Numeric-Symbolic Exact Rational Linear System Solver*

B. David Saunders, David Harlan Wood, and Bryan S. Youse
Dept. of Computer and Information Sciences, University of Delaware
Newark, Delaware, USA
saunders@udel.edu, wood@udel.edu, bryouse@udel.edu

ABSTRACT

An iterative refinement approach is taken to rational linear system solving. Such methods produce, for each entry of the solution vector, a rational approximation with denominator a power of 2. From this the correct rational entry can be reconstructed. Our iteration is a numeric-symbolic hybrid in that it uses an approximate numeric solver at each step together with a symbolic (exact arithmetic) residual computation and symbolic rational reconstruction. The rational solution may be checked symbolically (exactly). However, there is some possibility of failure of convergence, usually due to numeric ill-conditioning. Alternatively, the algorithm may be used to obtain an extended precision floating point approximation of *any* specified precision. In this case we cannot guarantee the result by rational reconstruction and an exact solution check, but the approach gives evidence (not proof) that the probability of error is extremely small. The chief contributions of the method and implementation are (1) confirmed continuation, (2) improved rational reconstruction, and (3) faster and more robust performance.

Categories and Subject Descriptors

G.4 [**Mathematical Software**]: Algorithm Design and Analysis; I.1.4 [**Symbolic and Algebraic Manipulation**]: Applications

General Terms

Algorithms, Design, Performance

Keywords

iterative refinement, rational linear system, rational reconstruction

*Research supported by National Science Foundation Grants CCF-0830130, CCF-108063

ISSAC'11, June 8–11, 2011, San Jose, California, USA.

1. INTRODUCTION

We address the problem of solving $Ax = b$ for a vector $x \in \mathbb{Q}^n$, given $A \in \mathbb{Q}^{m\times n}$ and $b \in \mathbb{Q}^m$. We will restrict ourselves to square ($m = n$), nonsingular matrices with integer entries of length d bits or fewer. This is the core problem. Also, in this paper we are concerned with dense matrices, which is to say, matrices that do not have so many zero entries that more specialized sparse matrix techniques should be applied. We do anticipate that the refined numeric-symbolic iterative approach presented here will also apply effectively to sparse systems.

We present a method which is a refinement of the numeric-symbolic method of Wan[27, 26]. Earlier work of Geddes and Zheng[10] showed the effectiveness of the numeric-symbolic iteration, but used higher precision (thus higher cost) steps in the residue computation. However, Wan's method has had only sporadic success as deployed in the field (in the LinBox[22] library). Here we present a new *confirmed continuation method* which is quite robust and effective. In general, numerical numerical iteration methods are intended to extend the number of correct digits in the partial solution x'. The confirmed continuation method verifies that these new digits overlap the previous iteration's partial solution. This is our assurance of progress, rather than the less reliable matrix condition number small norm of residual, $|b - Ax'|$, which is used in prior methods [27, 26, 20, 12]. Evidence from data suggests that the new version solves a larger class of problems, is more robust, and provides the fastest solutions for many dense linear systems.

Standard non-iterative methods such as Gaussian elimination in its various forms suffer from extreme expression swell when working in exact integer or rational number arithmetic. In fact, the solution vector x itself is typically a larger object than the inputs. When elimination is used, typically $O(n^2)$ large intermediate values are typically created, with concomitant large time and memory cost.

In view of such expression swell, it is remarkable that iterative methods provide for solution in $n^{3+o(1)}$ time and $n^{2+o(1)}$ space when input entry lengths are constant. (The factor $n^{o(1)}$ absorbs any factors logarithmic in n.) There are two contending approaches.

A classical approach for finding rational number solutions to linear systems is Dixon's modular method[6] which begins by solving the system modulo a prime, p, and proceeds to a p-adic approximation of the solution by Hensel lifting, and finishes with reconstruction of the rational solution from p-adic approximants of sufficient length.

The second approach is a numeric-symbolic combination introduced by Wan in his thesis[27, 26]. Out focus is on extending Wan's method to to a larger class of problems where the size of residuals is too pessimistic.

Historically, the idea of solving exact linear systems to arbitrarily high accuracy by numerical iterative refinement (earlier referred to as binary-cascade iterative-refinement process, BCIR) is attributed to H. Wozniakowski by Wilkinson[28]. Wozniakowski is also acknowledged in a 1981 paper by Kielbasinski[12]. This 1981 paper emphasized "using the lowest sufficient precision in the computation of residual vectors." The required precision was allowed to vary at each iterative step. The case when doubling the working precision suffices to compute sufficiently accurate residuals was also treated soon after in [20]. An important limitation of these early papers was the assumption that the condition number of the matrix A is known. In practice, the system's condition number is rarely known and estimators can fail drastically.

Wan introduced two innovations into iterative refinement. The first innovation is that knowledge of condition numbers is not required. Instead, the accuracy of intermediate approximation vectors is estimated by the computed residuals, $|b - Ax'|$. Wan's second innovation was to compute these residuals exactly, rather than in variable precision or double precision as in the two earlier papers. Accurate residuals are essential, of course, for the correctness of subsequent iterative steps. However, residuals, even exact residuals, do not always correctly assess the accuracy of approximate solutions.

Thus, Wan's approach is basically to iterate with a series of approximate numerical solutions, each contributing its possibly differing number of correct bits, and to do exact computation of the current residual (via truncation and scaling) at each iteration in preparation for the next. The exactly computed residual is used to estimate the number of reliable bits in the current numeric approximation, and to provide the required accuracy of the residual that is to be input into the next iterative step. The final step in Wan's method, just as in Dixon's method, is that rational construction of the solution is undertaken only after a sufficiently accurate (dyadic, in this case) rational approximation is obtained.

