Towards the Final Generation of Dense Linear Algebra Libraries

Robert A. van de Geijn
Department of Computer Sciences
The University of Texas at Austin
rvdg@cs.utexas.edu
http://www.cs.utexas.edu/users/flame/

Current FLAME Team

- UT Austin Department of Computer Sciences
  Paolo Bientinesi Maggie Myers Zaiqing Xu
  Ernie Chan Robert van de Geijn
  Tze Meng Low Field Van Zee
- UT Texas Advanced Computing Center
  Kazushige Goto Kent Milfelt
- UT Center for Space Research
  Brian Gunter
- Universidad Jaume I (Spain)
  Enrique Quintana-Orti Gregorio Quintana-Orti
- Industry
  Hewlett-Packard National Instruments IBM
  Intel NEC Solutions (America), Inc

Support

- National Science Foundation
  - Modest Funding through 2006
- Hewlett-Packard
  - Equipment donations
- Unrestricted grants
  - Dr. James Truchard (National Instruments)
  - NEC Solutions (America), Inc

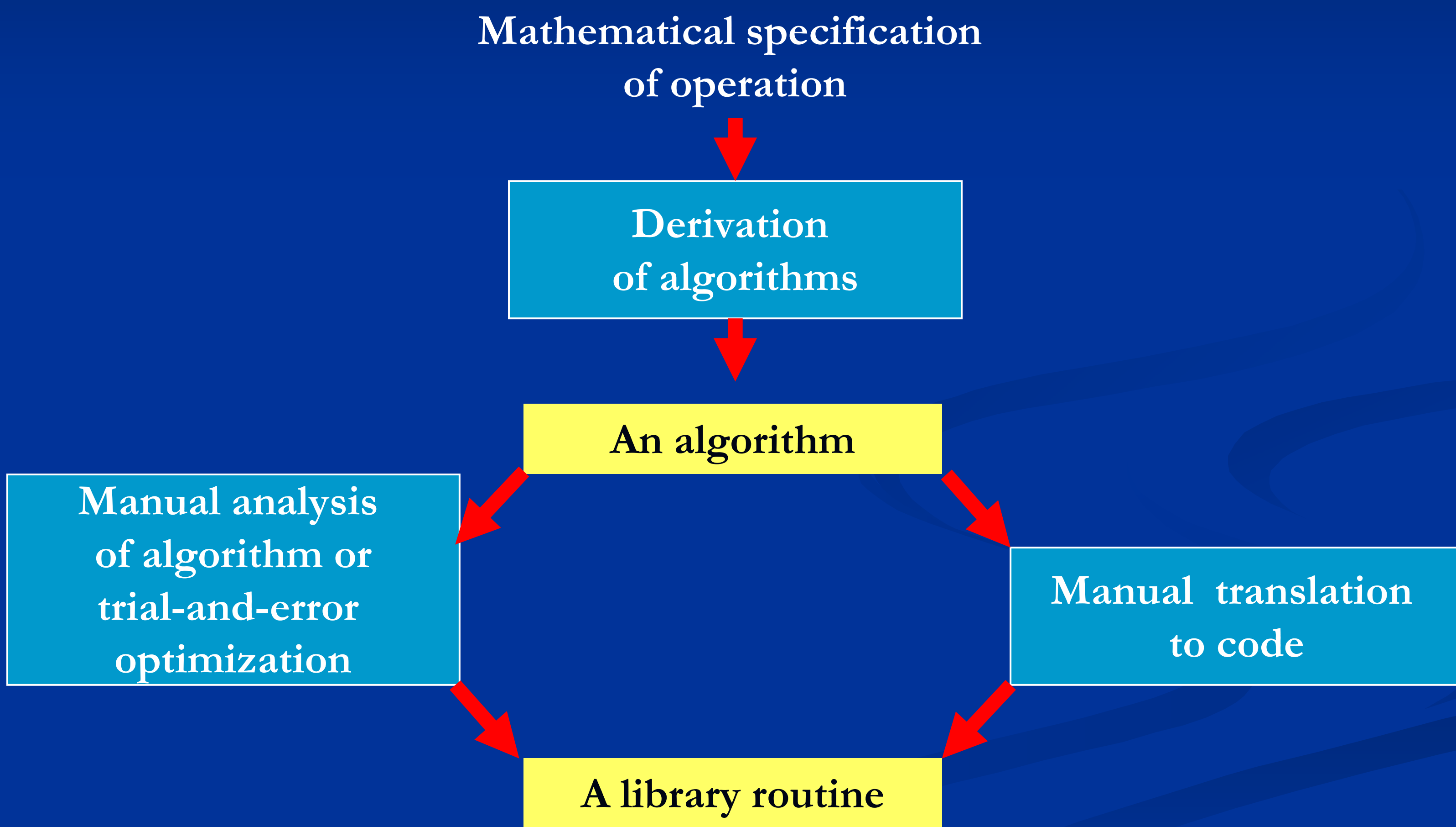
Motivation

- Developing dense linear algebra libraries
  - Traditional approach:
    - Evolve from existing libraries
    - Ask the question: What added functionality is needed?
    - Reactive to current needs
    - Always a “Next Generation” library
  - The Big Question:
    - Can we build a “Final Generation” library?
    - Proactive to (as of yet undefined) future needs

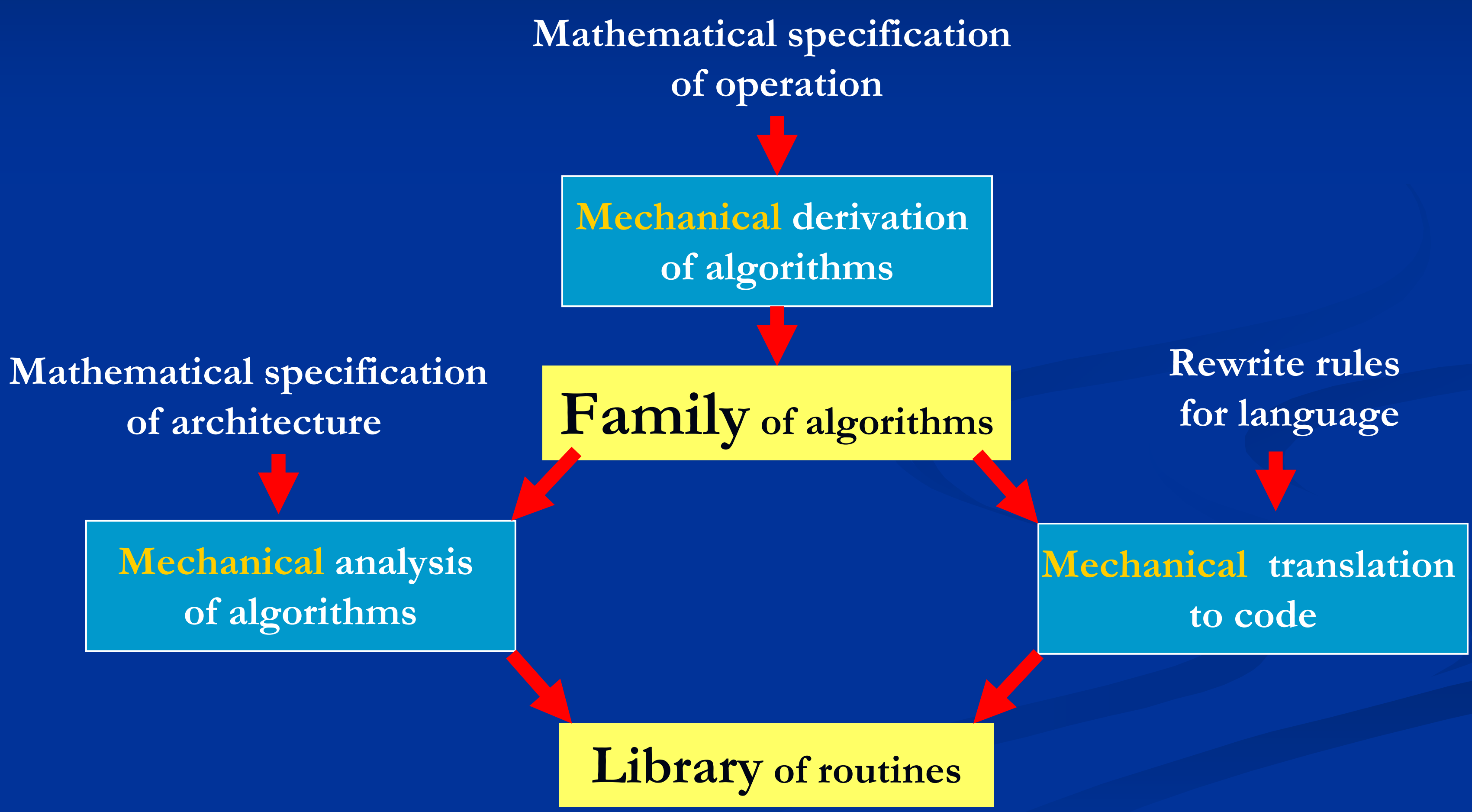
Motivation

- Properties of a Final Generation Library
  - Forward compatible to
    - New architectures
    - New languages
    - New datastructures
    - New operations

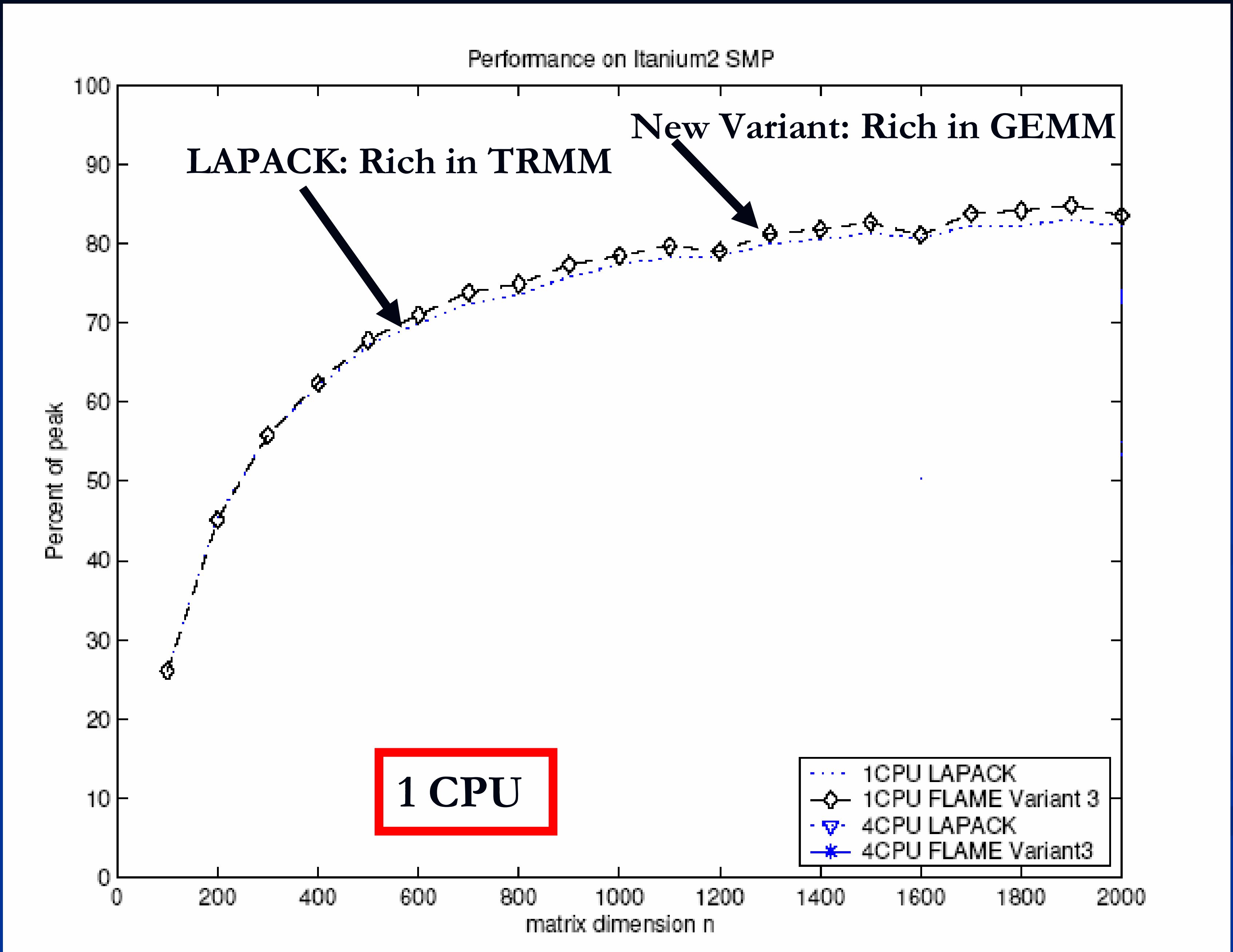
What do we do by hand?



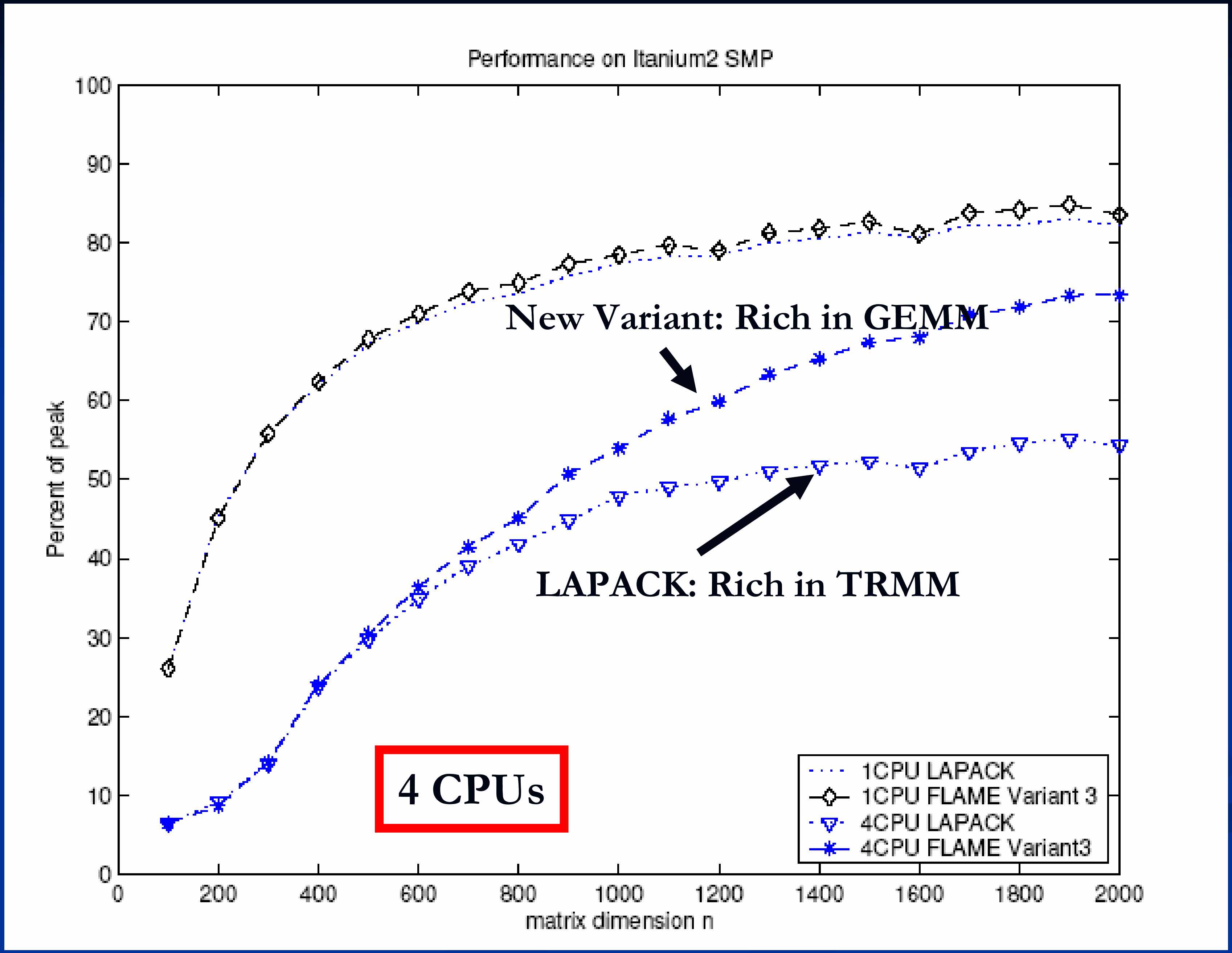
The Final Generation



Why a family of algorithms?



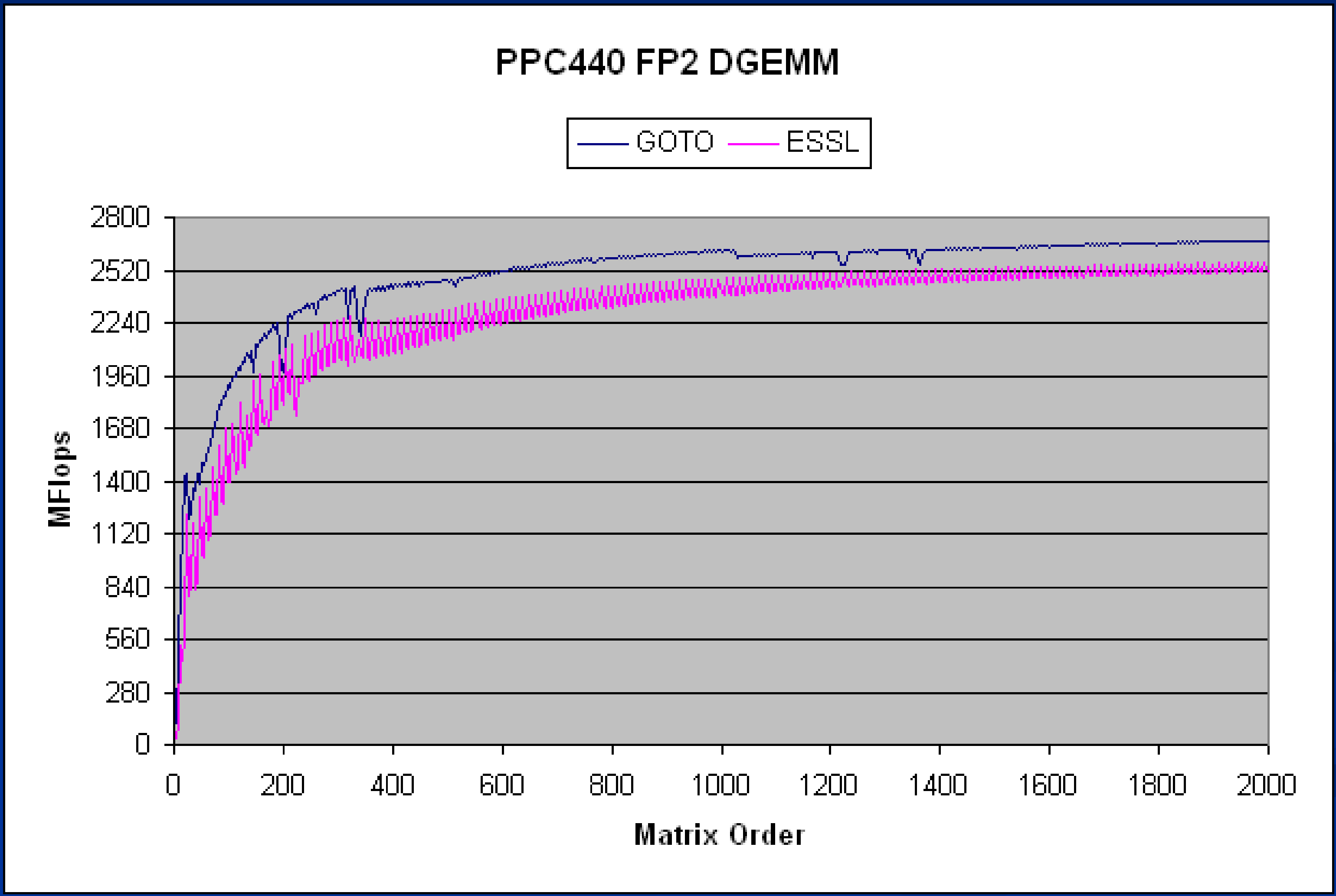
Performance on Itanium2 SMP
New Variant: Rich in GEMM
LAPACK: Rich in TRMM
1 CPU
1CPU LAPACK
1CPU FLAME Variant 3
4CPU LAPACK
4CPU FLAME Variant3
Percent of peak
matrix dimension n
0
10
20
30
40
50
60
70
80
90
100
0
200
400
600
800
1000
1200
1400
1600
1800
2000


