Parallel robust solution of triangular equations

Carl Christian Kjelgaard Mikkelsen
spock@cs.umu.se

Mirko Myllykoski, Angelika Schwarz, Lars Karlsson

Department of Computing Science
Umeå University

SIAM CSE
Virtual conference
March 1-5, 2021

UMEÅ UNIVERSITY

What is a robust algorithm?

All numbers are in the representable range

Example 1: $\boldsymbol{Ax} = \boldsymbol{b}$

$$\boldsymbol{Ax} := \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & -1 & 1 & & \\ -1 & -1 & -1 & 1 & \\ -1 & -1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 4 \\ 8 \\ 16 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} =: \boldsymbol{b}$$

Example 2: $\boldsymbol{SX} = \boldsymbol{X}\Lambda$

Matrix:

$$\boldsymbol{S} = \begin{bmatrix} 1 & & & & \\ -5 & 2 & & & \\ -5 & -5 & 3 & & \\ -5 & -5 & -5 & 4 & \\ -5 & -5 & -5 & -5 & 5 \end{bmatrix}$$

Eigenvectors:

$$\boldsymbol{X} = \begin{bmatrix} 1 & & & & \\ 5 & 1 & & & \\ 15 & 5 & 1 & & \\ 35 & 15 & 5 & 1 & \\ 70 & 35 & 15 & 5 & 1 \end{bmatrix}$$

Example 3: $\boldsymbol{AX} + \boldsymbol{XA}^T = \boldsymbol{bb}^T$

Matrices:

$$\boldsymbol{A} = \begin{bmatrix} \frac{1}{2} & & & & \\ -1 & \frac{1}{2} & & & \\ -1 & -1 & \frac{1}{2} & & \\ -1 & -1 & -1 & \frac{1}{2} & \\ -1 & -1 & -1 & -1 & \frac{1}{2} \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Solution:

$$\boldsymbol{X} = \begin{bmatrix} 1 & 2 & 4 & 8 & 16 \\ 2 & 5 & 12 & 28 & 64 \\ 4 & 12 & 33 & 86 & 216 \\ 8 & 28 & 86 & 245 & 664 \\ 16 & 64 & 216 & 664 & 1921 \end{bmatrix}$$

Robust triangular solvers in LAPACK

Subroutine	Problem
`xlatrs`	$\boldsymbol{A}\boldsymbol{x} = \alpha\boldsymbol{b}$
`xtrevc`	$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{X}\boldsymbol{\Lambda}$
`xtgevc`	$\boldsymbol{A}\boldsymbol{X} = \boldsymbol{B}\boldsymbol{X}\boldsymbol{\Lambda}$
`xtrsyl`	$\boldsymbol{A}\boldsymbol{X} + \boldsymbol{X}\boldsymbol{B} = \alpha\boldsymbol{C}$

All routines are scalar, sequential codes descended from `xlatrs`.

Robust triangular solvers in ScaLAPACK?

From PDLATTRS[1] (parallel dlatrs)

```
***** NO SCALING ***** Call PDTRSV for all cases
*****
```

From PZLATTRS[2] (parallel zlatrs)

```
This is very slow relative to PZTRSV. This should
only be used when scaling is necessary to control
overflow, or when it is modified to scale better.
```

[1] http://www.netlib.org/scalapack/explore-html/d6/da4/pdlatrs_8f_source.html, line 62. Retrieved March 3rd 2021

[2] http://www.netlib.org/scalapack/explore-html/df/d88/pzlattrs_8f_source.html, lines 36:38. Retrieved: March 3rd 2021

Regular backward substitution, $\boldsymbol{T}\boldsymbol{x} = \boldsymbol{b}$

Algorithm 1: $[\alpha, \boldsymbol{x}] = \texttt{BackSub}(\boldsymbol{T}, \boldsymbol{b})$

```
for j ← n, n − 1, . . . , 1 do
    x_j ← x_j / t_jj;
    if j > 1 then
        x_{1:j−1} ← x_{1:j−1} − T_{1:j−1,j} x_j;
return x;
```

Robust backward substitution, $\boldsymbol{T}\boldsymbol{x} = \alpha \boldsymbol{b}$

