Numeric-Symbolic Exact Rational Linear System Solver

B. David Saunders, David Wood, and Bryan Youse

University of Delaware

May 18, 2011

Motivation

$Ax = b$

Given $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$, compute $x \in \mathbb{Q}^n$.

Core problem specifics:

- Square $m = n$ matrices
- Matrix entries of length d bits or fewer
- Dense matrices

Background

Symbolic Approaches

- Dixon
  - Solve system modulo p
  - Use Hensel lifting to obtain a p-adic expansion of solution
  - Rational reconstruction from p-adic approximants

Background

Symbolic Approaches

- Dixon
  - Solve system modulo p
  - Use Hensel lifting to obtain a p-adic expansion of solution
  - Rational reconstruction from p-adic approximants
- Wan
  - Numeric iterative refinement to obtain dyadic number solution of high accuracy
  - Specifically, $2h$, the Hadamard bound of the input system.
  - Rational reconstruction from dyadic approximants

Fact

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

That is, though dense in the real number line, rational numbers with bound denominators are discrete.

Fact

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

That is, though dense in the real number line, rational numbers with bound denominators are discrete.

A given real number r can be represented by a continued fraction: $r = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}$, where $a_i \in \mathbb{Z}$.

Confirmed Continuation (Overlap)

At each iteration, we scale our partial solution $\hat{x}$, then split into two parts:

Confirmed Continuation (Overlap)

At each iteration, we scale our partial solution $\hat{x}$, then split into two parts:

- $\hat{x}_{int} = \lfloor \hat{x} \times 2^s \rfloor$

Confirmed Continuation (Overlap)

At each iteration, we scale our partial solution $\hat{x}$, then split into two parts:

- $\hat{x}_{int} = \lfloor \hat{x} \times 2^s \rfloor$
- $\hat{x}_{frac} = \hat{x} \times 2^s - \hat{x}_{int}$

Confirmed Continuation (Overlap)

At each iteration, we scale our partial solution $\hat{x}$, then split into two parts:

- $\hat{x}_{int} = \lfloor \hat{x} \times 2^s \rfloor$
- $\hat{x}_{frac} = \hat{x} \times 2^s - \hat{x}_{int}$

Before updating dyadic approximants, we *confirm continuation* of the iteration by verifying **overlap** between the current $\hat{x}'$ and the previous $\hat{x}_{frac}$.

- e.g. $\max |\hat{x}' - \hat{x}_{frac}| \leq \frac{1}{2^b}$ ensures b bits of overlap

Confirmed Continuation (Overlap)

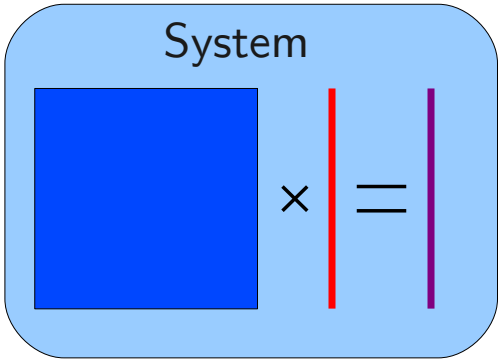
At each iteration, we scale our partial solution $\hat{x}$, then split into two parts:

- $\hat{x}_{int} = \lfloor \hat{x} \times 2^s \rfloor$
- $\hat{x}_{frac} = \hat{x} \times 2^s - \hat{x}_{int}$

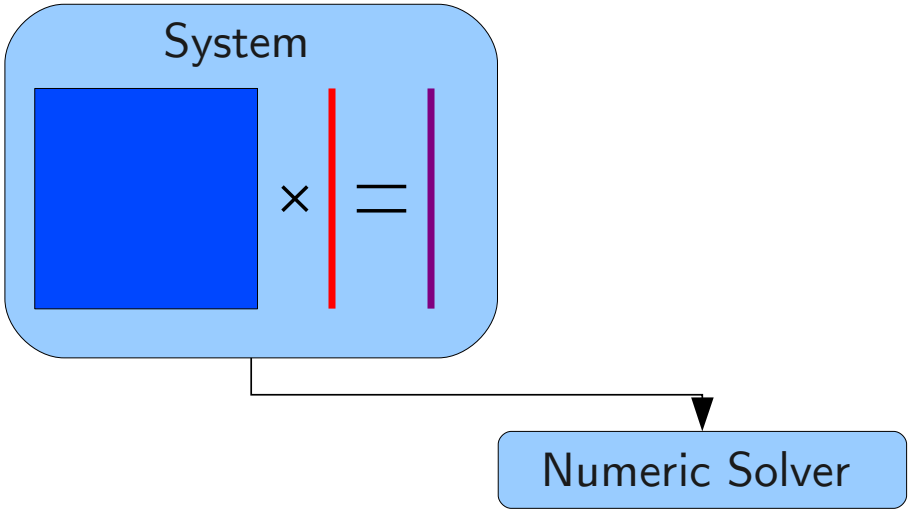
Before updating dyadic approximants, we *confirm continuation* of the iteration by verifying **overlap** between the current $\hat{x}'$ and the previous $\hat{x}_{frac}$.

- e.g. $\max |\hat{x}' - \hat{x}_{frac}| \leq \frac{1}{2^b}$ ensures b bits of overlap
- One bit is (typically) sufficient.

