nikiml's Antenna pages - Nec optimization scripts

Nikolay Mladenov

Motivation

When I first started working with antenna models I used exclusively 4nec2. It is an excellent tool and I still use it a lot, but as I was progressing I felt a need for certain optimization features that were not available. The biggest of those were utilization of multiple processor cores, optimizing over multiple sweep ranges (like uhf and vhf-hi) simultaneously and better optimization algorithms.

Tutorial

I finally came up with a small tutorial.

Installation

The scripts require Python 2.6 or higher (Python 3.* on Windows) and are installed as a python package.

After you download and unpack the distribution archive, run the installation command:

```
python setup.py install
```

You will also need to download and copy your favorite nec engine somewhere in your path or in YOUR_PYTHON_INSTALL_DIR\Lib\site-packages\nec\engines. Fow Windows I prefer using NEC2/MP engines for their speed. For Linux the scripts so far only work with the **nec** package.

Evaluation - nec.eval

The python module nec.eval can evaluate multiple sweeps on multiple cores using NEC2 engine. For example the command line:

```
python -m nec.eval --num-cores=4 --uhf --vhf-hi input.nec
```

or the shorter equivalent:

```
python -m nec.eval -N4 -uV input.nec
```

evaluates the raw and net gains, the swr, the real and imaginary part of the impedance for all the frequencies of the VHF-hi and UHF ranges. The speed of evaluation is proportional to the number of processor cores. The output looks like that:

```
  Freq RawGain NetGain     SWR    Real    Imag
================================================
   174    9.58 8.50981 2.75474 338.384 -334.657
   180    9.07 8.91442 1.46181 340.638 -115.141
  ..............................................
   210    8.15 7.83107 1.72513 467.768  120.939
   216     8.1 7.57544 2.01799 576.488  111.21
   470   12.72   12.02 2.25638 140.741 -64.605
   476   12.73 12.2266 1.98879 155.801 -46.7397
  ..............................................
   692   13.26 12.7623 1.98086 186.918 -120.199
   698   13.17 12.5209 2.18773 157.551 -100.902
   704   12.97 12.1492 2.41865 134.034 -76.905
```

if --html=1 is used (which is the default) an html file is generated containing the above output as well as 3D model, Horizontal pattern, gain charts and links. The links can be used in any web page, blog or forum.

Options

The options can be supplied by either the command line or by adding CMD--EVAL line in the comment section of the NEC file

```
CM My VHF-hi/UHF antenna
CM ...
CMD--EVAL -Vu --num-cores=4 -a15
CM ....
CE
.....
```

if the nec file contains FR cards and no sweeps are specified as options then nec.eval will evaluate the frequencies from the FR cards.

Below is the summary of the usage for nec.eval:

```
c:>python.exe -m nec.eval -h
Usage: eval.py [options]

Options:
  -h, --help            show this help message and exit
  -o DIR, --output-dir=DIR
                        output path [output]
  -i NEC_FILE, --input=NEC_FILE
                        input nec file
  -s SWEEP, --sweep=SWEEP
                        adds a sweep range e.g. -s (174,6,8) for vhf-hi freqs
  -C IMPEDANCE, --char-impedance=IMPEDANCE
                        The default is 300.0 Ohms.
  -u, --uhf, --uhf-52   adds a uhf (ch. 14-51) sweep
  -U, --uhf-69          adds a uhf (ch. 14-69) sweep
  -V, --vhf-hi          adds a vhf-hi (ch. 7-13) sweep
  -v, --vhf-lo          adds a vhf-lo (ch. 1-6) sweep
  -n NUM_CORES, --num-cores=NUM_CORES
                        number of cores to be used, default=4
  -a NUM_SEGMENTS, --auto-segmentation=NUM_SEGMENTS
                        autosegmentation level - set to 0 to turn
                        autosegmentation off, default=10
  -e NEC_ENGINE, --engine=NEC_ENGINE
                        nec engine file name, default=nec2dxs1k5
  --engine-takes-cmd-args=ENGINE_TAKES_CMD_ARGS
                        the nec engine takes command args, default=auto (which
                        means no on windows yes otherwise). Other options are
                        'yes' or 'no'.
  -d MIN_WIRE_DISTANCE, --min-wire-distance=MIN_WIRE_DISTANCE
                        minimum surface-to-surface distance allowed between
                        non-connecting wires, default=0.005
  --validate-geometry=VALIDATE_GEOMETRY
                        set to 0 to disable geometry validation
  --debug=DEBUG         turn on some loging
  --forward-dir=FORWARD_DIR
                        the forward direction, by default is 0 which means the
                        antenna forward is along X.
  --backward-dir=BACKWARD_DIR
                        the backward direction (relative to --forward-dir) to
                        which F/R and F/B are calculated. The default is 180
                        which means the exact opposite of the forward-dir
  --rear-angle=REAR_ANGLE
                        angle for calculating rear gain (max 270)
  --beamwidth-ratio=BEAMWIDTH_RATIO
                        ratio for calculating beam width in dB, default=3.01
  --vertical-gain       calculate vertical gain
  --horizontal-gain     calculate horizontal gain [default]
  --total-gain          calculate total gain
  -f FREQUENCY_DATA, --frequency_data=FREQUENCY_DATA
                        a map of frequency to (angle, expected_gain) tuple
  --cleanup=CLEANUP     IGNORED
  --param-values-file=PARAM_VALUES_FILE
                        Read the parameter values from file, generate
                        output.nec and evaluate it instead of the input file.
                        The file should contain two lines: space separated
                        parameter names on the first and space separated
                        values on the second.
  --agt-correction=AGT_CORRECTION
                        ignored. agt correction is always applied
  -c, --centers         run sweep on the channel centers
  --chart               IGNORED
  --js-model            IGNORED
  --html=HTML           output html file, set to 0 to disable
  --publish             output html file using http://clients.teksavvy.com/~nickm
                        for resources
```

