SCAN 2008

Sep. 29 - Oct. 3, 2008, at El Paso, TX, USA

Accurate Solution of Triangular Linear System

Philippe Langlois
DALI at University of Perpignan
France

Nicolas Louvet
INRIA and LIP at ENS Lyon
France

philippe.langlois@univ-perp.fr
nicolas.louvet@ens-lyon.fr

Motivation

How to improve and validate the accuracy of a floating point computation, without large computing time overheads ?

- Our main tool to improve the accuracy:

  compensation of the rounding errors.

- Existing results:
  - summation and dot product algorithms from T. Ogita, S. Oishi and S. Rump,
  - Horner algorithm for polynomial evaluation from the authors.

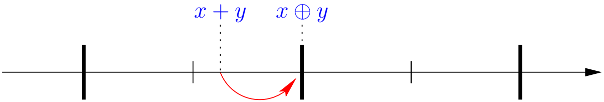
- Today's study: triangular system solving which is one of the basic block for numerical linear algebra.

Outline

Floating point arithmetic

Let $a, b \in \mathbb{F}$ and op $\in \{+, -, \times, /\}$ an arithmetic operation.

- $\mathrm{fl}(x \text{ op } y)$ = the exact x op y rounded to the nearest floating point value.



  Every arithmetic operation may suffer from a rounding error.

- Standard model of floating point arithmetic :

$$\mathrm{fl}(a \text{ op } b) = (1+\varepsilon)(a \text{ op } b), \quad \text{with} \quad |\varepsilon| \le \mathbf{u}.$$

  Working precision $\mathbf{u} = 2^{-p}$ (in rounding to the nearest rounding mode).

- In this talk: IEEE-754 binary fp arithmetic, rounding to the nearest, no underflow nor overflow.

Why and how to improve the accuracy?

- The general "rule of thumb" for backward stable algorithms:

  result accuracy $\lesssim$ condition number of the problem $\times$ computing precision.

Why and how to improve the accuracy?

- The general "rule of thumb" for backward stable algorithms:

  result accuracy $\lesssim$ condition number of the problem $\times$ computing precision.

- Classic solution to improve the accuracy: increase the working precision $\mathbf{u}$.
  - Hardware solution:
    - extended precision available in x87 fpu units.
  - Software solutions:
    - arbitrary precision library (the programmer choses its working precision): MP, MPFUN/ARPREC, MPFR.
    - fixed length expansions libraries: double-double, quad-double (Bailey *et al.*)
      $\hookrightarrow$ XBLAS library = BLAS + double-double (precision $\mathbf{u}^2$)

Why and how to improve the accuracy?

- The general "rule of thumb" for backward stable algorithms:

  result accuracy $\lesssim$ condition number of the problem $\times$ computing precision.

- Classic solution to improve the accuracy: increase the working precision $\mathbf{u}$.
  - Hardware solution:
    - extended precision available in x87 fpu units.
  - Software solutions:
    - arbitrary precision library (the programmer choses its working precision): MP, MPFUN/ARPREC, MPFR.
    - fixed length expansions libraries: double-double, quad-double (Bailey *et al.*)
      $\hookrightarrow$ XBLAS library = BLAS + double-double (precision $\mathbf{u}^2$)

- Alternative solution: compensated algorithms use corrections of the generated rounding errors.

Error-free transformations (EFT)

Error-Free Transformations are algorithms to compute the rounding errors at the current working precision.

Error-free transformations (EFT)

Error-Free Transformations are algorithms to compute the rounding errors at the current working precision.

$+$	$(x, y) = \text{2Sum}(a, b)$ such that $x = a \oplus b$ and $a + b = x + y$	6 flop	Knuth (74)
$\times$	$(x, y) = \text{2Prod}(a, b)$ such that $x = a \otimes b$ and $a \times b = x + y$	17 flop	Dekker (71)
$/$	$(q, r) = \text{DivRem}(a, b)$ such that $q = a \oslash b$, and $a = b \times q + r$.	20 flop	Pichat & Vignes (93)

with $a, b, x, y, q, r \in \mathbb{F}$.