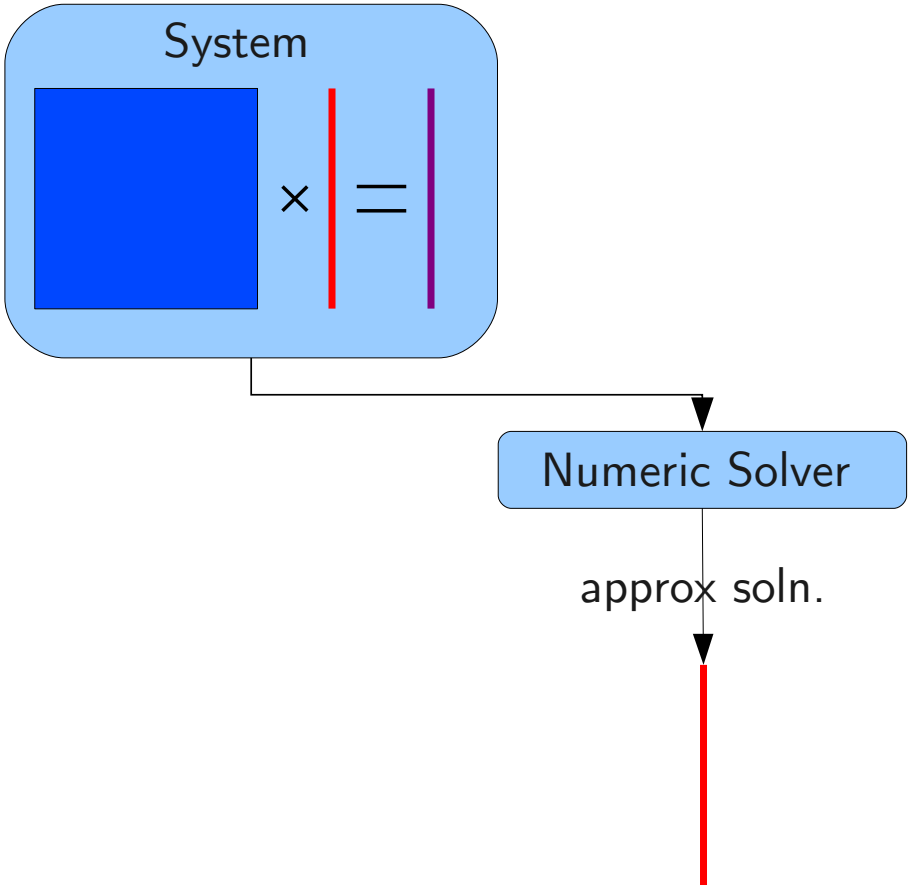
Iterative Refinement



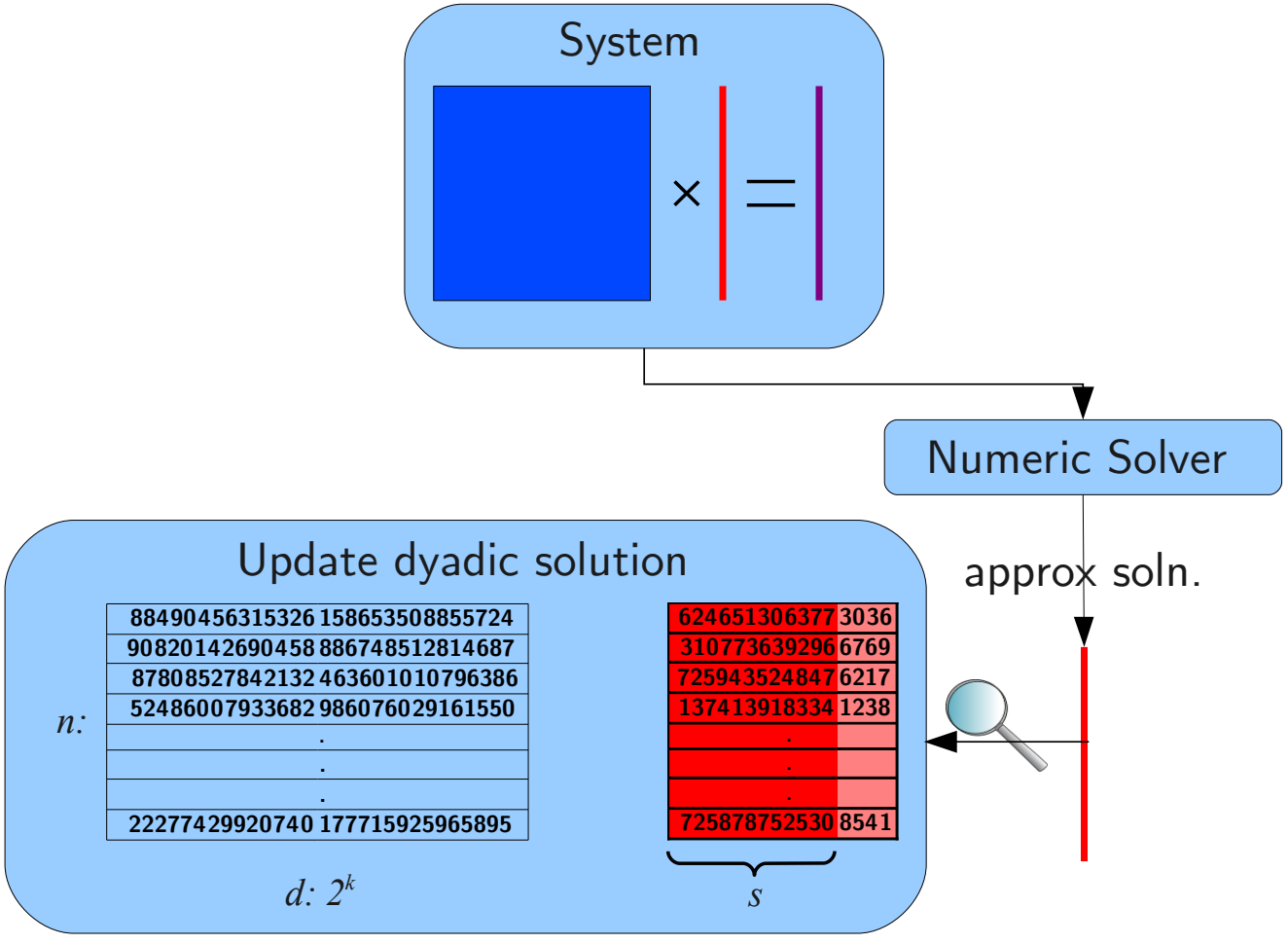
Iterative Refinement



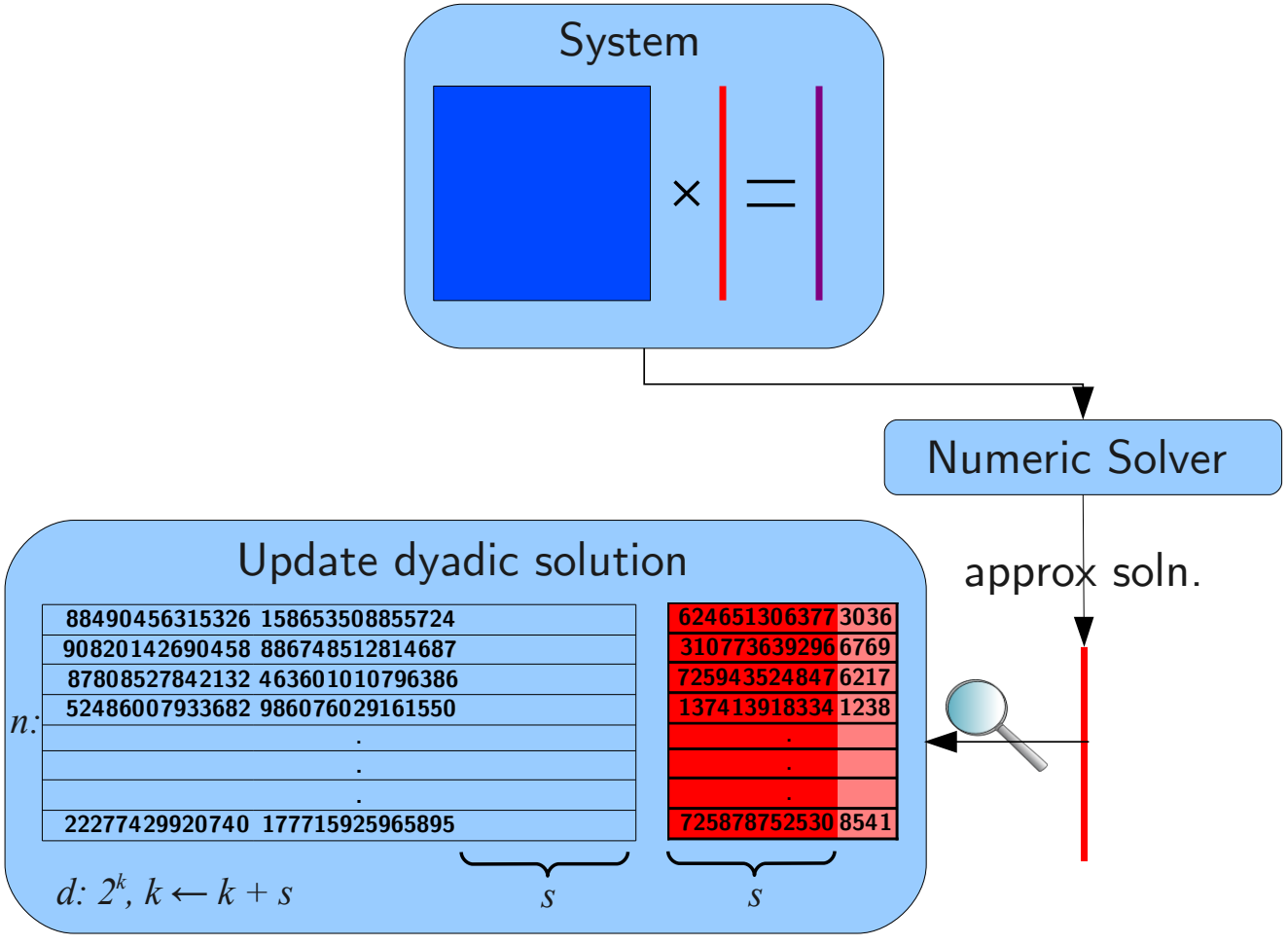
Iterative Refinement



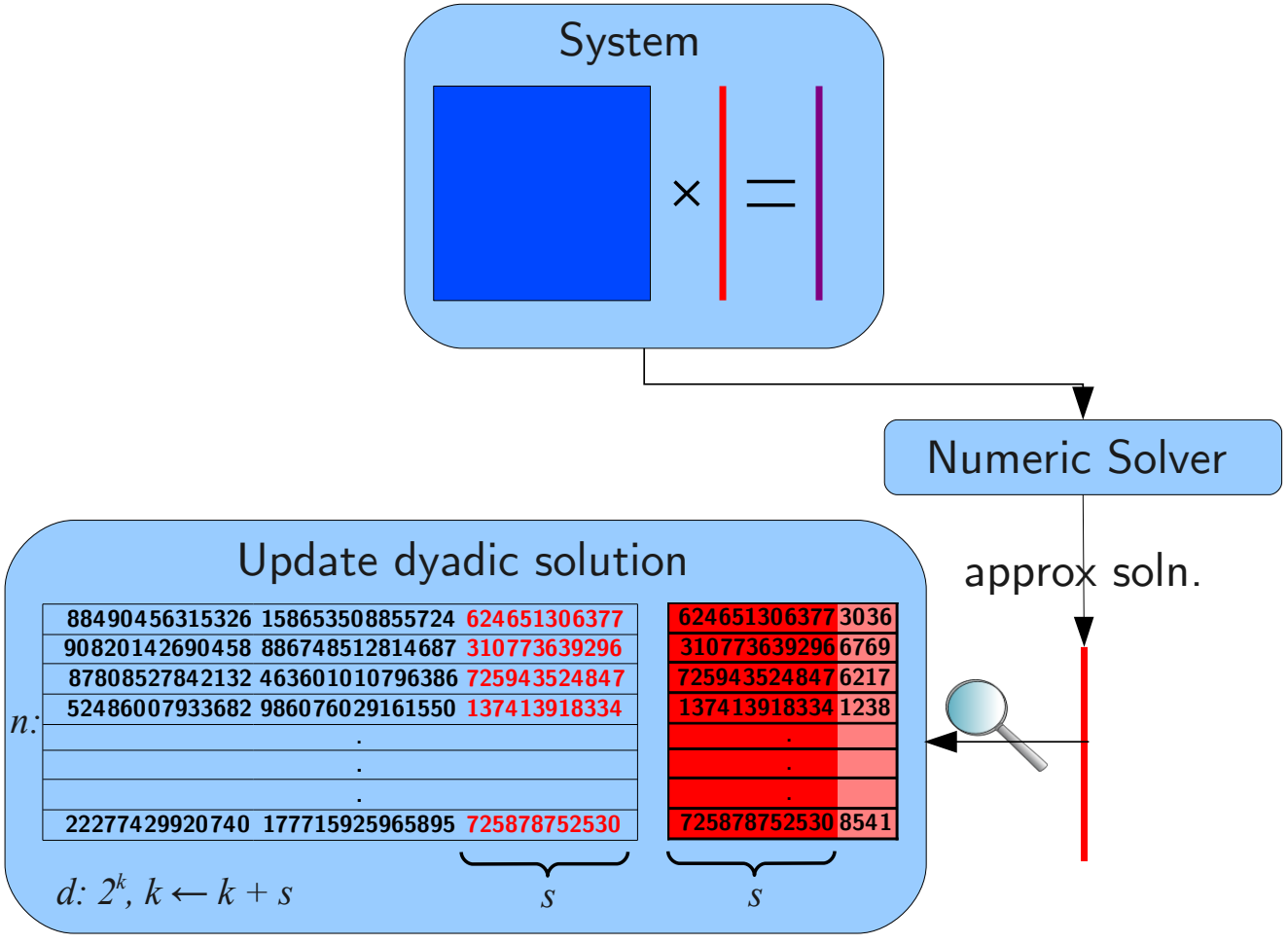
Iterative Refinement



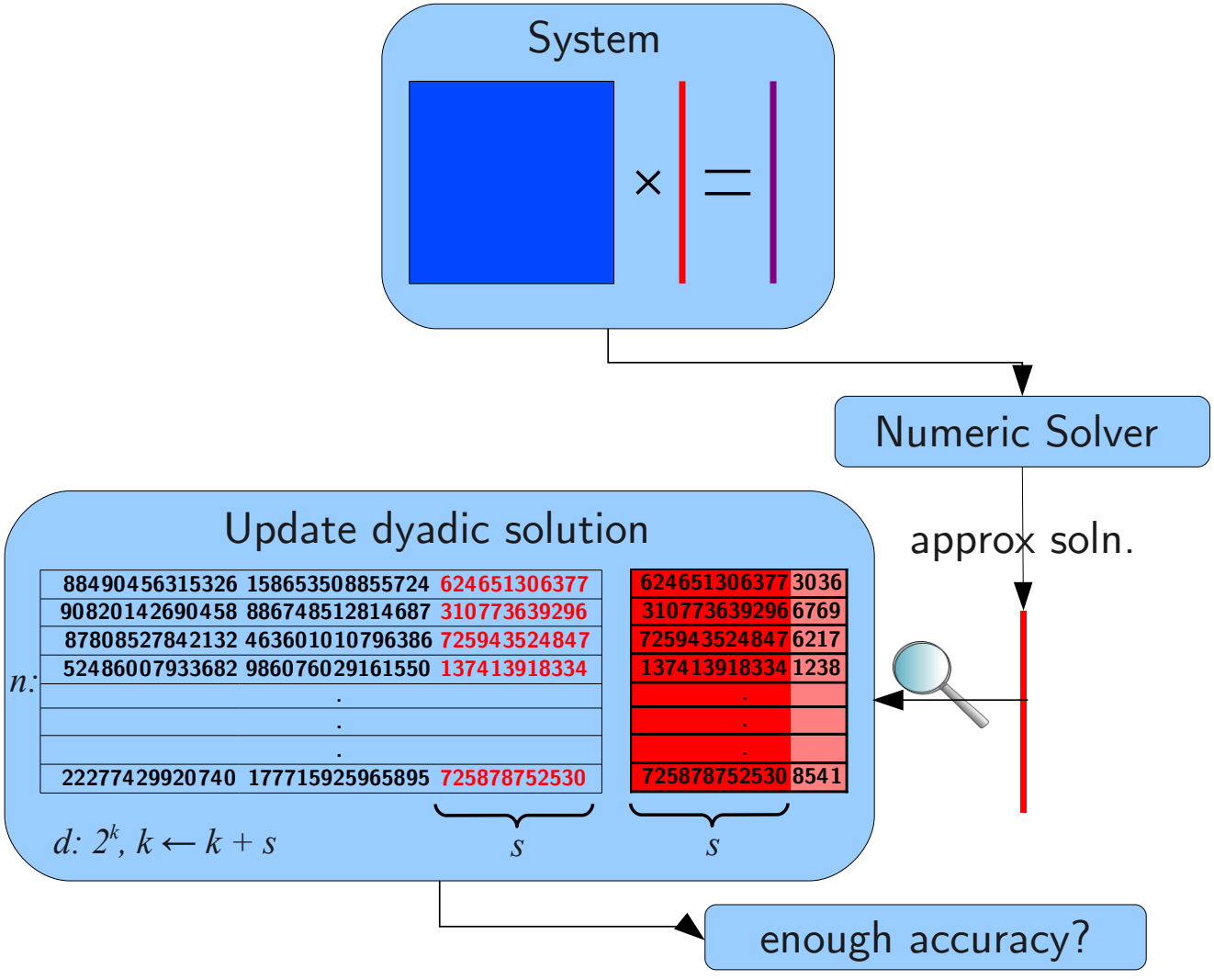
Iterative Refinement



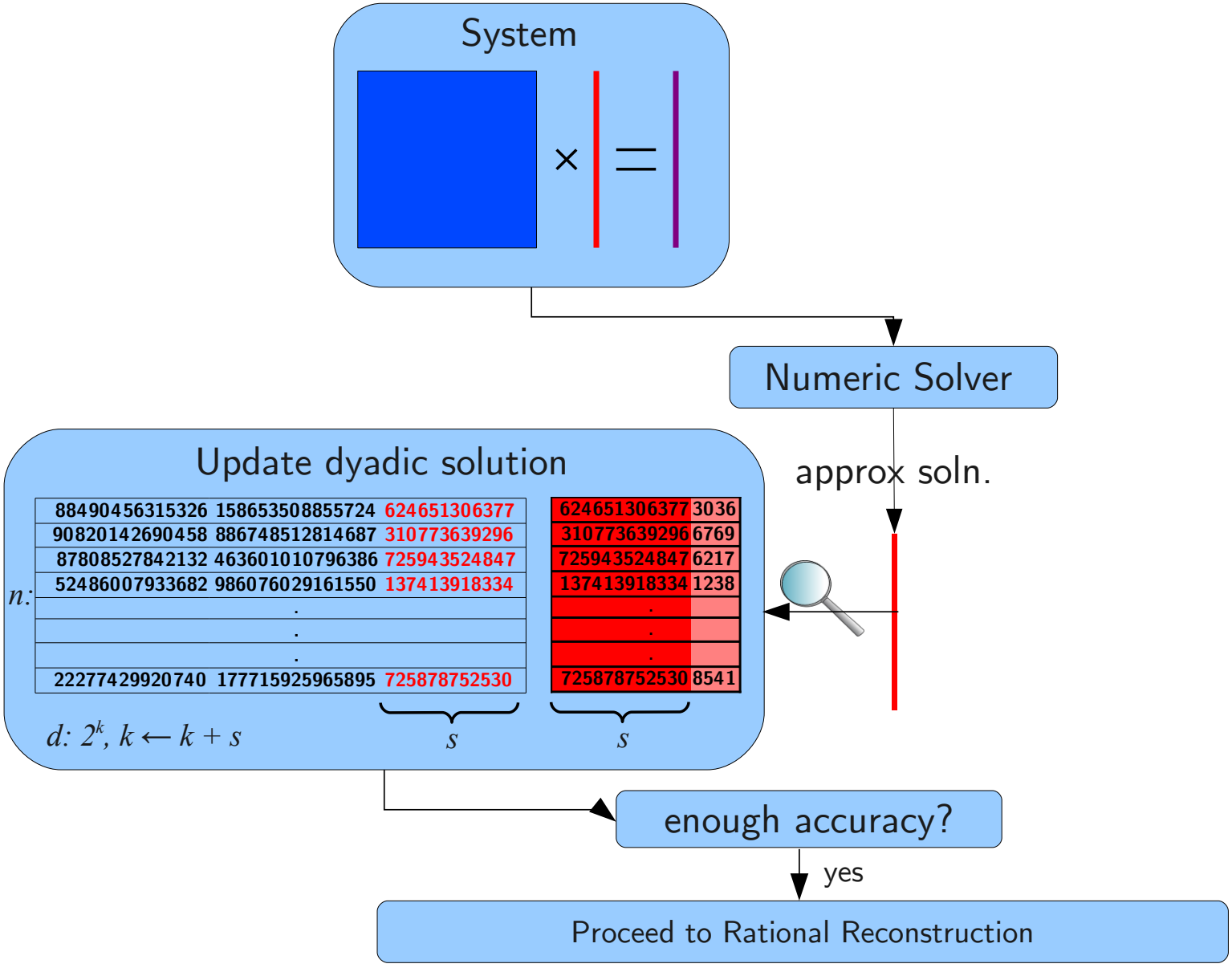
Iterative Refinement



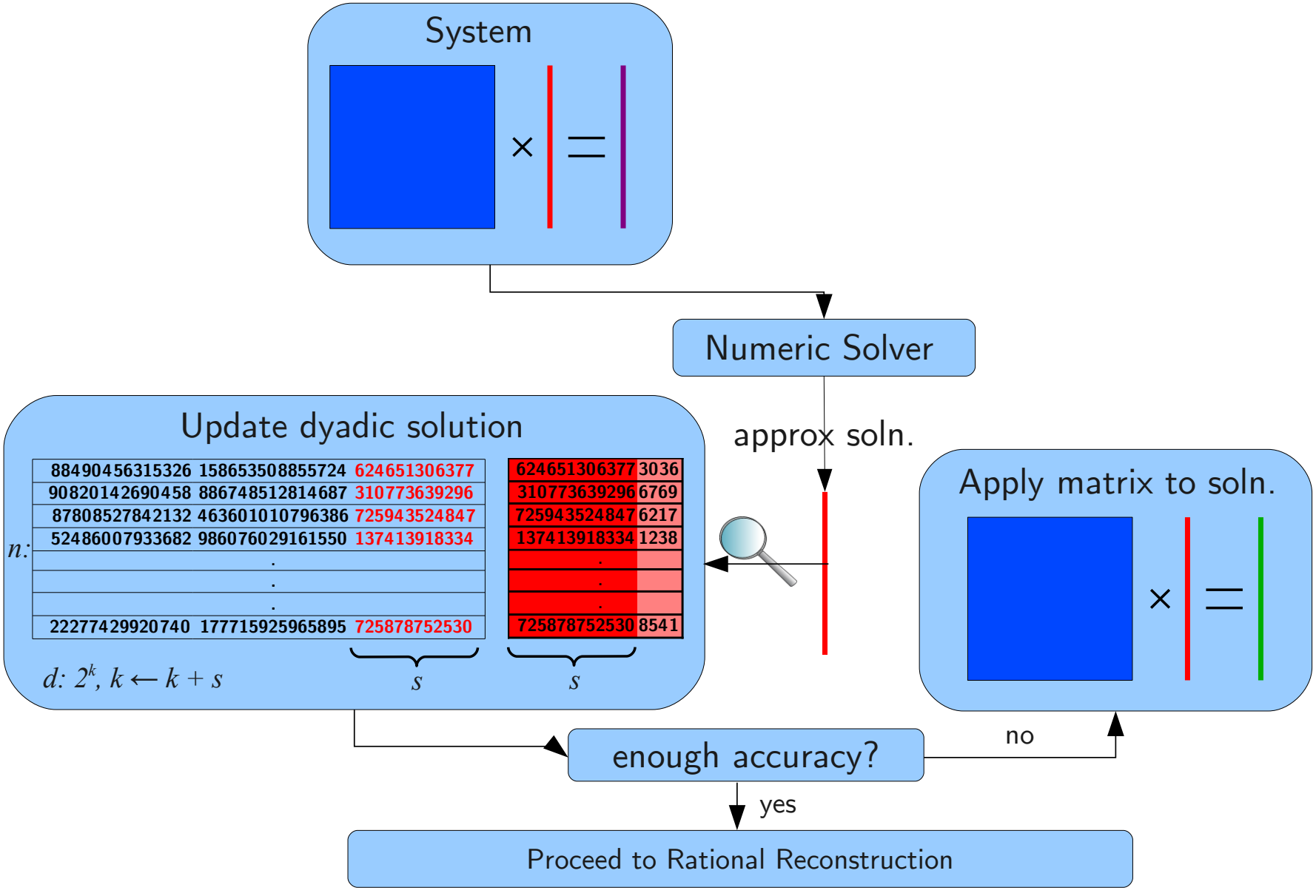
Iterative Refinement



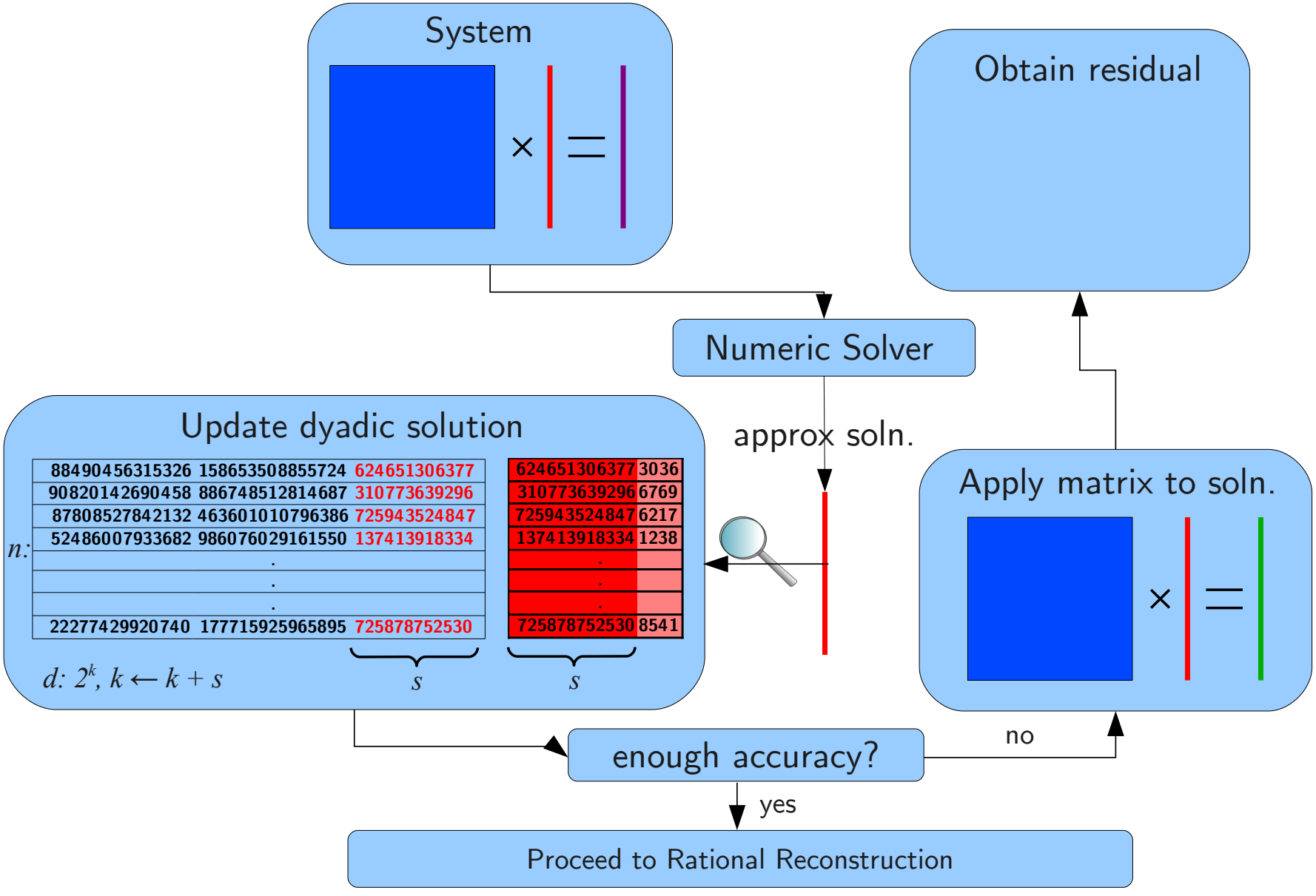
Iterative Refinement



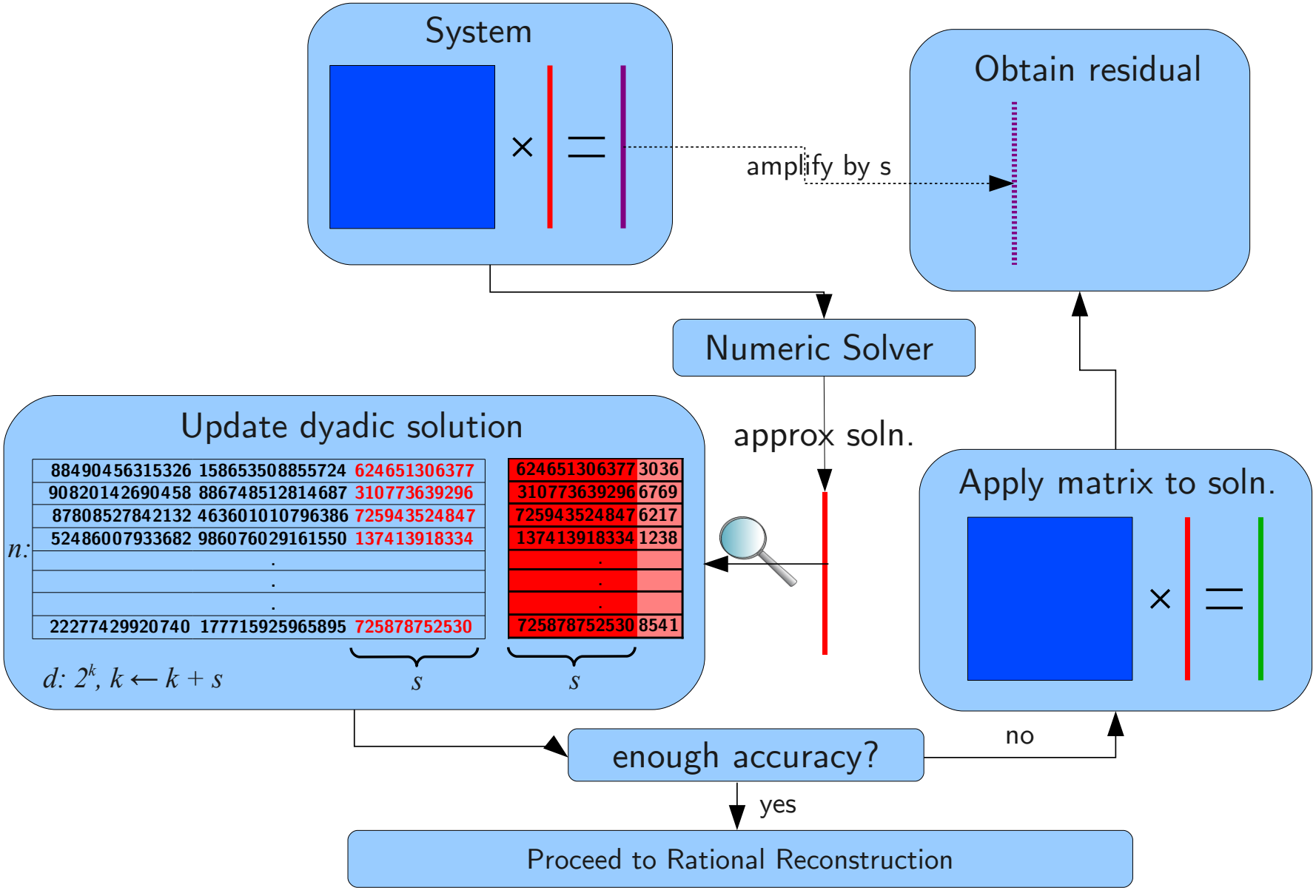
Iterative Refinement



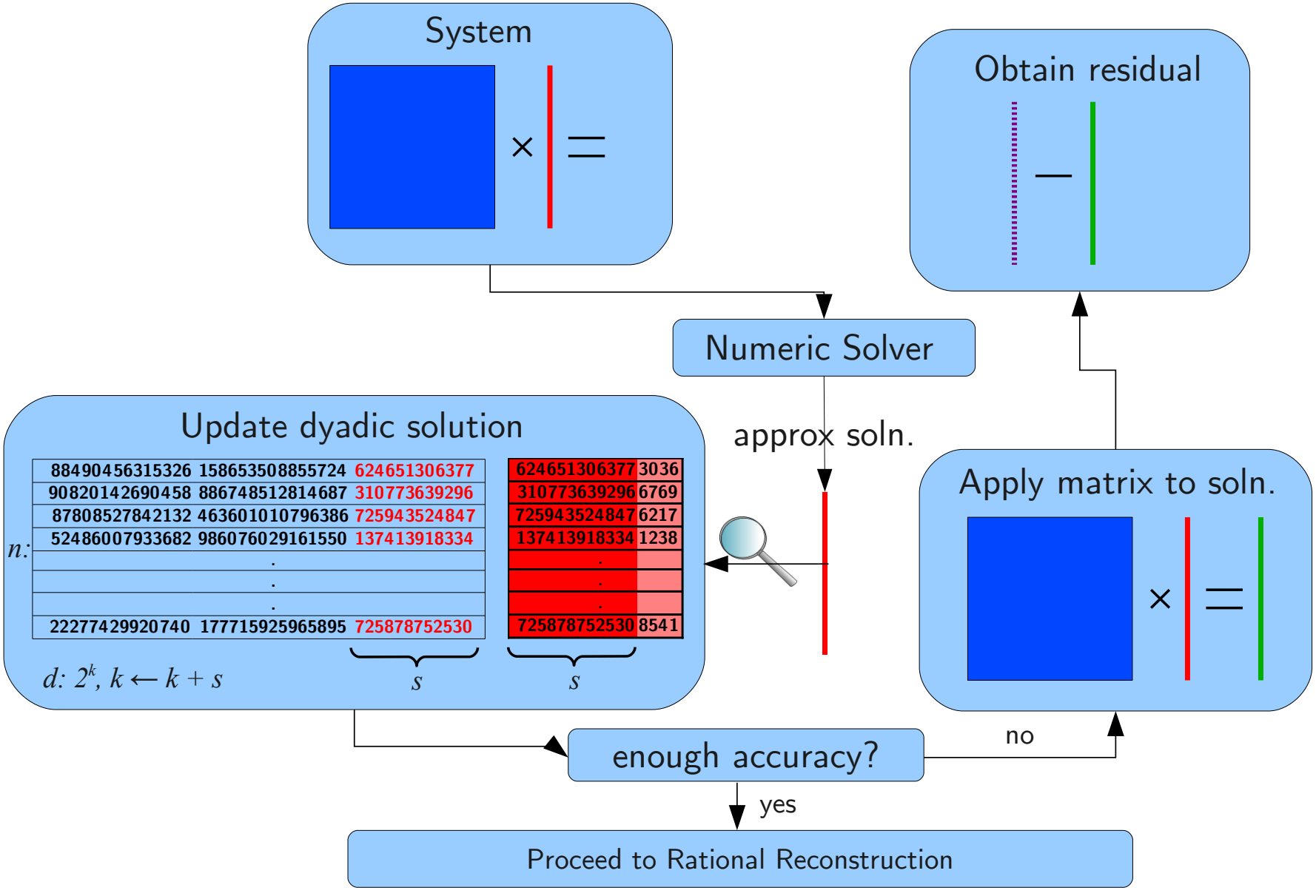
Iterative Refinement



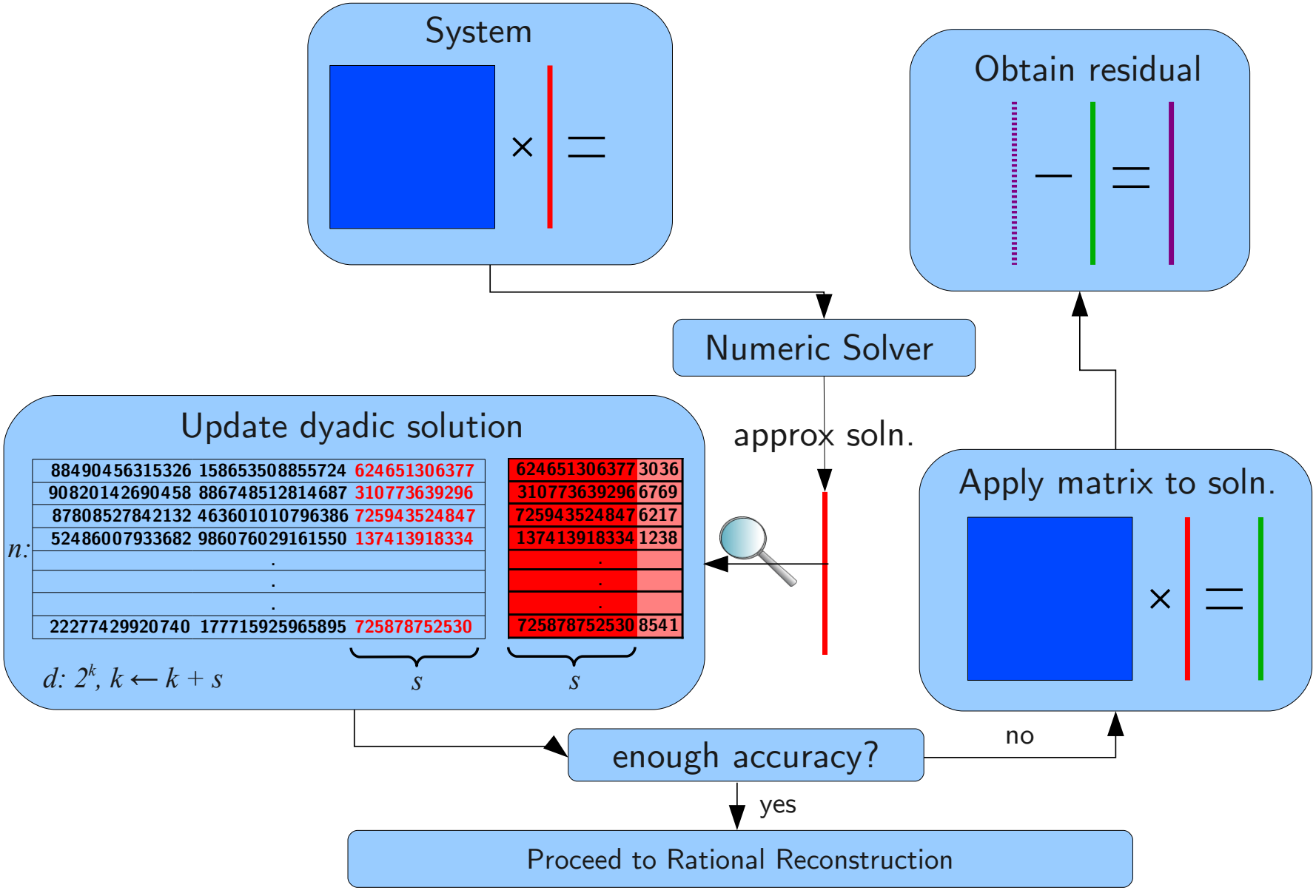
Iterative Refinement



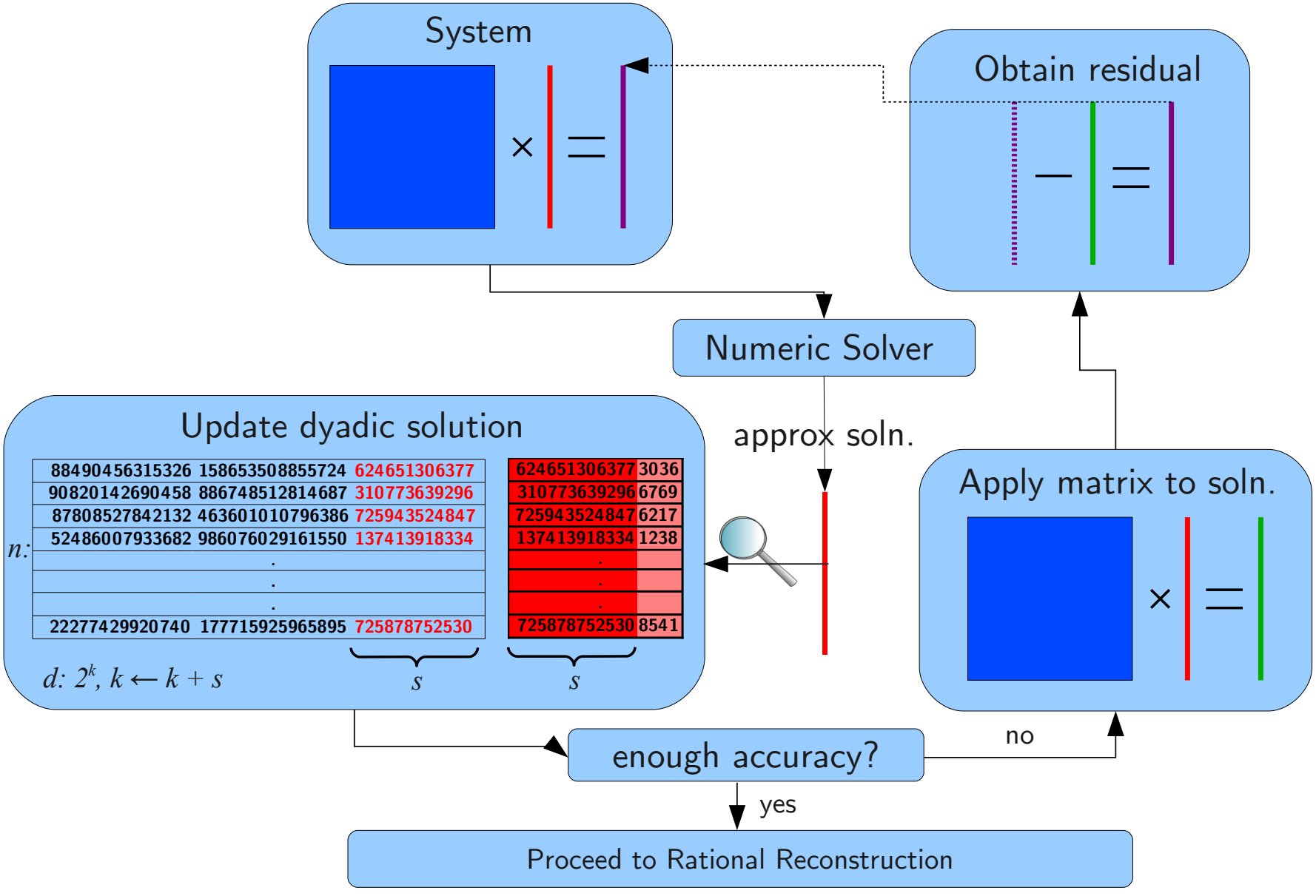
Iterative Refinement



Iterative Refinement



Iterative Refinement



Rational Reconstruction

- Reconstruct from p-adic digits $p_0, p_1, \ldots p_k$
  $n/d = \sum_{i=i_0}^{k} x_i p^i + \epsilon$
