# Algorithms for Verified Linear Algebra
# Verifying the solution of a linear system

## Cours de recherche master informatique

2 December 2011

# Agenda

# verifylss by S. Rump

- **1st step: approximate solving of the system $A_c x = b_c$ :**

```
R = inv(Ac),   x = R * bc
while (not enough) loop   % iterative refinement
    res = bc - Ac * x     % residual computation
    e = R * res,   x = x + e
```

- **2nd step: verification = enclosure of the error**

1st attempt: $\varepsilon$-inflation and Krawczyk iteration

```
[z] = R * ([b]-[A]*x)
while (not enough) loop
   [y] = epsilon-inflate([z])
   [y'] = [z] + (I-R*[A])*[y]
if [y'] in [y] then  return x, [y']
```

2nd attempt: Hansen-Bliek-Rohn-Ning-Kearfott formula

```
[b] = [b] - [A]*(x+[y]),   [A] = R*[A]
...
```

# Comments on `verifylss`

- Complexity: polynomial in time . . . for an NP-hard problem:
  no guarantee on the accuracy of the solution,
  failure is possible (either $\varepsilon$-inflation or
  Hansen-Bliek-Rohn-Ning-Kearfott if **A** is not an H-matrix).
- This algorithm is usually employed to verify the solution of a linear system with floating-point coefficients:
  interval arithmetic is used as a verification tool.

# Comments on `verifylss`

- stopping criterion for the first loop (iterative refinement on the linear system formed by midpoints of **A** and **b**):
  - number of iterations $< 15$, and
  - norm of new residual $< 0.1$ times norm of former residual.

  If this last criterion is not satisfied, the 1st step can also fail.
- stopping criterion for the second loop (determination of an enclosure of the error):
  - number of iterations $< 7$, and
  - Brouwer theorem did not apply and thus it is not proven that $[y]$ contains the error.

# Hansen-Bliek-Rohn-Ning-Kearfott formula

**Theorem**

Let $\mathbf{A} \in I\!R^{n\times n}$ be an H-matrix and $\mathbf{b} \in I\!R^{n}$.

Let us denote

$$\begin{aligned} z &= < A >^{-1} \cdot|\mathbf{b}| \\ d &= \text{diag}\,(< A >^{-1}) \ : \ d_i = ((< A >^{-1})_{i,i} \\ \alpha_i &= < A_{i,i} > -1/d_i \\ \beta_i &= \frac{z_i}{d_i} - |\mathbf{b}_i|. \end{aligned}$$

Then it holds that $\Sigma_{\exists\exists}(\mathbf{A}, \mathbf{b}) \subset \mathbf{x}$ with

$$\mathbf{x}_i = \frac{\mathbf{b}_i + [-\beta_i, \beta_i]}{\mathbf{A}_{i,i} + [-\alpha_i, \alpha_i]}.$$

Both are equal is $\mathbf{A}$ is diagonal.

# Hansen-Bliek-Rohn-Ning-Kearfott formula ... in MatLab

The system considered here is the residual system, where the matrix is premultiplied by an approximate inverse: $\mathbf{A} = R \cdot \mathbf{A}_{init}$ and $\mathbf{b} = \mathbf{b}_{init} - \mathbf{A}_{init}\mathbf{x}$.

```
cA = compmat (A)
B = inv(cA)
v = abs(B * ones(n,1))
setRound(-1)
u = cA * v
if ~ (all(u) > 0)
   x = NaN, return
else
   Res = cA * B - eye(n)
   setRound(1)
   w = ...
   d = v .* w' - diag(B)
   B = B + v * w'
   z = B * abs(b)
   beta = z ./ d - abs(b)
   alpha = diag(cA) + (-1) ./ diag(B)
   x=(b+midrad(0,beta))./(diag(A)+midrad(0,alpha))
```

# Agenda

# Problem: verified solution of a linear system

**Goals:** For a linear system $Ax = b$ with $A \in \mathbb{F}^{n \times n}$ non-singular and $b \in \mathbb{F}^n$, we want to

