First Steps Towards a Linear Algebra Compiler

Paolo Bientinesi

AICES, RWTH Aachen
pauldj@aices.rwth-aachen.de

December 10th, 2012
ETH Zürich

The Problem

Target operations

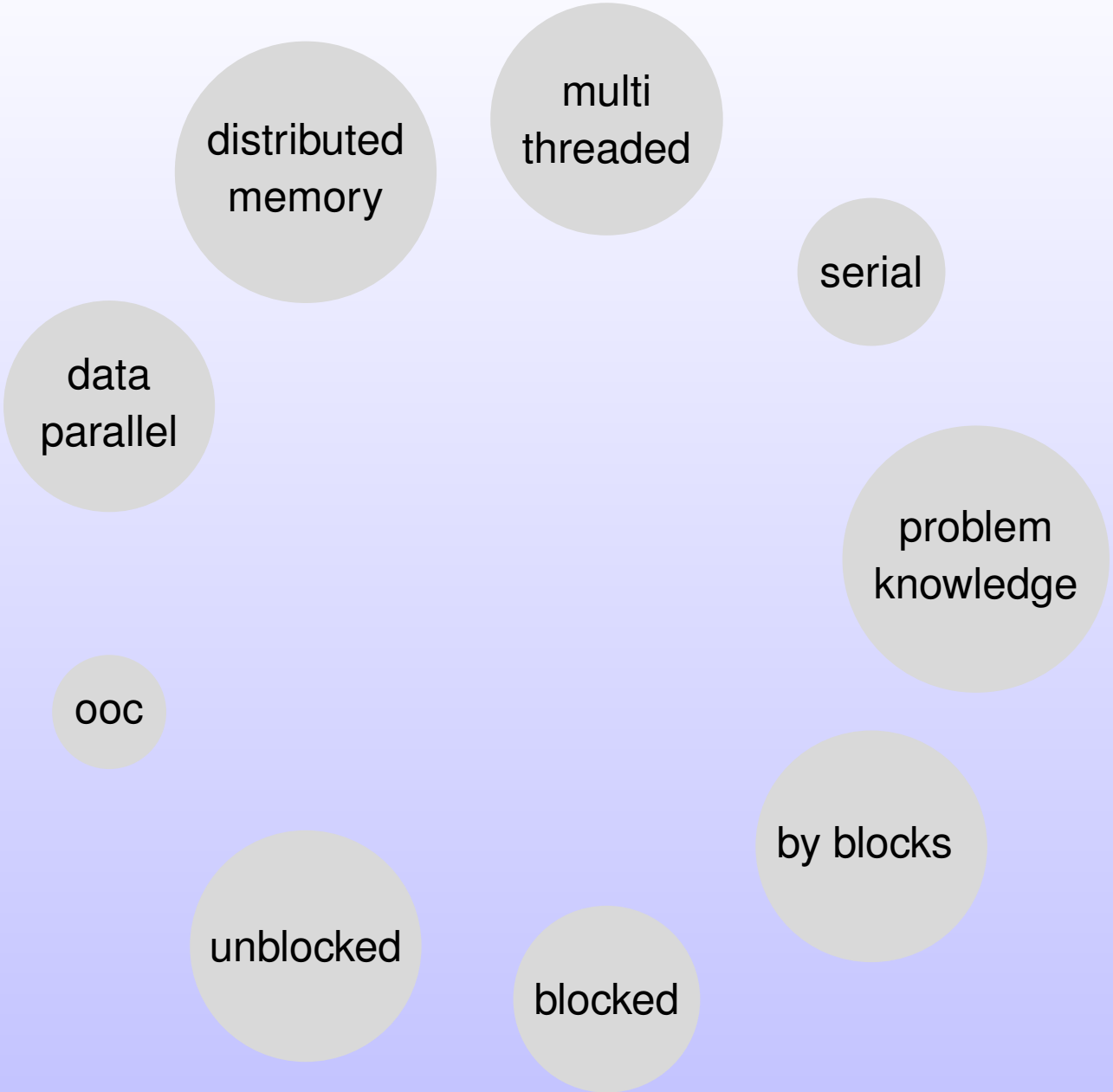
- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
- $\dfrac{d\,(Ax = b)}{dv} = ?$
- $\begin{cases} b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y \\ M = h\Phi + (1-h)I \end{cases}$
- $v_{ij} := R_i^{-T} Q_i^T y_j,$ with $\begin{cases} i = 1 \dots n \\ j = 1 \dots m \end{cases}$

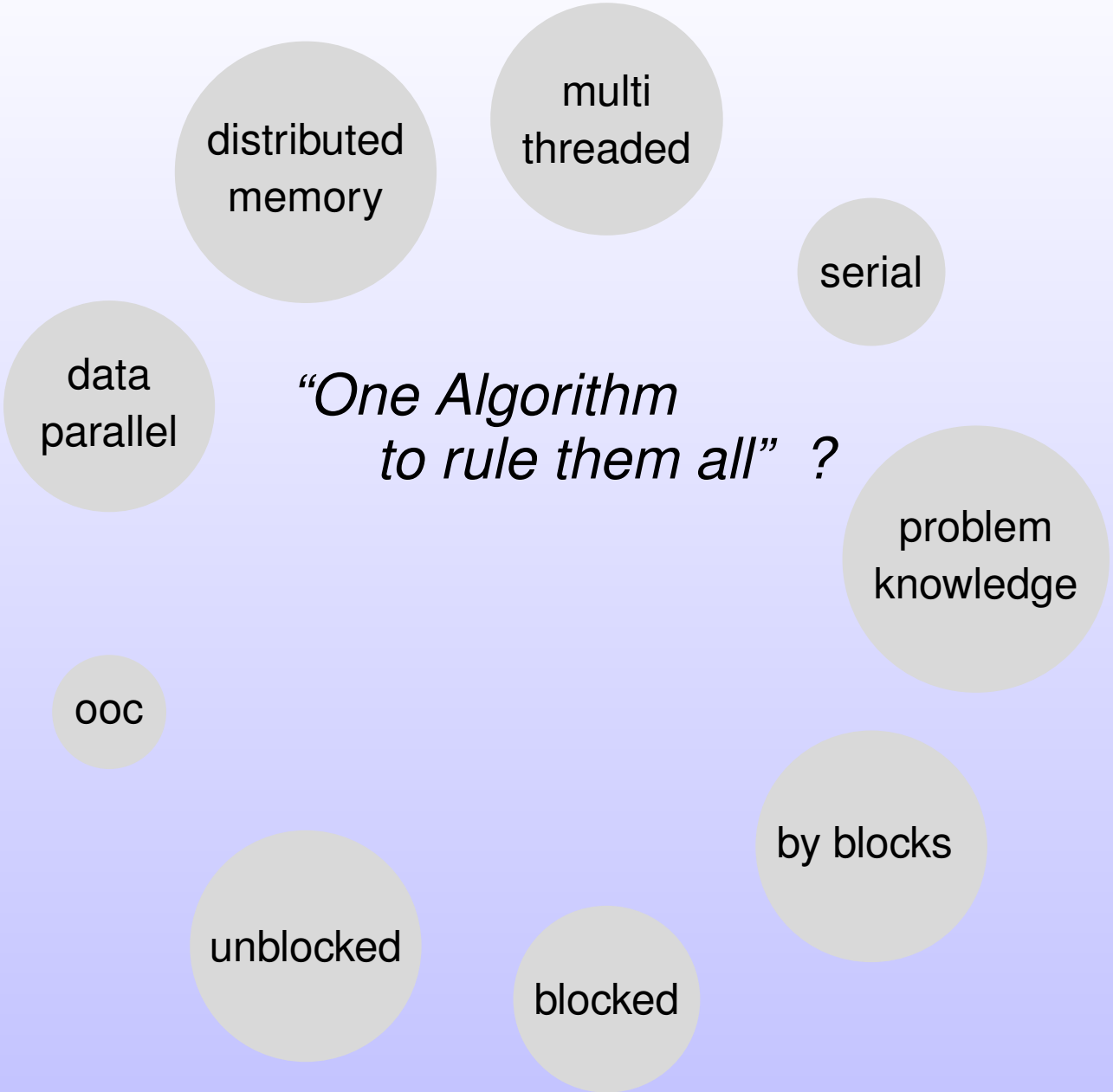
The Problem

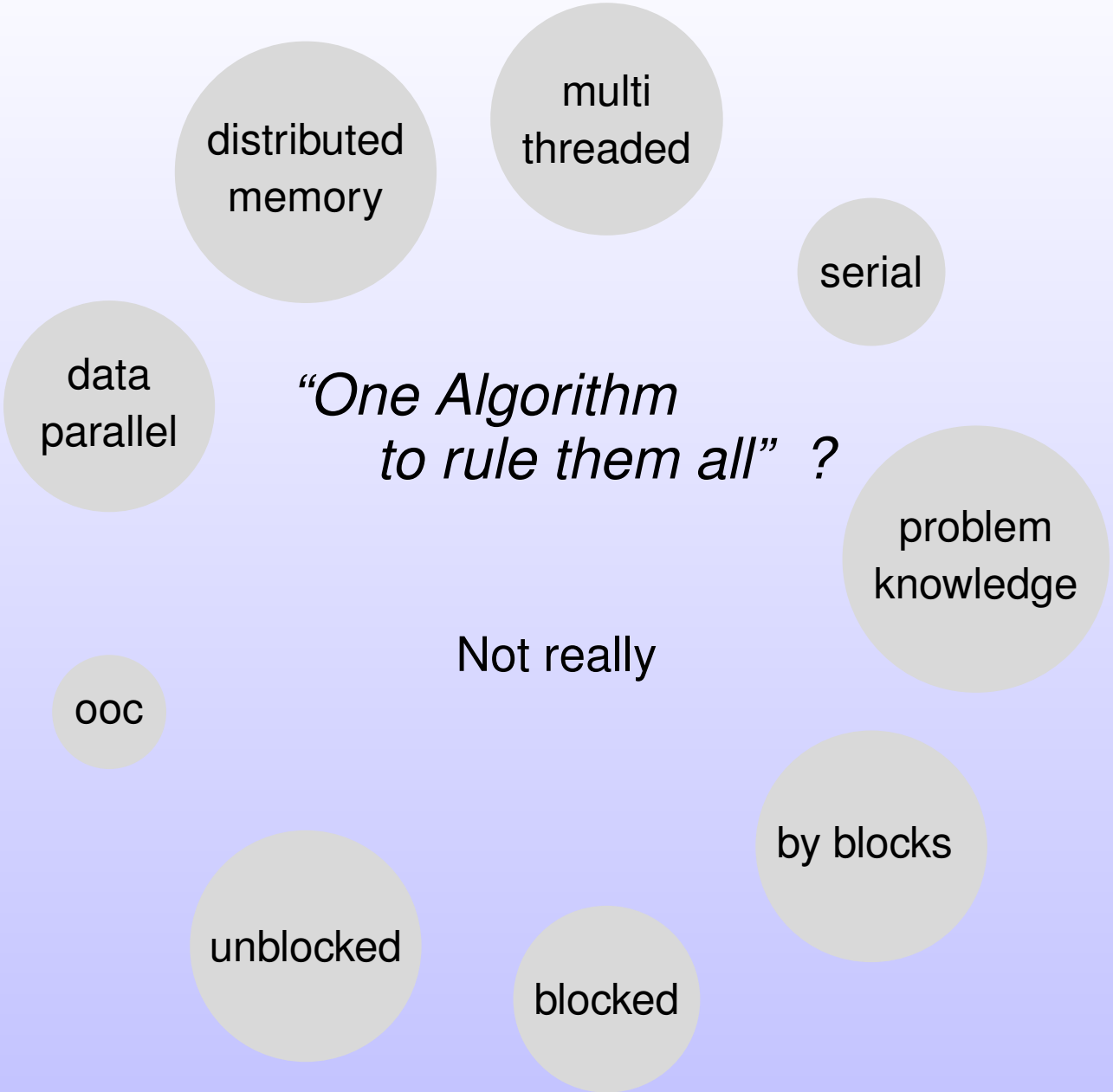
Target operations

- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
- $\dfrac{d\,(Ax = b)}{dv} = ?$
- $\begin{cases} b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y \\ M = h\Phi + (1-h)I \end{cases}$
- $v_{ij} := R_i^{-T} Q_i^T y_j, \quad \text{with} \quad \begin{cases} i = 1 \dots n \\ j = 1 \dots m \end{cases}$

