J. Symbolic Computation (1996) **11**, 1–000

Expression Swell Analysis of Some Matrix Invariants

MICHAEL WESTER

Department of Mathematics and Statistics, University of New Mexico

(Received)

A common problem that occurs when performing exact computations is expression swell, in which the size of expressions involved in a calculation grow dramatically. An important special case of this phenomenon is intermediate expression swell where, during the middle stages of a calculation, intermediate expressions can expand substantially, but the final results of the calculation are comparatively simple. Computing the determinant and the characteristic polynomial of a matrix are good calculations for examining the effects of expression swell, which are very striking, even for small matrices. A number of case studies involving matrices consisting of integer and rational number entries have been performed. In addition, a theoretical 'worst case expression swell arithmetic' has been developed and the results compared with those of the case studies.

A common phenomenon that occurs when performing exact computations is expression swell, in which the size of numbers and expressions involved in a calculation grow dramatically as the calculation progresses. A typical example of this phenomenon is the calculation of the roots of a third or fourth degree univariate polynomial which does not factor over the rational numbers and so the cubic or quartic formula must be used. As a particular example (Grosse and Moler, 1987), the characteristic polynomial of the 9×9 real symmetric Hankel matrix

$$\begin{pmatrix}
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
-1 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 \\
-1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\
1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\
1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\
-1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 \\
-1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 & 1
\end{pmatrix}$$

is given by

$$\lambda^9 + \lambda^8 - 40\lambda^7 - 24\lambda^6 + 240\lambda^5 + 144\lambda^4 ,$$

which can be factored over the rational numbers into

$$\lambda^4(\lambda + 6)(\lambda^4 - 5\lambda^3 - 10\lambda^2 + 36\lambda + 24) .$$

0747–7171/90/000000 + 00 $03.00/0

To complete the solution of the eigenvalue problem, the roots of the last factor must be extracted using the quartic formula. One of the roots is shown in Figure 1. An estimate of the size of this expression can be obtained by counting the number of operators and atomic operands involved in its construction. In this case, using the expression's internal representation in MACSYMA (Bogen et al, 1992), a size of 300 was computed. The other three roots are of the same complexity and also have sizes of 300. Of course, if these roots were to be evaluated numerically, each root would be reduced to a single complex floating point number (which would have a size of 5).

An important special case of the expression swell phenomenon is intermediate expression swell. This refers to a condition where, during the middle stages of a calculation, intermediate expressions can expand substantially, but the final results of the calculation are comparatively simple. A typical example here is the verification of a trigonometric or tensor identity. As an example of the latter, Figure 2 presents the results of MACSYMA computing the left-hand side of the Bianchi identity for a symmetric connection

$$K_j{}^{\ell}{}_{hk|p} + K_j{}^{\ell}{}_{kp|h} + K_j{}^{\ell}{}_{ph|k} \ ,$$

in terms of Christoffel symbols of the second kind. Here, K is the Riemann curvature tensor. This sum contains 72 terms, each of which is a product of 2 or 3 Christoffel symbols, for a total of 180 Christoffel symbols. However, upon simplifying this expression by consistently renaming the dummy indices, the simple result of zero is obtained, which verifies the identity.

Expression swell is a major problem in symbolic mathematical computations. As an expression grows in size, it takes up more and more memory and/or disk space, while also taking more and more time to be manipulated. A shortage of either of these resources can cause a computation to fail, even if the final result is known to be relatively simple. Sometimes a computation can be reorganized so that it uses less resources (or more of one and less of another) and thus succeeds where previously it failed. Sometimes nothing will help but the acquisition of more space (memory, disk, etc.) and/or an increase in the processing speed (bringing the time of a computation down to a reasonable level).

In the past, discussion of expression swell in symbolic mathematical computations has often been anecdotal. In this article, some quantitative results will be presented of the effects of expression swell when computing determinants and characteristic polynomials. These results will be a combination of case studies involving matrices consisting of integer and rational entries and some theoretical worst case (and other) analyses. The latter calculations will be done in terms of a worst case arithmetic which will be developed in detail. Many of the findings are quite striking, even for small matrices.

1. Theoretical Expression Swell Analysis

Since expression swell is such an important problem in symbolic mathematical computations, a very useful ability is to be able to predict the extent of this phenomenon during the course of a particular calculation. One way to do this is to set up an algorithm and run an extensive series of calculations under a variety of initial conditions. An analysis of the results will provide an idea (perhaps a good one) of the progress of expression growth during the calculation. In a subsequent section, a statistical survey of determinant and characteristic polynomial calculations under a limited set of initial conditions is presented. The problem with this approach for quantitatively assessing the expression swell

$$-\frac{\sqrt{-\frac{\left(18\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{2/3}-465\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{1/3}+1856\right)\sqrt{\frac{36\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{2/3}+465\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{1/3}+3712}{\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{1/3}}}+999\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{1/3}}{\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{1/3}\sqrt{\frac{36\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{2/3}+465\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{1/3}+3712}{\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{1/3}}}}}}{6\sqrt{2}}$$

$$-\frac{\sqrt{\frac{36\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{2/3}+465\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{1/3}+3712}{\left(\frac{16\sqrt{8071}\,i}{3}+\frac{25136}{27}\right)^{1/3}}}}{12}+\frac{5}{4}$$

Figure 1. One root of $\lambda^4-5\lambda^3-10\lambda^2+36\lambda+24$.

$$
\begin{aligned}
&-\Gamma_{\#19}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#19}{}_{\#27}\Gamma_{p}{}^{\#27}{}_{k}+\Gamma_{j}{}^{\ell}{}_{h,\#25}\Gamma_{p}{}^{\#25}{}_{k}+\Gamma_{\#19}{}^{\ell}{}_{\#22}\Gamma_{j}{}^{\#19}{}_{h}\Gamma_{p}{}^{\#22}{}_{k}-\Gamma_{j}{}^{\ell}{}_{\#20,h}\Gamma_{p}{}^{\#20}{}_{k}\\
&+\Gamma_{j}{}^{\ell}{}_{\#18,k}\Gamma_{p}{}^{\#18}{}_{h}+\Gamma_{\#10}{}^{\ell}{}_{k}\Gamma_{j}{}^{\#10}{}_{\#16}\Gamma_{p}{}^{\#16}{}_{h}-\Gamma_{\#10}{}^{\ell}{}_{\#12}\Gamma_{j}{}^{\#10}{}_{k}\Gamma_{p}{}^{\#12}{}_{h}-\Gamma_{j}{}^{\ell}{}_{k,\#11}\Gamma_{p}{}^{\#11}{}_{h}\\
&+\Gamma_{j}{}^{\ell}{}_{\#9,h}\Gamma_{k}{}^{\#9}{}_{p}+\Gamma_{\#1}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#1}{}_{\#7}\Gamma_{k}{}^{\#7}{}_{p}-\Gamma_{\#1}{}^{\ell}{}_{\#3}\Gamma_{j}{}^{\#1}{}_{h}\Gamma_{k}{}^{\#3}{}_{p}-\Gamma_{j}{}^{\ell}{}_{h,\#2}\Gamma_{k}{}^{\#2}{}_{p}+\Gamma_{j}{}^{\ell}{}_{p,\#18}\Gamma_{k}{}^{\#18}{}_{h}\\
&+\Gamma_{\#10}{}^{\ell}{}_{\#15}\Gamma_{j}{}^{\#10}{}_{p}\Gamma_{k}{}^{\#15}{}_{h}-\Gamma_{\#10}{}^{\ell}{}_{p}\Gamma_{j}{}^{\#10}{}_{\#13}\Gamma_{k}{}^{\#13}{}_{h}-\Gamma_{j}{}^{\ell}{}_{\#11,p}\Gamma_{k}{}^{\#11}{}_{h}-\Gamma_{h}{}^{\#20}{}_{k}\Gamma_{j}{}^{\ell}{}_{p,\#20}\\
&+\Gamma_{\#9}{}^{\ell}{}_{k,h}\Gamma_{j}{}^{\#9}{}_{p}+\Gamma_{\#1}{}^{\ell}{}_{h}\Gamma_{\#7}{}^{\#1}{}_{k}\Gamma_{j}{}^{\#7}{}_{p}-\Gamma_{\#1}{}^{\ell}{}_{k}\Gamma_{\#4}{}^{\#1}{}_{h}\Gamma_{j}{}^{\#4}{}_{p}+\Gamma_{\#19}{}^{\ell}{}_{h}\Gamma_{\#27}{}^{\#19}{}_{k}\Gamma_{j}{}^{\#27}{}_{p}\\
&+\Gamma_{\#20}{}^{\ell}{}_{k}\Gamma_{j}{}^{\#20}{}_{p,h}-\Gamma_{\#2}{}^{\ell}{}_{h,k}\Gamma_{j}{}^{\#2}{}_{p}+\Gamma_{\#19}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#19}{}_{p,k}-\Gamma_{\#19}{}^{\ell}{}_{\#26}\Gamma_{h}{}^{\#26}{}_{k}\Gamma_{j}{}^{\#19}{}_{p}\\
&+\Gamma_{\#19}{}^{\#26}{}_{h}\Gamma_{\#26}{}^{\ell}{}_{k}\Gamma_{j}{}^{\#19}{}_{p}-\Gamma_{\#19}{}^{\#26}{}_{k}\Gamma_{\#26}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#19}{}_{p}+\Gamma_{\#19}{}^{\ell}{}_{h,k}\Gamma_{j}{}^{\#19}{}_{p}-\Gamma_{\#18}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#18}{}_{p,k}\\
&-\Gamma_{\#10}{}^{\ell}{}_{k}\Gamma_{\#16}{}^{\#10}{}_{h}\Gamma_{j}{}^{\#16}{}_{p}-\Gamma_{\#10}{}^{\ell}{}_{k}\Gamma_{j}{}^{\#10}{}_{p,h}+\Gamma_{\#10}{}^{\#15}{}_{h}\Gamma_{\#15}{}^{\ell}{}_{k}\Gamma_{j}{}^{\#10}{}_{p}-\Gamma_{\#10}{}^{\#15}{}_{k}\Gamma_{\#15}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#10}{}_{p}\\
&-\Gamma_{\#10}{}^{\ell}{}_{k,h}\Gamma_{j}{}^{\#10}{}_{p}+\Gamma_{h}{}^{\#9}{}_{p}\Gamma_{j}{}^{\ell}{}_{k,\#9}-\Gamma_{\#9}{}^{\ell}{}_{p}\Gamma_{j}{}^{\#9}{}_{k,h}-\Gamma_{\#1}{}^{\ell}{}_{h}\Gamma_{\#7}{}^{\#1}{}_{p}\Gamma_{j}{}^{\#7}{}_{k}\\
&-\Gamma_{\#19}{}^{\ell}{}_{h}\Gamma_{\#27}{}^{\#19}{}_{p}\Gamma_{j}{}^{\#27}{}_{k}+\Gamma_{\#25}{}^{\ell}{}_{h,p}\Gamma_{j}{}^{\#25}{}_{k}+\Gamma_{\#19}{}^{\ell}{}_{p}\Gamma_{\#23}{}^{\#19}{}_{h}\Gamma_{j}{}^{\#23}{}_{k}-\Gamma_{\#20}{}^{\ell}{}_{p,h}\Gamma_{j}{}^{\#20}{}_{k}\\
&+\Gamma_{\#10}{}^{\ell}{}_{p}\Gamma_{\#13}{}^{\#10}{}_{h}\Gamma_{j}{}^{\#13}{}_{k}+\Gamma_{\#11}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#11}{}_{k,p}+\Gamma_{\#10}{}^{\ell}{}_{p}\Gamma_{j}{}^{\#10}{}_{k,h}-\Gamma_{\#10}{}^{\#12}{}_{h}\Gamma_{\#12}{}^{\ell}{}_{p}\Gamma_{j}{}^{\#10}{}_{k}\\
&+\Gamma_{\#10}{}^{\#12}{}_{p}\Gamma_{\#12}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#10}{}_{k}+\Gamma_{\#10}{}^{\ell}{}_{p,h}\Gamma_{j}{}^{\#10}{}_{k}-\Gamma_{\#1}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#1}{}_{k,p}+\Gamma_{\#1}{}^{\ell}{}_{\#6}\Gamma_{h}{}^{\#6}{}_{p}\Gamma_{j}{}^{\#1}{}_{k}\\
&-\Gamma_{\#1}{}^{\#6}{}_{h}\Gamma_{\#6}{}^{\ell}{}_{p}\Gamma_{j}{}^{\#1}{}_{k}+\Gamma_{\#1}{}^{\#6}{}_{p}\Gamma_{\#6}{}^{\ell}{}_{h}\Gamma_{j}{}^{\#1}{}_{k}-\Gamma_{\#1}{}^{\ell}{}_{h,p}\Gamma_{j}{}^{\#1}{}_{k}+\Gamma_{\#1}{}^{\ell}{}_{k}\Gamma_{\#4}{}^{\#1}{}_{p}\Gamma_{j}{}^{\#4}{}_{h}\\
&-\Gamma_{\#25}{}^{\ell}{}_{k}\Gamma_{j}{}^{\#25}{}_{h,p}-\Gamma_{\#19}{}^{\ell}{}_{p}\Gamma_{\#23}{}^{\#19}{}_{k}\Gamma_{j}{}^{\#23}{}_{h}+\Gamma_{\#2}{}^{\ell}{}_{p}\Gamma_{j}{}^{\#2}{}_{h,k}-\Gamma_{\#19}{}^{\ell}{}_{p}\Gamma_{j}{}^{\#19}{}_{h,k}\\
&+\Gamma_{\#19}{}^{\#22}{}_{k}\Gamma_{\#22}{}^{\ell}{}_{p}\Gamma_{j}{}^{\#19}{}_{h}-\Gamma_{\#19}{}^{\#22}{}_{p}\Gamma_{\#22}{}^{\ell}{}_{k}\Gamma_{j}{}^{\#19}{}_{h}-\Gamma_{\#19}{}^{\ell}{}_{p,k}\Gamma_{j}{}^{\#19}{}_{h}+\Gamma_{\#18}{}^{\ell}{}_{p,k}\Gamma_{j}{}^{\#18}{}_{h}\\
&+\Gamma_{\#10}{}^{\ell}{}_{k}\Gamma_{\#16}{}^{\#10}{}_{p}\Gamma_{j}{}^{\#16}{}_{h}-\Gamma_{\#10}{}^{\ell}{}_{p}\Gamma_{\#13}{}^{\#10}{}_{k}\Gamma_{j}{}^{\#13}{}_{h}-\Gamma_{\#11}{}^{\ell}{}_{k,p}\Gamma_{j}{}^{\#11}{}_{h}+\Gamma_{\#1}{}^{\ell}{}_{k}\Gamma_{j}{}^{\#1}{}_{h,p}\\
&+\Gamma_{\#1}{}^{\#3}{}_{k}\Gamma_{\#3}{}^{\ell}{}_{p}\Gamma_{j}{}^{\#1}{}_{h}-\Gamma_{\#1}{}^{\#3}{}_{p}\Gamma_{\#3}{}^{\ell}{}_{k}\Gamma_{j}{}^{\#1}{}_{h}+\Gamma_{\#1}{}^{\ell}{}_{k,p}\Gamma_{j}{}^{\#1}{}_{h}-\Gamma_{\#1}{}^{\ell}{}_{k}\Gamma_{h}{}^{\#4}{}_{p}\Gamma_{j}{}^{\#1}{}_{\#4}\\
&+\Gamma_{h}{}^{\#25}{}_{k}\Gamma_{j}{}^{\ell}{}_{\#25,p}+\Gamma_{\#19}{}^{\ell}{}_{p}\Gamma_{h}{}^{\#23}{}_{k}\Gamma_{j}{}^{\#19}{}_{\#23}-\Gamma_{h}{}^{\#2}{}_{p}\Gamma_{j}{}^{\ell}{}_{\#2,k}
\end{aligned}
$$