Algorithm 2: $[\alpha, \boldsymbol{x}] = \texttt{RobustBackSub}(\boldsymbol{T}, \boldsymbol{b})$

$\alpha \leftarrow 1$;
for $j \leftarrow n, n-1, \ldots, 1$ **do**
  $\beta = \texttt{ProtectDivision}(x_j, t_{jj})$;
  if $\beta \neq 1$ **then**
    $\boldsymbol{x} \leftarrow \beta \boldsymbol{x}, \alpha \leftarrow \beta \alpha$;
  $x_j \leftarrow x_j / t_{jj}$;
  if $j > 1$ **then**
    $\beta = \texttt{ProtectUpdate}(x_{\max}, \|\boldsymbol{T}_{1:j-1,j}\|_\infty, |x_j|)$;
    if $\beta \neq 1$ **then**
      $\boldsymbol{x} \leftarrow \beta \boldsymbol{x}, \alpha \leftarrow \beta \alpha$;
    $\boldsymbol{x}_{1:j-1} \leftarrow \boldsymbol{x}_{1:j-1} - \boldsymbol{T}_{1:j-1,j} x_j$;
    $x_{\max} \leftarrow \|\boldsymbol{x}_{1:j-1}\|_\infty$;
return $[\alpha, \boldsymbol{x}]$;

ProtectDivision

The division $y \leftarrow b/t$ is safe if

$$|b| \leq |t|\Omega, \quad \Omega = \text{largest normal FP number.} \tag{1}$$

ProtectDivision

The division $y \leftarrow b/t$ is safe if

$$|b| \leq |t|\Omega, \quad \Omega = \text{largest normal FP number.} \tag{1}$$

Problem: Find scaling $\alpha \in (0, 1]$ such that

$$|\alpha b| \leq |t|\Omega. \tag{2}$$

ProtectDivision

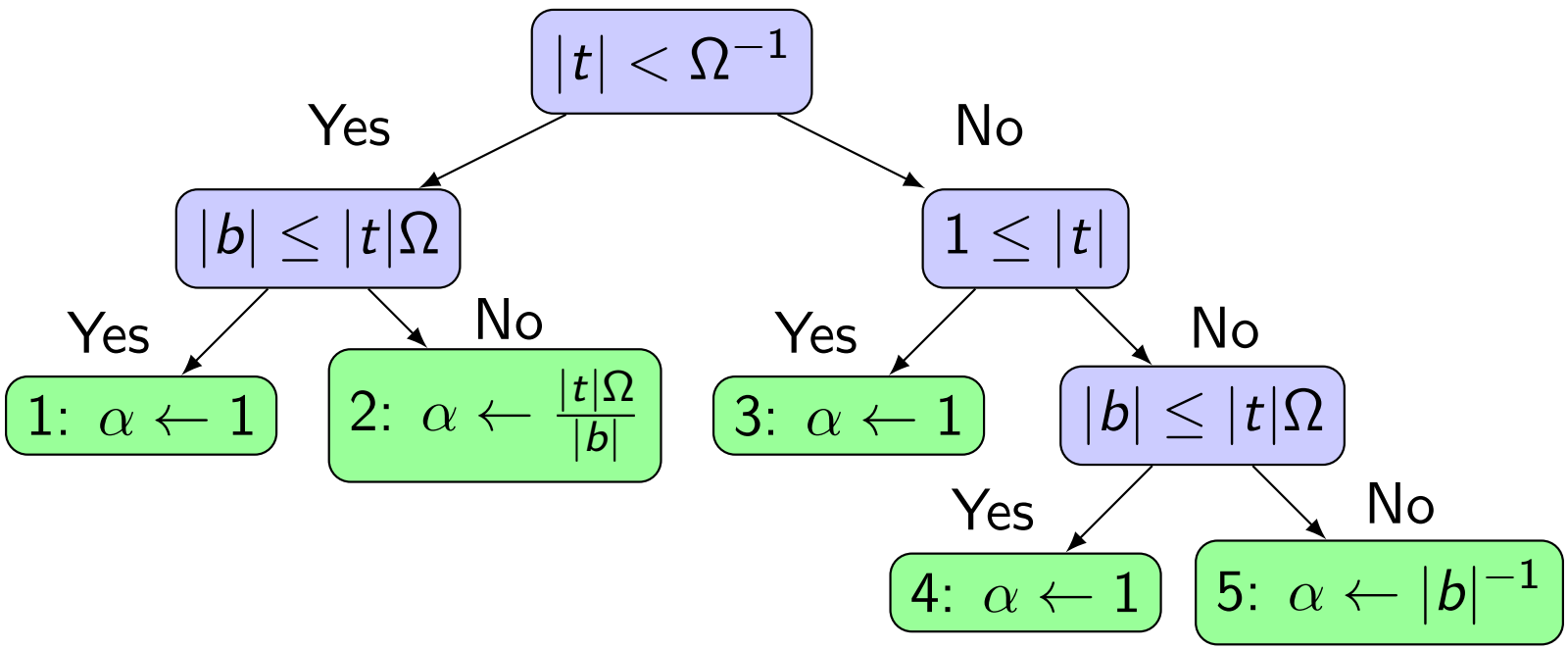
The division $y \leftarrow b/t$ is safe if

