, I selected ¼ wavelength on 1.83 MHz. That length is 88.7 feet, and two lines end to end will cover 177.4 feet. That seems like a useful length since the Flag antennas are 170 feet apart.

The Transformer list is:



Transformers

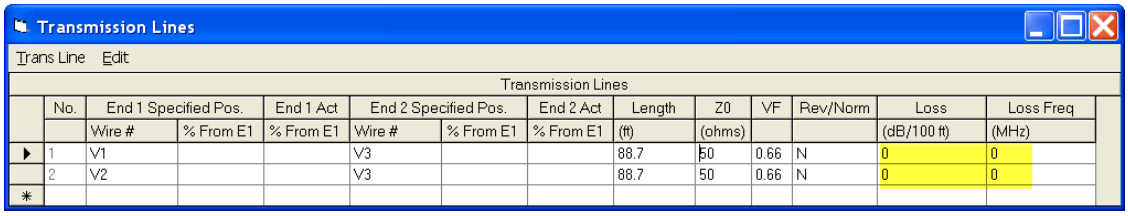
Xfmr Edit

Transformers

	No.	Port 1 Specified		Port 1 Act	Port 2 Specified		Port 2 Act	Port 1	Port 2	Rev/Norm
		Wire #	% From E1	% From E1	Wire #	% From E1	% From E1	Rel Z	Rel Z	
	1	V1			1	50	50	50	900	N
	2	V2			5	50	50	50	900	N
	3	V3			V4			25	50	N
*										

Figure 80 – Flag Array with Transmission Lines Transformer List

The transmission line list is:



Transmission Lines

Trans Line Edit

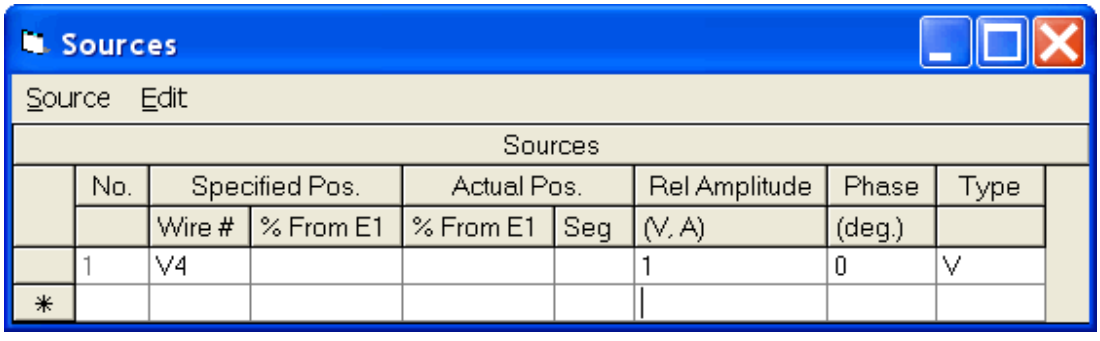
Transmission Lines

	No.	End 1 Specified Pos.		End 1 Act	End 2 Specified Pos.		End 2 Act	Length	Z0	VF	Rev/Norm	Loss	Loss Freq
		Wire #	% From E1	% From E1	Wire #	% From E1	% From E1	(ft)	(ohms)			(dB/100 ft)	(MHz)
▶	1	V1			V3			88.7	50	0.66	N	0	0
	2	V2			V3			88.7	50	0.66	N	0	0
*													

Figure 81 – Flag Array Transmission Lines

I highlighted the loss specification portion of the list. In this case, these cables will have no loss. If you want to model cable loss, then you need to obtain loss versus frequency data for the cable you are using. The ARRL books contain a loss table, and if you search on the Internet you will find many more. The loss is specified as dB per 100 feet of cable since my **Units** are feet. It's most accurate to pick a frequency that is close to the test frequency.

This model has a single source connected to V4.



Sources

Source Edit

Sources

	No.	Specified Pos.		Actual Pos.		Rel Amplitude	Phase	Type
		Wire #	% From E1	% From E1	Seg	(V, A)	(deg.)	
	1	V4				1	0	V
*								

Figure 82 – Flag Array with Transmission Lines Source

As promised, there is a single Source in this model, connected to the 50 Ohm side of the third transformer via V4.

We might start by asking if we get the same or similar results to the model with two Sources. Since all we added were two matched and lossless cables, there should be very little difference. Here is a comparison of the two Source and one Source models:

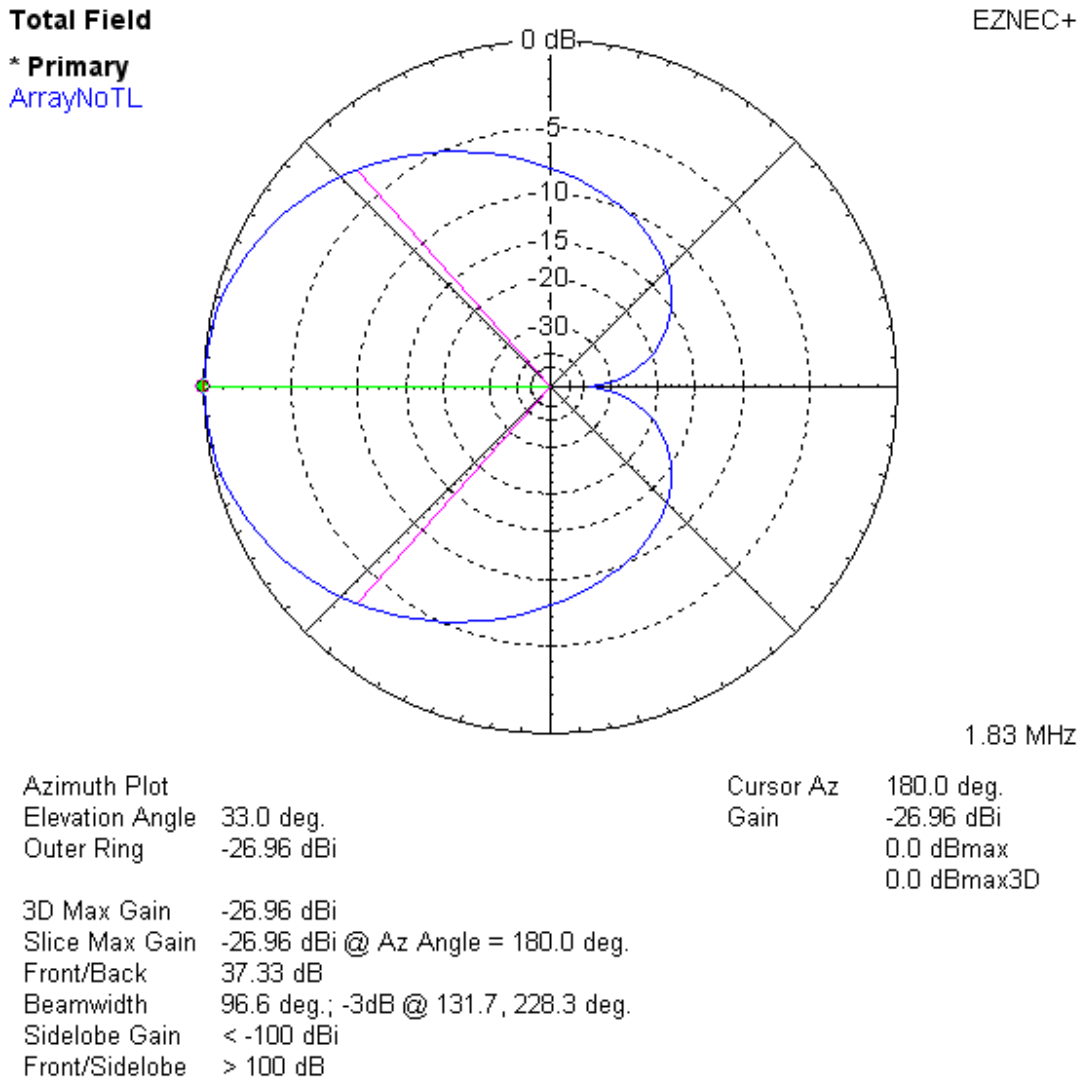


Figure 83 – Flag Array Pattern Comparison

The patterns are so close that the two traces overlap.

The SWR sweep from 1.8 through 7.3 MHz was virtually identical to the single Flag sweep.

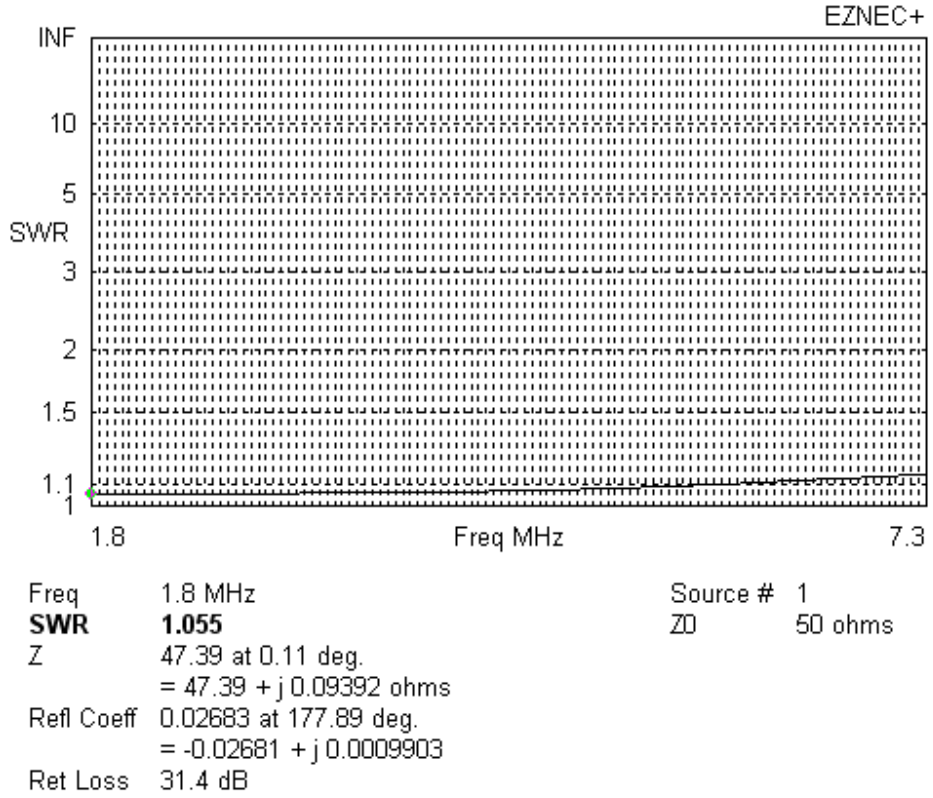


Figure 84 – Flag Array SWR Sweep

One data report that has not been examined is the Load Data report. The Flag antenna does include a Load, the 950 Ohm resistor. In the array, there are two resistors. The Load Data report for the previous model reveals:

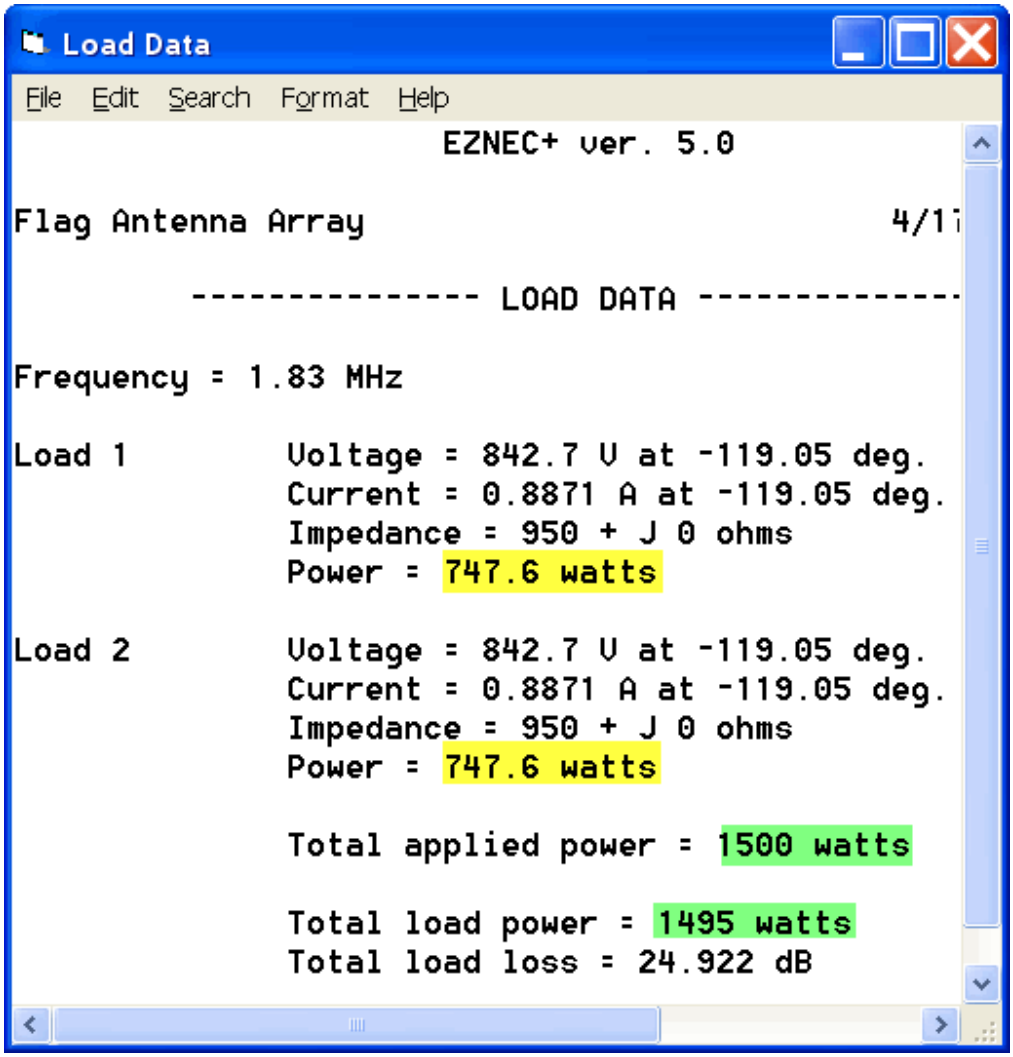


Figure 85 – Flag Array Load Resistors

This report explains why this type of antenna does not make a lot of sense as a transmitting antenna. Of the 1500 watts sent to the antenna, 1495 watts are consumed by the two resistors!

One last word on the subject of phasing. I made a mistake in wiring my transformer. I believe that there are four different ways to create the same phase inversion problem in the model. Three of these are rather easy to detect, but one is a bit more subtle.

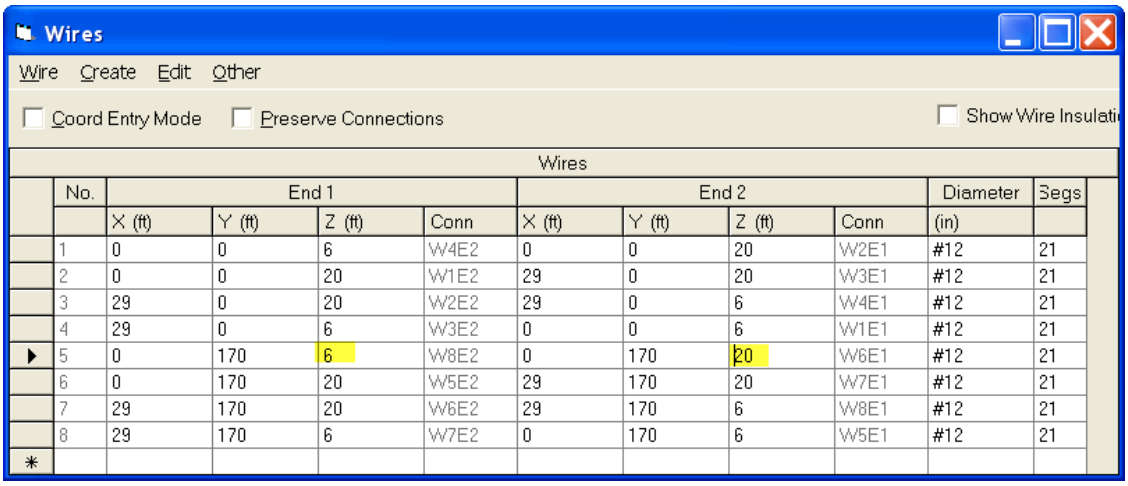
Let's list the three easy ways first.

1. Reverse one Flag transformer.
2. Reverse one Flag transmission line in the model that uses transmission lines.

3. Insert a 180 degree phase shift into one of the two Sources in the earlier model that uses two Sources and no transmission lines.

All of these stick out like a sore thumb in the model description, so it should be possible to see them and at least question *why did I do that?*

The last way to create an unwanted phase inversion is to reverse the Wire connected to the transformer! Here is the Wire list for the array.



Wires

	No.	End 1				End 2				Diameter	Segs
		X (ft)	Y (ft)	Z (ft)	Conn	X (ft)	Y (ft)	Z (ft)	Conn	(in)	
	1	0	0	6	W4E2	0	0	20	W2E1	#12	21
	2	0	0	20	W1E2	29	0	20	W3E1	#12	21
	3	29	0	20	W2E2	29	0	6	W4E1	#12	21
	4	29	0	6	W3E2	0	0	6	W1E1	#12	21
▶	5	0	170	6	W8E2	0	170	20	W6E1	#12	21
	6	0	170	20	W5E2	29	170	20	W7E1	#12	21
	7	29	170	20	W6E2	29	170	6	W8E1	#12	21
	8	29	170	6	W7E2	0	170	6	W5E1	#12	21
*											

Figure 86 – Flag Array Wire List

Wires 1 and 5 are the Wires that connect to the transformers. This was discussed at the start of the Flag section. Since Wire 1 came first, because it is part of the first Flag, let's declare that it is *correct*. Wire 5 is part of the second Flag. I have highlighted the Z values for the two ends of Wire 5; they are 6 and 20 feet. If I were to swap those two values, the model would show the phase inversion results. It would be just like going out to a working array, with a transformer connected to a wire, cutting the wire free, and reversing it. That's the same thing as reversing the transformer. Now if we did that to both Wires 1 and 5, we would still be in phase. The problem is when they have a different direction with respect to their individual Flags. Once this sort of reversal is buried inside the Wire table, it can be hard to find, since we have a sea of numbers to sort out.

The moral of the story is that it is important to have logical and consistent wiring conventions inside the model. This is especially true if the model performance is based upon particular phase relationships on Wires. This is always the case if there is more than one Source, but it can also happen if a single Source is split down multiple transmission lines or other junctions as it was in this case. Transformers and transmission lines also have a phase inversion option, but those are much easier to spot in their lists. With insertion objects, there is a specific orientation that the object takes with respect to the two ends.

RDF – Receiving Directivity Factor

The Average Gain will be used later in this document to help determine if model results are trustworthy. Average Gain can also be used to compute the *directivity*, which is the same as the *RDF* (Receiving Directivity Factor).

The term RDF was introduced by Tom, W8JI. The same computation has generally been called the *directivity* in texts. Section 2-8 of the second edition of the classic textbook *Antennas*, by John Kraus, W8JK (SK) defines directivity as:

The directivity is given by the ratio of the maximum radiation intensity to the average radiation intensity.

The term *ratio* usually implies division, but if we are working in logarithmic units such as dB, we use subtraction.