Figure 2. Left-hand side of the Bianchi identity for a symmetric connection expanded in terms of Christoffel symbols of the second kind.

inherent in a given calculation is that typically, a large number of sample calculations need to be made, which can be an expensive and time consuming proposition.

Generally, a better way of charting the course of expression growth is to make some kind of theoretical prediction. A common type of theoretical estimate is a bounding calculation. For an expression swell analysis of an algorithm, there are two important kinds of bounding calculations that can be performed. One involves worst case behavior in which the expression size is maximized for the outcome of each mathematical operation (the results of an addition, of applying a function to its arguments, of applying an operator to a function, etc.), while the other involves best case behavior in which the expression size is minimized for the outcome of each mathematical operation. In general, determining worst or best case behavior for expression swell of a general operation on general operands is extremely difficult. Even just the notion of expression size is not clearly defined. How is the size of an expression to be judged? Does it mean the number of terms in the expression, the number of operators and atomic operands, the number of bytes in some representation of the expression, or some other measure? Or perhaps, different measures are appropriate at different times.

Matters are considerably simplified if only the expression swell analysis of algorithms involving infinite precision integer and rational number expressions is considered. In these cases, expression size can be well-defined. For an integer, the number of digits in its base β ($\beta \in \mathbf{Z}^{\dagger} \geq 2$) representation is a good indicator of its size. (One could also take the absolute value of the number itself to represent its size, but this definition does not provide enough generality to be useful—it is no easier nor much more informative to do an analysis using this definition than it is to just do the calculation with the original integers.) This definition of the size of an integer is essentially a logarithmic measure. Indeed, the number of base β digits comprising an integer n, $\mathcal{N}_\beta(n)$, can alternatively be defined by

$$\mathcal{N}_\beta(n) \equiv \begin{cases} \lfloor \log_\beta |n| \rfloor + 1, & n \neq 0 \\ 0, & n = 0 \end{cases} \tag{1.1}$$

where $\lfloor\ \rfloor$ denotes the floor function. (In practice, the common case of $\mathcal{N}_{10}(n)$ will be computed directly by actually counting the number of decimal digits in n.) Similarly, the size of a rational number (which need not be in lowest terms) can be defined as the sum of the sizes of its numerator and denominator (or possibly as the maximum of these two values). For example, the size (base 10) of $-\frac{327}{1955}$ would be $\mathcal{N}_{10}(-327) + \mathcal{N}_{10}(1955) = 3 + 4 = 7$.

In order to deal only with integer and rational number expressions in an algorithm, the operations permitted on these quantities must be restricted as well. In particular, the expression swell analysis of an algorithm with solely integer inputs can proceed in a simple manner only if all the operations in the algorithm are integer or at least rational number preserving. Therefore, confining the scope of algorithms studied to only those that just use rational arithmetic operations (except perhaps at the last stage, such as when computing a polynomial from its coefficients) will make expression swell analysis manageable for those problems with such algorithms. A large number of the calculations from linear algebra fall into this category.

In the following subsections, an expression swell arithmetic is developed. This arithmetic will operate on numbers that represent the number of digits in classes of integers and rational numbers. Integer preserving operations will be treated separately from rational number preserving ones. This division is made since exact rational number arithmetic exhibits a more complex behavior than exact integer arithmetic. After developing this expression swell arithmetic and discussing an implementation of it in MACSYMA, examples of its use and comparisons of its predictions with the results of actual calculations will be presented in later sections.

1.1. INTEGER CALCULATIONS

Here, a worst case expression swell arithmetic will be developed for certain integer preserving operations. These include the rational arithmetic operations of addition, negation, multiplication, exact division and exponentiation to a nonnegative integer power. Also included are absolute values, exact square roots and greatest common divisors (GCDs). Precisely, worst case behavior under an integer preserving operation means that the result will contain the greatest number of digits that are possible (GCDs are a special case and will be discussed below).

† $\mathbf{Z}$ denotes the set of integers.

To begin this development, note that $\mathcal{N}_\beta$ partitions the integers into an infinite set of equivalence classes. Each equivalence class can be designated by a nonnegative integer: $\bar{0}, \bar{1}, \bar{2}, \ldots$ Thus, $\overline{n}$ will represent the set of integers that have exactly n base β digits. For example, $\bar{1} = \{-(\beta-1), \ldots, -1, 1, \ldots, \beta-1\}$ (as a special case, $\bar{0} = \{0\}$). The intent behind classifying the integers in this way is to permit the analysis of algorithms where the inputs have specified numbers of digits (are members of specified equivalence classes). The analyses will produce upper bounds on the number of digits in the results for any representatives chosen from the respective equivalence classes that are used as inputs to the algorithms.

Operations on the equivalence classes defined by $\mathcal{N}_\beta$ can yield one of two types of results. An operation can produce an integer value. An obvious example is $\mathcal{N}_\beta(\overline{n}) \equiv n$. That is, the number of digits in any representative of $\overline{n}$ is no greater than n (actually, is exactly n in this simple usage). This defines the operation of $\mathcal{N}_\beta$ on an equivalence class. The other possibility is that an operation on one or more equivalence classes will itself yield an equivalence class. For example, suppose

$$f(\overline{n}_1, \overline{n}_2, \ldots) \equiv \overline{g(n_1, n_2, \ldots)} \, .$$

Then, for worst case expression swell behavior, this will mean that $g(n_1, n_2, \ldots)$ will bound the number of digits in $f(n_1', n_2', \ldots)$ over all possible choices of $n_1', n_2', \ldots$ such that $\mathcal{N}_\beta(n_1') = n_1, \mathcal{N}_\beta(n_2') = n_2, \ldots$ Another way of writing this is

$$\mathcal{N}_\beta(f(\overline{n}_1, \overline{n}_2, \ldots)) \le \mathcal{N}_\beta(\overline{g(n_1, n_2, \ldots)}) = g(n_1, n_2, \ldots) \, . \tag{1.2}$$

For example, if $\beta = 10$, multiplying the equivalence class $\bar{2}$ with itself (i.e., multiplying together all possible pairs of 2-digit integers) will produce a set of 3 and 4-digit integers (ranging from $\pm 10 \cdot \pm 10 = \pm 100$ to $\pm 99 \cdot \pm 99 = \pm 9801$). Therefore, the product of $\bar{2}$ with itself will be defined to be $\bar{4}$ since no element of the product set will contain more than 4 digits. Note that the result could have been defined as $\bar{5}$ or $\bar{6}$ or $\ldots$, but these choices would not have been as good since they do not yield as strict a bound on the maximum number of digits as does $\bar{4}$.

The best choice for g in the above formulation will be a function that actually takes on the value of $\mathcal{N}_\beta(f(\overline{n}_1, \overline{n}_2, \ldots))$ [i.e., that maximizes the number of digits in $f(n_1', n_2', \ldots)$, where $\mathcal{N}_\beta(n_i') = n_i$ for $i = 1, 2, \ldots$]. For unary and binary operations, this is not difficult. Consider the unary operations first.

THEOREM 1.1. *The unary operations of integer absolute value and negation, when applied to the equivalence class $\overline{n}$, possess optimal bounding functions given by*

(i) $|\overline{n}| = \overline{n}$,
(ii) $\ominus \overline{n} = \overline{n}$.

This says that taking the absolute value or the negation of an integer does not change the number of digits it contains. The $\ominus$ is a "maximal expression swell" operator. This notation has been introduced to emphasize the distinction between worst case expression swell arithmetic and traditional integer arithmetic.

THEOREM 1.2. *The binary operations of integer addition, subtraction, multiplication and exact division, when applied to the equivalence classes $\overline{n}_1$ and $\overline{n}_2$, where $n_1 > 0$ and $n_2 > 0$, possess optimal bounding functions given by*

$$
\begin{array}{ll}
(i) & \overline{n}_1 \oplus \overline{n}_2 = \overline{\max(n_1,n_2)+1}\ , \\
(ii) & \overline{n}_1 \ominus \overline{n}_2 = \overline{\max(n_1,n_2)+1}\ , \\
(iii) & \overline{n}_1 \odot \overline{n}_2 = \begin{cases} \overline{n_1+n_2} & (n_1 > 1 \textit{ and } n_2 > 1 \textit{ for } \beta = 2) \\ \overline{\max(n_1,n_2)} & (\beta = 2 \textit{ and } \min(n_1,n_2) = 1), \end{cases} \\
(iv) & \overline{n}_1 \overset{\text{exact}}{\oplus\!\!\!\!\div} \overline{n}_2 = \overline{n_1 - n_2 + 1} \quad (n_1 \geq n_2).
\end{array}
$$

If $n_1 = 0$ then the right-hand sides of (i) and (ii) become $\overline{n}_2$ and the right-hand sides of (iii) and (iv) become $\bar{0}$. If $n_2 = 0$ then the right-hand sides of (i)–(iv) are respectively, $\overline{n}_1, \overline{n}_1, \bar{0}$ and undefined.

Proof. Worst case expression swell addition and subtraction are equivalent since in the worst case for regular subtraction, the two operands will be of opposite signs and so it will really be an addition of two terms of the same sign, which is the worst case for regular addition. Symbolically, this can be stated

$$\overline{n}_1 \ominus \overline{n}_2 = \overline{n}_1 \oplus (\ominus \overline{n}_2) = \overline{n}_1 \oplus \overline{n}_2\ .$$

The special cases involving $\bar{0}$ are straightforward, so assume $n_1 > 0$ and $n_2 > 0$. The worst cases for addition, multiplication and exact division can be achieved using all positive operands, so here $\overline{n}$ can be represented by the set of integers $\{\beta^{n-1}, \ldots, \beta^n - 1\}$ with $\beta \geq 2$. To demonstrate that $\overline{g(n_1,n_2)}$ optimally bounds $f(\overline{n}_1, \overline{n}_2)$, it is sufficient to show that the following relationship holds:

$$\beta^{g(n_1,n_2)-1} \leq f(n_1^*, n_2^*) \leq \beta^{g(n_1,n_2)} - 1\ .$$

n_1^* and n_2^* are chosen so as to maximize $f(n_1', n_2')$, where $\mathcal{N}_\beta(n_1') = n_1$ and $\mathcal{N}_\beta(n_2') = n_2$. The right inequality expresses that $g(n_1,n_2)$ is indeed an upper bound of $\mathcal{N}_\beta(f(\overline{n}_1, \overline{n}_2))$, while the left inequality indicates that the bound is optimal so that $\mathcal{N}_\beta(f(\overline{n}_1, \overline{n}_2)) = g(n_1,n_2)$.