$$|b| \leq |t|\Omega, \quad \Omega = \text{largest normal FP number.} \tag{1}$$

Problem: Find scaling $\alpha \in (0, 1]$ such that

$$|\alpha b| \leq |t|\Omega. \tag{2}$$

Solution:



ProtectUpdate

The linear update $\boldsymbol{y} \leftarrow \boldsymbol{y} - \boldsymbol{T}\boldsymbol{x}$ is safe if

$$\|\boldsymbol{y}\|_\infty + \|\boldsymbol{T}\|_\infty \|\boldsymbol{x}\|_\infty \leq \Omega, \quad \Omega = \text{largest normal FP number.} \quad (3)$$

ProtectUpdate

The linear update $\boldsymbol{y} \leftarrow \boldsymbol{y} - \boldsymbol{T}\boldsymbol{x}$ is safe if

$$\|\boldsymbol{y}\|_\infty + \|\boldsymbol{T}\|_\infty \|\boldsymbol{x}\|_\infty \leq \Omega, \quad \Omega = \text{largest normal FP number.} \tag{3}$$

Problem: Find $\zeta \in (0, 1]$ such that

$$\|\zeta\boldsymbol{y}\|_\infty + \|\boldsymbol{T}\|_\infty \|\zeta\boldsymbol{x}\|_\infty \leq \Omega. \tag{4}$$