1. compute an approximation $\tilde{x} \in \mathbb{F}^n$ of the exact solution $x^*$,
2. simultaneously bound the error upon $\tilde{x}$, or enclose it in an interval

$$\mathbf{e} \ni x^* - \tilde{x}.$$

**Remark:** denote by $e$ the error $x^* - \tilde{x}$.
Then $e$ is the solution of the residual system $Ae = b - A\tilde{x}$.
Indeed, $Ae = A(x^* - \tilde{x}) = Ax^* - A\tilde{x} - b - A\tilde{x}$.

# Method: contractant iteration
# Classical iterative refinement

**Wilkinson (1963), Higham (2000), Demmel et al. (2006) ..**

## Algorithm (Classical iterative refinement)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like syntax*

*while(not converged)*

$\tilde{r} = b - A\,\tilde{x}$

$\tilde{e} = A \setminus \tilde{r}$

$\tilde{x} = \tilde{x} + \tilde{e}$

*end*

*Output:* $\tilde{x}$

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like syntax*

*while(not converged)*

$\tilde{r} = b - A\,\tilde{x}$

$\tilde{e} = A \setminus \tilde{r}$

$\tilde{x} = \tilde{x} + \tilde{e}$

*end*

*Output:* $\tilde{x}$

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like syntax*

*while(not converged)*

$\mathbf{r} = [b - A\,\tilde{x}]$ *%* $A(x^* - \tilde{x}) \in \mathbf{r}$

$\tilde{e} = A \setminus \tilde{r}$

$\tilde{x} = \tilde{x} + \tilde{e}$

*end*

*Output:* $\tilde{x}$

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like syntax*

*while(not converged)*

$\mathbf{r} = [b - A\,\tilde{x}]$ *%* $A(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = A \setminus \mathbf{r}$ *%* $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \tilde{e}$

*end*

*Output:* $\tilde{x}$

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like syntax*

*while(not converged)*

$\mathbf{r} = [b - A\,\tilde{x}]$ *%* $A(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = A \setminus \mathbf{r}$ *%* $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\tilde{x}$

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like syntax*

*while(not converged)*

$\mathbf{r} = [b - A\,\tilde{x}]$ *%* $A(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = A \setminus \mathbf{r}$ *%* $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like syntax*

*while(not converged)*

$\mathbf{r} = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = A \setminus \mathbf{r}$ % $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

## Solving interval residual system?

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

*while(not converged)*

$\quad \mathbf{r} = [b - A\,\tilde{x}] \qquad \% \quad A(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = A \setminus \mathbf{r} \qquad \% \quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

**K** is close to Identity $\Rightarrow$ there are algorithms to solve this system.

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

*while(not converged)*

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}] \qquad \%\quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = A \setminus \mathbf{r} \qquad \%\quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

**K** is close to Identity $\Rightarrow$ there are algorithms to solve this system.

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

*while(not converged)*

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}] \qquad \%\quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = \mathbf{K} \setminus \mathbf{r} \qquad \%\quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

**K** is close to Identity $\Rightarrow$ there are algorithms to solve this system.

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

*while(not converged)*

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}] \qquad \%\quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = \mathbf{K} \setminus \mathbf{r} \qquad \%\quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

**K** is close to Identity $\Rightarrow$ there are algorithms to solve this system.

# Method: contractant iteration
# Interval iterative refinement

**Neumaier (1990), Rump (1999)**

## Algorithm (certifylss)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

*while(not converged)*

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}] \qquad \%\quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = \mathbf{K} \setminus \mathbf{r} \qquad \%\quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

This algorithm can fail, if it fails to solve the interval linear system.