Performance on Itanium2 SMP
Percent of peak
matrix dimension n
0
10
20
30
40
50
60
70
80
90
100
0
200
400
600
800
1000
1200
1400
1600
1800
2000
New Variant: Rich in GEMM
LAPACK: Rich in TRMM
4 CPUs
1CPU LAPACK
1CPU FLAME Variant 3
4CPU LAPACK
4CPU FLAME Variant3

A Note on Performance

- The FLAME team is recognized for performance:
  - GOTO BLAS by Kazushige Goto:
    - Fastest for essentially all platforms
  - FLAME:
    - Outperforms LAPACK
  - PLAPACK:
    - Outperforms ScaLAPACK for all major operations
- I will show few performance graph

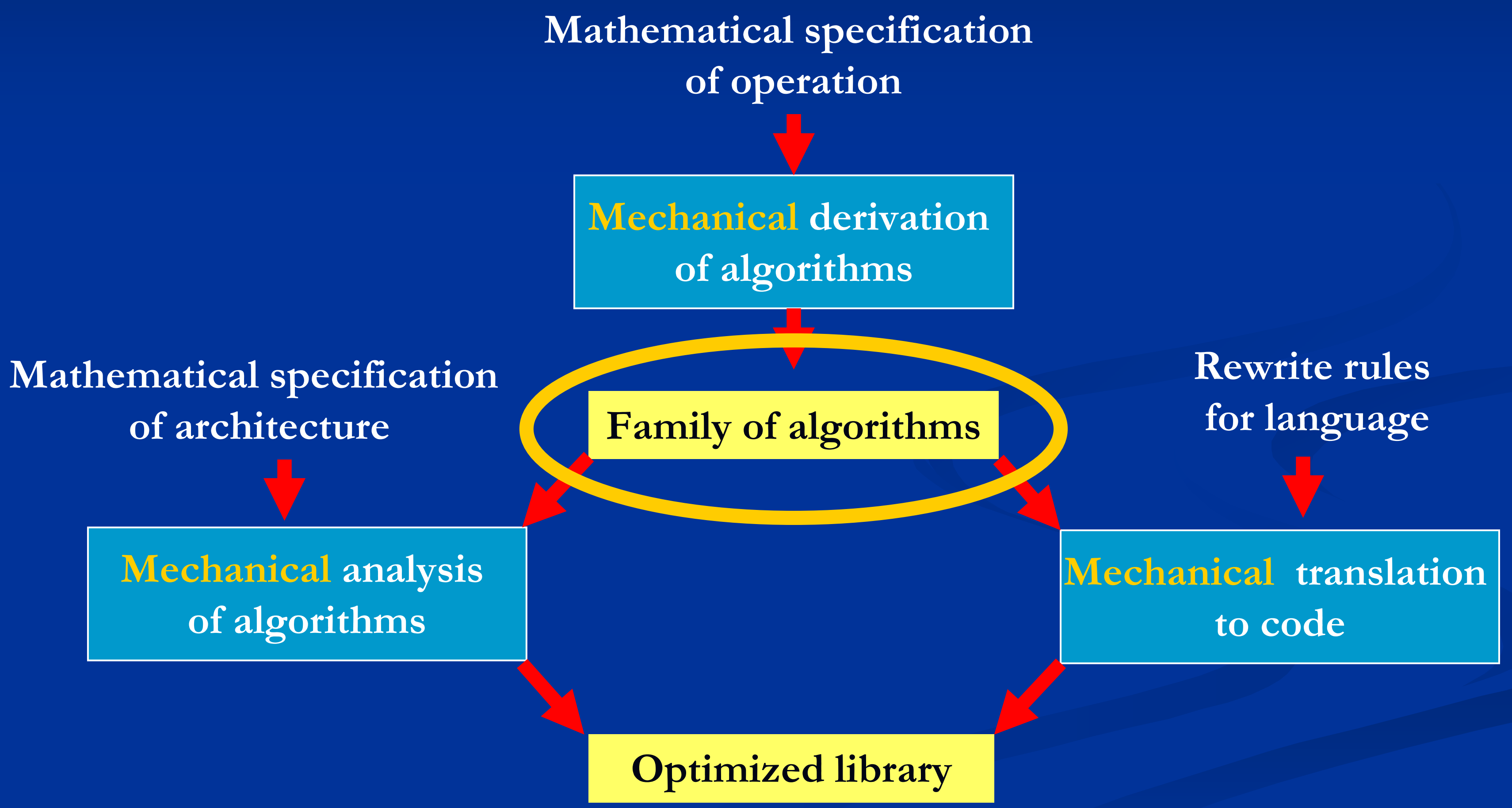
Kazushige Goto's BLAS



Overview

- New Notation for Expressing Algorithms
- APIs for Representing Algorithms in Code
- Mechanical Derivation of Algorithms
- Mechanical Analysis of Algorithms
- Addressing Future Challenges
- Conclusion

The Final Generation

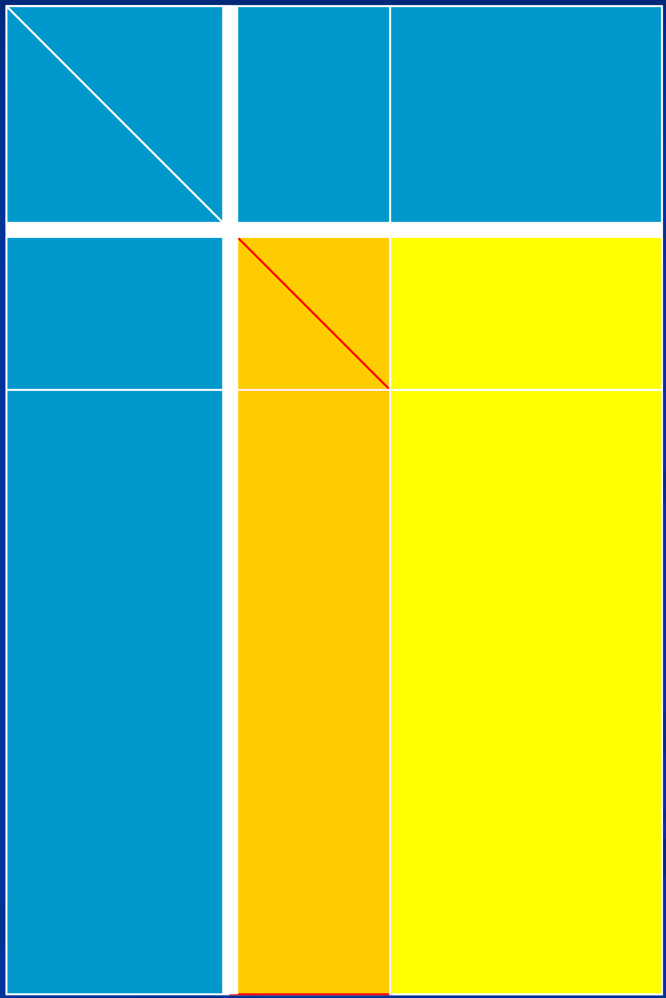


Step 1:

Change the Notation for Expressing Algorithms

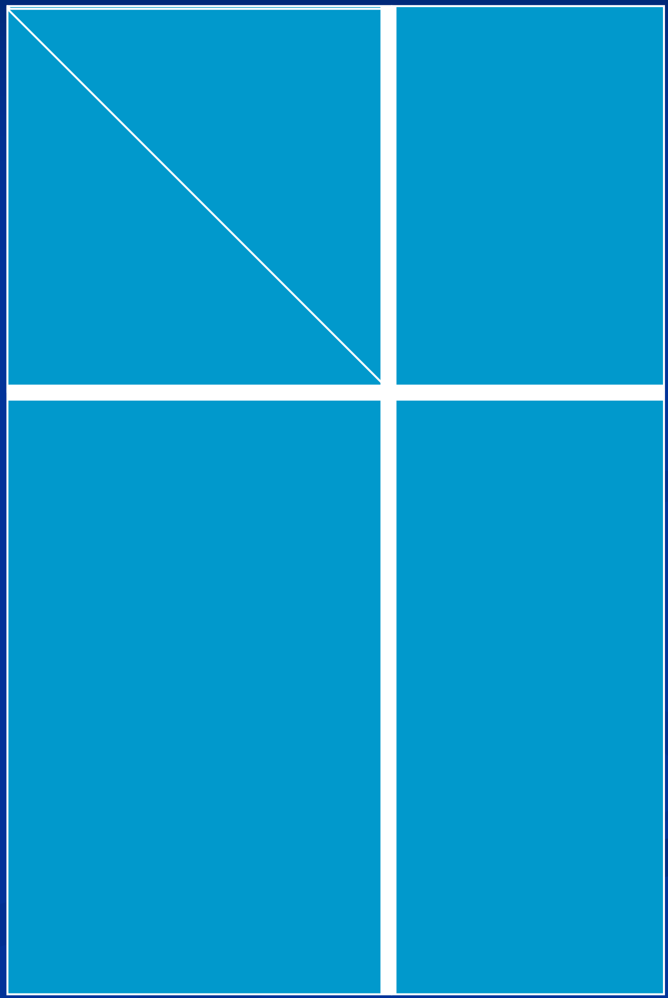
Example: QR factorization via Householder Transformations

- Blocked Algorithm:
  - Factor current panel
  - Form compact WY transform
  - Update rest of matrix

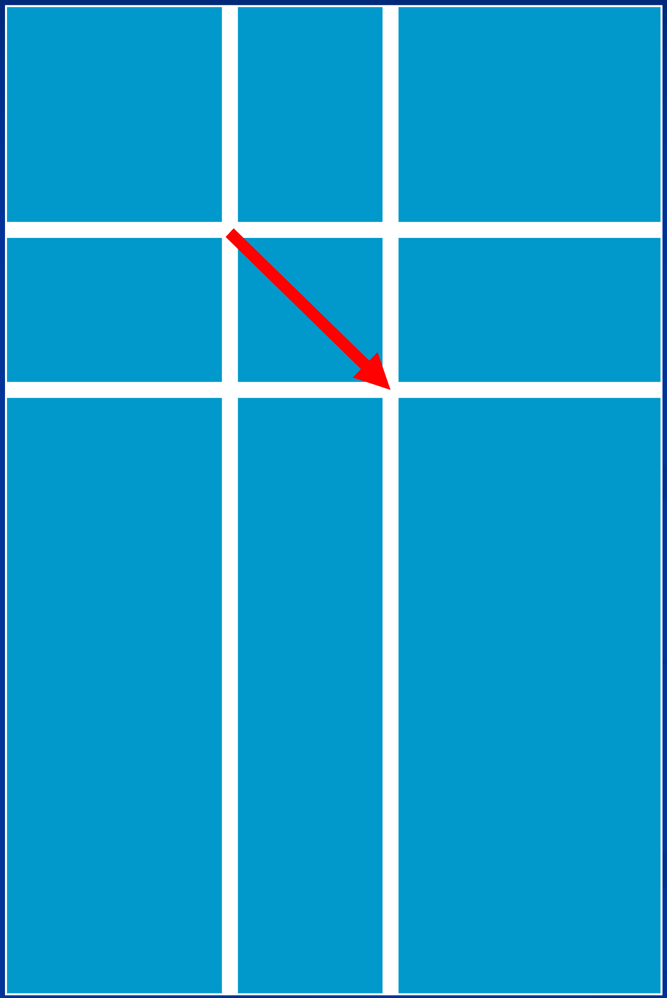


QR factorization via Householder transformations

- Blocked Algorithm:
  - Factor current panel
  - Form compact WY transform
  - Update rest of matrix
  - Move forward

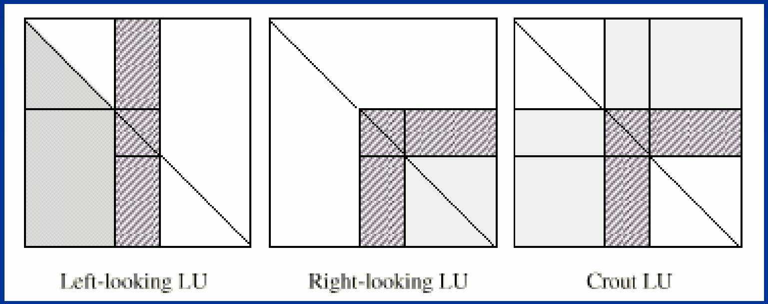


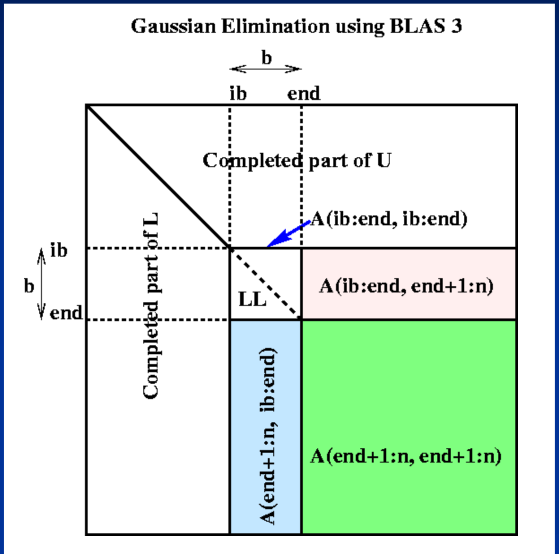
Capturing movement through matrices



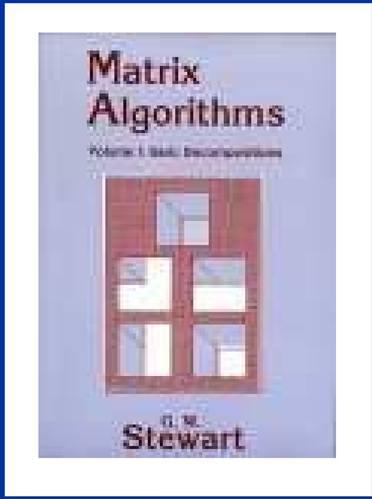
This picture has been around:

- In a typical talk on LAPACK:





- Pete Stewart's recent book:



Can the picture be the algorithm?

Algorithm: $[A, s] := \text{QRBLK}(A)$

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

where A_{TL} is 0×0 and s_T has 0 elements

while $n(A_{BR}) \neq 0$ **do**

Determine block size k

Repartition

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \rightarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} s_T \\ \hline s_B \end{array}\right) \rightarrow \left(\begin{array}{c} s_0 \\ \hline s_1 \\ \hline s_2 \end{array}\right)$$

where A_{11} **is** $k \times k$ **and** s_1 **has** k **elements**

$$\left[\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right), s_1\right] := \left[\left(\begin{array}{c} \{U \backslash R\}_{11} \\ \hline U_{21} \end{array}\right), s_1\right] = \text{QRUNB}\left(\left(\begin{array}{c} A_{11} \\ \hline A_{21} \end{array}\right)\right)$$

Compute S_1 from $\left[\left(\begin{array}{c} U_{11} \\ \hline U_{21} \end{array}\right), s_1\right]$

Update

$$\left(\begin{array}{c} A_{12} \\ \hline A_{22} \end{array}\right) := \left(I + \left(\begin{array}{c} U_{11} \\ \hline U_{21} \end{array}\right) S_1 \left(\begin{array}{c} U_{11} \\ U_{21} \end{array}\right)^T\right)^T \left(\begin{array}{c} A_{12} \\ \hline A_{22} \end{array}\right)$$

Continue with

$$\left(\begin{array}{c|c} A_{TL} & A_{TR} \\ \hline A_{BL} & A_{BR} \end{array}\right) \leftarrow \left(\begin{array}{c|c|c} A_{00} & A_{01} & A_{02} \\ \hline A_{10} & A_{11} & A_{12} \\ \hline A_{20} & A_{21} & A_{22} \end{array}\right) \text{ and } \left(\begin{array}{c} s_T \\ \hline s_B \end{array}\right) \leftarrow \left(\begin{array}{c} s_0 \\ \hline s_1 \\ \hline s_2 \end{array}\right)$$

endwhile

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 $n(A_{BR}) \neq 0$ **do**

Determine block size b

Repartition

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

where A_{11} **is** $b \times b$

$$\left[\left(\frac{A_{11}}{A_{21}}\right), s_1\right] := \left[\left(\frac{\{U \backslash R\}_{11}}{U_{21}}\right), s_1\right] = \mathrm{QR}\left(\frac{A_{11}}{A_{21}}\right)$$

Compute S_1 from $\left[\left(\frac{U_{11}}{U_{21}}\right), s_1\right]$

Update

$$\left(\frac{A_{12}}{A_{22}}\right) := \left(I + \left(\frac{U_{11}}{U_{21}}\right) S_1 \left(\begin{array}{c} U_{11} \\ U_{21} \end{array}\right)^T\right)^T \left(\frac{A_{12}}{A_{22}}\right)$$

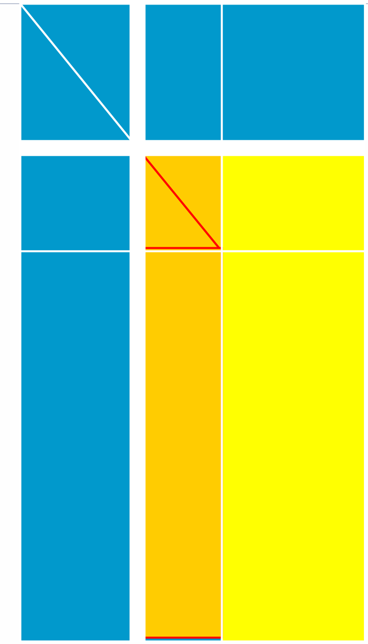
Continue with

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

endwhile

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 $n(A_{BR}) \neq 0$ **do**

Determine block size b

Repartition

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

where A_{11} is $b \times b$

$$\left[\left(\frac{A_{11}}{A_{21}}\right), s_1\right] := \left[\left(\frac{\{U \backslash R\}_{11}}{U_{21}}\right), s_1\right] = \mathrm{QR}\left(\frac{A_{11}}{A_{21}}\right)$$

Compute S_1 from $\left[\left(\frac{U_{11}}{U_{21}}\right), s_1\right]$

Update

$$\left(\frac{A_{12}}{A_{22}}\right) := \left(I + \left(\frac{U_{11}}{U_{21}}\right) S_1 \left(\begin{array}{c} U_{11} \\ U_{21} \end{array}\right)^T\right)^T \left(\frac{A_{12}}{A_{22}}\right)$$

Continue with

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

endwhile

Algorithm: $[A, t] := \mathrm{QR}(A)$

```
[ ATL, ATR, ...
  ABL, ABR ] = FLA_Part_2x2( A, 0, 0, 'FLA_TL' );

while ( size( ATL, 2 ) ~= size( A, 2 ) )
  b = min( size( ABR, 1 ), nb_alg );
  [ A00, A01, A02, ...
    A10, A11, A12, ...
    A20, A21, A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ...
                                              ABL, ABR, ...
  [ U1, s1 ] = QR_unb_var1( [ A11                b, b, 'FLA_BR' );
                              A21 ], s1 );
  [ A11, ...
    A21 ] = FLA_Part_2x1( U1, b, 'FLA_TOP' );
  S1  = Accum_S( U1, s1 );
  U11 = trilu( A11 );
  U21 = A21;
  W12 = S1' * ( U11' * A12 + U21' * A22 )
  A12 = A12 - U11 * W12;
  A22 = A22 - U21 * W12;
  [ ATL, ATR, ...
    ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00, A01, A02, ...
                                           A10, A11, A12, ...
                                           A20, A21, A22, 'FLA_TL' );
end
```