- **Objective:** High performance


distributed memory
multi threaded
serial
data parallel
problem knowledge
ooc
by blocks
unblocked
blocked


distributed memory
multi threaded
serial
data parallel
"One Algorithm to rule them all" ?
problem knowledge
ooc
by blocks
unblocked
blocked


distributed memory
multi threaded
serial
data parallel
"One Algorithm to rule them all" ?
problem knowledge
Not really
ooc
by blocks
unblocked
blocked

Approach: Compiler

Approach: Compiler

```
int main(  )
{
  double vec[] =
    {  .0, 0.1, 1.2, 3.3,
     -4.0, -55, .66,   7};

  int i;

  for( i=0; i<6; i++ )
    printf(" array[%d]=%g\n",
           i, vec[i] );

  return( 0 );
}
```

Approach: Compiler

```
int main(  )
{
  double vec[] =
    {  .0, 0.1, 1.2, 3.3,
     -4.0, -55, .66,   7};

  int i;

  for( i=0; i<6; i++ )
    printf(" array[%d]=%g\n",
           i, vec[i] );

  return( 0 );
}
```

```
[...]
L3:
movl -4(%rbp), %eax
cltq
movsd -96(%rbp,%rax,8), %xmm0
movl -4(%rbp), %esi
leaq LC10(%rip), %rdi
movl $1, %eax
call _printf
incl -4(%rbp)

L2:
cmpl $9, -4(%rbp)
jle L3
movl $0, %eax
leave
ret
```

Approach: Compiler

Two stages

Approach: Compiler

Two stages

1) Generation of algorithms
   - Formal methods
   - Decomposition

Approach: Compiler

Two stages

1) Generation of algorithms
   - Formal methods
   - Decomposition
2) Ranking
   - Analytic models
   - Sampling

Approach: Compiler

Two stages

1) Generation of algorithms
   - Formal methods
   - Decomposition
2) Ranking
   - Analytic models
   - Sampling

Acknowledgments

Diego Fabregat

Generation

Roman Iakymchuk

Modeling

Elmar Peise

Prediction

Deutsche Forschungsgemeinschaft

DFG

Language

```
Eqn ::=  Eqn ; Eqn  |  Eqn & Prop  |  Id := Exp  |  Id = Exp

Exp ::=  Exp bOp Exp  |  uOp( Exp )  |  Var_idx  |  Var

bOp ::=  +  |  *

uOp ::=  -  |  T  |  Inv  |  D

Var ::=  MatrixId  |  VectorId  |  ScalarId

Prop ::=  Input  | Output  |  MatProp

MatProp ::=  Diagonal | LowerTriangular | Hermitian | ...

idx ::=  [a-z]
```

Decomposition

Decomposition

Example: $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$

Input

```
b := times( inv( times( t(X), inv(M), X ) ), t(X), inv(M), t )
{ M, Input, Matrix, n x n, SPD }
{ X, Input, Matrix, n x p }
{ y, Input, Vector, n }
{ b, Output, Vector, p }
```

Decomposition

Example: $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$

Input

```
b := times( inv( times( t(X), inv(M), X ) ), t(X), inv(M), t )
{ M, Input, Matrix, n x n, SPD }
{ X, Input, Matrix, n x p }
{ y, Input, Vector, n }
{ b, Output, Vector, p }
```

Output

$M := M^{-1}$ `dsytri(M)`

$A := X^T M$ `gemm(X, M, A, trans, notrans)`

$B := AX$ `gemm(A, X, B, notrans, notrans)`

$B := B^{-1}$ `dsytri(B)`

$v := Ay$ `gemv(A, y, v, notrans, right)`

$b := Bv$ `gemv(B, v, b, notrans, right)`

Decomposition

Example: $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$

Input

```
b := times( inv( times( t(X), inv(M), X ) ), t(X), inv(M), t )
{ M, Input, Matrix, n x n, SPD }
{ X, Input, Matrix, n x p }
{ y, Input, Vector, n }
{ b, Output, Vector, p }
```

How to efficiently compute b?

- Matlab:

  ```
  b = inv( X' * inv(M) * X ) * X' * inv(M) * y
  ```

- Objective:
  Express b in terms of available building blocks (BLAS, LAPACK, . . .)

- High-performance $\leftrightarrow$ problem-specific knowledge

Algorithm: full breakdown

$$b := \left(X^T\ M^{-1}\ X\right)^{-1} X^T\ M^{-1}\ y$$

Algorithm: full breakdown

$$b := \left(X^T \, \mathbf{M}^{-1} \, X\right)^{-1} X^T \, \mathbf{M}^{-1} \, y$$

1. $LL^T = M$ `Chol Fact`

Algorithm: full breakdown

$$b := \left(X^T \left(LL^T\right)^{-1} X\right)^{-1} X^T \left(LL^T\right)^{-1} y$$

1. $LL^T = M$ `Chol Fact`

Algorithm: full breakdown

$$b := \left(X^T\ L^{-T}\ L^{-1}\ X\right)^{-1} X^T\ L^{-T}\ L^{-1}\ y$$

1. $LL^T = M$ `Chol Fact`

Algorithm: full breakdown

$$b := \left(\mathbf{X^T}\ \mathbf{L^{-T}}\ L^{-1}\ X\right)^{-1} \mathbf{X^T}\ \mathbf{L^{-T}}\ L^{-1}\ y$$

1. $LL^T = M$ — `Chol Fact`
2. $X^T \leftarrow X^T L^{-T}$ — `TRSM`

Algorithm: full breakdown

$$b := \left(X^T\ X\right)^{-1} X^T\ L^{-1}\ y$$

1. $LL^T = M$ — `Chol Fact`
2. $X^T \leftarrow X^T L^{-T}$ — `TRSM`

Algorithm: full breakdown

$$b := \left(X^T\ X\right)^{-1} X^T\ \mathbf{L^{-1}}\ \mathbf{y}$$

1	$LL^T = M$	`Chol Fact`
2	$X^T \leftarrow X^T L^{-T}$	`TRSM`
3	$y \leftarrow L^{-1} y$	`TRSV`

Algorithm: full breakdown

$$b := \left(X^T\ X\right)^{-1} X^T\ y$$

1	$LL^T = M$	`Chol Fact`
2	$X^T \leftarrow X^T L^{-T}$	`TRSM`
3	$y \leftarrow L^{-1} y$	`TRSV`

Algorithm: full breakdown

$$b := \left(\mathbf{X^T\ X}\right)^{-1} \mathbf{X^T\ y}$$

1	$LL^T = M$	`Chol Fact`
2	$X^T \leftarrow X^T L^{-T}$	`TRSM`
3	$y \leftarrow L^{-1} y$	`TRSV`
4	$b \leftarrow (X^T X)^{-1} X^T y$	`DGELS`

Algorithm: full breakdown

$$b := \left(X^T\ X\right)^{-1} X^T\ L^{-1}\ y$$

1. $LL^T = M$ — `Chol Fact`
2. $X^T \leftarrow X^T L^{-T}$ — `TRSM`

Algorithm: full breakdown

$$b := \left(X^T \, \mathbf{X}\right)^{-1} X^T \, L^{-1} \, y$$

1. $LL^T = M$ — `Chol Fact`
2. $X^T \leftarrow X^T L^{-T}$ — `TRSM`
3. $QR = X$ — `QR Fact`

Algorithm: full breakdown

$$b := \left(R^T\ Q^T\ Q\ R\right)^{-1} R^T\ Q^T\ L^{-1}\ y$$

1	$LL^T = M$	`Chol Fact`
2	$X^T \leftarrow X^T L^{-T}$	`TRSM`
3	$QR = X$	`QR Fact`

Algorithm: full breakdown

$$b := \left(R^T \ \left(Q^T \ Q\right) \ R\right)^{-1} R^T \ Q^T \ L^{-1} \ y$$

1. $LL^T = M$ — `Chol Fact`
2. $X^T \leftarrow X^T L^{-T}$ — `TRSM`
3. $QR = X$ — `QR Fact`

Algorithm: full breakdown

$$b := \left(R^T\ R\right)^{-1} R^T\ Q^T\ L^{-1}\ y$$

1. $LL^T = M$ — `Chol Fact`
2. $X^T \leftarrow X^T L^{-T}$ — `TRSM`
3. $QR = X$ — `QR Fact`

Algorithm: full breakdown

$$b := R^{-1}\ R^{-T}\ R^T\ Q^T\ L^{-1}\ y$$

1. $LL^T = M$ — `Chol Fact`
2. $X^T \leftarrow X^T L^{-T}$ — `TRSM`
3. $QR = X$ — `QR Fact`

Algorithm: full breakdown

$$b := R^{-1} \left(R^{-T} R^{T}\right) Q^{T} L^{-1} y$$

1. $LL^T = M$ — `Chol Fact`
2. $X^T \leftarrow X^T L^{-T}$ — `TRSM`
3. $QR = X$ — `QR Fact`

Algorithm: full breakdown

$$b := R^{-1}\, Q^T\, L^{-1}\, y$$

1. $LL^T = M$ — `Chol Fact`
2. $X^T \leftarrow X^T L^{-T}$ — `TRSM`
3. $QR = X$ — `QR Fact`

Algorithm: full breakdown

$$b := R^{-1}\, Q^T\, \mathbf{L^{-1}}\, \mathbf{y}$$

1	$LL^T = M$	`Chol Fact`
2	$X^T \leftarrow X^T L^{-T}$	`TRSM`
3	$QR = X$	`QR Fact`
4	$y \leftarrow L^{-1} y$	`TRSV`

Algorithm: full breakdown

$$b := R^{-1}\ \mathbf{Q^T}\ \mathbf{y}$$

1	$LL^T = M$	`Chol Fact`
2	$X^T \leftarrow X^T L^{-T}$	`TRSM`
3	$QR = X$	`QR Fact`
4	$y \leftarrow L^{-1} y$	`TRSV`
5	$b \leftarrow Q^T y$	`GEMV`

Algorithm: full breakdown

$$b := \mathbf{R^{-1}}\ \mathbf{b}$$

1	$LL^T = M$	`Chol Fact`
2	$X^T \leftarrow X^T L^{-T}$	`TRSM`
3	$QR = X$	`QR Fact`
4	$y \leftarrow L^{-1} y$	`TRSV`
5	$b \leftarrow Q^T y$	`GEMV`
6	$b \leftarrow R^{-1} b$	`TRSV`

CLAK

Domain-specific compiler engine

- Symbolic system (Mathematica)

CLAK

Domain-specific compiler engine

- Symbolic system (Mathematica)
- **Pattern matching** & rewrite rules
- Constructive: no exhaustive search

CLAK

Domain-specific compiler engine

- Symbolic system (Mathematica)
- **Pattern matching** & rewrite rules
- Constructive: no exhaustive search
- Knowledge: 1) encoded + 2) derived

CLAK

Domain-specific compiler engine

- Symbolic system (Mathematica)
- **Pattern matching** & rewrite rules
- Constructive: no exhaustive search
- Knowledge: 1) encoded + 2) derived

Matrix algebra

Sequences — dependencies

Inference of properties

Cost analysis

Building blocks

Code generation

1) Knowledge encoded

- Operand types: scalars, vectors, matrices

  Size: $(1|m) \times (1|n)$

  Properties: full rank, orthogonal, symmetric, triangular, SPD, diagonal, identity, ...

1) Knowledge encoded

- Operand types: scalars, vectors, matrices

  Size: $(1|m) \times (1|n)$

  Properties: full rank, orthogonal, symmetric, triangular, SPD, diagonal, identity, ...

- Linear Algebra operators: +, -, $\star$, ^{T}, $^{-1}$

  Linear Algebra properties: precedence, associativity, commutativity, distributivity of transposition and inversion...

1) Knowledge encoded

- Operand types: scalars, vectors, matrices

  Size: $(1|m) \times (1|n)$

  Properties: full rank, orthogonal, symmetric, triangular, SPD, diagonal, identity, ...