Algorithm (Knuth)

function [x,y] = 2Sum(a,b)

$x = a \oplus b$

$z = x \ominus a$

$y = (a \ominus (x \ominus z)) \oplus (b \ominus z)$

Algorithm (EFT for the division)

function $[q, r] = \text{DivRem}(a, b)$

$q = a \oslash b$

$[x, y] = \text{2Prod}(q, b)$

$r = (a \ominus x) \ominus y$

Substitution algorithm for $Tx = b$

Consider the triangular linear system $Tx = b$, with $T \in \mathbb{F}^{n \times n}$ and vector $b \in \mathbb{F}^n$,

$$\begin{pmatrix} t_{1,1} & & \\ \vdots & \ddots & \\ t_{n,1} & \cdots & t_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}.$$

The substitution algorithm computes $x_1, x_2, \ldots, x_n$, the solution of $Tx = b$, as

$$x_k = \frac{1}{t_{k,k}} \left(b_k - \sum_{i=1}^{k-1} t_{k,i} x_i \right),$$

Algorithm (Substitution)

function $\widehat{x} = \mathrm{TRSV}(T, b)$
  for $k = 1 : n$
    $\widehat{s}_{k,0} = b_k$
    for $i = 1 : k-1$
      $\widehat{p}_{k,i} = t_{k,i} \otimes \widehat{x}_i$
      $\widehat{s}_{k,i} = \widehat{s}_{k,i-1} \ominus \widehat{p}_{k,i}$
    end
    $\widehat{x}_k = \widehat{s}_{k,k-1} \oslash t_{k,k}$
  end

Accuracy of the substitution algorithm

- Skeel's condition number for $Tx = b$ is

$$\mathrm{cond}(T, x) := \frac{\| |T^{-1}||T||x| \|_\infty}{\|x\|_\infty}.$$

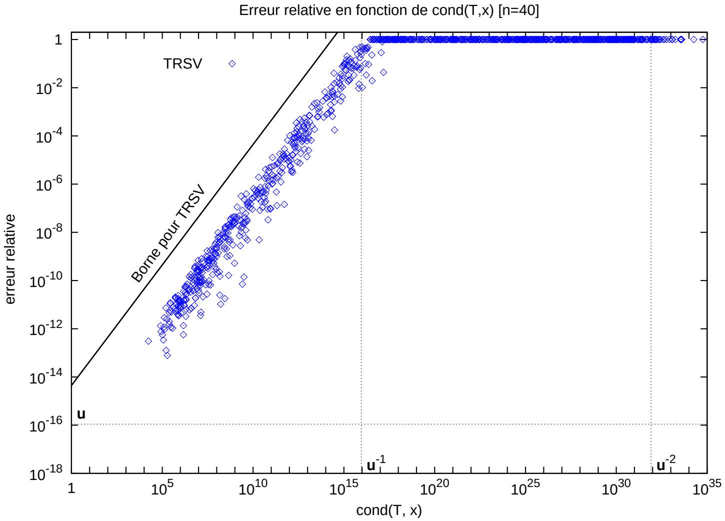
- The accuracy of $\widehat{x} = \mathrm{TRSV}(T, x)$ satisfies

$$\frac{\| \widehat{x} - x \|_\infty}{\| \widehat{x} \|_\infty} \leq n\mathbf{u}\, \mathrm{cond}(T, x) + \mathcal{O}(\mathbf{u}^2),$$

assuming $\mathrm{cond}(T) := \| |T^{-1}||T| \|_\infty \ll \frac{1}{n\mathbf{u}}$.

- The computed $\widehat{x}$ can be arbitrarily less accurate than the w.p. $\mathbf{u}$.

