A Compiler for Linear Algebra Operations

Paolo Bientinesi
In collaboration with Diego Fabregat

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

CScADS Autotuning Workshop 2012
August 13-14, 2012
Snowbird, Utah

```
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 );
}
```

```
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
```

$$M := L^{-1} A L^{-T}$$

$$M := L^{-1} A L^{-T}$$

```
>> M = inv(L) * A * inv(L');
```

$$M := L^{-1} A L^{-T}$$

```
>> M = inv(L) * A * inv(L');
>> M == M'
    ans =
       1   1   0   0   1
       1   1   0   0   0
       0   0   0   0   0
       0   0   0   0   0
       1   0   0   0   0
```

$$M := L^{-1} A L^{-T}$$

```
>> M = L \ A / L';
```

$$M := L^{-1} A L^{-T}$$

```
>> M = L \ A / L';
>> M == M'
    ans =
      1   0   0   1   0
      0   1   0   0   0
      0   0   0   0   0
      1   0   0   0   0
      0   0   0   0   1
```

$$M := L^{-1} A L^{-T}$$

```
>> M = L \ A / L';

>> M = tril(M,-1)' + tril(M,-1) +
       diag(real(diag(M)));
```

$$M := L^{-1} A L^{-T}$$

```
>> M = L \ A / L';
>> M = tril(M,-1)' + tril(M,-1) +
       diag(real(diag(M)));
>> M == M'
    ans =
       1   1   1   1   1
       1   1   1   1   1
       1   1   1   1   1
       1   1   1   1   1
       1   1   1   1   1
```

Domain-specific Compiler

Input:

```
M := L^-1 * A * L^-T;
Input(A, L);
Output(M);
Hermitian(A);
LowerTriangular(L);
```

Domain-specific Compiler

Input:

```
M := L^-1 * A * L^-T;
Input(A, L);
Output(M);
Hermitian(A);
LowerTriangular(L);
```

Objectives

- Express M in terms of available building blocks (BLAS, LAPACK, . . .)

Domain-specific Compiler

Input:

```
M := L^-1 * A * L^-T;
Input(A, L);
Output(M);
Hermitian(A);
LowerTriangular(L);
```

Objectives

- Express M in terms of available building blocks (BLAS, LAPACK, ...)
- High-performance

Domain-specific Compiler

Input:

```
M := L^-1 * A * L^-T;
Input(A, L);
Output(M);
Hermitian(A);
LowerTriangular(L);
```

Objectives

- Express M in terms of available building blocks (BLAS, LAPACK, ...)
- High-performance
- Exploit problem-specific knowledge

Domain-specific Compiler

Input:

```
M := L^-1 * A * L^-T;
Input(A, L);
Output(M);
Hermitian(A);
LowerTriangular(L);
```

Objectives

- Express M in terms of available building blocks (BLAS, LAPACK, . . .)
- High-performance
- Exploit problem-specific knowledge
- One vs. many algorithms

What this is talk is **NOT** about

$$\{LU = A,\ A, L, U \in \mathbb{R}^{n \times n},\ \texttt{LowerTriUni}(L),\ \texttt{UpperTri}(U)\} \Rightarrow$$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

While $size(A_{TL}) < size(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

$A_{01} := L_{00}^{-1} A_{01}$

$A_{10} := A_{10} U_{00}^{-1}$

$A_{11} := \mathsf{LU}(A_{11} - A_{10} A_{01})$

Continue

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

What this is talk is **NOT** about

$$\{LU = A,\ A, L, U \in \mathbb{R}^{n \times n},\ \texttt{LowerTriUni}(L),\ \texttt{UpperTri}(U)\} \Rightarrow$$

Partition $A \rightarrow \left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right)$

where A_{TL} is 0×0

While $size(A_{TL}) < size(A)$ **do**

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

$A_{01} := L_{00}^{-1} A_{01}$

$A_{10} := A_{10} U_{00}^{-1}$

$A_{11} := \mathsf{LU}(A_{11} - A_{10} A_{01})$

Continue

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right)$$

endwhile

Target operations

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

Target operations

- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
- $\dfrac{d\left(Ax = b\right)}{dv} = ?$

Target operations

- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
- $\dfrac{d\,(Ax = b)}{dv} = ?$
- $\begin{cases} b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y \\ M := h\Phi + (1-h)I \end{cases}$

Target operations

- $b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y$
- $\dfrac{d\,(Ax = b)}{dv} = ?$
- $\begin{cases} b := \left(X^T M^{-1} X\right)^{-1} X^T M^{-1} y \\ M := h\Phi + (1-h)I \end{cases}$
- $Ax_i = y_i \quad$ with $\quad i = 1 \ldots n$

Example

Linear regression with non-independent outcomes

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

Genome-wide association study

Inputs

- $M \in \mathcal{R}^{n \times n}, \quad SPD(M), \quad n \in [10^3, \ldots, 10^4]$
- $X \in \mathcal{R}^{n \times p}, \quad p \in [1, \ldots, 20]$
- $y \in \mathcal{R}^n$

Output

- $b \in \mathcal{R}^p$

⋆To be repeated thousands of times⋆

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

- Symbolic system (Mathematica)

CLAK

Domain-specific compiler

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

CLAK

Domain-specific compiler

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

CLAK

Domain-specific compiler

- 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
  $(R^T Q^T Q R)^{-1} R^T Q^T \quad \Rightarrow \quad R^{-1} Q^T$

Sequences

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` $j = 1 : t$

    $LL^T = M_j$

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

    $QR = X$

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

    $b \leftarrow Q^T y$

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

Sequences

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
```

Sequences

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
```

Sequences

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
```

Sequences

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
```

GWAS

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

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

Cost analysis

Cost analysis

Flop count

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

Cost analysis

Sample-based prediction

- Sample the kernels – not the algorithm!