- Linear Algebra operators: +, -, $\star$, ^{T}, $^{-1}$

  Linear Algebra properties: precedence, associativity, commutativity, distributivity of transposition and inversion...

- Interface to building blocks

1) Knowledge encoded

- Operand types: scalars, vectors, matrices

  Size: $(1|m) \times (1|n)$

  Properties: full rank, orthogonal, symmetric, triangular, SPD, diagonal, identity, ...

- Linear Algebra operators: +, -, $\star$, ^{T}, $^{-1}$

  Linear Algebra properties: precedence, associativity, commutativity, distributivity of transposition and inversion...

- Interface to building blocks

- Guidelines, priorities

2) Knowledge inferred

- Operand type and size
  $X \in \mathcal{R}^{n \times p},\ M \in \mathcal{R}^{n \times n} \quad \Rightarrow \quad X^T M^{-1} X \in \mathcal{R}^{p \times p}$

2) Knowledge inferred

- Operand type and size
  $X \in \mathcal{R}^{n \times p},\ M \in \mathcal{R}^{n \times n} \quad \Rightarrow \quad X^T M^{-1} X \in \mathcal{R}^{p \times p}$
- Properties
  - L is lower tri., D is diagonal, U is upper tri.
    $\Rightarrow \quad LD^{-1}U^{-T}$ is lower triangular
  - X is full rank, $SPD(M)$
    $\Rightarrow \quad SPD(X^T M^{-1} X)$

2) Knowledge inferred

- Operand type and size
  $X \in \mathcal{R}^{n \times p},\ M \in \mathcal{R}^{n \times n} \quad \Rightarrow \quad X^T M^{-1} X \in \mathcal{R}^{p \times p}$
- Properties
  - L is lower tri., D is diagonal, U is upper tri.
    $\Rightarrow \quad LD^{-1}U^{-T}$ is lower triangular
  - X is full rank, $SPD(M)$
    $\Rightarrow \quad SPD(X^T M^{-1} X)$
- Arithmetic
  $(R^T Q^T Q R)^{-1} R^T Q^T \quad \Rightarrow \quad R^{-1} Q^T$

Sequence of equations

Naive approach: `for i, for j, ...`

$$b_{ij} = \left(X_i^T M_j^{-1} X_i\right)^{-1} X_i^T M_j^{-1} y_j$$

`for` $i = 1 : m$

  `for` $j = 1 : t$

    $LL^T = M_j$

    $X^T \leftarrow X_i^T L^{-T}$

    $QR = X$

    $y \leftarrow L^{-1} y_j$

    $b \leftarrow Q^T y$

    $b_{ij} \leftarrow R^{-1} b$

Sequence of equations

Tracking the dependencies

$$LL^T = M_j \quad \Rightarrow \quad L_j L_j^T = M_j$$

$$X^T \leftarrow X_i^T L_j^{-T} \quad \Rightarrow \quad X_{ij}^T \leftarrow X_i^T L_j^{-T}$$

```
for i = 1 : m
    for j = 1 : t
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

Sequence of equations

Loop Transposition

$$LL^T = M_j \quad \Rightarrow \quad L_j L_j^T = M_j$$

$$X^T \leftarrow X_i^T L_j^{-T} \quad \Rightarrow \quad X_{ij}^T \leftarrow X_i^T L_j^{-T}$$

```
for i = 1 : m
    for j = 1 : t
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

```
for j = 1 : t
    for i = 1 : m
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

Sequence of equations

Reordering

$$LL^T = M_j \quad \Rightarrow \quad L_j L_j^T = M_j$$

$$X^T \leftarrow X_i^T L_j^{-T} \quad \Rightarrow \quad X_{ij}^T \leftarrow X_i^T L_j^{-T}$$

```
for i = 1 : m
    for j = 1 : t
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

```
for j = 1 : t
    for i = 1 : m
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

Sequence of equations

Reordering

$$LL^T = M_j \quad \Rightarrow \quad L_j L_j^T = M_j$$

$$X^T \leftarrow X_i^T L_j^{-T} \quad \Rightarrow \quad X_{ij}^T \leftarrow X_i^T L_j^{-T}$$

```
for i = 1 : m
    for j = 1 : t
        L_j L_j^T = M_j
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        y_j ← L_j^{-1} y_j
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

```
for j = 1 : t
    L_j L_j^T = M_j
    y_j ← L_j^{-1} y_j
    for i = 1 : m
        X_ij^T ← X_i^T L_j^{-T}
        Q_ij R_ij = X_ij
        b_ij ← Q_ij^T y_j
        b_ij ← R_ij^{-1} b_ij
```

A real example

Genome-wide association analysis

$$b_{ij} := (X_i^T M_j^{-1} X_i)^{-1} X_i^T M_j^{-1} y_j$$

Algorithm 1	Algorithm 2	...	Algorithm 20	...
$LL^T = M$	$LL^T = M$		$ZWZ^T = \Phi$	
$X := L^{-1}X$	$X := L^{-1}X$		$D := (hW + (1-h)I)^{-1}$	
$S := X^T X$	$QR := X$		$KK^T = D$	
$GG^T = S$	$y := L^{-1}y$		$X := Z^T X$	
$y := L^{-1}y$	$b := Q^T y$		$X := K^T X$	
$b := X^T y$	$b := R^{-1}b$		$QR := X$	
$b := G^{-1}b$			$y := L^{-1}y$	
$b := G^{-T}b$			$b := Q^T y$	
			$b := R^{-1}b$	

How to choose?

Flop count ?

Scenario	Alg. 1	Alg. 2	...	Alg. 20
Single instance	$O(n^3)$	$O(n^3)$	...	$O(n^3)$
2D sequence	$O(tn^3+mtn^2)$	$O(tn^3+mtn^2)$	...	$O(n^3+mtn)$

In general, not a good metric

Objective: Ranking

One operation $\rightarrow$ multiple algorithms

Metric,	Algorithm
	alg-1
	alg-2
	alg-3
	$\vdots$
	alg-n

Objective: Ranking

One operation $\rightarrow$ multiple algorithms

Metric,

Algorithm
alg-1
alg-2
alg-3
⋮
alg-n

$\Rightarrow$

Algorithm	Metric
alg-4	27.0
alg-1	22.5
alg-n	15.5
⋮	⋮
alg-13	1.07

Ranking of algorithms

Approach: Performance Modelling

1) Analytic Models
2) Sampling

Ranking of algorithms

Approach: Performance Modelling

1) Analytic Models
2) Sampling

Wishlist

- Speed
  - No direct execution of the algorithm
  - Possibly no execution at all

Ranking of algorithms

Approach: Performance Modelling

1) Analytic Models
2) Sampling

Wishlist

- Speed
  - No direct execution of the algorithm
  - Possibly no execution at all
- Accuracy $\Rightarrow$ accurate ranking

Ranking of algorithms

Approach: Performance Modelling

1) Analytic Models
2) Sampling

Wishlist

- Speed
  - No direct execution of the algorithm
  - Possibly no execution at all
- Accuracy $\Rightarrow$ accurate ranking
- Automation

1) Analytic modelling

- no execution of code
- models built from knowledge

1) Analytic modelling

- no execution of code
- models built from knowledge

Model (simplified version)

$$\texttt{Time} = \alpha \, \#\texttt{flops} + \sum_i \beta_i \, \#\texttt{miss}_i$$

1) Analytic modelling

- no execution of code
- models built from knowledge

Model (simplified version)

$$\texttt{Time} = \alpha \ \#\texttt{flops} + \sum_i \beta_i \ \#\texttt{miss}_i$$

- storage scheme
- size of the operands
- size and number of caches
- hardware & software prefetching

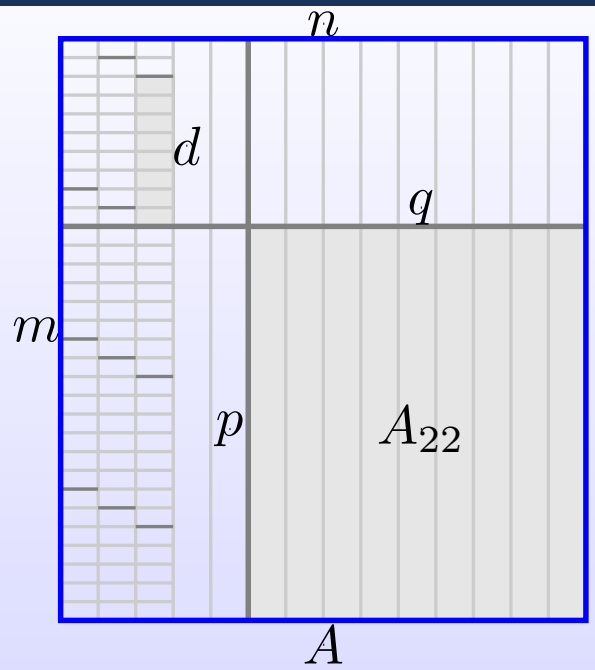
- how the algorithm traverses the operands
- size of cache-lines
- compiler optimizations
- ...