Substitution : accuracy w.r.t. cond(T, x)



How to compensate the rounding errors generated by the substitution algorithm?

Compensated triangular system solving

$\widehat{x} = \mathrm{TRSV}(T, x)$ is the solution to $Tx = b$ computed by substitution.

- Our goal: to compute an approximate $\widehat{c}$ of the correcting term

$$c = x - \widehat{x}, \quad \text{with} \quad c \in \mathbb{R}^{n \times 1},$$

and then compute the compensated solution $\overline{x} = \widehat{x} \oplus \widehat{c}$.

Compensated triangular system solving

$\widehat{x} = \mathrm{TRSV}(T, x)$ is the solution to $Tx = b$ computed by substitution.

- Our goal: to compute an approximate $\widehat{c}$ of the correcting term

$$c = x - \widehat{x}, \quad \text{with} \quad c \in \mathbb{R}^{n\times 1},$$

and then compute the compensated solution $\overline{x} = \widehat{x} \oplus \widehat{c}$.

- Since $Tc = b - T\widehat{x}$, c is the exact solution of the triangular system

$$Tc = r, \quad \text{with} \quad r = b - T\widehat{x} \in \mathbb{R}^{n\times 1}.$$

- In the classic *iterative refinement* frame,
  - r is the residual associated with computed $\widehat{x}$
  - the useful approach to compute an approximate residual $\widehat{r}$: evaluate $b - T\widehat{x}$ using twice the working precision ($\mathbf{u}^2$)

Computing the residual thanks to EFT

function $\widehat{x} = \mathrm{TRSV}(T, b)$
  for $k = 1 : n$
    $\widehat{s}_{k,0} = b_k$
    for $i = 1 : k-1$
      $\widehat{p}_{k,i} = t_{k,i} \otimes \widehat{x}_i$ $\{$rounding error $\pi_{k,i} \in \mathbb{F}$ $\}$
      $\widehat{s}_{k,i} = \widehat{s}_{k,i-1} \ominus \widehat{p}_{k,i}$ $\{$rounding error $\sigma_{k,i} \in \mathbb{F}$ $\}$
    end
    $\widehat{x}_k = \widehat{s}_{k,k-1} \oslash t_{k,k}$ $\{$rounding error $\rho_k / t_{k,k}$, with $\rho_k \in \mathbb{F}$ $\}$
  end

Proposition

Given $\widehat{x} = \mathrm{TRSV}(T, b) \in \mathbb{F}^n$, let $r = (r_1, \dots, r_n)^T \in \mathbb{R}^n$ be the residual $r = b - T\widehat{x}$ associated with $\widehat{x}$. Then we have exactly

$$r_k = \rho_k + \sum_{i=1}^{k-1} \sigma_{k,i} - \pi_{k,i}, \quad \text{for} \quad k = 1 : n.$$

Compensated substitution algorithm

Algorithm

function $\overline{x} = \mathrm{CompTRSV}(T, b)$

for $k = 1 : n$

$\widehat{s}_{k,0} = b_k$; $\widehat{r}_{k,0} = \widehat{c}_{k,0} = 0$

for $i = 1 : k-1$

$[\widehat{p}_{k,i}, \pi_{k,i}] = \mathrm{2Prod}(t_{k,i}, \widehat{x}_i)$

$[\widehat{s}_{k,i}, \sigma_{k,i}] = \mathrm{2Sum}(\widehat{s}_{k,i-1}, -\widehat{p}_{k,i})$

$\widehat{r}_{k,i} = \widehat{r}_{k,i-1} \oplus (\sigma_{k,i} \ominus \pi_{k,i})$

