Dropping Lowest Grades

Daniel M. Kane
Massachusetts Institute of Technology
Cambridge, MA, 02139
dankane@mit.edu

Jonathan M. Kane
University of Wisconsin-Whitewater
Whitewater, WI, 53190
kanej@uww.edu

Introduction

Many teachers allow students to drop the lowest score from a sequence of quizzes, tests, or homework assignments. When the number of grades is large, some teachers will even allow students to drop several of their lowest scores. A computer gradebook program would need to implement an algorithm to provide this feature (see one of many examples of computer gradebook software, for example, [4]). A natural criterion to decide which grades to drop would be to drop the set of grades that maximizes the student's final grade. In some circumstances, it can be non-trivial to determine the best grades to drop. Using natural brute force methods, the time needed to find this optimal set of grades to drop can grow exponentially as the number of grades involved grows making these methods impractical even on fast computers. We discuss some unexpected behavior exhibited by this problem and provide a simple and very efficient algorithm for finding the best set of grades to drop.

Grade Dropping

Assume that a teacher has given a sequence of $k > 0$ quizzes and will allow each student to drop r of the quiz grades. Suppose that for $j = 1, 2, 3, \ldots, k$ a

particular student has earned on quiz j a score of m_j points out of a possible n_j points. For simplicity assume earned scores are integers, and possible points are positive integers. Let N be an upper bound for the n_j. We will refer to the set of r grades that are dropped as the *deletion set*, and the set of $k-r$ grades that are not dropped as the *retained set*. The goal is to identify the deletion set which will result in the student receiving the highest possible final grade, the *optimal deletion set*.

If the teacher is only basing the student's final grade on the student's raw score, $\sum_{j=1}^{k} m_j$, then finding the best grades to drop is a simple matter of finding the r smallest m_j values and dropping them. For example, suppose that Alan has earned the quiz scores shown in Table 1.

TABLE 1 : Alan's Quiz scores					
Quiz	1	2	3	4	5
Score	2	6	24	3	6
Possible	8	12	40	4	24
Percentage	25	50	60	75	25

If the teacher wants to drop two quiz scores, this student does the best by dropping quizzes 1 and 4 since those are the two with the smallest number of points assigned, leaving the student with an accumulated quiz total of $6 + 24 + 6 = 36$, the largest possible sum of three scores. Notice that we dropped quiz 4, on which the student scored a higher percentage than on any other quiz.

On the other hand, if the teacher is basing the student's final grade on the ratio of total points earned to the total points possible, then the problem of finding the best set of r scores to drop is far more interesting. What we need is a subset $S \subset K = \{1, 2, 3, \ldots, k\}$ of $k-r$ retained grades so that the ratio $\dfrac{\sum_{j \in S} m_j}{\sum_{j \in S} n_j}$ is maximized. If all the quizzes are worth the same amount, that is, if all of the n_j are equal, then this reduces to finding the r smallest m_j values, just as it was in the above example.

Paradoxical Behavior

Intuitively, one might suspect that a way to obtain an optimal solution would be to drop those quiz grades where the student performed the worst either by obtaining the smallest number of points or by obtaining the smallest percentage grade, $\frac{m_j(100\%)}{n_j}$. However, this is not always the case as the following examples illustrate. Consider Beth's quiz scores shown in Table 2.

TABLE 2: Beth's Quiz Scores			
Quiz	1	2	3
Score	80	20	1
Possible	100	100	20
Percentage	80	20	5

It is clear that Beth performed worst on quiz 3 where she obtained the smallest raw score (1) and the smallest percentage grade (5%). If that grade is dropped, Beth's remaining quiz grades would give a mean score of $(80 + 20)/(100 + 100) = 50\%$. On the other hand, if quiz 2 is dropped instead, she would receive a mean score of $(80 + 1)/(100 + 20) = 67.5\%$. The reason for this is that quiz 3 is not worth very many points, so its impact on the final score is much smaller than that of quiz 2.