Optimization - nec.opt

nec.opt can be used to optimize a parametrized nec file. It has two optimization modes:

Global search
: this is the default search mode - finds the optimal configuration searching globaly. Provided by differential_evolution.py which was extracted from the package cctbx.

Local search
: requested with the -L command option - finds the local optimum close to the supplied model. Provided by the file simplex.py extracted from SCIPY which implements the Nelder-Mead or downhill simplex method.

The optimization works only on those parameters of the nec input that have limits specified in a comment like that:

```
SY feed = 0.0328 ' 0.01, 0.1
```

For example the command line:

```
c:>python -m nec.opt --num-cores=4 --vhf-hi -t(8.2,8.2) --uhf -t(12.5,13.5) input.nec
```

Searches, using differential evolution, an antenna based on the parametrized input.nec that has 8.2db in the VHF-hi range and 12.5db increasing to 13.5db in the UHF range.

Options

nec.opt options can be supplied from the command line or as CMD--OPT line in the comments section of the NEC file, e.g.

```
CM My VHF-hi/UHF antenna
CM Date: ...
CMD--OPT -s(174,14,4) -t(6,6.5,7,6.5) --swr-target=2.7 -s(470,19,13) -t(11,11.5,12,11.5) --swr-target=2
CMD--OPT -F max_gain_diff --num-cores=4 -a15 --de-np=50 -r restart.log
```

```
CM ....
CE
.....
```

multiple CMD--OPT lines are supported but one option should not be split between lines.