In the worst case for addition, n_1^* and n_2^* will be maximal, i.e., $n_1^* = \beta^{n_1} - 1$ and $n_2^* = \beta^{n_2} - 1$. Now,

$$
\begin{array}{rcl}
(\beta^{n_1} - 1) + (\beta^{n_2} - 1) & = & (\beta^{\max(n_1,n_2)} - 1) + (\beta^{\min(n_1,n_2)} - 1) \\
& \leq & 2(\beta^{\max(n_1,n_2)} - 1) \leq \beta(\beta^{\max(n_1,n_2)} - 1) \\
& < & \beta^{\max(n_1,n_2)+1} - 1\ ,
\end{array}
$$

so $\max(n_1,n_2)+1$ bounds $\mathcal{N}_\beta(\overline{n}_1 \oplus \overline{n}_2)$. Moreover,

$$(\beta^{\max(n_1,n_2)} - 1) + (\beta^{\min(n_1,n_2)} - 1) \geq \beta^{\max(n_1,n_2)}$$

since $\beta^{\min(n_1,n_2)} \geq 2$, thus this bound is optimal.

n_1^* and n_2^* will be defined similarly for worst case multiplication. Therefore,

$$(\beta^{n_1} - 1)(\beta^{n_2} - 1) = \beta^{n_1+n_2} - (\beta^{n_1} + \beta^{n_2}) + 1 < \beta^{n_1+n_2} - 1$$

shows that $\mathcal{N}_\beta(\overline{n}_1 \odot \overline{n}_2)$ is bounded by $n_1 + n_2$. If

$$\beta^{n_1+n_2} - (\beta^{n_1} + \beta^{n_2}) + 1 \geq \beta^{n_1+n_2-1}$$

or

$$(\beta - 1)\beta^{n_1+n_2-1} \geq \beta^{n_1} + \beta^{n_2} - 1\ , \tag{1.3}$$

then this bound is optimal. Note that

$$\beta^{n_1} + \beta^{n_2} - 1 = \beta^{\max(n_1,n_2)} + \beta^{\min(n_1,n_2)} - 1 < 2\beta^{\max(n_1,n_2)} . \tag{1.4}$$

Therefore, for $\beta \geq 3$,

$$2\beta^{\max(n_1,n_2)} \leq 2\beta^{n_1+n_2-1} \leq (\beta - 1)\beta^{n_1+n_2-1}$$

will be valid and (1.3) will hold as long as

$$\max(n_1, n_2) \leq n_1 + n_2 - 1 = \max(n_1, n_2) + \min(n_1, n_2) - 1$$

or $\min(n_1, n_2) \geq 1$, which is true by assumption.

For $\beta = 2$ [using (1.4) for the first inequality],

$$2^{n_1} + 2^{n_2} - 1 < 2^{\max(n_1,n_2)+1} \leq 2^{n_1+n_2-1}$$

will be correct and (1.3) will be satisfied whenever

$$\max(n_1, n_2) + 1 \leq n_1 + n_2 - 1 = \max(n_1, n_2) + \min(n_1, n_2) - 1$$

or $\min(n_1, n_2) \geq 2$. Note that $2^1 - 1 = 1$, hence

$$(2^{n_1} - 1)(2^{n_2} - 1) = (2^{\max(n_1,n_2)} - 1)(2^{\min(n_1,n_2)} - 1) = 2^{\max(n_1,n_2)} - 1$$

for the special situation when $\min(n_1, n_2) = 1$.

Finally, the worst case for exact division (with $n_1 \geq n_2$) occurs when n_1^* is as large as possible and n_2^* is as small as possible, so choose $n_1^* = \beta^{n_1} - 1$ and $n_2^* = \beta^{n_2-1}$. Now,

$$\frac{\beta^{n_1} - 1}{\beta^{n_2-1}} < \frac{\beta^{n_1}}{\beta^{n_2-1}} = \beta^{n_1-n_2+1}$$

and since the result of the division must be an integer, this implies that

$$\frac{\beta^{n_1} - 1}{\beta^{n_2-1}} \leq \beta^{n_1-n_2+1} - 1 ,$$

and so $n_1 - n_2 + 1$ bounds $\mathcal{N}_\beta(\overline{n}_1 \overset{\text{exact}}{\odiv} \overline{n}_2)$. Furthermore, this bound is optimal as can be seen from the following sequence of equivalent inequalities:

$$\frac{\beta^{n_1} - 1}{\beta^{n_2-1}} \geq \beta^{n_1-n_2} \Longleftrightarrow \beta^{n_1} - 1 \geq \beta^{n_1-1} \Longleftrightarrow (\beta - 1)\beta^{n_1-1} \geq 1 .$$

(Both of the factors on the left-hand side of the last inequality are ≥ 1 since $\beta \geq 2$ and n_1 is at least 1.) $\square$

To make the ideas presented in the above theorem more concrete, it is helpful to consider some decimal examples. For instance, the worst case of adding an n_1-digit number to an n_2-digit number occurs when all the digits in the larger number are 9's. Then adding a smaller or equal sized number of the same sign will at worst (in this particular case, will always) produce a result with one more digit than the original integer. For example, the validity of the assertion $\bar{4} \oplus \bar{3} = \bar{5}$ can be seen by looking at a worst case calculation like $9999 + 999 = 10998$. For multiplication, the worst case of expression size growth can be exhibited when both numbers consist of all 9's. In this case, the multiplication becomes $(10^{n_1} - 1)(10^{n_2} - 1) = 10^{n_1+n_2} - (10^{n_1} + 10^{n_2}) + 1$. The two middle terms will always bring the final number of digits in the result down to $n_1 + n_2$. Thus, the worst case $9999 \cdot 999 = 9989001$ establishes the validity of writing $\bar{4} \odot \bar{3} = \bar{7}$. Finally, as an example of worst case exact division, it is easy to demonstrate that $\bar{4} \overset{\text{exact}}{\oplus\!\!\!\!\div} \bar{3} = \bar{2}$ by noting that the worst situations are $9900 \div 100 = 99$ and $9999 \div 101 = 99$.

Like their integer arithmetic counterparts, worst case expression swell addition and multiplication are commutative. However, the binary operation $\bigoplus$, at least, is not associative, unlike ordinary addition. A simple example shows this. $(\bar{1} \bigoplus \bar{2}) \bigoplus \bar{3} = \bar{3} \bigoplus \bar{3} = \bar{4}$ while $\bar{1} \bigoplus (\bar{2} \bigoplus \bar{3}) = \bar{1} \bigoplus \bar{4} = \bar{5}$. Remember that the motivation behind developing an expression swell arithmetic is to establish bounds on the rate of expression growth. Therefore, if one way of ordering operations produces a tighter bound on expression swell than other methods, then this way is to be preferred. The above example suggests that the optimal arrangement for adding equivalence classes is to have them ordered in terms of increasing size and then proceed from left to right [i.e., add $\overline{n}_1, \overline{n}_2, \overline{n}_3$, where $n_1 \leq n_2 \leq n_3$, by $(\overline{n}_1 \bigoplus \overline{n}_2) \bigoplus \overline{n}_3$]. $\bigodot$ is associative as can be easily seen and so needs no further elaboration.

$\bigoplus$ and $\bigodot$ can also be thought of as n-ary operators (operators that can be applied to an indefinite number of operands). $\bigodot$ can be easily extended to handle an arbitrary number of factors due to its associativity when considered as a binary operator. Extending $\bigoplus$ is more difficult; however, some properties can be stated.

THEOREM 1.3. *The n-ary operations of integer multiplication and addition, when applied to the equivalence classes $\overline{n}_1, \ldots, \overline{n}_k$ and $\overline{n}$, respectively, where $n_i > 0$ for all i and $n > 0$, possess bounding functions given by*

$$\begin{array}{lll} (i) & \overline{n}_1 \bigodot \overline{n}_2 \bigodot \cdots \bigodot \overline{n}_k = \overline{n_1 + n_2 + \cdots + n_k} & (k > 0), \\ (ii) & k \bigodot \overline{n} \equiv \underbrace{\overline{n} \bigoplus \overline{n} \bigoplus \cdots \bigoplus \overline{n}}_{k \text{ terms}} = \overline{n + \mathcal{N}_\beta(k-1)} & (k > 0). \end{array}$$

If any of the $n_i = 0$ then the right-hand side of (i) becomes $\bar{0}$. If $n = 0$ then the right-hand side of (ii) is $\bar{0}$.

PROOF. Again, the special cases involving $\bar{0}$ are clear, so assume all the $n_i > 0$ and $n > 0$. The first bound can be demonstrated using already established properties of worst case expression swell arithmetic. Thus,

$$\overline{n}_1 \bigodot \overline{n}_2 \bigodot \overline{n}_3 = (\overline{n}_1 \bigodot \overline{n}_2) \bigodot \overline{n}_3 = \overline{n_1 + n_2} \bigodot \overline{n}_3 = \overline{n_1 + n_2 + n_3}$$

is the first inductive step of a sequence that leads to formula (i).

Now, the left-hand side of (ii) defines a shorthand notation for the n-ary sum that follows. The bound on the right can then be obtained by appealing to the definition of $\mathcal{N}_\beta$ and the method implied in (1.2), noting that the actual worst case representative of $k \bigodot \overline{n}$ is given by $k(\beta^n - 1)$. First, observe that

$$\begin{aligned} \mathcal{N}_\beta(k[\beta^n - 1]) &= \lfloor \log_\beta(k[\beta^n - 1]) \rfloor + 1 = \lfloor \log_\beta k + \log_\beta(\beta^n - 1) \rfloor + 1 \\ &\leq \lfloor \log_\beta k + \log_\beta \beta^n \rfloor + 1 = n + (\lfloor \log_\beta k \rfloor + 1) \\ &= n + \mathcal{N}_\beta(k) \ . \end{aligned}$$

To show the result given in (ii), note that $\mathcal{N}_\beta(k) = \mathcal{N}_\beta(k-1)$ except when $k = \beta^m$ $(m \in \mathbf{Z} \geq 0)$. Therefore,

$$\mathcal{N}_\beta(k[\beta^n - 1]) \leq n + \mathcal{N}_\beta(k-1)$$

Table 1. Actual versus theoretical worst case for base 10 n-ary addition.

k	$k \cdot 99$	$\mathcal{N}_{10}(k \cdot 99)$	$\mathcal{N}_{10}(k \odot \bar{2})$
1	99	2	2
2	198	3	3
10	990	3	3
11	1089	4	4
100	9900	4	4
101	9999	4	5
102	10098	5	5

is verified for all $k \neq \beta^m$. The situation for $k = \beta^m$ is

$$\begin{aligned} \mathcal{N}_\beta(\beta^m[\beta^n - 1]) &= \lfloor \log_\beta(\beta^m[\beta^n - 1])\rfloor + 1 = \lfloor \log_\beta \beta^m + \log_\beta(\beta^n - 1)\rfloor + 1 \\ &= m + (\lfloor \log_\beta(\beta^n - 1)\rfloor + 1) = m + \mathcal{N}_\beta(\beta^n - 1) \\ &= m + n = n + \mathcal{N}_\beta(\beta^m - 1) , \end{aligned}$$

which completes the proof. □

To see that the bound on the n-ary sum is not optimal, consider what happens in the worst case for decimal n-digit numbers. For instance, suppose $n = 2$ then Table 1 shows the results of actual worst case sums for various values of k as well as, in the last column, the theoretical quantities $\mathcal{N}_{10}(k \odot \bar{2})$. Notice that eventually $\mathcal{N}_{10}(k \odot \bar{2})$ will occasionally exceed the actual worst case, but this occurs only for relatively large values of k.

The value of k for which the bound given above first exceeds the actual number of digits of the worst case can be computed in general. $n + \mathcal{N}_\beta(k - 1)$ increments by one whenever $k = \beta^m + 1$ for $m \in \mathbf{Z} \geq 0$. At this value, the bound becomes

$$\mathcal{N}_\beta([\beta^m + 1] \odot \overline{n}) = n + \mathcal{N}_\beta(\beta^m) = n + m + 1 .$$

The corresponding actual worst case is given by

$$(\beta^m + 1)(\beta^n - 1) = \beta^{m+n} + \beta^n - \beta^m - 1 = (\beta^{m+n} - 1) + (\beta^n - \beta^m) .$$

If $m < n$ then this will be an integer of $n + m + 1$ (base β) digits. When $m = n$ (and $k = \beta^n + 1$), however, the number of digits will drop to $n + m$ and so will be overpredicted (by one) by the bound calculated above.