One conclusion is clear. As long as the number of grades to drop is smaller than the total number of grades, the optimal retained set of grades will always contain the grade that has the largest percentage score. If more than one grade shares the same largest percentage score, none of those grades will be dropped unless there are more of them than the number of retained grades. For example, with Beth's grades, quiz 1 will not be dropped. The reason for this is that if the retained set S contains any grade whose percentage is not the largest percentage, the average $\frac{\sum_{j\in S} m_j}{\sum_{j\in S} n_j}$ will be less than this largest percentage. S will then contain at least one grade whose percentage is less than or equal to the average of the grades in S. Removing that grade and replacing it with a grade with the largest percentage will raise the average since both the removal and the addition raise the average.

As seen with Beth's grades, the reverse argument does not work. That is, the grade with the smallest percentage score does not necessarily appear in the optimal deletion set. We can conclude that the grade with the smallest percentage will be among the grades retained if we want to get the smallest possible average score. But getting the smallest possible average score is not the goal.

One might hope that the best way to drop a set of r grades can be constructed inductively by finding the best one grade to drop, and then finding the best grade to drop from the remaining grades, and so on. This strategy turns out not to work. Consider Carl's quiz scores shown in Table 3.

TABLE 3: Carl's Quiz Scores				
Quiz	1	2	3	4
Score	100	42	14	3
Possible	100	91	55	38
Percentage	100	46	25	8

If we wish to drop just one grade, then the best score is obtained by dropping quiz 4 yielding an average of $(100 + 42 + 14)/(100 + 91 + 55) = 63.4\%$ as compared to 32.0% for dropping quiz 1, 60.6% for dropping quiz 2, and 63.3% for dropping quiz 3. If we need to drop two scores, it is best to drop quizzes 2 and 3 and retain quiz 4 to get the average $(100+3)/(100+38) = 74.6\%$ as compared to 74.3% for dropping quizzes 3 and 4, 73.5% for dropping quizzes 2 and 4, and 38.4% for dropping quizzes 1 and 4. Notice that the optimal deletion set of two grades does not include the best single grade to drop.

Also surprising is how slight changes to a problem can result in radically different results. To see this, consider Dale's eleven quiz grades displayed in Table 4.

TABLE 4: Dale's Quiz Scores						
Quiz	0	1	2	3	4	5
Score	$20+c$	$21-b_1$	$22-b_2$	$23-b_3$	$24-b_4$	$25-b_5$
Possible	40	42	44	46	48	50
Percentage	$50+\frac{c}{.40}$	$50-\frac{b_1}{.42}$	$50-\frac{b_2}{.44}$	$50-\frac{b_3}{.46}$	$50-\frac{b_4}{.48}$	$50-\frac{b_5}{.50}$
Quiz		6	7	8	9	10
Score		$26-b_6$	$27-b_7$	$28-b_8$	$29-b_9$	$30-b_{10}$
Possible		52	54	56	58	60
Percentage		$50-\frac{b_6}{.52}$	$50-\frac{b_7}{.54}$	$50-\frac{b_8}{.56}$	$50-\frac{b_9}{.58}$	$50-\frac{b_{10}}{.60}$

We consider several examples of Dale's quiz scores where c and each of the b_j are positive integers. Since quiz 0 is the only quiz with percentage over 50%, we would not want to drop quiz 0. If $A \subset \{1,2,3,4,5,6,7,8,9,10\}$ is the set of other quiz grades retained, the resulting average score is

$$\frac{20+c+\frac{\sum_{j\in A} n_j}{2}-\sum_{j\in A} b_j}{40+\sum_{j\in A} n_j} = 0.5+\frac{c-\sum_{j\in A} b_j}{40+\sum_{j\in A} n_j}.$$

First, let us set $c=4$ and each of the $b_j=1$. If we drop five quiz grades, the average score will be $0.5+\frac{4-\sum_{j\in A} b_j}{40+\sum_{j\in A} n_j} = 0.5-\frac{1}{40+\sum_{j\in A} n_j}$. So, to maximize this average, we want A to represent the quizzes with the largest possible values, n_j, in order to make the denominator of the fraction as large as possible. Thus, the optimal deletion set is $\{1,2,3,4,5\}$.