Example: GotoBLAS

Rank-k update

$A := A + xy^T$

GER, BLAS2

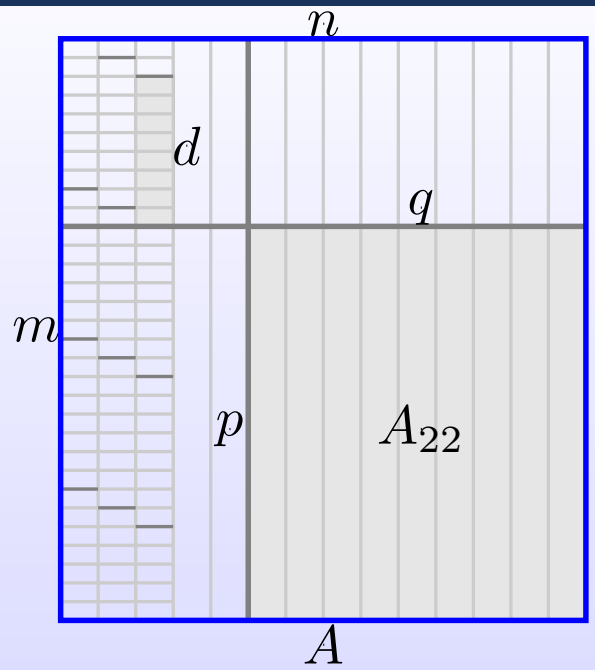


Example: GotoBLAS

Rank-k update

$A := A + xy^T$

`GER`, BLAS2



L1 misses =

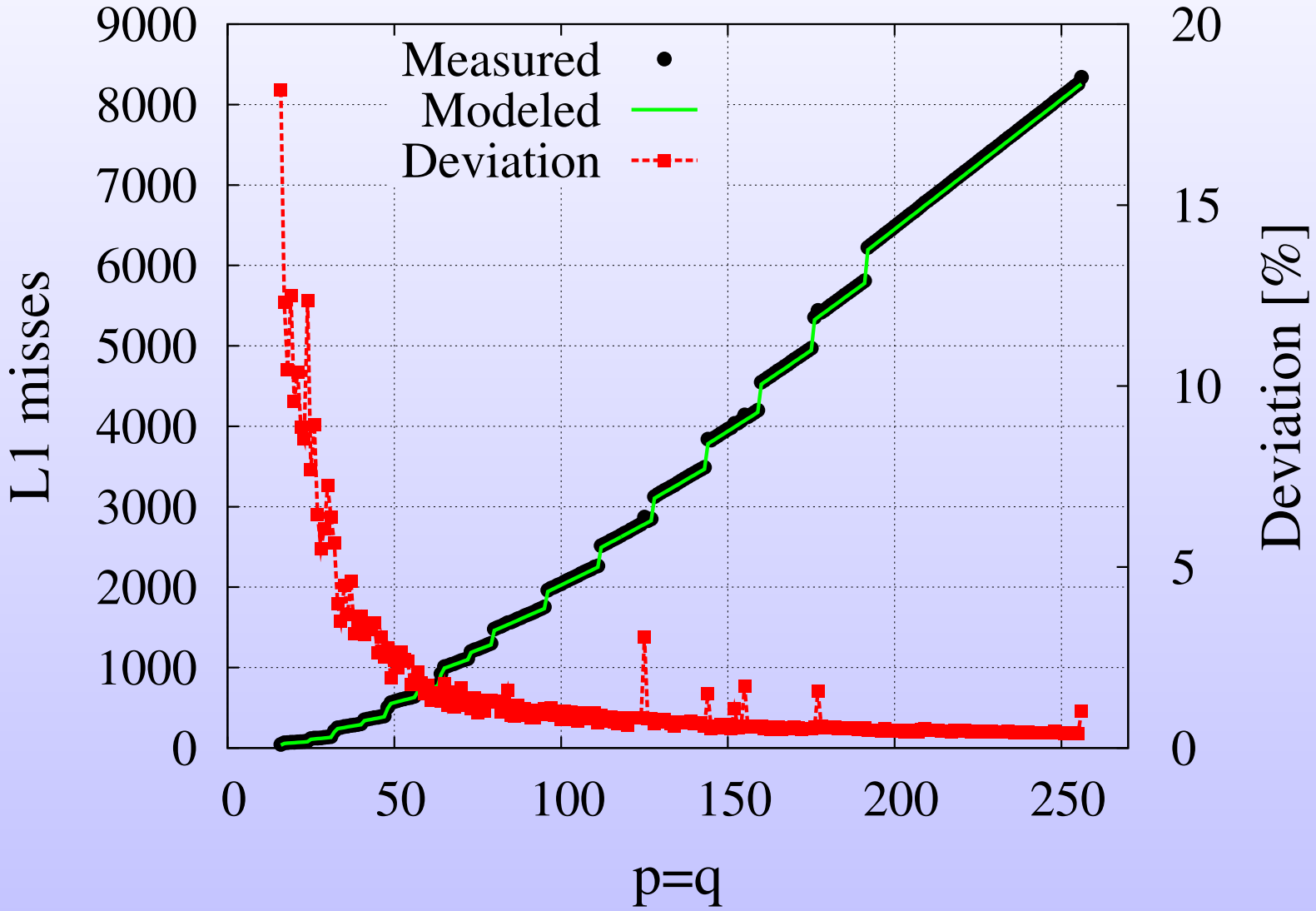
$$\begin{cases} \left\lceil \frac{p}{d} \right\rceil + \left\lceil \frac{q}{d} \right\rceil + \left\lfloor \frac{mq}{d} \right\rfloor, & \text{if } m - p < d \\ 2\left\lceil \frac{p}{d} \right\rceil + \left\lceil \frac{q}{d} \right\rceil + \sum_{i=1}^{q-1} \left(\left\lceil \frac{p + (mi \bmod d)}{d} \right\rceil + \eta(i) \right), & \text{otherwise} \end{cases}$$

with

$$\eta(i) = \min\left(d - 1, \left\lfloor \frac{m + (mi \bmod d)}{d} \right\rfloor - \left\lceil \frac{p + (mi \bmod d)}{d} \right\rceil\right)$$

Accuracy

Cache misses: GER, GotoBLAS2



Analytic models

Wishlist

Analytic models

Wishlist

- Speed ✓✖

Analytic models

Wishlist

- Speed ✓ ✖
  - No direct execution of the algorithm ✓
  - Possibly no execution at all ✓

Analytic models

Wishlist

- Speed ✓ ✖
  - No direct execution of the algorithm ✓
  - Possibly no execution at all ✓
- Accuracy ✓ $\Rightarrow$ accurate ranking

Analytic models

Wishlist

- Speed ✓✖
  - No direct execution of the algorithm ✓
  - Possibly no execution at all ✓
- Accuracy ✓ $\Rightarrow$ accurate ranking
- Automation ✖

2) Modelling through sampling

2) Modelling through sampling

Roadmap

- Sample the kernels – not the algorithm!

2) Modelling through sampling

Roadmap

- Sample the kernels – not the algorithm!
- Build multivariate piecewise polynomial models

2) Modelling through sampling

Roadmap

- Sample the kernels – not the algorithm!
- Build multivariate piecewise polynomial models
- Create a database

2) Modelling through sampling

Roadmap

- Sample the kernels – not the algorithm!
- Build multivariate piecewise polynomial models
- Create a database
- Algorithm execution $\equiv$ querying

Sampling

A X = B

```
dtrsm(side, uplo, transA, diag, m, n, alpha, A, ldA, B, ldB)
```

Sampling

A X = B

```
dtrsm(side, uplo, transA, diag, m, n, alpha, A, ldA, B, ldB)
```

blind sampling $\Rightarrow$ curse of dimensionality $\Rightarrow$ intractable
low accuracy

Sampling

A X = B

```
dtrsm(side, uplo, transA, diag, m, n, alpha, A, ldA, B, ldB)
```

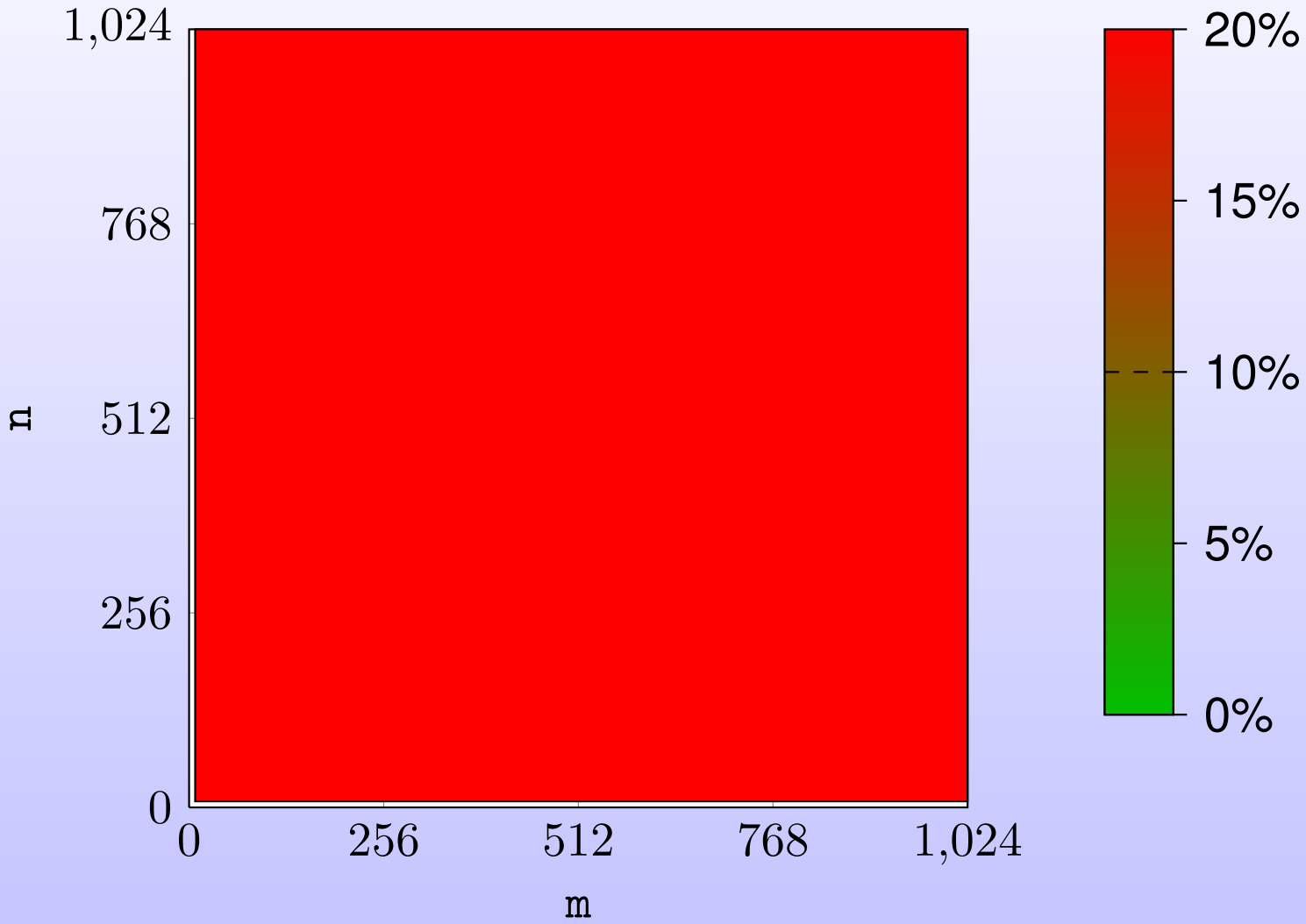
blind sampling $\Rightarrow$ curse of dimensionality $\Rightarrow$ intractable
low accuracy

Solution:

- Understand the kernels
- Integrate knowledge into the modeling and models

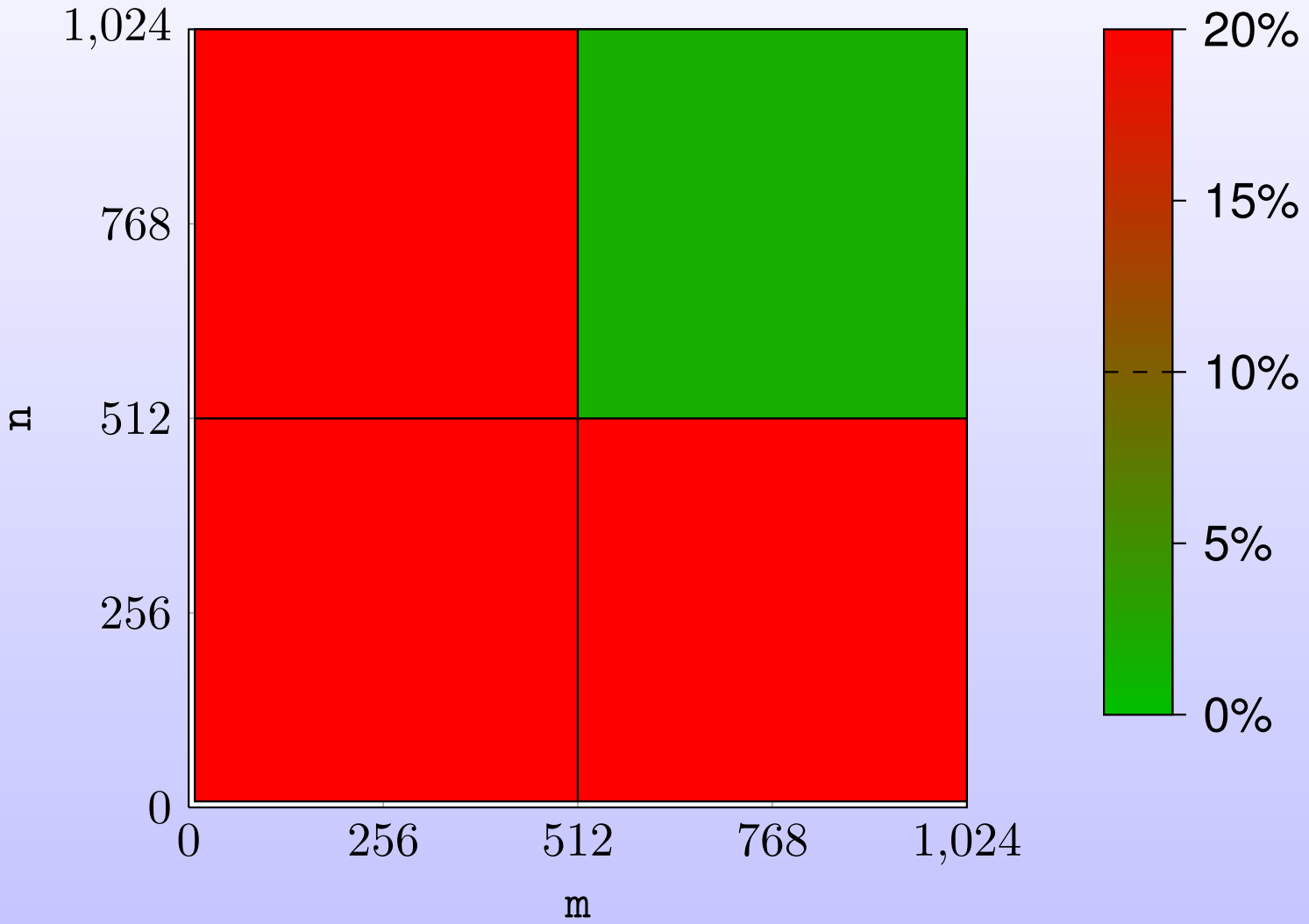
Adaptive Refinement

`dtrsm(L, L, N, N, m, n, .5,` L`, 2500,` B`, 2500)`



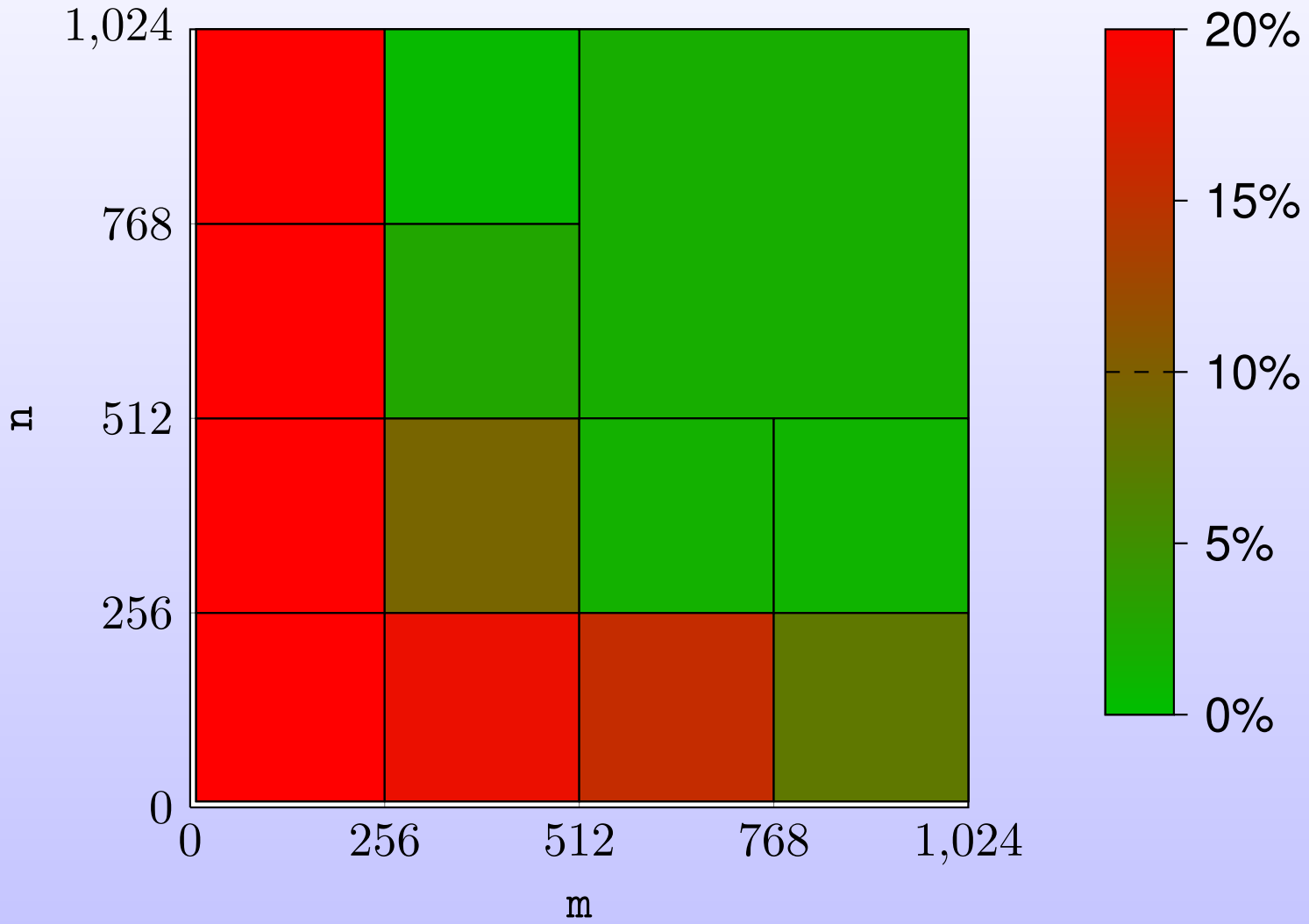
Adaptive Refinement

`dtrsm(L, L, N, N, m, n, .5,` L`, 2500,` B`, 2500)`



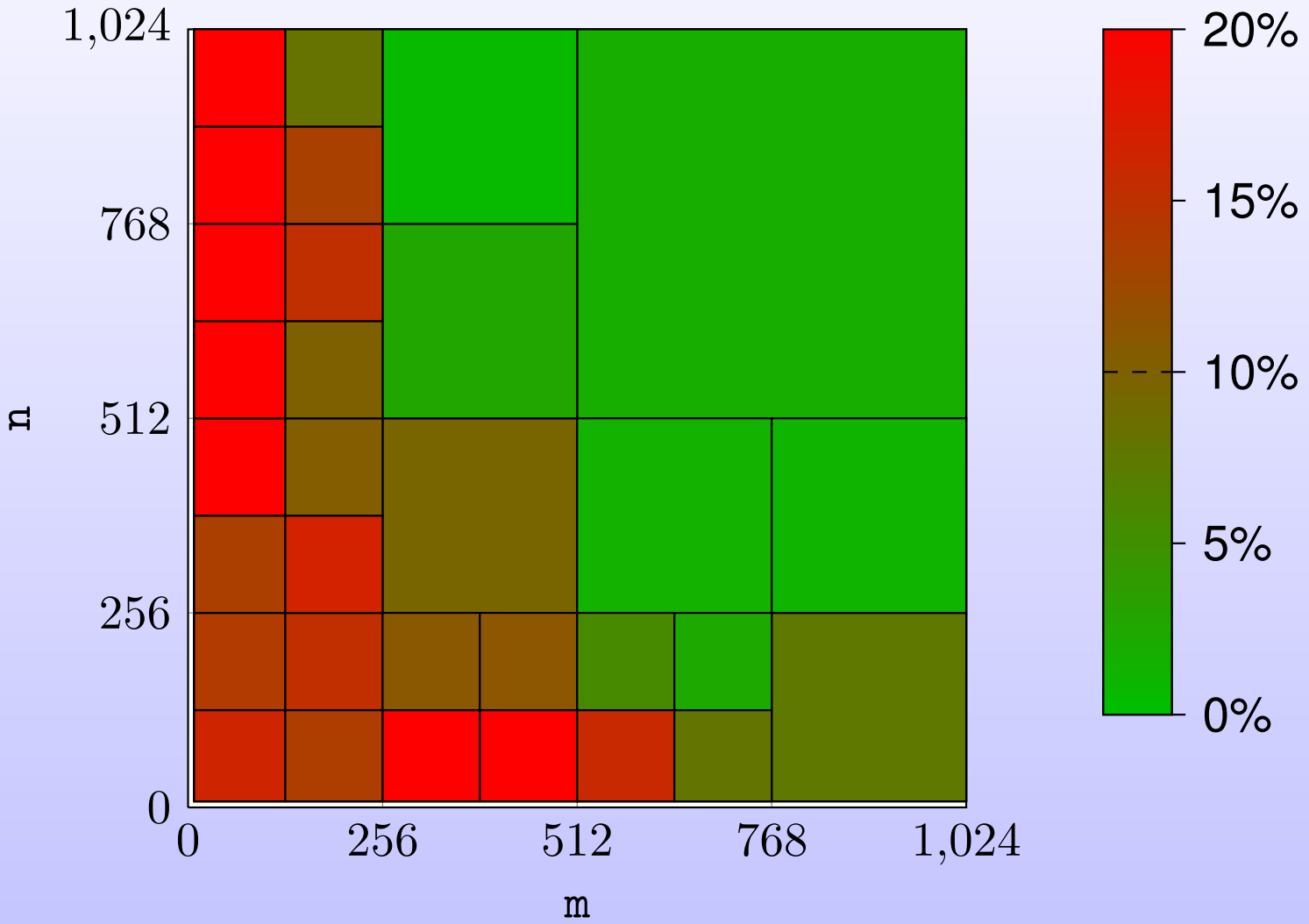
Adaptive Refinement

`dtrsm(L, L, N, N, m, n, .5,` L`, 2500,` B`, 2500)`



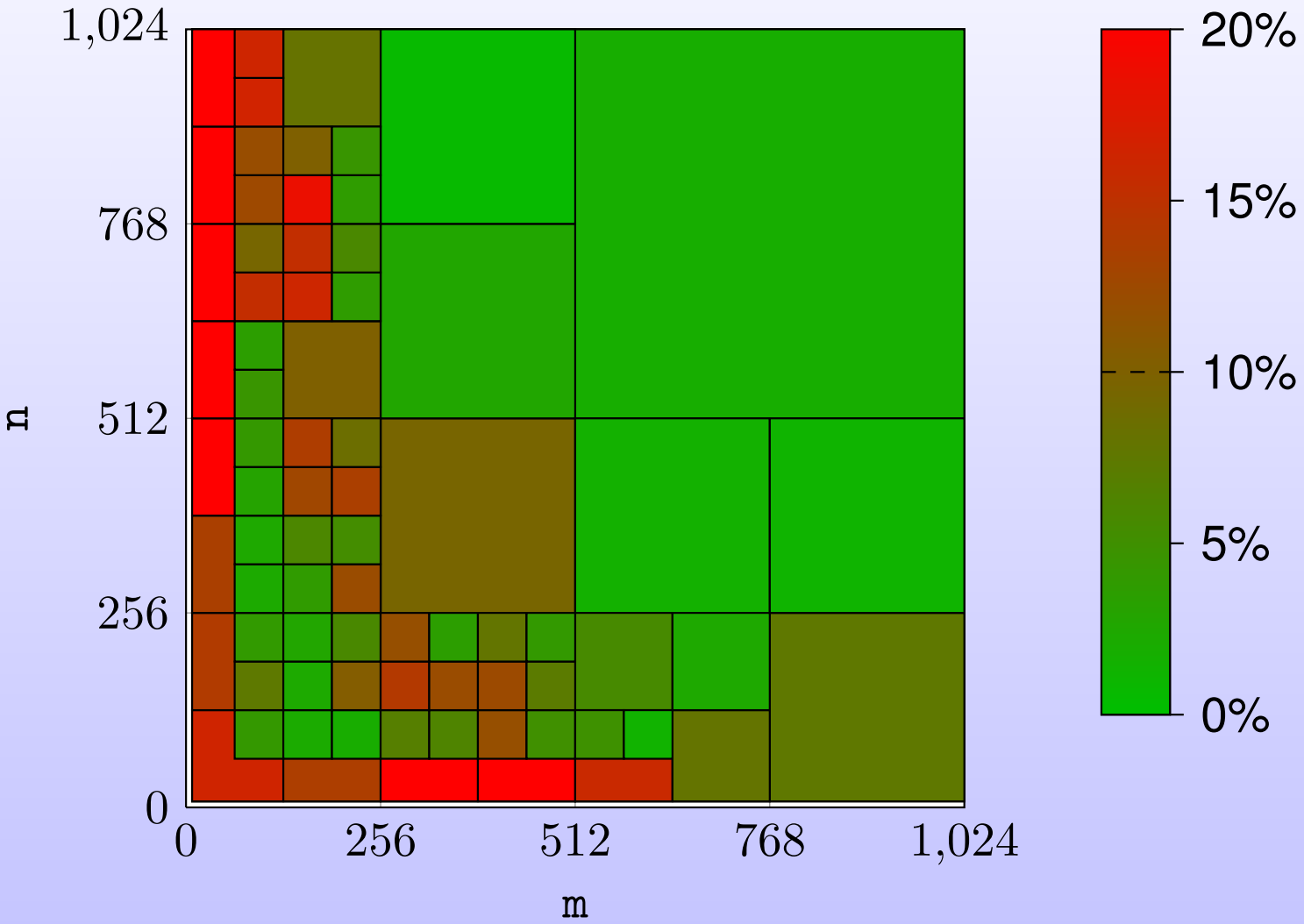
Adaptive Refinement

`dtrsm(L, L, N, N, m, n, .5,` L`, 2500,` B`, 2500)`



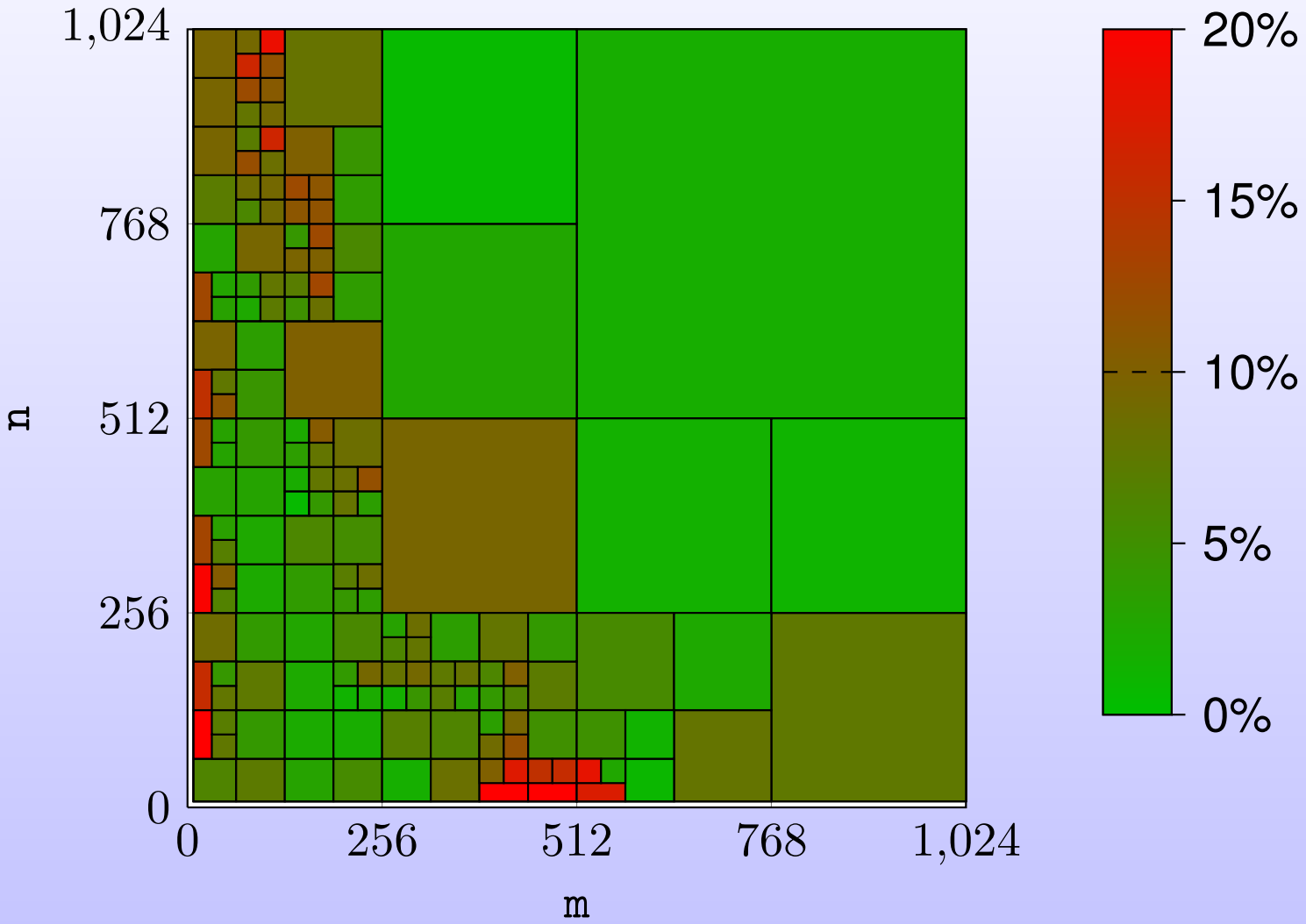
Adaptive Refinement

`dtrsm(L, L, N, N, m, n, .5,` L`, 2500,` B`, 2500)`



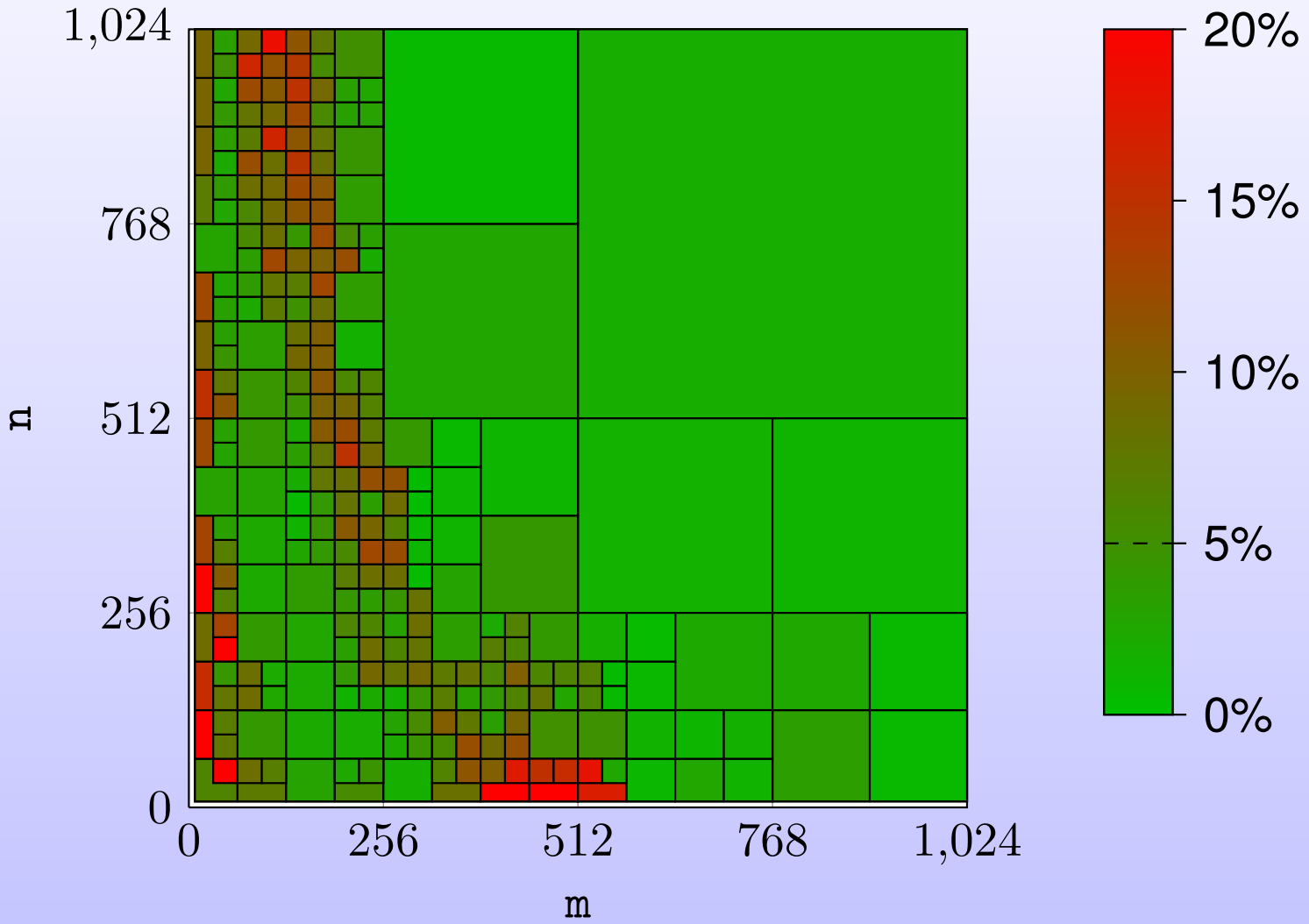
Adaptive Refinement

`dtrsm(L, L, N, N, m, n, .5,` L`, 2500,` B`, 2500)`



Adaptive Refinement

dtrsm(L, L, N, N, m, n, .5, L, 2500, B, 2500)



From algorithm to prediction

TriInv_1('L',300,A,300,100)

Partition $L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)$

where L_{TL} is 0×0

While $size(L_{TL}) < size(L)$ **do**