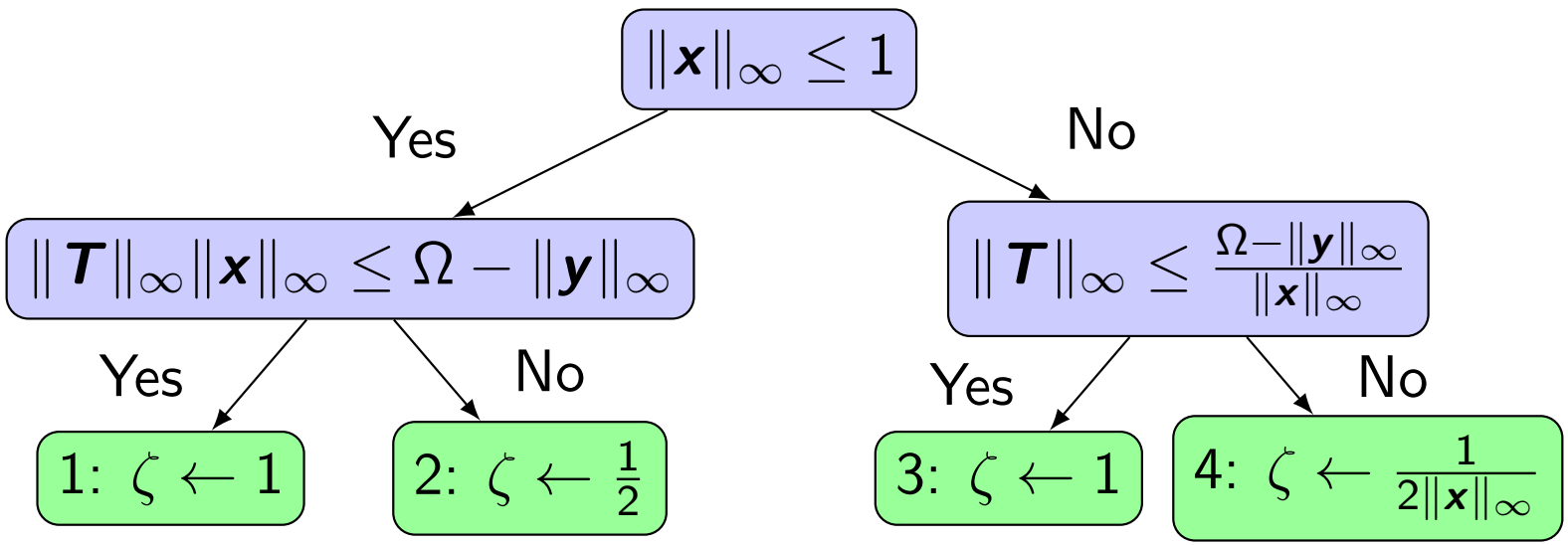
ProtectUpdate

The linear update $\boldsymbol{y} \leftarrow \boldsymbol{y} - \boldsymbol{T}\boldsymbol{x}$ is safe if

$$\|\boldsymbol{y}\|_\infty + \|\boldsymbol{T}\|_\infty \|\boldsymbol{x}\|_\infty \leq \Omega, \quad \Omega = \text{largest normal FP number.} \tag{3}$$

Problem: Find $\zeta \in (0, 1]$ such that

$$\|\zeta\boldsymbol{y}\|_\infty + \|\boldsymbol{T}\|_\infty \|\zeta\boldsymbol{x}\|_\infty \leq \Omega. \tag{4}$$



How to develop blocked, parallel and robust codes?

How to develop blocked, parallel and robust codes?

Goal: No significant difference between robust and non-robust codes!

How to develop blocked, parallel and robust codes?

Goal: No significant difference between robust and non-robust codes!

Key: Different processing units can use different units of measurement.

Step 1: Partition the problem

$$\boldsymbol{TX} := \begin{bmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} & \dots & \boldsymbol{T}_{1M} \\ & \boldsymbol{T}_{22} & \dots & \boldsymbol{T}_{2m} \\ & & \ddots & \vdots \\ & & & \boldsymbol{T}_{MM} \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_{11} & \boldsymbol{X}_{12} & \dots & \boldsymbol{X}_{1N} \\ \boldsymbol{X}_{21} & \ddots & \ddots & \boldsymbol{X}_{2N} \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{X}_{M1} & \dots & \dots & \boldsymbol{X}_{MN} \end{bmatrix}$$

$$= \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} & \dots & \boldsymbol{B}_{1N} \\ \boldsymbol{B}_{21} & \ddots & \ddots & \boldsymbol{B}_{2N} \\ \vdots & \ddots & \ddots & \vdots \\ \boldsymbol{B}_{M1} & \dots & \dots & \boldsymbol{B}_{MN} \end{bmatrix} =: \boldsymbol{B}$$

Step 2: Augment each block of data

A separate scaling factor α_{ijk} for each column of every block $\boldsymbol{X}_{ij}$.

Step 2: Augment each block of data

A separate scaling factor α_{ijk} for each column of every block $\boldsymbol{X}_{ij}$.

An augmented block

$$\langle \boldsymbol{\alpha}, \boldsymbol{X} \rangle$$

where

$$\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n), \quad \alpha_j \in (0, 1], \quad \boldsymbol{X} \in \mathbb{R}^{m \times n}$$

is used to represents the real matrix

$$\bar{\boldsymbol{X}} \in \mathbb{R}^{m \times n}$$

where

$$\bar{\boldsymbol{x}}_j = \alpha_j^{-1} \boldsymbol{x}_j, \quad j = 1, 2, \ldots, n.$$

Step 3: Replace kernels with robust kernels ...

Step 3: Replace kernels with robust kernels ...