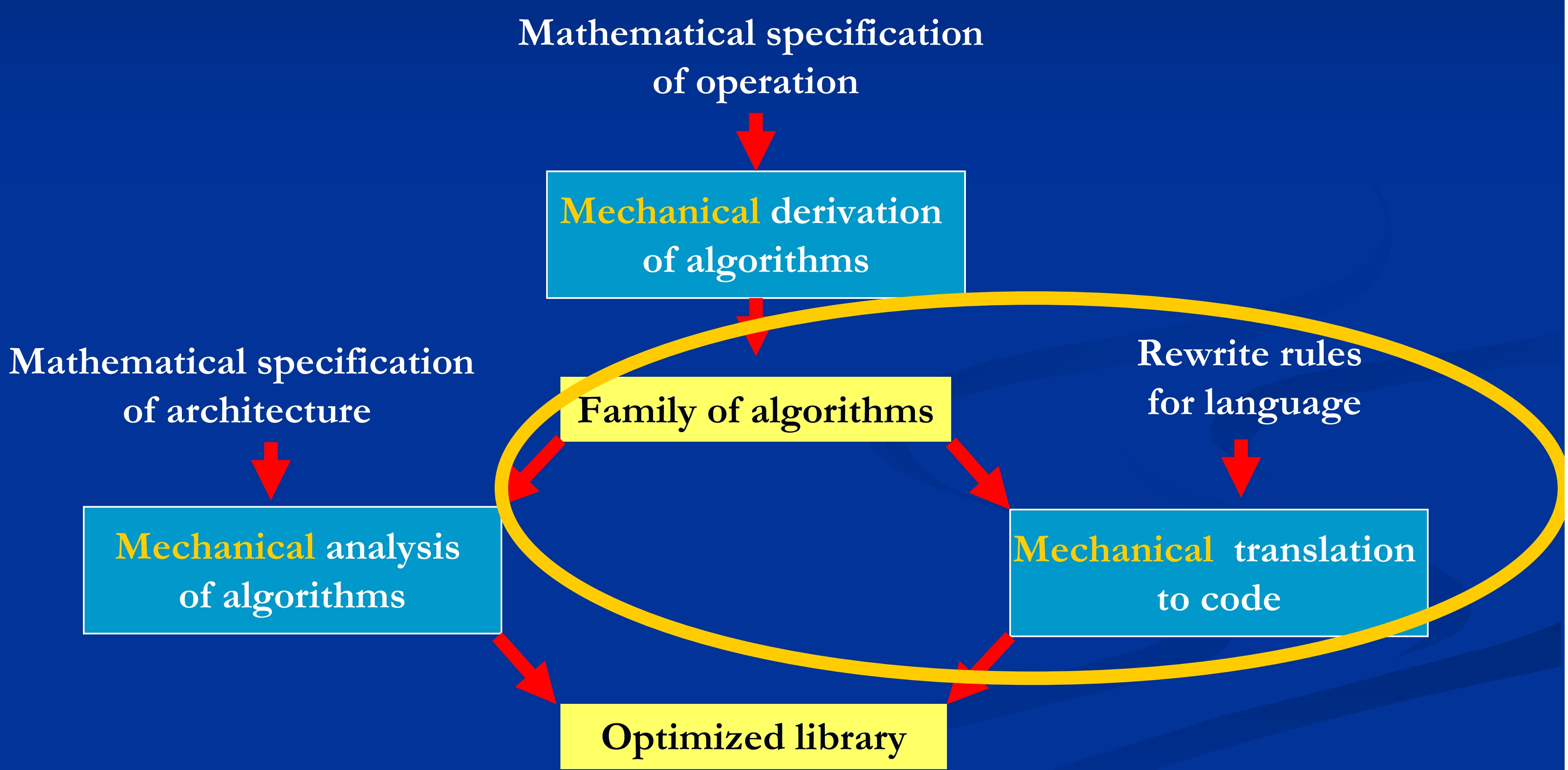
Update

$$\left(\frac{A_{12}}{A_{22}}\right) := \left(I + \left(\frac{U_{11}}{U_{21}}\right) S_1 \left(\begin{array}{c} U_{11} \\ U_{21} \end{array}\right)^T\right)^T \left(\frac{A_{12}}{A_{22}}\right)$$

Step 2:

APIs for Representing Algorithms in Code

The Final Generation



LAPACK API

```
      DO 10 I = 1, K - NX, NB
         IB = MIN( K-I-1, NB )
*
*        Compute the QR factorization of the current block
*        A(i:m,i:i+ib-1)
*
         CALL DGEQR2( M-I-1, IB, A( I, I ), LDA, TAU( I ), WORK,
     $                IINFO )
         IF( I+IB.LE.N ) THEN
*
*           Form the triangular factor of the block reflector
*           H = H(i) H(i+1) . . . H(i+ib-1)
*
            CALL DLARFT( 'Forward', 'Columnwise', M-I-1, IB,
     $                   A( I, I ), LDA, TAU( I ), WORK, LDWORK )
*
*           Apply H' to A(i:m,i+ib:n) from the left
*
            CALL DLARFB( 'Left', 'Transpose', 'Forward',
     $                   'Columnwise', M-I-1, N-I-IB-1, IB,
     $                   A( I, I ), LDA, WORK, LDWORK, A( I, I+IB ),
     $                   LDA, WORK( IB+1 ), LDWORK )
         END IF
   10 CONTINUE
```

```
      DO 10 I = 1, K - NX, NB
         IB = MIN( K-I-1, NB )
*
*        Compute the QR factorization of the current block
*        A(i:m,i:i+ib-1)
*
         CALL DGEQR2( M-I-1, IB, A( I, I ), LDA, TAU( I ), WORK,
     $                IINFO )
         IF( I+IB.LE.N ) THEN
*
*           Form the triangular factor of the block reflector
*           H = H(i) H(i+1) . . . H(i+ib-1)
*
            CALL DLARFT( 'Forward', 'Columnwise', M-I-1, IB,
     $                   A( I, I ), LDA, TAU( I ), WORK, LDWORK )
*
*           Apply H' to A(i:m,i+ib:n) from the left
*
            CALL DLARFB( 'Left', 'Transpose', 'Forward',
     $                   'Columnwise', M-I-1, N-I-IB-1, IB,
     $                   A( I, I ), LDA, WORK, LDWORK, A( I, I+IB ),
     $                   LDA, WORK( IB+1 ), LDWORK )
         END IF
   10 CONTINUE
```

Warning: I introduced an error!

Step 2:

New APIs that capture the algorithm in code

FLAME@lab

(FLAME/MATLAB API)

```
[ ATL, ATR, ...
  ABL, ABR ] = FLA_Part_2x2( A, 0, 0, 'FLA_TL' );

while ( size( ATL, 2 ) ~= size( A, 2 ) )
  b = min( size( ABR, 1 ), nb_alg );

  [ A00, A01, A02, ...
    A10, A11, A12, ...
    A20, A21, A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ...
                                             ABL, ABR, ...
                                             b, b, 'FLA_BR' );
  ------------------------------------------------------------
  [ U1, s1 ] = QR_unb_var1( [ A11
                              A21 ], s1 );
  [ A11, ...
    A21 ] = FLA_Part_2x1( U1, b, 'FLA_TOP' );
  S1  = Accum_S( U1, s1 );
  U11 = trilu( A11 );
  U21 = A21;
  W12 = S1' * ( U11' * A12 + U21' * A22 )
  A12 = A12 - U11 * W12;
  A22 = A22 - U21 * W12;
  ------------------------------------------------------------
  [ ATL, ATR, ...
    ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00, A01, A02, ...
                                           A10, A11, A12, ...
                                           A20, A21, A22, 'FLA_TL' );
end
```

Update

$$\left(\frac{A_{12}}{A_{22}}\right) := \left(I + \left(\frac{U_{11}}{U_{21}}\right) S_1 \begin{pmatrix} U_{11} \\ U_{21} \end{pmatrix}^T\right)^T \left(\frac{A_{12}}{A_{22}}\right)$$

```
[ ATL, ATR, ...
  ABL, ABR ] = FLA_Part_2x2( A, 0, 0, 'FLA_TL' );
[ sT, ...
  sB ] = FLA_Part_2x1( s, 0, 'FLA_TOP' );

while ( size( ATL, 2 ) ~= size( A, 2 ) )
  b = min( size( ABR, 1 ), nb_alg );
  [ A00, A01, A02, ...
    A10, A11, A12, ...
    A20, A21, A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ...
                                             ABL, ABR, b, b, 'FLA_BR' );
  [ s0, ...
    s1, ...
    s2 ] = FLA_Repart_2x1_to_3x1( sT, ...
                                  sB, b, 'FLA_BOTTOM' );
%------------------------------------------------------------%
  [ U1, s1 ] = QR_unb_var1( [ A11
                              A21 ], s1 );
  [ A11, ...
    A21 ] = FLA_Part_2x1( U1, b, 'FLA_TOP' );
  S1  = Accum_S( U1, s1 );
                    % update rest of matrix
  U11 = trilu( A11 );
  U21 = A21;
  W12 = S1' * ( U11' * A12 + U21' * A22 );
  A12 = A12 - U11 * W12;
  A22 = A22 - U21 * W12;
%------------------------------------------------------------%
  [ ATL, ATR, ...
    ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00, A01, A02, ...
                                           A10, A11, A12, ...
                                           A20, A21, A22, 'FLA_TL' );
  [ sT, ...
    sB ] = FLA_Cont_with_3x1_to_2x1( s0, ...
                                     s1, ...
                                     s2, 'FLA_TOP' );
end
```

```
[ ATL, ATR, ...
  ABL, ABR ] = FLA_Part_2x2( A, 0, 0, 'FLA_TL' );
[ sT, ...
  sB ] = FLA_Part_2x1( s, 0, 'FLA_TOP' );

while ( size( ATL, 2 ) ~= size( A, 2 ) )
  b = min( size( ABR, 1 ), nb_alg );
  [ A00, A01, A02, ...
    A10, A11, A12, ...
    A20, A21, A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ...
                                             ABL, ABR, b, b, 'FLA_BR' );
  [ s0, ...
    s1, ...
    s2 ] = FLA_Repart_2x1_to_3x1( sT, ...
                                  sB, b, 'FLA_BOTTOM' );
%------------------------------------------------------------%
  [ U1, s1 ] = QR_unb_var1( [ A11
                              A21 ], s1 );
  [ A11, ...
    A21 ] = FLA_Part_2x1( U1, b, 'FLA_TOP' );
  S1  = Accum_S( U1, s1 );
             % Update rest of matrix
  U11 = trilu( A11 );
  U21 = A21;
  W12 = S1' * ( U11' * A12 + U21' * A22 );
  A12 = A12 - U11 * W12;
  A22 = A22 - U21 * W12;
%------------------------------------------------------------%
  [ ATL, ATR, ...
    ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00, A01, A02, ...
                                           A10, A11, A12, ...
                                           A20, A21, A22, 'FLA_TL' );
  [ sT, ...
    sB ] = FLA_Cont_with_3x1_to_2x1( s0, ...
                                     s1, ...
                                     s2, 'FLA_TOP' );
end
```

```
[ ATL, ATR, ...
  ABL, ABR ] = FLA_Part_2x2( A, 0, 0, 'FLA_TL' );

while ( size( ATL, 2 ) ~= size( A, 2 ) )
  b = min( size( ABR, 1 ), nb_alg );
  [ A00, A01, A02, ...
    A10, A11, A12, ...
    A20, A21, A22 ] = FLA_Repart_2x2_to_3x3( ATL, ATR, ...
                                             ABL, ABR, b, b, 'FLA_BR' );
%------------------------------------------------------------%
  [ U1, s1 ] = QR_unb_var1( [ A11
                              A21 ], s1 );
  [ A11, ...
    A21 ] = FLA_Part_2x1( U1, b, 'FLA_TOP' );
  S1  = Accum_S( U1, s1 );
  U11 = trilu( A11 );
  U21 = A21;
  W12 = S1' * ( U11' * A12 + U21' * A22 );
  A12 = A12 - U11 * W12;
  A22 = A22 - U21 * W12;
%------------------------------------------------------------%
  [ ATL, ATR, ...
    ABL, ABR ] = FLA_Cont_with_3x3_to_2x2( A00, A01, A02, ...
                                           A10, A11, A12, ...
                                           A20, A21, A22, 'FLA_TL' );
end
```

update

```
function [ S ] = Accum_S( U, s )

U = triu( U );                  % U = lower unit trapezoidal part of U

s = ones( size( s ) ) ./ s;     % Set each element of s to its inverse

S = inv( triu( U' * U, 1 ) + diag( s ));

return
```


T. Joffrain, T. M. Low, E. Quintana-Orti, R. van de Geijn, and F. Van Zee, On Accumulating Householder Transformations. *TOMS* , to be revised.

X. Sun. Aggregations of elementary transformations. Tech Report. DUKE-TR-1996-03, 1996.

C. Puglisi. Modification of the Householder method based on the compact WY representation. SISC, 18, 723-726, 1992

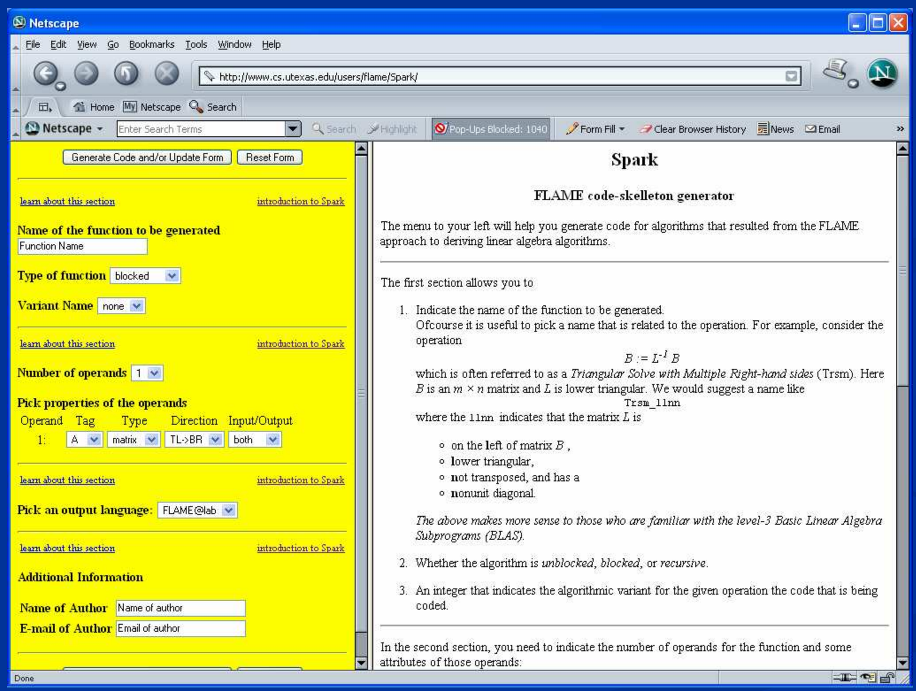
H.F. Walker. Implementation of the GMRES method using Householder transformations. SISC, 9, 1, 152-163, 1988

APIs for C

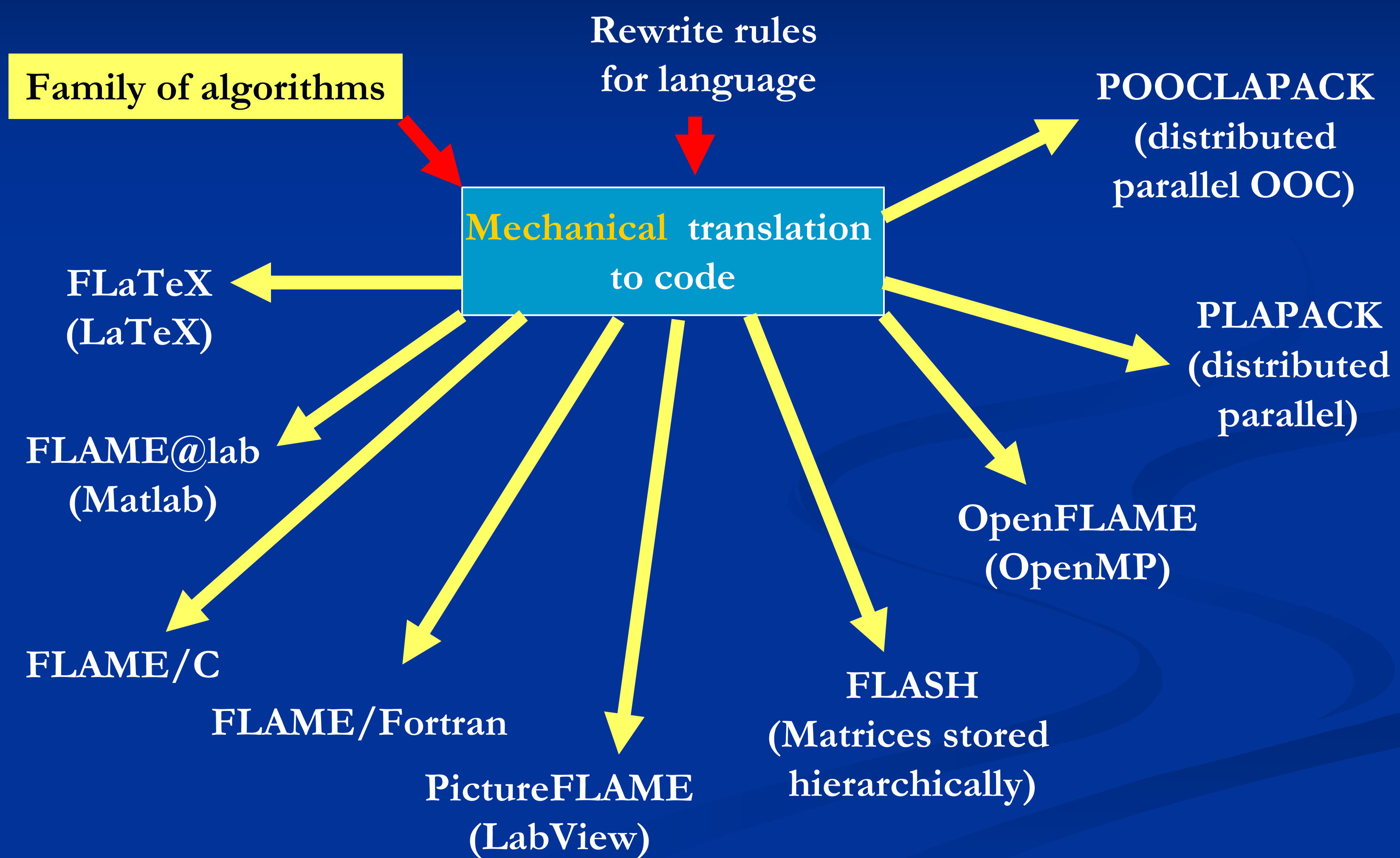
FLAME/C
QR factorization

```
  FLA_Part_2x2( A,    &ATL, &ATR,
                      &ABL, &ABR,     0, 0, FLA_TL );

  while ( FLA_Obj_width ( ATL ) != FLA_Obj_width ( A ) ){
    b = min( FLA_Obj_width( ABR ), nb_alg );

    FLA_Repart_2x2_to_3x3( ATL, /**/ ATR,       &A00, /**/ &A01, &A02,
                        /* ************* */   /* ******************** */
                                                &A10, /**/ &A11, &A12,
                           ABL, /**/ ABR,       &A20, /**/ &A21, &A22,
                           b, b, FLA_BR );
    /*------------------------------------------------------------*/

    FLA_QR_unb( A11,
                A21, s1 );

    FLA_Accum_S( A11,
                 A21, s1, S1 );

    FLA_Apply_blk_transform( FLA_LEFT, FLA_TRANSPOSE, A11, S1, A12,
                                                      A21,    A22 );
/*------------------------------------------------------------*/
    FLA_Cont_with_3x3_to_2x2( &ATL, /**/ &ATR,       A00, A01, /**/ A02,
                                                     A10, A11, /**/ A12,
                            /* ************** */  /* ****************** */
                              &ABL, /**/ &ABR,       A20, A21, /**/ A22,
                              FLA_TL );
  }
```