# Differences with `verifylss`

**Iterative refinement performed on the interval residual.**

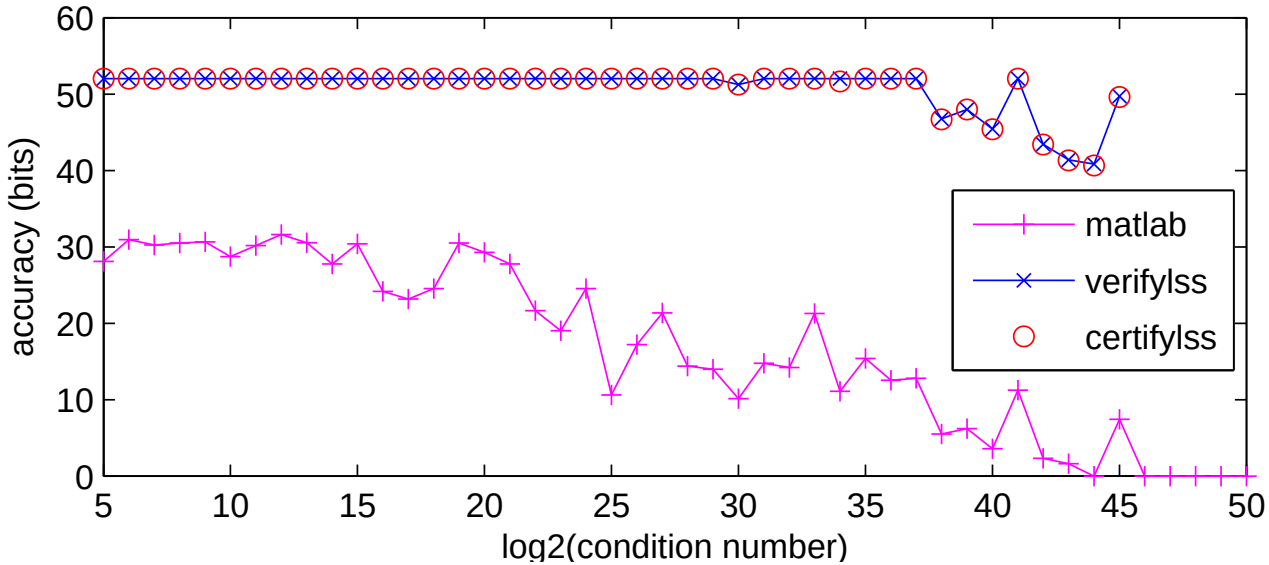
- initialization of $\mathbf{e}$: heuristic trying to determine $\mathbf{e}_0$, based on
  **Proposition:** let $A \in \mathbb{F}^{n\times n}$ and $R \in \mathbb{F}^{n\times n}$ be a floating-point approximate inverse of $A$.
  If $< [RA] > u \geq v > 0$ for some $u > 0$ then

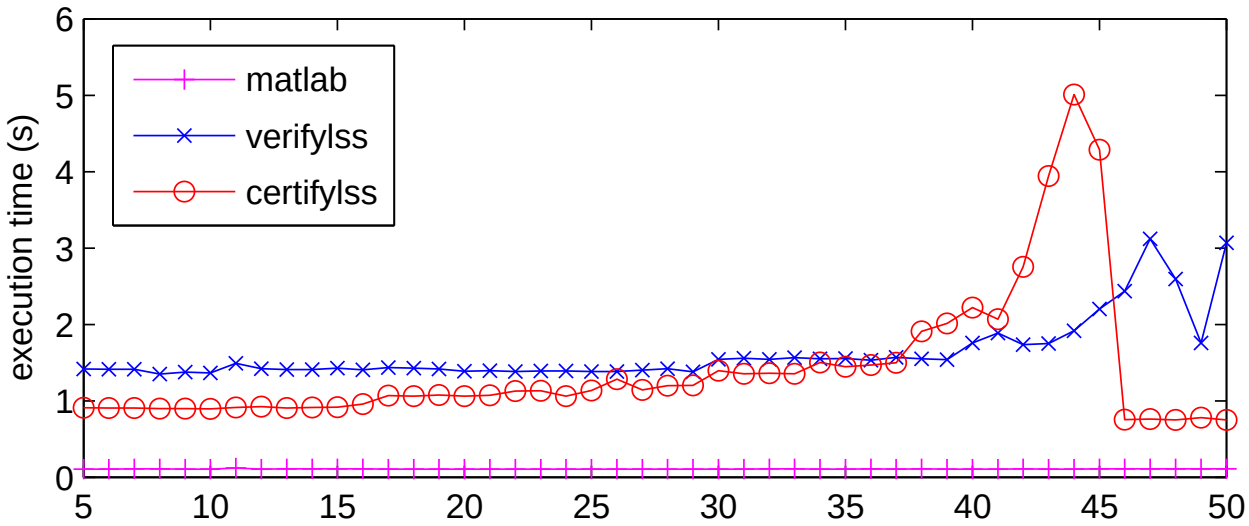
$$\begin{array}{rcl} |A^{-1}\mathbf{r}| & \leq & \|R\mathbf{r}\|_v u \\ A^{-1}\mathbf{r} & \subset & \|R\mathbf{r}\|_v [-u, u]. \end{array}$$