But if we just change the value of c from 4 to 6, the average score becomes $0.5+\frac{6-\sum_{j\in A} b_j}{40+\sum_{j\in A} n_j} = 0.5+\frac{1}{40+\sum_{j\in A} n_j}$. In this case we want A to represent the quizzes with the smallest possible values in order to make the denominator of the fraction as small as possible. Thus, the optimal deletion set is $\{6,7,8,9,10\}$. A slight change in c completely changed the optimal deletion set. Note that if $c=5$, all deletion sets which do not include quiz 0 result in the same average of 50%, so every set A of size five gives the same optimal average.

Next, consider what happens with Dale's quiz scores when $c = 11$ and $b_j = 2$ for each j. If we drop four quiz grades, the set A will have six elements, and the average score will be $0.5 + \frac{11-2\sum_{j\in A} b_j}{40+\sum_{j\in A} n_j} = 0.5 - \frac{1}{40+\sum_{j\in A} n_j}$. To maximize this average, we want to retain the quizzes with the largest possible scores, so the optimal deletion set is $\{1, 2, 3, 4\}$. If, on the other hand, we drop five quiz grades, the set A will have only five elements, and the average score becomes $0.5 + \frac{11-2\sum_{j\in A} b_j}{40+\sum_{j\in A} n_j} = 0.5 + \frac{1}{40+\sum_{j\in A} n_j}$. To maximize this average, we want to retain the quizzes with the smallest possible scores, so the optimal deletion set is $\{6, 7, 8, 9, 10\}$ which has no elements in common with the optimal deletion set when we dropped only four grades.

Finally, Dale's quiz scores can be used to show that the optimal deletion set when dropping four grades can overlap with the optimal deletion set when dropping five grades to whatever extent we like. Indeed, let t represent the number of grades we wish the two optimal deletion sets to have in common where t is one of the numbers 1, 2, 3, or 4. Set $b_j = 3$ for each j from 1 to t, and set $b_j = 2$ for each $j > t$. Let $c = 11$. If we drop four quizzes, and s is the number of retained quizzes which have their $b_j = 3$, the set A will have six elements, and the average score will be $0.5 + \frac{11-\sum_{j\in A} b_j}{40+\sum_{j\in A} n_j} = 0.5 - \frac{11-[s+2(6-s)]}{40+\sum_{j\in A} n_j} = 0.5 - \frac{1+s}{40+\sum_{j\in A} n_j}$. To maximize this average, s needs to be as small as possible (0), and we need to retain the quizzes with the largest possible score. This means the optimal deletion set is $\{1, 2, 3, 4\}$.

Now, if we drop five quiz scores, and s is the number retained quizzes which have their $b_j = 3$, the set A will have five elements, and the average score becomes $0.5 + \frac{11-\sum_{j\in A} b_j}{40+\sum_{j\in A} n_j} = 0.5 + \frac{11-[s+2(5-s)]}{40+\sum_{j\in A} n_j} = 0.5 + \frac{1-s}{40+\sum_{j\in A} n_j}$. To maximize this average, s needs to be 0 or else the numerator $1 - s$ will be less than or equal to zero, and the average will not exceed 50%. Thus, this average will be maximized only when we drop all the quizzes with $b_j = 3$ and retain quizzes with the smallest possible score. This means the optimal deletion set is the set containing the quizzes with $b_j = 3$ and as many of the high numbered quizzes as needed. Thus, the overlap between the optimal

deletion set when dropping four grades and the optimal deletion set when dropping five grades will be exactly the set of t grades with the $b_j = 3$.

