Implementing STV by Meek's method

I.D. Hill
d.hill928@btinternet.com

1 Introduction

At the time of the original implementation of STV by Meek's method [1] we were feeling our way. Later thought has shown that, in some respects, the details can be improved while keeping the overall plan. Thus my own later implementation, as part of a suite of programs to deal with the whole election process rather than just the vote counting, and to include other versions of STV as well as the Meek version, made some changes from that original implementation. The aim of this paper is to describe those changes and the reasons for them.

My program is written in the Pascal computer language. While designed to be used under the MS-DOS operating system, it can also be easily accessed from Windows XP.

In [1] Woodall gave mathematical proof that the Meek formulation has a unique solution for any given voting pattern, and that the method necessarily converges upon that solution. Strictly speaking that proof assumes infinite mathematical precision. In this paper I refer to that proof even though my implementation has only finite precision. Provided that the degree of precision is adequate, the approximation to Woodall's proof will be close enough for practical purposes.

2 Terminology

In [1] we used the term 'weight' for the fraction, of each vote or part of a vote received, that a candidate retains. This has now become known as the candidate's 'keep value', to be in accordance with the traditional term 'transfer value'.

We also used 'excess' for the amount of vote remaining after all candidates mentioned in the voter's preferences have received their shares. The more traditional, but longer, term 'non-transferable' is now used for this.

3 Arithmetic

In [1] the numbers of votes and the keep values were declared as 'real' variables in the computer sense. These would be represented in the computer in floating-point form, which is necessarily only approximate and there is no guarantee that exactly the same approximations will be used on different computer systems. Given the robustness of the Meek method, it is highly improbable that a different candidate would ever be elected because of this, except perhaps in the case of a tie, but it is thought wise to avoid even the possibility.

It is therefore better to make sure that the numbers are so represented that, although still approximate because only a finite number of decimal places is used, the results are necessarily identical on all computers. To achieve this, floating-point methods are avoided altogether, each 'real' number being represented by a pair of integers, integer arithmetic on computers being exact.

Assuming 32-bit integers to be available, the maximum allowable integer is 2147483647 so to allow 9 decimal places for the fractional part is safe and convenient. Thus a number such as 123.456, for example, is represented as a pair of integers with 123 as the value of its integral part and 456000000 as its fractional part. Adding or subtracting such numbers is simple enough, the integral parts are added or subtracted, and the fractional parts are added or subtracted. If the resulting fractional part exceeds 999999999, then 1000000000 is subtracted from it and 1 is added to the integral part. Similarly, if the resulting fractional part is negative, then 1000000000 is added to it and 1 is subtracted from the integral part. There is no need to worry about the whole number, rather than just its fractional part, ever being negative; that never happens within the Meek method.

Multiplication and division are not so simple, and special routines are necessary to enable them to be performed with no risk of overflow.

In principle, a fixed number of significant figures might be preferable to a fixed number of decimal places, but all that really matters is that the precision should be great enough as to ensure that the use of more precision would be virtually certain not to change the outcome. The fixed 9 decimal places undoubtedly satisfies this and is convenient.

4 Quota definition

Meek's formulation [2] used the integral part of $1 + T/(s+1)$, where T is the total number of active votes and s is the number of seats to be filled. He obviously intended that the initial 1 of this formula should be replaced by 1 in the last decimal place used, when not working solely in integers. An alternative approach is that of the second edition of Newland and Britton [3] in ignoring the initial 1 altogether if the calculation comes out exactly, while adding extra rules to ensure that no more than s candidates can be elected even in exceptional cases. In [1] we adopted the Newland and Britton approach (with the necessary extra rules) because the number of decimal places that would be used by a floating-point implementation was unknown.

When working solely in integers, or to only 2 decimal places as in Newland and Britton rules, there are advantages in their formulation, but those advantages are minimal where greater precision is used. For my implementation, therefore, I have included the addition of 0.000000001 to the quota, so that no extra rules are needed, while it is very hard to believe that such a tiny increment will ever cause any disadvantage.