One may say that, as input to the rational reconstruction, Dixon's method produces a Laurent series in powers of the prime p and numeric-symbolic iteration produces a Laurent series in powers of $1/2$. A bound is computed for the maximum length of the series necessary to assure rational reconstruction. Both of these methods have the same asymptotic complexity.

Reconstruction earlier in the process can be tried and will succeed if the rational numbers in the solution have smaller representations than the *a priori* bound. This is not done in the current implementation of Wan's method. Steffy studies this case in [21]. We offer a variant of rational reconstruction which recognizes potential early termination. But we distinguish these speculative results from the results that are *guaranteed* from known properties of the output and length of the Laurent series. Speculative results are be checked to see if they satisfy the original linear system.

One advantage of numeric-symbolic iteration is that it obtains the most significant digits first. One can stop short of the full rational reconstruction and instead take as the output the floating point values at any desired precision. A disadvantage of numeric-symbolic iteration is that it does require the numeric solver to obtain at least a few bits of accuracy at each iteration to be able to continue. Thus it is subject to failure due to ill-conditioning.

Wan's method has been implemented in LinBox[22, 7]. Some uses of linear system solving, for instance in Smith Normal Form computation, proceed by trying Wan's method and, if it fails, resorting to Dixon's. Unfortunately, past experience is that the numeric-symbolic iteration fails more often than not in this context. The confirmed continuation algorithm variant reported here is designed to significantly increase the success rate.

In section 2, we discuss Dixon's and Wan's iterations in more detail. Then in section 3 we describe our confirmed continuation method. In section 4 the rational reconstruction phase is discussed in detail. Finally our experiments are reported in section 5.

2. BACKGROUND

Here is a unified skeleton of Dixon's p-adic method and the numeric (dyadic) iteration for rational linear system solution. To unify the notation note that a rational number x may be written as a Laurent series $x = \sum_{i=k}^{\inf} x_i p^i$, where either p is a prime (p-adic expansion) or $p = 1/2$ (dyadic expansion). It will be convenient to think of the dyadic expansion in e bit chunks, in other words, use $p = 2^{-e}$. We specify that each x_i is integer and $0 \le x_i < p$ in the p-adic case, $0 \le x_i < 1/p$ in the dyadic case. In either case let $x \bmod p^l$ denote $\sum_{i=k}^{l-1} x_i p^i$. This will allow us to use the same modular language when discussing p-adic or dyadic expansions.

The skeleton of iterative refinement schemes to compute a solution x to $Ax = b$ is then the following.

1. Compute B such that $B = A^{-1} \bmod p$. That is, $A * B = I \bmod p$. The needed functionality of the object is, that for various vectors r, it can be used to accurately compute $A^{-1}r \bmod p$ in n^2 arithmetic steps. For instance, B could be represented by an LU decomposition of A.

2. By Cramer's rule the solution vector can be represented by quotients of $n \times n$ minors of (A, b). Compute a bound H (for instance the Hadamard bound) for these determinants. Let $k = \lceil \log_q(H) \rceil$, where $q = p$ if doing p-adic expansion and $q = 1/p$ if doing dyadic expansion. $2k$ terms of expansion are needed, in the worst case, to reconstruct the rational numbers in the solution vector.

3. Let $r_0 = b, y_0 = 0$. For i in $0..2k$ do the following:

   (a) $y_i = A^{-1} r_i \bmod p$.

   (b) $r_{i+1} = (r_i - Ay_i)/p$. Do this computation modulo p^2 at least. Since $r_i - Ay_i = 0 \bmod p$, after the division, r_{i+1} is the correct residual modulo p.

   (c) $y = y + y_i p^i$.

   Each of these steps corrects for the preceding residual by computing y to one more p-adic digit of accuracy. In other words $y = A^{-1} b \bmod p^{2k}$.

4. Apply rational reconstruction to y, p^{2k} to obtain x, the vector of rational numbers solution.

When p is a prime, this is Dixon's method. When $p = (1/2)^{30}$ this is essentially Wan's numeric-symbolic iterative scheme. The method succeeds so long as this is possible at each iteration. Thus there is the possibility of failure due to insufficient numeric accuracy in the iteration not present in Dixon's method. On the other hand, it is possible to exploit whatever amount of accuracy is achieved at each step, which could be more or fewer than 30 bits. In other words, there is no need to use the same power of 1/2 at each iteration. Wan's iteration (and others) adjust the number of bits used at each iteration to the accuracy of the solver.

The trick to this is to know the accuracy of the solver. The condition number and/or the norm of the residual (absolute and/or relative) have been used as guidance here. The residual norm idea is basically that in step 3b if $r_{i+1} = r_i - Ar_i$ is smaller by e bits than r_i, then also the first e bits of y_i are likely to be accurate. As is well known, this is not always the case.

The first contribution of our approach is to replace the use of the residual norm with an overlap confirmed continuation principle. Suppose it is believed that e bits are accurate. The residual norm based iteration would define y_i as $w \bmod 2^{-e}$. Thus $w = y_i + 2^{-e}q, (q \leq 1)$ and q is discarded[1]. Instead, we choose the exponent e' of 1/2 used at each iteration slightly conservatively. Let $e' = e - 1$ and use the decomposition $w = y_i + 2^{-e'}q$. We take a little less in y_i so as to be able to make use of q. Since we believe the numeric solver gave us e bits of accuracy, the first bit in each entry of q is presumably accurate. Thus y_{i+1} should agree with q in the leading bits. When this happens we say we have a *confirmed continuation*. When it fails, we recognize that w was not accurate to e bits, and make an adjustment as described in the next section.