Note that it is easy to construct examples similar to Dale's grades which include a very large number of quiz scores that exhibit the same paradoxical behavior as in the examples just given. Even though such examples exist only when the possible scores, the n_j values, are not all the same, paradoxical examples can still be constructed where the n_j values are all very close to each other, for example, within 1 of a fixed value. These examples are reminiscent of Simpson's Paradox (see [6]) which also deals with creating ratios by combining the numerators and denominators of other fractions.

Algorithms for Finding the Optimal Deletion Set

We return to the question of how one can identify the optimal deletion set when we want to drop r grades from a list of k quiz scores. One *brute force algorithm* would have us calculate the average grade for each possible set of $k - r$ retained grades. There are several well-known algorithms for enumerating all such subsets (see virtually any book on combinatorics, for example [1]). The arithmetic for calculating each average grade is straight forward. Unfortunately, even though checking any one average is very fast, the number of average grades which need to be calculated is given by the binomial coefficient $\binom{k}{r}$ which grows at a rate of $\frac{k^r}{r!}$. For small k and r, these calculations pose no problems. However, if a teacher wanted to drop just 10 grades from a list of 100 grades, even on a computer this algorithm would take far too long to be of any practical value.

The examples of the last section suggest that small changes in a problem can result in completely different optimal deletion sets. This indicates that we would run into difficulties by trying to implement either a *greedy algorithm* or a *dynamic programming algorithm*. These standard approaches to developing algorithms attempt to find solutions to problems by constructing an array of solutions to smaller problems which, in our case, have little bearing on the results of the original problem (see [3] for a discussion of how these methods

are used to generate algorithms).

The Optimal Drop Function

Our goal is to find the retained set $S \subset K = \{1, 2, 3, \ldots, k\}$ of size $k - r$ so that the ratio

$$\frac{\sum_{j \in S} m_j}{\sum_{j \in S} n_j} = q \tag{1}$$

is maximized. For each j define $f_j(q) = m_j - qn_j$. Then equation (1) is equivalent to

$$\sum_{j \in S} f_j(q) = 0. \tag{2}$$

Notice the the left-hand side of equation (1) is greater than q if and only if the left-hand side of equation (2) is greater than 0.

Since each $f_j(q)$ is a linear, decreasing function of q, for any given set S, $\sum_{j \in S} f_j(q)$ is also a linear, decreasing function of q. For a particular selection of retained grades, S, the equation $\sum_{j \in S} f_j(q) = 0$ is satisfied by the value of q which represents the average of the quizzes in S. We will have found the optimal set of retained quizzes, S_{best}, when we find the S where the associated average, q_{best}, is as large as possible. Define the *optimal drop function* to be

$$F(q) = \max\{\sum_{j \in S} f_j(q) : S \subseteq K, |S| = k - r\}. \tag{3}$$

Since F is the maximum of a finite number of linear, decreasing functions, it must be a piecewise linear, decreasing, concave up function. Moreover, $F(q_{best}) = 0$ since $\sum_{j \in S_{best}} f_j(q_{best}) = 0$ while for any other $S \subseteq K$ with $|S| = k - r$, it follows that $\sum_{j \in S} f_j(q_{best}) \leq 0$.

Consider, for example, Carl's quiz scores from Table 2 where we drop two of four quizzes. There are six possible sets S and six associated sums shown in Figure 1.

