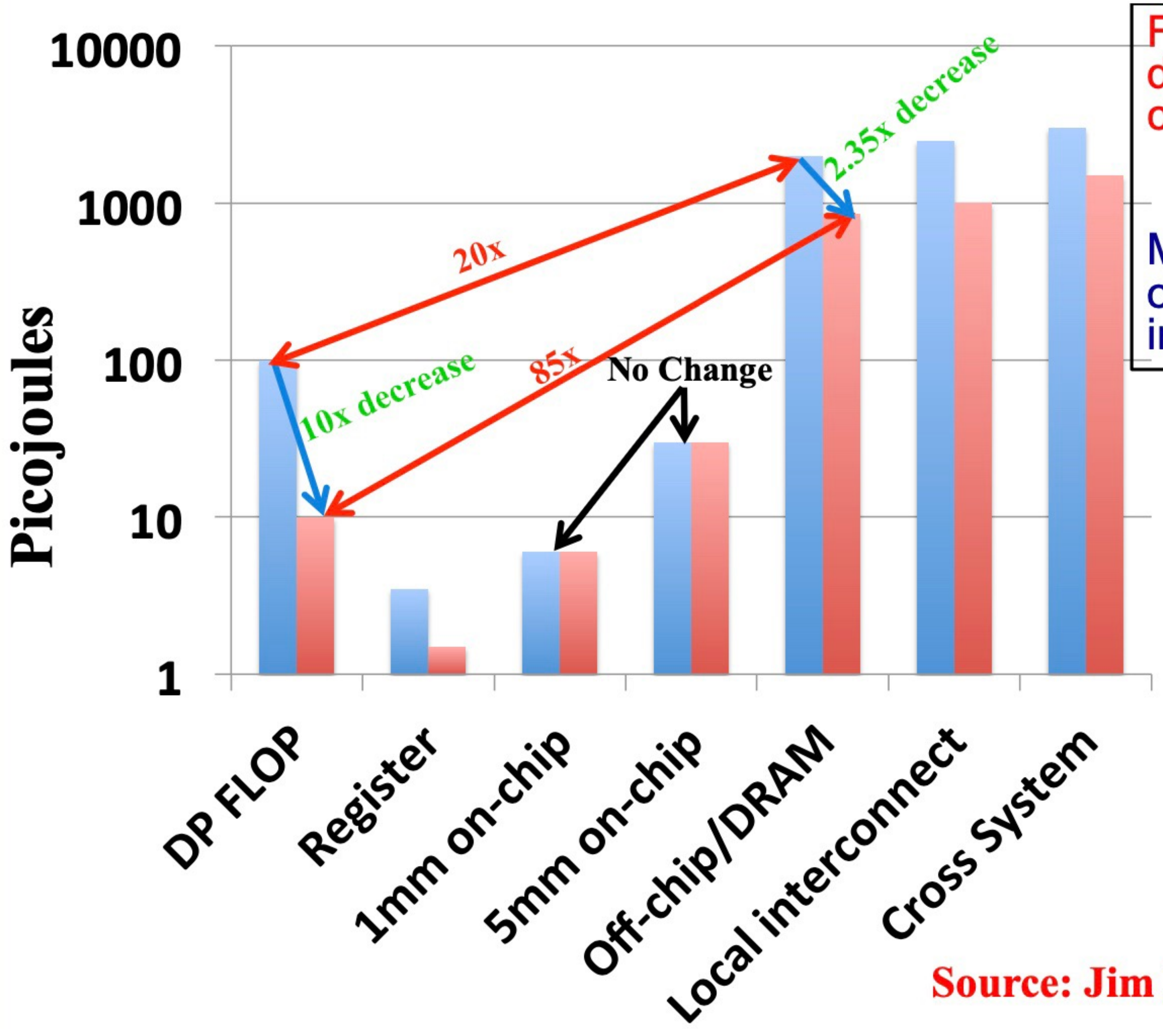
The I/O Requirements of Various Numerical Linear Algebra Kernels

Julien Langou

Wednesday June 26th 2024

Data Movement Cost: Energy Trends



FLOPs almost free; data movement cost is dominant

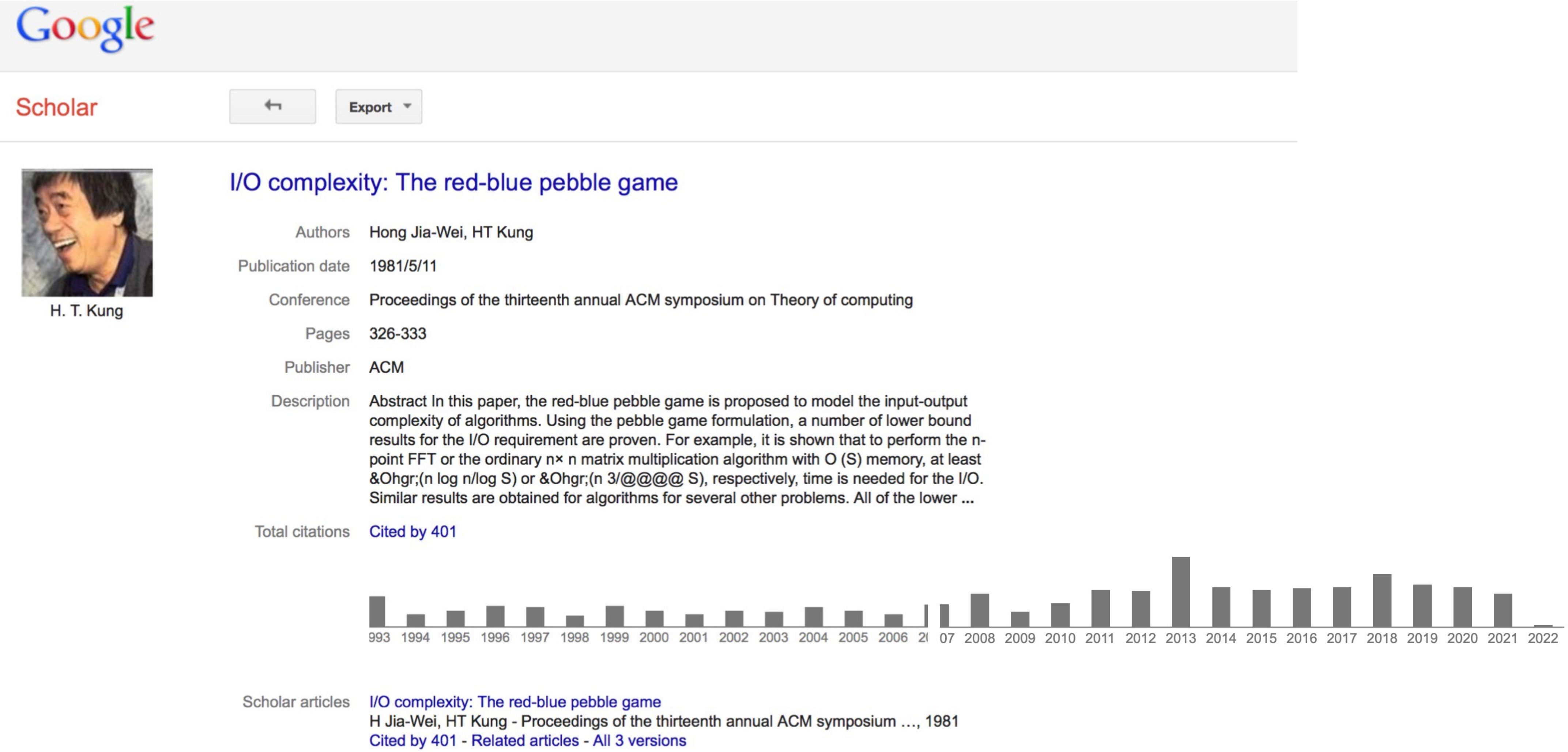
⬇

Minimizing amount of data movement increasingly critical

🟦 45 nm
🟥 11 nm

Source: Jim Demmel, John Shalf

2017

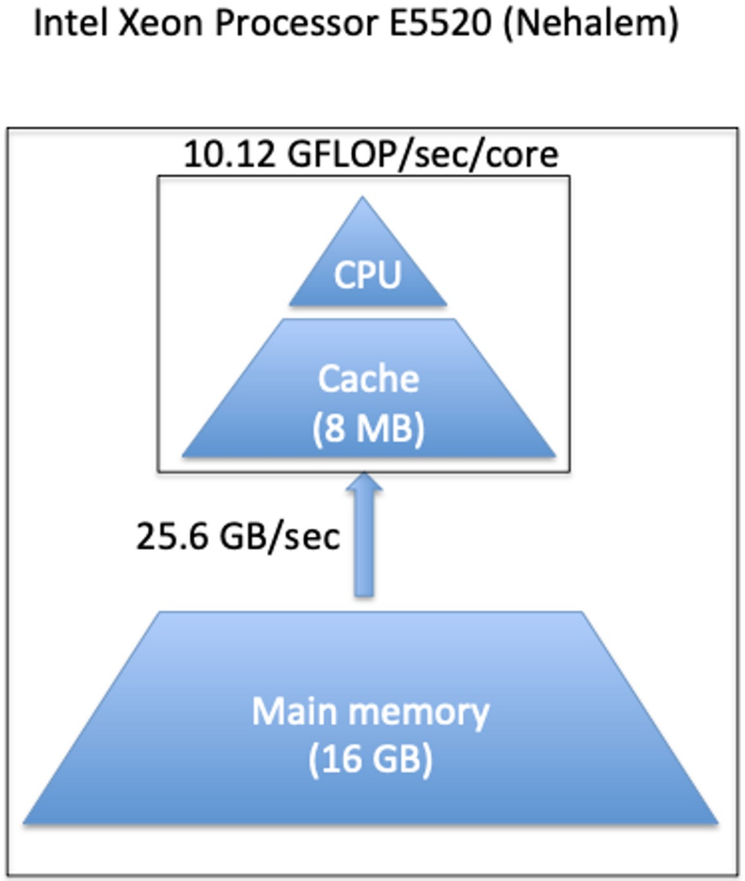


I/O complexity: The red-blue pebble game

Authors	Hong Jia-Wei, HT Kung
Publication date	1981/5/11
Conference	Proceedings of the thirteenth annual ACM symposium on Theory of computing
Pages	326-333
Publisher	ACM
Description	Abstract In this paper, the red-blue pebble game is proposed to model the input-output complexity of algorithms. Using the pebble game formulation, a number of lower bound results for the I/O requirement are proven. For example, it is shown that to perform the n-point FFT or the ordinary n× n matrix multiplication algorithm with O (S) memory, at least &Ohgr;(n log n/log S) or &Ohgr;(n 3/@@@@ S), respectively, time is needed for the I/O. Similar results are obtained for algorithms for several other problems. All of the lower ...
Total citations	Cited by 401