The RDF is the maximum gain minus the average gain, where both are expressed in dB. The higher the RDF, the more effective the antenna is for reception on the lower HF bands, where rejecting noise is far more important than gain. RDF or directivity is a measure of the advantage the main lobe direction has over all other directions averaged together.

As an example, let's compute the RDF of the Flag array modeled in a previous section.

The gain of the main lobe is simply the highest gain value in the 3D pattern. Here is the azimuth slice at the take off angle that includes the maximum gain for the Flag array.

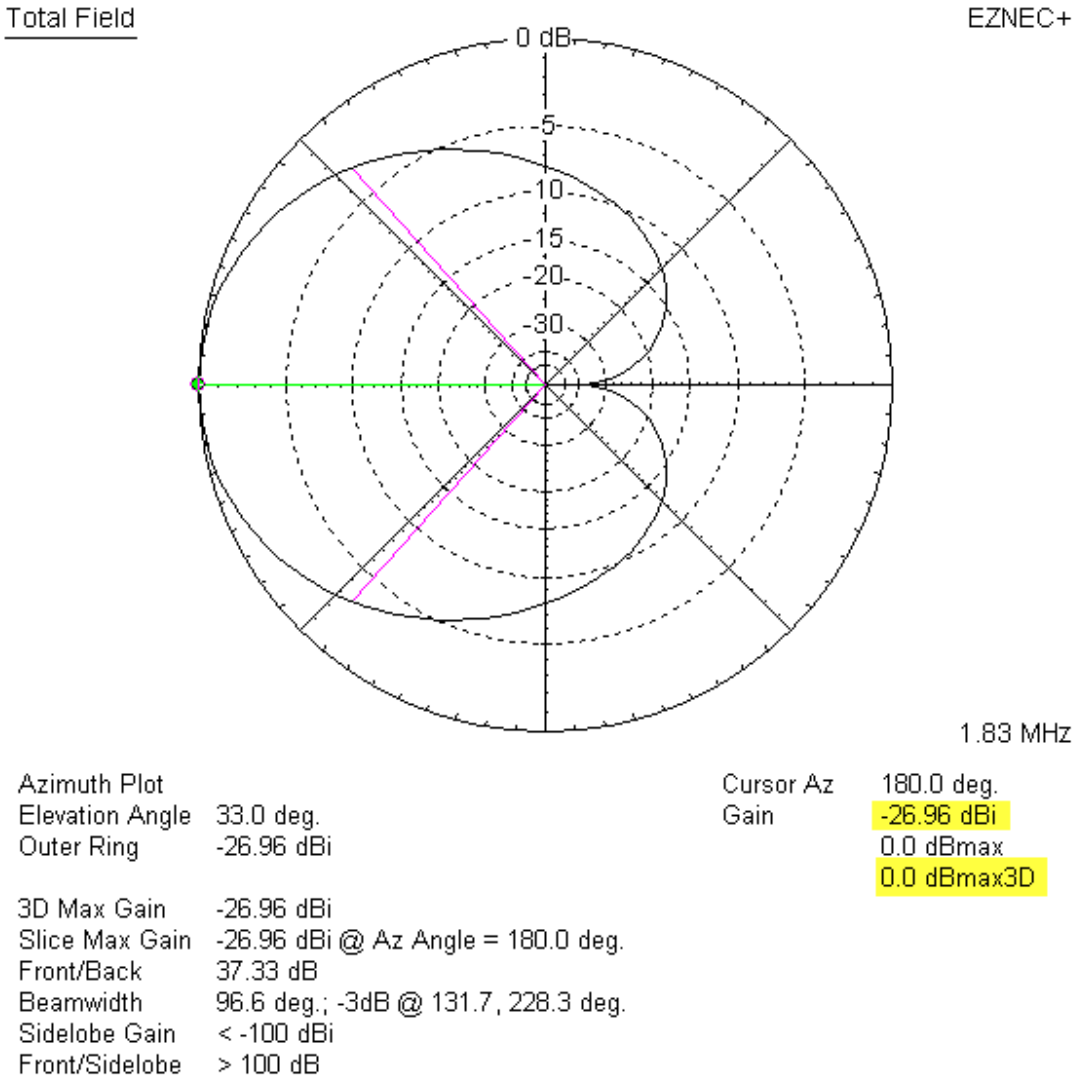


Figure 87 – Maximum Flag Array Gain

The maximum gain is -26.96 dBi. I also highlighted the difference between this value and the maximum 3D gain (dBmax3D). That value is 0, which means that we are truly at the maximum gain of the entire 3D pattern, **not** the maximum gain of this slice.

The Average Gain is available on the bottom of the main window. Note that the plot type **must** be 3D in order to compute and display the Average Gain.