nec.opt inherits all the options from nec.eval and adds few more :

```
c:>python -m nec.opt --help
Usage: opt.py [options]

Options:
  -h, --help            show this help message and exit
......all nec.eval options are here ......
  --noagt-correction
  -l FILE, --log-file=FILE
                        log file. The default is your_input_file.opt_log.
  -S, --seed-with-input
                        use the input file as one of the population members
                        (creates bias towards the input file if it has a good
                        score)
  -t TARGET_LEVEL, --target-level=TARGET_LEVEL
                        appends target level(s) for a sweep, the number of
                        target levels must match the number of sweeps and they
                        are paired positionally. Examples1: -s (174,6,8) -t
                        (8,9) means target levels linearly increasing from 8
                        to 9 for the frequencies from 174 to 216. Example2: -s
                        (174,6,8) -t (8, 8.5, 9.5, 9) means target levels of 8
                        for 174, 9 for 216 and gradually increasing levels
                        from 8.5 to 9.5 for the range 180 to 210
  -M MAX_ITER, --max-iter=MAX_ITER
                        The default is 10000. The script can be interrupted
                        with Ctrl+C at any time and it will output its current
                        best result as 'output.nec'
  -L, --local-search
  -T LOCAL_SEARCH_TOLERANCE, --local-search-tolerance=LOCAL_SEARCH_TOLERANCE
  -F TARGET_FUNCTION, --target-function=TARGET_FUNCTION
                        An expression composed of statistical tokens and any
                        of the nec file parameters, by default it is
                        'max(max_gain_diff, max_swr_diff)'. All statistical
                        tokens are of the form min_"value", max_"value",
                        ave_"value", min_ave_"value", max_ave_"value",
                        ave_min_"value" and ave_max_"value", where "value" is
                        one of the following: gain_diff, swr_diff, f2r_diff,
                        f2b_diff, net_gain, raw_gain, ml, swr, agt_correction,
                        f2r, f2b, real and  imag. A full access to all results
                        per frequency is also provided for the same tokens.
                        For example, results[0] gives access to all results
                        for the first sweep, results[0]["net_gain"] is an
                        array of all net gains for all frequencies of the
                        first sweep, and finally results[0]["net_gain"][0]
                        gives the net gain for the first frequency of the
                        first sweep. The numeric indices are from 0 to
                        count-1, where count is the number of sweeps for the
                        first index and the number of frequencies for the
                        second.
  --swr-target=SWR_TARGET
                        defines the swr target curve in the same way as target
                        gain is defined. the default value is flat swr (2,2).
                        One per sweep can be specified. The last one defined
                        is used as default if the sweeps are more.
  --f2r-target=F2R_TARGET
                        defines the f2r target curve in the same way as target
                        gain is defined. the default value is flat f2r
                        (15,15). One per sweep can be specified. The last one
                        defined is used as default if the sweeps are more.
  --f2b-target=F2B_TARGET
                        defines the f2b target curve in the same way as target
                        gain is defined. the default value is flat f2b
                        (15,15). One per sweep can be specified. The last one
                        defined is used as default if the sweeps are more.
  --de-dither=DE_DITHER
  --de-f=DE_F           The DE's differential parameter. Should be >.5, the
                        default is 0.55
  --de-cr=DE_CR         The DE's crossover parameter, the default is 0.9
  --de-np=DE_NP         The DE's population size parameter. The literature
                        recommends to use 10*(optimization_parameters). The
                        defaults is 50
  -P, --output-population
                        IGNORED
  -b OUTPUT_BEST, --output-best=OUTPUT_BEST
                        set to 0 or 1 to output the best score nec file as
                        'best.nec'. Default is -1 (output if not in local
                        search).
  -p PARAMETERS, --parameters=PARAMETERS
                        If not empty restrict the list of optimization
                        parameters to this list.
  -r RESTART_FILE, --restart=RESTART_FILE
                        restart from population saved in a file.
  --omni                parse all horizontal angles
  --quiet               disable all output but errors
  --verbose             enables extra output
  --strict-max-target   use if your target function has no averaging i.e. if
                        the result for a single frequency can be used to
                        declare a model as worse in comparison with the score
                        of another model. The default target function
                        max(max_swr_diff,max_gain_diff) is an example of such
                        function. Setting this option will speed up the
                        optimization, but it has to be used correctly.
  --engine-kill-time=ENGINE_KILL_TIME
                        Maximum time the nec engine is allowed to run before
                        it is considered hanging and killed. After 100
                        successful engine invocations this value is updated
                        with 10x the actual maximum running time of all
                        previous engine invocations
```

Target function

The target function is a user defined expression build using some or all of the available target function tokens. The available tokens are:

1. All symbols (**SY**) from the nec file.
   This allows some geometrical properties of the antenna to be optimized, for example one can minimize the boom length of a yagi

2. Statistical tokens calculated from the values reported by the nec engine.
   For every "value" calculated by the nec engine the optimizer defines 7 statistics all of which are valid tokens, and are composed by appending one or two of the prefixes "min_", "max_" and "ave_" to the name of the "value":

   - **max_value** - the maximum of "value"s across all evaluated frequencies
   - **ave_value** - the average of "value"s across all evaluated frequencies
   - **ave_max_value** - the average of all "max_value"s, where each "max_value" calculated from a single sweep and the average is taken across the sweeps
   - **max_ave_value** - the max of all "ave_value"s, where each "ave_value" calculated from a single sweep and the maximum is taken across the sweeps

and analogously

- **min_value**
- **min_ave_value** and
- **ave_min_value**

The "value"s evaluated for every frequency are:

- **gain_diff** - the difference between the target net gain and the achieved net gain
- **swr_diff** - the difference between the target swr and the achieved swr
- **f2r_diff** - the difference between the target F/R and the achieved F/R
- **f2b_diff** - the difference between the target F/B and the achieved F/B
- **net_gain** - the net gain of the antenna
- **raw_gain** - the raw gain of the antenna
- **ml** - the mismatch loss of the antenna (raw_gain - net_gain)
- **swr** - the swr
- **agt_correction** - the gain correction based on the AGT of the antenna
- **f2r** - F/R ratio
- **f2b** - F/B ratio
- **real** - the real impedance
- **imag** - the imaginary impedance

The "net_gain" for example spans the following 7 tokens: min_net_gain, max_net_gain, ave_net_gain, min_ave_net_gain, max_ave_net_gain, ave_min_net_gain and ave_max_net_gain.

When only one sweep is being optimized:

- the min_ave_"value"s and the max_ave_"value"s are the same as ave_"value"s
- the ave_min_"value"s are the same as min_"value"s and
- the ave_max_"value"s are the same as max_"value"s.

3. **New** - All results for all frequencies are accessible through the results variable. results[0], results[1], etc refer to the results for the first, second and so on sweeps, in the order they are specified on the command line.

   results[0]["net_gain"] refers to the array of all net_gain values for every frequency of the first sweep. For example the statistical token min_net_gain is equivalent to min(results[0]["net_gain"]) when only one sweep is defined.

   Finaly results[0]["net_gain"][0] refers to the net_gain of the first frequency of the first sweep.

4. **New** - --omni related tokens when --omni option is specified there are now 4 additional tokens that can be used in the target function:

   - omni_net represents the lowest net gain from all sampled angles. the tokens derived from it min_omni_net, ave_omni_net, min_ave_omni_net and ave_min_omni_net are designed to be maximized, ie there negatives should be used in the target function.
   - omni_gain_diff is the difference between omni_net and the target gain. the tokens derived from it max_-, ave_-, max_ave_ and ave_max_- omni_gain_diff are designed to be minimized.
   - around_net represents the average net gain from all sampled angles. the tokens derived from it min_around_net, ave_around_net, min_ave_around_net and ave_min_around_net are designed to be maximized, ie there negatives should be used in the target function.
   - around_gain_diff is the difference between around_net and the target gain. the tokens derived from it max_-, ave_-, max_ave_ and ave_max_- around_gain_diff are designed to be minimized.

The target function is always minimized, and that should be kept in mind when composing it.

For example it does not make sense to minimize the tokens "max_net_gain", "ave_net_gain", etc., since we need those generally maximized and that can be achieved by minimizing "-min_net_gain"(the negative "min_net_gain").
Also minimizing "ave_swr" is good, but minimizing "min_swr" is not, since that does not control the max_swr which can be quite excessive.

The general rule is that if you want a value minimized you should use its "max" or "ave" statistics, and if you want a value maximized use its negated "min" or negated "ave" statistics.

The "diff" values are designed for direct minimization.

On each iteration nec.opt outputs the parameter values and their **score** - the value of the target function.

If the default target function is used the **score** is the biggest of all differences between the desired target levels for each frequency and the achieved net gain for it. A **score of 0** means that all target levels have been at least achieved. A **negative score** means that all target levels have been exceeded and a **positive score** means that at least for one frequency the target level was not achieved.

Preparing the NEC file

Few changes are needed before the NEC file can be used with the optimize command:

1. Add CMD--OPT options line in the comments section
2. Add limits to the parameters that will be optimized

```
SY feed = 0.0328 ' 0.01, 0.1
```

3. Optionally add CMD--EVAL options line in the comments section
4. Optionaly add different colors to the antenna wires, by adding a comment after the GW line with the format #rgb or #rrggbb, e.g.

```
CE
.....
GW 1 11 x1 y1 z1 x2 y2 z2 radius ' #f00
....
```

the color applies to all consecutive wires until changed.

