Code Generation in Linnea

Henrik Barthels, Paolo Bientinesi

HPAC High Performance and Automatic Computing
RWTH AACHEN UNIVERSITY

Introduction

- How to compute the following expressions?

$$
\begin{aligned}
b &:= (X^T M^{-1} X)^{-1} X^T M^{-1} y \\
x &:= W(A^T (AWA^T)^{-1} b - c) \\
x &:= \left(A^{-T} B^T B A^{-1} + R^T [\Lambda(Rz)] R\right)^{-1} A^{-T} B^T B A^{-1} y \\
X_{k+1} &:= S(S^T A S)^{-1} S^T + (I_n - S(S^T A S)^{-1} S^T A) X_k (I_n - A S (S^T A S)^{-1} S^T)
\end{aligned}
$$

- High-level languages are easy to use, but performance is usually suboptimal.
- BLAS and LAPACK can be fast, but require a lot of expertise.
- Goal: Simplicity **and** performance.

Introduction

How to compute...

$$y' := H^\dagger y + (I_n - H^\dagger H)x$$ [TG17]

...with these operations?

$x := Ab$	$2n^2$
$X := AB$	$2n^3$
$x := a \pm b$	n
$X := A \pm B$	n^2

Introduction

How to compute...

$$y' := H^\dagger y + (I_n - H^\dagger H)x$$ [TG17]

...with these operations?

$x := Ab$	$2n^2$
$X := AB$	$2n^3$
$x := a \pm b$	n
$X := A \pm B$	n^2

$$M_1 := H^\dagger H$$
$$M_2 := I_n - M_1$$
$$m_3 := M_2 x$$
$$m_4 := H^\dagger y$$
$$y' := m_3 + m_4$$

$\Rightarrow 2n^3 + 5n^2 + n$ FLOPs

Introduction

How to compute...

$$y' := H^\dagger y + (I_n - H^\dagger H)x \qquad \text{[TG17]}$$
$$\Leftrightarrow y' := H^\dagger y + x - H^\dagger H x$$
$$\Leftrightarrow y' := H^\dagger (y - Hx) + x$$

...with these operations?

$x := Ab$	$2n^2$
$X := AB$	$2n^3$
$x := a \pm b$	n
$X := A \pm B$	n^2

$$M_1 := H^\dagger H$$
$$M_2 := I_n - M_1$$
$$m_3 := M_2 x$$
$$m_4 := H^\dagger y$$
$$y' := m_3 + m_4$$

$\Rightarrow 2n^3 + 5n^2 + n$ FLOPs

Introduction

How to compute...

$$y' := H^\dagger y + (I_n - H^\dagger H)x \qquad \text{[TG17]}$$
$$\Leftrightarrow y' := H^\dagger y + x - H^\dagger H x$$
$$\Leftrightarrow y' := H^\dagger (y - Hx) + x$$

...with these operations?

$x := Ab$	$2n^2$
$X := AB$	$2n^3$
$x := a \pm b$	n
$X := A \pm B$	n^2

$$M_1 := H^\dagger H$$
$$M_2 := I_n - M_1$$
$$m_3 := M_2 x$$
$$m_4 := H^\dagger y$$
$$y' := m_3 + m_4$$

$\Rightarrow 2n^3 + 5n^2 + n$ FLOPs

$$m_1 := Hx$$
$$m_2 := y - m_1$$
$$m_3 := H^\dagger m_2$$
$$y' := m_3 + x$$

$\Rightarrow 2n^2 + 2n$ FLOPs

Instruction Set

BLAS [DDC+90]

- $y \leftarrow Ax + y$
- $C \leftarrow AB + C$
- $B \leftarrow A^{-1}B$
- $\dots$

LAPACK [AB+99]

- Matrix factorizations.
- Eigensolvers.
- Solvers for linear systems with specific properties.