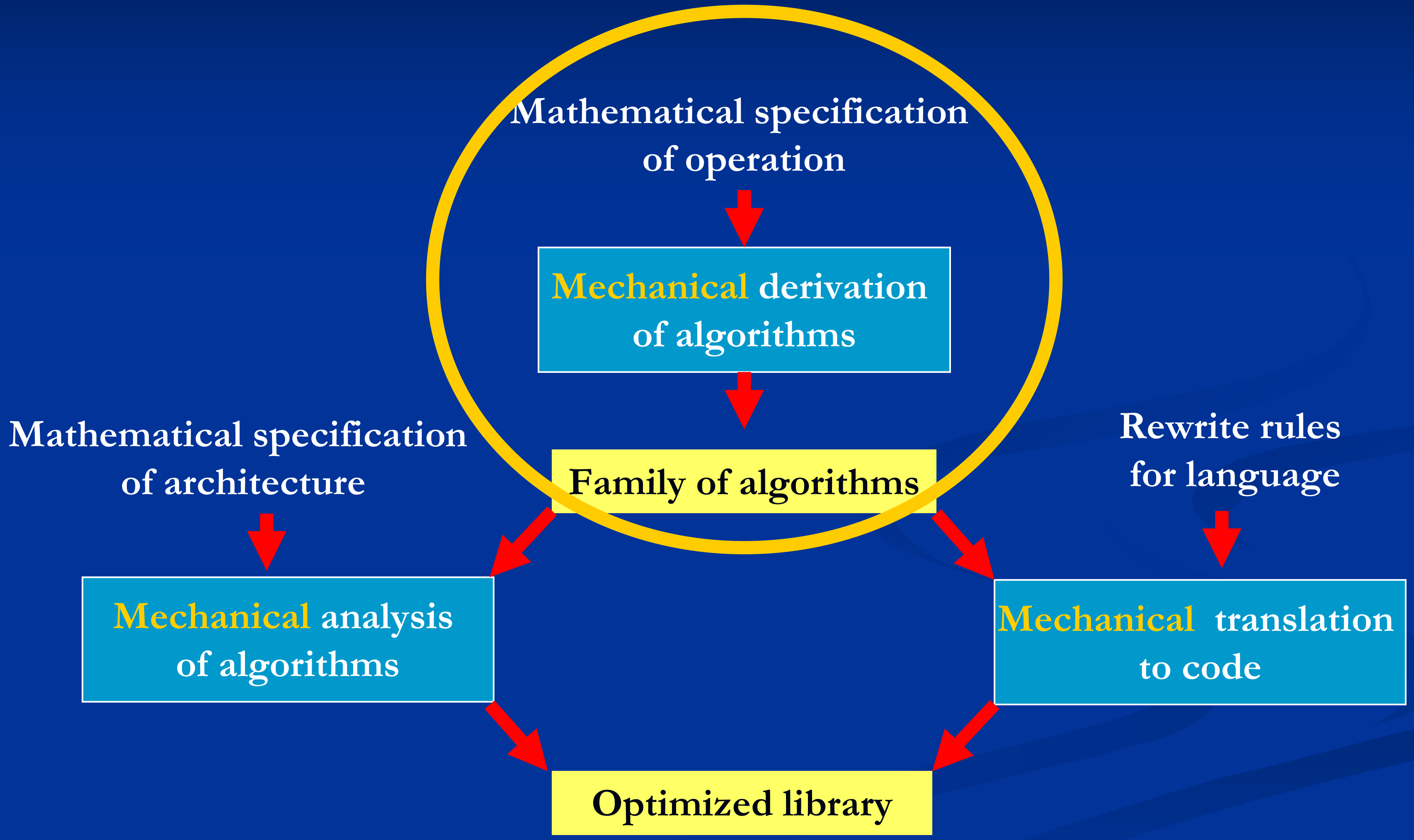
Spark: A Tool for Generating Representations



The Final Generation



The Final Generation



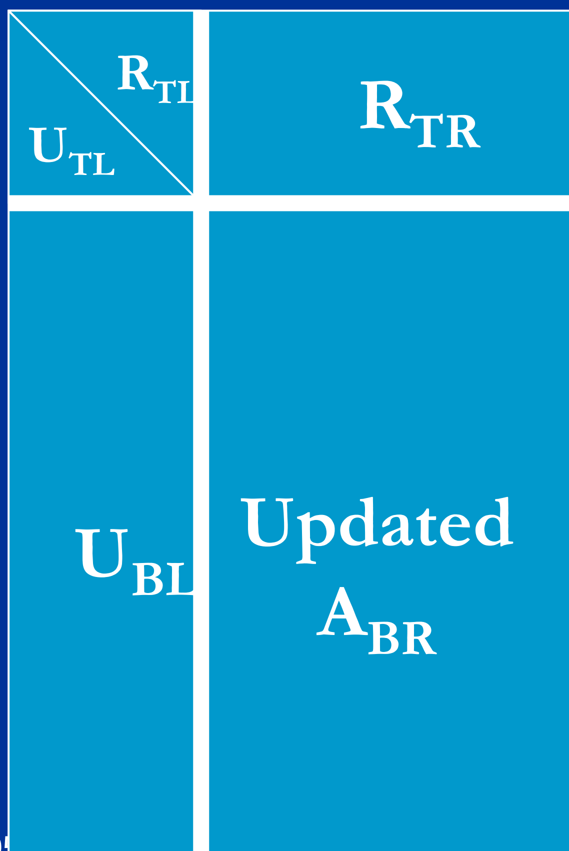
Some Wisdom from the Past

- The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness. But one should not first make the program and then prove its correctness, because then the requirement of providing the proof would only increase the poor programmer's burden. On the contrary: the programmer should let correctness proof and program grow hand in hand. (E.W. Dijkstra: "The Humble Programmer," 1972 Turing Award lecture, in *ACM Turing Award Lectures: The First Twenty Years, 1966-1985,* ACM Press, New York, 1987.)

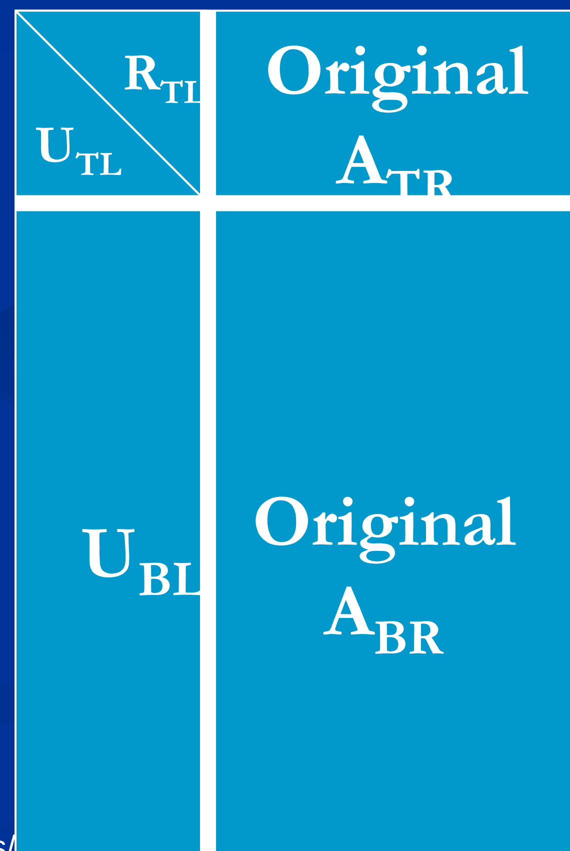
What else does the new notation buy us?

- State of the matrix at the top of the loop

Right-looking algorithm



Left-looking algorithm



Key insight

- Given the state that is to be maintained, an algorithm can be systematically derived.
- The method is sufficiently systematic that it can be made mechanical.

A simpler example: TRSM

- $B := U^{-1} B$ where U is upper triangular

$$\left(\frac{B_T}{B_B} \right) := \left(\frac{U_{TL}^{-1}(B_T - U_{TR} U_{BR}^{-1} B_B)}{U_{BR}^{-1} B_B} \right)$$

Mathematical specification
of operation

↓

**Mechanical derivation
of algorithms**

↓

Family of algorithms

$$\frac{U_{TL}^{-1}(B_T - U_{TR}\, U_{BR}^{-1} B_B)}{U_{BR}^{-1} B_B}$$

MATHEMATICA 5

Family of algorithms

Mechanical Derivation

Mathematical specification of operation

Mechanical derivation of algorithms

MATHEMATICA 5

Family of algorithms

Switch to Demo

Mathematica 5.0 - [SylvCoupled.nb -STUDENT VERSION-]

File Edit Cell Format Input Kernel Find Window Help

SylvCoupled.nb -STUDENT VERSION-

```
Notation[Ξ( A_, B_, C_
           D_, E_, F_ ) ⟺ coupledSylv[A_, B_, C_, D_, E_, F_]]
```

PMEs

1x2

2x1

2x2

```
worksheet[coupledSylv,
 {{"A", "UpperTriangular", "TL"}, {"B", "UpperTriangular", "BR"}, {"C", "Overwrite", "TR"},
  {"D", "UpperTriangular", "TL"}, {"E", "UpperTriangular", "BR"}, {"F", "TR", "Overwrite"}}]
```

Mathematica 5.0 - [SylvCoupled.nb -STUDENT VERSION-]

File Edit Cell Format Input Kernel Find Window Help

SylvCoupled.nb -STUDENT VERSION-

2x1

2x2

PME 2x2

Loop Inv 1

$$\left(\begin{array}{c|c} C_{TL} & C_{TR} \\ \hline \left(\Xi\left(\frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}}\right)\right)[\![1]\!] & C_{BR} \end{array}\right)$$

$$\left(\begin{array}{c|c} F_{TL} & F_{TR} \\ \hline \left(\Xi\left(\frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}}\right)\right)[\![2]\!] & F_{BR} \end{array}\right)$$

State at top of loop

Loop Inv 2

$$\left(\begin{array}{c|c} -A_{TR} . \left(\Xi\left(\frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}}\right)\right)[\![1]\!] + C_{TL} & C_{TR} \\ \hline \left(\Xi\left(\frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}}\right)\right)[\![1]\!] & C_{BR} \end{array}\right)$$

$$\left(\begin{array}{c|c} F_{TL} & F_{TR} \\ \hline \left(\Xi\left(\frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}}\right)\right)[\![2]\!] & F_{BR} \end{array}\right)$$

Mathematica 5.0 - [SylvCoupled.nb * -STUDENT VERSION-]

File Edit Cell Format Input Kernel Find Window Help

SylvCoupled.nb * -STUDENT VERSION-

Loop Inv 4

$$\begin{pmatrix} -A_{TR}\cdot\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]] + C_{TL} & C_{TR} \\ \left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]] & C_{BR} \end{pmatrix}$$

$$\begin{pmatrix} -D_{TR}\cdot\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]] + F_{TL} & F_{TR} \\ \left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]] & F_{BR} \end{pmatrix}$$

In[156]:=
```
coupledSylv4[
  {{aTL_, aTR_},
   {0,    aBR_}},
  {{bTL_, bTR_},
   {0,    bBR_}},
  {{cTL_, cTR_},
   {cBL_, cBR_}},

  {{dTL_, dTR_},
   {0,    dBR_}},
  {{eTL_, eTR_},
   {0,    eBR_}},
  {{fTL_, fTR_},
   {fBL_, fBR_}}] :=

 Module[{BL, BL1, BL2},
  BL = coupledSylv[aBR, bTL, cBL, dBR, eTL, fBL];
  {BL1, BL2} = assignParts[BL, 2];

  {
   {{cTL - prod[aTR, BL1], cTR},
    {BL1, cBR}},
```

← **Mathematic specification of state at top of loop**

Mathematica 5.0 - [SylvCoupled.nb * -STUDENT VERSION-]

File Edit Cell Format Input Kernel Find Window Help

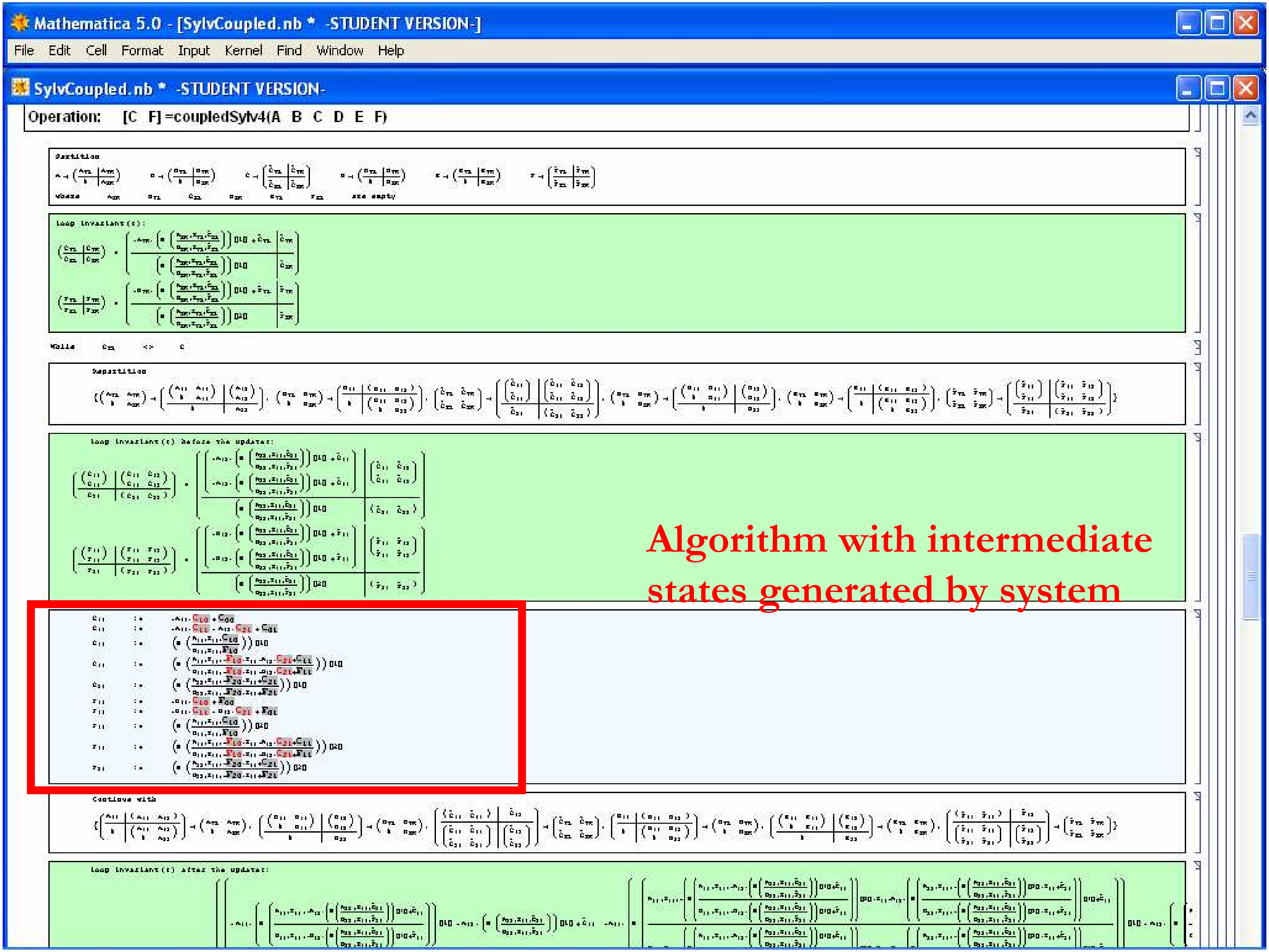
SylvCoupled.nb * -STUDENT VERSION-

```
   {cBL_, cBR_}},

  {{dTL_, dTR_},
   {0,    dBR_}},
  {{eTL_, eTR_},
   {0,    eBR_}},
  {{fTL_, fTR_},
   {fBL_, fBR_}}] :=

 Module[{BL, BL1, BL2},
  BL = coupledSylv[aBR, bTL, cBL, dBR, eTL, fBL];
  {BL1, BL2} = assignParts[BL, 2];

  {
   {{cTL - prod[aTR, BL1], cTR},
    {BL1, cBR}},

   {{fTL - prod[dTR, BL1], fTR},
    {BL2, fBR}}
  }
 ]

In[177]:= Map[myMatrixForm, coupledSylv4[mA, mB, mC, mD, mE, mF]] // ColumnForm;

worksheet[coupledSylv4,
 {{"A", "UpperTriangular", "TL"}, {"B", "UpperTriangular", "BR"}, {"C", "Overwrite", "TR"},
  {"D", "UpperTriangular", "TL"}, {"E", "UpperTriangular", "BR"}, {"F", "TR", "Overwrite"}}]
```

Function that generates algorithm

Loop Inv 5

$$\begin{pmatrix} \left(\Xi\left(\dfrac{A_{TL}, B_{TL}, -A_{TR}\cdot\left(\Xi\left(\frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}}\right)\right)[\![1]\!]+C_{TL}}{D_{TL}, E_{TL}, -D_{TR}\cdot\left(\Xi\left(\frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}}\right)\right)[\![1]\!]+F_{TL}}\right)\right)[\![1]\!] & C_{TR} \\ \left(\Xi\left(\dfrac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}}\right)\right)[\![1]\!] & C_{BR} \end{pmatrix}$$



Mathematica 5.0 - [SylvCoupled.nb * -STUDENT VERSION-]
File Edit Cell Format Input Kernel Find Window Help
SylvCoupled.nb * -STUDENT VERSION-
Operation: [C F] =coupledSylv4(A B C D E F)
Algorithm with intermediate states generated by system

Mathematica 5.0 - [SylvCoupled.nb * -STUDENT VERSION-]

File Edit Cell Format Input Kernel Find Window Help

Testbed5.nb -STUDENT VERSION-

SylvCoupled.nb * -STUDENT VERSION-

$$\left(F_{20} \;\middle|\; (F_{21} \quad F_{22})\right) \qquad \left(\Xi\left(\frac{A_{22}, B_{00}, \hat{C}_{20}}{D_{22}, E_{00}, \hat{F}_{20}}\right)\right)[[2]] \qquad (\hat{F}_{21}$$

$$C_{00} := -A_{01}.\mathbf{C_{10}} + \mathbf{C_{00}}$$

$$C_{01} := -A_{01}.\mathbf{C_{11}} - A_{02}.\mathbf{C_{21}} + \mathbf{C_{01}}$$

$$C_{10} := \left(\Xi\left(\frac{A_{11}, B_{00}, \mathbf{C_{10}}}{D_{11}, E_{00}, \mathbf{F_{10}}}\right)\right)[[1]]$$

$$C_{11} := \left(\Xi\left(\frac{A_{11}, B_{11}, -\mathbf{F_{10}}.B_{01} - A_{12}.\mathbf{C_{21}} + \mathbf{C_{11}}}{D_{11}, E_{11}, -\mathbf{F_{10}}.E_{01} - D_{12}.\mathbf{C_{21}} + \mathbf{F_{11}}}\right)\right)[[1]]$$

$$C_{21} := \left(\Xi\left(\frac{A_{22}, B_{11}, -\mathbf{F_{20}}.B_{01} + \mathbf{C_{21}}}{D_{22}, E_{11}, -\mathbf{F_{20}}.E_{01} + \mathbf{F_{21}}}\right)\right)[[1]]$$

$$F_{00} := -D_{01}.\mathbf{C_{10}} + \mathbf{F_{00}}$$

$$F_{01} := -D_{01}.\mathbf{C_{11}} - D_{02}.\mathbf{C_{21}} + \mathbf{F_{01}}$$

$$F_{10} := \left(\Xi\left(\frac{A_{11}, B_{00}, \mathbf{C_{10}}}{D_{11}, E_{00}, \mathbf{F_{10}}}\right)\right)[[2]]$$

$$F_{11} := \left(\Xi\left(\frac{A_{11}, B_{11}, -\mathbf{F_{10}}.B_{01} - A_{12}.\mathbf{C_{21}} + \mathbf{C_{11}}}{D_{11}, E_{11}, -\mathbf{F_{10}}.E_{01} - D_{12}.\mathbf{C_{21}} + \mathbf{F_{11}}}\right)\right)[[2]]$$

$$F_{21} := \left(\Xi\left(\frac{A_{22}, B_{11}, -\mathbf{F_{20}}.B_{01} + \mathbf{C_{21}}}{D_{22}, E_{11}, -\mathbf{F_{20}}.E_{01} + \mathbf{F_{21}}}\right)\right)[[2]]$$