  Idea: start from $u = e = (1, 1, \ldots 1)^t$ and modify $u$ if $v$ is not $\geq 0$.
  Failure of the algo if failure of this step.
- solve $\mathbf{Ke} = \mathbf{r}$ using Gauss-Seidel iteration: known to converge quicker than Krawczyk.

# Experimental Results

Dimension: 1000 $b = [1, \ldots, 1]^T$



# Method: contractant iteration
# Relaxed interval iterative refinement

## Algorithm (certifylss_relaxed)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

*while(not converged)*

$\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $RA(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = \mathbf{K} \setminus \mathbf{r}$ % $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

# Method: contractant iteration
# Relaxed interval iterative refinement

## Algorithm (certifylss_relaxed)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ *% $\hat{\mathbf{K}}$ is centered on a diagonal matrix*

*while(not converged)*

$\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $RA(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = \mathbf{K} \setminus \mathbf{r}$ % $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

# Method: contractant iteration
# Relaxed interval iterative refinement

## Algorithm (certifylss_relaxed)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

*while(not converged)*

$\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $RA(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$ % *cost: 1 floating-point matrix-vector product*

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

# Method: contractant iteration
# Relaxed interval iterative refinement

## Algorithm (certifylss_relaxed)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

*while(not converged)*

$\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $RA(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$ % *cost: 1 floating-point matrix-vector product*

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

# Relaxed method: details

The product of **A** centered in zero by **B** is $[-\bar{A} * \mathrm{mag}\mathbf{B}, \bar{A} * \mathrm{mag}\mathbf{B}]$, i.e. 1 FP matrix product.

Let us decompose **A** as $\mathbf{D} + \mathbf{L} + \mathbf{U}$. Then

$$\begin{array}{ll} \text{Jacobi:} & \mathbf{e}' = \mathbf{D}^{-1}(\mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{e}) \\ \text{Gauss-Seidel:} & \mathbf{e}' = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{L}\mathbf{e}' - \mathbf{U}\mathbf{e}) \end{array}$$

