Order of Magnitude Reasoning using Logarithms*

P. Pandurang Nayak
AI Research Branch, Mail Stop 269-2
NASA Ames Research Center
Moffett Field, CA 94035.
nayak@ptolemy.arc.nasa.gov

Abstract

Converting complex equations into simpler, more tractable equations usually involves approximation. Approximation is usually done by identifying and removing insignificant terms, while retaining significant ones. The significance of a term can be determined by order of magnitude reasoning. In this paper we describe NAPIER, an implemented order of magnitude reasoning system. NAPIER defines the order of magnitude of a quantity on a logarithmic scale, and uses a set of rules to propagate orders of magnitudes through equations. A novel feature of NAPIER is the way it handles non-linear simultaneous equations, using linear programming in conjunction with backtracking. We show that order of magnitude reasoning in NAPIER is, in general, intractable and then discuss an approximate reasoning technique that allows it to run fast in practice. Some of NAPIER's inference rules are heuristic, and we estimate the error introduced by their use.

1 INTRODUCTION

Mathematical models are pervasive in science and engineering. Both analytical and numerical techniques have been used to solve the equations resulting from such models. Analytical methods are applicable only to restricted classes of equations (e.g., linear systems). To apply analytical techniques to more complex classes of equations, scientists and engineers have had to find ways of approximating the equations to convert them into a simpler form.

With the advent of fast digital computers, the classes of equations that can be solved by numerical (rather than analytical) methods have grown considerably. However, numerical methods are not a panacea; they have their limitations too. For example, solving systems of non-linear equations can be time consuming, and a good initial guess is required to ensure convergence to a solution. Hence, scientists and engineers still strive to identify appropriate approximations so that the resulting equations are as simple as possible. In addition, numerical methods are sometimes inapplicable. For example, during conceptual design, exact numerical values for exogenous quantities are usually unavailable because most of the details of the design are unspecified. Under such circumstances, an engineer needs to fall back on analytical methods, thereby highlighting the role of appropriate approximations.

The approximation process usually involves identifying and removing insignificant terms, while retaining only the significant ones. Consider the following example, previously discussed in [Bennett, 1987; Raiman, 1991], from the domain of acid-base chemistry. An important task in this domain is to find the concentration of H^+ ions in a solution. The concentration of ions in solution depends on the dynamic equilibrium resulting from competing chemical reactions. Consider dissolving an acid, AH, in water. The two reversible reactions that occur, corresponding to the ionization of AH and H_2O, are shown in Figure 1.

$$\begin{aligned} AH &\rightleftharpoons H^+ + A^- \\ H_2O &\rightleftharpoons H^+ + OH^- \end{aligned}$$

Figure 1: Ionization reactions that occur on dissolving AH in water

The equilibrium concentrations of the three ions (H^+, OH^-, A^-) and the acid (AH) are determined by the equations shown in Figure 2. Square brackets denote concentrations; C_a is the initial concentration of the acid; K_w is the ion product of water; and K_a is the ionization constant of the acid.

As has been pointed out in [Bennett, 1987; Raiman,

*This paper originally appeared in the Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR92).

Charge balance:	$[H^+] = [A^-] + [OH^-]$	(1)
Mass balance:	$C_a = [A^-] + [AH]$	(2)
Acid ionization equilibrium:	$K_a[AH] = [A^-][H^+]$	(3)
Water ionization equilibrium:	$K_w = [OH^-][H^+]$	(4)

Figure 2: Equilibrium equations for the ionization reactions.

1991], solving this set of equations analytically for $[H^+]$ results in a cubic equation which is difficult to solve. In fact, in problems involving *polyprotic* acids, i.e., acids that can yield more than one H^+ ion, the closed form solution for $[H^+]$ can involve equations of degree five or higher, making the solution significantly harder.

An alternative to the above approach is to approximate the equations, and hence simplify them. For example, a chemist might guess that the acid is strong, so that $[A^-] \gg [OH^-]$ and $[A^-] \gg [AH]$. This justifies reducing the first equation to $[H^+] = [A^-]$ and the second equation to $C_a = [A^-]$, leading to a straightforward solution.

The reasoning following the assumptions that $[A^-] \gg [OH^-]$ and $[A^-] \gg [AH]$ is very nicely formalized in [Raiman, 1991]. But how are these assumptions justified? In [Bennett, 1987], Bennett suggests that such assumptions are justified by *domain specific* inference rules. In this paper we present a *domain-independent* method for justifying such assumptions. In particular, we define the *order of magnitude* of a quantity on a logarithmic scale. We show how the order of magnitude of exogenous quantities like C_a, K_w, and K_a can be propagated through a set of equations like Equations 1–4 to compute orders of magnitudes of the remaining quantities like $[H^+]$, $[OH^-]$, $[A^-]$, and $[AH]$. The computed orders of magnitude of $[A^-]$, $[OH^-]$, and $[AH]$ can be used to justify the above assumptions and simplify the equations.

The reasoning technique described here has been implemented in a program called NAPIER.[1] NAPIER has been successfully tested on multiple examples in the domain of electromechanical devices, with the number of equations per example ranging from about 25 to a little over 150.

In the next section, we define the order of magnitude of a quantity and present rules for propagating orders of magnitude. We show that these rules can be used to infer orders of magnitudes of quantities related by a set of non-linear simultaneous equations. Next, we show that, in general, order of magnitude reasoning is intractable, and then present an approximate reasoning technique that make it efficient in practice. Since some of the order of magnitude reasoning rules are heuristic, we next estimate the error introduced by their use. We conclude with a discussion of related work.