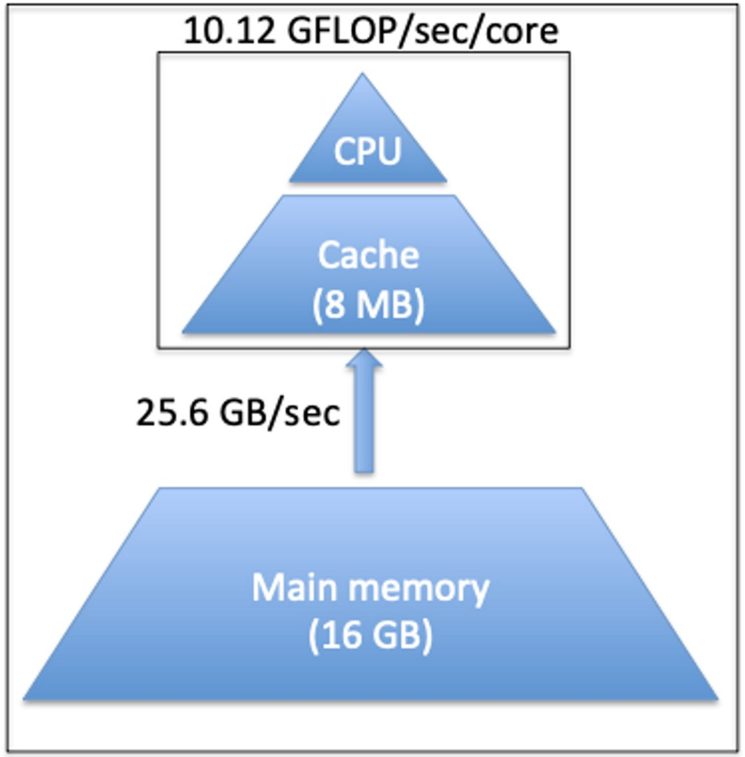
Scholar articles I/O complexity: The red-blue pebble game
H Jia-Wei, HT Kung - Proceedings of the thirteenth annual ACM symposium …, 1981
Cited by 401 - Related articles - All 3 versions

IO lower bound for computer programs

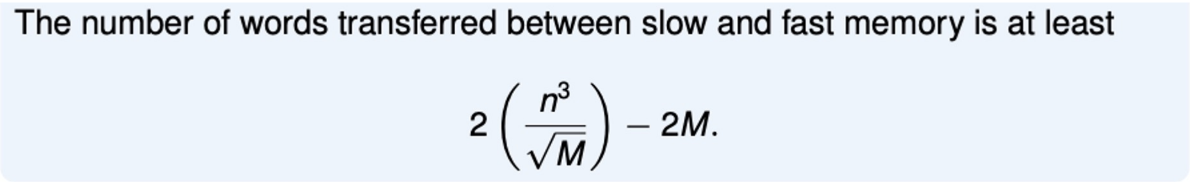


IO lower bound for computer programs

Intel Xeon Processor E5520 (Nehalem)



For MM

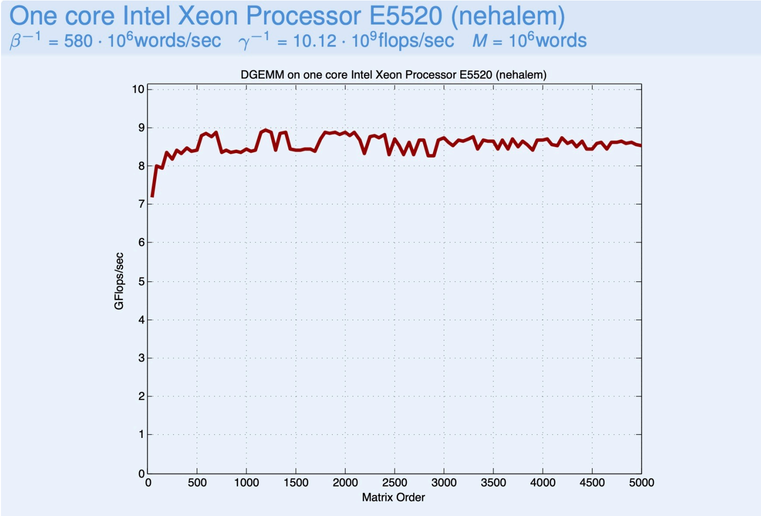

The number of words transferred between slow and fast memory is at least

$$2\left(\frac{n^3}{\sqrt{M}}\right) - 2M.$$

where n is matrix size and M is size of cache.

See: A Tight I/O Lower Bound for Matrix Multiplication. T. M. Smith, B. Lowery, J. Langou, R. A. van de Geijn https://arxiv.org/abs/1702.02017

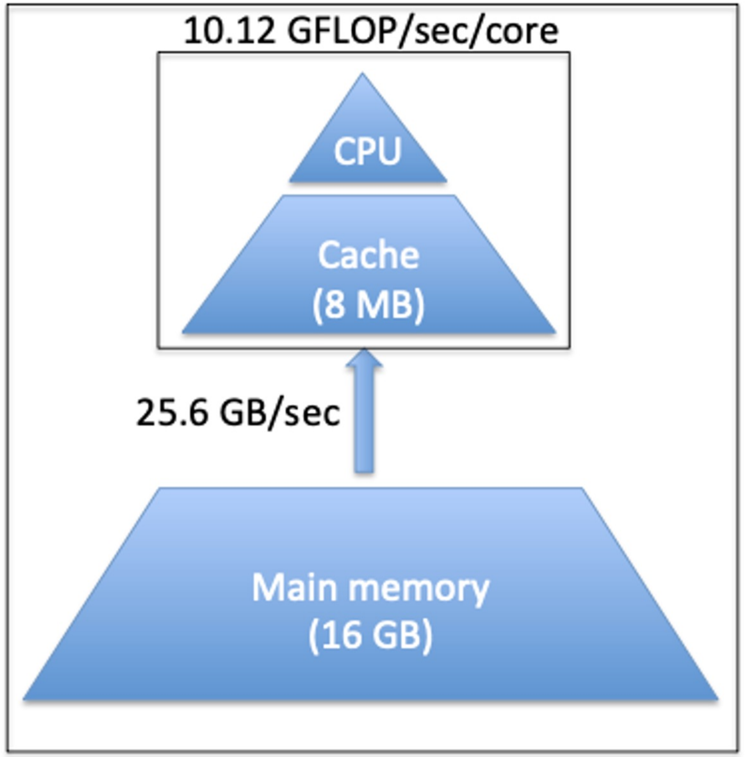
IO lower bound for computer programs



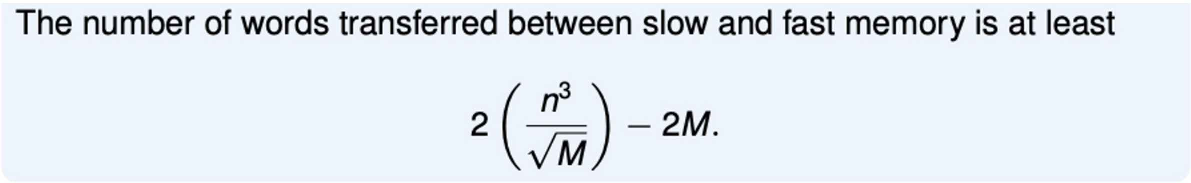
One core Intel Xeon Processor E5520 (nehalem)

$\beta^{-1} = 580 \cdot 10^6$ words/sec $\gamma^{-1} = 10.12 \cdot 10^9$ flops/sec $M = 10^6$ words

Intel Xeon Processor E5520 (Nehalem)



For MM

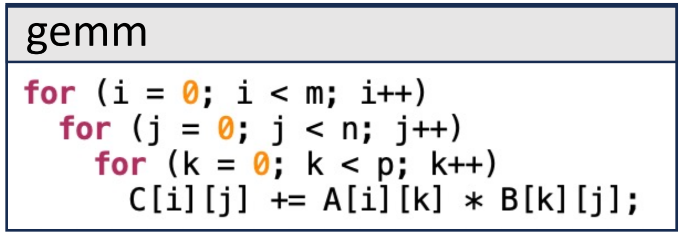


The number of words transferred between slow and fast memory is at least

$$2\left(\frac{n^3}{\sqrt{M}}\right) - 2M.$$

where n is matrix size and M is size of cache.

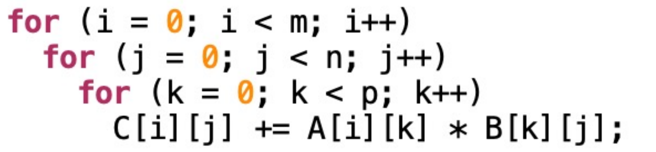
See: A Tight I/O Lower Bound for Matrix Multiplication. T. M. Smith, B. Lowery, J. Langou, R. A. van de Geijn https://arxiv.org/abs/1702.02017



gemm

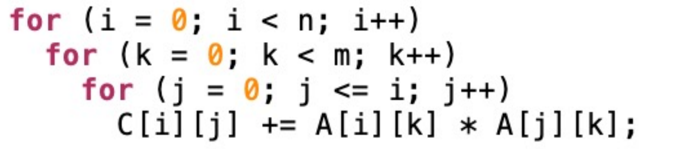
```
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < p; k++)
      C[i][j] += A[i][k] * B[k][j];
```

gemm

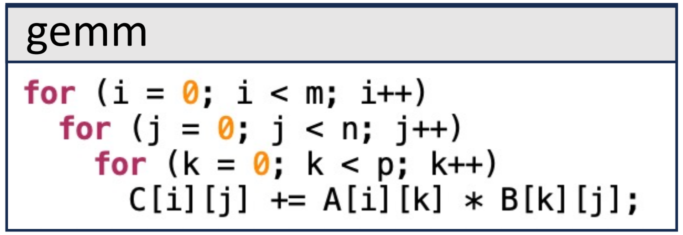


```
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < p; k++)
      C[i][j] += A[i][k] * B[k][j];
```

syrk

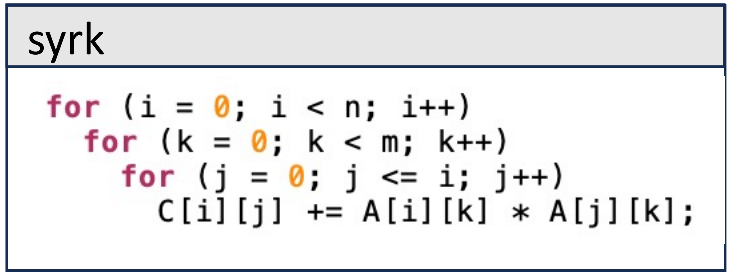


```
for (i = 0; i < n; i++)
  for (k = 0; k < m; k++)
    for (j = 0; j <= i; j++)
      C[i][j] += A[i][k] * A[j][k];
```



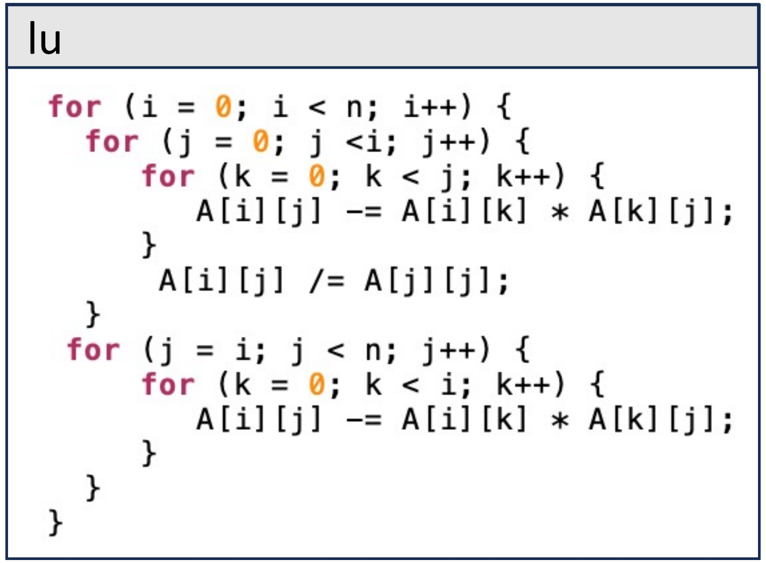
gemm

```
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < p; k++)
      C[i][j] += A[i][k] * B[k][j];
```