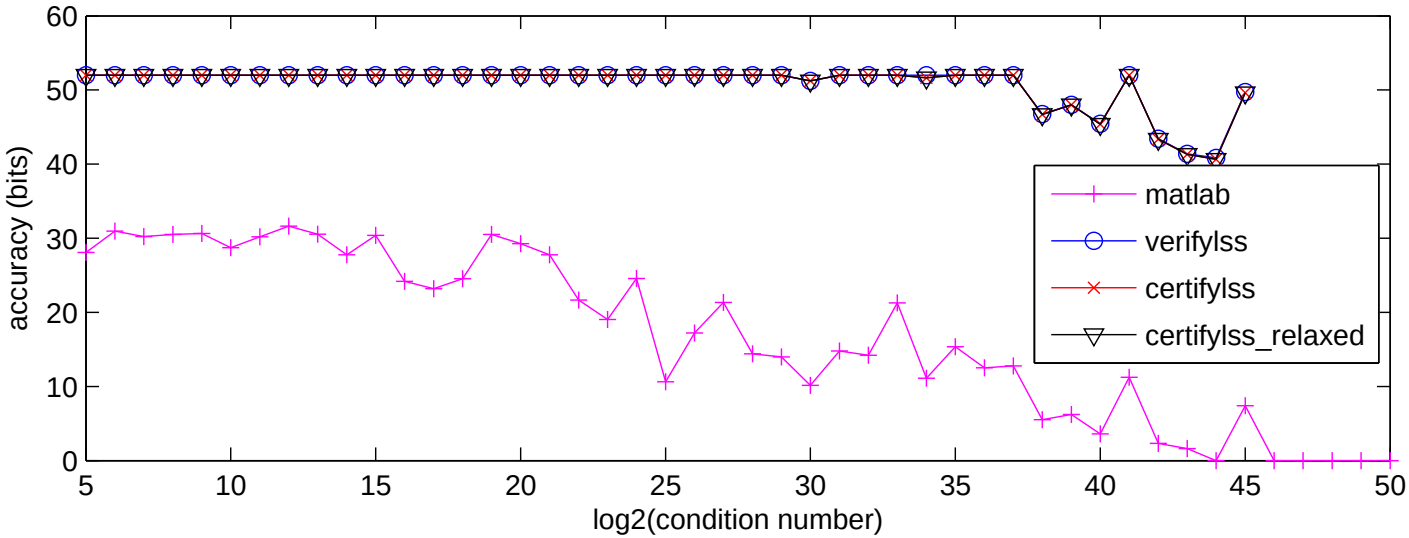
If **L** and **U** are inflated so as to be centered in 0:

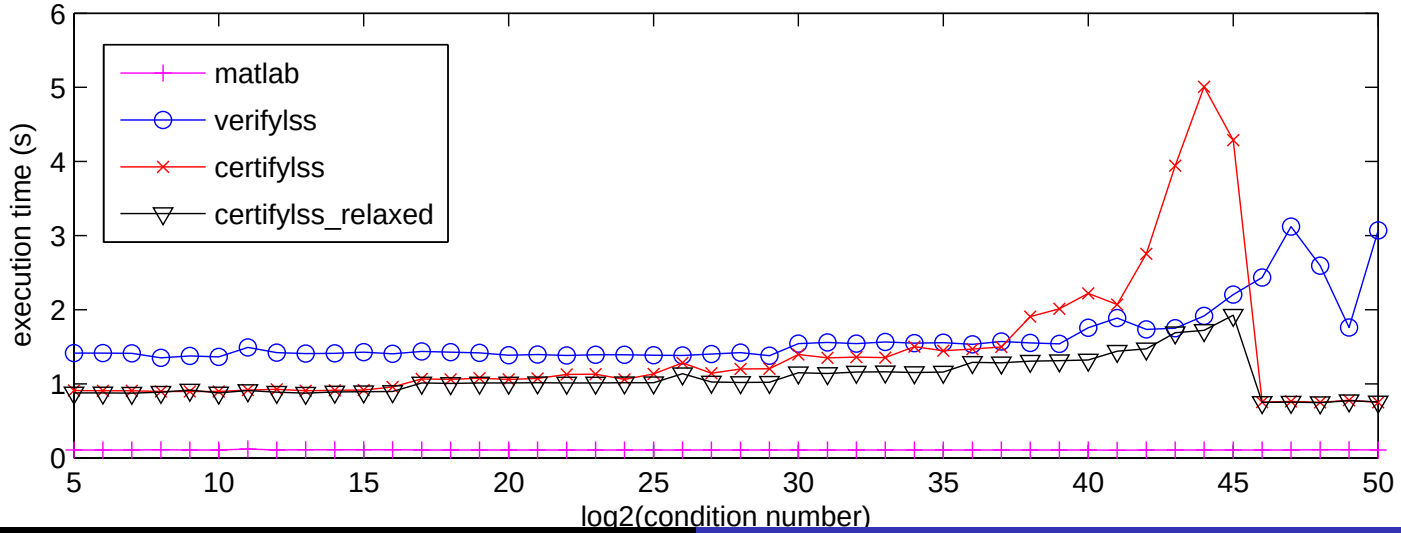
$$\mathbf{L}' = [-|\mathbf{L}|, |\mathbf{L}|] \quad \text{and} \quad \mathbf{U}' = [-|\mathbf{U}|, |\mathbf{U}|]$$