5 Output

In [1], mainly because we were still feeling our way at that time, more output was given than now seems sensible, producing two tables at each stage of the iteration, one to say, in effect, "Where are we now?", the other to say "What are we going to do about it?" There is really no need for any output for those iterations that do not elect or exclude any candidate, so immediate output has been cut down to just showing the names of candidates elected or excluded as those events occur, with storage in computer files of enough information to allow various forms of table to be easily produced when wanted.

There is also provision for an animated form of output, showing coloured lines on the screen performing the transfers of votes. This is deliberately slowed down to make it easy to watch.

6 Ties

In the event of a tie, where a candidate must be excluded and two or more are exactly equal in last place, [1] gave only a pseudo-random choice as the solution. In my implementation, I was persuaded by ERS Technical Committee to include the traditional 'ahead at first difference' criterion as a first tie-breaker, with a pseudo-random choice only if that did not solve it.

Strictly speaking this is contrary to Meek's stated principles on which his method is based, and was somewhat against my will, but it is unreasonable to expect to win every argument, and it does no real harm, particularly as ties hardly ever occur in real elections.

The pseudo-random method used is similar except that [1] calculated random numbers only if and when required. I have found it more convenient to assign such numbers to the candidates in the first instance and thus to have them already available if wanted. However I change the assigned numbers at each stage so that, if A is randomly preferred to B on the odd stages, then B is preferred to A on the even stages.

7 Election

In [1] candidates were not deemed elected until the end of an iteration. The keep values having converged, it was then considered whether any additional candidate had achieved the quota. Further thought has shown that it is absolutely safe to elect as soon as a candidate reaches the quota during the iterations and at once to start adjusting that candidate's keep value, along with those of any others already elected. This follows from Woodall's proof, given as part of [1], that if there is a feasible vector, then there is a unique solution vector — see that proof for the definitions of those terms.

8 Convergence

Both in [1] and my present implementation, the overall plan consists of iterations within iterations, the outer iterations being the operations up to and including the exclusion of a candidate, the inner iterations being the successive adjustments of keep values.

In [1] the inner iterations were taken as having converged when each elected candidate's votes were individually close enough to the current quota. This has been simplified to saying that the sum of the current surpluses of all the elected candidates must be no greater than 0.0001. It is almost certain in any case that, if such a small sum of all surpluses is ever reached, the lowest candidates are tied and further iterations would not separate them. Because of the short-cut exclusion rule mentioned below, however, it hardly ever happens that iterations need to proceed so far.

9 Short-cut exclusion rule

During the iterations, if it is found that the lowest candidate's current votes plus the total surplus of the elected candidates is less than the current votes of the next lowest candidate, it is certain that, if the iterations were continued all the way to convergence, that lowest candidate would necessarily still be the lowest and would have to be excluded. It is therefore safe to exclude the candidate at once. The next iterations will then start from a different point than would otherwise have been the case, but it follows from Woodall's proof that the next solution vector will still be the same, so the eventual result must be unchanged.

To see that, in these circumstances, the lowest candidate cannot catch up, it should be noted that the total number of votes remains unchanged and the effect of reducing the keep values of elected candidates is to pass their surplus votes to other candidates or, possibly, to non-transferable. If all the surpluses are passed to the lowest candidate, that candidate would necessarily, given the conditions, remain the lowest. If some are passed to other candidates that is even worse for the lowest, even if some of those candidates become elected.

The only point that needs more thought is to consider what happens if some surplus becomes non-transferable, resulting in a reduction of the quota. If n votes become non-transferable, the extra surplus created thereby is $mn/(s+1)$ where m is the number of elected candidates so far, and s is the number of seats. We know that m is less than s, because otherwise all seats are filled and the whole election is over. Therefore $mn/(s+1)$ is less than n, which shows that the amount that could have gone to the lowest candidate has been reduced.