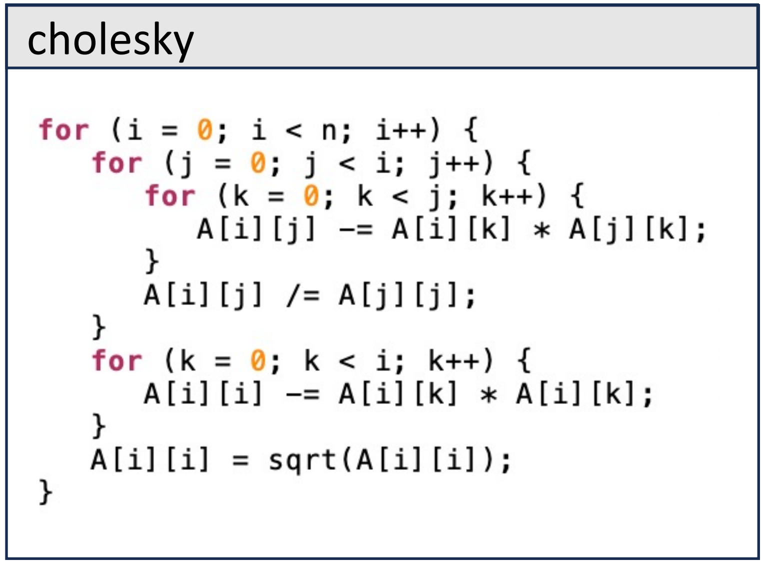
syrk

```
for (i = 0; i < n; i++)
  for (k = 0; k < m; k++)
    for (j = 0; j <= i; j++)
      C[i][j] += A[i][k] * A[j][k];
```



lu

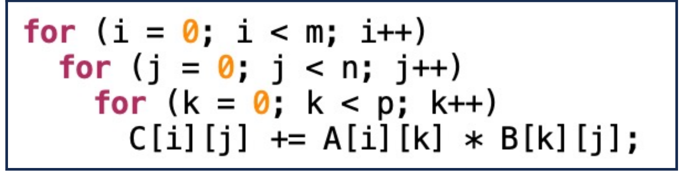
```
for (i = 0; i < n; i++) {
  for (j = 0; j <i; j++) {
     for (k = 0; k < j; k++) {
        A[i][j] -= A[i][k] * A[k][j];
     }
      A[i][j] /= A[j][j];
  }
 for (j = i; j < n; j++) {
     for (k = 0; k < i; k++) {
        A[i][j] -= A[i][k] * A[k][j];
     }
  }
}
```



cholesky

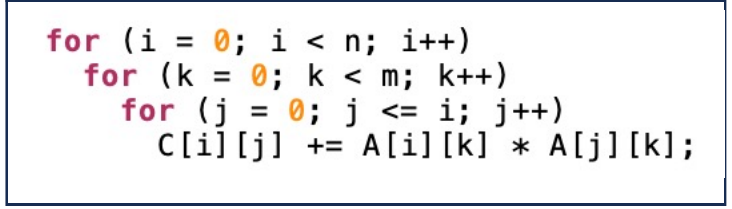
```
for (i = 0; i < n; i++) {
   for (j = 0; j < i; j++) {
      for (k = 0; k < j; k++) {
         A[i][j] -= A[i][k] * A[j][k];
      }
      A[i][j] /= A[j][j];
   }
   for (k = 0; k < i; k++) {
      A[i][i] -= A[i][k] * A[i][k];
   }
   A[i][i] = sqrt(A[i][i]);
}
```

gemm



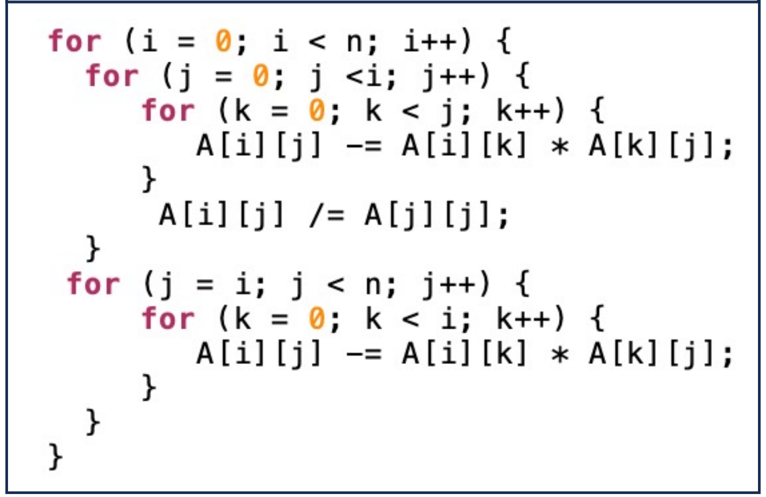
```
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < p; k++)
      C[i][j] += A[i][k] * B[k][j];
```

syrk



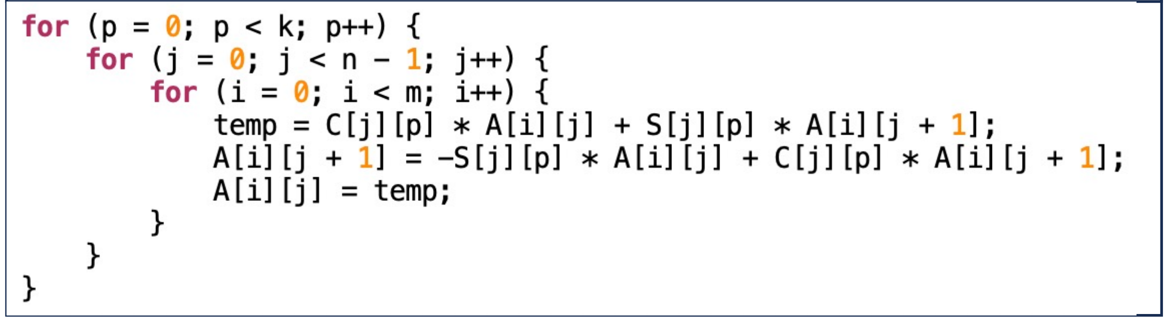
```
for (i = 0; i < n; i++)
  for (k = 0; k < m; k++)
    for (j = 0; j <= i; j++)
      C[i][j] += A[i][k] * A[j][k];
```

lu



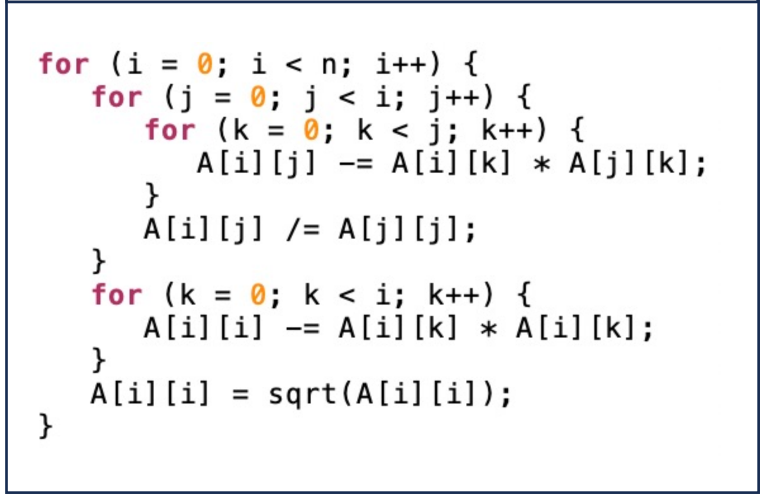
```
for (i = 0; i < n; i++) {
  for (j = 0; j <i; j++) {
     for (k = 0; k < j; k++) {
        A[i][j] -= A[i][k] * A[k][j];
     }
      A[i][j] /= A[j][j];
  }
 for (j = i; j < n; j++) {
     for (k = 0; k < i; k++) {
        A[i][j] -= A[i][k] * A[k][j];
     }
  }
}
```

apply_givens_rot_sequence



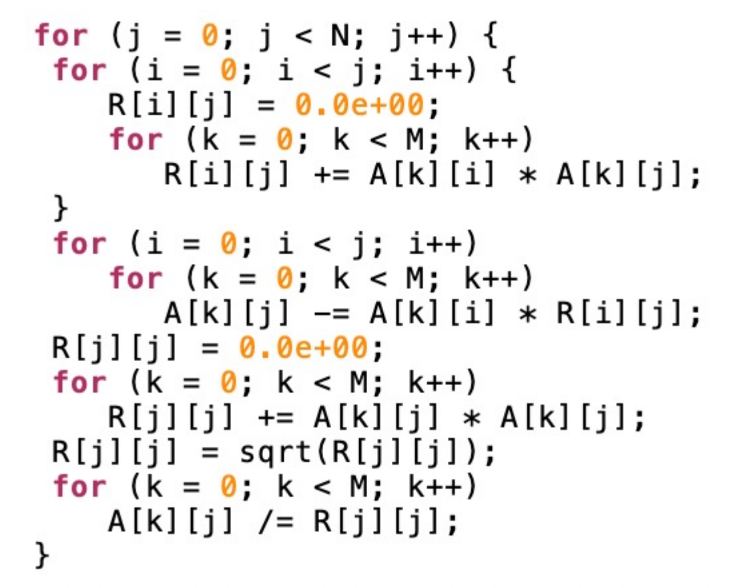
```
for (p = 0; p < k; p++) {
    for (j = 0; j < n - 1; j++) {
        for (i = 0; i < m; i++) {
            temp = C[j][p] * A[i][j] + S[j][p] * A[i][j + 1];
            A[i][j + 1] = -S[j][p] * A[i][j] + C[j][p] * A[i][j + 1];
            A[i][j] = temp;
        }
    }
}
```

cholesky



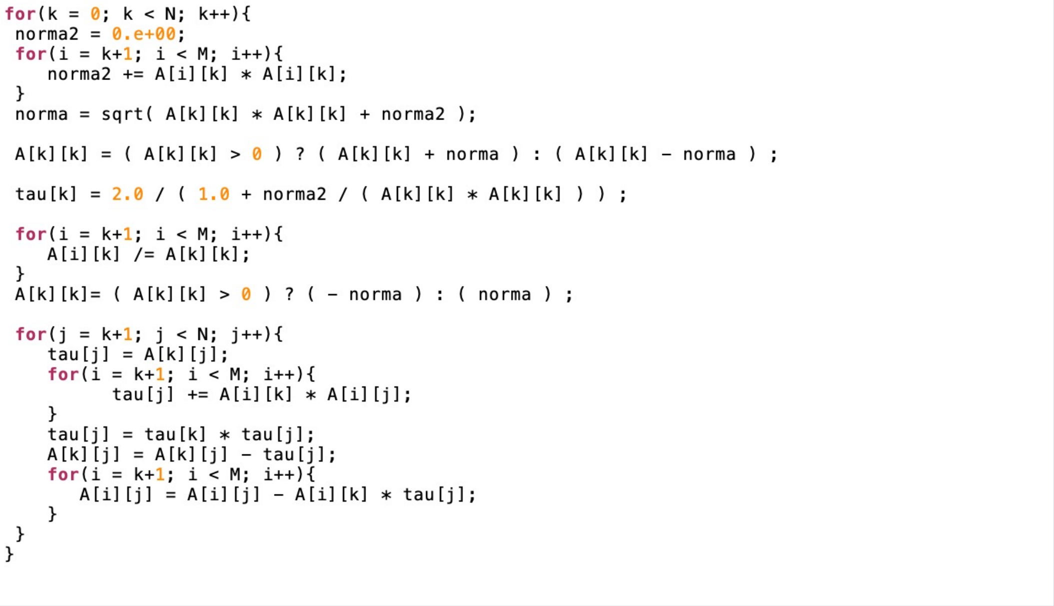
```
for (i = 0; i < n; i++) {
   for (j = 0; j < i; j++) {
      for (k = 0; k < j; k++) {
         A[i][j] -= A[i][k] * A[j][k];
      }
      A[i][j] /= A[j][j];
   }
   for (k = 0; k < i; k++) {
      A[i][i] -= A[i][k] * A[i][k];
   }
   A[i][i] = sqrt(A[i][i]);
}
```

