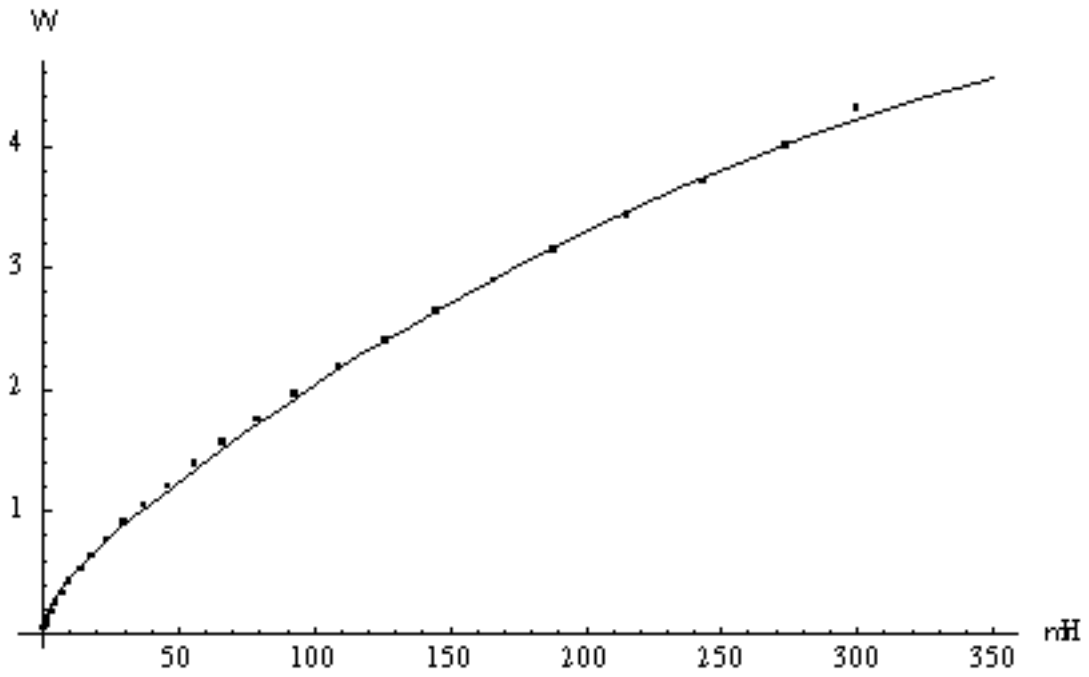
Using Polynomial Interpolation with 4nec2
by Duncan Cadd G0UTY

It sometimes occurs that in the course of an antenna simulation in 4nec2 it is necessary to include elements with electrical characteristics which must vary according to how the antenna is tuned or otherwise optimised. A very simple example is the use of a loading coil in a vertical HF quarter wave antenna in order to be able to shorten the monopole to a manageable size. The loading coil has an rf resistance which varies according to the total inductance of the coil and the working frequency, but although it is possible to have the 4nec2 optimiser adjust the coil inductance for resonance, the rf resistance cannot be adjusted automatically in due proportion.

In order to make the process more realistic, simple polynomial approximation can be employed so that 4nec2 produces automatic correction for the dependent variable, here the rf resistance of the coil. I will assume familiarity with 4nec2 and nec2.

First, I used ON4AA Rik Stroobandt's online inductance calculator[1] to obtain a table of inductance versus rf resistance for some arbitrary coil sizes on the 160m, 80m and 40m bands. I then used Mathematica to plot the rf resistance values against inductance and performed a curve fit to a simple polynomial.

For example, the 160m coil rf resistance vs. inductance graph is shown below:



The fits for each of the above bands were:

160m: $r = 0.0618595 + 0.0166301L - 0.0000156626L^2 + 0.10281LogL$

80m: $r = 0.0957032 + 0.0305569L - 0.0000753327L^2 + 0.0857773LogL$

40m: $r = 0.0537283 + 0.033388L \square 0.0000597368L^2 + 0.0577913LogL$

where r is the rf resistance in ohms, L is the coil inductance in microhenries and Log is the natural logarithm to base e. Maximum inductance values were 300μH on 160m, 112μH on 80m and 35μH on 40m.

