APPENDIX 4

ALGORITHM FOR CALCULATING THE DEVIATION FACTOR

Diagram of the Algorithm

```
 1.  begin
 2.      tot ← 0;
 3.      divisor ← 0;
 4.      for k ← 1 to entries do
 5.          begin
 6.              x ← data;
 7.              y ← data;
 8.              sum ← 0;
 9.              for l ← 1 to y do
10.                  begin
11.                      z ← data;
12.                      select
13.                          on z > x/2 do
14.                              sum ← sum + [(y-1)2    (x-z)2];
15.                          on 0 < z ≤ x/2 do
16.                              sum ← sum + [(y-1)2    z2];
17.                          on z = 0 do
18.                              sum ← sum + x2
19.                      endselect
20.                  end;
21.              tot ← tot + sum;
22.              divisor ← divisor + x
23.          end;
24.      deviation ← tot/divisor
25.  end.
```

Explanation of the Algorithm

Each line in the algorithm is numbered for the sake of reference. In the explanations below, the line number of the line to be discussed is given first, followed by comments. The algorithm is explained in terms of an analysis of lexical data, but it works the same way with the grammatical categories.

The data that the algorithm reads is laid out as follows. Each line of data represents a distinct Hebrew word that appears more than once in the chapter. The first element in each line of data is the number of times that a particular Hebrew word appears; this number is read into the variable *x* in line 6 of the algorithm (e.g., דָּבָר is rendered 8 times in LXX, so *x* = 8 [see table 1 in Appendix 2]). The next element in each line of data is the

number of distinct equivalents in the target language; this number is read into the variable *y* in line 7 of the algorithm (e.g., דָּבָר is rendered by 2 different Greek words, ρημα and λογος, so *y* = 2). There are *y* more elements in the line, each of which corresponds to the number of times the distinct equivalents in the target language occur (e.g., ρημα occurs 5 times and λογος occurs 3 times, so the whole line referring to דָּבָר reads 8, 2, 5, 3).

One terminological matter needs clarification before the algorithm is explained. The algorithm calculates the "amount of deviation", not the "deviation factor," for each line of data. The deviation factor for a line of data is equal to the amount of deviation divided by the total number of renderings for the Hebrew word in question (e.g., for דָּבָר, the amount of deviation determined by the algorithm is 18, which, when divided by the number of renderings [8], yields 2.25 for the deviation factor of the line). It is mandatory that amount of deviation be totaled, then divided by the total number of renderings at the end, rather than that the deviation factors of the lines be totaled, then divided by the number of lines of data at the end. If the second approach were used, a word like עולם, with only 2 entries, would be given the same weight as יהוה, which has 17 entries. Now it is time to explain the algorithm.

Line 2—*tot* contains a running total of the amount of deviation in each category.

Line 3—*divisor* contains the total number of items in the category, the sum of all the *x*'s (e.g., there are 132 Hebrew verbs, nouns, and adjectives that are rendered in LXX and are figured in the calculations in Appendix 2, so *divisor* = 132 at the end of the calculation [see line 4]).

Line 4—*entries* is the number of Hebrew words that occur more than once (e.g., there are 31 Hebrew verbs, nouns, and adjectives that are rendered more than once in LXX, so *entries* = 31; these 31 words are rendered a total of 132 times in LXX, so *divisor* will equal 132 at the end of the algorithm). The block of commands surrounded by "begin" and "end" is repeated *entries* times (i.e., 31 times in this case).

Lines 6 and 7—*x* and *y* are read from the line of data on each pass through the block of commands.

Line 8—*sum* will hold the amount of deviation for the current line of data.

Line 9—since there are *y* distinct renderings of the Hebrew word (in the case of דָּבָר, 2), the following block will be repeated *y* times.

Line 11—*z* is the number of times the item in the target language is used to render the Hebrew word (in the case of ρημα, *z* = 5).

Lines 12-19—this calculation is the heart of the algorithm. Within the range 0 to *x*, the distinct rendering in the target language is considered more consistent if *z* is closer to either end of the range (i.e., 0 or *x*) than if it is in the middle. The amount of deviation is the same whether *z* is 2 units from 0 (i.e., *z* = 2) or 2 units from *x* (i.e., *z* = *x*-2). The amount of deviation also increases with the number of distinct renderings in the target

language (e.g., if a Hebrew word is rendered consistently by a single Greek word, the factor *y*-1 = 0, and the amount of deviation for that line is 0; in the case of דָּבָר, *y*-1 = 1). Both *y*-1 and the second factor (the distance from the closer end point of the range; i.e., the smaller of *z* and *x*-*z*) are squared to reflect larger amounts of deviation further from the end points of the range. *z* = 0 is a special case, since normally *z* must be between 1 and *x*. *z* is 0 only when the target language fails to render any of the occurrences of a Hebrew form by its natural equivalent (e.g., if 4 Hebrew masculine nouns are rendered in Greek by 3 feminines and 1 neuter, a factor of 0 will be included in the line of data, alongside the 3 and the 1). *z* cannot equal 0 in lexical data, since lexemes in one language have no natural equivalent in another.

Line 21—by the time the algorithm gets here, *sum* contains the total amount of deviation for the line of data (the individual deviation factor for the line of data can be obtained by dividing *x* into *sum*). This figure is added to the running total in *tot.*

Line 22—*divisor* is increased by *x* to reflect the total number of words rendered.

Line 24—the sum of the amounts of deviation in each line, contained in *tot,* is divided by the number of elements analyzed, contained in *divisor,* to yield the deviation factor for the category.

A brief example of how the algorithm works is now in order, using the data from table 19, P's rendering of the conjunctions in MT. The data from table 19 is as follows:

Heb Lexeme	Syr Lexeme	Frequency
ו	ܘ	54
ו	ܐ	1
כִּי	ܐ	7
כִּי	ܐ ܠܐ	2

The two lines of data that correspond to this table are:

DATA: 55, 2, 54, 1 {55 ו's, 2 different renderings in P, 54 ܘ's, 1 ܐ}
DATA: 9,2,7,2 {9 כִּי's, 2 different renderings in P, 7 ܐ's, 2 ܐ ܠܐ's}

entries is equal to 2 (for the 2 lines of data).

The amount of deviation for the first line of data is 1^2 $1^2 + 1^2$ $1^2 = 2$.

The amount of deviation for the second line is 1^2 $2^2 + 1^2$ $2^2 = 8$.

The individual deviation factors are 2/55 = 0.04 (first line) and 8/9 = 0.89 (second line).

The overall deviation factor is (2+8) ÷ (55+9) = 10/64 = 0.16.