Continue with

Update to be performed

Mathematica 5.0 - [SylvCoupled.nb -STUDENT VERSION-]

File Edit Cell Format Input Kernel Find Window Help

SylvCoupled.nb -STUDENT VERSION-

$$\left(\Xi\left(\frac{A_{BR}, B_{TL}, C_{BL}}{D_{BR}, E_{TL}, F_{BL}}\right)\right)[[2]] \qquad \left(\Xi\left(\frac{A_{BR}, B_{BR}, -\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].B_{TR}+C_{BR}}{D_{BR}, E_{BR}, -\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].E_{TR}+F_{BR}}\right)\right)[[2]]$$

Loop Inv 49

$$\left(\begin{array}{ll} \left(\Xi\left(\frac{A_{TL},B_{TL},-A_{TR}.\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]]+C_{TL}}{D_{TL},E_{TL},-D_{TR}.\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]]+F_{TL}}\right)\right)[[1]] - \left(\Xi\left(\frac{A_{TL},B_{TL},-A_{TR}.\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]]+C_{TL}}{D_{TL},E_{TL},-D_{TR}.\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]]+F_{TL}}\right)\right)[[2]].B_{TR} - A_{TR}.\left(\Xi\left(\frac{A_{BR},B_{BR},-\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].B_{TR}+C_{BR}}{D_{BR},E_{BR},-\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].E_{TR}+F_{BR}}\right)\right)[[1]] + (& \\ \left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]] & \left(\Xi\left(\frac{A_{BR},B_{BR},-\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].B_{TR}+C_{BR}}{D_{BR},E_{BR},-\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].E_{TR}+F_{BR}}\right)\right)[[1]] \\ \left(\Xi\left(\frac{A_{TL},B_{TL},-A_{TR}.\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]]+C_{TL}}{D_{TL},E_{TL},-D_{TR}.\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]]+F_{TL}}\right)\right)[[2]] - \left(\Xi\left(\frac{A_{TL},B_{TL},-A_{TR}.\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]]+C_{TL}}{D_{TL},E_{TL},-D_{TR}.\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[1]]+F_{TL}}\right)\right)[[2]].E_{TR} - D_{TR}.\left(\Xi\left(\frac{A_{BR},B_{BR},-\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].B_{TR}+C_{BR}}{D_{BR},E_{BR},-\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].E_{TR}+F_{BR}}\right)\right)[[1]] + (& \\ \left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]] & \left(\Xi\left(\frac{A_{BR},B_{BR},-\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].B_{TR}+C_{BR}}{D_{BR},E_{BR},-\left(\Xi\left(\frac{A_{BR},B_{TL},C_{BL}}{D_{BR},E_{TL},F_{BL}}\right)\right)[[2]].E_{TR}+F_{BR}}\right)\right)[[2]] \end{array}\right.$$

```
worksheet[coupledSylv,
 {{"A", "UpperTriangular", "TL"}, {"B", "UpperTriangular", "BR"}, {"C", "Overwrite", "TR"},
  {"D", "UpperTriangular", "TL"}, {"E", "UpperTriangular", "BR"}, {"F", "TR", "Overwrite"}}]
```

Scope of Methodology

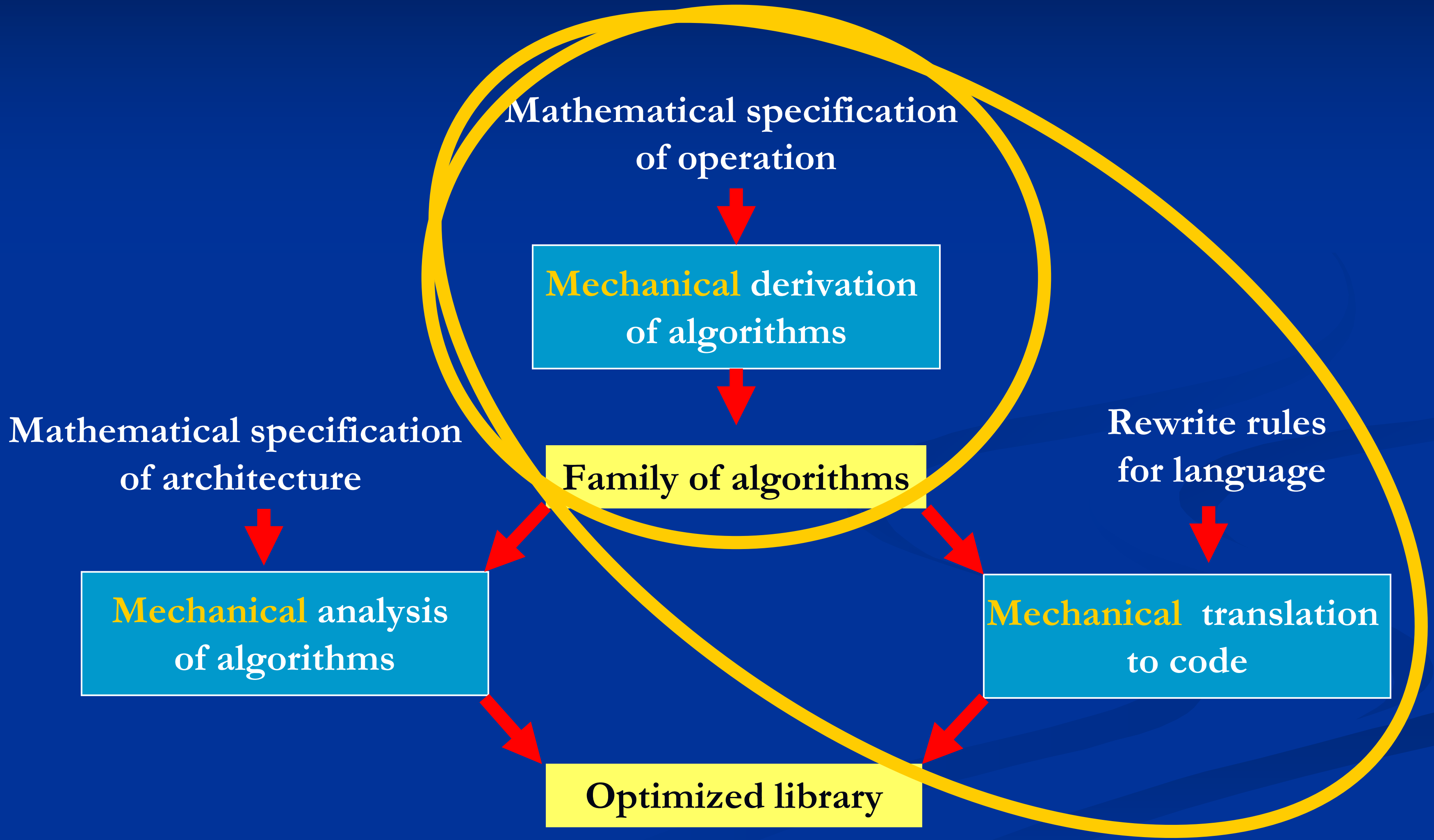
- Triangular Generalized Sylvester Equation

$$A X + Y B = C$$
$$D X + Y E = F$$

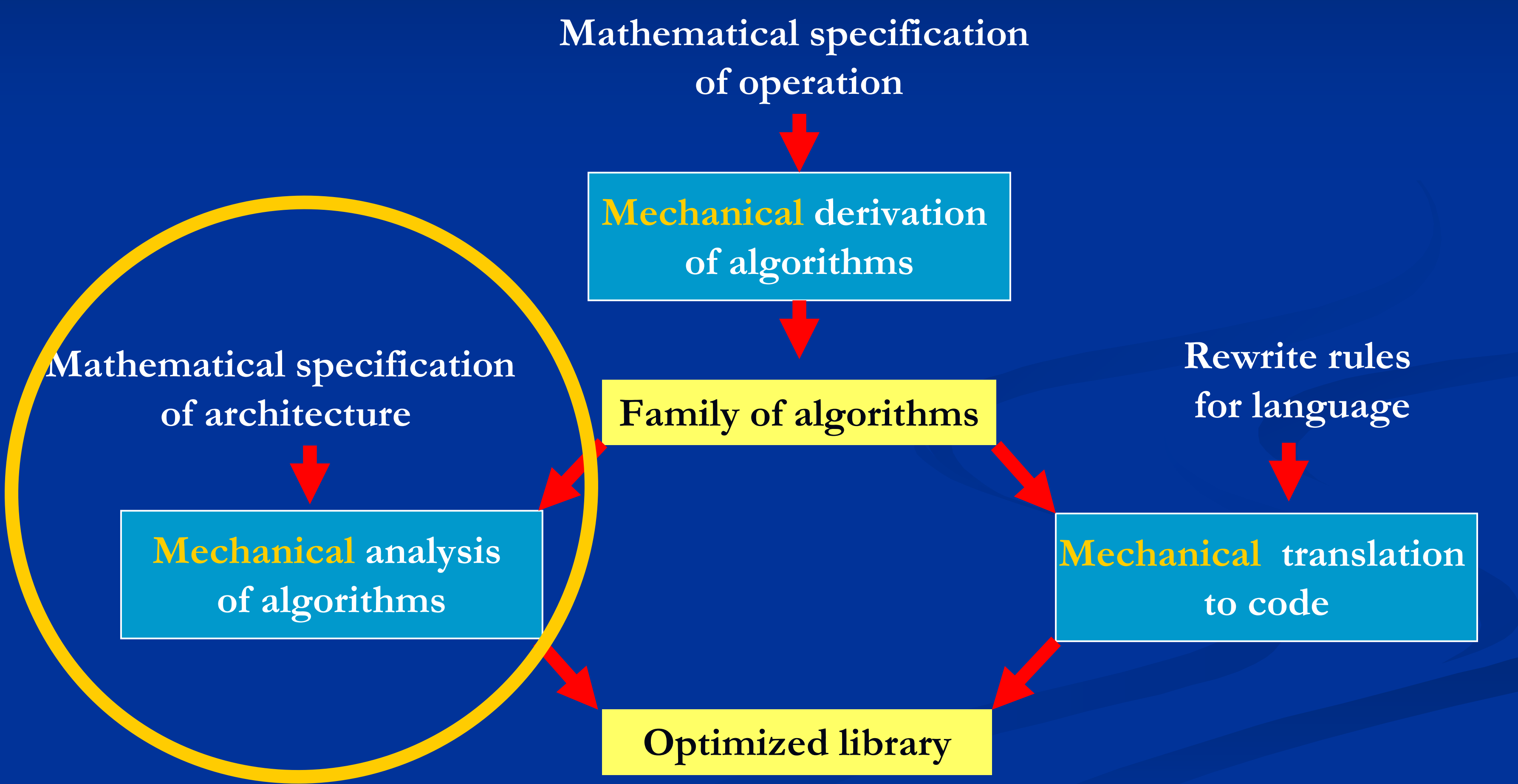
- A, B, D, E triangular
- X and Y overwrite C and F

- 57 algorithmic variants…
- Blocked and unblocked each
- Compose recursively for optimal performance…

The Final Generation



The Final Generation

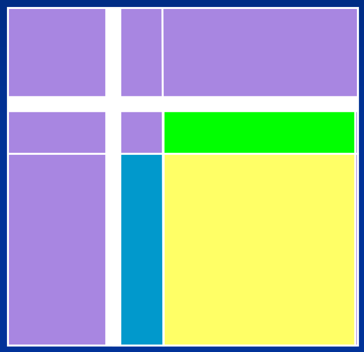


Mechanical Cost Analysis

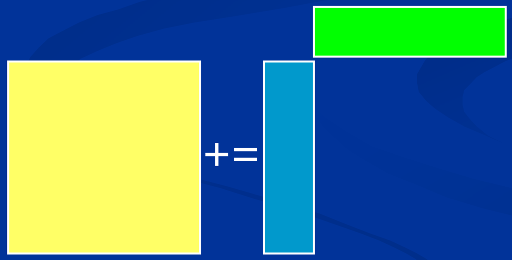
Dissertation of
John Gunnels
(for parallel distributed)

Mechanical Cost Analysis
Key: Vertical Integration

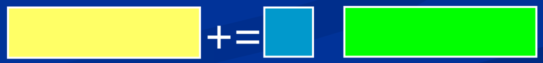
- Solvers (FLAME)



- BLAS (FLAME)



- Low level kernels (K.Goto)



Systematic

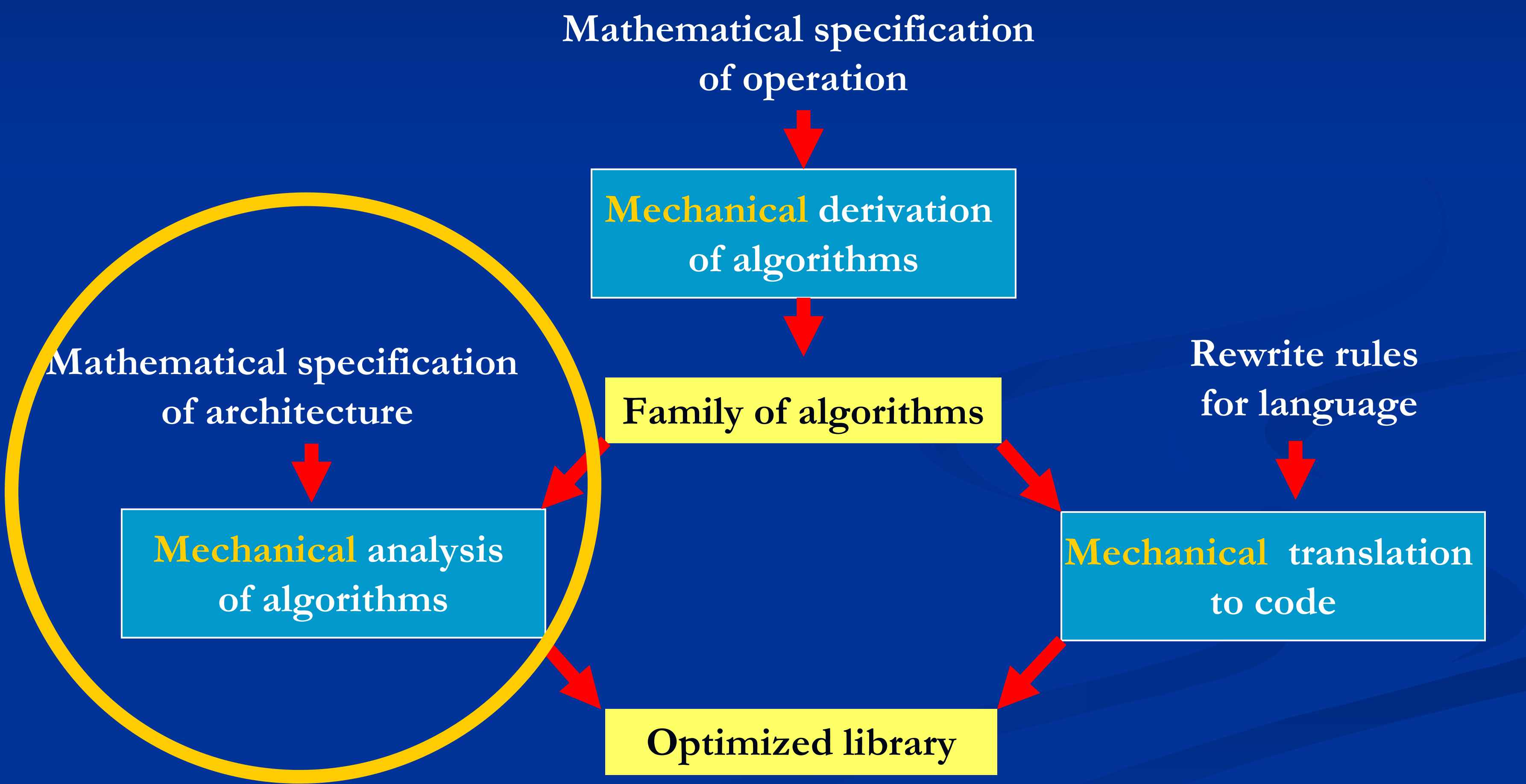
~~Mechanical~~ Stability Analysis

Paolo Bientinesi

(in progress)

Annotated Algorithm: $\kappa := x^T y$		$\check{\kappa} = x^T \Delta y$	Step
Partition $x \rightarrow \left(\frac{x_T}{x_B}\right)$, $y \rightarrow \left(\frac{y_T}{y_B}\right)$ where ...		$\Delta \rightarrow \begin{pmatrix} \Delta_T & 0 \\ 0 & \Delta_B \end{pmatrix}$	4
$\left\{ \check{\kappa} = \left[x_T^T y_T \right] \right\}$		$\left\{ \check{\kappa} = x_T^T \Delta_T y_T \right\}$	2
while $m(x_B) > 0$ **do**		$\{ k = m(x_T) \}$	3
Repartition $\left(\frac{x_T}{x_B}\right) \rightarrow \begin{pmatrix} x_0 \\ \chi_1 \\ x_2 \end{pmatrix}$, $\left(\frac{y_T}{y_B}\right) \rightarrow \begin{pmatrix} y_0 \\ \psi_1 \\ y_2 \end{pmatrix}$ where ...		$\begin{pmatrix} \Delta_T & 0 \\ 0 & \Delta_B \end{pmatrix} \rightarrow \begin{pmatrix} \Delta_0 & 0 & 0 \\ 0 & \delta_1 & 0 \\ 0 & 0 & \Delta_2 \end{pmatrix}$	5a
$\left\{ \check{\kappa} = \left[x_0^T y_0 \right] \right\}$		$\left\{ \check{\kappa} = x_0^T \Delta_0^{\{k\}} y_0 \right\}$	6
$\check{\kappa} := [\check{\kappa} + \chi_1 \psi_1]$	$\check{\kappa} := \left(\check{\kappa} + \chi_1 \psi_1 (1+\epsilon_*)\right)(1+\epsilon_+)$ $= x_0^T \Delta_0^{\{k\}} (1+\epsilon_+) y_0 +$ $\chi_1 (1+\epsilon_*)(1+\epsilon_+) \psi_1$	$\Delta_0 := \Delta_0 (1+\epsilon_+)$ $\delta_1 := (1+\epsilon_+)(1+\epsilon_*)$	8
$\left\{ \check{\kappa} = \left[x_0^T y_0 + \chi_1 \psi_1 \right] \right\}$		$\left\{ \check{\kappa} = \begin{pmatrix} x_0 \\ \chi_1 \end{pmatrix}^T \begin{bmatrix} \Delta_0^{\{k\}} & \\ & \delta_1 \end{bmatrix} \begin{pmatrix} y_0 \\ \psi_1 \end{pmatrix} = x_0^T \Delta_0^{\{k\}} y_0 + \chi_1 \delta_1 \psi_1 \right\}$	7
Continue with ...		$\begin{pmatrix} \Delta_T & 0 \\ 0 & \Delta_B \end{pmatrix} \leftarrow \begin{pmatrix} \Delta_0 & 0 & 0 \\ 0 & \delta_1 & 0 \\ 0 & 0 & \Delta_2 \end{pmatrix}$	5b
enddo			