  where $p^{k+1} | \epsilon$, i.e. error is small in the p-adic sense.
- Reconstruct from dyadic digits $x_0, x_1, \ldots x_k$
  $n/d = \sum_{i=i_0}^{k} x_i / 2^i + \epsilon$
  where error ϵ is small, $|\epsilon| < 1/2^{k+1}$

Rational Reconstruction

- Reconstruct from p-adic digits $p_0, p_1, \ldots p_k$
  $n/d = \sum_{i=i_0}^{k} x_i p^i + \epsilon$
  where $p^{k+1} | \epsilon$, i.e. error is small in the p-adic sense.
- Reconstruct from dyadic digits $x_0, x_1, \ldots x_k$
  $n/d = \sum_{i=i_0}^{k} x_i / 2^i + \epsilon$
  where error ϵ is small, $|\epsilon| < 1/2^{k+1}$

In both cases, a Euclidean remainder sequence can recover n/d. However, there are **important differences** in the use of remainder sequence:

Rational Reconstruction

- Reconstruct from p-adic digits $p_0, p_1, \ldots p_k$
  $n/d = \sum_{i=i_0}^{k} x_i p^i + \epsilon$
  where $p^{k+1} | \epsilon$, i.e. error is small in the p-adic sense.
- Reconstruct from dyadic digits $x_0, x_1, \ldots x_k$
  $n/d = \sum_{i=i_0}^{k} x_i / 2^i + \epsilon$