Storage Formats

An Algebra of Banded Matrices

Code Generation in Linnea

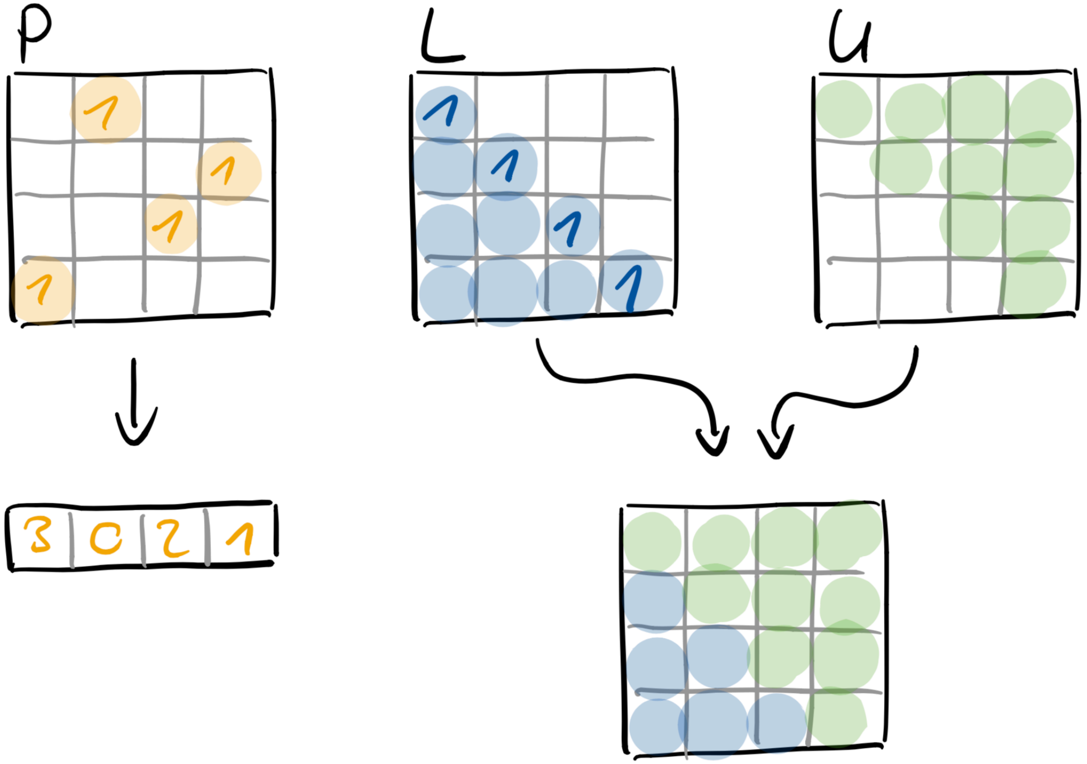
$w := AB^{-1}c$

$$L := \text{cholesky}(B)$$
$$v_1 := L^{-1}c$$
$$v_2 := L^{-T}v_1$$
$$w := Av_2$$

```
ml0 = A; ml1 = B; ml2 = c;
potrf!('L', ml1)
trsv!('L', 'N', 'N', ml1, ml2)
trsv!('L', 'T', 'N', ml1, ml2)
ml3 = Array{Float64}(1000)
gemv!('N', 1.0, ml0, ml2, 0.0, ml3)
w = ml3
```

Example: LU Factorization (`getrf`)

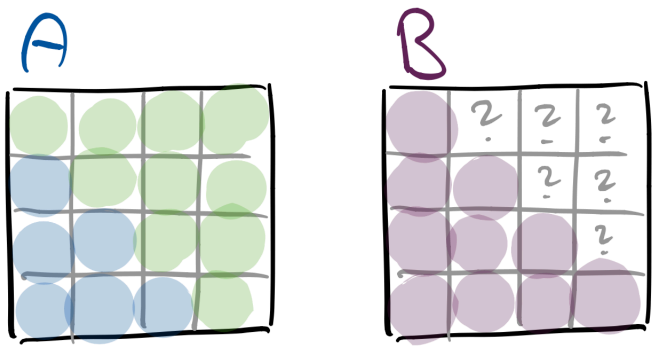
$A \to PLU$



Storage Formats

Example: Triangular solve (`trsm`)

$C \leftarrow A^{-1}B$



- A can be upper/lower triangular.
- A can have unit diagonal.