[1] John Napier (1550–1617), a Scottish nobleman, is credited with the first discovery of logarithms.

2 ORDER OF MAGNITUDE REASONING IN NAPIER

Order of magnitude reasoning in NAPIER is a form of interval reasoning. The *order of magnitude* of a quantity q (denoted $om(q)$) is defined as follows:

$$om(q) = \lfloor \log_b |q| \rfloor \quad (5)$$

where the base, b, of the logarithm is chosen to be the smallest number that can be considered to be "much larger" than 1. The choice of b is clearly domain and task dependent. In this paper we assume that $b = 10$. Note that the order of magnitude of a quantity, q, is independent of its sign, and hence $om(q) = om(-q)$. In what follows, we assume that the signs of all quantities have been determined, to the extent possible, prior to any reasoning about orders of magnitude using standard constraint satisfaction techniques.[2]

2.1 INFERENCE RULES IN NAPIER

Given the orders of magnitude of q_1 and q_2, NAPIER computes bounds on the orders of magnitude of arithmetic expressions involving q_1 and q_2, using the rules shown in Figure 3. The rules for $(q_1 + q_2)$ and $(q_1 - q_2)$ assume that q_1 and q_2 have the same sign so that the magnitudes of q_1 and q_2 are actually being added or subtracted, respectively. The rule for $(q_1 \pm q_2)$ is applicable to a sum or difference of q_1 and q_2 when the sign of at least one of q_1 and q_2 is unknown.

The rules for $(q_1 * q_2)$ and (q_1/q_2) (rules 1 and 2) follow directly from Equation 5 and the rules of interval arithmetic [Moore, 1979]. For example, if $om(q_1) = n_1$ and $om(q_2) = n_2$, it follows that $b^{n_1} \leq |q_1| < b^{n_1+1}$ and $b^{n_2} \leq |q_2| < b^{n_2+1}$. Using interval arithmetic, we get $b^{n_1+n_2} \leq |q_1 * q_2| < b^{n_1+n_2+2}$, and hence $n_1 + n_2 \leq om(q_1 * q_2) \leq n_1 + n_2 + 1$.

Like rules 1 and 2, rules 3a and 4a are also based on Equation 5 and interval arithmetic. Note, however,

[2] This assumption is unnecessarily strong. For example, if a and b are positive, constraint satisfaction alone is unable to deduce the sign of $a-b$. However, if $om(a) > om(b)$, then $a - b$ can be deduced to be positive.

1. $om(q_1) + om(q_2) \leq om(q_1 * q_2) \leq om(q_1) + om(q_2) + 1$
2. $om(q_1) - om(q_2) - 1 \leq om(q_1/q_2) \leq om(q_1) - om(q_2)$
3. a) $om(q_1) \leq om(q_1 + q_2) \leq om(q_1) + 1$ if $om(q_1) = om(q_2)$
   b) $om(q_1 + q_2) = om(q_1)$ if $om(q_1) > om(q_2)$
   c) $om(q_1 + q_2) = om(q_2)$ if $om(q_1) < om(q_2)$
4. a) $om(q_1 - q_2) \leq om(q_1)$ if $om(q_1) = om(q_2)$
   b) $om(q_1 - q_2) = om(q_1)$ if $om(q_1) > om(q_2)$
   c) $om(q_1 - q_2) = om(q_2)$ if $om(q_1) < om(q_2)$
5. a) $om(q_1 \pm q_2) \leq om(q_1) + 1$ if $om(q_1) = om(q_2)$
   b) $om(q_1 \pm q_2) = om(q_1)$ if $om(q_1) > om(q_2)$
   c) $om(q_1 \pm q_2) = om(q_2)$ if $om(q_1) < om(q_2)$

Figure 3: Rules for order of magnitude reasoning

that these rules predict larger intervals for $(q_1 + q_2)$ and $(q_1 - q_2)$ than interval arithmetic predicts under the same restrictions on q_1 and q_2. For example, if $om(q_1) = om(q_2) = n$, then interval arithmetic predicts that $(q_1 + q_2)$ is bounded by $2b^n$ and $2b^{n+1}$, while NAPIER predicts $n \leq om(q_1 + q_2) \leq n + 1$, which is equivalent to saying that $(q_1 + q_2)$ lies between b^n and b^{n+2}. This larger interval is a consequence of NAPIER being able to represent only intervals whose end points are integer powers of the chosen base. Further note that rules 3a and 4a are correct only if the base is greater than 2. This is reasonable given our heuristic for selecting the base (viz., 2 is unlikely to be considered to be "much larger" than 1).

Unlike the rules discussed thus far, rules 3b, 3c, 4b, and 4c are not guaranteed to be correct, but are heuristic rules. They are all based on the intuition that adding or subtracting a "small" quantity from a "large" quantity does not significantly affect the larger quantity. Since the base in Equation 5 is chosen as the smallest number that can be considered to be "much larger" than 1, the above intuition justifies these rules; the order of magnitude of a quantity is not affected by adding or subtracting quantities of a smaller order of magnitude. The inclusion of these heuristic order of magnitude rules differentiates NAPIER from standard interval reasoners. In Section 5, we estimate the error introduced by the use of these heuristic rules.