then Jacobi or Gauss-Seidel costs 1 FP matrix-vector product.
A BLAS2 routine can be used.
Convergence remains linear.
Accuracy of the solution: at most twice as wide as HBRNK.

# Relaxed method: Results

Dimension: 1000 $b = [1, \ldots, 1]^T$



# Method: contractant iteration
# Extra-precise relaxed interval iterative refinement

## Algorithm (certifylssx)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ *% $\hat{\mathbf{K}}$ is centered on a diagonal matrix*

*while(not converged)*

$\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$

$\mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$

$\mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

# Method: contractant iteration
# Extra-precise relaxed interval iterative refinement

## Algorithm (certifylssx)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ *% $\hat{\mathbf{K}}$ is centered on a diagonal matrix*

*while(not converged)*

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ *% $\mathbf{r}$ in twice the working precision*

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$

$\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

# Method: contractant iteration
# Extra-precise relaxed interval iterative refinement

## Algorithm (certifylssx)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

*while(not converged)*

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $\mathbf{r}$ *in twice the working precision*

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$ % $\tilde{x}$ *in twice the working precision*

$\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

*end*

*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

# Method: contractant iteration
# Extra-precise relaxed interval iterative refinement

## Algorithm (certifylssx)

*Input:* $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

*while(not converged)*

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $\mathbf{r}$ *in twice the working precision*

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$ % $\tilde{x}$ *in twice the working precision*