cgs – Classical Gram-Schmidt



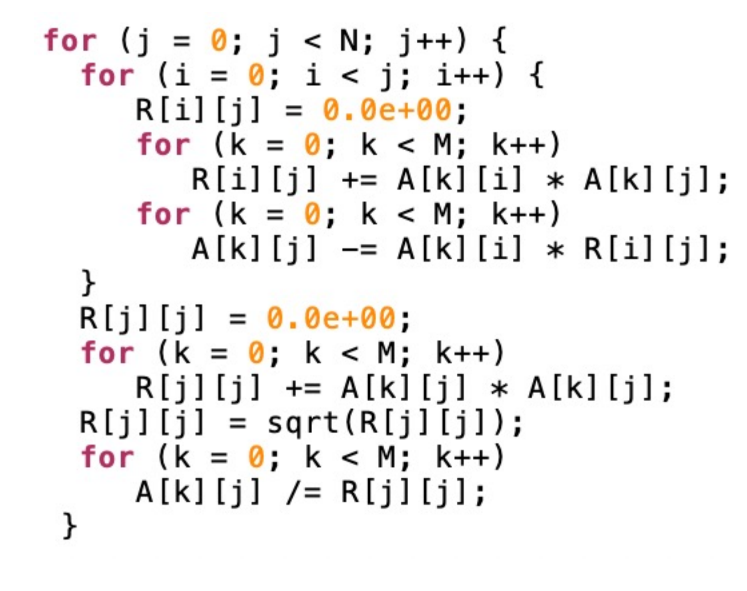
```
for (j = 0; j < N; j++) {
 for (i = 0; i < j; i++) {
    R[i][j] = 0.0e+00;
    for (k = 0; k < M; k++)
       R[i][j] += A[k][i] * A[k][j];
 }
 for (i = 0; i < j; i++)
    for (k = 0; k < M; k++)
       A[k][j] -= A[k][i] * R[i][j];
 R[j][j] = 0.0e+00;
 for (k = 0; k < M; k++)
    R[j][j] += A[k][j] * A[k][j];
 R[j][j] = sqrt(R[j][j]);
 for (k = 0; k < M; k++)
    A[k][j] /= R[j][j];
}
```

a2v – geqr2 – Householder QR factorization



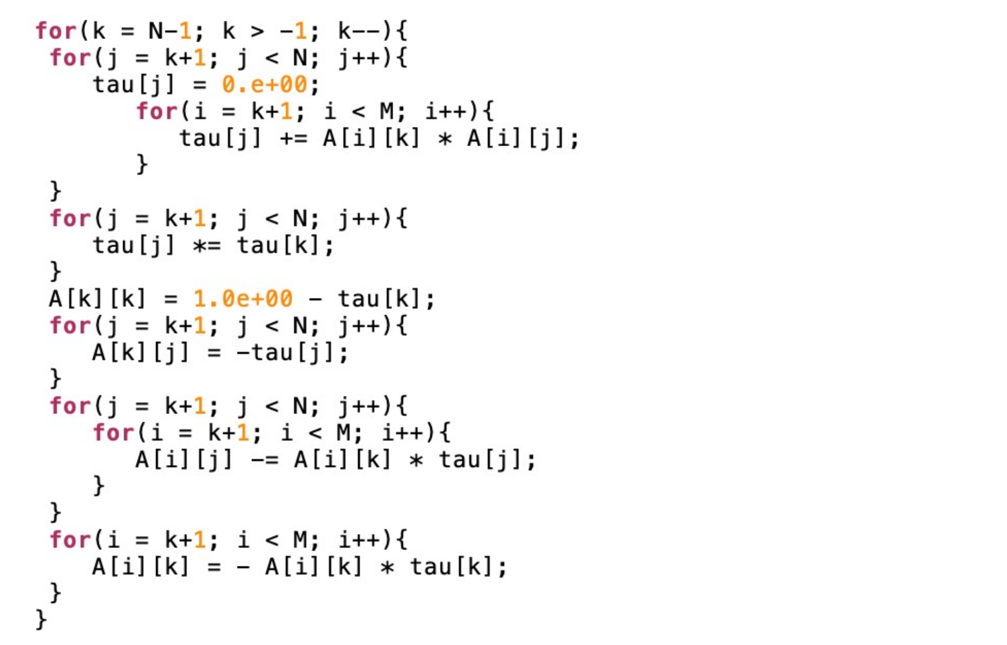
```
for(k = 0; k < N; k++){
 norma2 = 0.e+00;
 for(i = k+1; i < M; i++){
    norma2 += A[i][k] * A[i][k];
 }
 norma = sqrt( A[k][k] * A[k][k] + norma2 );

 A[k][k] = ( A[k][k] > 0 ) ? ( A[k][k] + norma ) : ( A[k][k] - norma ) ;

 tau[k] = 2.0 / ( 1.0 + norma2 / ( A[k][k] * A[k][k] ) ) ;

 for(i = k+1; i < M; i++){
    A[i][k] /= A[k][k];
 }
 A[k][k]= ( A[k][k] > 0 ) ? ( - norma ) : ( norma ) ;

 for(j = k+1; j < N; j++){
    tau[j] = A[k][j];
    for(i = k+1; i < M; i++){
          tau[j] += A[i][k] * A[i][j];
    }
    tau[j] = tau[k] * tau[j];
    A[k][j] = A[k][j] - tau[j];
    for(i = k+1; i < M; i++){
       A[i][j] = A[i][j] - A[i][k] * tau[j];
    }
 }
}
```

mgs – Modified Gram-Schmidt



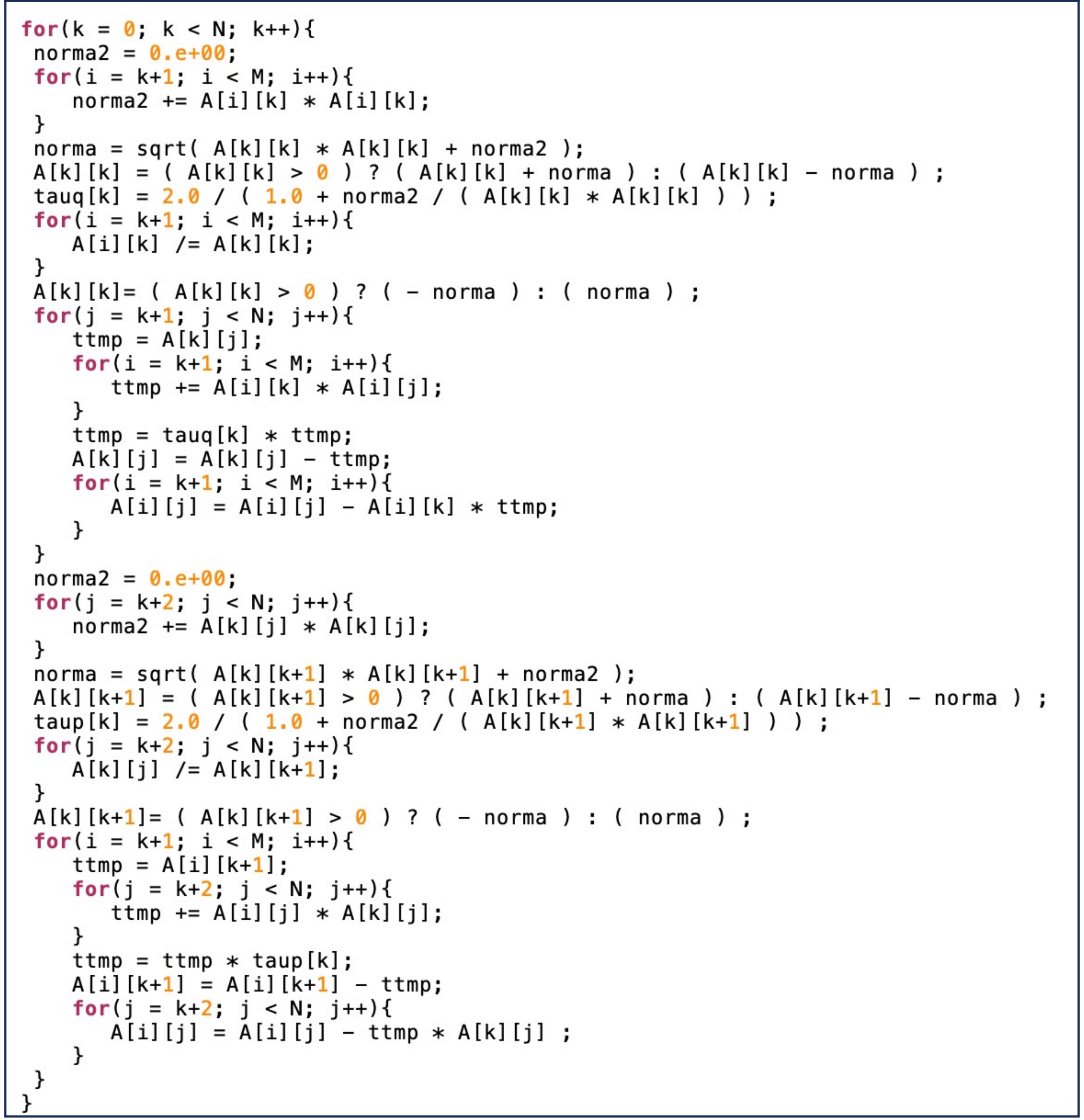
```
for (j = 0; j < N; j++) {
  for (i = 0; i < j; i++) {
     R[i][j] = 0.0e+00;
     for (k = 0; k < M; k++)
        R[i][j] += A[k][i] * A[k][j];
     for (k = 0; k < M; k++)
        A[k][j] -= A[k][i] * R[i][j];
  }
  R[j][j] = 0.0e+00;
  for (k = 0; k < M; k++)
     R[j][j] += A[k][j] * A[k][j];
  R[j][j] = sqrt(R[j][j]);
  for (k = 0; k < M; k++)
     A[k][j] /= R[j][j];
 }
```

