KI7NNP

Inductor Software Manual

Jed Marti

KI7NNP

February 14, 2024

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A manual for 4 programs used to generate 3D printable inductors.

coil Generate an OpenSCAD description of a 3D printable plastic coil form.

inductor Select a prediction equation, size limits and optimize turns and size for a requested inductance.

indequ A genetic algorithm program to optimize multivariate polynomials to fit measured inductances.

compare Compare equations against measured values.

The **indequ** program runs only under Linux and will work satisfactorily on a Raspberry PI 4, somewhat slowly, but exploits the maximum number of processor cores you specify. The other programs are command line driven and run under Windows and Linux.

The source code and executables can be downloaded and you're free to do whatever you want with them.

Chapter 2

coil: Generate CSG coil form

Once you've determined the size and number of turns for a coil, the **coil** program will generate an OpenSCAD file describing a cylinder with holes for the leads and channels appropriate to the size wire specified. For example, Figure 2.1 shows the OpenSCAD view of a coil with 20 turns ready to convert to STL and send to a printer.

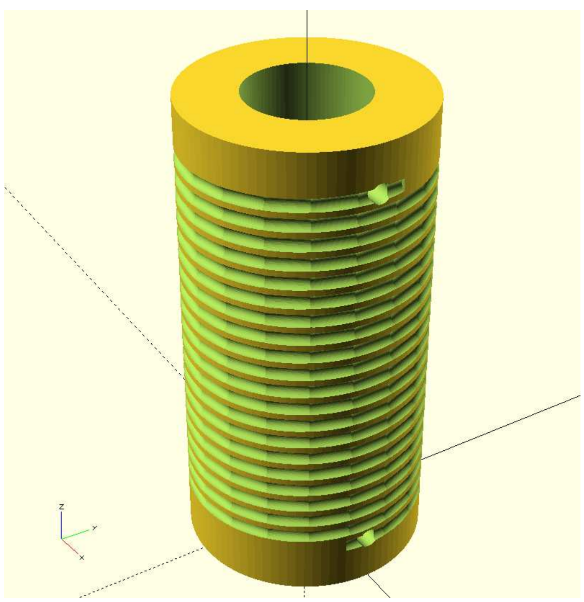

Figure 2.1: 20 turns, #22 wire, .5" diameter, 0.04" spacing

The program is completely command line driven.

coil *outfile*`.scad` [options]

Here *outfile*.scad is where the program puts the OpenSCAD description and must be present. `.scad` is the preferred suffix for files of this type but you can use anything. The options are of the form:

-*name* value

Table 2.1 gives the required options. The program will not run unless all three parameters are present. All parameters are case insensitive - file names are case sensitive unless running under Windows.

Parameter	**Description**
-length *float*	The coil length in inches.
-radius *float*	The coil radius in inches.
-turns *integer*	The number of turns.

Table 2.1: coil program required parameters.

Other parameters are optional. The program will supply default values if they are not present. Figure 2.1 was created by:

```
coil 20t.scad -length 0.8 -radius 0.25 -turns 20
```

Parameter	Def.	Description
-cylindersperturn *int*	18	The number of cylinders used for each coil turn. For diameters greater then 0.5", this value should be increased.
-end *float*	0.1	The amount of plastic overhang after the last turn. Must be greater than zero or the last turn hole will not hold the wire.
-formfaces *int*	90	The number of rectangles used to generate the cylinder. Can be decreased for smaller radii and increased for larger.
-gauge *int*	22	The wire gauge to be used. Values from 14 to 32 are known.
-layer *float*	0.011811	The printer layer thickness in inches.
-wallthickness *float*	0.125	The plastic cylinder wall thickness in inches. This should be increased for coils with diameters greater than 0.75 or for coils with heavier gauge wire.
-wireindentfaces *int*	9	The number of rectangles used for each cylinder in the wire channel. Should be increased for large gauge wire.
-wiresize *float*	0.028	The wire diameter in inches (overrides -gauge).

Table 2.2: coil program optional parameters.