Confirmed continuation is a heuristic, since when it succeeds we do not know with certainty that the solution is accurate. It will succeed very well when the numeric solver is unbiased and the intuition is that it will still do very well when there is bias. Let $A \in \mathbb{Z}^{n\times n}$ and let B be a representation of A^{-1}. Suppose B is unbiased, which means that, for any $b \in \mathbb{Z}^n, s \in \mathbb{Z}$, if $y = Bb \bmod 2^s$ and $y \neq A^{-1}b \bmod s$ then the direction of $B(b - Ay)$ is uniformly random. Observe that if B is unbiased then the probability is $1/2^n$ of a false one bit continuation confirmation. This is just the observation that there are 2^n patterns of n bits. This is a rather weak justification for our confirmed continuation heuristic since solvers are rarely if ever unbiased. However, in practice the heuristic is proving to be effective, allowing continuation in some cases in which the residual norm is discouragingly large.

In the next section our confirmed continuation method is described in more detail including the exploitation of techniques to discover the solution sooner when the actual numerators and denominator are smaller than the *a priori* bounds. This is variously called output sensitivity or early termination [5, 18]. Output sensitivity has been used primarily with Dixon's algorithm. The only study of it we know for numeric-symbolic iteration is [21].

[1]It is a central idea of Wan's approach to do this truncation so that the next residual may be computed *exactly*.

3. CONFIRMED CONTINUATION AND OUTPUT SENSITIVITY

Our variant on iterative refinement uses the same basic structure as previous implementations. That is, the system is solved numerically in a loop, with the solution at each iteration contributing some bits to the dyadic numerators and common denominator. Specifically, we call the solution in a step of the iteration $\hat{x}$, and divide it into two parts. $\hat{x}_{int}$ contains the higher order bits and is incorporated into the dyadic estimate. $\hat{x}_{frac}$ contains the lower order bits and is unused in Wan's algorithm. The residual vector obtained by applying A to $\hat{x}_{int}$ provides the right-hand side for the next iteration. The loop ends when it is determined the dyadic approximants contain enough information to reconstruct the true rational solution. This determination is made by checking against a pre-computed bound on the size of the rationals.

Algorithm 1 Overlap: Confirmed continuation iterative refinement to solve $Ax = b$

Input: $A \in \mathbb{Z}^{n\times n}, b \in \mathbb{Z}^n, k$. Output: $x \in \mathbb{Z}^n, 0 < q \in \mathbb{Z}$ such that $Ax = qb$.
Compute A^{-1}. {Numeric LU decomposition}
$N_{1..n} \leftarrow 0$. {dyadic numerators}
$D \leftarrow 1$. {common denominator}
loopbound $\leftarrow 2 \times \prod_{i=1}^{n} \|A_i\| \times b_{\max}$.
$r \leftarrow b$. {Residue of intermediate solutions}
$s \leftarrow 52 - \mathbf{bitlength}(n \times \|A\|_\infty \times \|b\|_\infty)$.
thresh $\leftarrow \frac{1}{2^k}$. {Threshold for overlap confirmation}
$\hat{x} \leftarrow A^{-1}r$.
while $D <$ loopbound **do**
  $\hat{x}_{int} \leftarrow \lfloor \hat{x} \times 2^s + 0.5 \rfloor$.
  $\hat{x}_{frac} \leftarrow \hat{x} - \hat{x}_{int}$.
  $r \leftarrow r \times 2^s - A\hat{x}_{int}$. {Update residual}
  $\hat{x} \leftarrow A^{-1}r$.
  if $||\hat{x} - \hat{x}_{frac}||_\infty >$ thresh **then**
    Shrink s, repeat iteration.
  else
    $N_{1..n} \leftarrow N_{1..n} \times 2^s + \hat{x}_{int}$. {Update dyadics}
    $D \leftarrow D \times 2^s$.
    if $r = 0$ **then**
      Return: N, D as x, d.
    end if
  end if
end while
Return: $x, d \leftarrow$ Algorithm 3 (N, D).

To ensure the accuracy of the numeric solver's solution at each iteration, we verify there is overlap between the current iteration's numeric solution and the discarded fractional portion of the previous solution. Overlap is demonstrated in the conditional statement where prospective solution $\hat{x}$ is checked against $\hat{x}_{frac}$, the leftover bits from the previous iteration. The vectors are subtracted and the maximal absolute value in the difference set is checked against a threshold $\frac{1}{2^k}$ to ensure k overlapping bits. In practice, we find one bit of overlap (i.e. $k = 1$) suffices to confirm continuation except for very small n.

Once this verification step is successful, we are able to explore seeking more bits of accuracy from the numeric solver. We treat the value s as an adjustable bit-shift length. Each numeric solution $\hat{x}$ is multiplied by 2^s in order to split into

$\hat{x}_{int}$ and $\hat{x}_{frac}$. That is, it is bit-shifted left by s. Likewise when we update the dyadic numerators N, we shift them left by s, then add the new information to the now zeroed lower s bits.

The value of s is at our disposal and allows the algorithm to adapt to changing accuracy from the numeric solver. Ideally it will hug the true number of accurate bits in the intermediate results as closely as possible. As long as some bits of accuracy are provided in each solve, the iteration can continue. Within the bounds of a 52-bit mantissa of a double-precision floating point number, we seek to maximize the shift length to minimize the number of iteration steps. Program speed is the foremost improvement provided by the confirmed continuation method as compared to the residual-norm based iterative refinement.