v2q – orqr2 – Construction of Q from Householder



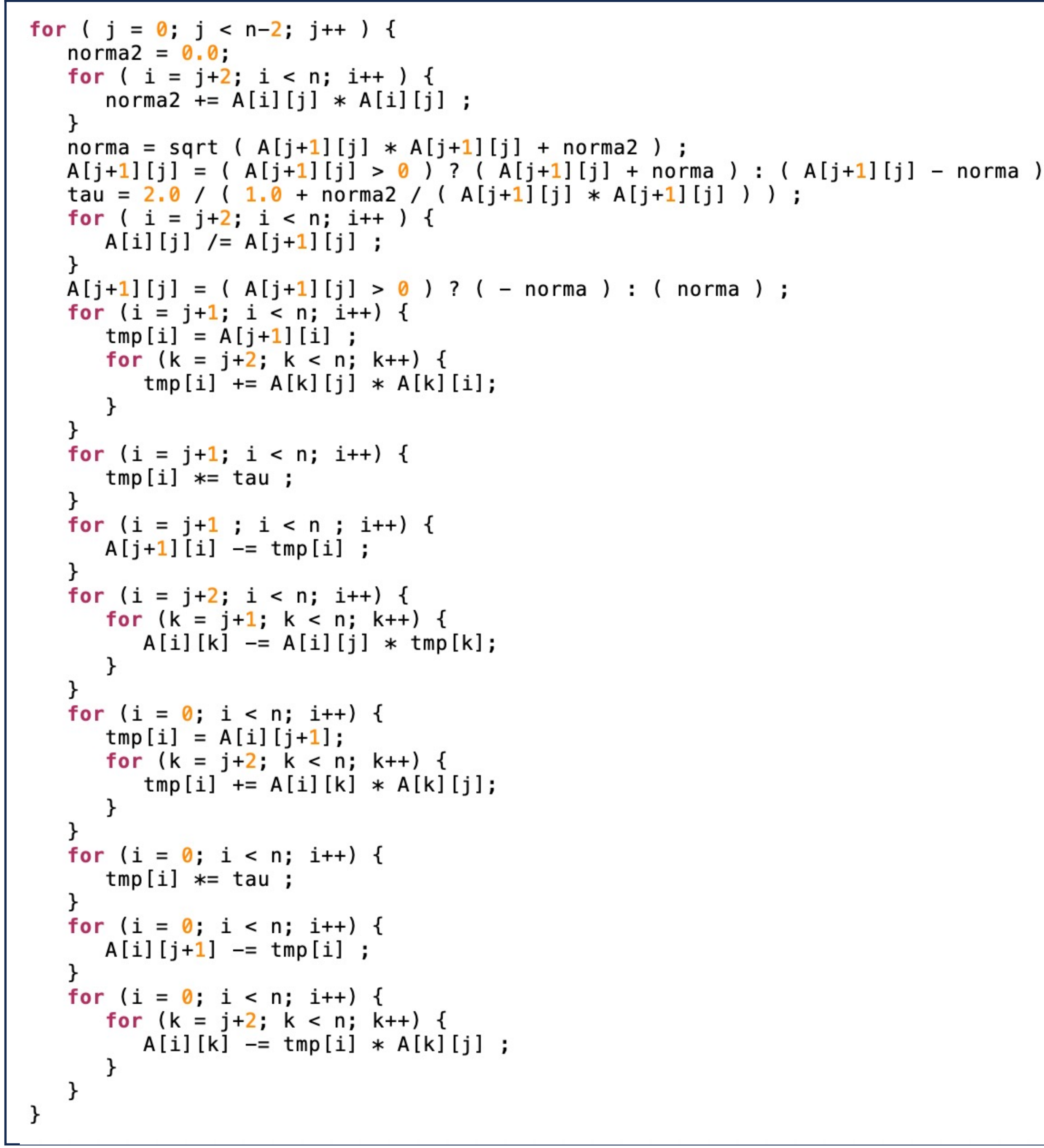
```
for(k = N-1; k > -1; k--){
 for(j = k+1; j < N; j++){
    tau[j] = 0.e+00;
       for(i = k+1; i < M; i++){
          tau[j] += A[i][k] * A[i][j];
       }
 }
 for(j = k+1; j < N; j++){
    tau[j] *= tau[k];
 }
 A[k][k] = 1.0e+00 - tau[k];
 for(j = k+1; j < N; j++){
    A[k][j] = -tau[j];
 }
 for(j = k+1; j < N; j++){
    for(i = k+1; i < M; i++){
       A[i][j] -= A[i][k] * tau[j];
    }
 }
 for(i = k+1; i < M; i++){
    A[i][k] = - A[i][k] * tau[k];
 }
}
```

gebd2 – reduction to bidiagonalization



```
for(k = 0; k < N; k++){
 norma2 = 0.e+00;
 for(i = k+1; i < M; i++){
    norma2 += A[i][k] * A[i][k];
 }
 norma = sqrt( A[k][k] * A[k][k] + norma2 );
 A[k][k] = ( A[k][k] > 0 ) ? ( A[k][k] + norma ) : ( A[k][k] - norma ) ;
 tauq[k] = 2.0 / ( 1.0 + norma2 / ( A[k][k] * A[k][k] ) ) ;
 for(i = k+1; i < M; i++){
    A[i][k] /= A[k][k];
 }
 A[k][k]= ( A[k][k] > 0 ) ? ( - norma ) : ( norma ) ;
 for(j = k+1; j < N; j++){
    ttmp = A[k][j];
    for(i = k+1; i < M; i++){
       ttmp += A[i][k] * A[i][j];
    }
    ttmp = tauq[k] * ttmp;
    A[k][j] = A[k][j] - ttmp;
    for(i = k+1; i < M; i++){
       A[i][j] = A[i][j] - A[i][k] * ttmp;
    }
 }
 norma2 = 0.e+00;
 for(j = k+2; j < N; j++){
    norma2 += A[k][j] * A[k][j];
 }
 norma = sqrt( A[k][k+1] * A[k][k+1] + norma2 );
 A[k][k+1] = ( A[k][k+1] > 0 ) ? ( A[k][k+1] + norma ) : ( A[k][k+1] - norma ) ;
 taup[k] = 2.0 / ( 1.0 + norma2 / ( A[k][k+1] * A[k][k+1] ) ) ;
 for(j = k+2; j < N; j++){
    A[k][j] /= A[k][k+1];
 }
 A[k][k+1]= ( A[k][k+1] > 0 ) ? ( - norma ) : ( norma ) ;
 for(i = k+1; i < M; i++){
    ttmp = A[i][k+1];
    for(j = k+2; j < N; j++){
       ttmp += A[i][j] * A[k][j];
    }
    ttmp = ttmp * taup[k];
    A[i][k+1] = A[i][k+1] - ttmp;
    for(j = k+2; j < N; j++){
       A[i][j] = A[i][j] - ttmp * A[k][j] ;
    }
 }
}
```

gehd2 – reduction to Hessenberg form



```
for ( j = 0; j < n-2; j++ ) {
   norma2 = 0.0;
   for ( i = j+2; i < n; i++ ) {
      norma2 += A[i][j] * A[i][j] ;
   }
   norma = sqrt ( A[j+1][j] * A[j+1][j] + norma2 ) ;
   A[j+1][j] = ( A[j+1][j] > 0 ) ? ( A[j+1][j] + norma ) : ( A[j+1][j] - norma )
   tau = 2.0 / ( 1.0 + norma2 / ( A[j+1][j] * A[j+1][j] ) ) ;
   for ( i = j+2; i < n; i++ ) {
      A[i][j] /= A[j+1][j] ;
   }
   A[j+1][j] = ( A[j+1][j] > 0 ) ? ( - norma ) : ( norma ) ;
   for (i = j+1; i < n; i++) {
      tmp[i] = A[j+1][i] ;
      for (k = j+2; k < n; k++) {
         tmp[i] += A[k][j] * A[k][i];
      }
   }
   for (i = j+1; i < n; i++) {
      tmp[i] *= tau ;
   }
   for (i = j+1 ; i < n ; i++) {
      A[j+1][i] -= tmp[i] ;
   }
   for (i = j+2; i < n; i++) {
      for (k = j+1; k < n; k++) {
         A[i][k] -= A[i][j] * tmp[k];
      }
   }
   for (i = 0; i < n; i++) {
      tmp[i] = A[i][j+1];
      for (k = j+2; k < n; k++) {
         tmp[i] += A[i][k] * A[k][j];
      }
   }
   for (i = 0; i < n; i++) {
      tmp[i] *= tau ;
   }
   for (i = 0; i < n; i++) {
      A[i][j+1] -= tmp[i] ;
   }
   for (i = 0; i < n; i++) {
      for (k = j+2; k < n; k++) {
         A[i][k] -= tmp[i] * A[k][j] ;
      }
   }
}
```

sytd2 – reduction to symmetric tridiagonal form

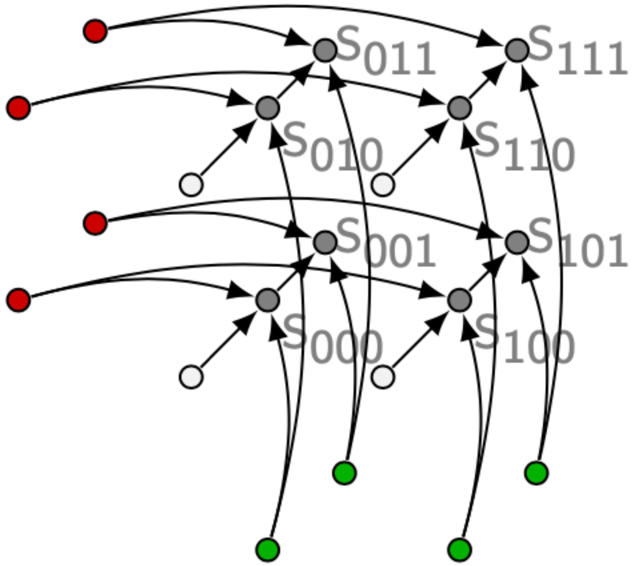
```
for(i = 0; i < n-2; i++){
   norma2 = 0.0e+00 ;
   for ( k = i+2; k < n ; k++ ) {
      norma2 += A[k][i] * A[k][i];
   }
   norma = sqrt( A[i+1][i] * A[i+1][i] + norma2 ) ;
   A[i+1][i] = ( A[i+1][i] > 0 ) ? (A[i+1][i] + norma) : (A[i+1][i] - norma) ;
   taui = 2.0e+00 / ( 1.0e+00 + norma2 / ( A[i+1][i] * A[i+1][i] ) ) ;

   for (k = i+2; k < n ; k++) {
      A[k][i] /= A[i+1][i] ;
   }
   A[i+1][i] = ( A[i+1][i] > 0.0e+00 ) ? ( - norma ) : ( norma ) ;
   for(k = i+1; k < n; k++){
      tau[k-1] = A[k][i+1];
      for(j = i+2; j <= k; j++){
         tau[k-1] += A[k][j] * A[j][i];
      }
      for(j = k+1; j < n; j++){
         tau[k-1] += A[j][k] * A[j][i];
      }
      tau[k-1] *= taui;
   }
   alpha = tau[i];
   for(k = i+2; k < n; k++){
      alpha += tau[k-1] * A[k][i];
   }
   alpha = -0.5e+00 * taui * alpha;
   tau[i] += alpha ;
   for(k = i+2; k < n; k++){
      tau[k-1] += alpha * A[k][i];
   }
   A[i+1][i+1] -= 2.0e+00 * tau[i] ;
   for(j = i+2; j < n; j++){
      A[j][i+1] -= tau[j-1] ;
      A[j][i+1] -= tau[i] * A[j][i];
      for(k = i+2; k <= j; k++){
         A[j][k] -= tau[j-1] * A[k][i];
         A[j][k] -= tau[k-1] * A[j][i];
      }
   }
   tau[i] = taui;
 }
```