Increasing the number of cylinders per turn makes for nicer coils but greatly increases the CPU time necessary for OpenSCAD to generate the STL files. The following generates the much nicer coil form in Figure 2.2. The resulting scad file is over 1400 lines long. and takes about and hour and 45 minutes on a fast machine.

```
coil t20x.scad -length 0.8 -radius 0.25 -turns 20 \
```

```
-cylindersperturn 72 -wireindentfaces 18
 theta = 1.45871
 wire length = 31.4261
 Cylinder length 0.0261884 inches
 Cylinder offset Z 0.000555556 inches
```

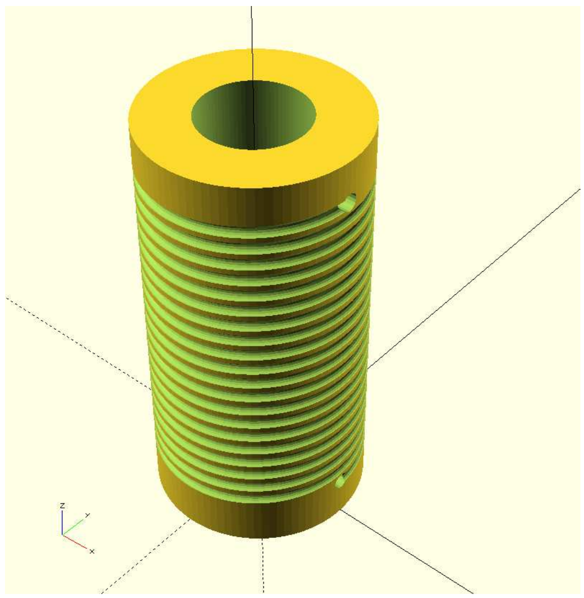

Figure 2.2: 20 turns, #22 wire, .5" diameter, 0.04" spacing

2.1 Source Code

This includes 3 source files and two executables in the **coil.tar** archive.

File	Description
00README	Description and instructions.
coil	Linux executable.
coil.c	Program source code.
coil.exe	Windows executable.
Makefile	Instructions for make program in Linux and Windows.
wire.h	What we know about enamel magnet wire.

Table 2.3: **coil.tar** archive contents.

Chapter 3

inductor: Best Approximation

This module conducts an exhaustive search for the best radius, length, and number of turns to achieve a requested inductance. It will also give an analysis of all the known equations and whether or not they apply to the value.

3.1 Command Line Parameters

For operation, the only required parameter is the inductance in micro Henries which can appear anywhere. A number without a prefix is assumed to be the request. For example:

```
inductor 5.6
```

requests the ARRL original equation for 5.6 μH.

All parameters are optional and each parameter name is prefixed with a -. The parameter names are not case sensitive.

Parameter	Default	Description
identifier	ARRL Original	The equation to use, see Table 3.2.
-gauge *int*	22	The wire gauge for winding the coil.
-maxlength *float* [mm | inches]	3	The coil's maximum length in inches (default) or millimeters if so specified.
-maxradius *float* [mm | inches]	1	The coil's maximum radius in inches (default) or millimeters if so specified.
-maxturns *int*	50	The maximum number of turns allowed.
-minlength *float* [mm | inches]	0.05	The coil's minimum lengh in inches (default) or millimeters if so specified.
-minradius *float* [mm | inches]	0.0625	The coil's minimum radius in inches (default) or millimeters if so specified.
-minturns *int*	1	The minimum number of turns to check.
-reslength *int*	100	The number of tests to make between the minimum length and the maximum length.
-resradius *int*	100	The number of tests to make between the minimum radius and the maximum radius.

Table 3.1: Inductor parameter forms.

Table 3.2 shows the known equations. Equation names are not case sensitive.

Name	Usage
ARRLOriginal	The original ARRL equation and constants. The default.
ARRLTurns	The ARRL equation tuned to the `turntest.csv` file. Only useful for .04 spacing #22 wire.
ARRLnls	The ARRL equation tuned to the `nls.csv` file. Only useful for 0.028" spacing #22 wire.
ARRLall22	The ARRL equation tuned to the `all22.csv` file. Only useful for #22 gauge wire.
ARRLalljan24	The ARRL equation tuned to the `alljan24.csv` file. Not very good predictor.
Simple	Function of length and radius. Only good for 0.028" spacing, #22 gauge wire. Tuned against `nls.csv`
Equation8	Not tuned - not good for anything.
EquationA	Includes wire gauge. Best for #22 gauge wire though.
EquationD	Best all around for all sizes and wire gauges. Tuned against `alljan24.csv`.
EquationF	No wire gauge but reasonable all around.
EquationI	Good all around and improving.
RFC	Poor all around.
RF1	Poor all around.
Lundin	Poor all around.
Miller	Poor all around.