$$\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline L_{10} & L_{11} & 0 \\ \hline L_{20} & L_{21} & L_{22} \end{array}\right)$$

$$L_{10} := \texttt{TRMM}(L_{10}, L_{00})$$
$$L_{10} := \texttt{TRSM}(-L_{11} L_{10})$$
$$L_{11} := \texttt{trinv}(L_{11})$$

$$\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline L_{10} & L_{11} & 0 \\ \hline L_{20} & L_{21} & L_{22} \end{array}\right)$$

endwhile

From algorithm to prediction

TriInv_1('L',300,A,300,100)

Partition $L \rightarrow \left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right)$

where L_{TL} is 0×0

While $size(L_{TL}) < size(L)$ **do**

$$\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline L_{10} & L_{11} & 0 \\ \hline L_{20} & L_{21} & L_{22} \end{array}\right)$$

$L_{10} := \texttt{TRMM}(L_{10}, L_{00})$

$L_{10} := \texttt{TRSM}(-L_{11} L_{10})$

$L_{11} := \texttt{trinv}(L_{11})$

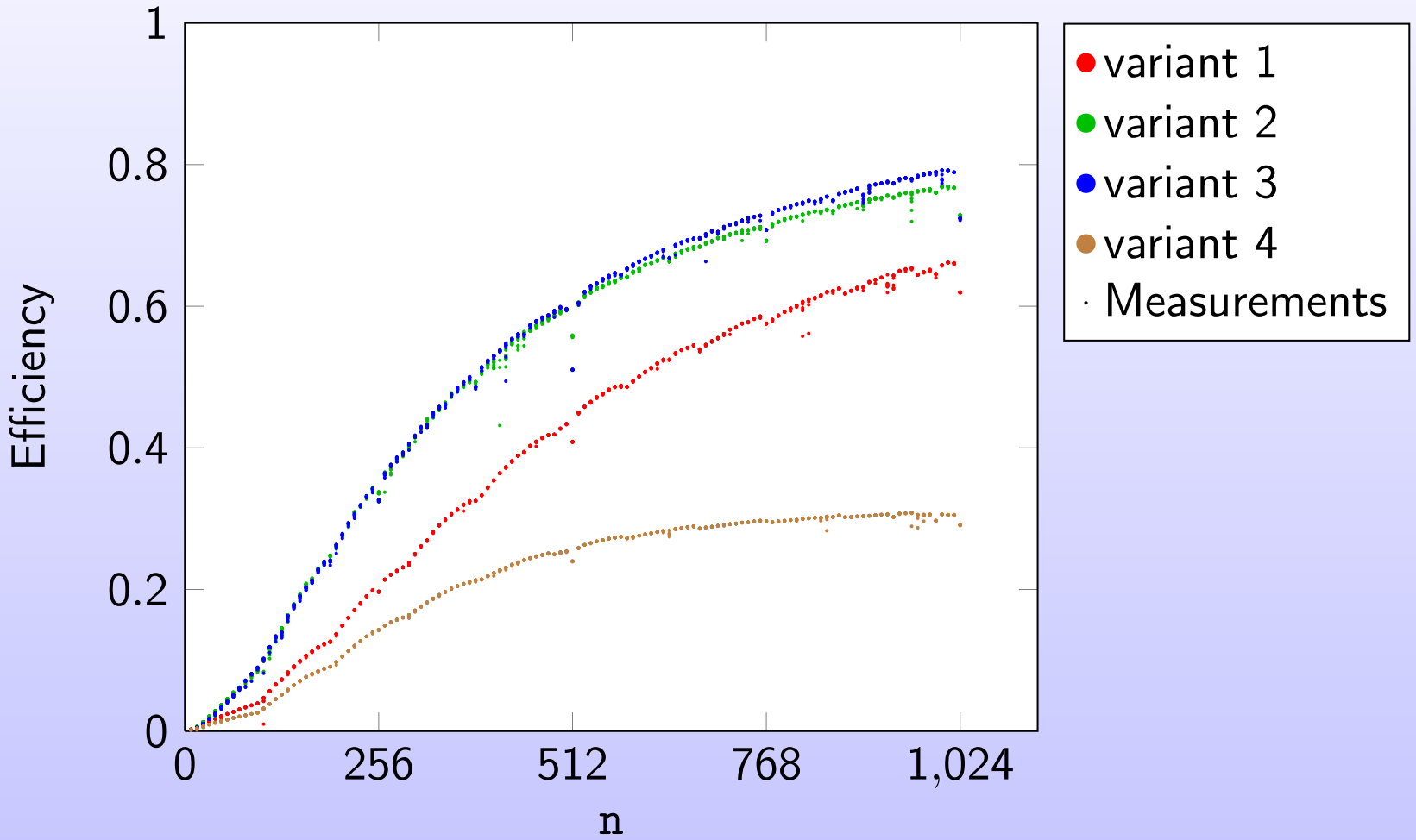
$$\left(\begin{array}{c|c} L_{TL} & 0 \\ \hline L_{BL} & L_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} L_{00} & 0 & 0 \\ \hline L_{10} & L_{11} & 0 \\ \hline L_{20} & L_{21} & L_{22} \end{array}\right)$$