Storage Formats

The Problem

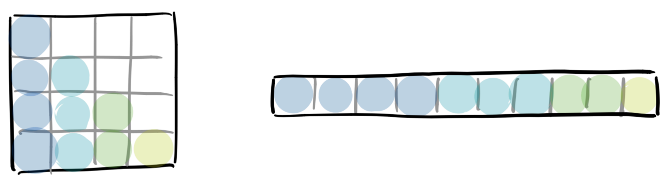
- We need to know how operands are stored.
- We need to know how kernels read operands.
- We need to be able to change storage formats.
- Goal: Only change storage formats when necessary.

Storage Formats

- (Unit-)Triangular matrices.
- Permutation.
- Symmetric matrices.

- Banded matrices.

- Packed (triangular and symmetric).



- Diagonal.

Storage Formats

Compatibility Relation

$$\texttt{lower\_triangular} \prec \texttt{full}$$
$$\texttt{full} \not\prec \texttt{lower\_triangular}$$

$\mathtt{a} \prec \mathtt{b}$ if

- all explicitly represented elements in $\mathtt{a}$ are explicitly represented in $\mathtt{b}$, and
- all elements in $\mathtt{a}$ have the same positions in $\mathtt{b}$.

Storage Formats

The Algorithm

- Given kernel K with input operands $M_1, M_2, \ldots$
- Kernels are annotated with required input formats $r_1, r_2, \ldots$
- Operands are annotated with current format $f_1, f_2, \ldots$
- Database of storage format conversions.

for all M_i:
    if $r_i \not\prec f_i$:
        find conversion $f_i \rightarrow f$ with $r_i \prec f$
        if conversion is in place:
            check if something gets overwritten

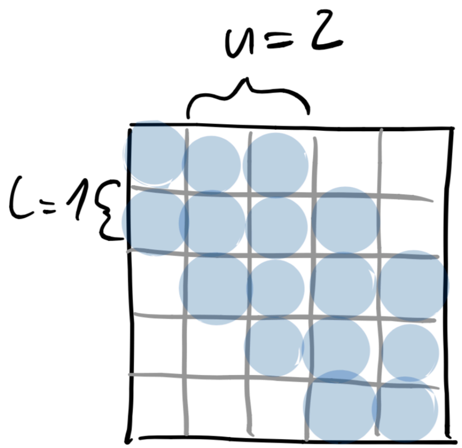
Storage Formats

An Algebra of Banded Matrices

High Performance and Automatic Computing

An Algebra of Banded Matrices

Upper and Lower Bandwidth



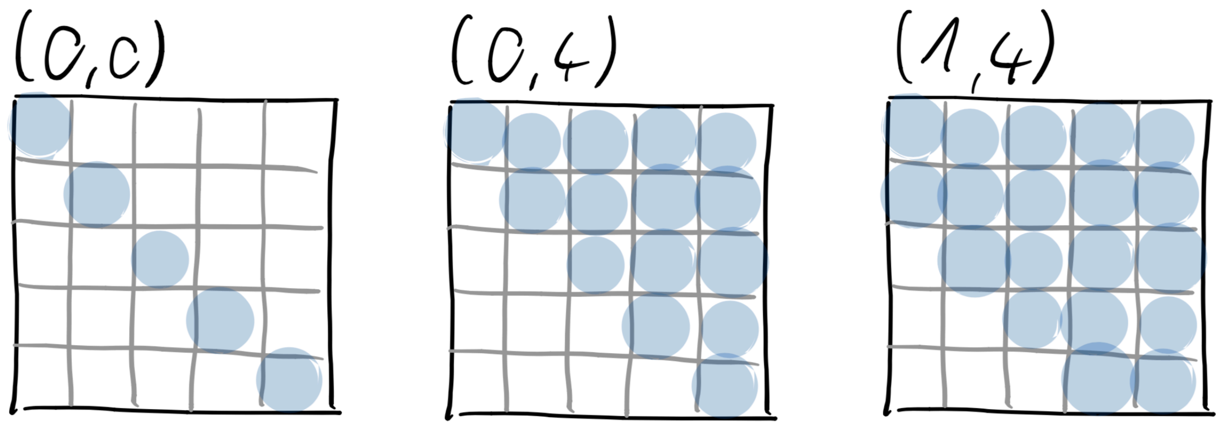
Definition [GVL13]:

$A \in \mathbb{R}^{n \times n}$ has

- lower bandwidth l if $i > j + l$ implies $a_{ij} = 0$, and
- upper bandwidth u if $j > i + u$ implies $a_{ij} = 0$.

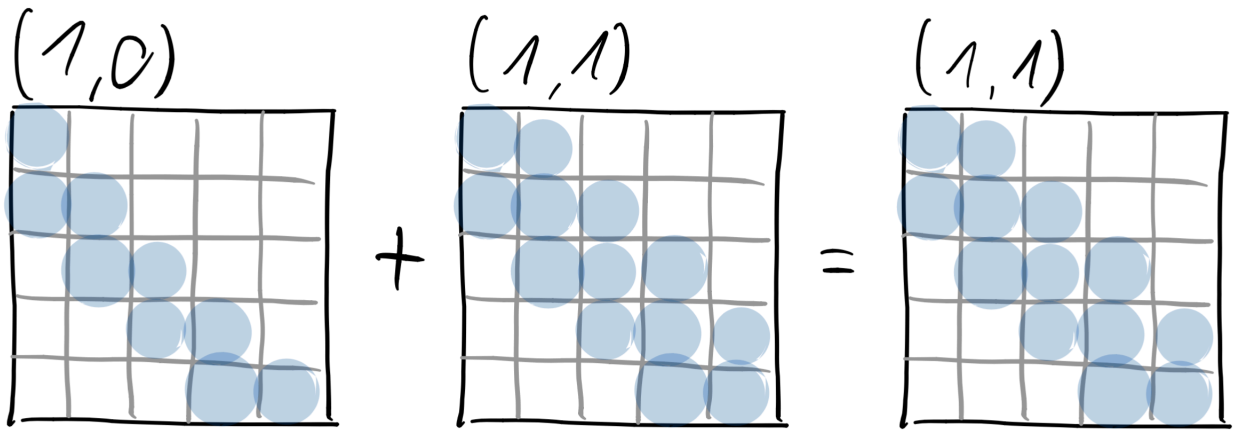
An Algebra of Banded Matrices

Examples



An Algebra of Banded Matrices

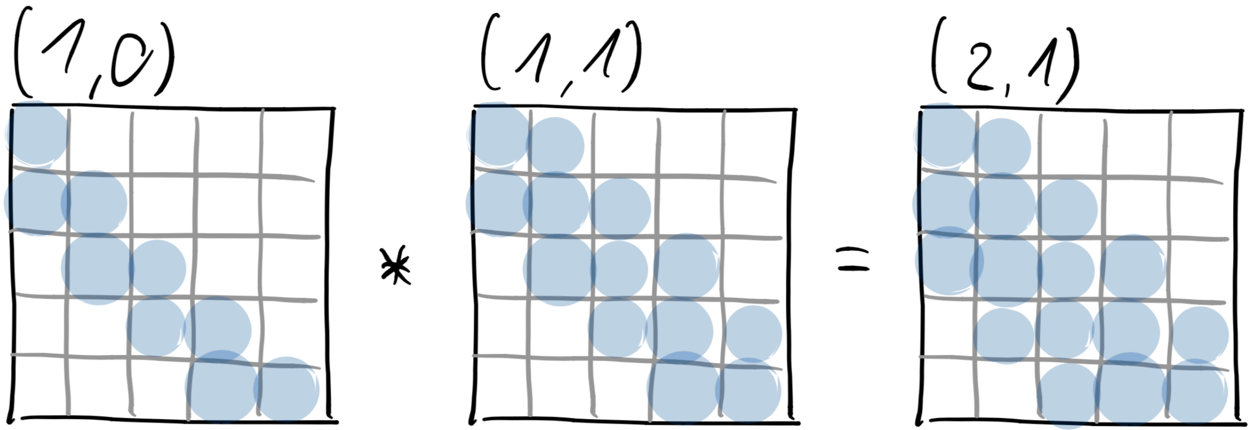
What about operations on banded matrices?



