

Multiple Iteration – Respectable Trial-and-Error

by: Meriwether L. Baxter, Penfield, New York

American Gear Manufacturers Association



TECHNICAL PAPER

Multiple Iteration – Respectable Trial-and-Error

Meriwether L. Baxter,
Penfield, New York

[The Statements and opinions contained herein are those of the author and should not be construed as an official action or opinion of the American Gear Manufacturers Association.]

ABSTRACT:

“Trial-and-error” was once a dirty word. Now, with computers, it has become the most valuable tool in engineering calculation. The procedure described in this paper permits any number of input variables (say A,B,C) to be systematically varied until an equal number of results (say X,Y,Z) are zero, regardless of the complexity of the equations relating A,B,C to X,Y,Z. It has been used successfully by the writer for up to seven variables.

This paper is not concerned with the structure of the computer program representing this procedure, but rather with the kinds of engineering problems that can use it, and how it can be used as a subroutine in engineering programs.

Two actual gear applications will be described. There will also be a discussion of problems of non-convergence that may arise, and the necessity for good starting values.

The program is written in elementary BASIC, so that it can be read and modified by the user if desired.

Copyright © 1990

American Gear Manufacturers Association
1500 King Street, Suite 201
Alexandria, Virginia, 22314

October, 1990

ISBN: 1-55589-563-8

MULTIPLE ITERATION Respectable Trial-and-Error

Meriwether L. Baxter, Gear Consultant
86 New Wickham Drive
Penfield, NY 14526

INTRODUCTION

Trial-and-error has been used since earliest times to solve difficult problems, and is, of course, still in use. An example is the "guess-and-divide" method of getting a square root, used with the old mechanical calculators, and on modern computer chips.

Iteration is simply logical and orderly trial-and-error. It is laborious and time-consuming when done by hand, but fast on a computer, and has become an important engineering tool.

Most engineers who write programs will have used iteration at some time; but it is likely that writing the iteration section of the program became so complicated that the actual engineering problem and the principles involved became obscured. Use of the present program as a subroutine permits one to keep the main program in focus.

This paper is not concerned with the iteration method or program, but rather with how to use it effectively.

TYPE OF APPLICABLE PROBLEMS

Generally, this method is used when we can calculate from input variables A,B,C, --- to answers X,Y,Z, ---, but it is impossible or impractical to solve in reverse.

The impossible cases include transcendental functions such as the arc involute, and very long and complicated programs such as are sometimes found with hypoid and worm gearing problems.

The engineer may decide that while the equations can probably be solved backward, he does not want to bother with it, and prefers to use iteration. Furthermore, there are usually multiple solutions to gear geometry problems, so that it is difficult to know which signs to use before radicals, or which quadrant to use for angles. With iteration the meaningful solution is forced by the starting values chosen.

GEAR GEOMETRY PROBLEMS

There are two specific geometry problems that the gear engineer may be asked to solve that lend themselves to iterative solution.

The first is to find points on the tooth surface of the member conjugate to a generating gear whose surface is known. In this case, the rotative position of the generating gear is selected, and we need to find points on the line of contact.

Two parameters are required to specify a point on the generating gear tooth surface, such as the radius and the axial position; with these specified we can calculate the location of the point and the direction of the surface normal.

The procedure for finding a contact point is therefore to select the value of one of these parameters (say the position along the face), and vary the other parameter (the radius) until the basic law of gear tooth contact is satisfied. Thus, only single iteration is usually required for solving this problem.

The second problem is much more difficult; this is to find the point of contact between known non-conjugate surfaces. Here again the rotative position of one member is selected; but we do not know the location of the contact point on either member, or the rotative position of the second member. Thus, there are five input parameters that can be changed to achieve contact.

1. The points on the two members must coincide in space (three tests will determine this).
2. The surface normals at that point must coincide (two more tests).

Thus, in general, a five-fold iteration is required. The second example given later is of this type.

HOW TO USE THIS SUBROUTINE

Refer to Figure 1, which is a flow chart of a program using this subroutine. The procedures in the boxes are used in the iteration; the rest is your program.

The first box shows the dimension statements required for iterating on up to ten variables.

The second box represents the calculations required to get good starting values for the variables. These are to be specified as V(1), V(2), etc.

The third box controls the iteration.

1. NV is the number of variables.
2. N, NI, KIE are set as shown.
3. VS is the initial variable step. It should be small, but large enough to have a meaningful effect on all results. .001 is reasonable.
4. VT is the allowable error in each result. For much gear engineering .00001 is small enough.
5. The FOR statement starts the iteration; N is the number of the try; 15*NZ is the number of tries permitted. If the iteration has not then succeeded there is a serious problem; failures will be discussed later.
6. The ON NI+1 GOTO statement sends the iteration to the first appearance of each variable; the first address should be repeated.