$\widehat{c}_{k,i} = \widehat{c}_{k,i-1} \oplus t_{k,i} \otimes \widehat{c}_i$

end

$[\widehat{x}_k, \rho_k] = \mathrm{DivRem}(\widehat{s}_{k-1}, t_{k,k})$

$\widehat{r}_k = \rho_k \oplus \widehat{r}_{k,k-1}$

$\widehat{c}_k = (\widehat{r}_k \ominus \widehat{c}_{k,k-1}) \oslash t_{k,k}$

end

$\overline{x} = \widehat{x} \oplus \widehat{c}$

This algorithm computes

- the approximate solution $\widehat{x} = (\widehat{x}_1, \ldots, \widehat{x}_n)^T = \mathrm{TRSV}(T, b)$ by the substitution algorithm;
- the rounding errors $\pi_{k,i}$, $\sigma_{k,i}$ and ρ_k;
- the residual vector $\widehat{r} = (\widehat{r}_1, \ldots, \widehat{r}_n)^T \approx b - T\widehat{x}$ as a function of $\pi_{k,i}$, $\sigma_{k,i}$ and ρ_k;
- the correcting term $Tc = b$ $\widehat{c} = (\widehat{c}_1, \ldots, \widehat{c}_n)^T = \mathrm{TRSV}(T, \widehat{r})$;
- the compensated solution $\overline{x} = \widehat{x} \oplus \widehat{c}$.

An *a priori* error bound for CompTRSV

The approximate residual vector $\widehat{r}$ computed in CompTRSV is as accurate as if it was computed in doubled working precision ($\mathbf{u}^2$).

Theorem

The accuracy of the compensated solution $\widehat{x} = \text{CompTRSV}(T, b)$ *satisfies*

$$\frac{\|\overline{x} - x\|_\infty}{\|x\|_\infty} \lesssim \mathbf{u} + f_n \mathbf{u}^2 \mathrm{K}(T, x) + \mathcal{O}(\mathbf{u}^3).$$

This error bound is more pessimistic than the "expected" error bound

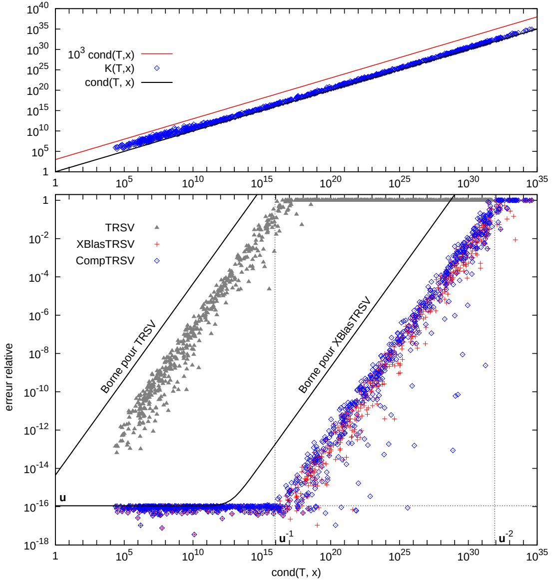
$$\mathbf{u} + g_n \mathbf{u}^2 \operatorname{cond}(T, x) + \mathcal{O}(\mathbf{u}^3),$$

since

$$\frac{\| |T^{-1}||T||x| \|_\infty}{\|x\|_\infty} =: \operatorname{cond}(T, x) \leq \mathrm{K}(T, x) := \frac{\| (|T^{-1}||T|)^2 |x| \|_\infty}{\|x\|_\infty}.$$

But we often observe $\mathrm{K}(T, x) \leq \alpha \operatorname{cond}(T, x)$ with α "small" in practice...