Finally, rule set 5 merely encompasses both rule sets 3 and 4. It is used to infer the order of magnitude of a sum or difference of two quantities when the signs of at least one of the two quantities is not known. To determine the order of magnitude of a sum or difference of two quantities, NAPIER selects the appropriate rule set from rule sets 3, 4, and 5, depending on the operation (sum or difference) and the signs of the two quantities. For example, consider the equation $q_3 = q_1 + q_2$. If q_1 and q_2 have the same sign, then rule set 3 is used to infer $om(q_3)$; if q_1 and q_2 have opposite signs, then rule set 4 is used to infer $om(q_3)$, since the magnitude of q_3 is really the difference of the magnitudes of q_1 and q_2; and if the signs of at least one of q_1 and q_2 is unknown, then rule set 5 is used to infer $om(q_3)$.

2.2 SET OF SIMULTANEOUS EQUATIONS

Until now, we have focussed exclusively on how NAPIER uses a single equation to propagate orders of magnitudes, i.e., how $om(q_1 op\ q_2)$ is computed from $om(q_1)$ and $om(q_2)$. However, the rules in Figure 3 can also be used to compute orders of magnitudes of quantities related by a set of (possibly non-linear) simultaneous equations. NAPIER uses these rules to convert a set of simultaneous equations into a set of constraints, where each constraint is a disjunction of a set of linear inequalities. Each equation in the set of simultaneous equations contributes a constraint as follows:

1. Product and quotient terms contribute a single set of linear inequalities according to rules 1 and 2, respectively. For example, $q_3 = q_1 * q_2$ contributes the following set:
$$\{om(q_1) + om(q_2) \leq om(q_3),\ om(q_3) \leq om(q_1) + om(q_2) + 1\}$$
2. Sum and difference terms contribute a disjunction of three sets of linear inequalities, using rule sets 3, 4, or 5, as applicable. Each disjunct corresponds to one of the rules (a, b, or c) in the applicable rule set. For example, assuming that q_1 and q_2 have the same sign, the equation $q_3 = q_1 - q_2$ contributes the following disjunction:[3]
$$\{om(q_3) \leq om(q_1), om(q_1) = om(q_2)\}$$
$$\vee$$
$$\{om(q_3) = om(q_1), om(q_1) \geq om(q_2) + 1\}$$
$$\vee$$
$$\{om(q_3) = om(q_2), om(q_1) \leq om(q_2) - 1\}$$
corresponding to rules 4a, 4b, and 4c, respectively.

[3]Since the order of magnitudes are integral, $om(q_1) > om(q_2)$ is equivalent to $om(q_1) \geq om(q_2) + 1$.

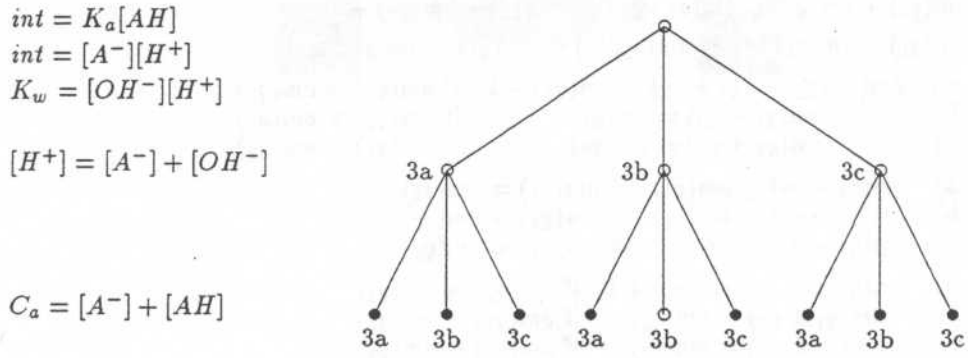

Figure 4: A backtrack tree.

NAPIER uses this set of constraints to compute bounds on the orders of magnitudes of the quantities. Since all the inequalities in the constraints are linear inequalities, NAPIER uses linear programming [Hillier and Lieberman, 1980], in conjunction with backtracking, to compute order of magnitude bounds. Backtracking is necessary to handle the disjunctions. We describe this algorithm next.

2.3 BACKTRACKING ALGORITHM

Let E denote the set of simultaneous equations being processed. NAPIER's backtracking procedure is best visualized as a depth-first traversal of a backtrack tree. Each level in the tree (except the root level) corresponds to one of the sum or difference terms in E. The root level corresponds to all the product and quotient terms in E. Each internal node has three children, corresponding to the three disjuncts in the constraint contributed by the sum or difference term at the level of the node's children. Each node in the tree has an associated set of linear inequalities defined as follows:

1. The set of inequalities at the root node consists of the union of the sets of inequalities contributed by each product and quotient term in E.
2. The set of inequalities at each non-root node consists of the union of (a) the inequalities at the node's parent; and (b) the inequalities in the disjunct associated with that node.

Starting at the root node, NAPIER traverses the backtrack tree in a depth-first manner. At each node it checks the consistency of the inequalities at that node. If the set is inconsistent, it immediately backtracks to the node's parent. If the set is consistent and it is a non-leaf node, it continues its depth-first traversal. If the set is consistent and it is a leaf node, it uses the inequalities to find the maximum and minimum values of the order of magnitude of each quantity. The bounds computed at each of the consistent leaf nodes are combined so that the lower bound of each quantity is the least lower bound and the upper bound is the greatest upper bound.

Since the inequalities at each node are linear, NAPIER uses the Simplex linear programming algorithm [Hillier and Lieberman, 1980; Press *et al.*, 1989] to check their consistency, and to compute the order of magnitude bounds at leaf nodes. However, from Equation 5 it follows that the order of magnitude of a quantity is integral. Hence, instead of using linear programming, NAPIER should use integer programming [Hillier and Lieberman, 1980]. Unfortunately, integer programming is known to be intractable [Karp, 1972], which leads to severe restrictions on the number of equations and the size of the backtrack tree that can be handled. Hence, to avoid such restrictions, NAPIER uses linear programming.