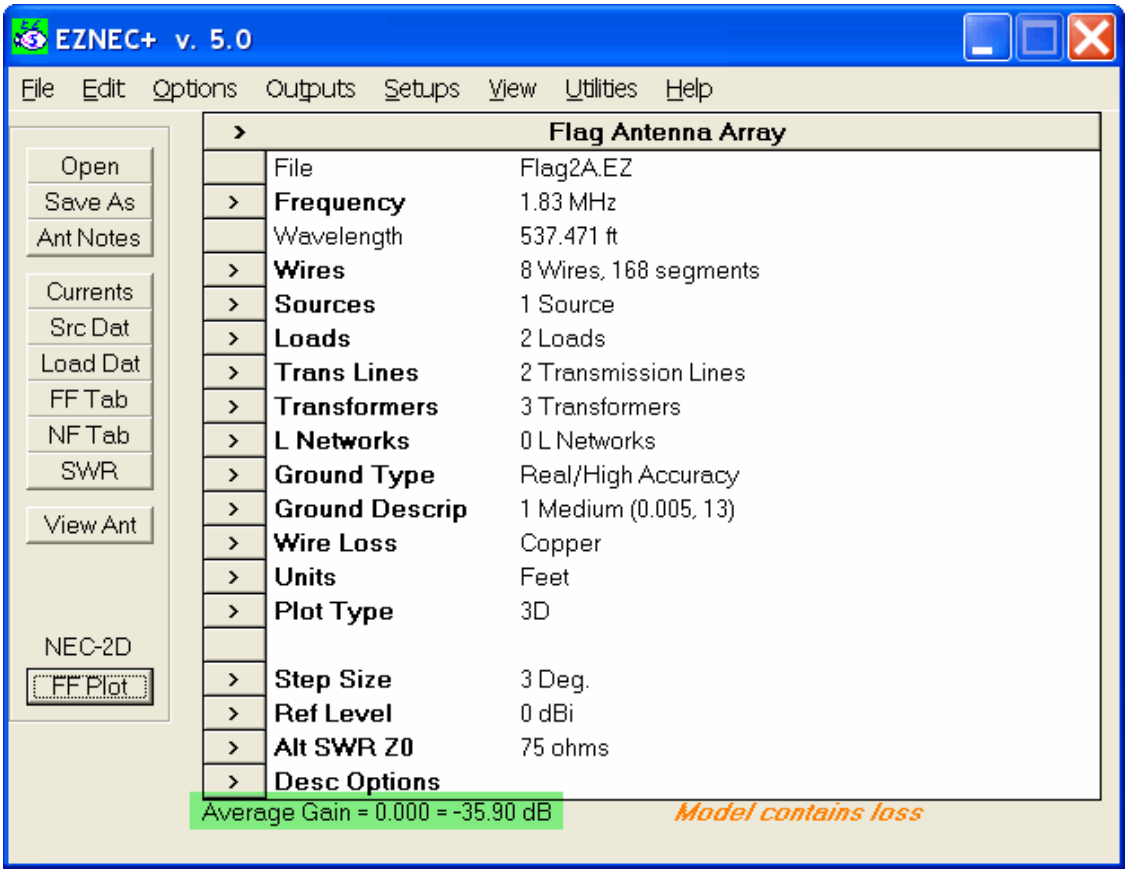


Figure 88 – Flag Array Average Gain

The Average Gain, in dB, is -35.90. We now have all we need to compute the RDF.

RDF = max_gain – average_gain = -26.96 – (-35.90) = 8.94 dB

When Average Gain is used for checking the model validity, it is essential to turn off all forms of loss. For the computation of RDF, we want the model to be in its normal operational state, including all loss.

Checks and Tests

A number of checks and tests can be used to increase confidence that model results can be trusted. The checks are run automatically, although they can also be executed on demand. The tests are always run manually.

If you suspect that your model is producing questionable results, you should take a minute or two to run all of the checks and tests, and make sure that you are

following all of the appropriate model guidelines. If you are finishing up a design project, and getting ready to start building the antenna, it's another good time to run all of the checks and tests.

These checks and tests are **not** a value judgment that you have a good antenna. They are a judgment about the operation of the NEC engine in response to the model. If checks or tests fail, the model results may not be trustworthy.

Segmentation Check

The segmentation check detects common violations of NEC guidelines. It is run automatically at various points in the program flow, but can also be run on demand.

As the name implies, the remedy for a segmentation check warning is to change the number of segments on, or adjust the diameter of, one or more Wires.

Geometry Check

Geometry checks are very important, and are performed as a first step before running a model. Geometry check errors are so important that the model will not run until the errors are fixed.

The Segmentation and Geometry Checks can be run on demands from the *Outputs* menu on the main window. I tend to run both at once using the *Outputs->Geom and Seg Check* menu command.

Model Convergence Test

The model convergence test is a more subtle form of segmentation check. Wires are divided into a specified number of segments. Conceptually, more segments results in more accurate model results. On the other hand, the time it takes to execute a model and produce results is approximately proportional to the square of the number of segments. Fewer segments implies faster results.

Within reason, the *correct* number of segments is the lowest number that produces results that are stable, and effectively converge to their final values. This guarantees the most accurate results as quickly as possible. Common parameters to check for convergence are maximum gain and Source impedance. If the antenna is highly directional, then checking parameters such as front to back ratio may make sense as well.

The Segmentation Check discussed in the previous section checks for segment numbers that are too low or too high according to the guidelines. In the convergence test, we are operating more in the middle. This is an area where modeling is more of an art than a science. With advances in computer performance, I tend to err on the high side, and I probably use more segments than necessary, but less then raise a Segmentation Check warning.

To perform a model convergence test, evaluate the target parameters for a range of segment values. This starts to become tedious as the number of Wires grows. If the antenna covers a broad range of frequencies, it is desirable to check across that dimension as well.

As an example, I took a resonant 80 meter dipole, 60 feet high, and ran it with a set of segment values, noting the maximum gain and Source impedance. The results were:

80 Meter Dipole Convergence Test			
Check Warning	**# Segments**	**Gain**	**Source Impedance**
Yes	5	6.26 dB	80.04 – j 0.78
Yes	7	6.29 dB	79.73 – j 0.06
Yes	9	6.30 dB	79.63 + j 0.28
No	11	6.31 dB	79.59 + j 0.47
No	15	6.32 dB	79.57 + j 0.68
No	21	6.32 dB	79.57 + j 0.84

For the values of 5, 7, and 9 segments, the Segmentation Check issued a warning that the segment length was too large. It appears as if the gain and impedance values are largely stable as the number of segments passes 15.

There exist Wire configurations and special cases that suggest using additional segments to improve accuracy. These are discussed in the EZNEC Help documentation.

EZNEC includes an *automatic segmentation* command. When executed, it will evaluate the segment count for every Wire in the model, and adjust the segment count according to the selected mode. There are two modes, *Conservative*, and *Minimum Recommended*. The Minimum Recommended mode attempts to reduce the segment count to the smallest value viewed by the program as reasonable. In the above example, the value selected by this mode was 5. The Conservative mode will produce a higher count, designed to lead to useful results. That mode selected 11 segments for this example, which is the first value that does not issue a warning.

Speaking for myself, I find the Conservative mode to be on the low side for my tastes. I would probably use 21 segments for this example. The EZNEC+ version I use allows up to 1500 segments, so, I've got a lot to play with and I am probably a bit sloppy in the conservation of my segments. If you are trying to fit a particular model into a smaller capacity version, you probably need to hone your segment management skills in order to get the most out of the software.

The automatic segmentation command is available on the Wire window menu, as *Wire->Auto Seg->Conservative* and *Wire->Auto Seg->Minimum Recommended*.

Average Gain Test

The Average Gain Test is based upon a simple concept. Assume a model with no loss. That means that the antenna is 100% efficient. Each watt of power fed to the antenna is radiated. If you were to mount power detectors completely around the antenna, and you added their results together, you should arrive at the same amount of power you fed into the antenna. Antennas are not amplifiers. They redistribute power, they don't create additional power. If there is no loss in the model, then the antenna should not be acting as an attenuator and losing power.

Said another way, every watt fed to the antenna is radiated in some direction, according to the pattern distribution. Watts are not being created nor lost. If we are creating or losing watts, then we should not trust the results. The problem usually arises from breaking one of the many modeling guidelines. As the saying goes, *Garbage In, Garbage Out*.

In the Average Gain Test, we temporarily turn off all lossy parts of the model. The Average Gain ratio should be 1.00, meaning that all watts have been accounted for. This is the same as 0.0 dB.

Folks who have studied this issue suggest that a valid range for the Average Gain is 0.95 through 1.05. Beyond that range, confidence in the results starts to drop.