The n-ary $\bigoplus$ operator introduced above is distinct from the binary version, although both versions do produce the same result of $\overline{n+1}$ for the common case of $\overline{n} \oplus \overline{n}$. For example, using n-ary $\bigoplus$, $\overline{n} \bigoplus \overline{n} \bigoplus \overline{n} = \overline{n+1}$ ($\beta \geq 3$), while under binary $\oplus$, this becomes $(\overline{n} \oplus \overline{n}) \oplus \overline{n} = \overline{n+1} \oplus \overline{n} = \overline{n+2}$. Again, the tighter bound is to be preferred when doing expression swell arithmetic, so when adding like terms, the n-ary operator will always be used. The infix representation of n-ary $\bigoplus$ is somewhat misleading, hence the alternative notation of $k \odot \overline{n}$ for a sum of k like terms will be adopted whenever possible. This notation has the property that $(k_1 \odot \overline{n}) \oplus (k_2 \odot \overline{n}) = (k_1 + k_2) \odot \overline{n}$ since the parenthesized terms on the left are really sums and for estimating bounds, it is best to make the whole expression into a single n-ary sum.

Whole number exponentiation can be defined in terms of n-ary multiplication. The bounds will be a special case of those derived above.

COROLLARY 1.4. *The binary operation of integer exponentiation to a nonnegative integer power, when applied to the equivalence class* $\overline{n}$, *possesses a bounding function given by*

(i) $\overline{n} \circledast 0 = \overline{1} \quad (n > 0)$,
(ii) $\overline{n} \circledast k \equiv \underbrace{\overline{n} \odot \overline{n} \odot \cdots \odot \overline{n}}_{k \text{ factors}} = \overline{kn} \quad (k > 0)$.

If $n = 0$ *then the right-hand side of (i) becomes undefined.*

Like n-ary addition, the results of expression swell exponentiation will also eventually exceed the actual worst case, but again, only for relatively large values of k. In particular, suppose $n > 0$ and consider

$$\begin{aligned}\mathcal{N}_\beta([\beta^n - 1]^k) &= \lfloor \log_\beta(\beta^n - 1)^k \rfloor + 1 = \left\lfloor \log_\beta \beta^{nk} \left(1 - \frac{1}{\beta^n}\right)^k \right\rfloor + 1 \\ &= \left\lfloor \log_\beta \beta^{nk} + \log_\beta \left(1 - \frac{1}{\beta^n}\right)^k \right\rfloor + 1 = nk + 1 + \left\lfloor \log_\beta \left(1 - \frac{1}{\beta^n}\right)^k \right\rfloor .\end{aligned}$$

The term on the far right of the last expression will be no greater than -1. Moreover, if

$$-(\ell + 1) \leq \log_\beta \left(1 - \frac{1}{\beta^n}\right)^k < -\ell$$

for $\ell \in \mathbf{Z} \geq 0$, then the theoretical prediction, kn, will exceed the actual result by exactly ℓ digits. The right inequality can be transformed into the following equivalent relationships:

$$\left(1 - \frac{1}{\beta^n}\right)^k < \beta^{-\ell} \iff k \ln \left(1 - \frac{1}{\beta^n}\right) < -\ell \ln \beta \iff -k \ln \left(1 - \frac{1}{\beta^n}\right) > \ell \ln \beta .$$

Since $\beta^n \geq 2$, this last inequality has the Taylor expansion

$$-k \left[-\frac{1}{\beta^n} - O\left(\frac{1}{\beta^{2n}}\right) \right] > \ell \ln \beta \qquad \text{or} \qquad k > \ell \beta^n \ln \beta - O\left(\frac{1}{\beta^n}\right) .$$

Thus, the actual worst case number of digits will diverge from the theoretical bound starting at $k \approx \beta^n \ln \beta$. Table 2 shows results for $\beta = 10$. The values in the last column converge quite clearly to $\ln 10 = 2.30258\ldots$

Two more worst case expression swell operations need to be defined to complete the theoretical framework which will allow algorithms involving integer arithmetic to be analyzed.

THEOREM 1.5. *The unary operation of exact integer square root, when applied to the equivalence class* $\overline{n}$, *possesses an optimal bounding function given by*

$$\sqrt{\overline{n}} = \overline{\left\lceil \frac{n}{2} \right\rceil} = \begin{cases} \overline{\left(\frac{n}{2}\right)} & (n \text{ even}) \\ \overline{\left(\frac{n+1}{2}\right)} & (n \text{ odd}), \end{cases}$$

where $\lceil\ \rceil$ *denotes the ceiling function.*

Table 2. Actual versus theoretical worst case for base 10 exponentiation. k is chosen to be the first k such that $\mathcal{N}_{10}([10^n-1]^k) < \mathcal{N}_{10}(\overline{n} \circledcirc k) = kn$.

n	k	$\mathcal{N}_{10}([10^n-1]^k)$	$\mathcal{N}_{10}(\overline{n} \circledcirc k)$	$k/10^n$
1	22	21	22	2.2000
2	230	459	460	2.3000
3	2302	6905	6906	2.3020
4	23025	92099	92100	2.3025

PROOF. Consider the even case first. The demonstration for $n = 0$ is clear so assume $n \geq 2$. The worst case for taking a square root of an n-digit number occurs for $n^* = \beta^n - 1$. Now, $\sqrt{\beta^n - 1} < \sqrt{\beta^n} = \beta^{n/2}$, which implies that $\sqrt{\beta^n - 1} \leq \beta^{n/2} - 1$ since the result must be an integer. This establishes $n/2$ as a bound on $\mathcal{N}_\beta(\sqrt{\overline{n}})$. This bound is optimal as can be seen from the following equivalent inequalities:

$$\beta^{(n/2)-1} \leq \sqrt{\beta^n - 1} \Longleftrightarrow \beta^{n-2} \leq \beta^n - 1 \Longleftrightarrow \beta^{n-2}(\beta^2 - 1) \geq 1$$

(note $\beta \geq 2$). If n is odd (and so $n \geq 1$),

$$\sqrt{\beta^n - 1} < \sqrt{\beta^n} < \sqrt{\beta^{n+1}} = \beta^{(n+1)/2} ,$$

hence, $\sqrt{\beta^n - 1} \leq \beta^{(n+1)/2} - 1$. This bound is optimal, also, since

$$\beta^{(n-1)/2} \leq \sqrt{\beta^n - 1} \Longleftrightarrow \beta^{n-1} \leq \beta^n - 1 \Longleftrightarrow \beta^{n-1}(\beta - 1) \geq 1 ,$$

which is clearly true. □

DEFINITION 1.6. *The n-ary operation of integer greatest common divisor, when applied to the equivalence classes $\overline{n}_1, \ldots, \overline{n}_k$, is given by*

$\gcd(\overline{n}_1, \overline{n}_2, \ldots, \overline{n}_k) \equiv \bar{1} \quad (k > 1)$.

The definition of the GCD seems at first inconsistent with the other operations, but it is really quite appropriate for worst case expression swell behavior. Typically in symbolic mathematical calculations, GCDs are used to find the common factors of sets of expressions in order to simplify subsequent computations in some way (e.g., removing the common factors from the numerator and denominator of a rational number, reducing it to lowest terms). Therefore, expression swell will be maximized if the quantities involved in a calculation are all relatively prime with respect to one another. This implies that the GCD is always one. As an example, consider the least common multiple (LCM) of an n_1-digit integer and an n_2-digit integer. In the worst case, the LCM becomes

$$\mathrm{lcm}(\overline{n}_1, \overline{n}_2) \equiv (\overline{n}_1 \odot \overline{n}_2) \overset{\text{exact}}{\ominus\!\!\!\div} \gcd(\overline{n}_1, \overline{n}_2) = \overline{n_1 + n_2} \overset{\text{exact}}{\ominus\!\!\!\div} \bar{1} = \overline{n_1 + n_2} .$$

In a manner similar to the above, a best case expression swell arithmetic for integer preserving operations can be developed. Best case behavior under an integer preserving operation means that the result will contain the least number of digits that are possible. Hence, this arithmetic will provide a lower bound on expression growth during a computation. For example, under these circumstances, the definition of the GCD would yield $\min(n_1, n_2, \ldots, n_k)$ [assuming $n_i > 0$ for all i]. However, the best case for addition and subtraction of integers with the same number of digits is zero (catastrophic cancellation), so the results of such an analysis may be quite a bit less interesting than for worst case behavior. In this article, no analysis of best case expression swell behavior will be given.

1.2. RATIONAL NUMBER CALCULATIONS

Worst case expression swell arithmetic can also be developed for rational number operations. Essentially, rational expression swell arithmetic can be considered as an extension of integer expression swell arithmetic to ordered pairs of integer equivalence classes. These ordered pairs will be denoted like $\frac{\overline{m}}{\overline{n}}$, which represents the set of rational numbers with m-digit numerators and n-digit denominators. Since this is worst case arithmetic, the components of the ordered pairs will be assumed to be relatively prime, implying that the corresponding set of rational numbers is in lowest terms and cannot be reduced in size. This is consistent with defining integer GCDs to be one as was done in the previous subsection. Note that a given rational number (e.g., $\frac{2}{4} \in \frac{\overline{1}}{\overline{1}}$) need not actually be in lowest terms, but for worst case arithmetic operations, this assumption will always be made nevertheless. With these definitions, as with the integers, $\mathcal{N}_\beta$ can be seen to partition the rational numbers into a countably infinite set of equivalence classes. $\frac{\overline{m}}{\overline{n}}$ will then be the general notation for one of these equivalence classes where m and n can take on any integer value satisfying $m \geq 0$ and $n > 0$ (if $m = 0$ then n is only allowed to be 1). Note that $\mathcal{N}_\beta(\frac{\overline{m}}{\overline{n}}) \equiv m + n$ as was proposed earlier.

Now, worst case rational expression swell arithmetic can be defined by analogy with ordinary rational arithmetic in terms of the operations of the previously examined worst case integer expression swell arithmetic. This is done as follows:

THEOREM 1.7. *The unary operations of rational absolute value, negation and square root, when applied to the equivalence class* $\frac{\overline{m}}{\overline{n}}$, *possess bounding functions given by*

(i) $\left|\frac{\overline{m}}{\overline{n}}\right| \equiv \frac{|\overline{m}|}{|\overline{n}|} = \frac{\overline{m}}{\overline{n}}$,

(ii) $\ominus\frac{\overline{m}}{\overline{n}} \equiv \frac{\ominus\overline{m}}{\overline{n}} = \frac{\overline{m}}{\overline{n}}$,

(iii) $\sqrt{\frac{\overline{m}}{\overline{n}}} \equiv \frac{\sqrt{\overline{m}}}{\sqrt{\overline{n}}} = \overline{\lceil \frac{m}{2} \rceil} / \overline{\lceil \frac{n}{2} \rceil}$.

THEOREM 1.8. *The binary operations of rational addition, subtraction, multiplication, division, exact division and exponentiation to an integer power when applied to the equivalence classes* $\frac{\overline{m_1}}{\overline{n_1}}$, $\frac{\overline{m_2}}{\overline{n_2}}$ *and* $\frac{\overline{m}}{\overline{n}}$, *where* $m_1, m_2, m > 0$, *possess bounding functions given by*

$$
\begin{array}{rl}
\textit{(i)} & \frac{\overline{m}_1}{\overline{n}_1} \oplus \frac{\overline{m}_2}{\overline{n}_2} \equiv \frac{(\overline{m}_1 \odot \overline{n}_2) \oplus (\overline{n}_1 \odot \overline{m}_2)}{\overline{n}_1 \odot \overline{n}_2} = \frac{\overline{m_1+n_2} \oplus \overline{n_1+m_2}}{\overline{n_1+n_2}} \,, \\
\textit{(ii)} & \frac{\overline{m}_1}{\overline{n}_1} \ominus \frac{\overline{m}_2}{\overline{n}_2} \equiv \frac{(\overline{m}_1 \odot \overline{n}_2) \ominus (\overline{n}_1 \odot \overline{m}_2)}{\overline{n}_1 \odot \overline{n}_2} = \frac{\overline{m_1+n_2} \oplus \overline{n_1+m_2}}{\overline{n_1+n_2}} \,, \\
\textit{(iii)} & \frac{\overline{m}_1}{\overline{n}_1} \odot \frac{\overline{m}_2}{\overline{n}_2} \equiv \frac{\overline{m}_1 \odot \overline{m}_2}{\overline{n}_1 \odot \overline{n}_2} = \frac{\overline{m_1+m_2}}{\overline{n_1+n_2}} \,, \\
\textit{(iv)} & \frac{\overline{m}_1}{\overline{n}_1} \oslash \frac{\overline{m}_2}{\overline{n}_2} \equiv \frac{\overline{m}_1}{\overline{n}_1} \odot \frac{\overline{n}_2}{\overline{m}_2} = \frac{\overline{m_1+n_2}}{\overline{n_1+m_2}} \,, \\
\textit{(v)} & \frac{\overline{m}_1}{\overline{n}_1} \overset{\text{exact}}{\oslash} \frac{\overline{m}_2}{\overline{n}_2} \equiv \frac{\overline{m}_1 \overset{\text{exact}}{\oslash} \overline{m}_2}{\overline{n}_1 \overset{\text{exact}}{\oslash} \overline{n}_2} = \frac{\overline{m_1-m_2+1}}{\overline{n_1-n_2+1}} \,, \\
\textit{(vi)} & \left(\frac{\overline{m}}{\overline{n}}\right) \mathbin{\textcircled{\uparrow}} 0 \equiv \frac{\overline{m} \obar 0}{\overline{n} \obar 0} = \frac{\overline{1}}{\overline{1}} \,, \\
\textit{(vii)} & \left(\frac{\overline{m}}{\overline{n}}\right) \mathbin{\textcircled{\uparrow}} k \equiv \frac{\overline{m} \obar k}{\overline{n} \obar k} = \frac{\overline{km}}{\overline{kn}} \quad (k > 0), \\
\textit{(viii)} & \left(\frac{\overline{m}}{\overline{n}}\right) \mathbin{\textcircled{\uparrow}} (-k) \equiv \left(\frac{\overline{n}}{\overline{m}}\right) \mathbin{\textcircled{\uparrow}} k = \frac{\overline{kn}}{\overline{km}} \quad (k > 0).
\end{array}
$$