It is important to note that, while bounds computed by linear programming are not guaranteed to be tight[4], they are guaranteed to be correct: upper bounds will be greater than or equal to integer programming upper bounds, and lower bounds will be less than or equal integer programming lower bounds. In addition, we have found that, in practice, linear programming bounds are usually integral, in which case there is no loss of solution quality.

2.4 EXAMPLE

We now illustrate the above procedure using Equations 1–4. Let us assume that the exogenous orders of magnitude are as follows: $om(K_w) = -14$, $om(K_a) = -2$, $om(C_a) = -5$. This corresponds to a moderately strong solution of a strong acid. The backtrack tree resulting from these equations is shown in Figure 4. The equations associated with each level are shown on the left of the tree. Note that Equation 3 had to be split into two product expressions, with the introduction of an intermediate variable int. The rules (and hence the

[4] A bound b_1, interpreted as an interval, is said to be *tight* with respect to a bound b_2 if b_1 and b_2 are identical. b_1 is *looser* than b_2 if b_1 contains b_2.

disjuncts) associated with each non-root node are displayed near each node. Nodes that are filled in are the inconsistent nodes. For example, the left most leaf node can be seen to be inconsistent using the following line of reasoning. Applying rule 3a to Equations 1 and 2, we get

$$om([OH^-]) = om([A^-]) = om([AH])$$
$$om([A^-]) \leq om(C_a) \leq om([A^-]) + 1$$
$$om([A^-]) \leq om([H^+]) \leq om([A^-]) + 1$$

Since $om(C_a) = -5$, it follows that the least value of $om([A^-])$ is -6. Hence the least values of $om([OH^-])$ and $om([H^+])$ are also -6, and hence the least value of $om([OH^-][H^+])$ is -12. But rule 1 applied to Equation 4 requires that:

$$om([OH^-][H^+]) = om(K_w) = -14$$

which leads to a contradiction.

Of course, NAPIER doesn't need the above line of reasoning to infer inconsistencies; it reaches the same conclusion using linear programming.

The only consistent set of inequalities at the leaf nodes is the middle most leaf node, corresponding to assuming that $om([A^-]) > om([OH^-])$ and $om([A^-]) > om([AH])$. The quantity bounds calculated at this node are as follows:[5]

$$\begin{aligned} om([H^+]) &= -5 \\ om([OH^-]) &= (-10, -9) \\ om([AH]) &= (-9, -7) \\ om([A^-]) &= -5 \end{aligned}$$

Since $[A^-]$ is at least two orders of magnitude greater than $[AH]$, and at least four orders of magnitude greater than $[OH^-]$, a chemist is justified in making the assumptions that $[A^-] \gg [OH^-]$ and $[A^-] \gg [AH]$. These assumptions can then be used to simplify the equations, as discussed earlier.

A slight variation of the above example illustrates the importance of having such justifications. Suppose that, instead of having $om(C_a) = -5$, we had $om(C_a) = -8$. This corresponds to a weak solution of the same strong acid. Using this new value for $om(C_a)$, NAPIER predicts the following bounds on the orders of magnitude:

$$\begin{aligned} om([H^+]) &= -7 \\ om([OH^-]) &= (-8, -7) \\ om([AH]) &= (-14, -12) \\ om([A^-]) &= -8 \end{aligned}$$

These values justify the assumption that $[A^-] \gg [AH]$, but the other assumption, $[A^-] \gg [OH^-]$, is seen to be completely unjustified. This means that only Equation 2 can be simplified. Hence, NAPIER is a useful tool in justifying the order of magnitude assumptions that scientists and engineers make in simplifying equations.

In addition to its role in justifying order of magnitude assumptions, NAPIER's predictions can also be used directly. For example, if all the chemist is interested in is the approximate pH of the solution[6], then NAPIER's predictions can be used directly: in the first case, the pH is between 5 and 4; in the second case, the pH is between 7 and 6. Note that NAPIER was able to make these predictions using approximate values of C_a, K_w, and K_a. This feature makes it particularly useful during conceptual design.

3 ORDER OF MAGNITUDE REASONING IS INTRACTABLE

The backtracking algorithm described in the previous section, generates a tree whose worst case size is exponential in the number of sum and difference expressions. In this section we show that order of magnitude reasoning using the rules in Figure 3 is intractable, even if orders of magnitude are not required to be integral. This means that NAPIER can do little better than generate a backtrack tree whose worst case size is exponential.

We start by defining the decision problem corresponding to finding the maximum order of magnitude of a quantity:

Definition 1 *(*ORDER OF MAGNITUDE REASONING*) Let E be a set of equations, and let V be the set of quantities used in E. Let $X \subseteq V$ be the set of exogenous quantities, with known orders of magnitude. Let $q \in V$ be a quantity and let B be an integer. Let $s : V \rightarrow \{+, -, unknown\}$ be a function that assigns signs to the quantities in V. (Quantities with unknown signs are assigned "unknown.") Assuming that the order of magnitude of a quantity is not required to be integral, is the maximum value of $om(q)$, derived using the rules in Figure 3 on the set E, greater than or equal to B?*

We now state the following theorem without detailed proof:

Theorem 1 *The* ORDER OF MAGNITUDE REASONING *problem is NP-complete.*[7]

The proof of this theorem is based on a reduction from an arbitrary instance of 3SAT. Briefly, the reduction introduces a quantity for each literal in the instance of 3SAT. Equations are added to ensure that quantities corresponding to complementary literals have the

[5] $om(q) = (l, u)$ represents the fact that $l \leq om(q) \leq u$

[6] The pH of a solution is defined to be $-\log_{10}[H^+]$.