gghd2 – reduction to triangular Hessenberg form

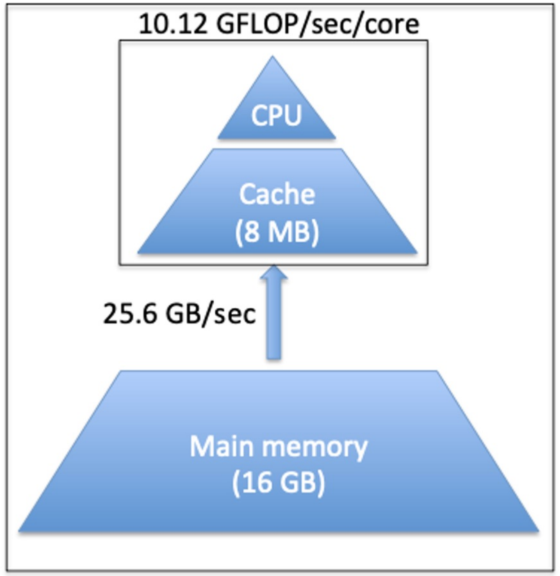
```
for (j = 0; j < _PB_N-2; j++) {
  for (i = _PB_N-2; i > j; i--) {
    nrm = SQRT_FUN ( A[i][j] * A[i][j] + A[i+1][j] * A[i+1][j] );
    c = A[i][j] / nrm;
    s = A[i+1][j] / nrm;
    A[i][j] = nrm;
    A[i+1][j] = SCALAR_VAL(0.0);
    for (k = j+1; k < _PB_N; k++) {
      tmp = c * A[i][k] + s * A[i+1][k];
      A[i+1][k] = - s * A[i][k] + c * A[i+1][k];
      A[i][k] = tmp;
    }
    for (k = i; k < _PB_N; k++) {
      tmp = c * B[i][k] + s * B[i+1][k];
      B[i+1][k] = - s * B[i][k] + c * B[i+1][k];
      B[i][k] = tmp;
    }
    for (k = 0; k < _PB_N; k++) {
      tmp = c * Q[i][k] + s * Q[i+1][k];
      Q[i+1][k] = - s * Q[i][k] + c * Q[i+1][k];
      Q[i][k] = tmp;
    }
    nrm = SQRT_FUN ( B[i+1][i+1] * B[i+1][i+1] + B[i+1][i] * B[i+1][i] );
    c = B[i+1][i+1] / nrm;
    s = B[i+1][i] / nrm;
    B[i+1][i+1] = nrm;
    B[i+1][i] = SCALAR_VAL(0.0);
    for (k = 0; k <= i; k++) {
      tmp = c * B[k][i] - s * B[k][i+1];
      B[k][i+1] = s * B[k][i] + c * B[k][i+1];
      B[k][i] = tmp;
    }
    for (k = 0; k < _PB_N; k++) {
      tmp = c * A[k][i] - s * A[k][i+1];
      A[k][i+1] = s * A[k][i] + c * A[k][i+1];
      A[k][i] = tmp;
    }
    for (k = i-j; k < _PB_N; k++) {
      tmp = c * Z[k][i] - s * Z[k][i+1];
      Z[k][i+1] = s * Z[k][i] + c * Z[k][i+1];
      Z[k][i] = tmp;
    }
  }
}
```

```
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < p; k++)
      C[i][j] += A[i][k] * B[k][j];
```

```
C[0][0] += A[0][0] * B[0][0];  // S000
C[0][0] += A[0][1] * B[1][0];  // S001
C[0][0] += A[0][2] * B[2][0];  // S002
C[0][0] += A[0][3] * B[3][0];  // S003
…
C[0][1] += A[0][0] * B[0][1];  // S010
C[0][1] += A[0][1] * B[1][1];  // S011
C[0][1] += A[0][2] * B[2][1];  // S012
C[0][1] += A[0][3] * B[3][1];  // S013
```



Intel Xeon Processor E5520 (Nehalem)



10.12 GFLOP/sec/core

CPU

Cache (8 MB)

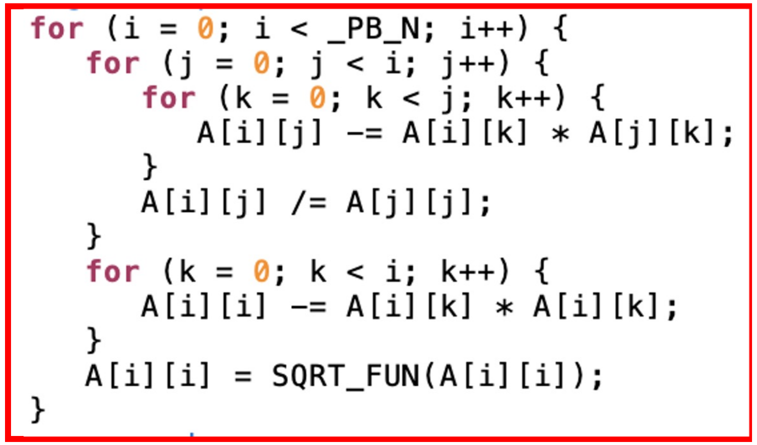
25.6 GB/sec

Main memory (16 GB)

gemm

A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)

input is an affine code



```
for (i = 0; i < _PB_N; i++) {
   for (j = 0; j < i; j++) {
      for (k = 0; k < j; k++) {
         A[i][j] -= A[i][k] * A[j][k];
      }
      A[i][j] /= A[j][j];
   }
   for (k = 0; k < i; k++) {
      A[i][i] -= A[i][k] * A[i][k];
   }
   A[i][i] = SQRT_FUN(A[i][i]);
}
```

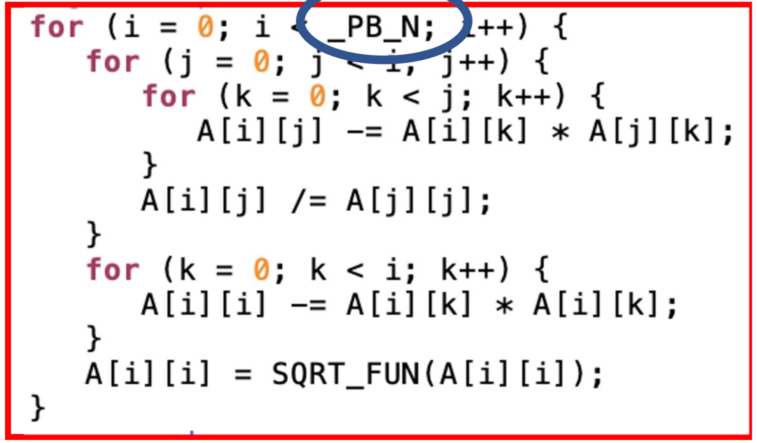
cholesky code

Cholesky

A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)

input is an affine code

with parameter **N** and parameter **S** for cache size



```
for (i = 0; i < _PB_N; i++) {
   for (j = 0; j < i; j++) {
      for (k = 0; k < j; k++) {
         A[i][j] -= A[i][k] * A[j][k];
      }
      A[i][j] /= A[j][j];
   }
   for (k = 0; k < i; k++) {
      A[i][i] -= A[i][k] * A[i][k];
   }
   A[i][i] = SQRT_FUN(A[i][i]);
}
```

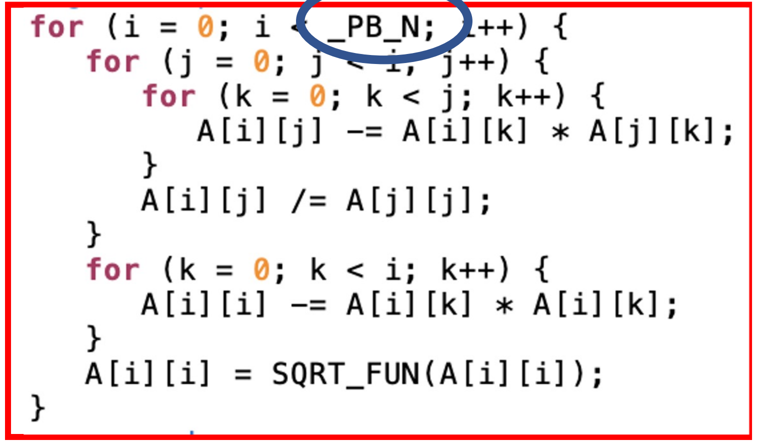
cholesky code

Unit is for **N** for, **S** for and IO lower bound is “number”.

Cholesky

A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)

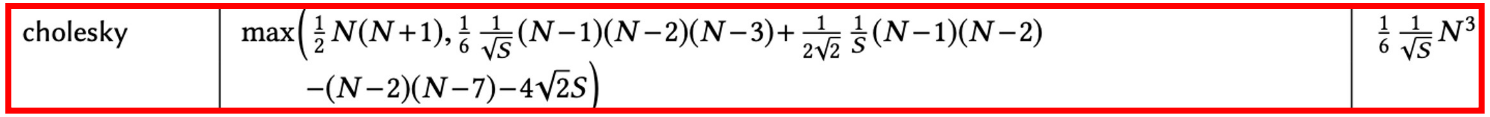
input is an affine code with parameter **N** and parameter **S** for cache size



```
for (i = 0; i < _PB_N; i++) {
   for (j = 0; j < i; j++) {
      for (k = 0; k < j; k++) {
         A[i][j] -= A[i][k] * A[j][k];
      }
      A[i][j] /= A[j][j];
   }
   for (k = 0; k < i; k++) {
      A[i][i] -= A[i][k] * A[i][k];
   }
   A[i][i] = SQRT_FUN(A[i][i]);
}
```

cholesky code

Output #1 is an IO lower bound



cholesky	$\max\left(\frac{1}{2}N(N+1), \frac{1}{6}\frac{1}{\sqrt{S}}(N-1)(N-2)(N-3)+\frac{1}{2\sqrt{2}}\frac{1}{S}(N-1)(N-2) -(N-2)(N-7)-4\sqrt{2}S\right)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$

IO lower bound
with parameter **N** and parameter **S** for cache size

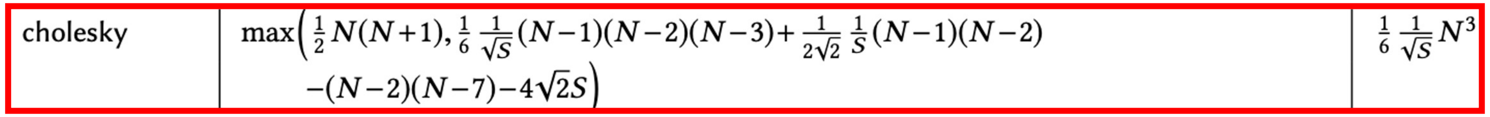
asymptotic
IO lower bound

Unit is for **N** for, **S** for and IO lower bound is “number”.

Cholesky

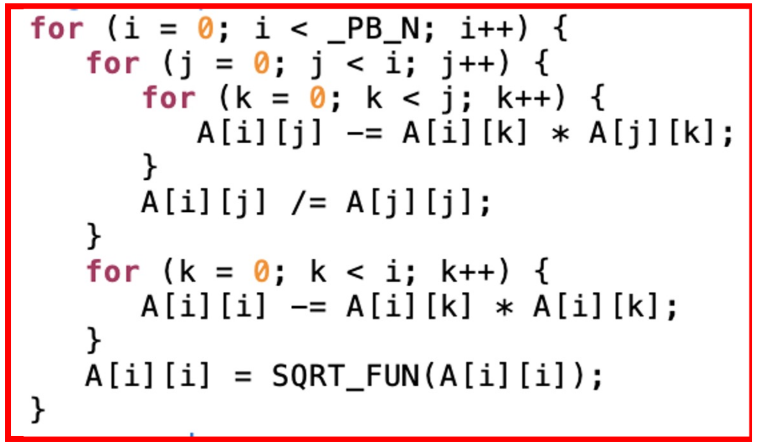
A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)