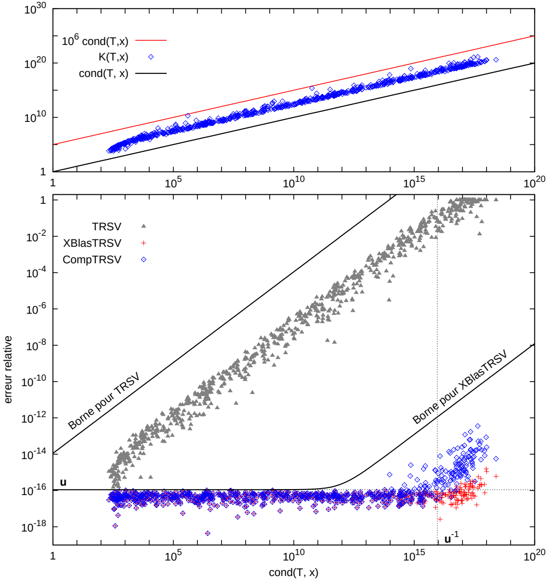
Practical accuracy of CompTRSV w.r.t. cond(T, x)



n=40,
systems generated
by "method 1":
$K(T,x) \leq 10^3 \, \mathrm{cond}(T,x)$

In these experiments,
CompTRSV is as accurate
as XBlasTRSV

Practical accuracy of CompTRSV w.r.t. cond(T, x)



n=100,

systems generated by "method 2":

$K(T, x) \leq 10^6 \, \mathrm{cond}(T, x)$

The accuracy of CompTRSV is not as good as expected...

From CompTRSV to CompTRSV2

Algorithm

function $\overline{x} = \text{CompTRSV}(T, b)$

for $k = 1 : n$

$\widehat{s}_{k,0} = b_k;\ \widehat{r}_{k,0} = \widehat{c}_{k,0} = 0$

for $i = 1 : k-1$

$[\widehat{p}_{k,i}, \pi_{k,i}] = \text{2Prod}(t_{k,i}, \widehat{x}_i)$

$[\widehat{s}_{k,i}, \sigma_{k,i}] = \text{2Sum}(\widehat{s}_{k,i-1}, -\widehat{p}_{k,i})$

$\widehat{r}_{k,i} = \widehat{r}_{k,i-1} \oplus (\sigma_{k,i} \ominus \pi_{k,i})$

$\widehat{c}_{k,i} = \widehat{c}_{k,i-1} \oplus t_{k,i} \otimes \widehat{c}_i$

end

$[\widehat{x}_k, \rho_k] = \text{DivRem}(\widehat{s}_{k-1}, t_{k,k})$

$\widehat{r}_k = \rho_k \oplus \widehat{r}_{k,k-1}$

$\widehat{c}_k = (\widehat{r}_k \ominus \widehat{c}_{k,k-1}) \oslash t_{k,k}$

end

$\overline{x} = \widehat{x} \oplus \widehat{c}$

Iteration k produces $\widehat{x}_k$ and $\widehat{c}_k$ as a function of

- $\widehat{x}_1, \ldots, \widehat{x}_{k-1}$
- $\widehat{c}_1, \ldots, \widehat{c}_{k-1}$

with $\widehat{c}_k \approx x_k - \widehat{x}_k$

Vector compensation:

corrected $\overline{x}$ is computed only after whole $\widehat{x}$ and $\widehat{c}$ have been computed

Issue: component compensation, *i.e.* , compute $\overline{x}_k$ at iteration k

From CompTRSV to CompTRSV2

Algorithm

function $\overline{x} = \text{CompTRSV2}(T, b)$

for $k = 1 : n$

$\widehat{s}_{k,0} = b_k$; $\widehat{r}_{k,0} = \widehat{c}_{k,0} = 0$

for $i = 1 : k-1$

$[\widehat{p}_{k,i}, \pi_{k,i}] = \text{2Prod}(t_{k,i}, \overline{x}_i)$

$[\widehat{s}_{k,i}, \sigma_{k,i}] = \text{2Sum}(\widehat{s}_{k,i-1}, -\widehat{p}_{k,i})$

$\widehat{r}_{k,i} = \widehat{r}_{k,i-1} \oplus (\sigma_{k,i} \ominus \pi_{k,i})$