Bandwidth of $A + B$ is $(\max(l_A, l_B), \max(u_A, u_B))$.

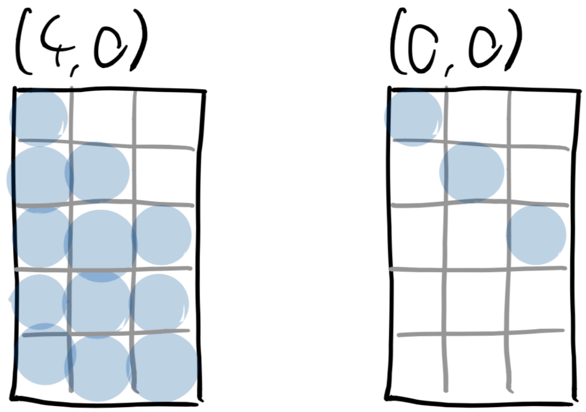
An Algebra of Banded Matrices

What about operations on banded matrices?

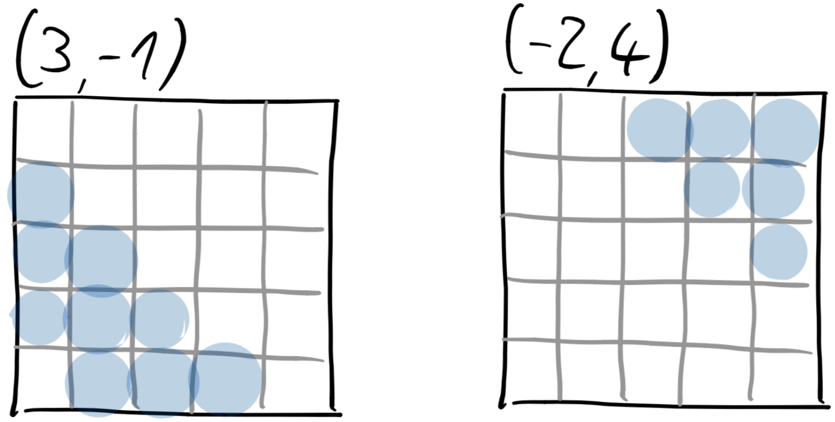


Bandwidth of $A \cdot B$ is $(l_A + l_B, u_A + u_B)$.