Output #1 is an IO lower bound



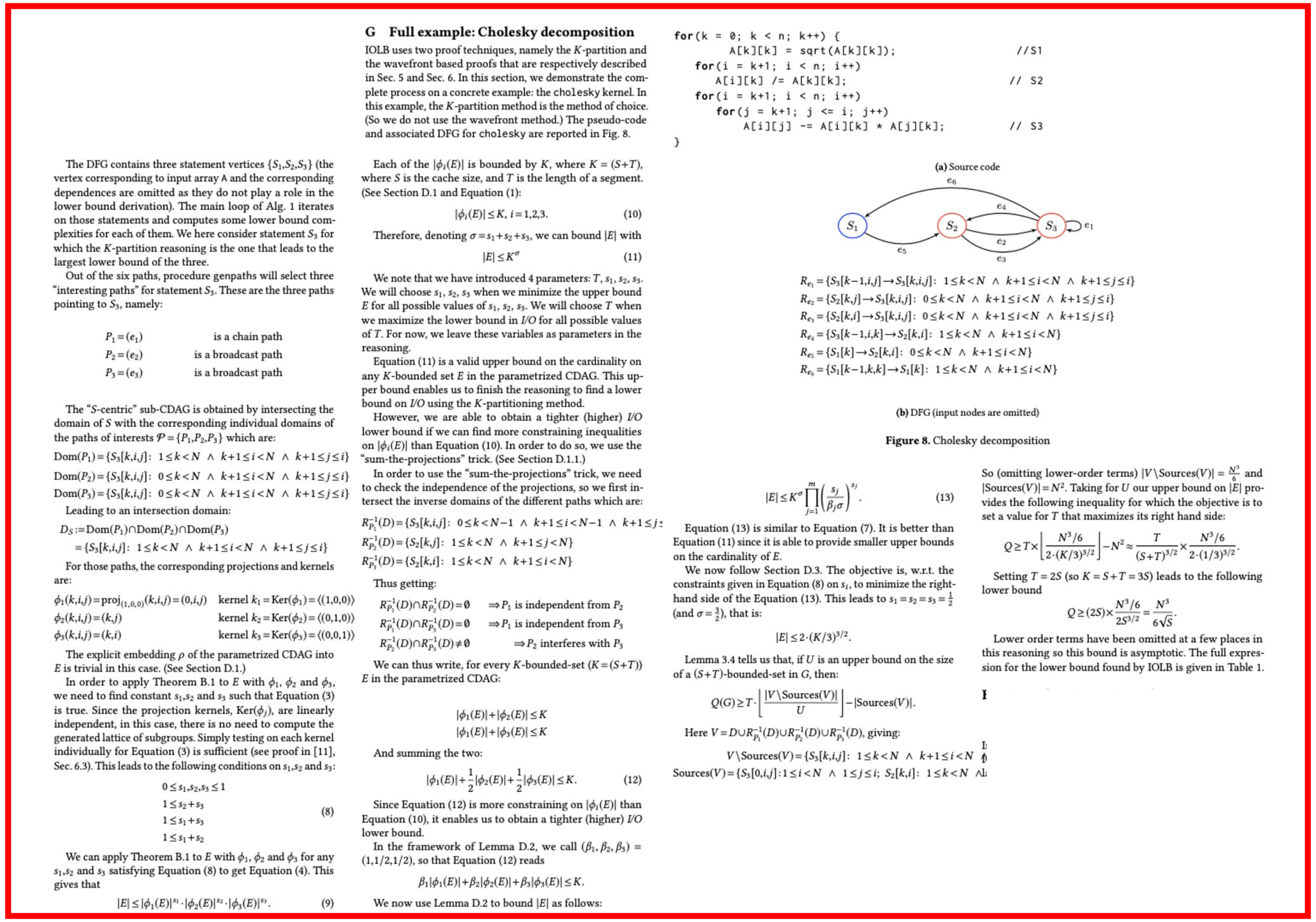
cholesky	$\max\left(\frac{1}{2}N(N+1), \frac{1}{6}\frac{1}{\sqrt{S}}(N-1)(N-2)(N-3)+\frac{1}{2\sqrt{2}}\frac{1}{S}(N-1)(N-2) -(N-2)(N-7)-4\sqrt{2}S\right)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$

input is an affine code



```
for (i = 0; i < _PB_N; i++) {
   for (j = 0; j < i; j++) {
      for (k = 0; k < j; k++) {
         A[i][j] -= A[i][k] * A[j][k];
      }
      A[i][j] /= A[j][j];
   }
   for (k = 0; k < i; k++) {
      A[i][i] -= A[i][k] * A[i][k];
   }
   A[i][i] = SQRT_FUN(A[i][i]);
}
```

cholesky code



G Full example: Cholesky decomposition

IOLB uses two proof techniques, namely the K-partition and the wavefront based proofs that are respectively described in Sec. 5 and Sec. 6. In this section, we demonstrate the complete process on a concrete example: the `cholesky` kernel. In this example, the K-partition method is the method of choice. (So we do not use the wavefront method.) The pseudo-code and associated DFG for `cholesky` are reported in Fig. 8.

The DFG contains three statement vertices $\{S_1,S_2,S_3\}$ (the vertex corresponding to input array A and the corresponding dependences are omitted as they do not play a role in the lower bound derivation). The main loop of Alg. 1 iterates on those statements and computes some lower bound complexities for each of them. We here consider statement S_3 for which the K-partition reasoning is the one that leads to the largest lower bound of the three.

Out of the six paths, procedure genpaths will select three "interesting paths" for statement S_3. These are the three paths pointing to S_3, namely:

$$P_1=(e_1) \quad \text{is a chain path}$$
$$P_2=(e_2) \quad \text{is a broadcast path}$$
$$P_3=(e_3) \quad \text{is a broadcast path}$$

The "S-centric" sub-CDAG is obtained by intersecting the domain of S with the corresponding individual domains of the paths of interests $\mathcal{P}=\{P_1,P_2,P_3\}$ which are:

$$\mathrm{Dom}(P_1)=\{S_3[k,i,j]:\ 1\le k<N \ \wedge\ k+1\le i<N \ \wedge\ k+1\le j\le i\}$$
$$\mathrm{Dom}(P_2)=\{S_3[k,i,j]:\ 0\le k<N \ \wedge\ k+1\le i<N \ \wedge\ k+1\le j\le i\}$$
$$\mathrm{Dom}(P_3)=\{S_3[k,i,j]:\ 0\le k<N \ \wedge\ k+1\le i<N \ \wedge\ k+1\le j\le i\}$$

Leading to an intersection domain:

$$D_S:=\mathrm{Dom}(P_1)\cap\mathrm{Dom}(P_2)\cap\mathrm{Dom}(P_3)$$
$$=\{S_3[k,i,j]:\ 1\le k<N \ \wedge\ k+1\le i<N \ \wedge\ k+1\le j\le i\}$$

For those paths, the corresponding projections and kernels are:

$$\phi_1(k,i,j)=\mathrm{proj}_{(1,0,0)}(k,i,j)=(0,i,j) \quad \text{kernel } k_1=\mathrm{Ker}(\phi_1)=\langle(1,0,0)\rangle$$
$$\phi_2(k,i,j)=(k,j) \quad \text{kernel } k_2=\mathrm{Ker}(\phi_2)=\langle(0,1,0)\rangle$$
$$\phi_3(k,i,j)=(k,i) \quad \text{kernel } k_3=\mathrm{Ker}(\phi_3)=\langle(0,0,1)\rangle$$

The explicit embedding ρ of the parametrized CDAG into E is trivial in this case. (See Section D.1.)

In order to apply Theorem B.1 to E with ϕ_1, ϕ_2 and ϕ_3, we need to find constant s_1,s_2 and s_3 such that Equation (3) is true. Since the projection kernels, $\mathrm{Ker}(\phi_j)$, are linearly independent, in this case, there is no need to compute the generated lattice of subgroups. Simply testing on each kernel individually for Equation (3) is sufficient (see proof in [11], Sec. 6.3). This leads to the following conditions on s_1,s_2 and s_3:

$$\begin{aligned} &0\le s_1,s_2,s_3\le 1\\ &1\le s_2+s_3\\ &1\le s_1+s_3\\ &1\le s_1+s_2 \end{aligned} \tag{8}$$

We can apply Theorem B.1 to E with ϕ_1, ϕ_2 and ϕ_3 for any s_1,s_2 and s_3 satisfying Equation (8) to get Equation (4). This gives that

$$|E|\le|\phi_1(E)|^{s_1}\cdot|\phi_2(E)|^{s_2}\cdot|\phi_3(E)|^{s_3}. \tag{9}$$

Each of the $|\phi_i(E)|$ is bounded by K, where $K=(S+T)$, where S is the cache size, and T is the length of a segment. (See Section D.1 and Equation (1):

$$|\phi_i(E)|\le K,\ i=1,2,3. \tag{10}$$

Therefore, denoting $\sigma=s_1+s_2+s_3$, we can bound $|E|$ with

$$|E|\le K^{\sigma} \tag{11}$$

We note that we have introduced 4 parameters: T, s_1, s_2, s_3. We will choose s_1, s_2, s_3 when we minimize the upper bound E for all possible values of s_1, s_2, s_3. We will choose T when we maximize the lower bound in I/O for all possible values of T. For now, we leave these variables as parameters in the reasoning.

Equation (11) is a valid upper bound on the cardinality on any K-bounded set E in the parametrized CDAG. This upper bound enables us to finish the reasoning to find a lower bound on I/O using the K-partitioning method.

However, we are able to obtain a tighter (higher) I/O lower bound if we can find more constraining inequalities on $|\phi_i(E)|$ than Equation (10). In order to do so, we use the "sum-the-projections" trick. (See Section D.1.1.)

In order to use the "sum-the-projections" trick, we need to check the independence of the projections, so we first intersect the inverse domains of the different paths which are:

$$R_{P_1}^{-1}(D)=\{S_3[k,i,j]:\ 0\le k<N-1 \ \wedge\ k+1\le i<N-1 \ \wedge\ k+1\le j\le$$
$$R_{P_2}^{-1}(D)=\{S_2[k,j]:\ 1\le k<N \ \wedge\ k+1\le j<N\}$$
$$R_{P_3}^{-1}(D)=\{S_2[k,i]:\ 1\le k<N \ \wedge\ k+1\le i<N\}$$

Thus getting:

$$R_{P_1}^{-1}(D)\cap R_{P_2}^{-1}(D)=\emptyset \quad \Rightarrow P_1 \text{ is independent from } P_2$$
$$R_{P_1}^{-1}(D)\cap R_{P_3}^{-1}(D)=\emptyset \quad \Rightarrow P_1 \text{ is independent from } P_3$$
$$R_{P_2}^{-1}(D)\cap R_{P_3}^{-1}(D)\neq\emptyset \quad \Rightarrow P_2 \text{ interferes with } P_3$$

We can thus write, for every K-bounded-set $(K=(S+T))$ E in the parametrized CDAG:

$$|\phi_1(E)|+|\phi_2(E)|\le K$$
$$|\phi_1(E)|+|\phi_3(E)|\le K$$

And summing the two:

$$|\phi_1(E)|+\frac{1}{2}|\phi_2(E)|+\frac{1}{2}|\phi_3(E)|\le K. \tag{12}$$

Since Equation (12) is more constraining on $|\phi_i(E)|$ than Equation (10), it enables us to obtain a tighter (higher) I/O lower bound.