endwhile

```
dtrmm(100, 0, 1, 300, 300)
dtrsm(100, 0, -1, 300, 300)
triinv_1('L', 100, 300, 1)
dtrmm(100, 100, 1, 300, 300)
dtrsm(100, 100, -1, 300, 300)
triinv_1('L', 100, 300, 1)
dtrmm(100, 200, 1, 300, 300)
dtrsm(100, 200, -1, 300, 300)
triinv_1('L', 100, 300, 1)
```

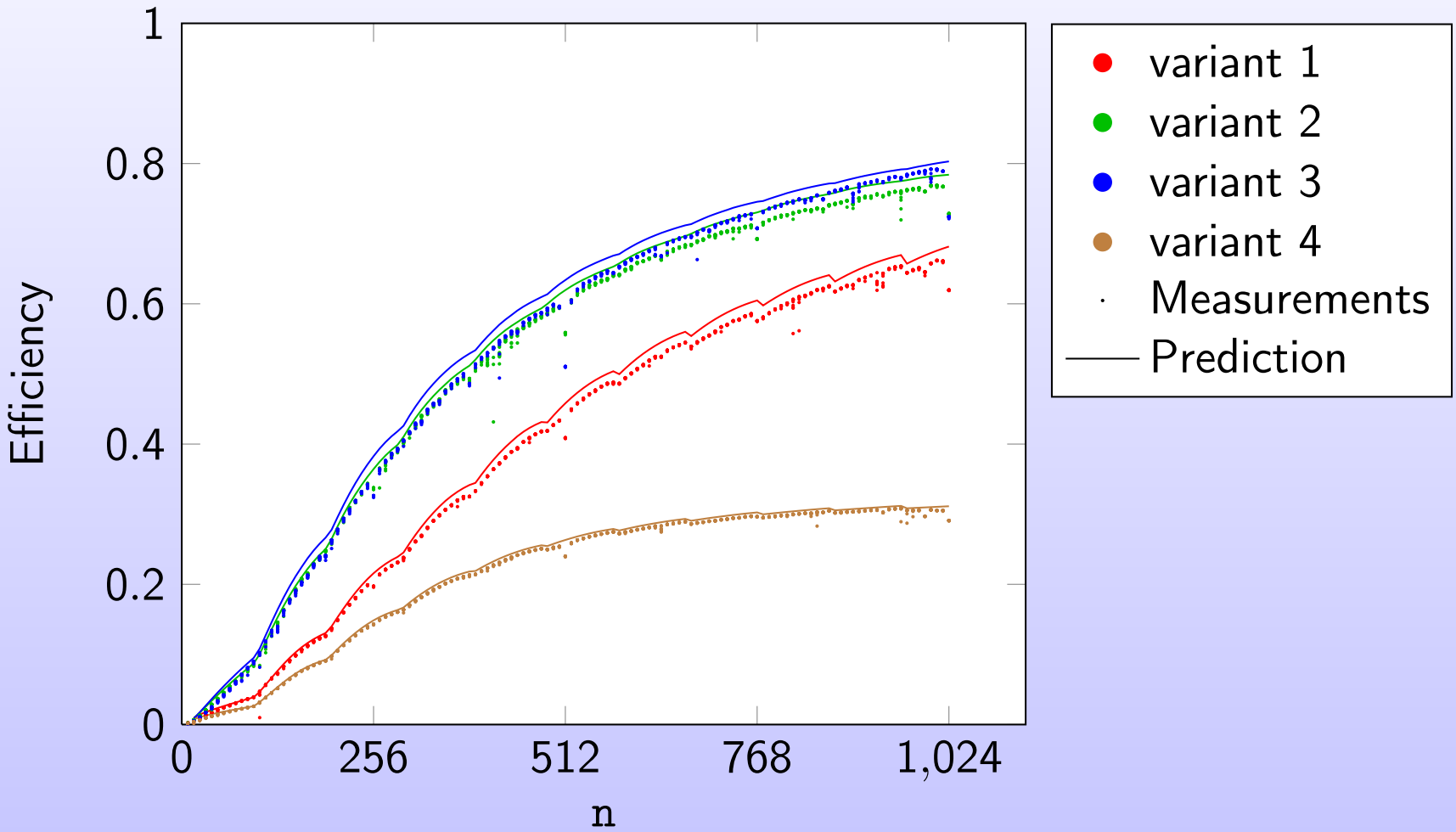
Results: Intel Harpertown E5450

$X := L^{-1}$



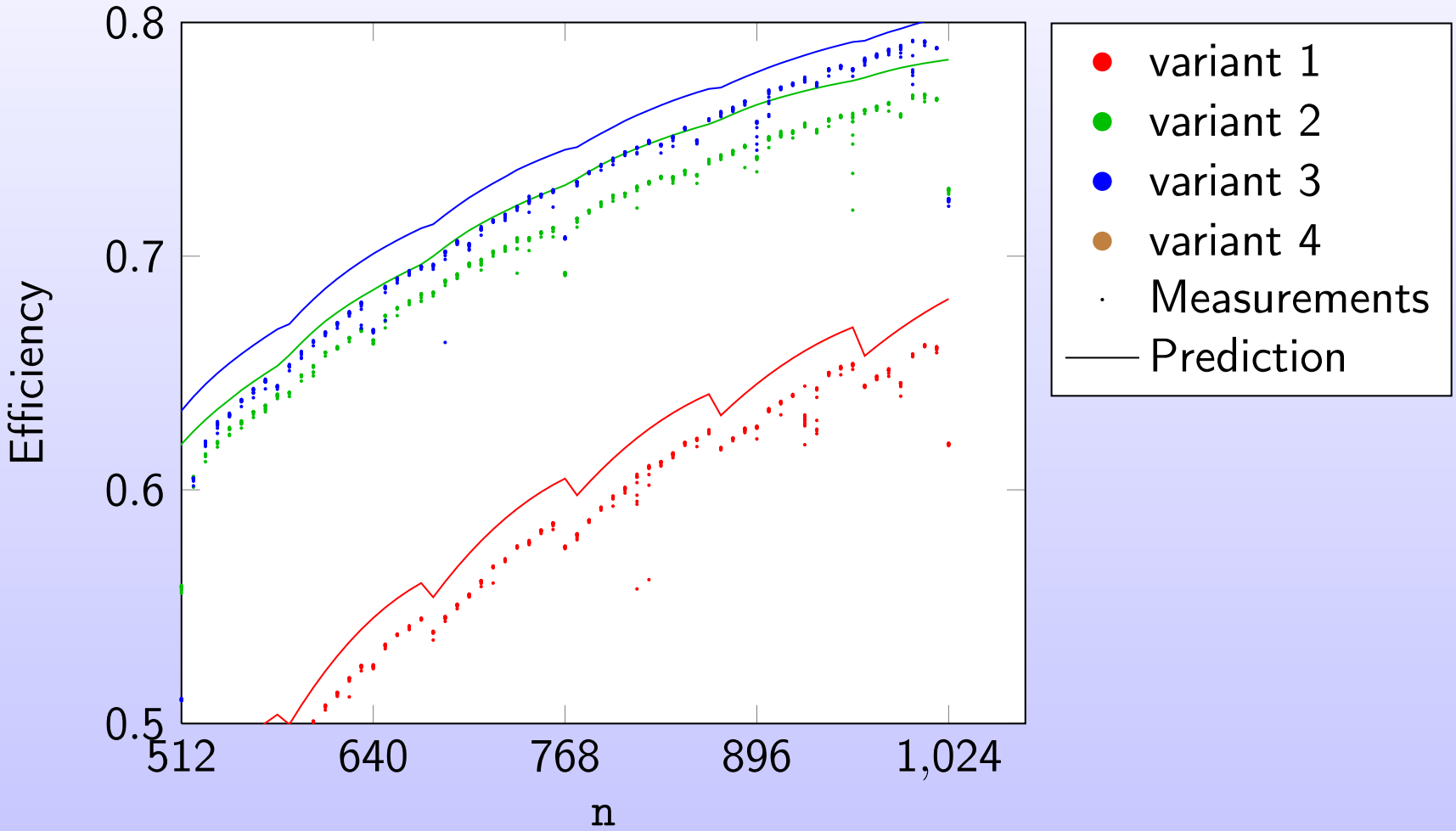
Results: Ranking

$X := L^{-1}$



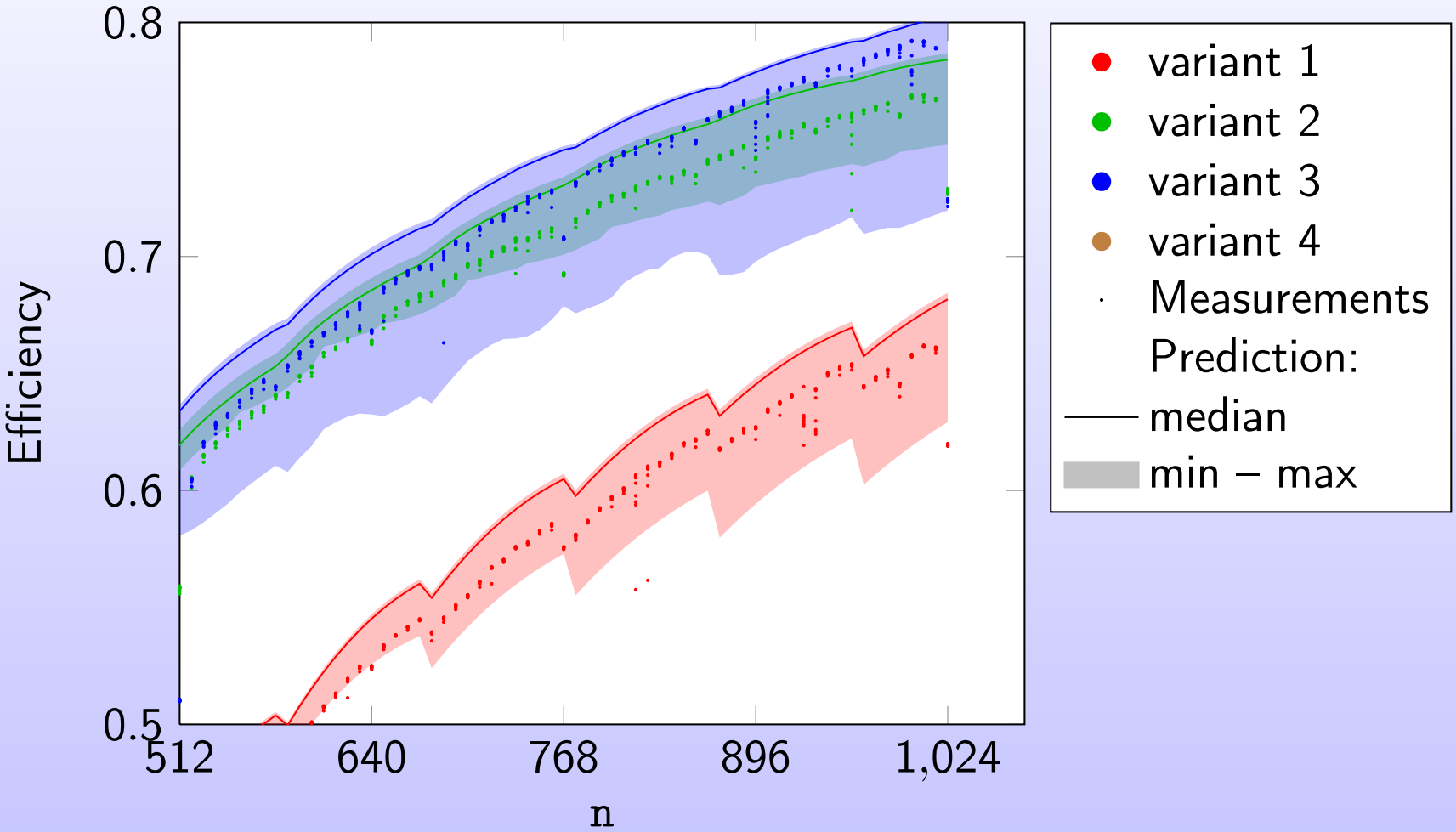
Results: Zoom

$X := L^{-1}$



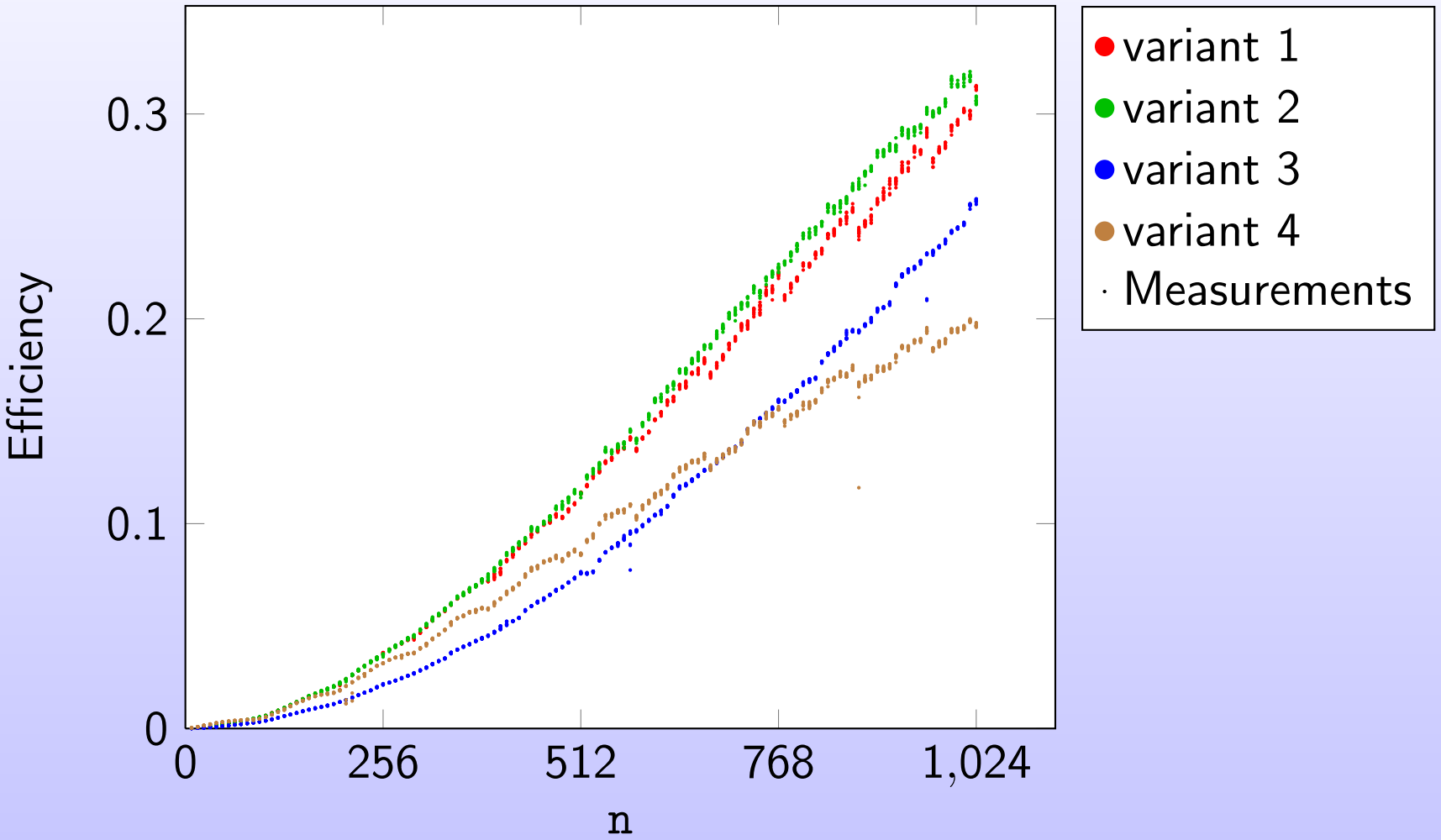
Results: Statistics

$X := L^{-1}$



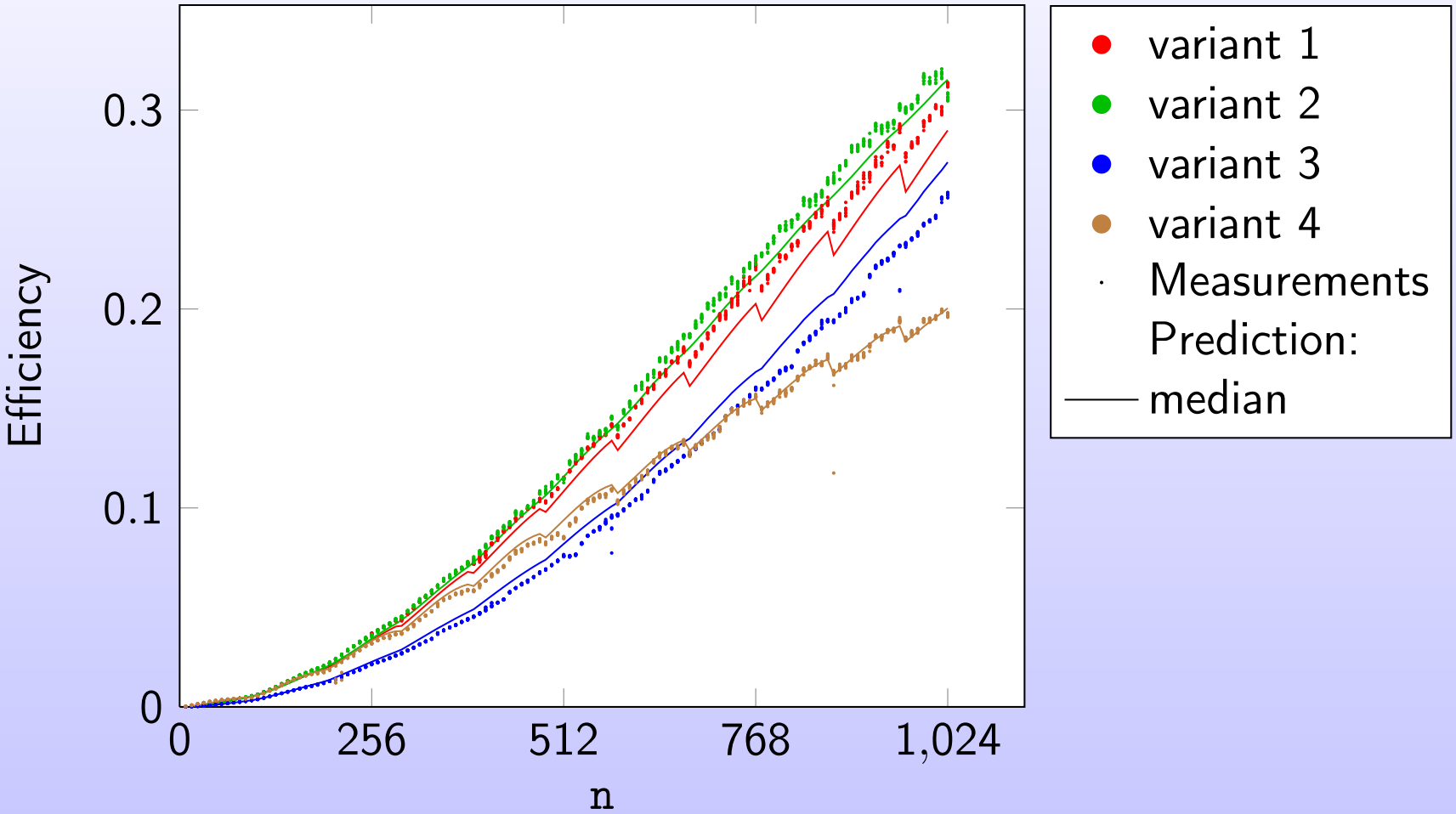
Results: Sandy Bridge

Different architecture, different ranking



Results: Ranking

Different architecture, different ranking



Sylvester equation – 16 variants

$AX + XB = C$

Sylvester equation – 16 variants

$AX + XB = C$

Variant	Efficiency predicted
Var-1	27.03%
Var-2	22.52%
Var-5	15.51%
Var-6	13.72%
Var-16	1.79%
Var-3	1.52%
Var-4	1.50%
Var-8	1.49%
Var-10	1.43%
Var-15	1.43%
Var-9	1.40%
Var-14	1.34%
Var-12	1.29%
Var-7	1.06%
Var-11	1.04%
Var-13	1.01%

Sylvester equation – 16 variants

$AX + XB = C$

Variant	Efficiency predicted	Efficiency measured
Var-1	27.03%	24.04%
Var-2	22.52%	21.07%
Var-5	15.51%	18.82%
Var-6	13.72%	18.51%
Var-16	1.79%	2.21%
Var-3	1.52%	1.52%
Var-4	1.50%	1.45%
Var-8	1.49%	1.37%
Var-10	1.43%	1.53%
Var-15	1.43%	1.52%