[7] See [Garey and Johnson, 1979] for a comprehensive introduction to the theory of intractability.

Example number	Number of equations	Number of +/− terms	Time (sec) on an Explorer II	
			All equations	With causal ordering
1	28	11	2733	2.0
2	31	11	2435	1.0
3	45	14	–	2.9
4	60	24	–	2.7
5	80	25	–	37.2
6	110	32	–	35.9
7	111	32	–	94.6
8	119	35	–	20.4
9	145	43	–	45.2
10	163	50	–	21.0

Table 1: NAPIER's run times with and without causal ordering.

property that the order of magnitude of one of them must be 0 and the order of magnitude of the other one must be 1. The mapping between truth assignments and orders of magnitudes is straightforward: a literal is true if and only if the corresponding quantity's order of magnitude is 1. Additional equations involving the above quantities and the special quantity q are then introduced, and the bound B is defined to ensure that the maximum value of $om(q)$ is greater than or equal to B if and only if all the clauses are satisfied. See [Nayak, 1992] for details.

Assuming $P \neq NP$, Theorem 1 tells us that, in the worst case, NAPIER will have to generate a backtrack tree whose size is exponential in the number of sum and difference terms. Unfortunately, the exponential blow up does occur in practice. We have used NAPIER in the domain of electromechanical devices, as part of the automated model selection system described in [Nayak *et al.*, 1992]. Table 1 summarizes NAPIER's performance on models of ten different devices (see [Nayak, 1992] for a description of the devices).

The second column in this table shows the total number of equations in each example, while the third column shows the the total number of sum and difference terms. The fourth column shows the time it took NAPIER to run its backtracking algorithm on the complete set of equations.[8] NAPIER was given a maximum of one hour to solve each example; a "–" entry in column four denotes that NAPIER could not solve the example in an hour. As is clear from the table, only the two smallest examples could be solved in under an hour, each taking over 40 minutes. Hence, NAPIER appears to be quite impractical, except for the smallest examples. To make it practical, we now develop an approximate reasoning scheme for NAPIER that trades off accuracy for speed.

[8]The fifth column will be discussed in the next section.

4 APPROXIMATION ALGORITHMS IN NAPIER

The backtrack tree developed by NAPIER is, in the worst case, exponential in the number of sum and difference terms in the set of equations under consideration. Hence, to make NAPIER practically useful, it is important to decrease the number of sum and difference terms that are handled at any one time. We now discuss a method for doing this, based on a *dependency ordering* of the equations.

4.1 ORDERING THE EQUATIONS

The dependency ordering of equations that we consider is the *causal ordering*, described in [Iwasaki and Simon, 1986]. The causal ordering specifies the order in which equations are to be solved, and identifies minimal sets of equations that *must* be solved simultaneously. The causal ordering can be viewed as a directed acyclic graph. Each node in the graph consists of a set of equations that must be solved simultaneously. There is an edge from node n_1 to node n_2 if the equations at n_2 use a quantity whose value is determined by the equations at n_1.

NAPIER processes the equation sets in the order specified by the causal ordering: equation sets earlier in the ordering are processed first. NAPIER bounds the orders of magnitudes of the quantities used in an equation set, and uses these bounds as exogenous bounds for equation sets later in the ordering.

The use of the above dependency ordering has a significant computational advantage. A large set of equations, with many sum and difference terms, can often be broken down into many small sets of equations, with each equation set having very few sum and difference terms. Hence, NAPIER can process each equation set in the dependency ordering very fast. Column five in Table 1 shows the time it took NAPIER to solve the ten examples using causal ordering. It takes NAPIER from a few seconds to under two minutes to solve each

Example #	Δ_{max}	Example #	Δ_{max}
1	11	6	7
2	11	7	4
3	9	8	5
4	7	9	6
5	7	10	6

Table 2: Maximum value of Δ for each example.

of these examples, showing that causal ordering has made NAPIER practical for large sets of equations.

4.2 LOSS OF ACCURACY

The drawback of using the dependency ordering is that global constraints can be lost, leading to excessively loose bounds on the orders of magnitudes. Consider, for example, the set $\{y_1 = x_1 * x_2, y_2 = x_3/y_1, y_3 = y_1 * y_2\}$, and let x_1, x_2, and x_3 be exogenous with orders of magnitude 0. The dependency ordering generated from this set of equations is:

$$\begin{array}{ccc} & \{y_2 = x_3/y_1\} & \\ \nearrow & & \searrow \\ \{y_1 = x_1 * x_2\} & \longrightarrow & \{y_3 = y_1 * y_2\} \end{array}$$

Using this dependency ordering, NAPIER computes the order of magnitude of y_3 as follows: from the first equation it computes $om(y_1)$ to be between 0 and 1; from the second equation, and the calculated bound on $om(y_1)$, it computes $om(y_2)$ to be between -2 and 0; and from the third equation and the calculated bounds on $om(y_1)$ and $om(y_2)$, it computes $om(y_3)$ to be between -2 and 2. However, if all three equations were considered simultaneously, NAPIER computes $om(y_3)$ to be between -1 and 1.