Problem: Given $\langle \boldsymbol{\alpha}, \boldsymbol{X} \rangle$ and $\langle \boldsymbol{\beta}, \boldsymbol{Y} \rangle$ find $\langle \boldsymbol{\nu}, \boldsymbol{Z} \rangle$ such that

$$\bar{\boldsymbol{Z}} = \bar{\boldsymbol{Y}} - \boldsymbol{T}\bar{\boldsymbol{X}}$$

or equivalently

$$\nu_j^{-1}\boldsymbol{z}_j = \beta_j^{-1}\boldsymbol{y}_j - \boldsymbol{T}\left(\alpha_j^{-1}\boldsymbol{x}_j\right), \quad j = 1, 2, \ldots, n$$

Step 3: Replace kernels with robust kernels ...

Problem: Given $\langle \boldsymbol{\alpha}, \boldsymbol{X} \rangle$ and $\langle \boldsymbol{\beta}, \boldsymbol{Y} \rangle$ find $\langle \boldsymbol{\nu}, \boldsymbol{Z} \rangle$ such that

$$\bar{\boldsymbol{Z}} = \bar{\boldsymbol{Y}} - \boldsymbol{T}\bar{\boldsymbol{X}}$$

or equivalently

$$\nu_j^{-1}\boldsymbol{z}_j = \beta_j^{-1}\boldsymbol{y}_j - \boldsymbol{T}\left(\alpha_j^{-1}\boldsymbol{x}_j\right), \quad j = 1, 2, \ldots, n$$

Algorithm 3: $\langle \boldsymbol{\nu}, \mathsf{Z} \rangle \leftarrow$ `RobustGEMM`$(\langle \boldsymbol{\beta}, \mathsf{Y} \rangle, \mathsf{T}, \langle \boldsymbol{\alpha}, \mathsf{X} \rangle)$

---

1 **for** $j \leftarrow 1, 2, \ldots, n$ **do**
2 $\quad \gamma_j \leftarrow \min\{\alpha_j, \beta_j\}$;
3 $\quad \zeta_j \leftarrow$ `ProtectUpdate`$((\gamma_j/\beta_j)\|\mathsf{y}_j\|_\infty, \|\mathsf{T}\|_\infty, (\gamma_j/\alpha_j)\|\mathsf{x}_j\|_\infty)$;
4 $\quad \nu_j \leftarrow \zeta_j\gamma_j$;
5 $\mathsf{Z} \leftarrow \left[\mathsf{Y}\mathrm{diag}(\boldsymbol{\nu} \oslash \boldsymbol{\beta})\right] - \mathsf{T}\left[\mathsf{X}\mathrm{diag}(\boldsymbol{\nu} \oslash \boldsymbol{\alpha})\right]$;
6 **Return** $\langle \boldsymbol{\nu}, \mathsf{Z} \rangle$;