It is now possible to create a 4nec2 model with one of these equations and use it to set the rf resistance of a coil automatically whilst the optimiser is adjusting the coil inductance for the best SWR. I have included some nec2 input files as examples. For the 160m monopole which is 5m tall with a simple capacity hat and base loading coil, the key lines are:

```
SY seg2Linp=150
SY seg2Lvar=abs(seg2Linp)+0.3
SY seg2L=abs(seg2Lvar/1E6)
SY rco1=0.0518595+0.01
SY rco2=0.0156301+0.001
SY rco3=(1.46626E-5)+(0.1E-5)
SY rco4=0.10181+0.001
SY seg2R=abs(rco1+(rco2*seg2Lvar)-
(rco3*seg2Lvar^2)+(rco4*Log(abs(seg2Lvar))))
```

and

```
LD    0       100    2       2       seg2R          seg2L
```

The first line enters a guess for the correct loading coil inductance, which will be applied to the second segment of the antenna, seg2Linp=150 i.e. the guess is 150μH. Unless the optimiser is constrained to give positive answers, all manner of problems can occur, so the next line uses the abs function to ensure the inductance is always positive and adds 0.3μH to set the minimum of the loading coil inductance range.

The next statement seg2L=abs(seg2Lvar/1E6) converts the 150 (μH) input value to henries.

The four following lines specify the four coefficients for the polynomial. Note that each has to be entered as a sum of two values so that the 4nec2 optimiser will not treat the four coefficients as variables. Note also that rco3=(1.46626E-5)+(0.1E-5) has to be written with both sets of brackets in place because of a bug in the 4nec2 parser which Arie Voors is aware of and may have fixed by now. Until the bug is fixed, entering rco3=1.46626E-5+0.1E-5 will generate an error.

The final assignment is the value of the polynomial approximation to the rf resistance of the coil, seg2R=abs(rco1+(rco2*seg2Lvar)-(rco3*seg2Lvar^2)+ (rco4*Log(abs(seg2Lvar)))).

Lastly, these two loading values are inserted into the appropriate location:

```
LD    0       100    2       2       seg2R          seg2L
```

This file, *160m loaded vertical 5m variload.nec* can now be run in the 4nec2 optimiser with the single variable seg2Linp selected. After a few iterations, the optimiser will have found a good match and will produce something like *160m loaded vertical 5m variload-opt.nec* which is also included with this document.

If the optimised version is examined it will be seen that the key line specifying the inductive load on segment 2:

SY seg2Linp=161.7633

has been updated with the correct value of tuning/loading inductance. Looking at the last nec2 input file to 4nec2, or alternatively the symbol conversion will also show what value was last calculated for the rf resistance of the coil, about 2.9 ohms.

This is the simplest application of polynomials to antenna modelling in 4nec2.

• • • • • •

A much more useful and correspondingly more complicated use of polynomials is to accelerate greatly the simulation of Yagi-Uda beam antennas.

The Yagi-Uda beam can be specified in terms of a table of element lengths and spacings, and this results in a list of N element lengths and N-1 element spacings. For an N element beam, the 4nec2 optimiser would therefore have to step through 2N-1 variables on each iteration cycle.

Except for the smallest arrays, this is unnecessary and wasteful of effort!

Let us assume that a simple polynomial will provide a close approximation to the "optimisation hypersurface" [/Dr. Evil impersonation], i.e. that the element lengths can be modelled effectively by one polynomial, and the element separations by another polynomial.