Finding a good shift length s is a matter of starting at 1 and iteratively doubling until no overlap is evident or the hard ceiling of 52 is reached. The absence of overlap is an indication that we obtained fewer than s bits of numeric accuracy, and we must back off. Required to do this is a copy of the last successful $\hat{x}$. From this we must repeat the extraction of bits using a smaller s, recompute residual, r, and finally solve against this adjusted right-hand side. We use a binary search to determine the maximum value of s that produces overlap, which sits between the failed shift length and the last successful shift length. Algorithm 1 omits these details for brevity, simply initializing s to a sensible starting point.

If the step applying A to $\hat{x}_{int}$ is done in double precision and produces values that cannot fit into the mantissa of a double floating point number, this operation computes an inexact residual. The next iteration would then be solving the wrong problem. This error is detected by neither the norm-based approaches nor the overlap method, since both approaches only guard against numerical inaccuracy of the partial solutions themselves. If the numeric solver itself is accurate, repeated divergence from the problem we intend to solve will be undetected. The algorithm completes after sufficiently many iterations, and reports dyadic estimates that have no hope of being reconstructed into the correct rational solution to the original problem.

To avoid this error, we employ big-integer arithmetic (using GMP) in the residual update, but only when necessary, specifically when $\|A\|_\infty \times \|\hat{x}_{int}\|_\infty \geq 2^{52}$, which is a conservative condition.

The matrix norm is computed beforehand, so it costs only $O(n)$ work per iteration to compute the vector norm. The flexibility of this approach both prevents the aforementioned divergent behavior and allows for the use of quicker, double precision computation of the exact residual in many cases. Our experience is that for borderline problems that require some bignum residual computation, the need is rare amongst iterations.

Sometimes the numerators and denominator of the final rational solution are significantly smaller than the worst case bound computed *a priori*. When this is the case, it is possible to obtain dyadic approximants of sufficient length to reconstruct the solution before the iterative refinement would normally end. Our early termination strategy is designed to improve running time for these cases. It is sketched in Algorithm 2.

The core iterative refinement loop is still in place, but every so often it is stopped to attempt a rational reconstruction

Algorithm 2 Ov-ET: Confirmed continuation iterative refinement w/ Early Termination to solve $Ax = b$

This is Algorithm 1, replacing the while loop (iterative refinement) with:
bound $\leftarrow \prod_{i=1}^{n} \|A_i\|_2$. {Hadamard bound}
while bound $<$ loopbound **do**
  while $D <$ bound **do**
    while loop in Algorithm 1.
  end while
  bound $\leftarrow \sqrt{bound \times loopbound}$.
  $i \leftarrow$ **random**$(1..n)$. {Select random element}
  if Algorithm 3 (N_i, D) is **success then**
    if $x, d \leftarrow$ Algorithm 3 (N, D) is **success then**
      Return: x, d.
    end if
  end if
end while
Return: $x, d \leftarrow$ Algorithm 3 (N, D).

from the current dyadic approximation. Specifically it is initially stopped at the halfway point to the worst case bound, that is, as soon as D is larger than the Hadamard bound for $\det(A)$, which is the initial value of *bound* in Algorithm 2. A single-element rational reconstruction is attempted using a random element from the numerator vector N and denominator D. Success here provides encouragement for attempting a full vector reconstruction with all elements of N, which is then performed. Success on the vector reconstruction provides a speculative or guaranteed solution, depending on the reconstructed denominator and length of the dyadic approximation. After a solution verification, we terminate here, potentially saving many iterations.

Upon failure to rationally reconstruct the solution on an early attempt the *bound* is set to the bitwise half-way point between itself and *loopbound*, the point at which iterative refinement would end without early termination. The new value of *bound* serves as the next checkpoint for an early termination attempt. This is a binary search that keeps reporting "higher" after failed guesses. The strategy ensures the number of attempts is logarithmic in the number of iterations required. Also reconstruction attempts are of increasing density as the full iteration bound is approached, which address the expectation that successful early termination becomes increasingly likely. We remark that van Hoeij and Monagan [25] and Steffy [21] also use a logarithmic number of iterations but with increasing density of trials at the low numbered iterations rather than at the end as we do. Either approach ensures good asymptotic behaviour. Which is better in practice is an open question. For good performance in practice, One might use a more uniform spacing of reconstruction trials with frequency such that reconstruction cost does not exceed a specified fraction of overall cost.

4. DYADIC RATIONAL TO RATIONAL RECONSTRUCTION

In an earlier section we made a point of the similarity between numeric approximation and p-adic approximation. When it comes to the rational reconstruction, both may be expressed in terms of extended Euclidean algorithm remainder sequences. However there is a difference. In rational reconstruction from a residue and modulus, the a remain-

der serves as numerator and the coefficient of the residue as denominator of the approximated rational. The coefficient of the modulus is ignored. In contrast, for dyadic to rational reconstruction we use the two coefficients for the rational and the remainder serves to measure the error of approximation as we explain next.

First consider a single entry of the solution vector. The input to the reconstruction problem is a dyadic n/d (with d a power of 2) together with a known bound B for the denominator of the approximated rational a/b. Let us say that a/b is *well approximated* by n/d if $|a/b - n/d| < 1/2d$. By this definition, n/d can never well approximate the midpoint between $(n \pm 1)/d$ and n/d. But this midpoint has larger denominator, and the rational reconstruction process described below never finds a/b when $b > d$ in any case. In the system solving application, the rational reconstruction would fail but the next iteration would compute a/b exactly and terminate with residual 0.