Cost: $O(n^3)$. Overhead: $O(n^2) + O(n^2)$

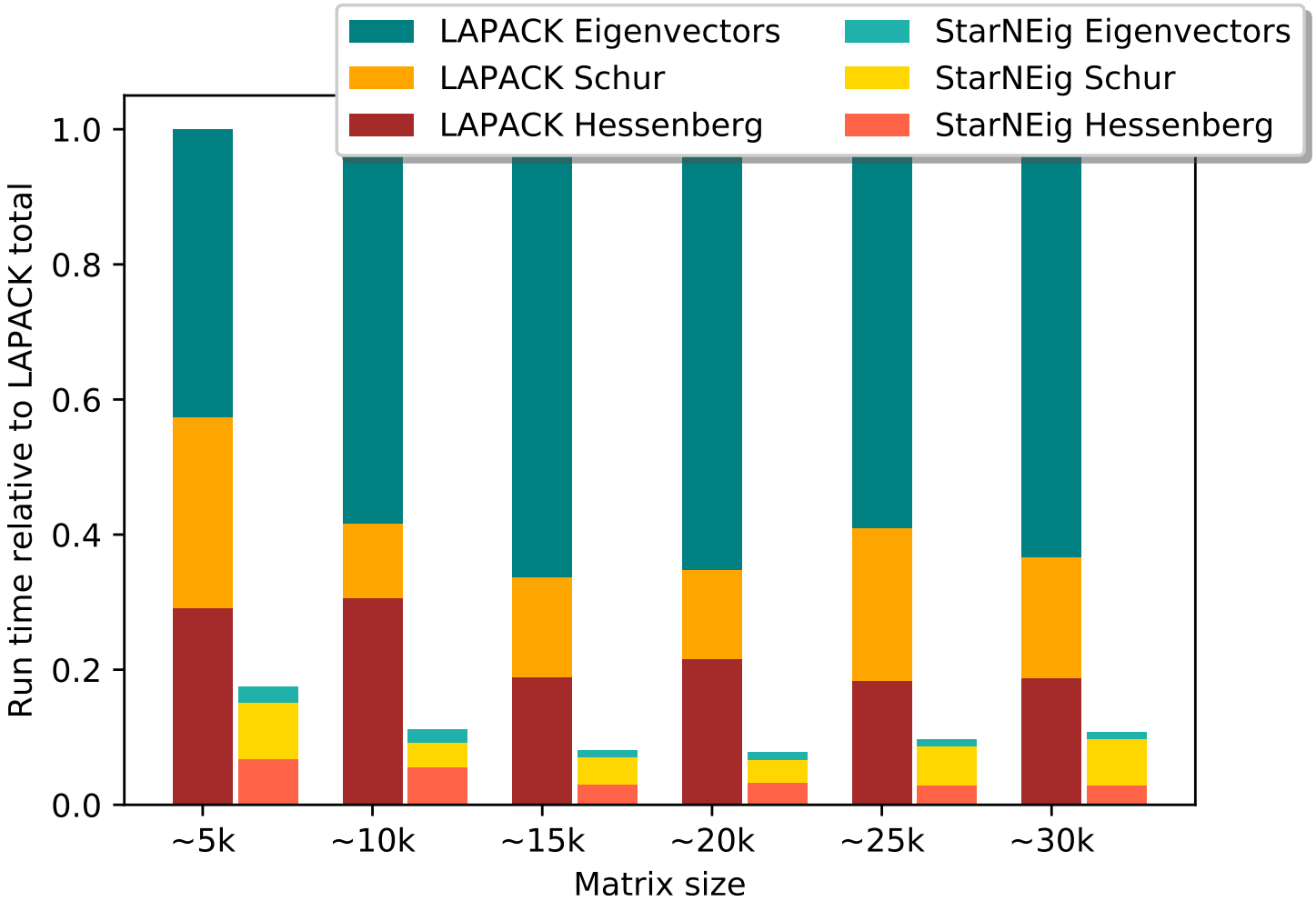
Robust solution of partitioned system $TX = B$.

Robust solution of partitioned system $\boldsymbol{TX} = \boldsymbol{B}$.

Algorithm 4: $\langle \alpha, \mathsf{X} \rangle \leftarrow$ `RobustBlockedSolveMRHS`(T, B)

1 **for** $(i, j) \in \{1, 2, \ldots, M\} \times \{1, 2, \ldots, N\}$ **do**
2 $\langle \boldsymbol{\alpha}_{ij}, \mathsf{X}_{ij} \rangle \leftarrow \langle \mathsf{e}, \mathsf{B}_{ij} \rangle$;
3 **for** $j \leftarrow M, M-1, \ldots, 1$ **do**
4 **for** $k \leftarrow 1, 2, \ldots, N$ **do**
5 $\langle \boldsymbol{\alpha}_{jk}, \mathsf{X}_{jk} \rangle \leftarrow$ `RobustTRSM`$(\mathsf{T}_{jj}, \langle \boldsymbol{\alpha}_{jk}, \mathsf{X}_{jk} \rangle)$;
6 **for** $i \leftarrow 1, 2, \ldots, j-1$ **do**
7 **for** $k \leftarrow 1, 2, \ldots, N$ **do**
8 $\langle \boldsymbol{\alpha}_{ik}, \mathsf{X}_{ik} \rangle \leftarrow$ `RobustGEMM`$(\langle \boldsymbol{\alpha}_{ik}, \mathsf{X}_{ik} \rangle, \mathsf{T}_{ij}, \langle \boldsymbol{\alpha}_{jk}, \mathsf{X}_{jk} \rangle)$;
9 **for** $k \leftarrow 1, 2, \ldots, N$ **do**
10 $\boldsymbol{\alpha}_k \leftarrow \min_i \{\boldsymbol{\alpha}_{ik}\}$;
11 **for** $i \leftarrow 1, 2, \ldots, M$ **do**
12 $\mathsf{X}_{ik} \leftarrow \mathsf{X}_{ik} \mathrm{diag}(\boldsymbol{\alpha}_k \oslash \boldsymbol{\alpha}_{ik})$;
13 $\boldsymbol{\alpha} \leftarrow \begin{bmatrix} \boldsymbol{\alpha}_1 & \boldsymbol{\alpha}_2 & \cdots & \boldsymbol{\alpha}_N \end{bmatrix}$;
14 **Return** $\langle \boldsymbol{\alpha}, \mathsf{X} \rangle$;