The Final Generation

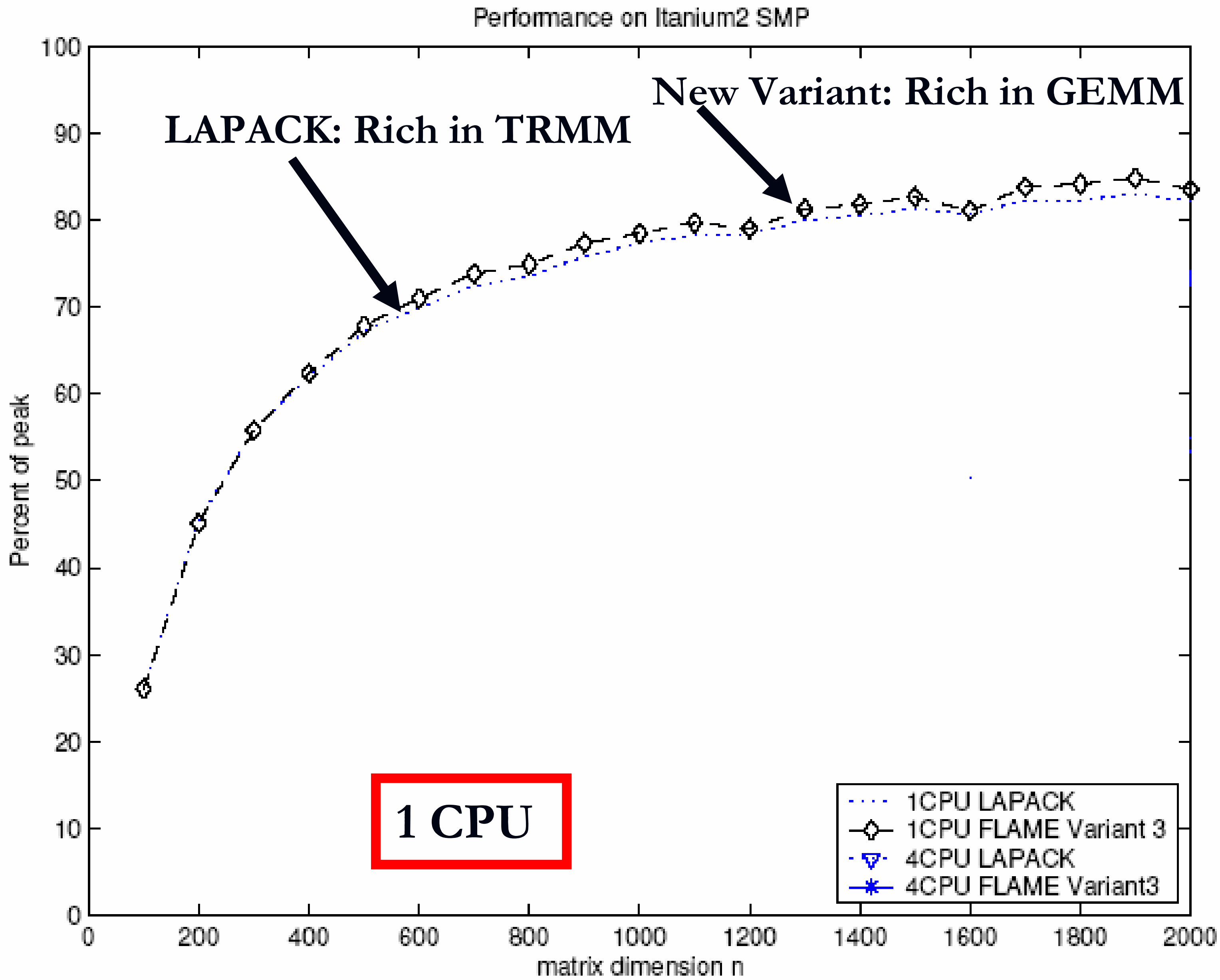


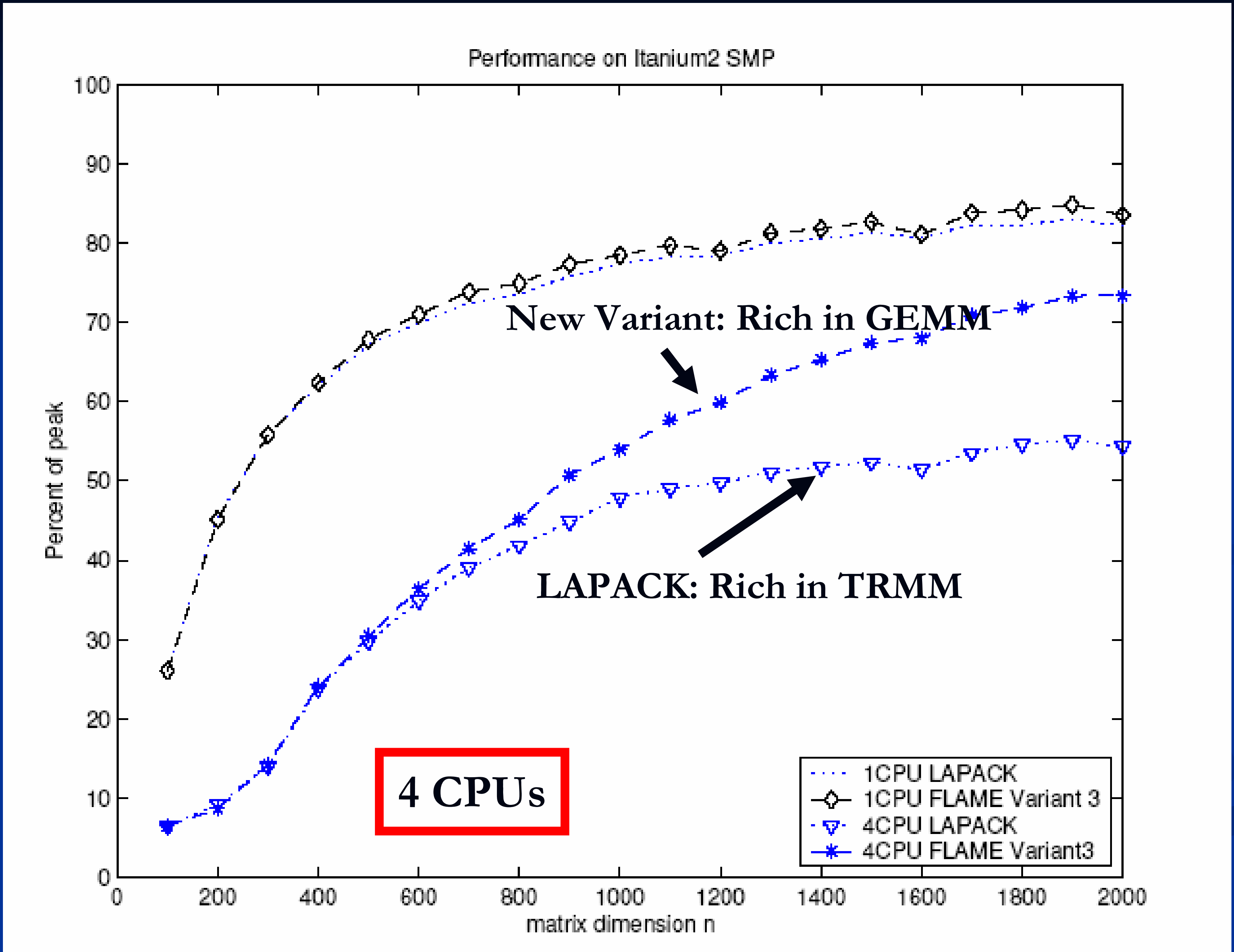
Future Challenges: Multicore Processors

- Processors with two cores available now
- All HP users will have to cope
- 32-128 cores per processor in 10 years?
- 32 processor SMP x 32 cores = 1024 way SMP parallelism
- Virtually no literature on SCALABLE linear algebra libraries for SMPs

Shared Memory Parallelism

- Traditional approach:
  - Only from multithreaded BLAS
- Observation:
  - Better speedup if parallelism is exposed at a higher level
  - Scalability requires 2D work distribution
- FLAME approach:
  - Choose the best algorithmic variant
  - Work queuing (e.g., OpenMP task queues)


Performance on Itanium2 SMP
LAPACK: Rich in TRMM
New Variant: Rich in GEMM
1 CPU
1CPU LAPACK
1CPU FLAME Variant 3
4CPU LAPACK
4CPU FLAME Variant3
Percent of peak
matrix dimension n
0
10
20
30
40
50
60
70
80
90
100
0
200
400
600
800
1000
1200
1400
1600
1800
2000



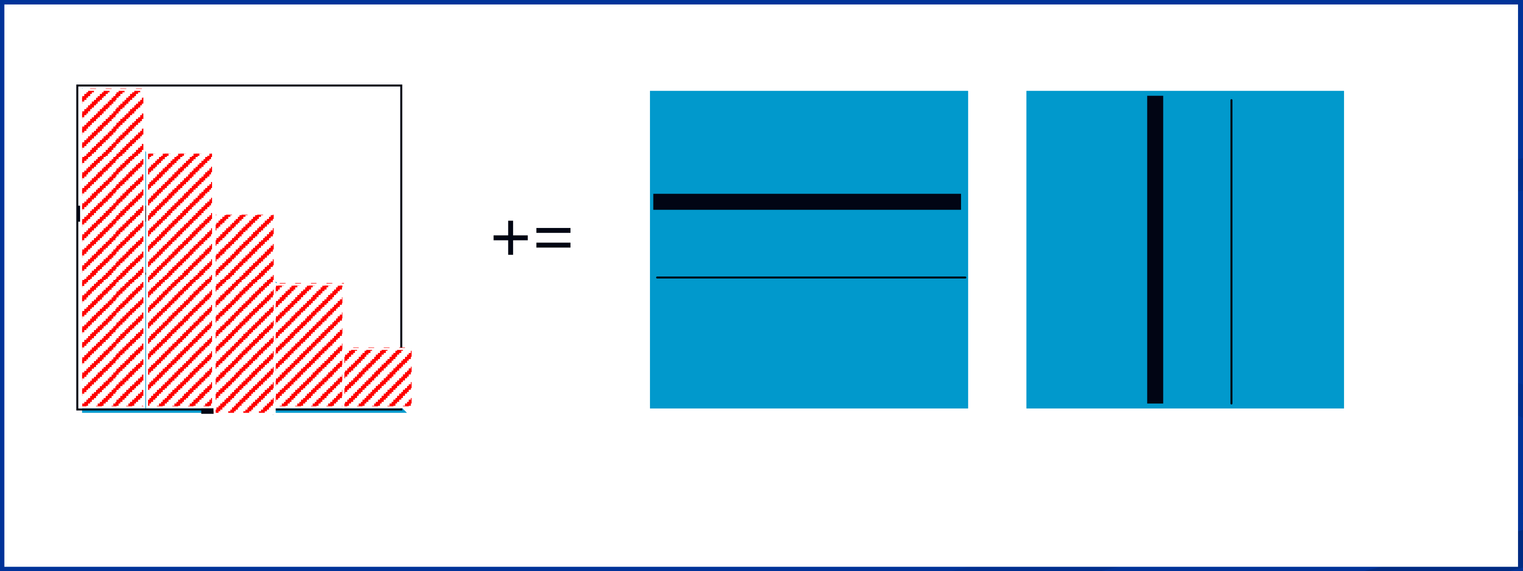
Performance on Itanium2 SMP
Percent of peak
matrix dimension n
0
10
20
30
40
50
60
70
80
90
100
0
200
400
600
800
1000
1200
1400
1600
1800
2000
New Variant: Rich in GEMM
LAPACK: Rich in TRMM
4 CPUs
1CPU LAPACK
1CPU FLAME Variant 3
4CPU LAPACK
4CPU FLAME Variant3

Work queuing

- Simple example:
  - Symmetric rank-k update

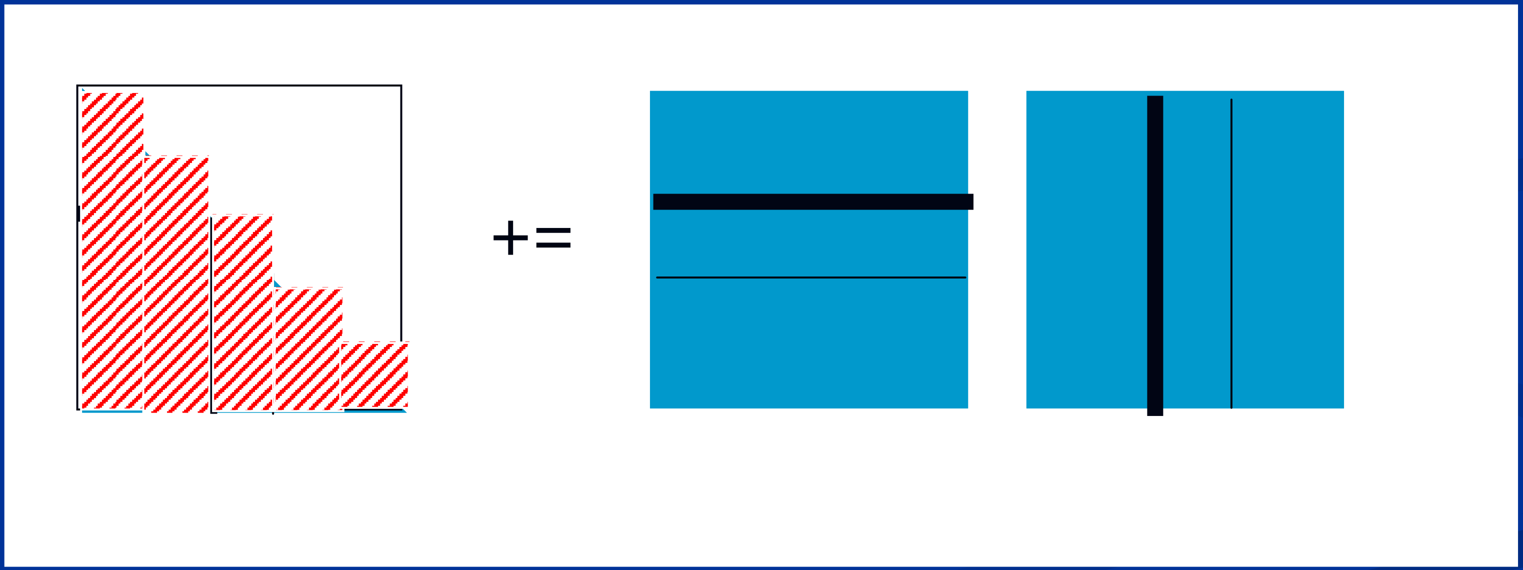
Work queuing

- Simple example:
  - Symmetric rank-k update



Shared Memory Parallelism

- Example:
  - Symmetric rank-k update



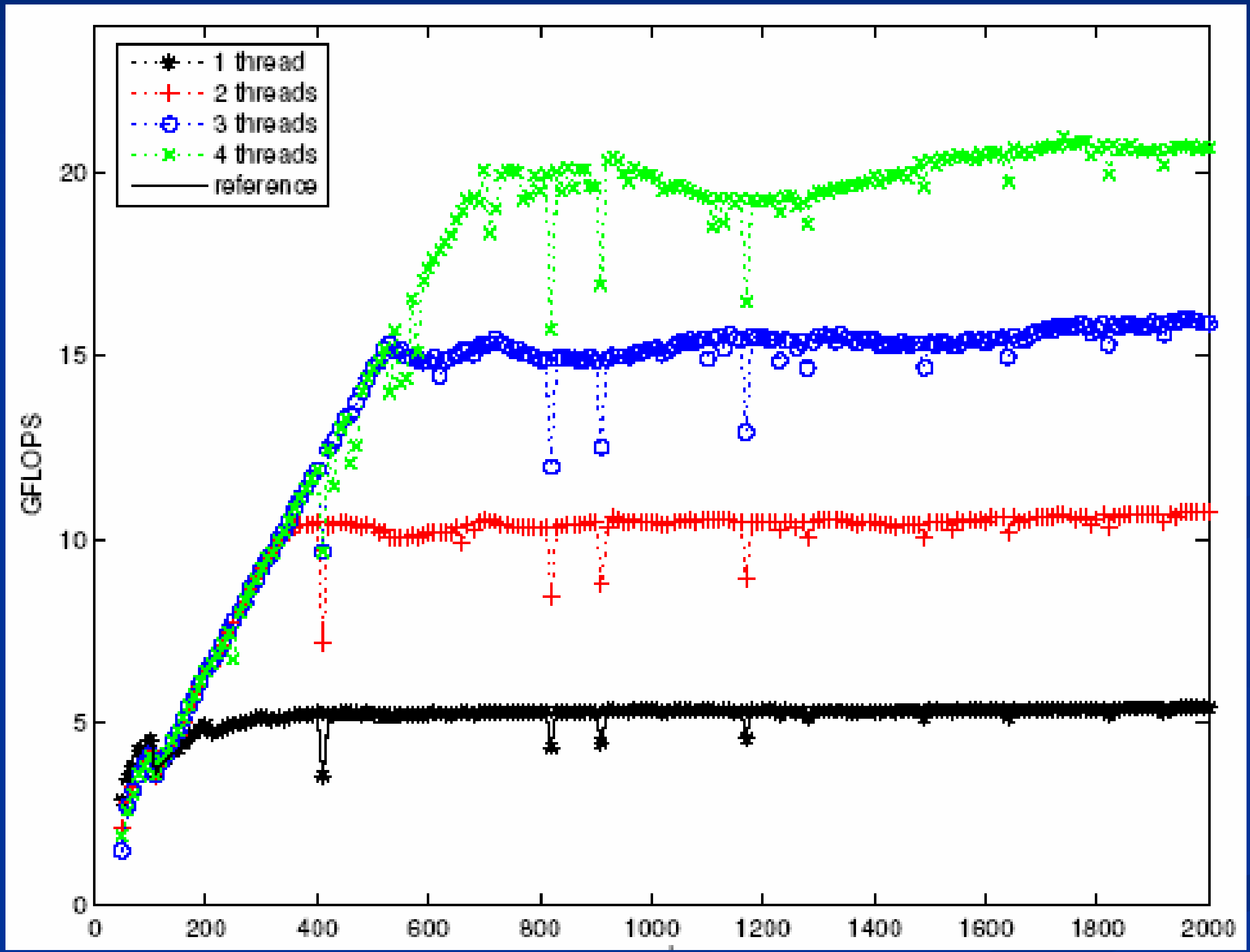
```
while ( FLA_Obj_length( CTL ) < FLA_Obj_length( C ) ){
  b = min( FLA_Obj_length( CBR ), nb_alg );

  FLA_Repart_2x2_to_3x3( CTL, /**/ CTR,    &C00, /**/ &C01, &C02,
                        /*************/   /********************/
                                            &C10, /**/ &C11, &C12,
                         CBL, /**/ CBR,    &C20, /**/ &C21, &C22,
                         b, b, FLA_BR );
  FLA_Repart_2x1_to_3x1( AT,                 &A0,
                       /* ** */            /* ** */
                                             &A1,
                         AB,                 &A2,    b, FLA_BOTTOM );
  /*------------------------------------------------------------*/


    FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE, ONE, A0, A1, ONE, C10 );
    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, ONE, A1, ONE, C11 );

  /*------------------------------------------------------------*/
  FLA_Cont_with_3x3_to_2x2( &CTL, /**/ &CTR,  C00, C01, /**/ C02,
                                               C10, C11, /**/ C12,
                            /**************/ /*****************/
                            &CBL, /**/ &CBR,  C20, C21, /**/ C22,
                            FLA_TL );
  FLA_Cont_with_3x1_to_2x1( &AT,                  A0,
                                                  A1,
                            /* ** */            /* ** */
                            &AB,                  A2,     FLA_TOP );
}
```

```
while ( FLA_Obj_length( CTL ) < FLA_Obj_length( C ) ){
  b = min( FLA_Obj_length( CBR ), nb_alg );

  FLA_Repart_2x2_to_3x3( CTL, /**/ CTR,    &C00, /**/ &C01, &C02,
                        /*************/   /******************/
                                            &C10, /**/ &C11, &C12,
                         CBL, /**/ CBR,    &C20, /**/ &C21, &C22,
                         b, b, FLA_BR );
  FLA_Repart_2x1_to_3x1( AT,                &A0,
                      /* ** */            /* ** */
                                            &A1,
                         AB,                &A2,    b, FLA_BOTTOM );
  /*------------------------------------------------------------*/

    FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE, ONE, A0, A1, ONE, C10 );
    FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, ONE, A1, ONE, C11 );

  /*------------------------------------------------------------*/
  FLA_Cont_with_3x3_to_2x2( &CTL, /**/ &CTR,  C00, C01, /**/ C02,
                                               C10, C11, /**/ C12,
                           /**************/ /*****************/
                            &CBL, /**/ &CBR,  C20, C21, /**/ C22,
                            FLA_TL );
  FLA_Cont_with_3x1_to_2x1( &AT,                  A0,
                                                   A1,
                           /* ** */              /* ** */
                            &AB,                  A2,     FLA_TOP );
}
```