Table 3.2: **inductor** program equation names.

3.2 A Sample Run

I'm looking for a form for a $5.4\mu H$ inductor so I start with the defaults.

```
inductor 5.4
inductor V1.00.004
For equation ARRLOriginal:
Best result 5.4 uH, 23 turns, length = 2.8525",  radius = 0.5875"
Recommendations:
        ARRLOriginal: marginal
           ARRLTurns: marginal
             ARRLnls: marginal
```

```
           ARRLall22: reasonable
           ARRLjan24: marginal
              Simple: marginal
           Equation8: notRecommended
           EquationA: highlyRecommended
           EquationD: marginal
           EquationF: recommended
           EquationG: notRecommended
           EquationH: notRecommended
           EquationI: notRecommended
                 RFC: marginal
                 RF1: marginal
              Lundin: marginal

Execute this:
coil inductor.scad -radius 0.5875 -length 2.8525 -turns 23
```

If you copy the final line and execute it, the OpenSCAD file will be generated.
The program recommends EquationA or EquationF so I rerun using EquationA. Notice that the recommended form is much smaller.

```
inductor 5.4 equationa
inductor V1.00.004
For equation EquationA:
Best result 5.39996 uH, 47 turns, length = 1.3185",  radius = 0.19375"
Recommendations:
        ARRLOriginal: marginal
           ARRLTurns: marginal
             ARRLnls: marginal
           ARRLall22: marginal
           ARRLjan24: marginal
              Simple: marginal
           Equation8: notRecommended
           EquationA: highlyRecommended
           EquationD: marginal
           EquationF: marginal
           EquationG: notRecommended
```

```
          EquationH: notRecommended
          EquationI: notRecommended
                RFC: marginal
                RF1: marginal

Execute this:
coil inductor.scad -radius 0.19375 -length 1.3185 -turns 47
```

3.3 The Source Code

The **compare** and **inductor** programs are included in the **allind.tar** archive.

File	Description
00README	Description and instructions.
bestind.c	Exhaustive search for best solution.
cmdline.c	Command line parsing for **indcutor**.
compare	Linux executable.
compare.c	Main program for **compare**.
compare.exe	32 bit Windows executable.
compare.h	Prototypes, data structures for **compare**.
equations.c	Current known equations for both programs.
equations.h	Prototypes for equations.c.
inductor	Linux executable for **inductor**.
inductor.c	Main program for **inductor**.
inductor.exe	32 bit Windows executable of **inductor**.
inductor.h	Prototypes, structures for **inductor**.
Makefile	Linux and Microsoft Makefile.
readind.c	Read inductor csv files for **compare**.
wire.c	What we know about wire gauges.
all22.csv	Measure data for all #22 gauge coils.
alljan24.csv	All measure data for coils $> 0.5\mu H$.
nls.csv	Fixed spacing #22 gauge coils.
turntest.csv	Fixed spacing/diameter #22 gauge coils.

Table 3.3: Contents of **allind.tar**

Chapter 4

compare: Analysis

The **compare** program runs all equations against a database and shows the test limits and RMS error. It runs on both Linux and Windows.

The test files are in Comma Separated Variable format - numbers and so on separated with commas. Table 4.1 gives the fields and their values.

Name	**Description**
Henrys	The measured inductance in Henrys.
Radius	The coil radius in meters.
wire	The wire diameter in inches.
gauge	The wire gauge number.
Turns	The number of coil turns.
Length	The coil length in meters.
FileName	The original file name from which the data was derived.

Table 4.1: Database CSV file columns.