$\widehat{c}_{k,i} = \widehat{c}_{k,i-1} \oplus t_{k,i} \otimes \overline{y}_i$

end

$[\widehat{x}_k, \rho_k] = \text{DivRem}(\widehat{s}_{k-1}, t_{k,k})$

$\widehat{r}_k = \rho_k \oplus \widehat{r}_{k,k-1}$

$\widehat{c}_k = (\widehat{r}_k \ominus \widehat{c}_{k,k-1}) \oslash t_{k,k}$

$[\overline{x}_k, \overline{y}_k] = \text{2Sum}(\widehat{x}_k, \widehat{c}_k)$

end

Iteration k produces

- $\widehat{x}_k$ w.r.t. $\overline{x}_1, \dots, \overline{x}_{k-1}$
- $\widehat{c}_k$ s.t. $\widehat{c}_k \approx x_k - \widehat{x}_k$

Since $[\overline{x}_k, \overline{y}_k] = \text{2Sum}(\widehat{x}_k, \widehat{c}_k)$,

- $\overline{x}_k = \widehat{x}_k \oplus \widehat{c}_k$
- $\overline{x}_k + \overline{y}_k = \widehat{x}_k + \widehat{c}_k$

and $\overline{y}_k \approx x_k - \overline{x}_k$

Iteration k computes $\overline{x}_k$ and $\overline{y}_k$ w.r.t.

- $\overline{x}_1, \dots, \overline{x}_{k-1}$
- $\overline{y}_1, \dots, \overline{y}_{k-1}$

with $\overline{y}_k \approx x_k - \overline{x}_k$

Accuracy of CompTRSV2

In algorithm CompTRSV,

$$\overline{x} = \begin{pmatrix} \overline{x}_1 \\ \vdots \\ \overline{x}_n \end{pmatrix} \in \mathbb{F}^n, \quad \overline{y} = \begin{pmatrix} \overline{y}_1 \\ \vdots \\ \overline{y}_n \end{pmatrix} \in \mathbb{F}^n \quad \text{and} \quad \overline{x}+\overline{y} = \begin{pmatrix} \overline{x}_1 + \overline{y}_1 \\ \vdots \\ \overline{x}_n + \overline{y}_n \end{pmatrix} \in \mathbb{R}^n.$$

The vector $\overline{x} + \overline{y}$ is an approximate solution of the system $Tx = b$,
and the exact solution of a slightly perturbed system,

$$(T + \Delta T)(\overline{x} + \overline{y}) = (b + \Delta b), \quad \text{with} \quad |\Delta T| \leq \gamma_{6n}^2 |T| \quad \text{and} \quad |\Delta b| \leq \gamma_{6n}^2 |b|,$$

where $\gamma_{6n} \approx 6n\mathbf{u}$.

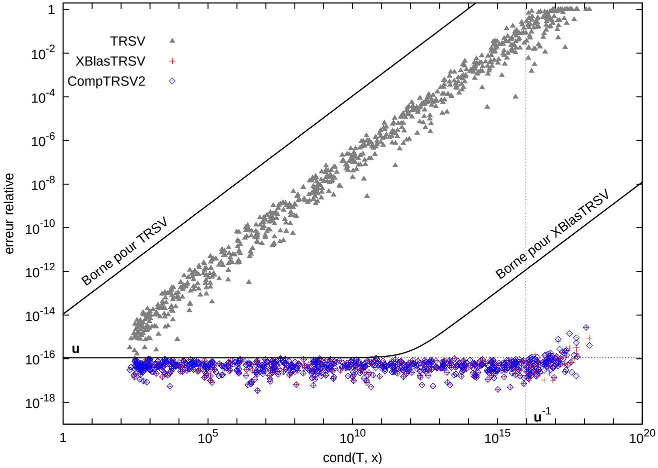
Theorem

The accuracy of the compensated solution $\overline{x} = \mathrm{CompTRSV2}(T, b)$ *satisfies*

$$\frac{\|\overline{x} - x\|_\infty}{\|x\|_\infty} \lesssim \mathbf{u} + 72n^2\mathbf{u}^2 \,\mathrm{cond}(T, x) + \mathcal{O}(\mathbf{u}^3).$$