If $m_1 = 0$ then the right-hand sides of (i) and (ii) become $\frac{\overline{m}_2}{\overline{n}_2}$, and the right-hand sides of (iii)–(v) become $\frac{\overline{0}}{\overline{1}}$. If $m_2 = 0$ then the right-hand sides of (i)–(v) are respectively, $\frac{\overline{m}_1}{\overline{n}_1}$, $\frac{\overline{m}_1}{\overline{n}_1}$, $\frac{\overline{0}}{\overline{1}}$, undefined and undefined. If $m = 0$ then the right-hand sides of (vi) and (viii) will be undefined and (vii) will be $\frac{\overline{0}}{\overline{1}}$.

A new operation, non-exact division, has been introduced but like the rest of the rational operations, it is adapted directly from ordinary rational arithmetic and so should come as no surprise. Finally, n-ary addition and multiplication for rational expression swell arithmetic are simple extensions of the binary versions.

Theorem 1.9. *The n-ary operations of rational multiplication and addition, when applied to the equivalence classes $\frac{\overline{m}_1}{\overline{n}_1}, \ldots, \frac{\overline{m}_k}{\overline{n}_k}$ $(k > 0)$, where $m_i > 0$ for all i, possess bounding functions given by*

$$
\begin{array}{rl}
\textit{(i)} & \frac{\overline{m}_1}{\overline{n}_1} \odot \frac{\overline{m}_2}{\overline{n}_2} \odot \cdots \odot \frac{\overline{m}_k}{\overline{n}_k} \equiv \frac{\overline{m}_1 \odot \overline{m}_2 \odot \cdots \odot \overline{m}_k}{\overline{n}_1 \odot \overline{n}_2 \odot \cdots \odot \overline{n}_k} = \frac{\overline{m_1+m_2+\cdots+m_k}}{\overline{n_1+n_2+\cdots+n_k}} \,, \\
\textit{(ii)} & \frac{\overline{m}_1}{\overline{n}_1} \oplus \frac{\overline{m}_2}{\overline{n}_2} \oplus \cdots \oplus \frac{\overline{m}_k}{\overline{n}_k} \equiv \frac{(\overline{m}_1 \odot \overline{n}_2 \odot \cdots \odot \overline{n}_k) \oplus (\overline{n}_1 \odot \overline{m}_2 \odot \cdots \odot \overline{n}_k) \oplus \cdots \oplus (\overline{n}_1 \odot \overline{n}_2 \odot \cdots \odot \overline{m}_k)}{\overline{n}_1 \odot \overline{n}_2 \odot \cdots \odot \overline{n}_k} \\
& \qquad = \frac{\overline{n-n_1+m_1} \oplus \overline{n-n_2+m_2} \oplus \cdots \oplus \overline{n-n_k+m_k}}{\overline{n}} \quad \textit{where } n \equiv \sum_{i=1}^{k} n_i \,.
\end{array}
$$

If any of the $m_i = 0$ then the right-hand side of (i) becomes $\frac{\overline{0}}{\overline{1}}$ and those terms with $m_i = 0$ are excluded from the sum in (ii).

1.3. MACSYMA IMPLEMENTATION

To obtain some practical experience with the above concepts, both integer and rational number worst case expression swell arithmetic were implemented in MACSYMA for $\beta = 10$. The top-level interface to expression swell arithmetic operations (there also exists a LISP-level interface which is accessed slightly differently) is provided by the functions abs_(x), neg_(x), add_($term_1$, $term_2$, ...), sub_(x,y), mul_($factor_1$, $factor_2$, ...), div_(x,y), ediv_(x,y) [Exact DIVision], power_(x,y) and gcd_(x,y) [$\mathcal{N}_{10}$ is performed by ndigits(e)]. If the global variable exprswell is false (its default value), then these functions perform ordinary arithmetic. However, if exprswell is set to true, then these functions treat their arguments as equivalence classes of $\mathcal{N}_{10}$ (when appropriate) and perform worst case expression swell arithmetic on them (no mixing of integer and rational number equivalence classes is permitted). Thus, an algorithm can be implemented using this set of functions in place of the normal MACSYMA arithmetic operators except where expression swell operations are never appropriate, such as index calculations or incrementing loop indices. Then, with exprswell set to its default value, the algorithm can be run in a normal manner. However, by simply changing the value of exprswell, an expression swell analysis of the algorithm for inputs from a given set of equivalence classes can be performed.

The MACSYMA implementation of general integer worst case expression swell n-ary addition is an extension of the n-ary addition of like terms defined previously. The terms are first sorted into ascending order, then after eliminating any zeroes, the leading set of like terms (which may be a single term) is combined using the n-ary addition of Theorem 1.3. The result is tacked onto the beginning of the list containing the remainder of the terms, which is then resorted, if necessary, to maintain the terms in ascending order. Next, the leading set of like terms is combined, with perhaps an initial non-like term, once again using the n-ary addition of Theorem 1.3 (the non-like term, if there is one, will be treated as just another like term in the n-ary addition). These last two steps will then repeat until the list contains a single term. For example, $\bar{1} \oplus \bar{1} \oplus \bar{2} \oplus \bar{2} \oplus \bar{4} \oplus \bar{4} = \bar{2} \oplus \bar{2} \oplus \bar{2} \oplus \bar{4} \oplus \bar{4} = \bar{3} \oplus \bar{4} \oplus \bar{4} = \bar{5}$. This algorithm was chosen to try to minimize the results of general n-ary addition and thus provide as tight a bound as possible.

2. Matrix Algorithm Analysis

Here, worst case expression swell arithmetic is applied to the analysis of some of the major matrix determinant and characteristic polynomial algorithms (and relationships). The next section will then examine the results of a series of numerical case studies performed using both the MACSYMA implementation of expression swell arithmetic as well as various sets of numbers.

2.1. DETERMINANT COMPUTATIONS

To start with, consider an analysis performed on the definition of the determinant:

$$\det A = \sum_{\mathbf{j} \in \pi_n} (-1)^{\mathcal{P}(\mathbf{j})} \prod_{i=1}^{n} a_{i j_i} \ ,$$

where π_n is the set of all permutations of $1, \ldots, n$ and $\mathcal{P}(\mathbf{j})$ is the parity (0 for even, 1 for odd) of the permutation $\mathbf{j} \equiv (j_1, \ldots, j_n)$. For an $n \times n$ matrix A, this will produce a sum of $n!$ products of n factors each, half of which are added and the rest subtracted. Therefore, if the entries of the matrix are all m-digit integers, then

$$\begin{aligned} \mathcal{N}_\beta(\det_d A) &\leq \mathcal{N}_\beta\left(\left[\frac{n!}{2} \odot (\overline{m} \Uparrow n)\right] \ominus \left[\frac{n!}{2} \odot (\overline{m} \Uparrow n)\right]\right) \\ &\leq \mathcal{N}_\beta\left(\left[\frac{n!}{2} \odot (\overline{m} \Uparrow n)\right] \oplus \left[\frac{n!}{2} \odot (\overline{m} \Uparrow n)\right]\right) \\ &\leq \mathcal{N}_\beta(n! \odot (\overline{m} \Uparrow n)) \leq \mathcal{N}_\beta(n! \odot \overline{nm}) \leq nm + \mathcal{N}_\beta(n! - 1) \ . \end{aligned} \tag{2.1}$$

One of the major schemes generally implemented in computer algebra systems (CASs) for computing determinants is some variant of Gaussian elimination. The Bareiss style algorithms (Bareiss, 1968), especially, are common as they are integer preserving. The simplest (and most frequently seen) of these is the one-step method where, during the k^{th} stage, the elements $a_{ij}^{(k)}$ in the $(n-k)\times(n-k+1)$ lower right block are computed by

$$a_{ij}^{(k)} = \frac{a_{kk}^{(k-1)}a_{ij}^{(k-1)} - a_{ik}^{(k-1)}a_{kj}^{(k-1)}}{a_{k-1,k-1}^{(k-2)}}\,, \quad (j = k, \ldots, n;\ i = k+1, \ldots, n).$$

$a_{00}^{(-1)}$ is defined to be 1, and the division will be exact. If A consists of m-digit integer elements, worst case expression swell analysis of this algorithm reveals that

$$\mathcal{N}_\beta(\det_G A) \leq nm + (n-1)^2\,. \tag{2.2}$$

This can be seen by performing the elimination on a representative matrix using expression swell arithmetic. In particular, for a 4×4 matrix of m-digit integers, the reduction proceeds as follows:

$$\begin{pmatrix} m & m & m & m \\ m & m & m & m \\ m & m & m & m \\ m & m & m & m \end{pmatrix} \rightarrow \begin{pmatrix} m & m & m & m \\ 0 & 2m+1 & 2m+1 & 2m+1 \\ 0 & 2m+1 & 2m+1 & 2m+1 \\ 0 & 2m+1 & 2m+1 & 2m+1 \end{pmatrix}$$

$$\rightarrow \begin{pmatrix} m & m & m & m \\ 0 & 2m+1 & 2m+1 & 2m+1 \\ 0 & 0 & 3m+4 & 3m+4 \\ 0 & 0 & 3m+4 & 3m+4 \end{pmatrix} \rightarrow \begin{pmatrix} m & m & m & m \\ 0 & 2m+1 & 2m+1 & 2m+1 \\ 0 & 0 & 3m+4 & 3m+4 \\ 0 & 0 & 0 & 4m+9 \end{pmatrix}.$$

For more detail, consider the second stage calculation of the (3,4) element:

$$\begin{aligned} \overline{a_{34}^{(2)}} &= [(\overline{a_{22}^{(1)}} \odot \overline{a_{34}^{(1)}}) \ominus (\overline{a_{32}^{(1)}} \odot \overline{a_{24}^{(1)}})] \overset{\text{exact}}{\oslash} \overline{a_{11}^{(0)}} \\ &= [(\overline{2m+1} \odot \overline{2m+1}) \ominus (\overline{2m+1} \odot \overline{2m+1})] \overset{\text{exact}}{\oslash} \overline{m} \\ &= [\overline{4m+2} \ominus \overline{4m+2}] \overset{\text{exact}}{\oslash} \overline{m} = \overline{4m+3} \overset{\text{exact}}{\oslash} \overline{m} = \overline{3m+4}\,. \end{aligned}$$

Thus, the right-hand side of (2.2) is a solution of the recurrence relation

$$r_n = (2r_{n-1}+1) - r_{n-2} + 1 = 2(r_{n-1}+1) - r_{n-2}$$

with initial conditions of $r_0 = 1$ and $r_1 = m$.

A comparison of the above two theoretical worst case determinations of the size of the determinant of an $n \times n$ matrix with m-digit integer entries shows that the former yields a tighter and thus a better bound. This can be seen by examining the difference

$$\begin{aligned} \mathcal{N}_\beta(\det_G A) - \mathcal{N}_\beta(\det_d A) &\sim (n-1)^2 - \mathcal{N}_\beta(n!-1) \\ &= (n-1)^2 - \lfloor \log_\beta(n!-1) \rfloor - 1 \qquad (n > 1) \\ &\approx n^2 - 2n - \left\lfloor \log_\beta \sqrt{2\pi n}\left(\tfrac{n}{e}\right)^n \right\rfloor \qquad (n \to \infty) \\ &\approx n^2 - n\log_\beta n + n(\log_\beta e - 2) \qquad (n \to \infty). \end{aligned}$$

The second line is a result of the definition of $\mathcal{N}_\beta$, and the third line is a result of approximating $n!$ (and hence $n!-1$) for n large by Stirling's formula. Indeed, $\mathcal{N}_\beta(\det_G A)$ will strictly dominate $\mathcal{N}_\beta(\det_d A)$ for all $n > 2$. The primary reason for this behavior is that worst case expression swell analysis is performed one stage at a time while following the Gaussian elimination algorithm, but all at once when using the definition of the determinant. The latter analysis takes full advantage of the stricter bounds produced by combining as many n-ary arithmetical operations as possible at one time, and so yields the better estimate. The former analysis does produce bounds on the size of the matrix elements at each stage of the elimination, but any overestimates from previous stages tend to accumulate.