As an example I will consider a 70 element beam for the 23cm band. Normally, it would be impossible to get such a thing to run in a reasonable time, but on a ten year old computer with 1.6GHz processor, this will optimise in half an hour from scratch. Ordinarily, such an array would demand 70 element lengths and 69 element spacings, each to be specified as an independent variable. Not only does cycling through so many variables take forever, but there is a gargantuan multitude of local minima in which the program can be trapped. By using a simple polynomial expression for each of the element lengths and spacings, the total number of variables can be reduced from 139 to just nine and the propensity of getting stuck in a local minimum is greatly reduced because of the tendency of simple polynomials to follow gentle curves instead of sharp discontinuities.

These variables are coefficients in the following polynomials. For element lengths there are four coefficients (N is simply the element number in both polynomials.)

$$Len(N) = LenCo1 \square (LenCo2 \square N) + (LenCo3 \square N^2) \square (LenCo4 \square N^3)$$

and for spacings, five coefficients:

$$Sep[N] = Sep[N-1] + SepCo1 + (SepCo2 / N^2) + (SepCo3 / N) + (SepCo4 * N) + (SepCo5 * N^2)$$

The separation of element 1 is taken as zero by default, and each subsequent element spacing is cumulative upon the previous element spacing, Sep[N]=Sep[N-1] plus polynomial approximation involving SepCo1 to 5. I initialised these coefficients by taking a 2m antenna design, scaling it and adding more elements, and performed a simple curve fit in Mathematica. I should mention that I am only using Mathematica because I have an old version to hand, and there are probably free programs which can do the simple things required.

Using the *Yagi70_1296.nec* file, the following lines set up the nine coefficients:

```
SY SepV1=125012.8
SY SepV2=225716.4
SY SepV3=312172.9
SY SepV4=358423.1
SY SepV5=141009.7       'All separation variables done.
SY LenV1=541301
SY LenV2=732643
SY LenV3=161489
SY LenV4=134215         'All length variables done.
SY SepCo1=abs(SepV1/1000.0)
SY SepCo2=abs(SepV2/1000.0)
SY SepCo3=abs(SepV3/1000.0)
SY SepCo4=abs(SepV4/1000000.0)
SY SepCo5=abs(SepV5/10000000.0)   'All separation variables converted to
coefficients.
SY LenCo1=abs(LenV1/10000.0)
SY LenCo2=abs(LenV2/1000000.0)
SY LenCo3=abs(LenV3/10000000.0)
SY LenCo4=abs(LenV4/1000000000.0)       'All length variables converted to
coefficients.
```

Note that the number of symbols used by 4nec2 must be increased in order for the example file to run. I set this to 512 symbols. You will need to restart 4nec2 after doing this.

The next lines calculate the sixtynine element separations:

```
SY Sep1=0.0+abs(SepCo1+SepCo2-SepCo3+SepCo4-SepCo5)
SY Sep2=Sep1+abs(SepCo1+(SepCo2/4.0)-(SepCo3/2.0)+(SepCo4*2.0)-
(SepCo5*4.0))
SY Sep3=Sep2+abs(SepCo1+(SepCo2/9.0)-(SepCo3/3.0)+(SepCo4*3.0)-
(SepCo5*9.0))
        •       •       •
SY Sep67=Sep66+abs(SepCo1+(SepCo2/4489)-(SepCo3/67)+(SepCo4*67)-
(SepCo5*4489))
```

```
SY Sep68=Sep67+abs(SepCo1+(SepCo2/4624)-(SepCo3/68)+(SepCo4*68)-
(SepCo5*4624))
SY Sep69=Sep68+abs(SepCo1+(SepCo2/4761)-(SepCo3/69)+(SepCo4*69)-
(SepCo5*4761))
```