The reason for the looser bound in the first case stems from not enforcing some global constraints. For example, the lower bound of $om(y_3)$ can be -2 only when $om(y_1) = 0$ and $om(y_2) = -2$. However, when $om(y_1)$ is 0, the second equation dictates that the lowest that $om(y_2)$ can be is -1. This fact is lost when the third equation is processed by itself.

More generally, the above problem occurs when a quantity, like y_3, depends on two or more quantities, like y_1 and y_2, whose values have been determined by equations that are earlier in the causal ordering. In using these previously determined values, NAPIER disregards any additional constraints that might hold between those values. Hence, bounds computed based on these values may not be as tight as possible.

NAPIER can partially address this problem by combining adjacent sets of equations in the dependency ordering. This allows more equations to be handled simultaneously, so that more global constraints can be incorporated. However, combining adjacent sets of equations can lead to an increase in the number of sum and difference terms that must be handled simultaneously. Hence, adjacent sets are combined only when the number of sum and difference terms in the resulting set does not increase beyond a threshold (call this threshold Δ).

Combining adjacent sets of equations, as described above, also allows us to partially empirically evaluate the effect of causal ordering on accuracy. We ran NAPIER a number of times on each of our examples, using increasing values of Δ, allowing a maximum of 15 minutes per run. Table 2 shows the maximum value of Δ used for each example. We then compared the bounds that were computed without combining adjacent sets with the bounds that were computed with the maximum setting of Δ. Interestingly, we found that there was *no* loss of accuracy—the bounds computed with and without combining adjacent sets were identical.

To understand the reason for this somewhat surprising result, we now analyze the source of the additional constraints on previously determined values. Let us assume that $om(p_3)$ is computed using previously computed values of $om(p_1)$ and $om(p_2)$. Additional constraints on the values of $om(p_1)$ and $om(p_2)$ stem from one of two sources: (a) $om(p_1)$ and $om(p_2)$ are determined simultaneously; and (b) the value of $om(p_1)$ is used in computing the value of $om(p_2)$, i.e., the values of one of these quantities depends on the value of the other. Point (a) manifests itself as a node in the causal ordering which contains more than one equation. Point (b) manifests itself as multiple paths between two nodes in the causal ordering.

Hence, if the causal ordering, viewed as a graph, satisfies the following two properties:

1. each node contains exactly one equation; and
2. there is at most one path between any two nodes;

then we can show that there will be *no* additional constraints between previously determined values. Hence, there is no loss of accuracy in using the causal ordering.

Table 3 shows how closely the causal orderings generated from our examples match the above two properties. The second and third columns of this table show the maximum and average number of equations per node, respectively. One can see that, in all cases, the

Example number	Equations per node		# of extra edges
	Maximum	Average	
1	7	1.27	1
2	7	1.24	0
3	7	1.15	1
4	1	1.00	0
5	12	1.29	1
6	18	1.29	2
7	17	1.26	6
8	9	1.25	2
9	18	1.21	3
10	16	1.10	0

Table 3: Properties of the causal ordering graph

average number of equations per node is very close to 1. The fourth column shows the minimum number of edges that must be removed from the causal ordering to ensure that there is at most one path between any two nodes. One can see that, in most cases these numbers are very small. Hence, the above analysis provides us with some insight into the reasons underlying the fact that, in our examples, the bounds computed with and without combining adjacent sets are identical.

5 ERROR ESTIMATION

In this section, we estimate the error introduced by the use of the heuristic rules introduced in Section 2.1. We then analyze some alternate order of magnitude rules that seem intuitively plausible, and show that these rules introduce unacceptably large errors. The analysis is done using probability theory and is based on interpreting each quantity as a *random variable*.[9]

5.1 ESTIMATING THE ERROR OF HEURISTIC RULES

We start by analyzing rule 3b. Let Q, Q_1, and Q_2 be quantities such that $Q = Q_1 + Q_2$. Let f_{Q_1} and f_{Q_2} be the probability density functions of Q_1 and Q_2, respectively, and let f_{Q_1,Q_2} be their joint probability density function. (Briefly, $f_{Q_1}(q_1)$ is the probability that Q_1 lies between q_1 and $q_1 + dq_1$, and $f_{Q_1,Q_2}(q_1, q_2)$ is the probability that Q_1 lies between q_1 and $q_1 + dq_1$, and Q_2 lies between q_2 and $q_2 + dq_2$.) Since $Q = Q_1 + Q_2$, it follows that the probability that Q lies between l and u, for any values l and u, is:

$$Prob\{l \leq Q < u\} = \int_{-\infty}^{\infty} \int_{l-q_1}^{u-q_1} f_{Q_1,Q_2}(q_1, q_2) dq_2 dq_1 \quad (6)$$

Let us now assume that $om(Q_1) = n_1$ and $om(Q_2) = n_2$, with $n_1 > n_2$. Under these conditions, rule 3b states that $om(Q) = n_1$, i.e., $b^{n_1} \leq Q < b^{n_1+1}$. To estimate the error, ϵ(Rule 3b), in rule 3b, we must calculate the probability that Q lies outside the region from b^{n_1} to b^{n_1+1}:

$$\begin{aligned} &\epsilon(\text{Rule 3b}) \\ &= 1 - Prob\{b^{n_1} \leq Q < b^{n_1+1}\} \\ &= 1 - \int_{-\infty}^{\infty} \int_{b^{n_1}-q_1}^{b^{n_1+1}-q_1} f_{Q_1,Q_2}(q_1, q_2) dq_2 dq_1 \quad (7) \end{aligned}$$

To evaluate this integral, we make the following assumptions:

Assumption 1: Q_1 and Q_2 are independent random variables. Hence, the joint probablity density of Q_1 and Q_2 is just the product of the individual probablity densities:

$$f_{Q_1,Q_2}(q_1, q_2) = f_{Q_1}(q_1) f_{Q_2}(q_2) \quad (8)$$

Assumption 2: Q_1 and Q_2 are uniformly distributed on the intervals $[b^{n_1}, b^{n_1+1})$ and $[b^{n_2}, b^{n_2+1})$, respectively:

$$f_{Q_1}(q_1) = \begin{cases} \frac{1}{b^{n_1+1} - b^{n_1}} & \text{if } b^{n_1} \leq q_1 < b^{n_1+1} \\ 0 & \text{otherwise} \end{cases}$$

$$f_{Q_2}(q_2) = \begin{cases} \frac{1}{b^{n_2+1} - b^{n_2}} & \text{if } b^{n_2} \leq q_2 < b^{n_2+1} \\ 0 & \text{otherwise} \end{cases}$$

Using these assumptions, we get the following result (see [Nayak, 1992] for details):

$$\epsilon(\text{Rule 3b}) = \frac{b+1}{2b^{n_1-n_2}(b-1)} \quad (9)$$

Hence, under Assumptions 1 and 2, the error in rule 3b is maximum when $(n_1 - n_2)$ is minimum, i.e., $(n_1 - n_2) = 1$, which occurs when quantities of consecutive orders of magnitude are being added. When $b = 10$, the maximum error is 6.11%. The error in rule 4b can also be shown to be $(b+1)/2b^{n_1-n_2}(b-1)$ in a similar way.

[9] See [Davenport, 1970] for an introduction to probability theory and random variables.

5.2 ALTERNATE ORDER OF MAGNITUDE RULES

The above error estimation techniques can also be used to analyze the alternate order of magnitude reasoning rules shown in Figure 5. We have selected these rules because they seem intuitively very appealing. However, while these rules may appear intuitively appealing, they are also unacceptably error-prone.

1′) $om(q_1 * q_2) = om(q_1) + om(q_2)$

2′) $om(q_1/q_2) = om(q_1) - om(q_2)$

3a′) $om(q_1 + q_2) = om(q_1)$ if $om(q_1) = om(q_2)$

Figure 5: Alternate rules for order of magnitude reasoning.

In particular, we can use the techniques in the previous section to show that:

$$\epsilon(\text{Rule } 1') = 1 - (b \ln b - b + 1)/(b-1)^2 \quad (10)$$

$$\epsilon(\text{Rule } 2') = 1/2 \quad (11)$$

$$\epsilon(\text{Rule } 3a') = 1 - (b-2)^2/2(b-1)^2 \quad (12)$$

Substituting $b = 10$ into the above equations tells us that the error in rule 1′ is 82.68%, the error in rule 2′ is 50%, and the error in rule 3a′ is 60.49%. We believe that these errors are unacceptably large, and hence have chosen not to include these rules in NAPIER.

6 RELATED WORK

Order of magnitude reasoning has been widely studied in AI. Murthy [Murthy, 1988] was the first to propose the use of a logarithmic scale for the order of magnitude of a quantity. In that paper, he also provides rules of inference to infer new orders of magnitude from old ones. Some of these rules are similar to ours. For example, he includes rules 3b, 3c, 4b, and 4c. However, instead of 1, he proposes the rule $om(q_1 * q_2) = om(q_1) + om(q_2)$ (which is rule 1′), and instead of rule 3a, he proposes the rule $om(q_1 + q_2) = om(q_1)$ when $om(q_1) = om(q_2)$ (which is rule 3a′). As we saw in Section 5.2, the estimated error in these rules is too large, and hence we have chosen not to include them in NAPIER. Unlike our work, Murthy provides no analysis of how his inference rules can be used to find the order of magnitudes of quantities related by sets of simultaneous equations. In addition, we also analyze the complexity of order of magnitude inference, and present an approximate reasoning technique that works well in practice.

Raiman [Raiman, 1991; Raiman, 1986] explores the foundations of symbolic order of magnitude reasoning. He defines a variety of order of magnitude scales, such as *Close* and *Comparable*, built out of the basic order of magnitude granularities, *Small* and *Rough*. He introduces ESTIMATES, a system to solve order of magnitude equations. The primary difference between NAPIER and ESTIMATES is one of emphasis: NAPIER can be viewed as providing justifications for making order of magnitude assumptions; ESTIMATES can be viewed as a formalization of the use of such order of magnitude assumptions to symbolically manipulate and simplify equations.

Order of magnitude reasoning in the O(M) formalism [Mavrovouniotis and Stephanopolous, 1987] uses a parameter e to represent the largest quantity that can be considered to be "much smaller" than 1. This is analogous to the parameter b in NAPIER (i.e., $b = 1/e$). However, there are a number of differences between O(M) and NAPIER. First, the O(M) formalism is based on order of magnitude relations *between* quantities. Hence, it works best when equations involve only *links* (links are ratios of quantities). NAPIER, on the other hand, is based on the order of magnitudes of the quantities themselves, and hence works with any algebraic equations. This is advantageous because it is not always possible to convert equations into equations involving only links. Second, O(M) requires equations to be converted into *assignments*, which allow a new relation or range to be inferred from already known relations. This is a serious restriction since equations can be converted to assignments only in the absence of simultaneous equations. As we have seen, NAPIER does not have this restriction.