  where error ϵ is small, $|\epsilon| < 1/2^{k+1}$

In both cases, a Euclidean remainder sequence can recover n/d. However, there are **important differences** in the use of remainder sequence:

- P-adically a remainder sequence entry is (r_i, s_i, t_i), where $r_i = s_i x + t_i p^k$, the remainder r_i is a *numerator candidate* and coefficient s_i of the given residue x is a denominator candidate.

Rational Reconstruction

- Reconstruct from p-adic digits $p_0, p_1, \ldots p_k$
  $n/d = \sum_{i=i_0}^{k} x_i p^i + \epsilon$
  where $p^{k+1} | \epsilon$, i.e. error is small in the p-adic sense.
- Reconstruct from dyadic digits $x_0, x_1, \ldots x_k$
  $n/d = \sum_{i=i_0}^{k} x_i / 2^i + \epsilon$
  where error ϵ is small, $|\epsilon| < 1/2^{k+1}$

In both cases, a Euclidean remainder sequence can recover n/d. However, there are **important differences** in the use of remainder sequence:

- P-adically a remainder sequence entry is (r_i, s_i, t_i), where $r_i = s_i x + t_i p^k$, the remainder r_i is a *numerator candidate* and coefficient s_i of the given residue x is a denominator candidate.
- Dyadically a remainder sequence entry is (r_i, s_i, t_i), where $r_i = s_i x + t_i 2^k$, the remainder r_i is a *measure of error* and coefficient s_i of the given dyadic approximation x is a denominator candidate.

Output sensitivity (early termination)

- No random choice of prime, therefore no probabilistic early termination.
- But *guaranteed* early termination in some cases:

Example

Solving $Ax = b$, suppose:

- $x_1 = 9/17$
- the Hadamard bound for $\det(A)$ is $h = 2^8 = 256$.

In general we need $2\lg(h) = 16$ bits of approximation.

Output sensitivity (example)

```
    |\^/|     Maple 12 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2008
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> #convergents of approximation to 7 bits.
> convert(68/128, confrac, 'cv'): cv;
                                              17