Proposition 1. *If two distinct fractions a/b and p/q are well approximated by n/d then $d < bq$.*

The proposition follows from the fact that $1 \leq |pb - aq|$ (nonzero integer) and the triangle inequality: $1/qb \leq |p/q - a/b| \leq |p/q - n/d| + |n/d - a/b| < 1/2d + 1/2d = 1/d$,

Proposition 2. *If a/b is well approximated by n/d and $d \geq bB$, then no other fraction with denominator bounded by B is well approximated. Also n/d well approximates at most one rational with denominator bound B when $d \geq B^2$.*

Proposition 4 follows from the previous proposition since $bq \leq bB$, when p/q is a second well approximated fraction with denominator bounded by B.

This allows for a *guaranteed* early termination (output sensitive) strategy in the numeric-symbolic iteration. In the Dixon method, early termination is a probabilistic matter (the prime used is chosen at random). It cannot be so in numeric-symbolic iteration, because there is no randomness used.

Reconstruction of the sought fraction a/b is done with the extended Euclidean algorithm remainder sequence of n, d. Define this to be (r_i, q_i, p_i) such that $r_i = q_i n - p_i d$, with $q_0 = p_1 = 1$ and $q_i = p_0 = 0$. We have altered the usual treatment slightly so that p_i and q_i are positive (and strictly increasing) for $i > 1$, while the remainders alternate in sign and decrease in absolute value. Let Q be defined by Euclidean division on the remainders: $|r_{i-1}| = Q|r_i| + r$, with $0 \leq r < |r_i|$. Then the recursion is $r_{i+1} = Qr_i + r_{i-1}$, $p_{i+1} = Qp_i + p_{i-1}$, and $q_{i+1} = Qq_i + q_{i-1}$. Also the determinants $p_i q_{i+1} - p_{i+1} q_i$ are alternately 1 and -1. See e.g. [9] for properties of remainder sequences and continued fractions.

Proposition 3. *The coefficients p, q in a term (r, q, p) of the remainder sequence define a rational p/q well approximated by n/d and denominator bounded by B if and only if $2|r| < q \leq B$.*

This follows from $r = qn - pd$ so that $|r|/qd = |p/q - n/d| < 1/2d$ (and $q \leq B$ by hypothesis).

Proposition 4. *Given n, d, B, let (r, q, p) be the last term such that $q < B$ in the remainder sequence of n, d. This term defines the best approximated B bounded fraction p/q of any term in the remainder sequence.*

When n/d well approximates a rational a/b and $d < bB$ then $a/b = p/q$, i.e. is defined by this term of the remainder sequence.

This follows because $|r|$ is decreasing and q increasing in the remainder sequence. The claim that the rational will be found in the remainder sequence follows from Theorem 4.4 in [26]. Half extended gcd computation computation ((r, q) rather than (r, q, p)) lowers the cost, with p computed post hoc only for the term of interest.

When this last term below the bound defines a well approximated rational p/q, i.e. $2|r| < q$, we say we have a "guaranteed" reconstruction. When that is not the case, it is still possible that we have found the correct rational. As mentioned in the previous section, sometimes by good luck this leads to successful solutions even when the iteration has not proceeded far enough to have a guaranteed well approximated answer.

Thus we may offer the last approximant from the remainder sequence with denominator bounded by B. It is speculative if $d > bB$ and guaranteed to be the unique solution otherwise. It is never necessary to go beyond $d = B^2$. As the experiments attest, trial reconstructions during the numeric iteration process, can be effective at achieving early termination. The vector reconstruction described next helps keep the cost of these trials low.

To construct a solution in the form of a vector of numerators $x \in \mathbb{Z}^n$ and common denominator q from a vector of $n \in \mathbb{Z}^n$, and common (power of 2) denominator d, we can often avoid reconstructing each entry separately with a remainder sequence computation. We compute x_i as $x_i = [n_i q/d]$. In other words, x_i is the quotient in the division $n_i q = x_i d + r$, with $-d/2 < r < d/2$. The error of the approximation is then $x_i/q - n_i/d| = r/qd$. If this error is bounded by $1/2d$, x_i/q is well approximated by n/d. Thus we have a well approximated result if and only if $2r < q$. When single division fails to produce a well approximated x_i/q, resort to a full remainder sequence. This leads to the following algorithm.

The first loop discovers new factors of the common denominator as it goes along. In practice one or two full reconstructions are needed and the remainder are done by the single division before the if statement. The backward propagation of new factors is delayed to the second loop, to avoid a quadratic number of multiplications. In the worst case this algorithm amounts to n gcd computations. In the best case it is one gcd and $n-1$ checked divisions with remainder. Experimentally we have encountered essentially the best case, with a very few full gcd computations.

A rational reconstruction algorithm presented in another paper of this proceedings achieves a better asymptotic complexity than that of n independent scalar rational reconstructions [2]. That concerns reconstruction from residues and modulus and may be adaptable to the dyadic to rational setting.