$\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

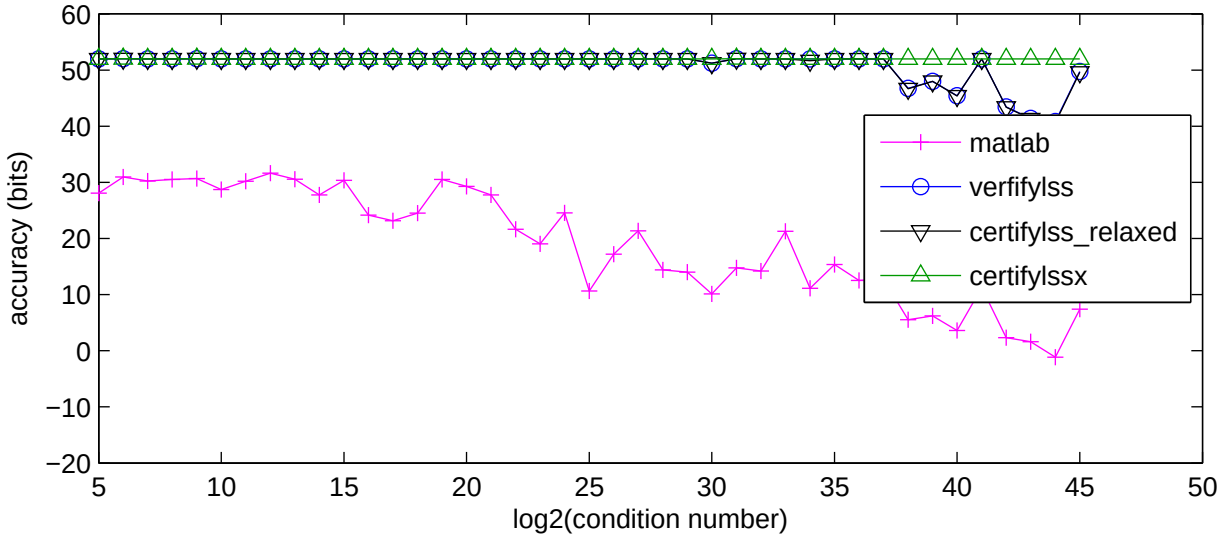
*end*

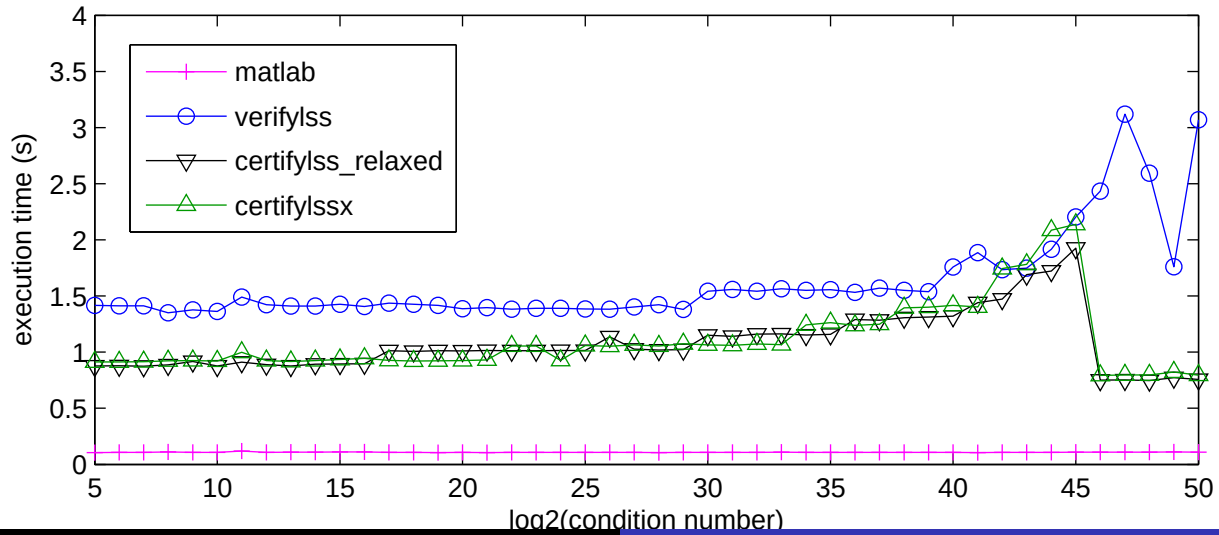
*Output:* $\mathbf{x} = \tilde{x} + \mathbf{e}$

**Implementation:** careful tuning of the precision of each variable (no doubling for $\mathbf{e}$: useless and costly).

# Extra-precise relaxed method: Results

Dimension: 1000 $b = [1, \ldots, 1]^T$



# Agenda

# Recap of the chapter on interval arithmetic

## Outline of the chapter

- historical and panoramic viewpoint:
  including algorithms, complexity result on the difficulty to evaluate a polynomial exactly
- linear algebra:
  building block = matrix multiplication
  discussion of two representations: inf-sup versus mid-rad
  goals = accuracy **and** efficiency
- linear algebra:
  building block = linear system solving
  complexity: NP-hard
  algorithms (Rump, Nguyen) built by assembling basic blocks
  ($\varepsilon$-inflation, Krawczyk, Gauss-Seidel...)
- main use (here) for verification of floating-point computations

# Philosophical conclusion

## Morale

- keep your goals in mind (accuracy, efficiency)
- reuse optimized blocks (BLAS3)
- build algorithms by assembling building blocks
- interval arithmetic can be a tool for verification purposes