Generalization to Non-Square Matrices

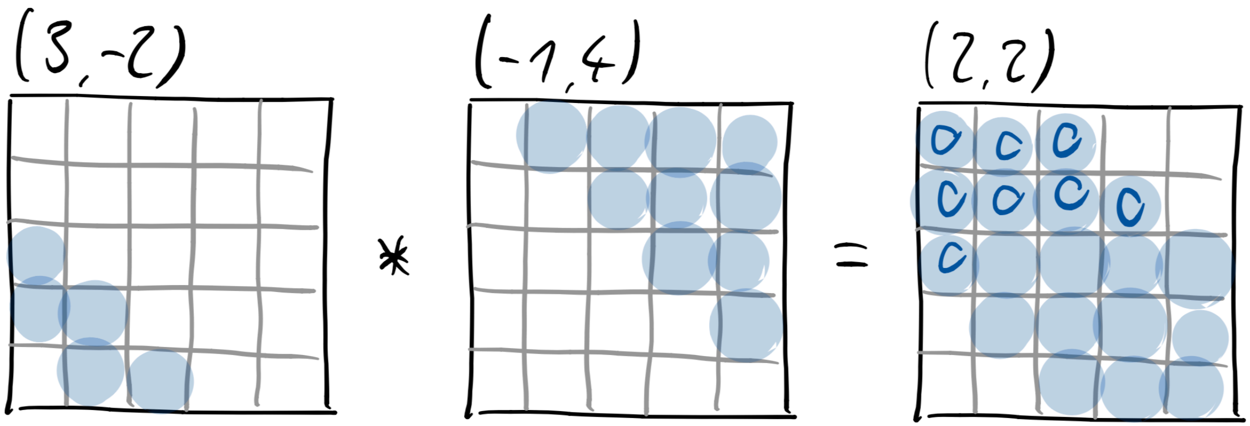


Generalization to Negative Bandwidth

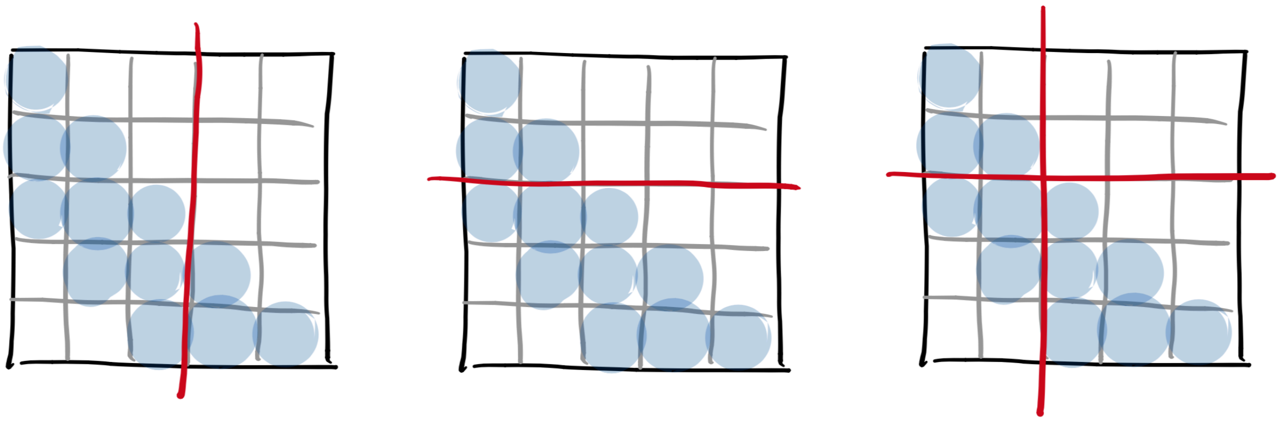


$l + u + 1 \leqslant 0$ implies that A is zero.

Problem: Overapproximation



Partitioning Matrices



An Algebra of Banded Matrices

- Simple propagation of matrix properties.
- Type system for banded matrices (`github.com/JuliaLang/julia`, issue #8240).
- Better support in libraries.