Similar arguments show that, if two or more lowest candidates have a total number of votes that, together with the current surplus, is less than the votes of the candidate next above, it is safe to exclude them all at once, provided that enough would remain to fill all seats. I have not implemented this (except in the special case where several lowest candidates have zero votes) believing it to be simpler to explain what is going on if only one at a time is excluded.

With traditional style STV it is important that rules are firmly laid down as to whether or not multiple exclusions are to be made, because it can change the result. Thus, for example, Newland and Britton rules [3] insist that multiple exclusions must be made when possible, whereas Church of England rules [4] insist on only one at a time. With Meek rules, however, it is optional, as the result is necessarily the same either way. The fact that I exclude only one at a time is not intended to suggest that there is anything wrong, within a Meek system, with multiple exclusions if others wish to use them.

10 Equality of preference

Meek [2] suggested allowing voters to express equality of preference where desired. In [1] this option was not included. My program does include the option but there are some difficulties involved, as explained in detail in [5]. I continue to hold the conclusion expressed there that "the complications may be too many to be worth it ... [but] the facility is strongly valued by a significant number of electors".

11 Constraints

Not proposed by Meek, the program also allows constraints, whereby a maximum number, or a minimum number, may be laid down for certain categories among those elected. I dislike such constraints in principle [6], but they are necessary in certain circumstances in the Church of England [4] and, if the Church ever wished to update its procedures to use Meek-style STV, it would be necessary to demonstrate that it could cope with this additional complication.

At present the main thing for which constraints may be wanted is the filling of casual vacancies, where this is done by recounting the original votes with the late occupier of the vacant seat withdrawn. The constraint that is then necessary is to disallow exclusion of any existing seat holder.

12 STV in New Zealand

Those working on the introduction of STV for certain elections in New Zealand, having decided that the Meek rules were what they wanted, had my implementation available to them, and most of its details given above, such as the 9-decimal place working, and the figure of 0.0001 for the total surplus to indicate convergence, have been incorporated into their Act of Parliament [7].

There is, of course, no objection to these details having been used, but I hope that it will not become 'folklore' that they must be used and that Meek has not been properly implemented otherwise.

13 Acknowledgements

I thank both the editor and the referee for suggestions that led to substantial improvements in this paper.

14 References

[1] I.D. Hill, B.A. Wichmann and D.R. Woodall. Algorithm 123 – Single Transferable Vote by Meek's method. *Computer Journal*, 30, 277-281, 1987.

[2] B.L. Meek. A new approach to the Single Transferable Vote. *Voting matters*, Issue 1, 1-11, 1994.

[3] R.A. Newland and F.S. Britton. How to conduct an election by the Single Transferable Vote, 2nd edition. Electoral Reform Society, 1976.

[4] GS1327: General Synod. Single Transferable Vote regulations 1998.

[5] I.D. Hill. Difficulties with equality of preference. *Voting matters*, issue 13, 8-9, 2001.

[6] I.D. Hill. STV with constraints. *Voting matters*, issue 9, 2-4, 1998.

[7] New Zealand Government. Local Electoral Regulations 2001. Schedule 1A.

Editorial Postscript

I (Brian Wichmann) also have an implementation of the Meek algorithm, written in Ada 95. I have not made this generally available for several reasons: firstly, it lacks any system for preparing the data and even any adequate diagnostics on incorrect data (and hence is just a counting program); secondly the program has a number of extensions written to aid some investigations (typically reported in *Voting matters*); thirdly the program does not perform the arithmetic exactly correctly. There are a number of small differences between my Ada 95 version and the version in this paper; ties are broken differently and I will exclude several candidates together having the same number of votes provided it is safe to do so.

In 2000, I did perform a check of the Meek implementation described in this paper against the original version published in 1987 [1]. The report of this validation can now be found on the McDougall website. One interesting finding was that a test (M135) was actually a tie between two candidates for exclusion. However, both programs performed slightly different calculations in approximating the solution in such a way that neither reported a tie and the differences in the rounding resulted in a different exclusion. This is not considered a fault as, where there is really a tie, either result is acceptable.