To our knowledge, prior algorithms do not assume n/d well approximates (to accuracy $1/2d$) and so do not exploit the guarantee of uniqueness as we do, particularly when using the early termination strategy. However, Cabay [4] gave a guarantee of early termination based on a sufficiently long

Algorithm 3 Vector DyadicToRational

Input: $N \in \mathbb{Z}^n, D \in \mathbb{Z}$. Output: $x \in \mathbb{Z}^n, 0 < d \in \mathbb{Z}$, flag, such that flag is "fail" or N/D well approximates x/d and flag is "speculative" or "guaranteed".
$d \leftarrow 1$.
for i from 1 to n **do**
  $x_i \leftarrow [N_i q/d]$.
  if x_i fails the well approximation test **then**
    x_i, d_i, flag = ScalarDyadicToRational(N_i, D).
    if flag = "fail", return "fail".
    Compute the factorizations $d = a_i g, d_i = b_i g$, where $g = \gcd(d, d_i)$. The new common denominator is $d \leftarrow a_i d_i$, so set $x_i \leftarrow x_i \times a_i$. Prior numerators must be multiplied by b_i. Enqueue b_i for that later.
  end if
end for
$B \leftarrow 1$.
for i from n down to 1 **do**
  $x_i \leftarrow x_i \times B$;
  if $b_i \neq 1$ **then**
    $B \leftarrow B \times b_i$.
  end if
end for
return x, d, flag. [If any scalar reconstruction was speculative, flag = "speculative", otherwise flag = "guaranteed".]

sequence of iterations resulting in the same reconstructed rational. This was in the context of Chinese remaindering, but should apply to Hensel lifting and numeric-symbolic iteration as well. Note that our guarantee comes from a single reconstruction, not a series. Steffy exploits speculative reconstructions as we do and gives a guaranteeing condition based on the error of the approximation [21, lemma 2.6]. Note that our overlap heuristic provides evidence on the error that is independent of a condition number estimation. It could be useful to accept reconstructions earlier in the iteration using Steffy's condition and our measure of the error. We have not yet experimented with this.

5. EXPERIMENTS

For test matrices, we use the following 8 matrix families $H_n, J_n, Q_n, S_n, m_n, M_n, R_n, Z_n$ described next.

H_n**:** The inverse of the $n \times n$ Hilbert matrix. This is a famously ill-conditioned matrix. The condition number of H_n

$$\kappa(H_n) = \|H_n\|_2 \left\|H_n^{-1}\right\|_2 \approx c\ 33.97^n/\sqrt{n}$$

where c is a constant, is quoted in[1]. We find that our numeric solvers – both the residual norm based and the overlap confirmed continuation approach – can handle this matrix only up to $n = 11$. On the other hand, Dixon's p-adic iteration can handle any size, provided p is chosen large enough to insure the nonsingularity of H_n mod p. For instance, Dixon does the $n = 100$ case in 12 seconds. This class is introduced only to illustrate the potential for numeric solver failure due to ill-condition.

J_n**:** This matrix is twice the $n \times n$ Jordan block for the eigenvalue 1/2. We multiply by 2 to get an integer matrix. It is a square matrix with 1's on the diagonal, and 2's on the first subdiagonal. Numerical computations for matrices with repeated eigenvalues are notoriously difficult. The inverse matrix contains $(-2)^j$ on the j-th subdiagonal. For $n > 1023$, the matrix J_n^{-1} is not representable in double precision (infinity entries), and for smaller n it poses numerical challenges.

Examples of numeric failure to converge			
matrix	Dixon	Wan	Overlap
J_{994}	2.27	1.77	**0.0850**
J_{995}	2.23	fail	**0.0920**
J_{1022}	2.40	fail	**0.100**
J_{1023}	**2.38**	fail	fail
J_{2000}	**13.3**	fail	fail
Q_{500}	2.07	fail	**0.81**
Q_{1000}	15.0	fail	**7.29**
Q_{2000}	121	fail	**70.3**
Q_{4000}	1460	fail	**633**

Table 1: Dixon is p-adic iteration, Wan is numeric-symbolic iteration using residual norm based continuation, Overlap is the confirmed continuation of this paper. Times are in seconds.

Table 1 shows that the numeric-symbolic solvers are faster than the p-adic lifting when they work, but they have difficulties with J_n.

For reasons we do not completely understand, on another machine the thresholds at which the solvers have convergence problems are different than in Table 1. We suspect it has to do with a different LAPACK version. On that machine, for n larger than 54, Wan's residual norm based solver is discouraged by the first residual and gives up.

For n larger than 1023, infinities (numbers not representable in double precision) defeat all numeric solving. The bottom left entry of J_n^{-1} is 2^{n-1} which is not representable when $n \geq 1024$. However, the overlap solver fails at $n = 1023$. Although the inverse matrix itself is just barely representable, some numbers which occur in the matrix-vector products are not representable in this case.

Q_n**:** Let $Q_n = DLD$, where L is the $n \times n$ Lehmer matrix[23, 14], with elements $L_{i,j} = \min(i,j)/\max(i,j)$, and D is the diagonal matrix with $D_{i,i} = i$. Thus Q is an integral Brownian matrix[11] with $Q_{i,j} = \min(i,j)^2$ ("Q" is for quadratic). A closed form expression, $\det(Q_n) = 2^{-n}(2n)!/n!$ follows from[13].

Note that $Lx = b$ iff $x = Dy$ and $Qy = Db$ (also $De_1 = e_1$). Being an integer matrix, Q fits in our experimental framework while rational L does not.

Table 1 includes Q_n measurements concerning numeric difficulties. In his recent experiments, Steffy[21] used the Lehmer matrix as an example where Dixon's method works but numeric-symbolic iteration does not. We include it here because it shows a striking difference between residual norm based continuation and overlap confirmed continuation in numeric-symbolic iteration. In fact, Wan's code in LinBox fails on Q_n for $n > 26$.

The examples of Q_n, along with the remaining five classes of examples, are used for our performance study shown in Table 2. These test cases are in many collections of matrices used for testing. A notable such collection[3] is maintained by John Burkardt.

