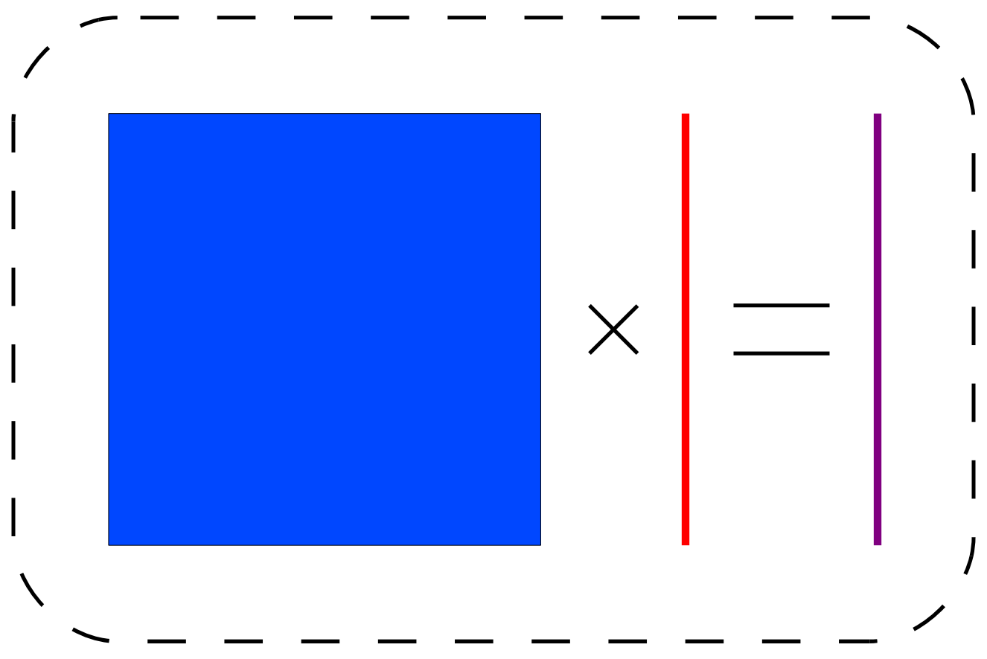
Numeric/symbolic exact rational linear system solver

Bryan Youse, David Wood, B. David Saunders
University of Delaware, Newark, Delaware

Thanks to NSF Grant CCF-0830130

Solving a rational linear system: $Ax = b$



Symbolic

Software: LinBox, Maple, ...

Modular Methods

- Dixon's method: *p*-adic lifting
- Chinese Remaindering: computation modulo many prime numbers

- **Deliver the exact answer without error**
- **Not subject to machine limitations**
- Computationally expensive

Numeric

Software: LAPACK, MATLAB, many more

Direct Methods

- Gaussian Elimination
- QR Factorization

Iterative Methods

- Jacobi's method
- Lanczos' method
- GMRES method

- **Well studied; highly tuned**
- Solution accuracy limited by machine word size to floating point precision