The determinant of a matrix can also be bounded using Hadamard's inequality. This relationship can be expressed for an $n \times n$ matrix A by

$$|\det A|^2 \leq \prod_{i=1}^{n} \left(\sum_{j=1}^{n} a_{ij}^2 \right) .$$

If all the elements of A are m-digit integers, then here worst case expression swell analysis gives

$$\begin{aligned} \mathcal{N}_\beta(|\det_H A|^2) &\leq \mathcal{N}_\beta([n \odot (\overline{m} \mathbin{\textcircled{$\uparrow$}} 2)] \mathbin{\textcircled{$\uparrow$}} n) \leq \mathcal{N}_\beta([n \odot \overline{2m}\,] \mathbin{\textcircled{$\uparrow$}} n) \\ &\leq \mathcal{N}_\beta(\overline{2m + \mathcal{N}_\beta(n-1)} \mathbin{\textcircled{$\uparrow$}} n) \leq \mathcal{N}_\beta(\overline{2nm + n\mathcal{N}_\beta(n-1)}) . \end{aligned}$$

Hence,

$$\mathcal{N}_\beta(\det_H A) \leq nm + \left\lceil \frac{1}{2} n \mathcal{N}_\beta(n-1) \right\rceil . \tag{2.3}$$

This estimate is of about the same asymptotic order as the one produced by the definition of the determinant.

Rational expression swell arithmetic can also be applied fruitfully to Hadamard's inequality. Suppose B is an $n \times n$ matrix with rational number entries consisting of m-digit numerators and denominators, then

$$\mathcal{N}_\beta(|\det_H B|^2) \leq \mathcal{N}_\beta\left(\left[n \odot \left(\frac{\overline{m}}{\overline{m}} \mathbin{\textcircled{$\uparrow$}} 2 \right) \right] \mathbin{\textcircled{$\uparrow$}} n \right) \leq \mathcal{N}_\beta\left(\left[n \odot \frac{\overline{2m}}{\overline{2m}} \right] \mathbin{\textcircled{$\uparrow$}} n \right) .$$

Now,

$$\begin{aligned} k \odot \frac{\overline{m}}{\overline{m}} &\equiv \underbrace{\frac{\overline{m}}{\overline{m}} \oplus \cdots \oplus \frac{\overline{m}}{\overline{m}}}_{k \text{ terms}} = \frac{\overbrace{\overbrace{(\overline{m} \odot \cdots \odot \overline{m})}^{k \text{ factors}} \oplus \cdots \oplus \overbrace{(\overline{m} \odot \cdots \odot \overline{m})}^{k \text{ factors}}}^{k \text{ terms}}}{\underbrace{\overline{m} \odot \cdots \odot \overline{m}}_{k \text{ factors}}} \\ &= \frac{k \odot \overline{km}}{\overline{km}} = \frac{\overline{km + \mathcal{N}_\beta(k-1)}}{\overline{km}} , \end{aligned}$$

therefore,

$$\mathcal{N}_\beta(|\det_H B|^2) \leq \mathcal{N}_\beta\left(\frac{\overline{2nm + \mathcal{N}_\beta(n-1)}}{\overline{2nm}} \mathbin{\textcircled{$\uparrow$}} n \right) \leq \mathcal{N}_\beta\left(\frac{\overline{2n^2m + n\mathcal{N}_\beta(n-1)}}{\overline{2n^2m}} \right) .$$

Taking an exact square root once again,

$$\mathcal{N}_\beta(\det_H B) \leq \mathcal{N}_\beta\left(\frac{\overline{n^2m + \lceil n\mathcal{N}_\beta(n-1)/2 \rceil}}{\overline{n^2m}} \right) = 2n^2m + \left\lceil \frac{1}{2} n\mathcal{N}_\beta(n-1) \right\rceil .$$

2.2. CHARACTERISTIC POLYNOMIAL COMPUTATIONS

Generally in CASs, the characteristic polynomial of a matrix A is calculated by forming the matrix $A - \lambda I$, where λ is a scalar variable, and then computing its determinant. The determinant will always involve polynomial arithmetic, even for a purely numerical matrix, so it is difficult to directly perform expression swell analysis on this method. However, in the previous subsection, expression swell analysis was performed on the determinant calculation of certain integer and rational number matrices. Since the determinant is plus or minus the constant term of the characteristic polynomial, which for nonsingular matrices is typically among the largest coefficients of the polynomial in gross size, these analyses should give a good bound on the size of the largest coefficient of the characteristic polynomial for matrices with entries of the same size.

A second method, which computes the coefficients of the characteristic polynomial of a purely numerical matrix using no polynomial arithmetic, involves taking the trace of powers of the matrix. This procedure, known as the method of Leverrier, takes advantage of the fact that for an $n \times n$ matrix A with eigenvalues $\{\lambda_i\}_{i=1}^n$,

$$\operatorname{tr} A^k = \sum_{i=1}^{n} \lambda_i^k \,, \quad (k = 0, 1, \ldots).$$

The coefficients of the characteristic polynomial can be obtained from the symmetric functions of the above quantities for $k = 1, \ldots, n$ (Stone, 1972). This method requires $O(n^4)$ operations and so is not practical for large matrices. Nevertheless, it is instructive to perform worst case expression swell analysis on this algorithm. Initially, consider the first portion of the algorithm, which computes the various traces. If the entries of A are m-digit integers, then the number of digits in the entries of $AA = A^2$ will be bounded by $2m + \mathcal{N}_\beta(n-1)$ [i.e., $\mathcal{N}_\beta(n \odot [\overline{m} \odot \overline{m}])$]. For $AA^2 = A^3$, the number of digits in the entries will be bounded by $3m + 2\mathcal{N}_\beta(n-1)$, and in general, the number of digits in the entries of $AA^{k-1} = A^k$ will be bounded by $km + (k-1)\mathcal{N}_\beta(n-1)$. Hence,

$$\begin{aligned}\mathcal{N}_\beta(\operatorname{tr} A^k) &= \mathcal{N}_\beta\left(\sum_{i=1}^{n} (A^k)_{ii}\right) \le \mathcal{N}_\beta(n \odot \overline{km + (k-1)\mathcal{N}_\beta(n-1)}) \\ &\le k[m + \mathcal{N}_\beta(n-1)] \,.\end{aligned}$$

Now, the coefficient c_k of λ^k $(0 \le k \le n-1)$ in the characteristic polynomial $c_A(\lambda)$ of A will be composed of $p(n-k)$ terms, each of which will be proportional to a product of traces such that the total sum of powers involved is $n-k$. Here, $p(n)$ is the number of partitions of the integer n (i.e., the number of ways n can be written as a sum of positive integers where order does not matter), and is computable from Hardy and Ramanujan's formula (Cohen, 1978, p. 79) by

$$p(n) \approx \frac{1}{4\sqrt{3}\, n} e^{\pi\sqrt{2n/3}} \,.$$

For example, the coefficient of λ for A, 4×4, is

$$c_1 = \frac{\operatorname{tr} A^3}{3} - \frac{(\operatorname{tr} A^2)(\operatorname{tr} A)}{2} + \frac{(\operatorname{tr} A)^3}{6} \,,$$

which consists of $p(4-1) = p(3) = 3$ terms. It is easily seen that $\mathcal{N}_\beta([\operatorname{tr} A] \circledast k) = \mathcal{N}_\beta(\operatorname{tr} A^k)$, therefore, ignoring divisions,

$$\begin{aligned} \mathcal{N}_\beta(c_k) &\le \mathcal{N}_\beta(p(n-k) \odot \overline{(n-k)[m + \mathcal{N}_\beta(n-1)]}) \\ &\le (n-k)[m + \mathcal{N}_\beta(n-1)] + \mathcal{N}_\beta(p(n-k) - 1) \,. \end{aligned}$$

Note that $\mathcal{N}_\beta(c_{n-1}) \sim \mathcal{N}_\beta(\operatorname{tr} A)$ [since $p(1) = 1$], which is consistent with $c_{n-1} = -\operatorname{tr} A$. Also,

$$\mathcal{N}_\beta(\det_L A) = \mathcal{N}_\beta(c_0) \le nm + n\mathcal{N}_\beta(n-1) + \mathcal{N}_\beta(p(n) - 1) \,, \tag{2.4}$$

hence,

$$\begin{aligned} \mathcal{N}_\beta(\det_L A) &\approx nm + n(\lfloor \log_\beta(n-1) \rfloor + 1) + \left\lfloor \log_\beta \left(\frac{1}{4\sqrt{3}\,n} e^{\pi\sqrt{2n/3}} - 1 \right) \right\rfloor + 1 \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (n > 1) \\ &\approx nm + n \log_\beta n + n \qquad (n \to \infty). \end{aligned}$$

This estimate compares favorably to the one derived from the definition of the determinant (although it will be seen in the next section that it is somewhat higher).

A third way to compute the characteristic polynomial of a matrix is to transform the matrix into upper Hessenberg or tridiagonal form via a similarity reduction (which will preserve the characteristic polynomial), and then compute the characteristic polynomial of the transformed matrix using a computationally cheap algorithm. A matrix is upper Hessenberg if all the entries below the subdiagonal are zero (i.e., H is upper Hessenberg if $h_{ij} = 0$ whenever $i > j + 1$). The reduction of a general matrix into upper Hessenberg form proceeds via a series of elementary similarity transformations. Once an upper Hessenberg matrix, H, similar to the original matrix, has been constructed, the characteristic polynomial of the upper Hessenberg matrix (and thus of the original matrix) is calculated next. The basic algorithm is to create a triangular system of equations using H, which can then be easily solved for the coefficients of the characteristic polynomial (Johnson and Riess, 1981). Worst case expression swell analysis of this kind of procedure is difficult to generalize and so will not be attempted. In the following section, results from the analyses of specific examples will be presented. These analyses were performed using the MACSYMA implementation of worst case expression swell arithmetic.

3. Results of Case Studies

It is all very well to derive theoretical bounds on the possible expression swell in a calculation, but when compared to the results of actual computations, how good of an estimate of real behavior do these bounds really provide? In order to answer this question, a number of case studies were performed involving matrices of various sizes with initial integer or rational number entries. Some of these results provide quite striking examples of intermediate expression swell in action.

To begin with, Table 3 presents the four worst case bounds [relations (2.3), (2.1), (2.4) and (2.2), respectively, of the previous section] derived for the determinant of an $n \times n$ matrix containing m-digit integer entries, for a variety of values of n with $m = 4$ ($\beta = 10$ will be assumed throughout this section). Hadamard's inequality yields the lowest upper estimate on the number of digits in the determinant, although the bounds derived from the definition of the determinant and Leverrier's method fall within the

Table 3. Various predictions of the maximum number of digits comprising the determinant of an $n \times n$ integer general matrix ($m = 4$).

n	Hadamard's inequality	Analysis of the Definition	Method of Leverrier	Gaussian elimination
3	14	13	16	16
4	18	18	21	25
5	23	23	26	36
6	27	27	32	49
7	32	32	37	64
8	36	37	42	81
9	41	42	47	100
10	45	47	52	121
20	100	99	123	441
30	150	153	184	961
40	200	208	245	1681
50	250	265	306	2601
60	300	322	367	3721
70	350	381	427	5041
80	400	439	488	6561
90	450	499	548	8281
100	500	558	609	10201

same approximate asymptotic order. As noted before, worst case bounds derived from the fraction-free Gaussian elimination algorithm grow much faster than those computed from the other methods.

Table 4 presents the results of actual calculations performed on $n \times n$ matrices for values of n ranging from 3 through 10, where the initial matrix elements were 4-digit integers. The table is divided into three sections. The first section (a) exhibits the greatest number of digits encountered in the elements of the upper Hessenberg matrix produced by a similarity transformation of the original matrix using a division-free version of the algorithm. The column labeled exprswell shows the results produced by setting the global variable exprswell to true in MACSYMA. The next column displays the outcome of using an initial matrix composed of the n^2 largest 4-digit primes. The last three columns in this table present the results of a statistical sampling in which matrices consisting initially of random 4-digit integers were used. The first pair of numbers is the mean and standard deviation obtained from a series of calculations whose number is given by N_{samples}.† The last column gives the minimum and maximum values for the largest number of digits attained.