Four summary data files are included in **allind.tar**, see Table 3.3 on page **??** for the distribution file. The subsets and their restrictions are summarized by **compare** in Table 4.2.

Subset	Turns	Gauge	Diameter	Length	μH
turntest.csv	8 - 50	22	.5"	.32"-2"	.612 - 7.06
nls.csv	5 - 40	22	.258" - .866"	.14" - 1.12"	.516 - 19.8
all22.csv	5 - 40	22	.261" - 2"	.14" - 3"	.502 - 40.73
alljan24.csv	5 - 50	14 - 28	.261" - 2"	.14" - 3"	.502 - 40.73

Table 4.2: Database restrictions.

4.1 Source Code

The source code and CSV data files are part of the **allind.tar** archive. See Section 3.3 on page 21.

Chapter 5

indequm: Genetic Optimization

The **indequm** program attempts to optimize a multivariate equation with the variables:

Variable	Description
N	The number of turns.
R	The coil radius in meters.
L	The coil length in meters.
W	The wire radius in meters.

Table 5.1: Polynomial Variables

Most equations include the use of μ_0 in the belief that cores with different permeability might allow the use of μ_r. This is covered somewhat in the experimental report [2].

5.1 Genetic Algorithm

Genetic algorithm optimization is an important AI technique for problems with many solutions [1].

Our problem is to optimize the coefficients of multivariate polynomials to fit a set of data much like linear list squares or polynomial regression. Each multivariate polynomial is a sum of coefficients and the variables of Table 5.1. A part of a cubic solution might be:

$$\mathcal{R} = \sum_{i=0}^{3} r_i R^i \tag{5.1}$$

For example, the following with cubic components is a reasonable equation that includes wire radius.

$$\mathcal{L} = \mu_0 \frac{(r_0 + r_1 R + r_2 R^2 + r_3 R^3)(t_0 + t_1 N + t_2 N^2)(w_0 + w_1 W + w_2 W^2)}{l_0 + l_1 L + l_2 L^2 + l_3 L^3} \tag{5.2}$$

The program assigns values to r_i, evaluates the equation for the variables of Table 5.1 against all measured coils computing a "goodness". If the computed inductance for coil j is L_j and the measured value is M_j then the goodness value for a set of $m+1$ coils is:

$$goodness_j = \sqrt{\frac{\sum_{j=0}^{m} \frac{(L_j - M_j)^2}{M_j}}{m+1}} \tag{5.3}$$

The process is repeated with the best values being the solution.

5.2 The Control File

To start the process, a file specifies the ranges and resolution of all coefficients and some control variables to help the process along. The control file can have blank lines, comments begin with # or \\ in the first columns.

5.2.1 Fixed Variables

Table 5.2 has control values common to all equations.

Variable	Type	Def.	Description
cvarname	*string*	NULL	Variable name for C file table.
direction	*double*	0.01	Change direction this percent (breeding parameter).
javarname	*string*	NULL	Variable name for object file.
maxgenerations	*int32_t*	100000	The number of generations to run before quitting.
minormax	*int32_t*	8388608	How many minor evaluations to do before giving up.
noise	*int32_t*	1	How much debug to display.
pctchange	*double*	0.01	If range expanding enabled, do this percent.
population	*int32_t*	64	Number of tests in a generation.
processes	*int32_t*	1	Number of processes for concurrent processing.
quitat	*int32_t*	-1	If -1, quit on **maxgenerations** otherwise quit after this number of generations without an improvement.
raiselower	*int32_t*	10	Check for range problem this number of generations.
randomminor	*double*	0.33	Percentage of parameters to adjust randomly.
rangeexpand	*int32_t*	0	If non-zero check for range expansion.
resultc	*string*	NULL	Where to put C file with result values.
resultfile	*string*	NULL	Where to put final results in text form.
resultja	*string*	NULL	Where to put object descriptions.
save	*int32_t*	4	Number of best results to save and breed.
seed	*int64_t*	1033888277	Random number seed for Twister.
tries	*int32_t*	20	How many attempts to breed before giving up.

Table 5.2: Fixed control values.

5.2.2 Coefficients

The equation to fit is specified at compile time and a series of macros expands the symbol table for each coefficient. For example r_2 is the coefficient `r2`. The available symbols are specific to equation.