                            [0, 1, 1/2, 8/15, --]
                                              32
> #unique close small denominator convergent is wrong.
```

Output sensitivity (example)

```
    |\^/|     Maple 12 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2008
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> #convergents of approximation to 7 bits.
> convert(68/128, confrac, 'cv'): cv;
                                          17
                          [0, 1, 1/2, 8/15, --]
                                          32
> #unique close small denominator convergent is wrong.


> #convergents of approximation to 8 bits.
> convert(135/256, confrac, 'cv'): cv;
                                    10  19  29  135
                  [0, 1, 1/2, 9/17, --, --, --, ---]
                                    19  36  55  256
> #correct small rational is among several possibilities.
```

Output sensitivity (example)

```
    |\^/|     Maple 12 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2008
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> #convergents of approximation to 7 bits.
> convert(68/128, confrac, 'cv'): cv;
                                          17
                          [0, 1, 1/2, 8/15, --]
                                          32
> #unique close small denominator convergent is wrong.


> #convergents of approximation to 8 bits.
> convert(135/256, confrac, 'cv'): cv;
                                    10  19  29  135
                  [0, 1, 1/2, 9/17, --, --, --, ---]
                                    19  36  55  256
> #correct small rational is among several possibilities.


> #convergents of approximation to 9 bits.
> convert(271/512, confrac, 'cv'): cv;
                                          271
                          [0, 1, 1/2, 9/17, ---]
                                          512
> #unique close small denominator convergent is right, but we can't be certain.
```