NAPIER is also related to interval reasoning discussed in [Moore, 1979; Simmons, 1986; Sacks, 1987]. NAPIER can be viewed as interval reasoning in which the end points of the interval are restricted to a particular set of points of the form b^n, with specified base b, and any integer n. The drawback of this restriction is that under certain conditions, compared to interval reasoning, the bounds inferred by NAPIER are unnecessarily loose (e.g., see the discussion of rule 3a in Section 2.1). The advantage of this restriction is that, unlike traditional interval reasoners, NAPIER is able to use sets of non-linear simultaneous equations to infer quantity bounds. In addition, the ability to simultaneously process all the equations in a set allows NAPIER to exploit global constraints to compute tighter bounds (see Section 4). Another distinguishing characteristic of NAPIER, which classifies it as an order of magnitude reasoning system rather than just an interval reasoner, is the use of heuristic rules (e.g, rule 3b).

7 CONCLUSIONS

In this paper we described an implemented order of magnitude reasoning system called NAPIER. NAPIER defines the order of magnitude of a quantity on a logarithmic scale and uses a set of rules to propagate order of magnitudes through equations. A novel fea-

ture of NAPIER is its handling of non-linear simultaneous equations. Since the order of magnitude reasoning rules are all disjunctions of linear inequalities, NAPIER is able to use linear programming, in conjunction with backtracking, to find bounds on the order of magnitudes of quantities related by sets of non-linear simultaneous equations.

We also showed that order of magnitude reasoning using NAPIER's rules is intractable. Hence, NAPIER uses an approximate reasoning technique, based on causal ordering, leading to a practically useful system. This approximate reasoning technique trades off accuracy for speed, though in practice there does not appear to be any loss of accuracy.

Some of NAPIER's rules are heuristic rules, and we have estimated the error introduced by the use of these rules. We have also shown that intuitively appealing alternate heuristic rules lead to large estimated errors.

NAPIER has been extensively used in an automated model selection system described in [Nayak *et al.*, 1992]. We believe that NAPIER will find wide spread applications in different aspects of engineering and scientific problem solving.

Acknowledgements

I would like to thank Richard Fikes, Pat Hayes, and Leo Joskowicz for useful discussions on order of magnitude reasoning, and for comments on earlier drafts of this paper. Thanks also to the two anonymous reviewers, whose detailed comments helped improve the paper. Pandurang Nayak was supported by an IBM Graduate Technical Fellowship. Additional support for this research was provided by the Defense Advanced Research Projects Agency under NASA Grant NAG 2-581 (under ARPA order number 6822), by NASA under NASA Grant NCC 2-537, and by IBM under agreement number 14780042.

References

[Bennett, 1987] Bennett, Scott W. 1987. Approximation in mathematical domains. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Los Altos, CA. International Joint Conferences on Artificial Intelligence, Inc., Morgan Kaufmann Publishers, Inc. 239–241.

[Davenport, 1970] Davenport, Wilbur B. Jr. 1970. *Probability and Random Processes.* McGraw-Hill Book Company.

[Garey and Johnson, 1979] Garey, Michael R. and Johnson, David S. 1979. *Computers and Intractability.* W. H. Freeman and Company.

[Hillier and Lieberman, 1980] Hillier, Fredrick S. and Lieberman, Gerald J. 1980. *Introduction to Operations Reesearch.* Holden-Day, Inc., third edition.

[Iwasaki and Simon, 1986] Iwasaki, Yumi and Simon, Herbert A. 1986. Causality in device behavior. *Artificial Intelligence* 29:3–32.

[Karp, 1972] Karp, Richard M. 1972. Reducibility among combinatorial problems. In Miller, R. E. and Thatcher, J. W., editors 1972, *Complexity of Computer Computations.* Plenum Press, New York. 85–103.

[Mavrovouniotis and Stephanopolous, 1987] Mavrovouniotis, M. and Stephanopolous, G. 1987. Reasoning with orders of magnitude and approximate relations. In *Proceedings of the Sixth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence.

[Moore, 1979] Moore, Ramon E. 1979. *Methods and Applications of Interval Analysis.* SIAM Studies in Applied Mathematics. SIAM, Philadelphia.

[Murthy, 1988] Murthy, Seshashayee S. 1988. Qualitative reasoning at multiple resolutions. In *Proceedings of the Seventh National Conference on Artificial Intelligence.* American Association for Artificial Intelligence. 296–300.

[Nayak *et al.*, 1992] Nayak, P. Pandurang; Joskowicz, Leo; and Addanki, Sanjaya 1992. Automated model selection using context-dependent behaviors. In *Proceedings of the Tenth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence. 710–716.

[Nayak, 1992] Nayak, P. Pandurang 1992. *Automated Modeling of Physical Systems.* Ph.D. Dissertation, Stanford University, Department of Computer Science, Stanford, CA.

[Press *et al.*, 1989] Press, William H.; Flannery, Brian P.; Teukolsky, Saul A.; and Vetterling, William T. 1989. *Numerical Recipes in Pascal: The Art of Scientific Computing.* Cambridge University Press.

[Raiman, 1986] Raiman, Olivier 1986. Order of magnitude reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence. 100–104.

[Raiman, 1991] Raiman, Olivier 1991. Order of magnitude reasoning. *Artificial Intelligence* 51:11–38.

[Sacks, 1987] Sacks, Elisha 1987. Hierarchical reasoning about inequalities. In *Proceedings of the Sixth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence, Morgan Kaufmann Publishers, Inc. 649–654.

[Simmons, 1986] Simmons, Reid 1986. "Commonsense" arithmetic reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence.* American Association for Artificial Intelligence. 118–124.