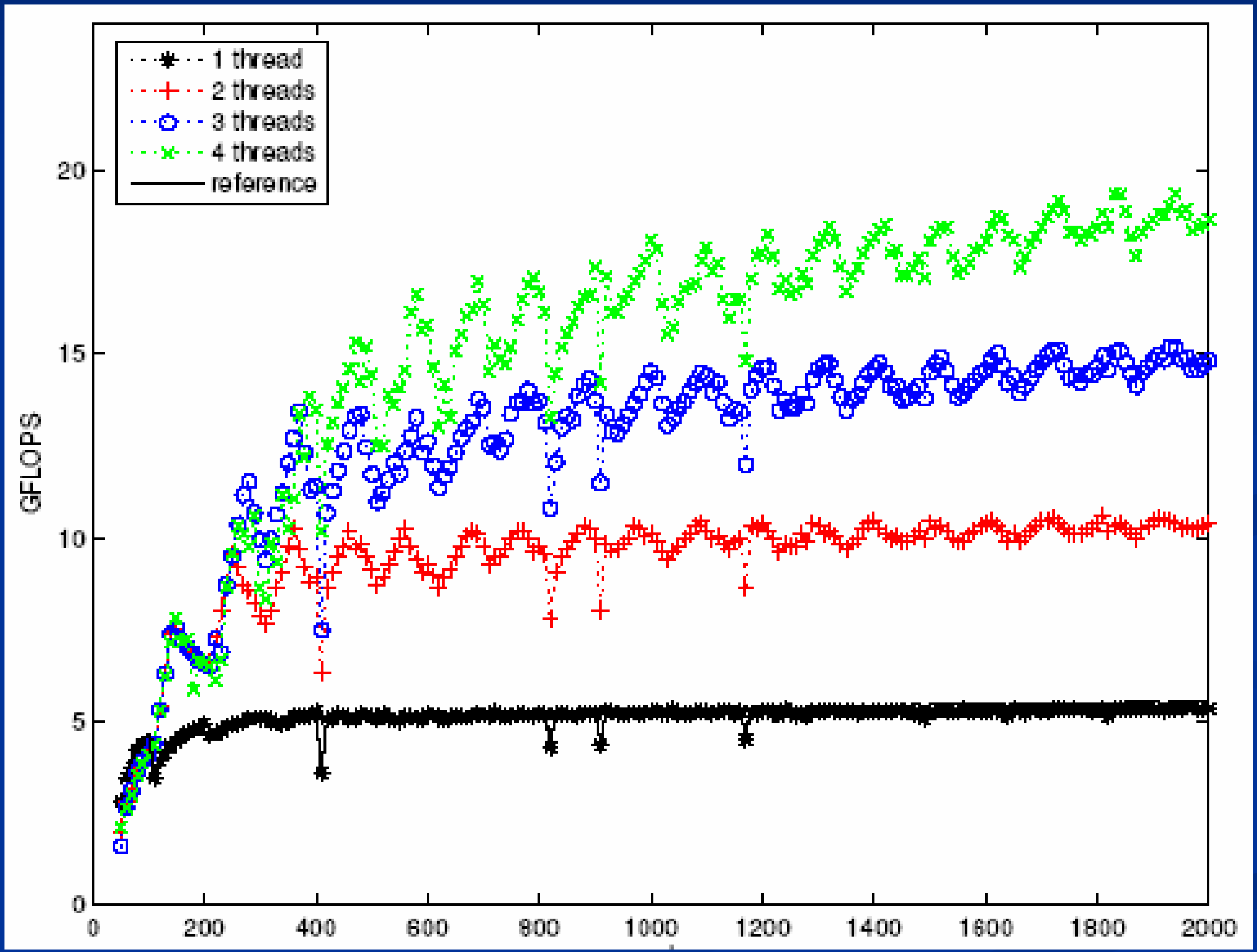
```
#pragma intel omp parallel taskq
{
  while ( FLA_Obj_length( CTL ) < FLA_Obj_length( C ) ){
    b = min( FLA_Obj_length( CBR ), nb_alg );

    FLA_Repart_2x2_to_3x3( CTL, /**/ CTR,    &C00, /**/ &C01, &C02,
                         /*************/   /********************/
                                            &C10, /**/ &C11, &C12,
                           CBL, /**/ CBR,    &C20, /**/ &C21, &C22,
                           b, b, FLA_BR );
    FLA_Repart_2x1_to_3x1( AT,                &A0,
                         /* ** */           /* ** */
                                              &A1,
                           AB,                &A2,    b, FLA_BOTTOM );
    /*--------------------------------------------------------------*/
    #pragma intel omp task captureprivate( A0, A1, C10, C11 )
    {
      FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE, ONE, A0, A1, ONE, C10 );
      FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, ONE, A1, ONE, C11 );
    } /* end task */
    /*--------------------------------------------------------------*/
    FLA_Cont_with_3x3_to_2x2( &CTL, /**/ &CTR,   C00, C01, /**/ C02,
                                                 C10, C11, /**/ C12,
                            /**************/ /*****************/
                              &CBL, /**/ &CBR,   C20, C21, /**/ C22,
                              FLA_TL );
    FLA_Cont_with_3x1_to_2x1( &AT,                    A0,
                                                      A1,
                            /* ** */               /* ** */
                              &AB,                    A2,     FLA_TOP );
  }
}   /* end of taskq */
```

```
#pragma intel omp parallel taskq
 {
 while ( FLA_Obj_length( AT ) < FLA_Obj_length( A ) ){
   b = min( FLA_Obj_length( AB ), nb_alg );

   FLA_Repart_2x1_to_3x1( AT,                &A0,
                       /* ** */           /* ** */
                                          &A1,
                          AB,             &A2,        b, FLA_BOTTOM );
   FLA_Repart_2x2_to_3x3( CTL, /**/ CTR,     &C00, /**/ &C01, &C02,
                       /* ************* */ /* ******************** */
                                          &C10, /**/ &C11, &C12,
                          CBL, /**/ CBR,     &C20, /**/ &C21, &C22, b, b, FLA_BR );

   /*------------------------------------------------------------------------------*/
 #pragma intel omp task captureprivate(A2, A1, C11, C21)
 {
     FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE, ONE, A2, A1, ONE, C21 );
     FLA_Syrk( FLA_LOWER_TRIANGULAR, FLA_NO_TRANSPOSE, ONE, A1, ONE, C11 );
 }
   /*------------------------------------------------------------------------------*/

   FLA_Cont_with_3x1_to_2x1( &AT,                A0,
                                               A1,
                           /* ** */           /* ** */
                             &AB,              A2,     FLA_TOP );
   FLA_Cont_with_3x3_to_2x2( &CTL, /**/ &CTR,     C00, C01, /**/ C02,
                                                C10, C11, /**/ C12,
                           /* ************** *//* ****************** */
                             &CBL, /**/ &CBR,     C20, C21, /**/ C22,    FLA_TL );
 }
```

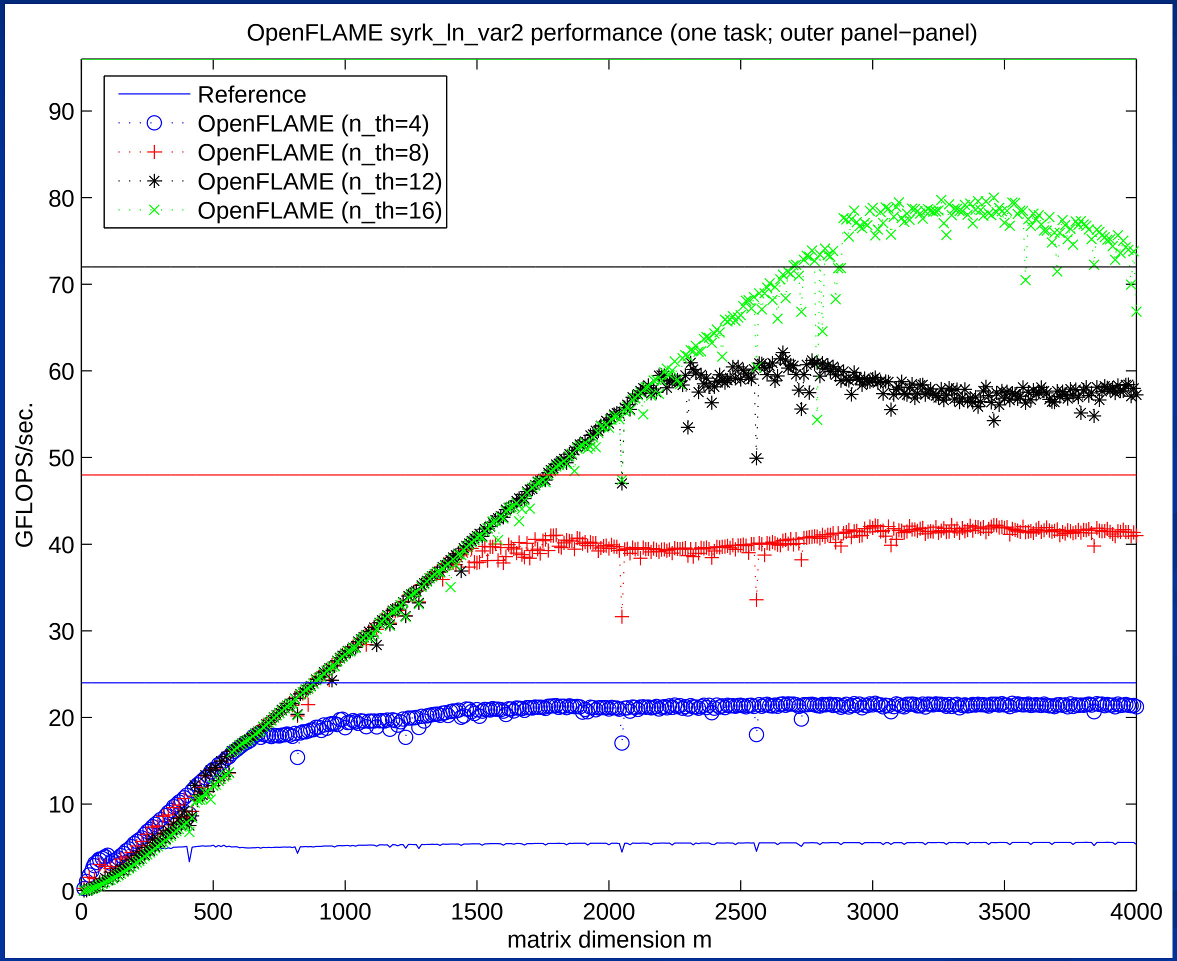
Performance HP 4CPU Itanium2 dsyrk variant 2



Performance HP 4CPU Itanium2 dsyrk variant 3

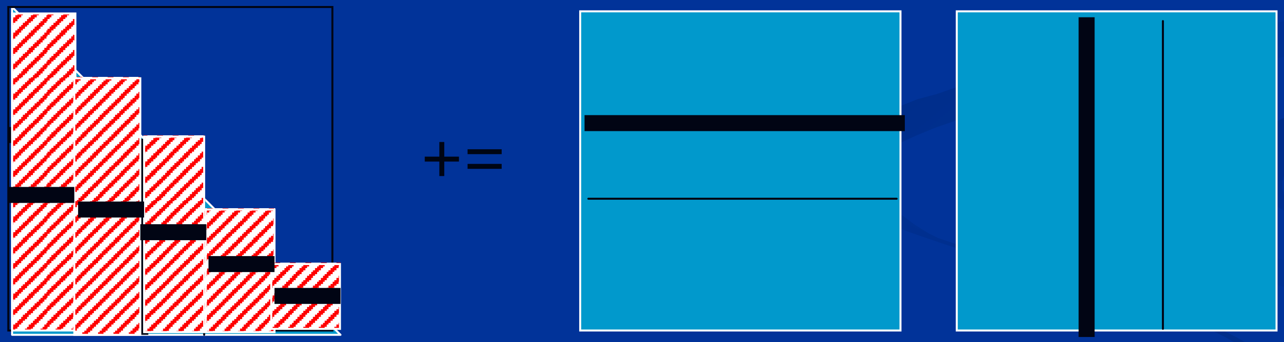


Performance on NEC 16 CPU Itanium2 system (1.5 GHz)



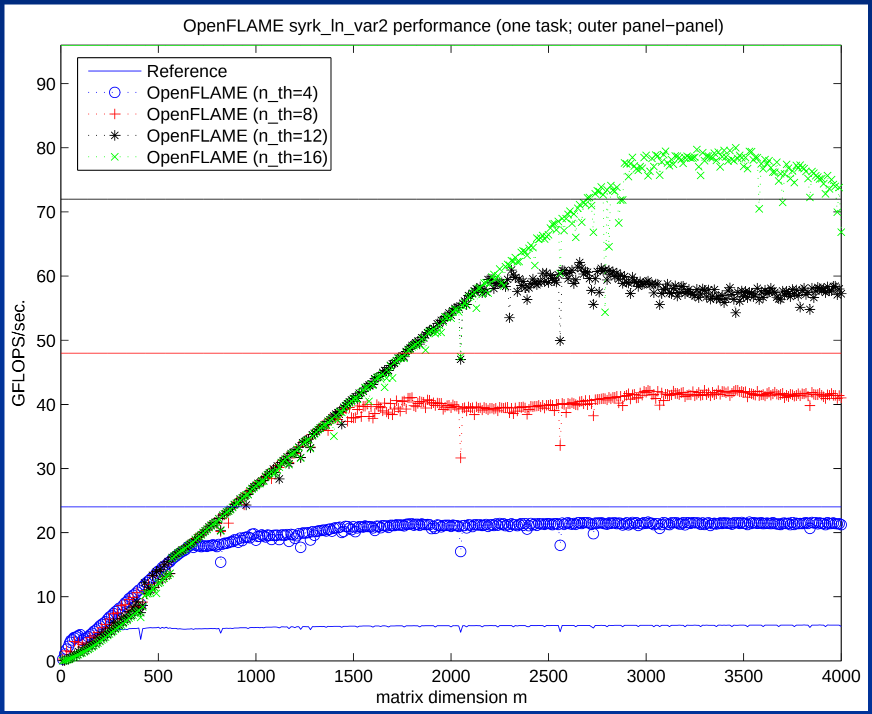
Work queuing

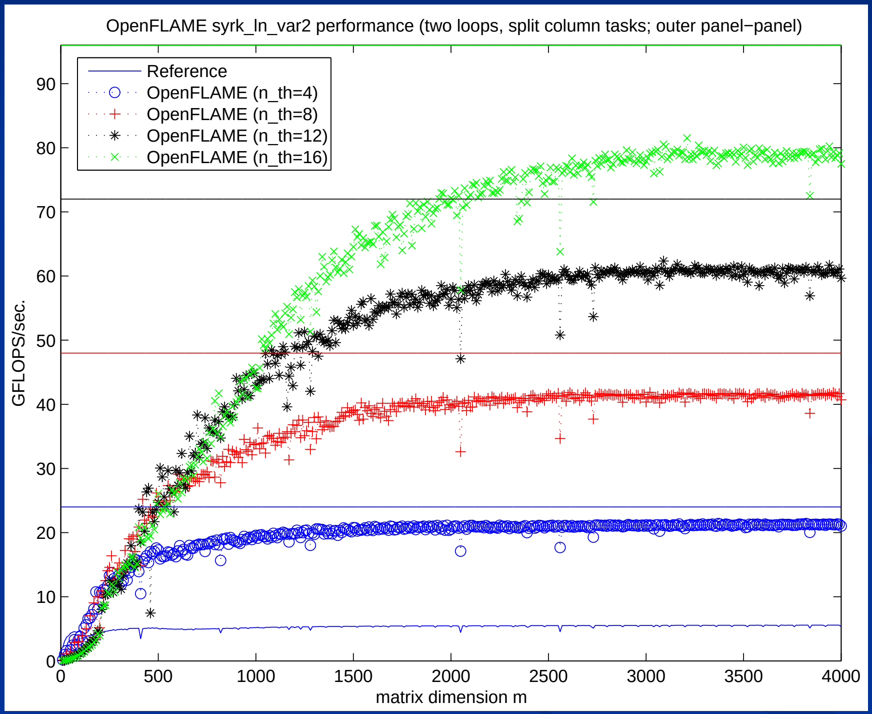
- Example:
  - Symmetric rank-k update



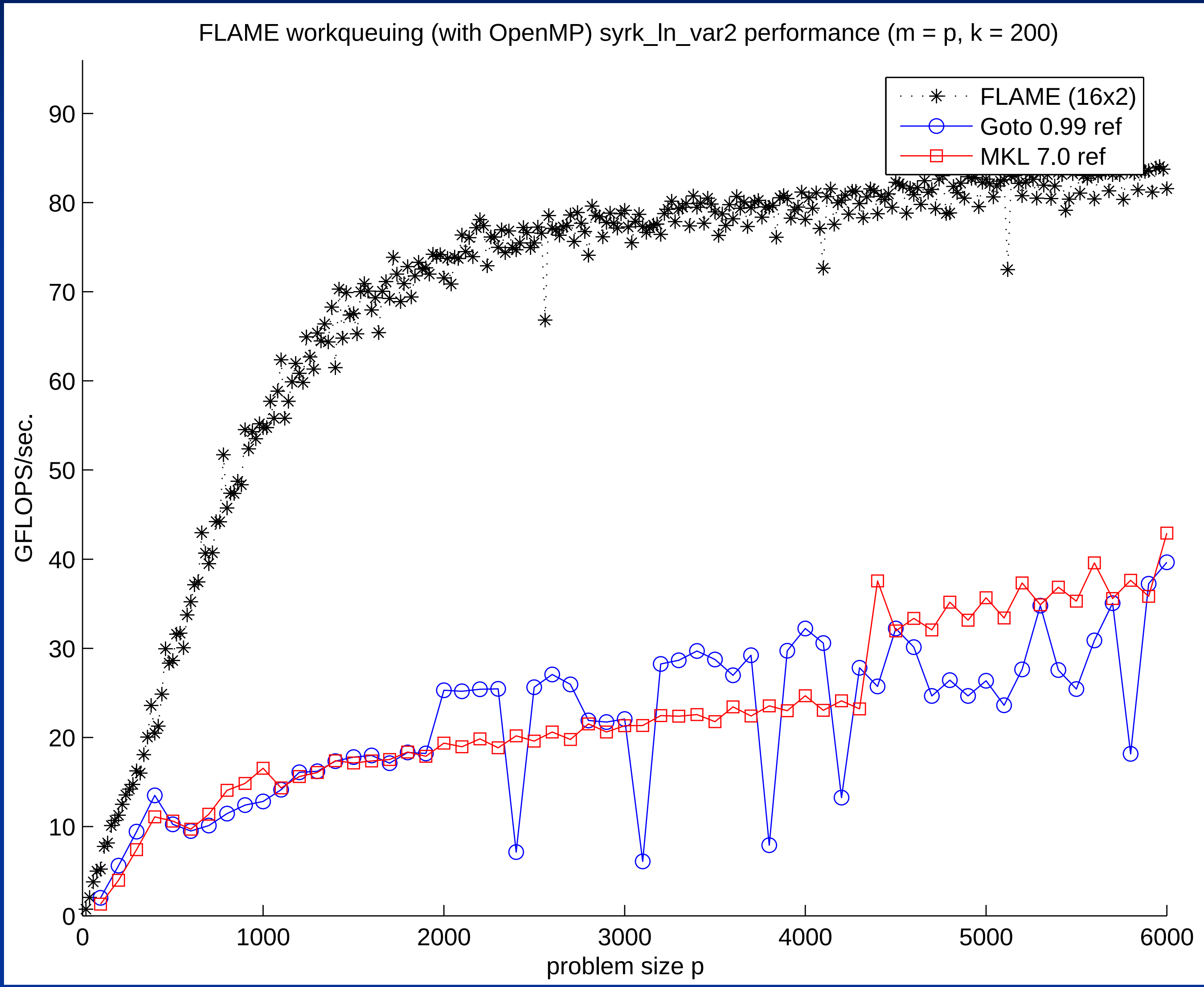
```
#pragma intel omp parallel taskq
 {
 while ( FLA_Obj_length( AT ) < FLA_Obj_length( A ) ){

  b = min( FLA_Obj_length( AB ), nb_alg );

  FLA_Repart_2x1_to_3x1( AT,                &A0,
                      /* ** */            /* ** */
                                            &A1,
                         AB,                &A2,        b, FLA_BOTTOM );

  FLA_Repart_2x2_to_3x3( CTL, /**/ CTR,       &C00, /**/ &C01, &C02,
                      /* ************* */   /* ******************** */
                                              &C10, /**/ &C11, &C12,
                         CBL, /**/ CBR,       &C20, /**/ &C21, &C22,
                         b, b, FLA_BR );

   /*------------------------------------------------------------*/

   b2 = FLA_Obj_length( A2 )/2;

   FLA_Part_2x1( A2,     &A2_T,
                         &A2_B,            b2, FLA_TOP );

   FLA_Part_2x1( C21,    &C21_T,
                         &C21_B,           b2, FLA_TOP );

     /*------------------------------------------------------------*/
     #pragma intel omp task captureprivate(A2_T, A1, C21_T)
       {

       FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE,
                 ONE, A2_T, A1, ONE, C21_T );
       }
       #pragma intel omp task captureprivate(A2_B, A1, C21_B)
       {

       FLA_Gemm( FLA_NO_TRANSPOSE, FLA_TRANSPOSE,
                 ONE, A2_B, A1, ONE, C21_B );
       }
     /*------------------------------------------------------------*/
   /*------------------------------------------------------------*/

   FLA_Cont_with_3x1_to_2x1( &AT,                A0,
                                                A1,
                        /* ** */           /* ** */
                          &AB,                A2,     FLA_TOP );

  FLA_Cont_with_3x3_to_2x2( &CTL, /**/ &CTR,       C00, C01, /**/ C02,
                                                   C10, C11, /**/ C12,
                        /* ************** */  /* ****************** */
                          &CBL, /**/ &CBR,       C20, C21, /**/ C22,
                          FLA_TL );

 }
```