Having converted the separation coefficients to spacings, there are 69 lines converting millimetres to metres and then seventy lines which calculate the element lengths:

```
SY Len1=abs(LenCo1-LenCo2+LenCo3-LenCo4)
SY Len2=abs(LenCo1-(LenCo2*2)+(LenCo3*4)-(LenCo4*8))
SY Len3=abs(LenCo1-(LenCo2*3)+(LenCo3*9)-(LenCo4*27))
      •     •     •
SY Len68=abs(LenCo1-(LenCo2*68)+(LenCo3*4624)-(LenCo4*314432))
SY Len69=abs(LenCo1-(LenCo2*69)+(LenCo3*4761)-(LenCo4*328509))
SY Len70=abs(LenCo1-(LenCo2*70)+(LenCo3*4900)-(LenCo4*343000))
```

These are also converted from millimetres to metres, following which all the calculated values are specified in standard nec2 format with 15 segments per element:

```
GW   1     15   0        -Len1s   0   0        Len1s    0
0.00150
GW   2     15   Sep1s    -Len2s   0   Sep1s    Len2s    0
0.00150
GW   3     15   Sep2s    -Len3s   0   Sep2s    Len3s    0
0.00150
      •     •     •
GW   30    15   Sep67s   -Len68s  0   Sep67s   Len68s   0
0.00150
GW   30    15   Sep68s   -Len69s  0   Sep68s   Len69s   0
0.00150
GW   30    15   Sep69s   -Len70s  0   Sep69s   Len70s   0
0.00150
```

The file is then completed in standard nec2 format for element resistivity corresponding to aluminium and free space simulation at 1296MHz:

```
GE   0
LD   5   1    0   0   2.88E+07
LD   5   2    0   0   2.88E+07
LD   5   3    0   0   2.88E+07
LD   5   4    0   0   2.88E+07
LD   5   5    0   0   2.88E+07
LD   5   10   0   0   2.88E+07
LD   5   20   0   0   2.88E+07
LD   5   30   0   0   2.88E+07
GN   -1
EK
EX   0   2    8   0   1.00E+00   0   0   0
FR   0   0    0   0   1296.0     0
EN
```

The result of running this input file is Yagi70_1296-opt.nec and this was obtained in half an hour using the following optimiser settings:

SWR 30
Gain 100
F/B 10
F/R 10
R 10
X-in 100
Rad 5

Resolution 10 degrees.

The characteristic impedance was set as 20Ω.

There are other possible tweaks for such an input file, for example 4nec2 has an autosegmentation feature which can keep the number of segments proportional to the element length. Alternatively this can be implemented directly, by specifying that the driven element has e.g. fifteen segments and calculating each element's segment number from the element lengths given by the polynomials.

It is of course possible to comment out or delete lines from the included example and shift the frequency if required. If major changes are desired, I have found that reducing the array to half a dozen elements and changing the frequency 50MHz at a time usually leads pretty smoothly to the desired result. Elements can then be uncommented/restored to obtain the number of elements desired at the new working frequency.

The polynomials specified seem to do an adequate job, and though I have tried more complicated expressions, little seems to be gained. Logarithmic polynomials may be an obvious choice and they do work, but are very "picky" and can be difficult to get to convergence. I found that when using log polynomials I had to reduce the array to three elements when making drastic frequency changes. It feels very much like playing with a phase-locked loop, as there is a definite range over which the polynomial coefficients will "track" and outside of which "lock" cannot be obtained.

This article is the end result of four years' experimentation and the only array which has so far defied all attempts at fully automatic optimisation is the Landstorfer-Sacher array. This has been something of a bête noir for me and has occupied me on and off since the late 1990s. I did send my interim results to the late L.B. Cebik, W4RNL, who kindly put them on his website - but he forgot my name and callsign in the process! The L-S array is still a pain to model and an even bigger pain to optimise and unfortunately I think it unlikely that a satisfactory result can be obtained with the above methodology.

However, simple polynomials can greatly increase the range of simulations accessible and make more effective use of the amateur's time and computing resources.

1. ON4AA Rik Stroobandt's inductance calculator is available at http://hamwaves.com/antennas/inductance.html last accessed 4/10/13.