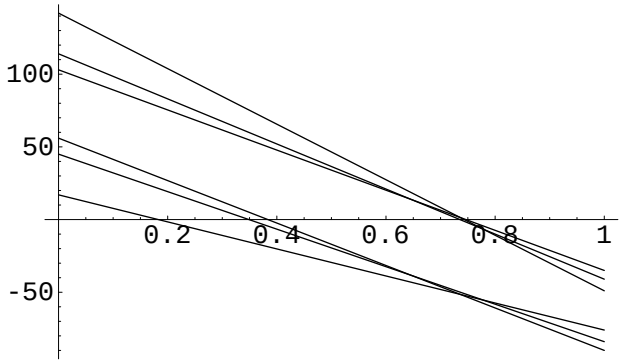

Figure 1: The six possible sums of two f_j

The function F in this case has the graph

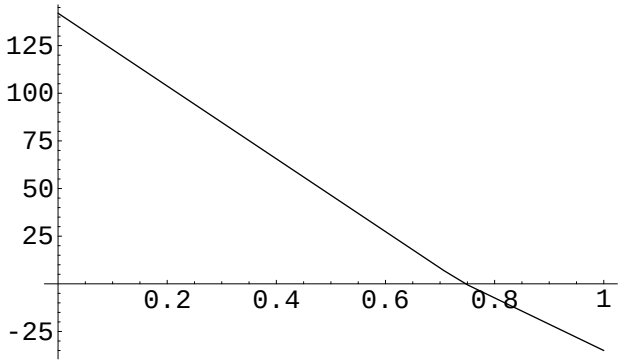

Figure 2: The graph of F

The problem of determining the best set of r grades to drop is now equivalent to finding the subset $S \subset K$ with $|S| = k - r$ and a rational number q, so that $F(q) = \sum_{j \in S} f_j(q) = 0$. The advantage of considering the function F is that it is a simple matter to evaluate $F(q)$ for any given q. Indeed, given a list of k grades $m_1, m_2, \ldots, m_k$ and k maximum possible scores $n_1, n_2, \ldots, n_k$, a number, r, of grades to drop, and a real number q, one merely has to evaluate each $f_j(q) = m_j - qn_j$ for each $j = 1, 2, \ldots, k$. Then one identifies the

$k - r$ largest values among the $f_j(q)$ values. The set S becomes the set of j values corresponding to the largest $f_j(q)$ values. Finally, $F(q)$ is calculated as $\sum_{j \in S} f_j(q)$. Since there are well-known efficient algorithms for identifying the largest values out of a collection of numbers (see virtually any book about data structures or algorithms, for example, [3] or [5]), $F(q)$ can be calculated efficiently.

It remains to find the value of q where $F(q) = 0$. Since for a given S, $\sum_{j \in S} f_j(q)$ is linear, the graph of F can change slope at a value of q only if the associated set S changes at this value of q. For each q we can consider the collection of the k values of $f_j(q)$ for $j = 1, 2, 3, ..., k$. We can order these values in decreasing order. As the value of q changes, the values of the $f_j(q)$ change, and their order changes. Notice that the values of S depend only on the order of the $f_j(q)$, and hence the set S changes only when the order of $f_j(q)$ changes. Since each f_j is a continuous function, the order of $f_i(q)$ and $f_j(q)$ can change only for values of q where $f_i(q) = f_j(q)$. Notice that since each f_j is linear, this occurs at most once for every pair of i and j. Therefore, the set S cannot change at more than $\binom{k}{2}$ values of q since there are only that many pairs of i and j.

The condition $f_i(q) = f_j(q)$ occurs when $m_i - qn_i = m_j - qn_j$, or when $q = \frac{m_i - m_j}{n_i - n_j}$. Thus, if the graph of F changes slope at some value q, q has to be a rational number with denominator bounded by N (recall that N is an upper bound for all the n_i). Since $\frac{\sum_{j \in S_{best}} m_j}{\sum_{j \in S_{best}} n_j} = q_{best}$, q_{best} is a rational number with denominator no larger than $(k - r)N$.

This can be used to find S_{best} and q_{best}. One could identify all the values of q where $f_j(q) = f_i(q)$ for some two values i and j. Then, by evaluating $F(q)$ at each of those points, the function F can be constructed since it is linear between each of those values of q. From this, one can easily find where $F(q) = 0$. But there are more efficient ways to find where $F(q) = 0$.