Cost analysis

Sample-based prediction

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

Cost analysis

Sample-based prediction

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

Cost analysis

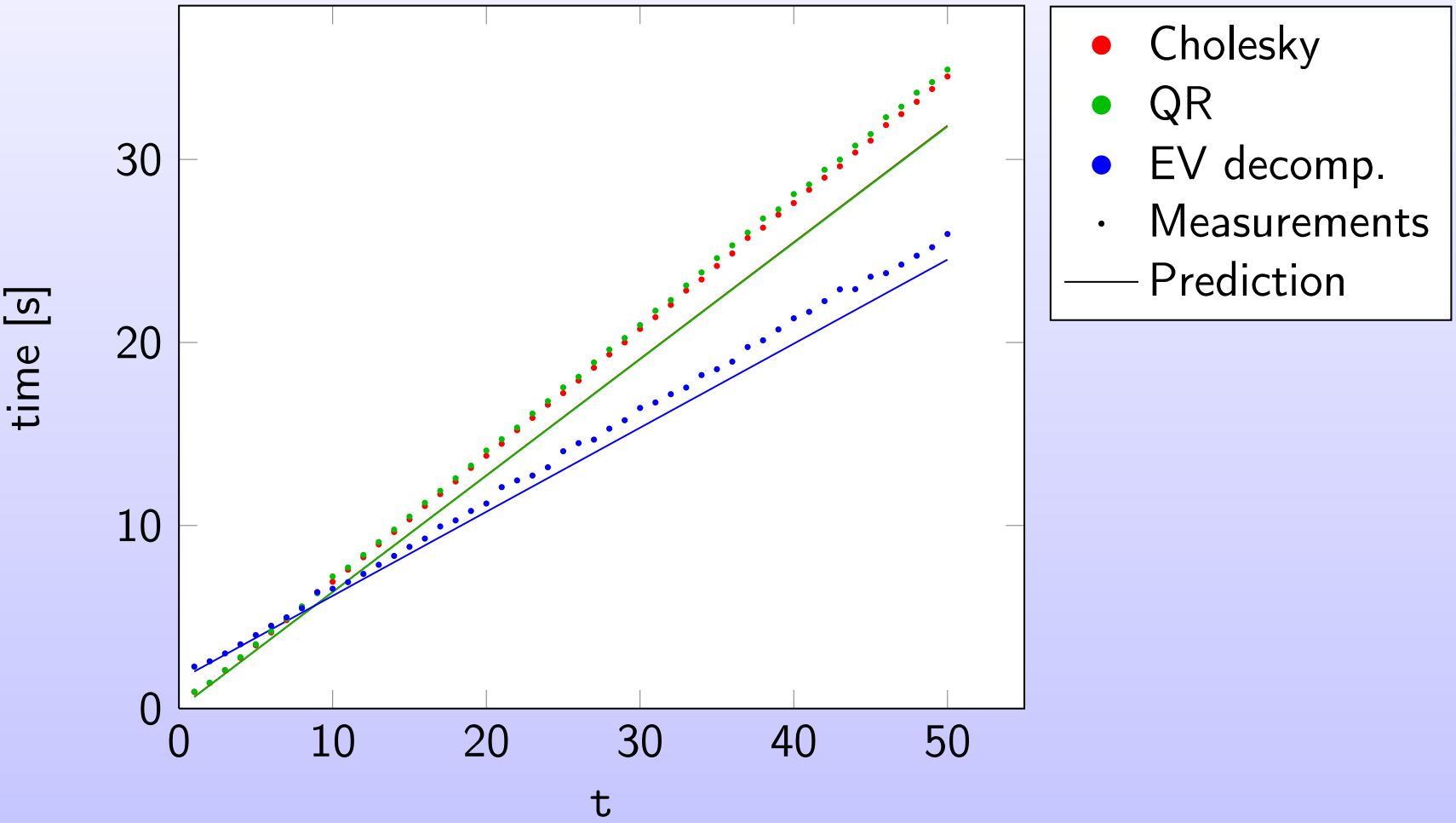
Sample-based prediction

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

Performance

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

$n = 5000 \quad p = 4 \quad m = 3000$



Conclusions & future work

CLAK

- Linear algebra compiler: algorithm & code generator

Conclusions & future work

CLAK

- Linear algebra compiler:
  algorithm & code generator
- Domain specific knowledge:
  1) encoded + 2) inferred

Conclusions & future work

CLAK

- Linear algebra compiler:
  algorithm & code generator
- Domain specific knowledge:
  1) encoded + 2) inferred
- Performance: n-fold speedups

Conclusions & future work

CLAK

- Linear algebra compiler:
  algorithm & code generator
- Domain specific knowledge:
  1) encoded + 2) inferred
- Performance: n-fold speedups

TODO list

... we've only started!