In the fourth box the errors E(1), E(2), etc. have been calculated and we GOTO the subroutine at 10000, where the errors are tested and iteration proceeds by going to NEXT N if necessary. After NEXT N the iteration has failed if KE=2 or 0, and the programmer should GOTO an appropriate statement number. Otherwise, the iteration has succeeded and calculation may proceed.

ITERATION FAILURES

Iteration can fail in three ways. If KE=0, then convergence did not occur in the specified number of tries.

If KE=2, the procedure for solving NV simultaneous equations found the equations were incompatible with the variable values existing at that time. More about this later.

The third failure type can be program stoppage with the error message "Illegal Function Call", usually indicating that the iteration is vacillating wildly.

FAILURE CAUSES

1. Bad data, or programming error. Find and fix.
2. Poor starting values. This is by far the most likely cause of failure. It is essential that the iteration be a "downhill run" from the starting values to solution. Fortunately, in most engineering problems we can estimate the starting values fairly accurately.
3. Poor logic:
 - a. No solution exists. Why? (After all, two reasonable gear tooth surfaces usually contact somewhere, even if off the tooth boundaries).
 - b. Multiple solutions close to each other. This may occur, for example, near a profile cusp in undercut or fillet calculations. A different way to attack the problem will be needed.
 - c. The variables chosen do not effectively control the errors.

It is not generally effective to widen tolerances, change the starting step, or go to double-precision to correct failures. Try to find the real cause.

SAMPLE PROBLEM #1

Figure 2 is a listing of a simple program to calculate the arc involute. There is, of course, only a single variable.

The program itself runs from line 100 to line 520. The iteration subroutine runs from 10000 to 10800, and has been MERGED into the basic program to make a runnable unit.



It will be seen that the sections of this program correspond with the blocks in the flowchart of Figure 1. Only a few comments are required:

1. The starting value equation in line 310 is based on the first term of a power series. Its effectiveness can be judged from the results in Figure 3.
2. Lines 400 to 410 constitute the calculation block. In line 380 it will be seen that the number of the first entry line is repeated. This should always be done.
3. In line 500 the iteration has been successful and the result is printed. The program then proceeds by going back for a new involute function.

Figure 3 gives the results of this program for a range of involute functions from 0 to 1.4. It will be observed that the differences between the starting value and the answer become larger and larger, until failure occurs at an involute function of 1.4. A different procedure, or starting value equation is therefore required for large angles.

SAMPLE PROBLEM #2

Problem 2 is much more complicated, and involves five variables, and a program to track the point of contact between spherical tooth surfaces operating on parallel axes. Input data are:

C center distance
X axial distance between sphere centers
L(I) axis to sphere center (I=1 for pinion,
 2 for gear)
R(I) sphere radius

For each run, the angular position of the pinion sphere center AL(1) is also specified.

Figure 4 shows the arrangement.

The five variables that are to be altered to get contact are:

AA(I) latitude angle of point on sphere
AB(I) longitude angle of point on sphere
AL(2) angular position of gear sphere center

The requirements for contact are that the point on the pinion sphere coincide with the gear point:

$VPP(K) = VGP(K)$
(vectors from pinion center to point)

and that the surface normals coincide

$VPN(K) = VGN(K)$
(normal vectors - only two needed)

Figure 5 is a listing of this program.

Line 170 gives the dimension statements for iteration.

Lines 200-300 relate to the fixed data.

Line 310 accepts the desired pinion rotative position.

Lines 500-560 calculate the starting values. The equations are fairly crude, based on contact near center, but it will be seen from the results that they are sufficient.

Lines 600-630 control the iteration. Note that in the GOTO statement the first entry is repeated. This is necessary; there will always be NV+1 line numbers in this statement.

Lines 700-1060, plus the subroutine starting at 5000, contain the calculations for errors E(1) to E(5), after which we go to the iteration subroutine at 10000.

Lines 1130 to 1180 print results and continue.

Figure 6 shows results of runs of this program. Pinion angular positions were varied from -6 degrees to center in two degree intervals; then the pinion angle was increased by larger intervals until failure finally occurred with a pinion angle of 90 degrees, where contact between the spheres became impossible. The fairly crude formulas for the starting values were nevertheless good enough for this wide range of positions.

The reader may recognize that this particular example can be solved directly by considering it as a planar four-bar linkage. However, by introducing an angle between the shafts, iteration would be required.

CONCLUSION

It is hoped that this program will be of value to gear engineers, by permitting their concentration on gearing principles rather than computation techniques.