The Average Gain value is only displayed for the 3D plot type, since a 3D plot is conceptually the same as putting power detectors completely around an antenna.

The hard part of performing the Average Gain Test is turning off all loss in the model. Loss can be located in many parts of a model, including:

1. Wire loss.
2. Load resistance.
3. Ground Loss.
4. Transmission line loss.

5. L Network resistance.

After performing the test, remember to return the model to its normal state. This test is not really testing the model, it's testing the NEC engines performance in running the model.

The first example to consider is a 20 meter 5 element Yagi supplied with the EZNEC package. The file name is **20m5elya.EZ**. The only modification I made was to raise the antenna off of the ground (Z = 0) so that it could be tested with a *perfect* ground model in addition to *free space*. Here is the antenna view of the model.

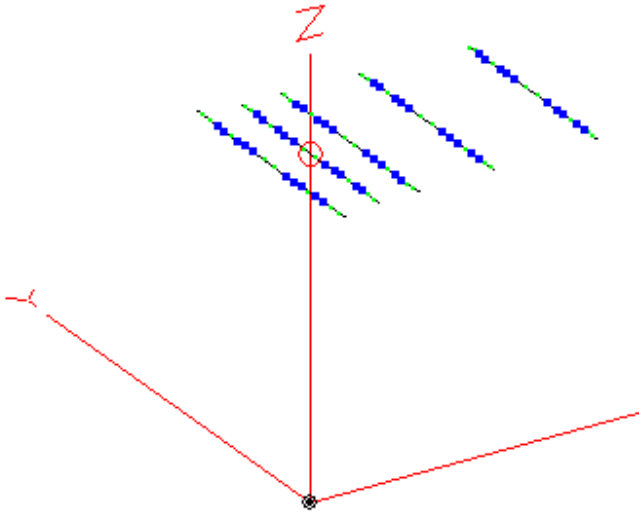


Figure 89 – 5 Element 20 meter Yagi

After running the model, the main window shows:

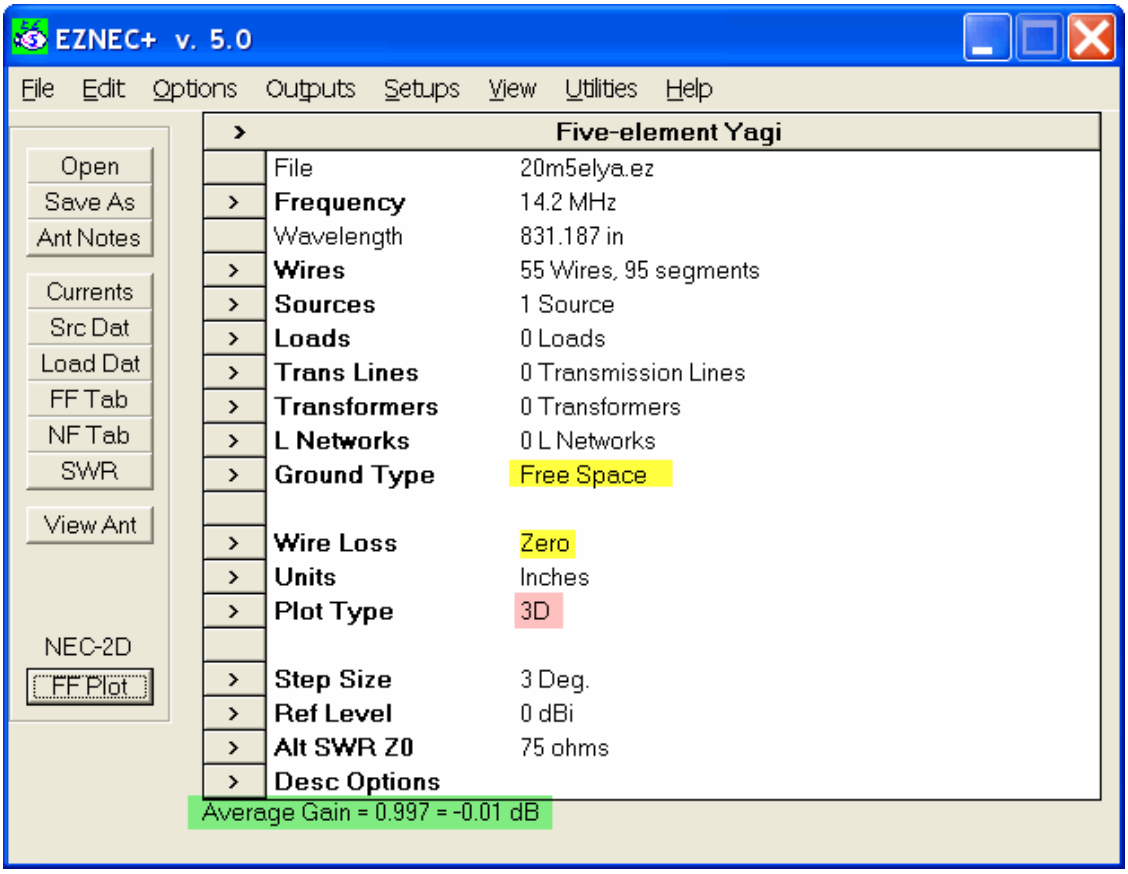


Figure 90 – Yagi in Free Space

Highlighted are the **Ground Type** and **Wire Loss** that indicate lossless selections. Also highlighted is the 3D **Plot Type** with a 3 degree **Step Size**.

The Average gain has been highlighted in green, and reports a value of 0.997, within the target range of 0.95 through 1.05. This model passes the Average Gain Test. If the **Ground Type** is changed to *Perfect*, the model continues to pass the test.