Output sensitivity (example)

```
> #convergents of approximation to 12 bits.
> convert(2168/4096, confrac, 'cv'): cv;
                                                271
                              [0, 1, 1/2, 9/17, ---]
                                                512

> #guaranteed early termination: small denominator convergent is only possible solution.
```

Output sensitivity (example)

```
> #convergents of approximation to 12 bits.
> convert(2168/4096, confrac, 'cv'): cv;
                                          271
                          [0, 1, 1/2, 9/17, ---]
                                          512

> #guaranteed early termination: small denominator convergent is only possible solution.


> #convergents of approximation to 16 bits.
> convert((32*1084)/65536, confrac, 'cv'): cv;
                                          271
                          [0, 1, 1/2, 9/17, ---]
                                          512

> #Get same result from full approximation, not using output sensitivity.
```

Vector rational reconstruction

Suppose solution to Ax = b is (9/17, 3/11, 7/187) and we have computed the approximation to 12 bits: $\frac{x_1=2168, x_2=1117, x_3=153}{4096}$

Vector rational reconstruction

Suppose solution to Ax = b is (9/17, 3/11, 7/187) and we have computed the approximation to 12 bits: $\frac{x_1=2168, x_2=1117, x_3=153}{4096}$

- x_1: As we saw, 9/17 is found with certain correctness.
  17 = lcm of denominators so far.

Vector rational reconstruction

Suppose solution to Ax = b is (9/17, 3/11, 7/187) and we have computed the approximation to 12 bits: $\frac{x_1=2168, x_2=1117, x_3=153}{4096}$

- x_1: As we saw, 9/17 is found with certain correctness.
  17 = lcm of denominators so far.
- x_2: Try division once $(1117 * 17 = 5 * 4096 - 1491)$.
  Try again – success $(1117 * 187 = 51 * 4096 - 17)$
  $n/d = 51/187 = 3/11$
  187 = lcm of denominators so far.

Vector rational reconstruction

Suppose solution to Ax = b is (9/17, 3/11, 7/187) and we have computed the approximation to 12 bits: $\frac{x_1=2168, x_2=1117, x_3=153}{4096}$

- x_1: As we saw, 9/17 is found with certain correctness.
  17 = lcm of denominators so far.
- x_2: Try division once $(1117 * 17 = 5 * 4096 - 1491)$.
  Try again – success $(1117 * 187 = 51 * 4096 - 17)$
  $n/d = 51/187 = 3/11$
  187 = lcm of denominators so far.
- x_3: $|n/187 - 153/4096|$ should be less than $1/8192$.
  One division with remainder yields $153 * 187 = 7 * 4096 - 61$.
  n = 7 and small enough remainder.

Data

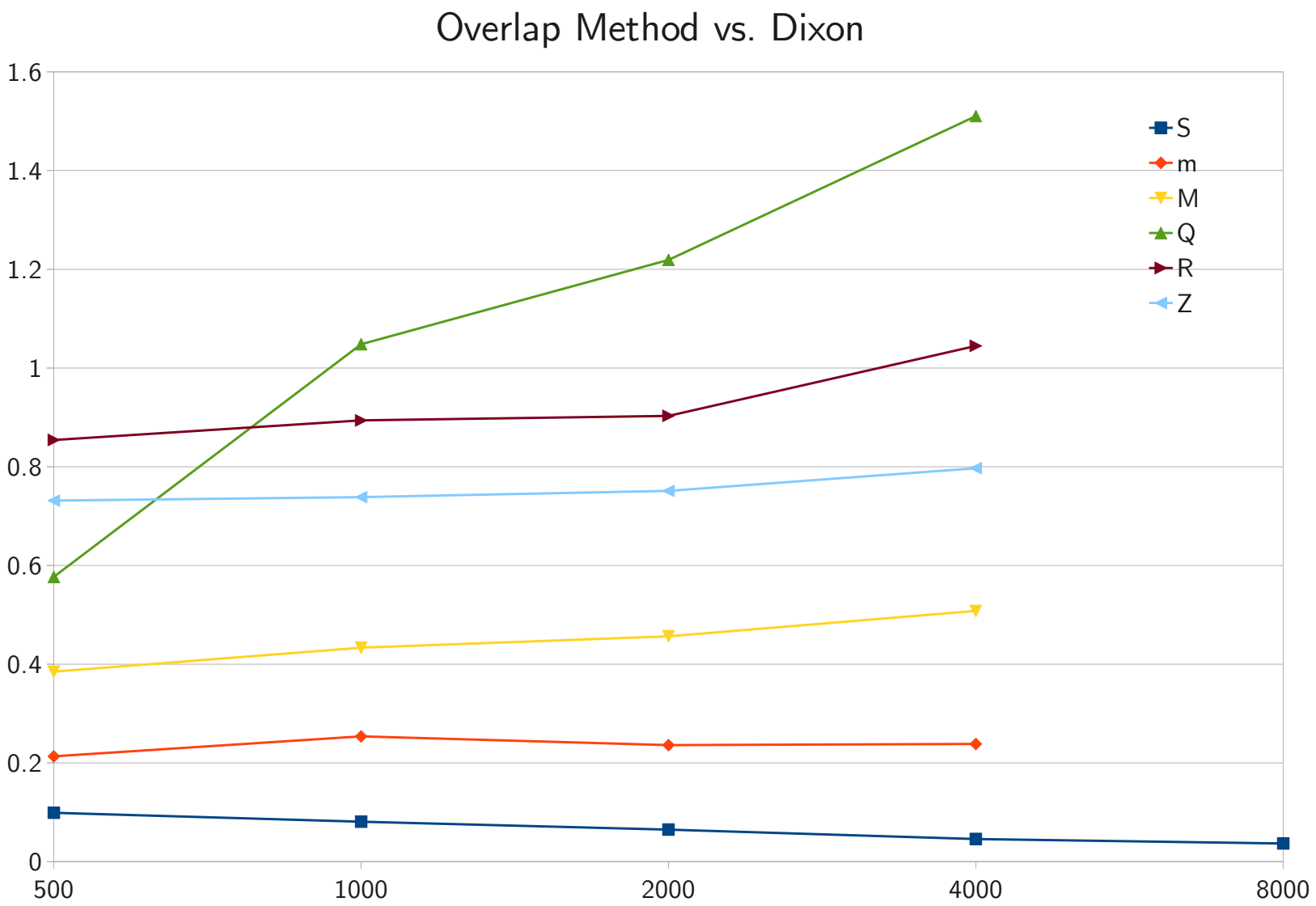
matrix	Dixon	Wan	Overlap	Ov-ET
S_{512}	0.728	0.711	0.0723	**0.0721**
S_{1024}	4.58	4.75	0.380	**0.371**
S_{2048}	32.3	36.6	**2.08**	2.10
S_{4096}	255	297	**11.7**	**11.7**
S_{8192}	2240	2517	**77.6**	82.6
M_{500}	1.46	fail	1.06	**0.562**
M_{1000}	10.2	fail	8.56	**4.42**
M_{2000}	82.8	fail	75.0	**37.8**
M_{4000}	628	fail	658	**319**
M_{8000}	mem	fail	6170.6	**3049**
Z_{500}	0.793	0.864	0.584	**0.580**
Z_{1000}	6.04	6.38	**4.41**	4.46
Z_{2000}	45.4	52.15	35.8	**34.1**
Z_{4000}	340	439	318	**271**
Z_{8000}	mem	5771	2584	**2474**

$$S_1 = (1), S_{2n} = \begin{pmatrix} S_n & S_n \\ S_n & -S_n \end{pmatrix}$$

$$M = \begin{pmatrix} 1 & 2 & 3 & \cdots \\ 2 & 2 & 3 & \cdots \\ 3 & 3 & 3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Z_n is an $n \times n$ $\{0, 1\}$-matrix, with p $= 1/2$ for a given position containing a 1.

Relative Performance



The End