Section (b) of Table 4 shows the number of digits contained in the largest coefficient of the characteristic polynomial that was computed directly from the corresponding upper Hessenberg matrix of section (a). Due to the division-free nature of the similarity reduction to upper Hessenberg form that was used, a straightforward computation of the characteristic polynomial will produce, in general, a non-monic result. However, the

† In some cases, when a sequence of calculations was performed as here, later calculations were done fewer times due to time constraints and/or random problems occurring as a result of running large MACSYMA jobs continuously for days at a time (such as occasional memory corruption resulting from the growth of a MACSYMA job).

Table 4. Maximum number of digits comprising the results of various transformations applied to an $n \times n$ integer general matrix ($m = 4$).

n	exprswell	primes	random numbers	N_{samples}	min $\rightarrow$ max
3	14	13	11.42 ± 0.75	100	9 → 12
4	43	27	30.73 ± 2.10	100	24 → 35
5	130	59	74.98 ± 2.66	100	68 → 81
6	391	98	142.38 ± 4.26	100	130 → 152
7	1174	155	249.75 ± 6.75	100	233 → 262
8	3523	240	366.50 ± 9.53	100	343 → 388
9	10570	334	585.30 ± 11.79	100	556 → 612
10	31711	469	825.92 ± 14.22	13	797 → 844

a. Division free transformation into Hessenberg form.

n	exprswell	primes	random numbers	N_{samples}	min $\rightarrow$ max
3	43	31	30.09 ± 2.28	100	22 → 34
4	172	92	115.60 ± 7.90	100	90 → 128
5	648	277	364.62 ± 13.36	100	331 → 393
6	2341	561	840.63 ± 25.23	100	765 → 898
7	8209	1058	1731.99 ± 47.74	100	1616 → 1824
8	28170	1882	2911.08 ± 77.09	100	2720 → 3081
9	95110	2959	5243.04 ± 106.39	100	4977 → 5483
10	317083	4644	8243.42 ± 147.58	12	7942 → 8422

b. Characteristic polynomial of the Hessenberg matrix.

n	exprswell	primes	random numbers	N_{samples}	min $\rightarrow$ max
3	17	7	11.40 ± 0.84	10	10 → 12
4	25	7	15.20 ± 0.63	10	14 → 16
5	34	8	18.70 ± 0.67	10	18 → 20
6	44	8	22.70 ± 0.95	10	21 → 24
7	55	11	26.50 ± 0.97	10	24 → 27
8	67	11	30.60 ± 0.52	10	30 → 31
9	80	13	34.20 ± 0.63	10	33 → 35
10	94	14	37.90 ± 0.57	10	37 → 39

c. Above divided by its leading coefficient.

leading coefficient of this polynomial will exactly divide all the other coefficients, producing a monic polynomial whose constant term will be plus or minus the determinant of the original matrix. Essentially, this particular procedure for computing the monic characteristic polynomial of an integer matrix reserves all divisions until the final step of the calculation. The maximum number of digits found in a coefficient (nearly always the constant term) of the final, simplified characteristic polynomial is presented in the final section (c) of Table 4.

The trend displayed in Table 4 is quite typical of calculations involving intermediate expression swell. Even though the final numbers have relatively few digits, intermediate computations in this particular algorithm are already creating integers whose size is approaching 5000 digits when n is just 10. It is interesting to note that the matrices with prime entries pretty much provide the smallest actual results. The opposite conclusion prevails in most of the other calculations surveyed here.

The theoretical limits on the size of the determinant, tabulated in Table 3, nicely bound the results in Table 4c. The exprswell calculations tend to greatly overestimate the maximum number of digits actually produced in the first two phases of the computation, except for the lowest values of n. However, after the final exact divisions, these bounds drop down to much more reasonable estimates. One has to be careful, though, in interpreting the results of the exprswell computations for this last calculation. The coefficients of the non-monic characteristic polynomials in Table 4b were determined by a series of calculations that maximized the number of digits at each step. In order to truly maximize the number of digits in the coefficients of the final monic polynomials, the leading coefficients of the non-monic polynomials (which act as divisors) should really have been minimized in the course of their calculation (using a best case expression swell arithmetic). This action, however, could have impacted the results of the worst case expression swell arithmetic in earlier phases of the calculations. Thus, the occurrence of exact divisions in an algorithm such as this one can lead to theoretical uncertainties in verifying whether the analysis does indeed produce a bounding calculation, although in practice, no exceptions have been found so far.

Table 5 presents the maximum number of digits found in coefficients of the characteristic polynomials of upper Hessenberg matrices whose initial nonzero entries were 4-digit integers. The exprswell calculations and the results from the statistical survey for $n = 3$ through 10 are remarkably similar to those exhibited in Table 4c. These similarities appear to imply that the results of computing characteristic polynomials of integer Hessenberg matrices will give a very good indication of the trends to be expected when computing monic characteristic polynomials of integer general matrices. This is useful since there are no theoretical uncertainties here as there were above, because this algorithm involves no divisions. Also, the computation of the characteristic polynomial of an integer Hessenberg matrix is reasonably fast as well as conservative of memory, so it was possible to perform calculations up through $n = 100$.[†]

The trends of Table 5 become more pronounced for $n > 10$. The worst case maximums derived from Hadamard's inequality continue to provide good bounds on the actual results, although they become less good for increasing n. For $n > 10$, these bounds, which are growing linearly, begin to leave the real recorded maximums, which are also growing about linearly, further and further behind. Generalizing these data tendencies

[†] All the calculations in this section were performed on Sun 3/160 workstations with 4 megabytes of memory running under a version of the UNIX operating system (SunOS 3.4).

Table 5. Maximum number of digits comprising the coefficients of the characteristic polynomial of an $n \times n$ integer Hessenberg matrix ($m = 4$).

n	exprswell	primes	random numbers	N_{samples}	min $\rightarrow$ max
3	17	9	11.46 ± 0.54	100	$10 \rightarrow 12$
4	25	14	15.27 ± 0.60	100	$13 \rightarrow 16$
5	34	15	19.13 ± 0.68	100	$17 \rightarrow 20$
6	44	21	23.00 ± 0.70	100	$21 \rightarrow 24$
7	55	23	26.51 ± 0.75	100	$24 \rightarrow 28$
8	67	29	30.35 ± 0.73	100	$28 \rightarrow 32$
9	80	33	34.12 ± 0.76	100	$33 \rightarrow 36$
10	94	38	37.91 ± 0.79	100	$36 \rightarrow 39$
20	289	—	75.40 ± 0.89	5	$74 \rightarrow 76$
30	584	—	113.40 ± 1.14	5	$112 \rightarrow 115$
40	979	—	150.60 ± 1.34	5	$150 \rightarrow 153$
50	1474	—	187.60 ± 1.14	5	$186 \rightarrow 189$
60	2069	—	227.60 ± 1.52	5	$226 \rightarrow 230$
70	2764	—	264.40 ± 1.14	5	$263 \rightarrow 266$
80	3559	—	301.80 ± 1.30	5	$301 \rightarrow 304$
90	4454	—	341.60 ± 1.95	5	$339 \rightarrow 344$
100	5449	—	378.00 ± 2.83	5	$374 \rightarrow 380$

Table 6. Maximum number of digits comprising the coefficients of the characteristic polynomial of an $n \times n$ integer Hessenberg matrix.

n	$m = 4$	$m = 5$	$m = 6$
5	20	25	30
10	39	49	59
20	76	97	116
30	115	146	176
40	153	193	232
50	189	242	292

(as well as those found for $m = 5$ and 6—see Table 6), a good empirical bound on the maximum number of digits in the determinant of an $n \times n$ integer matrix (A), which has entries consisting of no greater than m digits, appears to be given simply by

$$\mathcal{N}_{10}(\det A) \leq nm \ .$$

Note that this is the worst case bound for the determinant of an $n \times n$ diagonal matrix ($\overline{m} \circledUparrow n = \overline{nm}$).

Table 7 shows what happens when the Hessenberg matrix is initially filled with 4-digit rational numbers (the numerators and denominators are both 4-digit integers) and the matrix is first "derationalized." Derationalization is the process of converting a matrix of rational numbers into a matrix of integers by multiplying the matrix by the least common multiple of the denominators of its rational number entries. This number (call it d) is an implicit divisor of the integer matrix, and if the matrix is involved in subsequent

Table 7. Maximum number of digits comprising the coefficients of the characteristic polynomial of an $n \times n$ "derationalized" Hessenberg matrix, initially filled with 4-digit rational numbers.

n	exprswell	primes	random numbers	N_{samples}	min $\rightarrow$ max
3	107	93	78.07 ± 5.23	100	65 → 89
4	225	205	162.47 ± 11.43	100	123 → 187
5	404	376	283.81 ± 19.35	100	236 → 331
6	656	620	451.50 ± 30.81	100	346 → 507
7	993	946	657.26 ± 45.02	100	536 → 746
8	1427	1368	934.24 ± 56.08	100	804 → 1066
9	1970	1895	1245.57 ± 80.66	100	1080 → 1426
10	2634	2540	1612.45 ± 112.96	100	1245 → 1932

calculations, d needs to be taken into account. In this particular example, the characteristic polynomial will again, in general, be non-monic, but in this case, dividing the coefficients by the leading coefficient is not guaranteed to be exact (since this is really the characteristic polynomial of a rational number matrix), and so is not performed.

In Table 7, the "prime" rational numbers (rational numbers whose numerators and denominators are prime integers) provided the worst actual results of expression swell. The observation that "prime" rational numbers typically generate the greatest expression growth can also be noted in other calculations that start off with matrices of rational numbers, completely contrary to what was observed earlier for calculations that started with integer matrices. A nice discovery is that the exprswell calculations produced good bounds on the worst case expression swell. This may be due to the following consideration. The prime computations will be particularly large since the denominators of the initial rational number entries will all be big and relatively prime to each other, so that derationalization will create nearly the maximum possible integer entries for a matrix of a given size undergoing this transformation. The exprswell calculations will act as if they are computing the characteristic polynomials of integer Hessenberg matrices with large effective values of m (essentially, the original m times the number of nonzero elements in the matrix). Thus, any excesses in the exprswell computations (which seem to depend only on n) will be overwhelmed (at least, at these values of n) by the large effective values of m. Table 8 shows "prime" and exprswell results for three values of m, demonstrating that the relative error narrows as m increases, a result which is to be expected if the above argument is correct.

In Table 9 is presented the results of another characteristic polynomial computation via an intermediate Hessenberg transformation. This time, the entries of the initial general matrices were 4-digit rational numbers and all calculations were done in rational arithmetic. As an example, Figure 3 exhibits the complete results of a typical calculation. Note that the numbers in the table represent the number of digits in the maximal rational number (i.e., the largest value obtained by summing the number of digits in the numerator and denominator of each rational number under consideration).

Again, the calculations that started off with "prime" rational number matrices produced the greatest growth in expression size. Also, like the computations whose results were shown in Table 4, expression swell decreased dramatically between the intermedi-

Table 8. Maximum number of digits comprising the coefficients of the characteristic polynomial of an $n \times n$ "derationalized" Hessenberg matrix, initially filled with m-digit rational numbers.

	exprswell			primes		
n	$m = 4$	$m = 5$	$m = 6$	$m = 4$	$m = 5$	$m = 6$
3	107	131	155	93	116	140
4	225	277	329	205	253	307
5	404	499	594	376	471	565
6	656	812	968	620	777	931
7	993	1231	1469	946	1186	1424
8	1427	1771	2115	1368	1716	2059
9	1970	2447	2924	1895	2380	2858
10	2634	3274	3914	2540	3194	3835

Table 9. Maximum number of digits comprising the results of transformations applied to an $n \times n$ general matrix, initially filled with 4-digit rational numbers.

n	exprswell	primes	random numbers	N_{samples}	min → max
3	82	61	50.15 ± 4.84	100	36 → 60
4	911	216	176.29 ± 11.92	100	137 → 202
5	13534	622	504.29 ± 27.15	100	438 → 564
6	255497	1369	1096.85 ± 46.77	93	983 → 1223
7	5852292	2588	2013.28 ± 98.13	100	1722 → 2206
8	157563119	4313	3275.51 ± 130.41	57	3027 → 3584
9	4874278234	6671	4971.33 ± 170.71	12	4730 → 5310
10	170327525637	9717	7194.50 ± 198.44	6	6932 → 7488

a. Transformation into Hessenberg form.

n	exprswell	primes	random numbers	N_{samples}	min → max
3	711	69	60.34 ± 4.05	100	46 → 69
4	15812	125	105.24 ± 5.19	100	91 → 117
5	493670	198	163.02 ± 6.45	100	145 → 178
6	19254949	286	231.30 ± 8.34	93	208 → 250
7	898326065	390	309.65 ± 11.57	100	281 → 334
8	48860659874	508	399.82 ± 14.61	57	372 → 440
9	3039845981556	642	498.25 ± 11.41	12	484 → 518
10	213099621017855	791	611.50 ± 11.43	6	596 → 631

b. Characteristic polynomial of the Hessenberg matrix.

$$A = \begin{pmatrix} \frac{9533}{9539} & \frac{9547}{9551} & \frac{9587}{9601} & \frac{9613}{9619} & \frac{9623}{9629} \\ \frac{9631}{9643} & \frac{9649}{9661} & \frac{9677}{9679} & \frac{9689}{9697} & \frac{9719}{9721} \\ \frac{9733}{9739} & \frac{9743}{9749} & \frac{9767}{9769} & \frac{9781}{9787} & \frac{9791}{9803} \\ \frac{9811}{9817} & \frac{9829}{9833} & \frac{9839}{9851} & \frac{9857}{9859} & \frac{9871}{9883} \\ \frac{9887}{9901} & \frac{9907}{9923} & \frac{9929}{9931} & \frac{9941}{9949} & \frac{9967}{9973} \end{pmatrix}$$

a. Initial matrix.

$$\mathcal{N}_{10}(H) = \begin{pmatrix} \frac{4}{4} & \frac{33}{32} & \frac{105}{104} & \frac{175}{175} & \frac{4}{4} \\ \frac{4}{4} & \frac{33}{32} & \frac{106}{105} & \frac{175}{175} & \frac{4}{4} \\ 0 & \frac{53}{56} & \frac{113}{117} & \frac{187}{190} & \frac{13}{16} \\ 0 & 0 & \frac{155}{158} & \frac{229}{231} & \frac{69}{72} \\ 0 & 0 & 0 & \frac{309}{313} & \frac{143}{146} \end{pmatrix}$$

b. The sizes of the entries in the upper Hessenberg matrix H that is similar to A.

$$\mathcal{N}_{10}(c_H(\lambda)) = \lambda^5 + \frac{21}{20}\lambda^4 + \frac{98}{100}\lambda^3 + \frac{95}{100}\lambda^2 + \frac{92}{100}\lambda + \frac{88}{100}$$

c. The sizes of the coefficients of the characteristic polynomial $c_H(\lambda) = c_A(\lambda)$.