Results

Results

Example: $w := AB^{-1}c$

Naive

```
w = A*inv(B)*c
```

Recommended

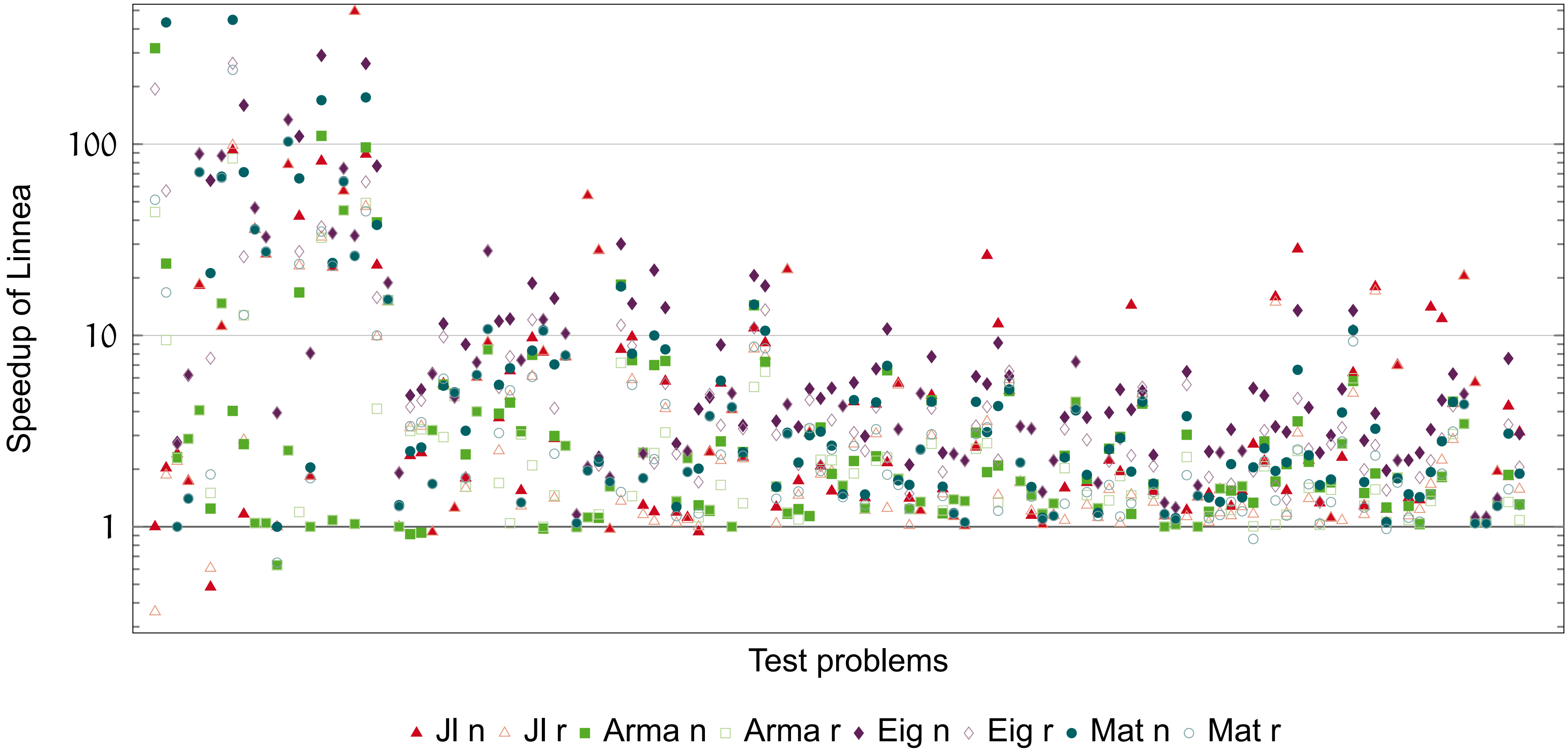
```
w = A*(B\c)
```

Generated

```
ml0 = A; ml1 = B; ml2 = c;
potrf!('L', ml1)
trsv!('L', 'N', 'N', ml1, ml2)
trsv!('L', 'T', 'N', ml1, ml2)
ml3 = Array{Float64}(1000)
gemv!('N', 1.0, ml0, ml2, 0.0, ml3)
w = ml3
```