LAPACK versus StarNEig



New Software

StarNEig

1. MS227 talk by Dr. Mirko Myllykoski.
2. A library for solving nonsymmetric eigenvalue problems:

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}$$
$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{B}\boldsymbol{x}$$

3. Uses the StarPU runtime system from INRIA.
4. Download

`https://nlafet.github.io/StarNEig/`

Contact: `mirkom@cs.umu.se`, `spock@cs.umu.se`

Backup slides

Caution: Underflow *can* still destroy the result!

$\Omega =$ largest normal number, $\quad \epsilon =$ smallest subnormal number

$$\boldsymbol{Tx} := \left[\begin{array}{c|ccccc} \frac{1}{2} & & & & & \\ \hline & 1 & & & & \\ & -2 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & & -2 & 1 \end{array}\right] \left[\begin{array}{c} 2\Omega \\ \hline \epsilon \\ 2^1\epsilon \\ \vdots \\ 2^{m-1}\epsilon \end{array}\right] = \left[\begin{array}{c} \Omega \\ \hline \epsilon \\ 0 \\ \vdots \\ 0 \end{array}\right] =: \boldsymbol{b}.$$

Scaling flushes ϵ to zero and destroys the solution

Performance degradation of robust solvers?

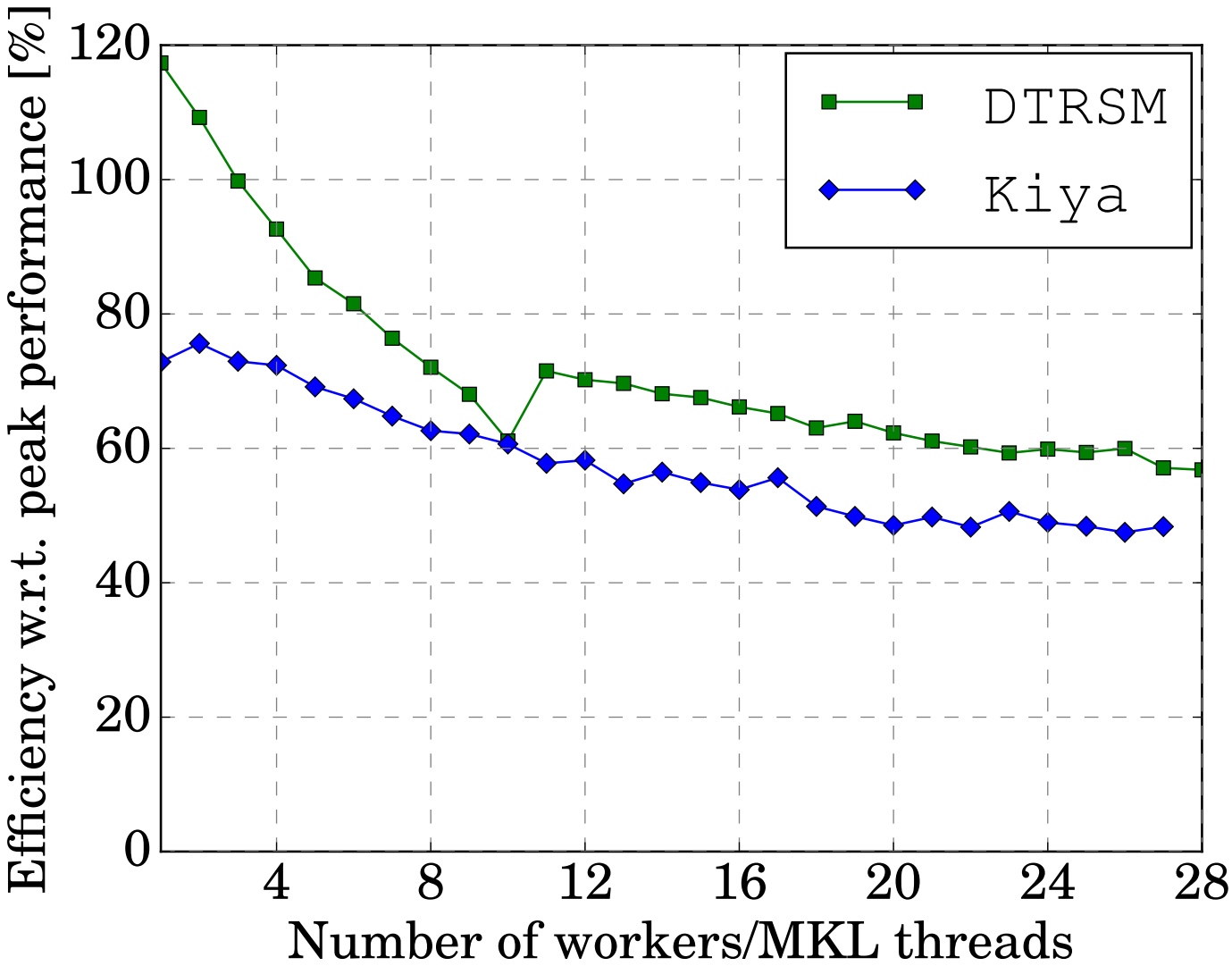


Figure: Triangular linear system $m = 40000$ and $n = 2000$

Bibliography I

[1] Carl Christian Kjelgaard Mikkelsen, Angelika Beatrix Schwarz, and Lars Karlsson.
Parallel robust solution of triangular linear systems.
Concurrency and Computation: Practice and Experience, 31(19):e5064, 2019.