Three of our test matrices (Q_n, m_n, and $\mathrm{M_n}$) are Brownian matrices[11] in that they have have a "echelon" structure: that is, the elements obey $b_{i,j+1} = b_{i,j}, j > i$, and $\mathrm{b_{i+1,j}} =$

$b_{i,j}, i > j$, for all i, j. Many Brownian matrices have known closed form results for inverses, determinants, factorization, etc. One source of special results is the following observation[11]. If the matrix P is taken to be a Jordan block corresponding to a repeated eigenvalue of -1, then PBP^T is tridiagonal if and only if B is Brownian.

S_n: The $n \times n$ Hadamard matrix using Sylvester's definition. $S_1 = (1), S_{2n} = \begin{pmatrix} S_n & S_n \\ S_n & -S_n \end{pmatrix}$. This definition results in n being a power of two. The determinant of S_n equals $n^{n/2}$, which is sharp for the Hadamard bound. Thus, for any integral right hand side, the solution is a dyadic rational vector. This provides a test of early termination due to a zero residual.

Algorithm performance comparisons				
Matrix	Dixon	Wan	Overlap	Ov-ET
S_{512}	0.728	0.711	0.0723	**0.0721**
m_{500}	1.28	1.34	**0.273**	**0.273**
M_{500}	1.46	fail	1.06	**0.562**
Q_{500}	2.41	fail	2.98	**1.39**
R_{500}	1.09	1.04	1.05	**0.931**
Z_{500}	0.793	0.864	0.584	**0.580**
S_{1024}	4.58	4.75	0.380	**0.371**
m_{1000}	8.83	10.8	**2.24**	**2.24**
M_{1000}	10.2	fail	8.56	**4.42**
Q_{1000}	**16.6**	fail	24.5	17.4
R_{1000}	7.25	fail	6.87	**6.48**
Z_{1000}	6.04	6.38	**4.41**	4.46
S_{2048}	32.3	36.6	**2.08**	2.10
m_{2000}	72.0	89.6	**17.1**	**17.0**
M_{2000}	82.8	fail	75.0	**37.8**
Q_{2000}	**137**	fail	243	167
R_{2000}	54.6	fail	53.7	**49.3**
Z_{2000}	45.4	52.15	35.8	**34.1**
S_{4096}	255	297	**11.7**	**11.7**
m_{4000}	579	783	**138**	**138**
M_{4000}	628	fail	658	**319**
Q_{4000}	**1519**	fail	3274	2294
R_{4000}	**380**	fail	393	397
Z_{4000}	340	439	318	**271**
S_{8192}	2240	2517	**77.6**	82.6
m_{8000}	mem	6802	**1133**	1138
M_{8000}	mem	fail	6170.6	**3049**
Q_{8000}	mem	fail	33684	**27367**
R_{8000}	mem	fail	**2625**	2710
Z_{8000}	mem	5771	2584	**2474**

Table 2: Dixon, Wan, and Overlap columns are as in Table 1. Ov-ET is Overlap with with early termination enabled. "mem" denotes out of memory. Times are in seconds.

m_n: The $n \times n$ matrix with $m_{i,j} = \min(i, j)$. Because this a Brownian matrix[11], PBP^T is tridiagonal, and in this case, the tridiagonal matrix is the identity matrix. Thus, determinant of m_n is 1, so the solution is integral for any integral right hand side. This is another case where early termination due to zero residual is expected. But the entries in the inverse are larger than in the Hadamard matrix case, so more iterations may be needed for this example.

M_n: The $n \times n$ matrix with $M_{i,j} = \max(i, j)$. The determinant of this matrix is $(-1)^{n+1}n$, so the solution vector has denominator much smaller than the Hadamard bound predicts. The determinant of M_n is found by reversing its rows and columns, which does not change its determinant. The result is a Brownian matrix. Its tridiagonal form using the matrix P is the identity matrix — except for the value $(-1)^n n$ in the upper left corner.

The previous three test cases can benefit from early termination. Q_n and the following 2 are expected to benefit less from output sensitivity.

R_n: An $n \times n$ matrix with random entries in $(-100, 100)$.

Z_n: An $n \times n$ $\{0, 1\}$-matrix with probability 1/2 of a 1 in a given position. This is meant to represent some commonly occurring applications. The LinBox library is often used to compute Smith forms of incidence matrices, where invariant factor computation involves solving linear systems with random right hand sides.

Reported are run times on a 3.0 GHz Intel Pentium D processor in a Linux 2.6.32 environment. All codes used are in LinBox, svn revision 3639, and will be included in the next LinBox release. The experiments were run with right hand sides being e_1, the first column of the identity matrix. This provides a direct comparison to Steffy's examples [21] and in some cases allows checking a known solution formula. But for Q_n, M_n, and $\mathbb{Z}_n$, the right hand sides used are random with entries in $(-100, 100)$. This is to create an interesting early termination situation in the case of Q_n and M_n (where the solution for rhs e_1 is obtained in the first iteration). For Z_n it is in view of the expected application.

S_{2^k} and m_n are examples where the determinant is a power of two and considerably less than the Hadamard bound. Thus an early termination due to a perfect dyadic expansion with residual zero can occur and no rational reconstruction is needed. Both forms of the Overlap algorithm verify this. The current implementation of Wan's method does not check for an exactly zero residual, though there no reason it could not. The determinant (thus the denominator of solution vector) of S_n is $n = 2^k$ and the Hadamard bound is considerably larger, $n^{n/2}$. Output sensitive termination due to zero residual accounts for the factor of 10 or more speedups. The determinant of m_n is 1 and the Hadamard bound is larger than that of S_n. For right hand side e_1 the solution vector is $2e_2 - e_1$ so that essentially no iteration is required if early termination (Dixon) or zero residual detection (Overlap) is used. In these cases all the time is in matrix factorization which is about 4 times more costly modulo a prime (Dixon) than numerically using LAPACK (Overlap). The Wan's implementation lacks the early termination so does a full iteration. That more than offsets the faster matrix factorization than in Dixon making Wan's the slowest on the m_n family.