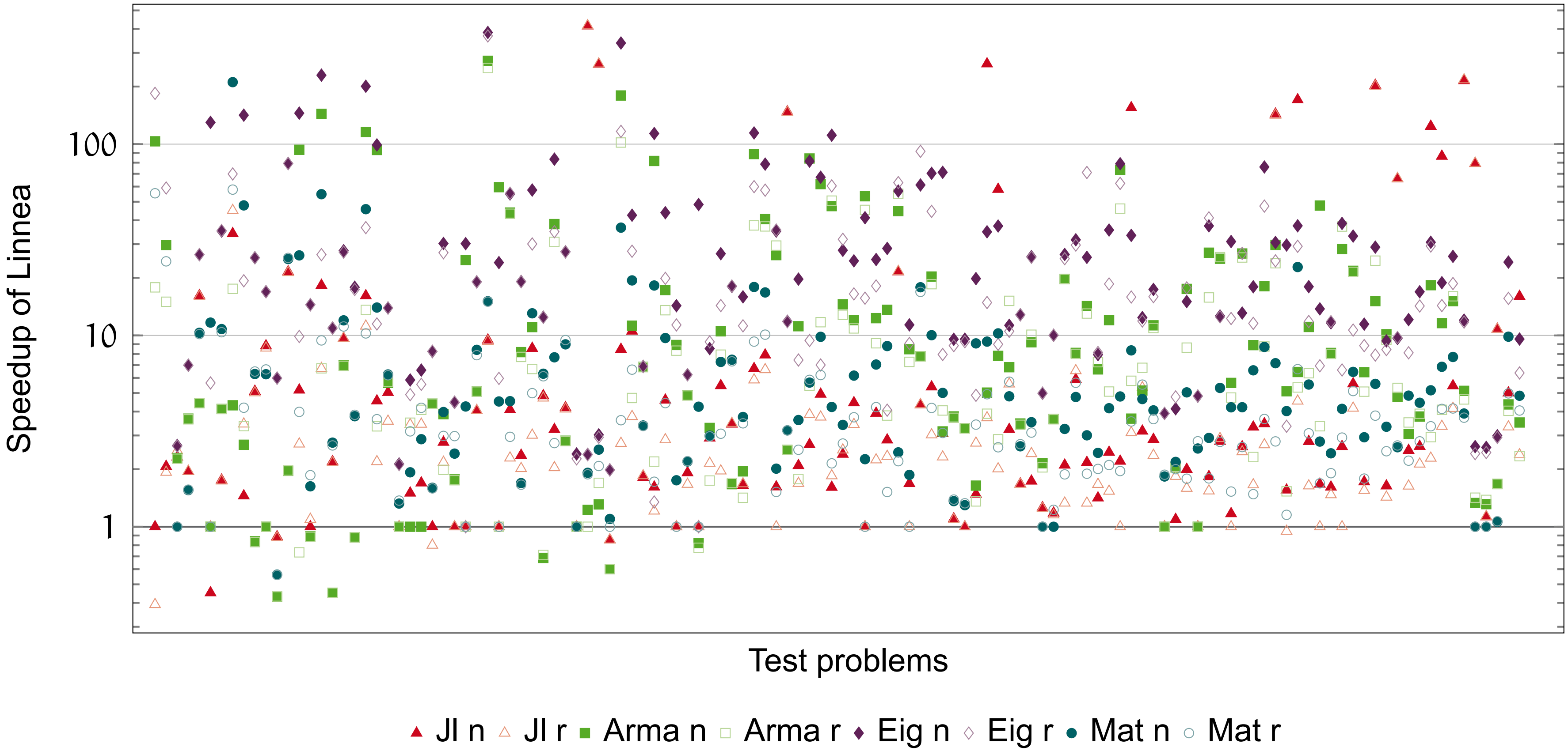
Results

1 Thread



Results

24 Threads



References

[AB+99] Edward Anderson, Zhaojun Bai, et al. *LAPACK Users' guide*, volume 9. SIAM, 1999.

[DDC+90] Jack J. Dongarra, Jeremy Du Croz, et al. A set of Level 3 Basic Linear Algebra Subprograms. *ACM TOMS*, 16(1):1–17, 1990.

[GVL13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*, volume 4. Johns Hopkins, 2013.

[HB15] Torsten Hoefler and Roberto Belli. Scientific Benchmarking of Parallel Computing Systems: Twelve Ways to Tell the Masses When Reporting Performance Results. In *the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 73–12, New York, New York, USA, November 2015. ACM.

[TG17] Tom Tirer and Raja Giryes. Image Restoration by Iterative Denoising and Backward Projections. *arXiv.org*, pages 138–142, October 2017.

Linnea is available online: `https://github.com/HPAC/linnea`