If I change the **Wire Loss** to *aluminum*, we have the following result:

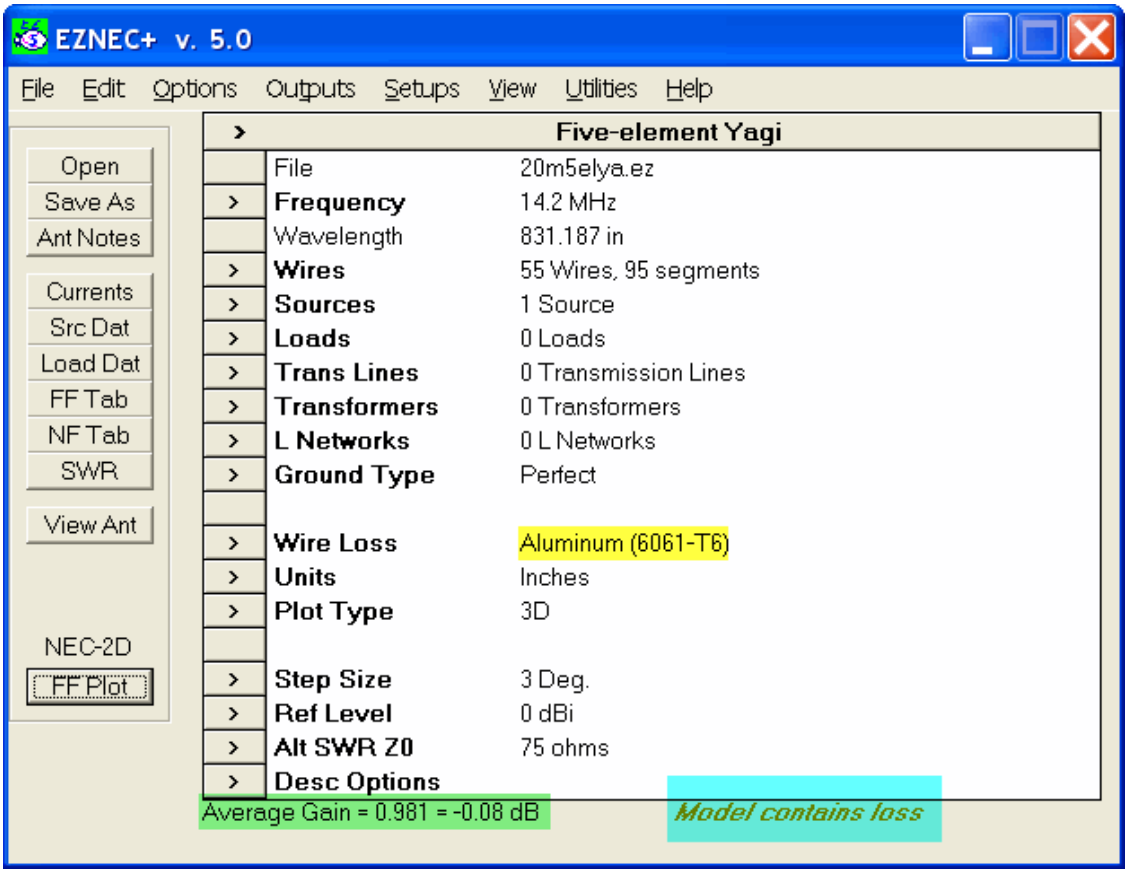


Figure 91 – Yagi with Loss due to Aluminum Wire

On one hand we are closer to the Average Gain Test lower limit since the value is 0.981. On the other hand, the program also reports that the Model contains loss, due to the aluminum Wires. The conclusion is that this is not a fair test, and we should change the **Wire Loss** to be zero.

Here is an example copied from the web site of Cecil Moore, W5DXP (www.w5dxp.com). This interesting model violates a number of guidelines, and produces wildly inaccurate results. The *antenna* looks like:

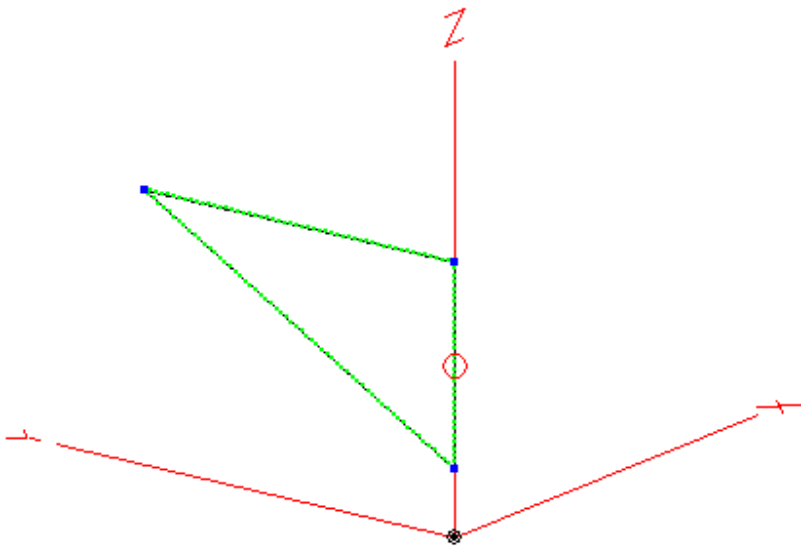


Figure 92 – SUPRGAIN.EZ

The antenna looks innocent, although that acute angle at the point is a source of concern. There is a fourth Wire that runs parallel and very close to one of the other three. That is also a modeling concern. The segmentation and geometry checks report no errors. What really stands out is the gain. Here is the azimuth pattern at the take off angle of maximum gain.