Practical accuracy of CompTRSV2 w.r.t. cond(T, x)

n=100, systems generated by “method 2”



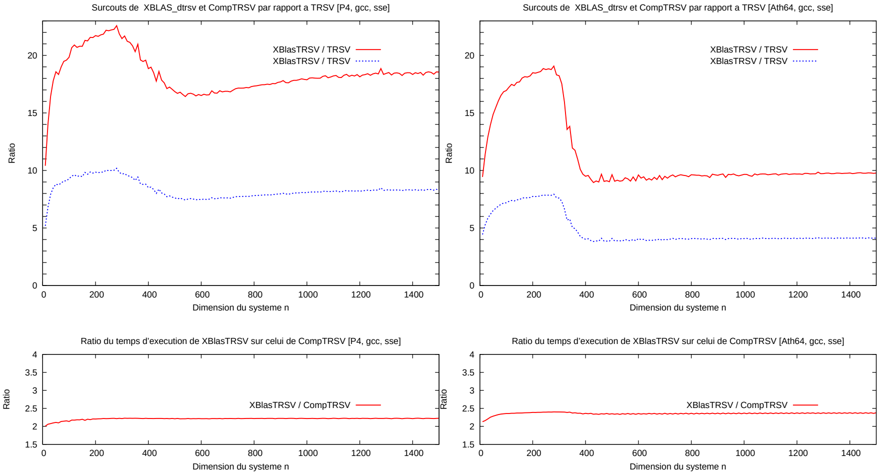
CompTRSV2 is as accurate as

the classic substitution algorithm performed in twice the working precision ($\mathbf{u}^2$).

Running time comparisons

Top: overhead ratios to double the accuracy while increasing dimension n

Down: CompTRSV and CompTRSV2 run at least twice as fast as XBlasTRSV



Conclusion

- We have presented a compensated substitution algorithm, CompTRSV:
  - *a priori* error bound,
  - and practical numerical behavior

  not entirely satisfying...

- We have also presented an improvement of this method, CompTRSV2: the solution computed by CompTRSV2 is as accurate as if it was computed by the substitution algorithm in twice the working precision ($\mathbf{u}^2$).

- CompTRSV and CompTRSV2 runs at least twice as fast as XBlasTRSV.

Componentwise condition numbers for linear systems

Let $Ax = b$ be a linear system, with $A \in \mathbb{R}^{n\times n}$ non singular and $b \in \mathbb{R}^{n\times 1}$. Given $E \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^{n\times 1}$, with $|E| \geq 0$ and $|f| \geq 0$, Higham defines the following componentwise condition number,

$$\mathrm{cond}_{E,f}(A,x) = \lim_{\varepsilon\to 0} \sup \Big\{ \frac{\|\Delta x\|_\infty}{\varepsilon \|x\|_\infty}, (A+\Delta A)(x+\Delta x) = b + \Delta b, \\ |\Delta A| \leq \varepsilon E, |\Delta b| \leq \varepsilon f \Big\},$$

and proves that

$$\mathrm{cond}_{E,f}(A,x) = \frac{\| |A^{-1}|(E|x| + f)\|_\infty}{\|x\|_\infty}.$$

For the special case $E = |A|$ and $f = |b|$, we use Skeel's condition number,

$$\mathrm{cond}(A,x) = \frac{\| |A^{-1}||A||x| \|_\infty}{\|x\|_\infty},$$

which differs from $\mathrm{cond}_{|A|,|b|}(A,x)$ by at most a factor 2.

Floating-point operations counts

- TRSV: n^2
- CompTRSV and CompTRSV2: $27n^2/2 + O(n)$
- XBlasTRSV: $45n^2/2 + O(n)$