Performance on NEC 16 CPU Itanium2 system (1.5 GHz)

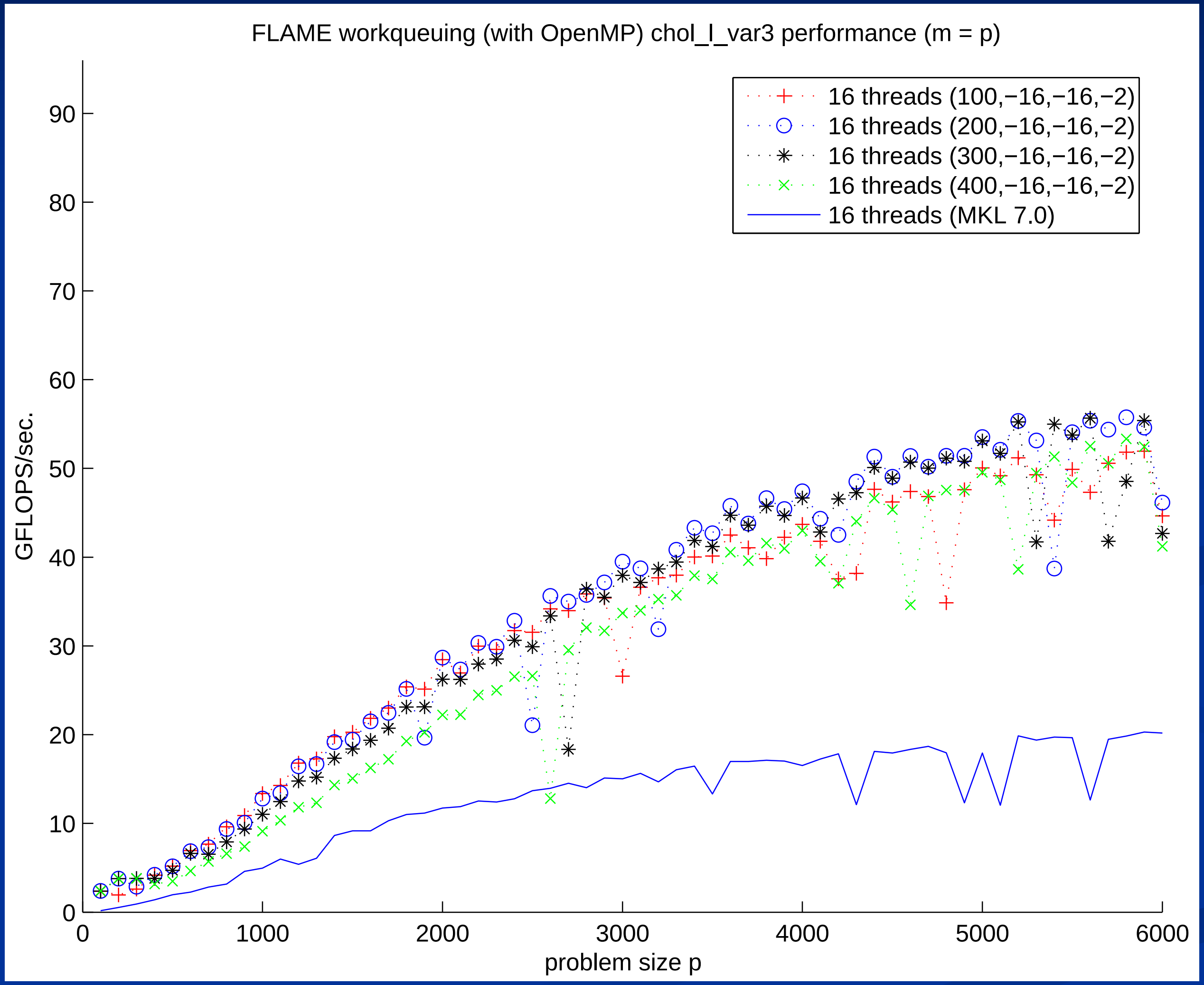




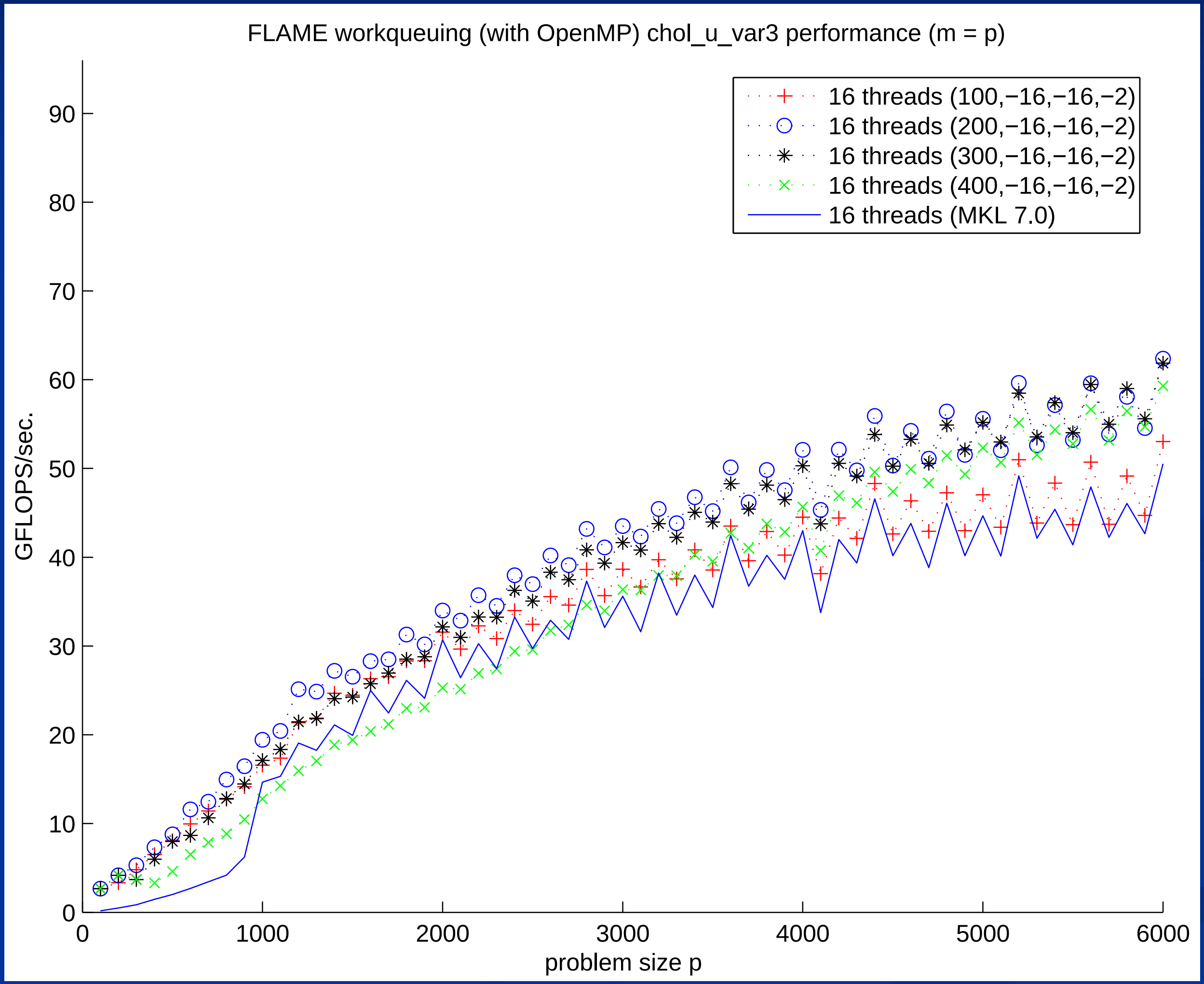
Syrk performance



Cholesky Factorization Performance

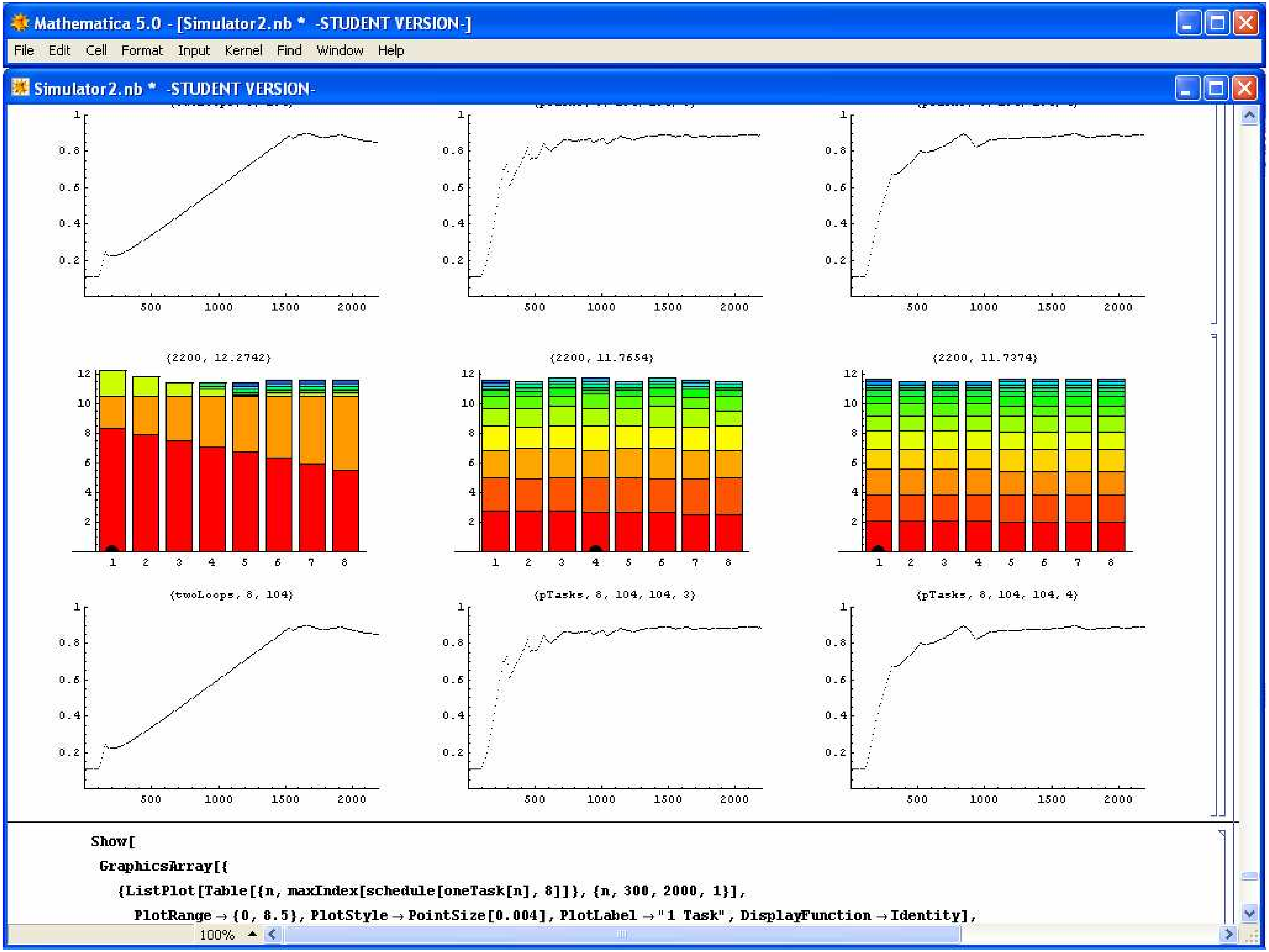


Cholesky Factorization Performance

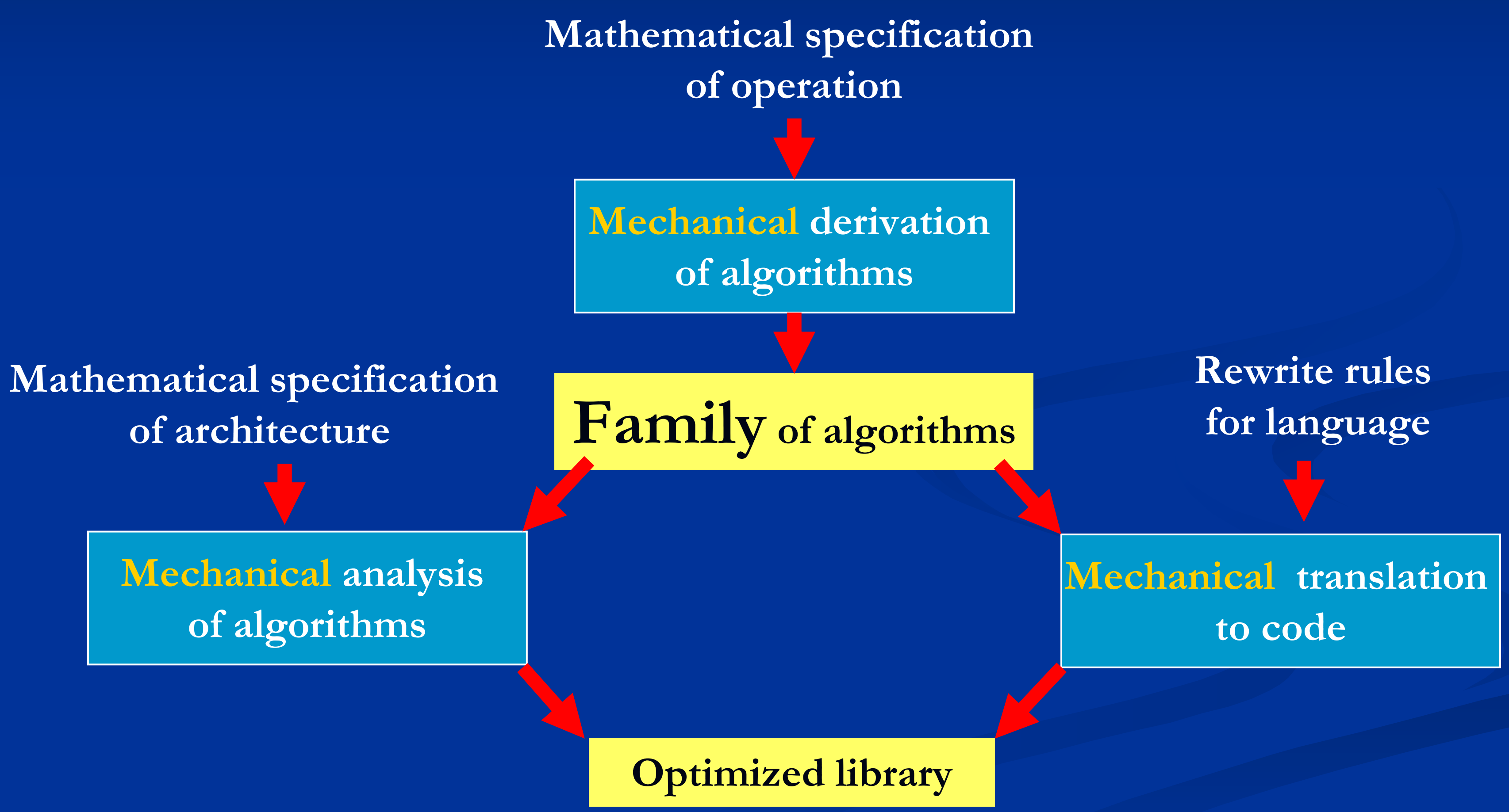


Switch to Demo

Skip demo slides


Mathematica 5.0 - [Simulator2.nb * -STUDENT VERSION-]
File Edit Cell Format Input Kernel Find Window Help
Simulator2.nb * -STUDENT VERSION-
{2200, 12.2742}
{2200, 11.7654}
{2200, 11.7374}
{twoLoops, 8, 104}
{pTasks, 8, 104, 104, 3}
{pTasks, 8, 104, 104, 4}
Show[
GraphicsArray[{
{ListPlot[Table[{n, maxIndex[schedule[oneTask[n], 8]]}, {n, 300, 2000, 1}],
PlotRange → {0, 8.5}, PlotStyle → PointSize[0.004], PlotLabel → "1 Task", DisplayFunction → Identity],
100%

Conclusion


Mathematical specification of operation
Mechanical derivation of algorithms
Family of algorithms
Mathematical specification of architecture
Rewrite rules for language
Mechanical analysis of algorithms
Mechanical translation to code
Optimized library

Conclusion

- Mechanical generation of libraries supports
  - New architectures
    - Architectures currently supported: Sequential, SMP parallel, distributed memory parallel
  - New languages
    - Languages supported: LaTeX, C, Matlab, Fortran, C+MPI, C+OpenMP, Mathematica, Haskell, LabView's G
  - New datastructures
    - Datastructures supported: column-major storage, banded, dense stored by blocks, sparse hierarchical, out-of-core
  - New operations
    - Mechanical derivation of algorithms for all BLAS3, LAPACK, many operations in control theory

More Information

http://www.cs.utexas.edu/users/flame/

http://www.cs.utexas.edu/users/flame/pubs.html

rvdg@cs.utexas.edu

What is needed?

- Project so far concentrates on the science that supports the approach and prototyping of tools
- Full-blown integration of all tools requires
  - Full-time postdoc
  - Full-time professional programmer
  - Part-time web developer
  - Several graduate students
  - Collaboration with the Texas Advanced Computing Center