Download

Current version 0.16.134: Archived - For windows nec-0.16.134.zip and for Linux nec-0.16.134.tar.gz
Windows binaries for **python 3.*** - win32 and win64

If you get error trying the links above use this version nec-0.16.134.tar.gz and install the NEC engines separately*

Version 0.16.134 :

- added new --omni target tokens **omni_net**, **omni_gain_diff**, **around_net** and **around_gain_diff**

Version 0.16.133 :

- Fixed GH card handling
- added predefined PI symbol

Version 0.16.132 :

- Fixed SC card handling
- Work around linux nec2 temp file problem

Version 0.16.130 :

- Fixed antenna.js handling of asymmetric horizontal patterns

Version 0.16.129 :

- Added --publish option to the nec.eval options

Version 0.16.128 :

- Fixed LD card regression

Version 0.16.127 :

- Improved LD card handling

Version 0.16.126 :

- Fixed linux installation and python 2 support

Version 0.16.125 :

- Created a pattern viewer, eval now outputs a pattern viewer link
- bugfix in AGT calculation when ground is present
- bugfix in source tag calculation during autosegmentation when the ofiginal source tag is not in the middle, start or end of the wire

Version 0.15.124 :

- presentation improvements: raw gain in chart, file name in model, chart and pattern view, pattern font size
- bugfix in fixed segmentation
- bugfix in segment counting with geometry validation off

Version 0.14.120 - fixed handling of TL, GA and GH cards

Version 0.14.117 - added --validate-geometry options, added error recovery after var evaluation failure, fixed integer division on python 2.x

Version 0.14.115 - implemented process monitor which kills hanging engine processes

Version 0.13.114 - improved error reporting and input validation

Version 0.12.113 - has the following new features:

- automatic selection of engine based on number of segments
- Simplified restart file generation and optimizer state stats
- The windows packages come bundled with NEC/MP

Version 0.11.106 - has the following new features: Fixes the linux installation and adds nec_eval.sh and nec_opt.sh to the path (both scripts expect nec2 to be installed)

Version 0.11.105 - has the following new features:

- html output - includes 3D model, horizontal pattern and web links
- nec.opt and nec.eval options can be specified in the comment section of the nec file after CMD--OPT and CMD--EVAL "cards" (see options sections above)
- the installer registers evaluate and optimize commands on nec files (engines may be added to Python\Lib\site-packages\nec\engines)
- autosegmentation support for EX 5 cards
- automatic restart file generation
- swr, f2r and f2b targets can now be specified on per sweep and as curves
- all results are now accessible for use in the target function(see target function section above)
- some bug and typ0 fixes

Versions 0.10.77, 0.10.78, 0.10.79 and 0.10.80 addressed multiple python 3 related problems in the local search

Version 0.10.76 fixed new bug in transformation lines handling

Version 0.10.74 it has been almost a year since the previous version, so the list of changes is incomplete but includes:

- support for Python 3,
- support for NEC/MP
- fixed autosegmentation to work with TL and NT cards
- changed the default parameters for the Differential Evolution
- changed the way average of logarithmic values is calculated: before it was average(10*log(gain)), now it is 10*log(average(gain))
- added support for ^ operator to calculate exponents
- fixed other minor typ0s and bugs and hopefully did not create too many new ones

Version 0.9.71 restored removed output cleanup code

Version 0.9.70 reworked boundary handling, fixed DE cr bug, reworked sweep splitting

Version 0.8.68 fixed optimization problem with --forward-dir

Version 0.8.67 added --horizontal-, --vertical- and --total-gain options

Version 0.8.65 added --forward-dir and --backward-dir options

Version 0.7.64 fixed broken F/B ratio optimization

Version 0.7.63 added quiet and verbose mode, fixed swr only optimization, renamed target_level option

Version 0.7.61 fixed the autosegmentation to better handle short source wires.

Version 0.7.58-60 fixed few min <-> max typos.

Version 0.7.57 adds many new target function tokens.

Version 0.7.56 works around a problem on some engines reporting negative real impedance.

Version 0.7.55 allows the nec file parameters (even the calculated ones) to be used in the target function.

Version 0.7.54 added f/r optimization option.
Version 0.6.52 fixed broken (in 0.5.51) {freq:(angles,target)} optimization.
The version 0.5.51 was 5-6 times faster on the global search than 0.4.46. The --agt option is no longer used and agt correction is applied always.

License

The python scripts in the package are covered by GPL except for differential_evolution.py and simplex.py which have their own license in the code.