Each coefficient must have a specification in the control file.

coefficient-name, low, high, major-increment, minor-increment, minor-count

When deciding what coefficient values to test, the system randomly selects values between *low* and *high* but spaced *major-increment* apart. The double-precision values are used to generate a hash code into a quadratic search table. If this set of coefficients has already been evaluated, then the system tries for another set. It will attempt this **tries** times (default 20) before giving up.

Once the system determines a set of coefficients, it evaluates up to **minormax** variations around the selected values. Here the *minor-increment* and *minor-count* is used to go plus and minus $\frac{minor-count}{2}$ on either side of the major value. The best result is returned. Note that this can sometimes result in the solution exceeding the specified range by a small amount.

This approach is desirable to spread the computation load amongst several processor cores while minimizing the amount of interprocessor communication that must be used. Generally it is best to tune the minor values to give a compute time of 1 to 5 seconds. While the system is running typing a Enter will display the current solution state and the time per process in microseconds. The number of coefficient values tested is a product of all the *minor-count*'s. Too many and you're wasting compute time, too few and you spend all your time in interprocess communication.

coefficient-name one or two letters and the index.

low The lowest value the coefficient is allowed to have. Double-precision floating-point.

high The highest value the coefficient is allowed to have. Double-precision floating-point.

major-increment Initial testing increment between *low* and *high*. Double-precision floating-point.

minor-increment The increment around the major value.

minor-count The number of minor values to test.

5.2.3 Best Estimate

You can start the program with your best estimate of the solution. This can be a guess or the result of a previous run where the coefficients were near the boundaries you set. To do so, you specify each coefficient name prefixed with **best**. Thus for two radius coefficient ranges:

```
...
r0,        -10,         10,      0.01,      0.002,   5
r1,       -100,       100,        0.1,       0.02,   5
...
```

you might enter:

```
bestr0    -0.13595004421482
bestr1    64.53039162318042
```

The system will use these values at startup instead of completely random values.

5.3 Execution

indequ is command line driven.

indequ *inductors*.csv *control file*.indequ

inductors.csv is the file of measured inductances that you wish to fit. For example, **all22.csv** shown previously. Or you can make your own measurements.

The second argument is the control file, typically with the **.indequ** suffix.

When compiled for Equation D, we see the following:

marti@ulam: /ki7nnp/indequm$ **indequ**
indequ V20.03.004 EQUATION D
Usage: indequ <testinductors> [<ranges>] [outf]
Coefficients: r0 r1 r2 r3 t0 t1 t2 w0 w1 w2 l0 l1 l2 l3

We see the equation the system was compiled for and the coefficients that are to be explored.

During a run, every new improved test, previous best goodness values. Every generation is also shown. For example

```
...
gen = 3615 done already 354376, rejected 1768
354367, 0.0397822, 0.0397822, 0.0397822, 0.0397822
gen = 3616 done already 354474, rejected 1769
354458, 0.0397822, 0.0397822, 0.0397822, 0.0397822
354499, 0.0397822, 0.0397822, 0.0397822, 0.0397822
gen = 3617 done already 354572, rejected 1770
354559, 0.0397822, 0.0397822, 0.0397822, 0.0397822
354606, 0.0397822, 0.0397822, 0.0397822, 0.0397822
gen = 3618 done already 354670, rejected 1771
gen = 3619 done already 354768, rejected 1771
gen = 3620 done already 354866, rejected 1774
...
```