Var-9	1.40%	1.48%
Var-14	1.34%	1.33%
Var-12	1.29%	1.43%
Var-7	1.06%	1.16%
Var-11	1.04%	1.07%
Var-13	1.01%	1.01%

Sylvester equation – 16 variants

$AX + XB = C$

Variant	Efficiency predicted	Efficiency measured
Var-1	27.03%	24.04%
Var-2	22.52%	21.07%
Var-5	15.51%	18.82%
Var-6	13.72%	18.51%
Var-16	1.79%	2.21%
Var-3	1.52%	1.52%
Var-4	1.50%	1.45%
Var-8	1.49%	1.37%
Var-10	1.43%	1.53%
Var-15	1.43%	1.52%
Var-9	1.40%	1.48%
Var-14	1.34%	1.33%
Var-12	1.29%	1.43%
Var-7	1.06%	1.16%
Var-11	1.04%	1.07%
Var-13	1.01%	1.01%

Modelling through sampling

Wishlist

Modelling through sampling

Wishlist

- Speed ✓
  - No direct execution of the algorithm ✓
  - Possibly no execution at all ✖

Modelling through sampling

Wishlist

- Speed ✓
  - No direct execution of the algorithm ✓
  - Possibly no execution at all ✖
- Accuracy ✓ accurate ranking!

Modelling through sampling

Wishlist

- Speed ✓
  - No direct execution of the algorithm ✓
  - Possibly no execution at all ✖
- Accuracy ✓ accurate ranking!
- Automation ✓

Conclusions

Generation

- Formal methods
- Decomposition

Analysis

- Analytic modelling
- Sample-based modelling

Conclusions

Generation

- Formal methods
- Decomposition

Analysis

- Analytic modelling
- Sample-based modelling

Future work

- Combine generation and analysis
- Extension: distributed memory, ooc
- Extension: tensors
- Beyond dense linear algebra?
- Stability analysis?