M_n and Q_n results show early termination saving a factor of about 2, the most available with sufficient dyadic approximation for a guaranteed rational reconstruction. Further speedup is possible from very early speculative reconstructions. We have not explored this.

In the data for the random entry matrix, R_n, and random $\{0, 1\}$-matrix, Z_n, we see variable speedups up to a factor of 1.8 over Dixon's p-adic lifting, sometimes aided a bit by early termination. Significant early termination is not generally expected for these matrix families.

The overlap method should work well with sparse numeric solvers (both direct and iterative) for sparse matrices as well. In that case performance asymptotically better than obtained with Dixon's method can be expected. The significant symbolic competition will be the method of Eberly, et al[8]. We intend to explore the sparse matrix problem in the future. To this end we have made a LinBox interface to call MATLAB functions.

6. REFERENCES

[1] Bernhard Beckermann. The condition number of real Vandermonde, Krylov and positive definite Hankel matrices. *Numerische Mathematik*, 85:553–577, 1997.

[2] C. Bright and A. Storjohan. Vector rational number reconstruction. In *ISSAC '11*. ACM, 2011.

[3] John Burkardt. TEST MAT Test Matrices. http://people.sc.fsu.edu/~jburkardt/c_src/test_mat/test_mat.html.

[4] Stanley Cabay. Exact solution of linear equations. In *Proceedings of the second ACM symposium on Symbolic and algebraic manipulation*, SYMSAC '71, pages 392–398, New York, NY, USA, 1971. ACM.

[5] Z. Chen and A. Storjohann. A BLAS based C library for exact linear algebra on integer matrices. In *Proc. of ISSAC'05*, pages 92–99. ACM Press, 2005.

[6] J. D. Dixon. Exact solution of linear equations using p-adic expansion. *Numer. Math.*, pages 137–141, 1982.

[7] J-G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B. D. Saunders, W. Turner, and G. Villard. Linbox: A generic library for exact linear algebra. In *ICMS'02*, pages 40–50, 2002.

[8] W. Eberly, M. Giesbrecht, P. Giorgi, A. Storjohann, and G. Villard. Solving sparse rational linear systems. In *Proc. of ISSAC'06*, pages 63–70. ACM Press, 2006.

[9] Joachim Von Zur Gathen and Jurgen Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[10] Keith O. Geddes and Wei Wei Zheng. Exploiting fast hardware floating point in high precision computation. In J. Rafael Sendra, editor, *ISSAC*, pages 111–118. ACM, 2003.

[11] M. J. C. Gover and S. Barnett. Brownian matrices: properties and extensions. *International Journal of Systems Science*, 17(2):381–386, 1986.

[12] Andrzej Kiełbasiński. Iterative refinement for linear systems in variable-precision arithmetic. *BIT*, 21(1):97–103, 1981.

[13] E. Kilic and P. Stanica. The Lehmer matrix and its recursive analogue. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 74(2):195–205, 2010.

[14] D. H. Lehmer. Solutions to problem E710, proposed by D. H. Lehmer: The inverse of a matrix, November 1946.

[15] Robert T. Moenck and John H. Carter. Approximate algorithms to derive exact solutions to systems of linear equations. In *Proceedings of the International Symposium on on Symbolic and Algebraic Computation*, pages 65–73, London, UK, 1979. Springer-Verlag.

[16] Michael B. Monagan. Maximal quotient rational reconstruction: an almost optimal algorithm for rational reconstruction. In Jaime Gutierrez, editor, *ISSAC*, pages 243–249. ACM, 2004.

[17] Teo Mora, editor. *Symbolic and Algebraic Computation, International Symposium ISSAC 2002, Lille, France, July 7-10, 2002, Proceedings*. ACM, 2002.

[18] T. Mulders and A. Storjohann. Certified dense linear system solving. *Jounal of symbolic computation*, 37(4), 2004.

[19] Victor Y. Pan and Xinmao Wang. Acceleration of euclidean algorithm and extensions. In Mora [17], pages 207–213.

[20] Alicja Smoktunowicz and Jolanta Sokolnicka. Binary cascades iterative refinement in doubled-mantissa arithmetics. *BIT*, 24(1):123–127, 1984.

[21] Daniel Steffy. Exact solutions to linear systems of equations using output sensitive lifting. *ACM Communications in Computer Algebra*, 44(4):160–182, 2010.

[22] The LinBox Team. LinBox, a C++ library for exact linear algebra. http://www.linalg.org/.

[23] John Todd. *Basic Numerical Mathematics, Vol. 2: Numerical Algebra*. Birkhäuser, Basel, and Academic Press, New York, 1977.

[24] Silvio Ursic and Cyro Patarra. Exact solution of systems of linear equations with iterative methods. *SIAM Journal on Algebraic and Discrete Methods*, 4(1):111–115, 1983.

[25] Mark van Hoeij and Michael B. Monagan. A modular gcd algorithm over number fields presented with multiple extensions. In Mora [17], pages 109–116.

[26] Zhengdong Wan. *Computing the Smith Forms of Integer Matrices and Solving Related Problems*. PhD thesis, University of Delaware, Newark, DE, 2005.

[27] Zhengdong Wan. An algorithm to solve integer linear systems exactly using numerical methods. *Journal of Symbolic Computation*, 41:621–632, 2006.

[28] James H. Wilkinson. *Rounding Errors in Algebraic Processes*. Dover Publications, Incorporated, 1994.