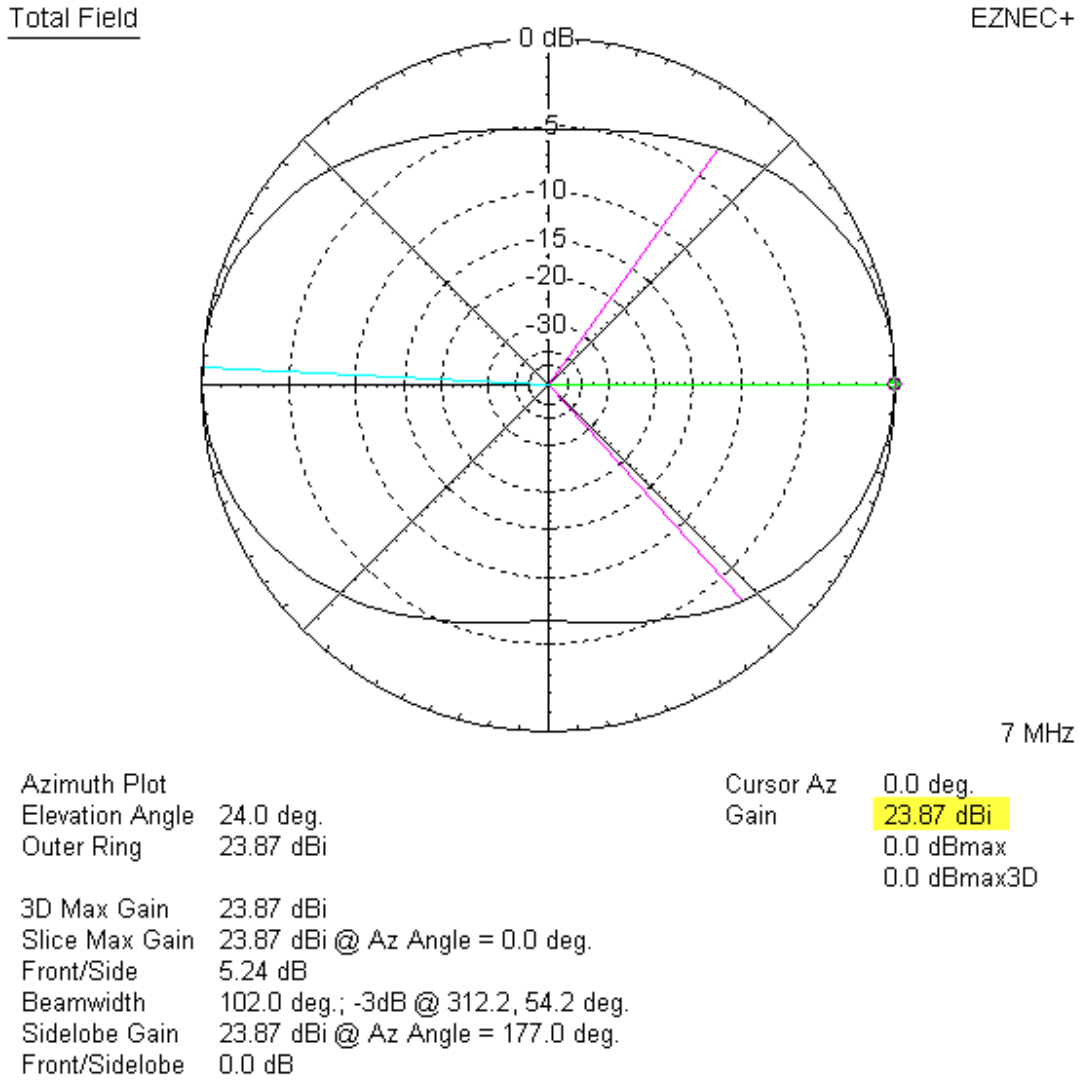


Figure 93 – SUPRGAIN.EZ Azimuth Pattern

The model reports a jaw dropping 23.87 dBi gain! This is a case were we should dash to perform the Average Gain Test. After removing the wire loss and switching to a perfect ground, the main window reports:

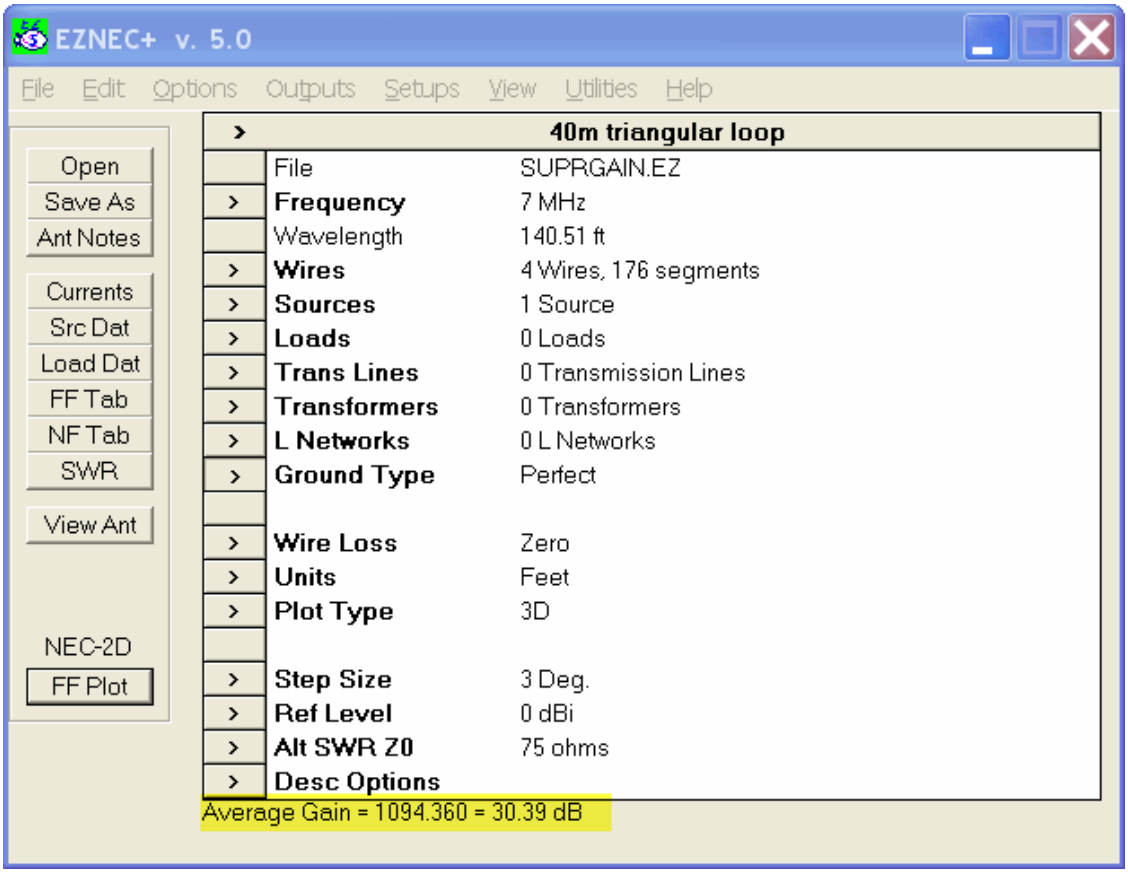


Figure 94 – SUPRGAIN.EZ Average Gain Test

The Average Gain is 1094.360, a far cry from the acceptable range of 0.95 to 1.05.

As the saying goes, *if it looks too good to be true, it probably is*. In this case, that is very true. The results from this model cannot be trusted.

If the Average Gain Test value is outside of the acceptable range, you have several choices. One is to examine the model and see if you are violating any of the guidelines. Correct guideline errors until the Average Gain returns to an acceptable value. If that is not possible, then the Average Gain value itself can be used to correct several antenna parameters. This process is described in the EZNEC Help documentation.

Conclusion

These examples illustrate one of the techniques that I use to keep myself out of modeling trouble. I try to evolve a model through series of steps where I can develop and check results all along the way, and stop at the first point where

there a problem. There is little sense in making a model more complicated if there is a concern.

Most antenna initial models can usually be created in just a few minutes. This assumes you have an idea of what you want to build, and you are transferring the design into the model.

Where modeling can take an unlimited amount of time is in the investigation of alternatives and variations. For me, I find that process enjoyable and illuminating as opposed to tedious and frustrating. Your mileage may vary. In writing this document I realized that over the last decade I've probably spent 10 times as many hours modeling antennas as being on the air.

This document will end up on the Internet at some point, and the models presented will be available for download. Please email me at ordy@seed-solutions.com to get the address.