e5064 cpe.5064.

[2] Carl Christian Kjelgaard Mikkelsen.
Well-conditioned eigenvalue problems that overflow.
`https://arxiv.org/abs/2005.05356`, 2020.

[3] Carl Christian Kjelgaard Mikkelsen and Mirko Myllykoski.
Parallel robust computation of generalized eigenvectors of matrix pencils.
In Roman et. al. Wyrzykowski, editor, *PPAM*, number 12043 in LNCS, pages 58–69, Cham, 2020. Springer.

Bibliography II

[4] Mirko Myllykoski and Carl Christian Kjelgaard Mikkelsen.
Introduction to StarNEig—A Task-Based Library for Solving Nonsymmetric Eigenvalue Problems.
In Roman et. al Wyrzykowski, editor, *PPAM*, number 12043 in LNCS, pages 70–81, Cham, 2020. Springer.

[5] Mirko Myllykoski and Carl Christian Kjelgaard Mikkelsen.
Task-based, GPU-accelerated and robust library for solving dense nonsymmetric eigenvalue problems.
Concurrency and Computation: Practice and Experience, page e5915, 2020.

[6] Angelika Schwarz and Carl Christian Kjelgaard Mikkelsen.
Robust Task-Parallel Solution of the Triangular Sylvester Equation.
In Roman et. al. Wyrzykowski, editor, *PPAM*, number 12043 in LNCS, pages 82–92, Cham, 2020. Springer.

Bibliography III

[7] Angelika Schwarz, Carl Christian Kjelgaard Mikkelsen, and Lars Karlsson.
Robust parallel eigenvector computation for the non-symmetric eigenvalue problem.
Parallel Computing, 100:102707, 2020.