The Bisection Algorithm

An even more efficient algorithm is obtained by approximating the q where $F(q) = 0$ using the *bisection method* (see virtually any book about numerical analysis, for example, [2]). Since we know that q_{best} must lie in the interval between the minimum and maximum values of $\frac{m_j}{n_j}$, we begin by setting $q_{high} = \max_j \left\{\frac{m_j}{n_j}\right\}$, $q_{low} = \min_j \left\{\frac{m_j}{n_j}\right\}$, and $q_{middle} = \frac{q_{min}+q_{max}}{2}$. Then we calculate $F(q_{middle})$ and its associated set S. If $F(q_{middle}) < 0$, we reset q_{low} to q_{middle}. Otherwise we reset q_{high} to q_{middle}. Finally, we reset q_{middle} to $\frac{q_{low}+q_{high}}{2}$. We repeatedly calculate q_{middle}, $F(q_{middle})$, S, and reset q_{high}, q_{low}, and q_{middle} until $q_{high} - q_{low} < \frac{1}{2(k-r)N^2}$. At that point the value of S is S_{best}. Then, q_{best} can be calculated from S_{best}.

How do we know that this final set S is S_{best}? To answer this, we carefully consider the function F. Recall that F is piecewise linear, decreasing, and concave up. If F is linear in a neighborhood of q_{best}, then the distance between q_{best} and the next q where F changes slope is the distance between a rational number with denominator at most N and a rational number with denominator at most $(k-r)N$, which must be at least $\frac{1}{(k-r)N^2}$. So our approximation to q_{best} must be closer to q_{best} than to this closest point of slope change. Thus, the set S associated with this approximation is S_{best}.

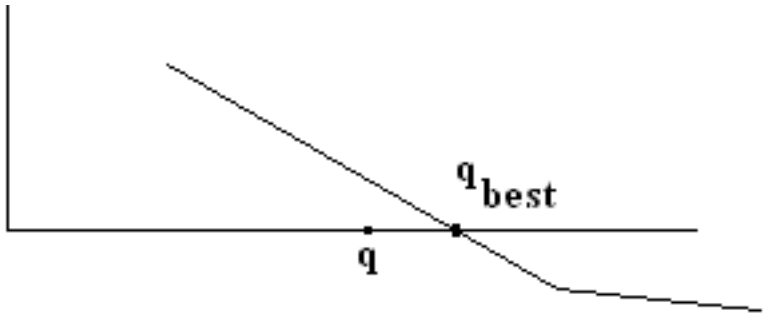

Figure 3: q and q_{best} when F is linear near q_{best}

If F were to change slope at q_{best}, then our approximation to q_{best} would

be associated with one of two different sets of grades where both of these sets are associated with the average q_{best} and, thus, are equally good sets of grades to drop.

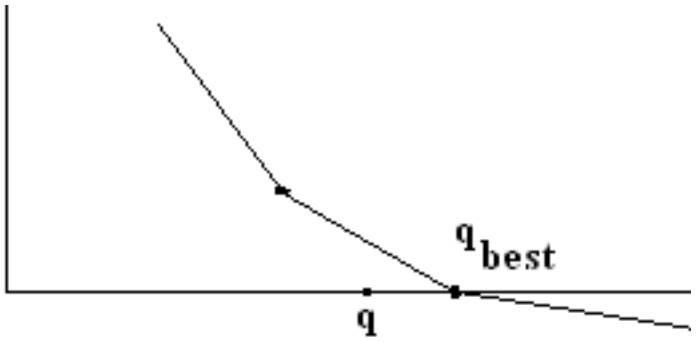

Figure 4: q and q_{best} when F is not linear at q_{best}

A More Efficient Algorithm

An improvement can be found in the bisection algorithm by considering the geometry of the graph of F.