We see that the current best goodness is 0.397822 and that we're currently working generation 3621. If you hit enter, you would see something like:

```
...
EQUATION I: time/process: 2.85141e+06 us
// Goodness = 0.0397899
   r0 =     -0.13527124632082
   r1 =     64.30593666573590
   r2 =   7088.43599922746398
   r3 = -11000.00000000000000
   w0 = -2216.00305291556879
   w1 = -5228.82295293508560
   w2 = -31088.07463817565076
   w3 = -499740.00000001396984
   t0 = -2183.19041861923824
   t1 =    579.79185831925872
   t2 =   1464.66711457692463
   t3 =     -4.99710017432386
   l0 =    -23.92692960071267
```

```
   l1 = -2693.45269235206251
   l2 = 11213.24250139096876
   l3 = 41129.11641499128746
range table
  r0:        -10          10        0.01      0.002   5 (2001)
  r1:       -100         100         0.1       0.02   5 (2001)
  r2:          0       10000           1        0.2   5 (10001)
  r3:     -11000       -5000           1        0.2   5 (6001)
  w0:      -3000       -2000         0.1       0.02   5 (10001)
  w1:      -7000       -5000           1        0.2   5 (2001)
  w2:     -50000       -5000           1        0.2   5 (45001)
  w3:    -500000      -45000           1        0.2   5 (455001)
  t0:      -2500       -1000         0.1       0.02   5 (15001)
  t1:          0        1000         0.1       0.02   5 (10001)
  t2:       1000        2000         0.1       0.02   5 (10001)
  t3:        -10          10         0.1       0.02   5 (201)
  l0:       -100           0         0.1       0.02   5 (1001)
  l1:      -3000       -2000         0.1       0.02   5 (10001)
  l2:      11000       12000         0.1       0.02   5 (10001)
  l3:      22000       50000         0.1       0.02   5 (280001)
gen = 1253 done already 122900, rejected 577
122895, 0.0397899, 0.0397899, 0.0397899, 0.0397899
122901, 0.0397899, 0.0397899, 0.0397899, 0.0397899
...
```

We're taking about 2.85 seconds for each process (not each generation), and a current goodness of 0.0397899. Following are the current best coefficient values and the range table. We see here that coefficient **w3** is nearing its -500000 limit. We probably need to adjust the limit downward and restart (this can be done automatically by setting the **rangeexpand** parameter to 1.

The generation number (1253) is the number of times **population** tests has been run. The "done already" is the total number of coefficient sets tested, and the "rejected" value is the number of sets rejected because they've already been run.

5.4 Compilation

Because the system uses the Linux process functionality to distribute computation, it will not run under Windows. You need to modify **CFLAGS** to indicate with equation you want to compile for.

If you're having trouble with process cleanup, you may wish to change from -Ofast to -O1. This sometimes seems to make a difference though the code runs much slower.

5.5 Adding a New Equation

New equations are added to *equations.h*. The template is:

#elif defined(*yourname*)
#define EQUATION "*yourname*"
#define VARS \
    X($coef_0$) \
    X($coef_1$) \
    ⋮
    X($coef_n$)

#define LAST s_$coef_n$

#define VARSI \
    X($coef_n$) \
    ⋮
    X($coef_0$)\

#define DEN   *equation for denominator*
#define NUM   *equation for numerator*

The modify **Makefile** and define *yourname* and do:

```
make clean
make
```

Build a configuration file with the ranges for your coefficients and sit back and watch the corn grow.

5.6 Source Code

Files in **indequm.tar**.

file	**Contents**
00README	Archive contents description.
breed.c	Takes a set of one or more coefficients and modifies the values. There are several different ways to do this.
coils.c	Showing results, writing them to files.
equations.h	Equations to compile - add yours here.
o evaluate.c	Do the minor scan on a set of coefficients for the defined equation.
geneticproc.c	The main loop for parcelling out coefficient sets to inactive processes and capturing their results.
genrand.c	Generate a completely random solution.
indequ.c	Main program.
indequ.h	Prototypes and data structures.
kbhit.c	Checks for an input character to interrupt the program.
library.c	Keep track of tested solutions so they aren't repeated.
Makefile	Build for Linux. Fix CFLAGS for equation.
process.c	Subtask control and program exit.
readind.c	Read the measured inductor CSV file.
sort.c	Sort results by measured inductor - worst first.
storage.c	Manage dynamic storage - reuse data structures.
symtab.c	External name interface.
time.c	Used for timing sub processes.
twister.c	64 bit random number generator.
vars*.indequ	Various control files for the different equations.

Table 5.3: **indequ** source code

Index

Bibliography

[1] David E. Goldbert. *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley, 1989.

[2] Jed Marti. Experiments with 3D printed coil forms. *//http://www.cog9llc.com.*