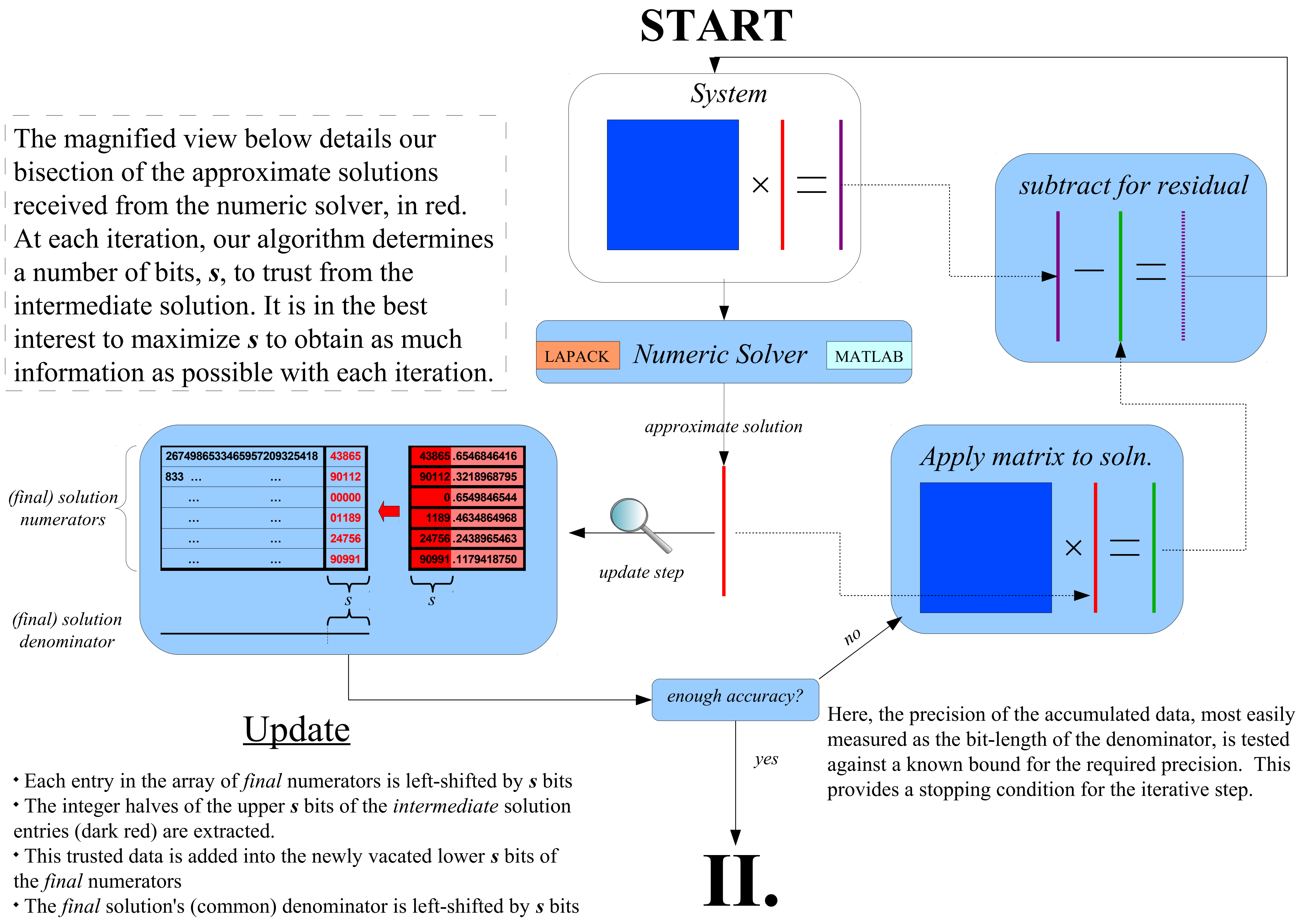
Numeric/symbolic Hybrid

The best of both worlds! Our approach, an improvement on earlier work by Wan [1], can solve exactly well-conditioned integer linear systems using numeric solving methods. This is made possible by the following **fact:**

> If two rational numbers $r_1 = \frac{a}{b}$, $r_2 = \frac{c}{d}$ are given with $\gcd(a, b) = 1$, $\gcd(c, d) = 1$, and $r_1 \neq r_2$, then $|r_1 - r_2| \geq \frac{1}{bd}$

i.e. rational numbers with bound denominators are **discrete**. If we can obtain a solution with very high accuracy, we can reconstruct the rational solution

I. *Iterative Refinement*



The magnified view below details our bisection of the approximate solutions received from the numeric solver, in red. At each iteration, our algorithm determines a number of bits, ***s***, to trust from the intermediate solution. It is in the best interest to maximize ***s*** to obtain as much information as possible with each iteration.

Update

- Each entry in the array of *final* numerators is left-shifted by ***s*** bits
- The integer halves of the upper ***s*** bits of the *intermediate* solution entries (dark red) are extracted.
- This trusted data is added into the newly vacated lower ***s*** bits of the *final* numerators
- The *final* solution's (common) denominator is left-shifted by ***s*** bits

Here, the precision of the accumulated data, most easily measured as the bit-length of the denominator, is tested against a known bound for the required precision. This provides a stopping condition for the iterative step.

Our main improvement over Wan's approach is what we call **"confirmed continuation".** The data highlighted in light red (above) represents the previously unused fractional portion of the partial solution. If the high order bits from this data *overlap* with the high order bits from our next partial solution, we have confirmed continuation. We are now sure the data accrued from the numeric solver is accurate. This Overlap method is more robust and faster to check at each iteration than Wan's residual norm approach [1].

II. *Rational Reconstruction*

The Iterative Refinement step produces a vector of numerators and their common denominator. These can be viewed as real numbers. The size of the denominator of the exact rational solution is less than the product of the Hadamard bound of the input matrix and the norm of the input right-hand side vector. Because we know this bound, we can reconstruct the solution.

The provided real numbers can be viewed as a continued fraction (right). The rational solution we seek is some truncation of this fraction, which is unique, owing to the central fact presented above. We use a modified form of Euclid's gcd algorithm to determine the location for this truncation

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\cdots}}}}$$

$$a_{0,}\, a_{1,} \ldots \in \mathbb{Z}$$

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**

Outlook

A running time (s) comparison of solving multiple sizes of diverse classes of dense systems with asymptotically equivalent algorithms: Dixon's *p*-adic lifting method [2], Wan's method, and our Overlap method with and without early termination. For details on these matrix classes and experimental environment see [3]. Clearly, the efficiency that employing a numeric solver provides is worthwhile. Our *confirmed continuation* method is shown to be both more robust and faster than Wan's initial take on numeric/symbolic iterative refinement. Also demonstrated are the memory limitations of traditional symbolic approaches. **Current work**: using this method with specialized sparse numerical solvers (both direct and iterative) in place of generic dense solvers.

[1] Zhendong Wan. An algorithm to solve integer linear systems exactly using numerical methods. Journal of Symbolic Computation, 2006.
[2] G.I. Malaschonok. Solution of Systems of Linear Equations by the *p*-adic Method. Programming and Computer Software, Vol. 29, 2003.
[3] B. David Saunders, David Harlan Wood, and Bryan Youse. Numeric-Symbolic Exact Rational Linear System Solver. To appear in Proc. of ISSAC 2011.