Figure 5 shows several of the linear pieces which form the graph of F. Suppose the value $q_1 < q_{best}$ is chosen at random, and $F(q_1)$ is calculated yielding the associated set S_1 of grades to keep. Consider the linear piece of the graph of F passing through the point $(q_1, F(q_1))$. Let q_2 be the location where this linear piece crosses the x-axis. This q_2 is the average of the grades of S_1. Since the graph of F is concave up, q_2 lies strictly between q_1 and q_{best}. Iterating this process will yield a sequence of q_j which reach q_{best} after finitely many steps. At that point, $F(q)$ will be 0. If the value of q_1 happened to be larger than q_{best}, one iteration of this process will yield a q_2 less than or equal to q_{best}.

Note that the determination of the point where $F(q)$ is 0 poses no problem. Each q_j used in this algorithm will be a rational number. In practice,

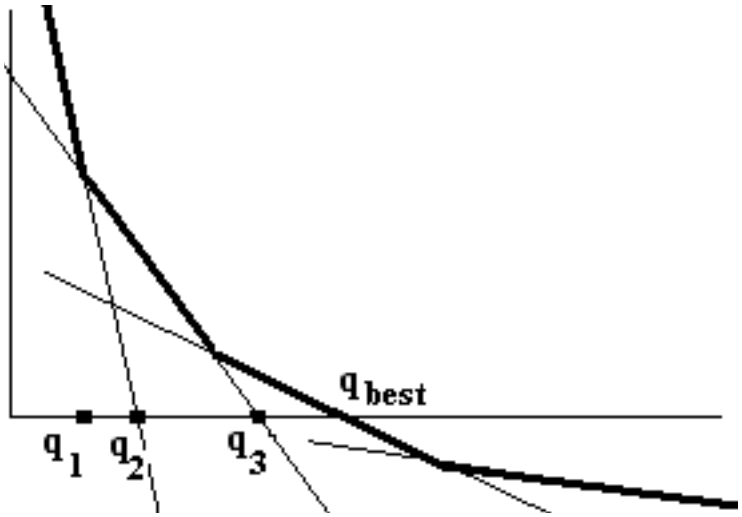

Figure 5: Sequence of q_j's approaching q_{best}

rather than calculating $F(q)$, one would calculate $F(q)$ multiplied by the denominator of q. Doing this allows $F(q)$ to be calculated using integer (fixed point) arithmetic which is not subject to the round-off error and inaccuracy problems common when using real (floating point) arithmetic.

Although it is not clear from the above discussion that this algorithm will run any faster than the bisection method algorithm, extensive running of simulations suggest that the algorithm always converges very rapidly requiring only a very small number of iterations to solve the most complicated problems. For example, we randomly generated many sets of quiz grades each containing 1,000 grades. Using this algorithm to drop 300 of the 1,000 grades, we never found a case where more than five iterations were needed to identify the optimal deletion set. This makes the algorithm particularly well suited for implementation in a computer gradebook program. Why does this algorithm converge so rapidly? Perhaps it is because it is essentially Newton's method applied to the piecewise linear function F.

Acknowledgements

We would like to thank the referees and editor for their helpful suggestions in the preparation of this paper.

References

[1] Richard A. Brualdi, *Introductory Combinatorics* 3rd ed., Prentice-Hall, 1999.

[2] Richard L. Burden & J. Douglas Faires, *Numerical Analysis* 8th ed., Brooks Cole, 2004.

[3] T.H. Cormen, C.E. Leiserson, R.L. Rivest, & C. Stein, *Introduction to Algorithms* 2nd ed., MIT Press, 2001.

[4] Jon Kane, *GRADE GUIDE* ver. 5.5, gradebook software, http://www.gradeguide.com, 2005.

[5] Donald E. Knuth, *The Art of Computer Programming*, Addison-Wesley Reading, 1998.

[6] Ronald Meester, *A Natural Introduction to Probability Theory*, Birkhuser Verlag, 2003.