Figure 3. Stages in the expression swell analysis of the computation of the characteristic polynomial of a 5×5 general matrix containing "prime" 4-digit rational numbers.

ate results found in the entries of the Hessenberg matrices and the final numbers that comprised the coefficients of the characteristic polynomials. These latter numbers appear to be quite nicely bounded by $2n^2m$, which is very similar to the bound derived from Hadamard's inequality. (In this case, the worst case bound for a diagonal matrix does not quite match up: $\overline{\frac{m}{m}} \mathbin{\textcircled{\uparrow}} n = \overline{\frac{nm}{nm}} = \overline{2nm}$.) The bounds computed from the MACSYMA implementation of rational worst case expression swell arithmetic are huge and provide virtually no useful information.

As an experiment to see what effect GCDs have on rational number computations, the above calculations on "prime" rational number matrices were repeated with the GCD function forced always to return one, thus allowing no cancellation of common factors. These results are compared with those from Table 9, in which GCDs were taken freely, in Table 10. It is quite clear from this comparison of maximal numbers of digits that GCDs, which are taken every time a new rational number is formed in MACSYMA, have important effects in minimizing expression swell which, in the samples reviewed in Table 10, is growing exponentially for the "no GCDs" cases. The reason that the results for "no GCDs" are complete only through $n = 6$ is because the MACSYMA computations for bigger cases ran out of available memory.

One last computation of interest is the transformation of a symmetric matrix into tridiagonal form [the modified fast Givens' method for a real matrix—see (Gentleman, 1973, Hammarling, 1974, Moler and Wilkinson, 1977, Wester, 1992)]. Table 11 displays the results for six values of n. A notable feature is that although the final results consist of less than half the number of digits compared to those for a general matrix (Table 9), the intermediate forms, starting at $n = 7$, produce larger maximal expression swell. Thus, it can be seen that the modified fast Givens' algorithm will quickly become unwieldly for increasing n, and it will be better to just use the general purpose Hessenberg algorithm.

Table 10. Maximum number of digits comprising the results of the transformation of an $n \times n$ general matrix, initially filled with 4-digit "prime" rational numbers, into Hessenberg form and then taking the characteristic polynomial of the Hessenberg matrix.

	Hessenberg matrix		Characteristic polynomial	
n	GCDs	no GCDs	GCDs	no GCDs
3	61	61	69	122
4	216	379	125	565
5	622	2360	198	3129
6	1369	14203	286	18365
7	2588	82576	390	——

Table 11. Maximum number of digits comprising the results of transformations applied to an $n \times n$ symmetric matrix, initially filled with 4-digit rational numbers.

n	primes	random numbers	N_{samples}	min $\rightarrow$ max
3	29	24.31 ± 1.82	100	$17 \rightarrow 27$
4	135	118.96 ± 6.02	100	$96 \rightarrow 133$
5	451	377.51 ± 21.17	100	$334 \rightarrow 424$
6	1299	1066.30 ± 57.21	80	$873 \rightarrow 1199$
7	2856	2237.50 ± 107.46	20	$2074 \rightarrow 2475$
8	5302	3997.60 ± 120.55	5	$3882 \rightarrow 4163$

a. Transformation into tridiagonal form.

n	primes	random numbers	N_{samples}	min $\rightarrow$ max
3	30	30.64 ± 2.49	100	$18 \rightarrow 34$
4	56	53.45 ± 3.11	100	$43 \rightarrow 60$
5	89	82.22 ± 4.49	100	$71 \rightarrow 91$
6	130	118.61 ± 5.55	80	$105 \rightarrow 129$
7	178	155.80 ± 5.82	20	$142 \rightarrow 166$
8	236	197.20 ± 6.22	5	$190 \rightarrow 207$

b. Characteristic polynomial of the tridiagonal matrix.

4. Concluding Remarks

Expression swell is a significant and in the long run, an inevitable problem of symbolic computation. Since expression swell is ultimately unavoidable, one goal should be to make it at least manageable. One way to accomplish this is to develop a collection of procedures which will allow a user to be able to predict the progress of expression growth for a given calculation. In this article, an attempt has been made to develop some tools to pursue the above goal for a certain class of computations. The application of these tools has had mixed success.

One tool for charting expression growth is to perform a variety of calculations with varying initial conditions, and measure the sizes of the final and various intermediate results. This procedure, although tedious, has produced some interesting conclusions about the maximum size of coefficients in the characteristic polynomials of certain integer and rational number matrices, where the nonzero entries were initially uniform in size.

A second tool is worst case (and best case) expression swell arithmetic. This can be used both for deriving general bounds on expression swell over a class of calculations, and also for determining such bounds for specific computations. Some of the bounds looked at above have produced good estimates on real maximal expression size, while others have given outrageous overestimates. At this preliminary stage in the development of a useful expression swell arithmetic, some observations can be made based on the experience gained here.

For integer worst case expression swell arithmetic, the best results (the tightest bounds) have come about when analyses have been performed on algorithms or inequalities that have minimized the mix of operations needed to obtain an estimate. For example, the analysis of the Gaussian elimination algorithm of Section 2.1 and many of the computed bounds in the previous section required a complex, multi-stage calculation and in many cases, this resulted in excessive estimates, even for small values of the matrix dimension. On the other hand, simple formulas like Hadamard's inequality, where the various arithmetical operations are reasonably consolidated, gave good bounds on the expression swell. A major reason for this behavior is that expression swell arithmetic ignores past history. Thus, if a calculation produces a worst case outcome that just barely exceeds n digits [e.g., $(\beta + 1) \odot \overline{n - 2}$], the result will normally be treated as if it was the largest possible n-digit integer in the very next calculation, effectively neglecting the number's origin. Thus, the more steps there are in a computation, the more often these jumps in (interpreted) value can occur. Of course, this phenomenon also allows the analysis to be greatly simplified.

Working with an arbitrary integer base β instead of 10 (the latter was chosen in the previous section only for convenience) can help to soften the effect of jumps in value if a base smaller than 10 is selected. This is because integers having the same number of base β digits will form more and smaller sets as β decreases, thus producing finer divisions of the integers. Therefore, expression swell bounds can be made more precise and jumps in value can be lessened (for instance, 1111_{10} would jump to 9999_{10} using the decimal definition $\mathcal{N}_{10}(n)$, but it would only jump to 2047_{10} with a binary definition, where $\mathcal{N}_2$ would count the number of bits in n).

Another factor in why the expression swell arithmetic developed here sometimes tends to greatly overestimate expression growth is that no provision has been (or is easily able to be) made to account for the effects of subtractive cancellation (which can be significant for procedures such as the solution of a triangular system of equations) and cancellation of

Table 12. The number of seconds needed to add, multiply or take the GCD of a pair of random m-digit integers in MAPLE 4.1 running on a Sun 3/160, where the results have been averaged over 1000 trials. The GCDs were computed using the Euclidean algorithm, taking the average number of iterations listed. The last column displays the fraction of the GCDs found to be unity.

m	$+$	$\times$	GCD	iterations	GCD $= 1$
1	.00253	.00259	.02161	1.982	0.617
2	.00253	.00268	.02268	3.951	0.595
3	.00264	.00278	.02303	5.837	0.594
4	.00263	.00279	.02349	7.811	0.587
5	.00440	.00283	.02545	9.781	0.617
6	.00268	.00281	.02353	11.785	0.614
7	.00272	.00490	.02378	13.777	0.617
8	.00270	.00287	.02377	15.723	0.611
9	.00270	.00288	.02589	17.525	0.619
10	.00278	.00294	.02437	19.304	0.615

common factors in non-exact quotients. A theorem from number theory due to Dirichlet states that the probability that two integers chosen at random are relatively prime is $6/\pi^2$ or about 60.8% (Knuth, 1981, p. 324). (In Table 12, the results of doing this experiment in MAPLE (Char et al, 1988) are shown, along with some relevant timing data.) Performing rational arithmetic operations again and again on elements of a given matrix involves numbers that are, after a while, far from random and so GCDs play a significant role, as was seen in Table 10.

The other major reason why the rational worst case expression swell arithmetic often did so poorly is that rational operations involve extensive integer arithmetic, so that the jump in value effect is greatly magnified. Nonetheless, this arithmetic did provide bounds on the size of other smaller expressions involved in the calculations which were more reasonable, as well as indicating the relative ranking by size of the various expressions present at a given stage (e.g., matrix elements or polynomial coefficients). These last remarks also apply to integer expression swell arithmetic.

Worst case expression swell arithmetic is an attempt to effectively systematize bounding calculations for a certain class of problems, while providing a great deal of flexibility in their application (e.g., the elements in a matrix need not be considered initially uniform in size in order to obtain a bound). In this article, only numerical computations were considered, but these or like techniques could also be applied to the coefficients and exponents of polynomial or other calculations. One has to be careful with worst (or best) case analyses for complicated algorithms, as what is worst case at one stage may be best case (or more usually, a mix) at another. Finally, these analyses are useful for not only bounding expression swell (and thus computer memory usage), but also for setting limits on CPU time consumption. For example, the time required to multiply n_1 by n_2 using the simplest algorithm is $O(\mathcal{N}_\beta(n_1) \cdot \mathcal{N}_\beta(n_2))$ (Akritas, 1988).

References

[Akr88] Alkiviadis G. Akritas, "A New Method for Computing Polynomial Greatest Common Divisors and Polynomial Remainder Sequences", *Numerische Mathematik*, Volume 52, 1988, 119–127.

[Bar68] Erwin H. Bareiss, "Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination", *Mathematics of Computation*, Volume 22, Number 103, July 1968, 565–578.

[Coh78] Daniel I. A. Cohen, *Basic Techniques of Combinatorial Theory*, John Wiley & Sons, 1978.

[Com88] Computer Aided Mathematics Group, *MACSYMA Reference Manual: Version 13*, Symbolics Inc., November 1988.

[Gen73] W. Morven Gentleman, "Least Squares Computations by Givens Transformations Without Square Roots", *Journal of the Institute of Mathematics and its Applications*, Volume 12, 1973, 329–336.

[Gro87] Eric Grosse and Cleve Moler, "Underflow Can Hurt", *SIAM News*, Volume 20, Number 6, November 1987, p. 1.

[Ham74] Sven Hammarling, "A Note on Modifications to the Givens Plane Rotation", *Journal of the Institute of Mathematics and its Applications*, Volume 13, 1974, 215–218.

[Joh81] Lee W. Johnson and R. Dean Riess, *Introduction to Linear Algebra*, Addison-Wesley Publishing Company, 1981.

[Knu81] Donald E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Second Edition, Addison-Wesley Publishing Company, 1981.

[Mol77] C. B. Moler and J. H. Wilkinson, *Fast Givens Methods for Matrix Eigenvalue Problems*, 1977 (draft).

[Sto72] A. P. Stone, "Induced Transformations on Exterior Product Spaces", *Tensor, N.S.*, Volume 23, 1972, 147–150.

[Wes92] Michael James Wester, *Symbolic Calculation and Expression Swell Analysis of Matrix Determinants and Eigenstuff*, Ph.D. dissertation, University of New Mexico, Albuquerque, New Mexico, December 1992.