In the framework of Lemma D.2, we call $(\beta_1,\beta_2,\beta_3) = (1,1/2,1/2)$, so that Equation (12) reads

$$\beta_1|\phi_1(E)|+\beta_2|\phi_2(E)|+\beta_3|\phi_3(E)|\le K.$$

We now use Lemma D.2 to bound $|E|$ as follows:

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                    //S1
   for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                          // S2
   for(i = k+1; i < n; i++)
      for(j = k+1; j <= i; j++)
         A[i][j] -= A[i][k] * A[j][k];             // S3
}
```

(a) Source code

$$R_{e_1}=\{S_3[k-1,i,j]\to S_3[k,i,j]:\ 1\le k<N \ \wedge\ k+1\le i<N \ \wedge\ k+1\le j\le i\}$$
$$R_{e_2}=\{S_2[k,j]\to S_3[k,i,j]:\ 0\le k<N \ \wedge\ k+1\le i<N \ \wedge\ k+1\le j\le i\}$$
$$R_{e_3}=\{S_2[k,i]\to S_3[k,i,j]:\ 0\le k<N \ \wedge\ k+1\le i<N \ \wedge\ k+1\le j\le i\}$$
$$R_{e_4}=\{S_3[k-1,i,k]\to S_2[k,i]:\ 1\le k<N \ \wedge\ k+1\le i<N\}$$
$$R_{e_5}=\{S_1[k]\to S_2[k,i]:\ 0\le k<N \ \wedge\ k+1\le i<N\}$$
$$R_{e_6}=\{S_1[k-1,k,k]\to S_1[k]:\ 1\le k<N \ \wedge\ k+1\le i<N\}$$

(b) DFG (input nodes are omitted)

Figure 8. Cholesky decomposition

$$|E|\le K^{\sigma}\prod_{j=1}^{m}\left(\frac{s_j}{\beta_j\sigma}\right)^{s_j}. \tag{13}$$

Equation (13) is similar to Equation (7). It is better than Equation (11) since it is able to provide smaller upper bounds on the cardinality of E.

We now follow Section D.3. The objective is, w.r.t. the constraints given in Equation (8) on s_i, to minimize the right-hand side of the Equation (13). This leads to $s_1=s_2=s_3=\frac{1}{2}$ (and $\sigma=\frac{3}{2}$), that is:

$$|E|\le 2\cdot(K/3)^{3/2}.$$

Lemma 3.4 tells us that, if U is an upper bound on the size of a $(S+T)$-bounded-set in G, then:

$$Q(G)\ge T\cdot\left\lfloor\frac{|V\setminus\mathrm{Sources}(V)|}{U}\right\rfloor-|\mathrm{Sources}(V)|.$$

Here $V=D\cup R_{P_1}^{-1}(D)\cup R_{P_2}^{-1}(D)\cup R_{P_3}^{-1}(D)$, giving:

$$V\setminus\mathrm{Sources}(V)=\{S_3[k,i,j]:\ 1\le k<N \ \wedge\ k+1\le i<N$$
$$\mathrm{Sources}(V)=\{S_3[0,i,j]:1\le i<N \ \wedge\ 1\le j\le i;\ S_2[k,i]:\ 1\le k<N \ \wedge$$

So (omitting lower-order terms) $|V\setminus\mathrm{Sources}(V)|=\frac{N^3}{6}$ and $|\mathrm{Sources}(V)|=N^2$. Taking for U our upper bound on $|E|$ provides the following inequality for which the objective is to set a value for T that maximizes its right hand side:

$$Q\ge T\times\left\lfloor\frac{N^3/6}{2\cdot(K/3)^{3/2}}\right\rfloor-N^2\approx\frac{T}{(S+T)^{3/2}}\times\frac{N^3/6}{2\cdot(1/3)^{3/2}}.$$

Setting $T=2S$ (so $K=S+T=3S$) leads to the following lower bound

$$Q\ge(2S)\times\frac{N^3/6}{2S^{3/2}}=\frac{N^3}{6\sqrt{S}}.$$

Lower order terms have been omitted at a few places in this reasoning so this bound is asymptotic. The full expression for the lower bound found by IOLB is given in Table 1.

Output #2 is a proof

Cholesky

We consider a segmentation of the execution every T IOs. For each segment, we have at most K = S + T “input” data to perform the statements in the segment.

We consider a segmentation of the execution every T IOs. For each segment, we have at most K = S + T input data to perform the statements in the segment.

the maximum number of statements in a segment

We consider a segmentation of the execution every T IOs. For each segment, we have at most K = S + T input data to perform the statements in the segment.

$$2 \cdot (K/3)^{3/2}.$$

the maximum number of statements in a segment

Cholesky

We consider a segmentation of the execution every T IOs. For each segment, we have at most K = S + T input data to perform the statements in the segment.

the total number of statements in the execution

the IO of a segment

$$Q \geq T \times \left\lfloor \frac{N^3/6}{2 \cdot (K/3)^{3/2}} \right\rfloor$$

The minimum number of segments in a schedule

the maximum number of statements in a segment

is greater than or equal to

the IO of any schedule

Cholesky

We consider a segmentation of the execution every T IOs. For each segment, we have at most K = S + T input data to perform the statements in the segment.

$$2 \cdot (K/3)^{3/2} .$$

↑

the maximum number of statements in a segment

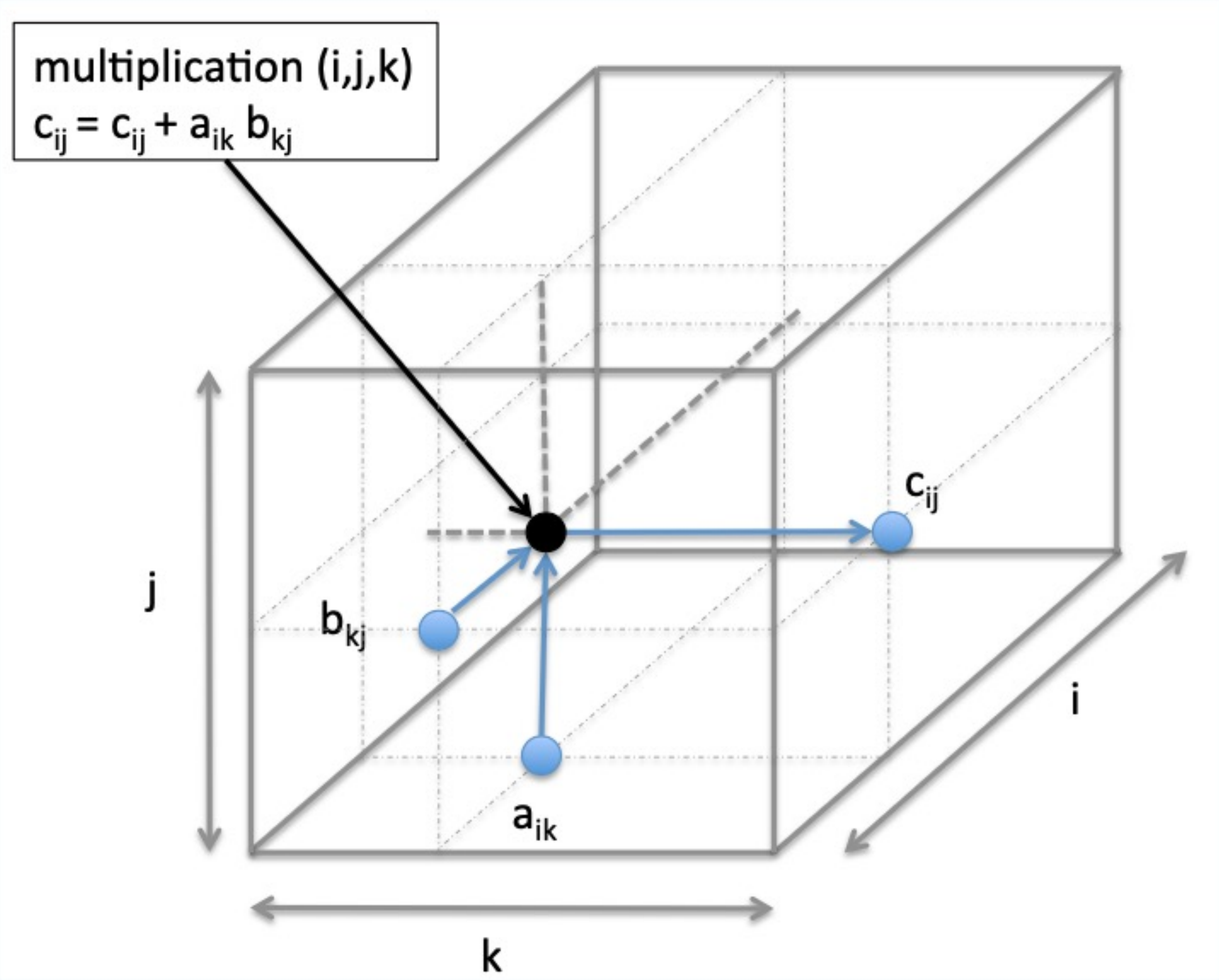
Cholesky

What do we want to compute?

We want to compute the n^3

$$c_{ijk} = a_{ik} b_{kj} .$$

The computation of c_{ijk} requires a_{ik} and b_{kj} to be in cache.

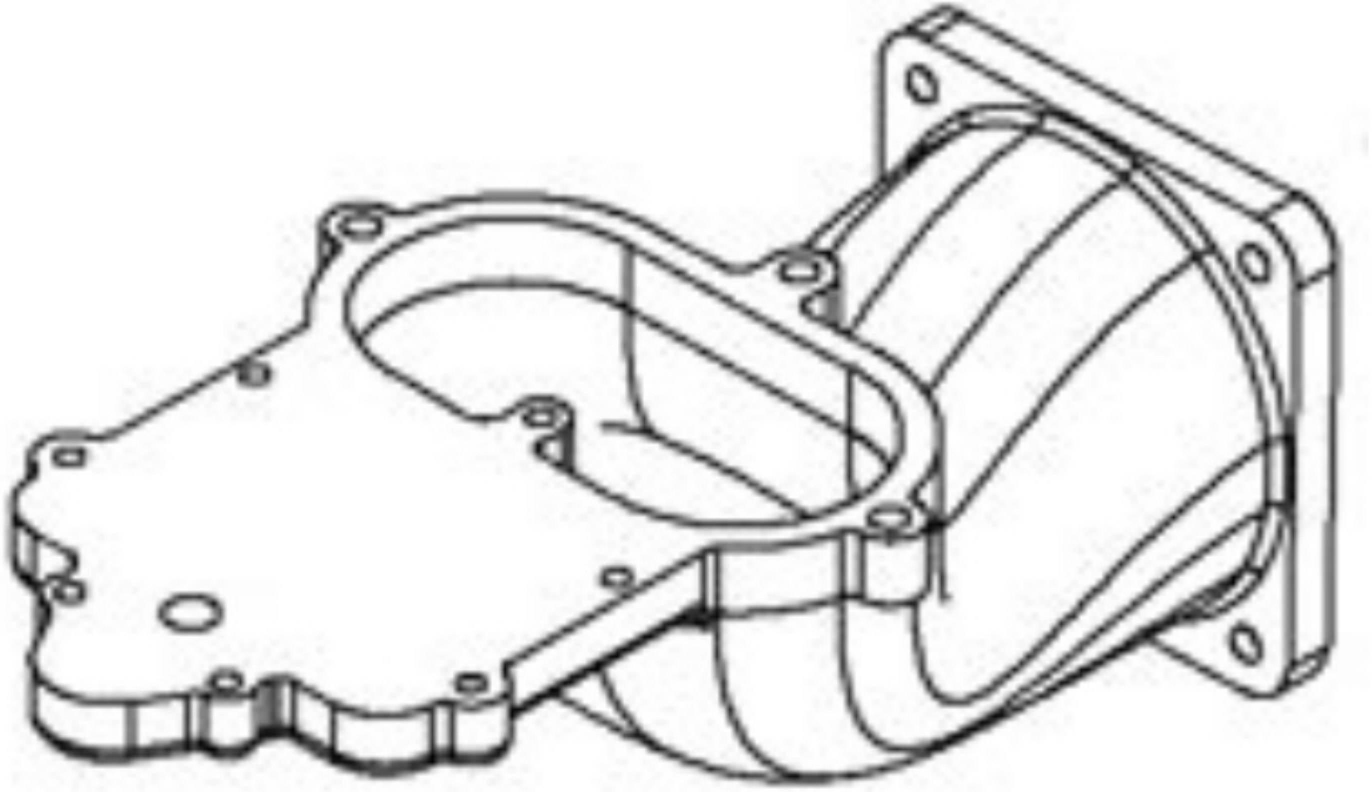


multiplication (i,j,k)
$c_{ij} = c_{ij} + a_{ik} b_{kj}$

Lemma (Loomis-Whitney Inequality)

Let $V \in Z[3]$ be a finite set, and let V_x, V_y, and V_z be orthogonal projections of V onto the coordinate planes. The cardinality of V, $|V|$, satisfies

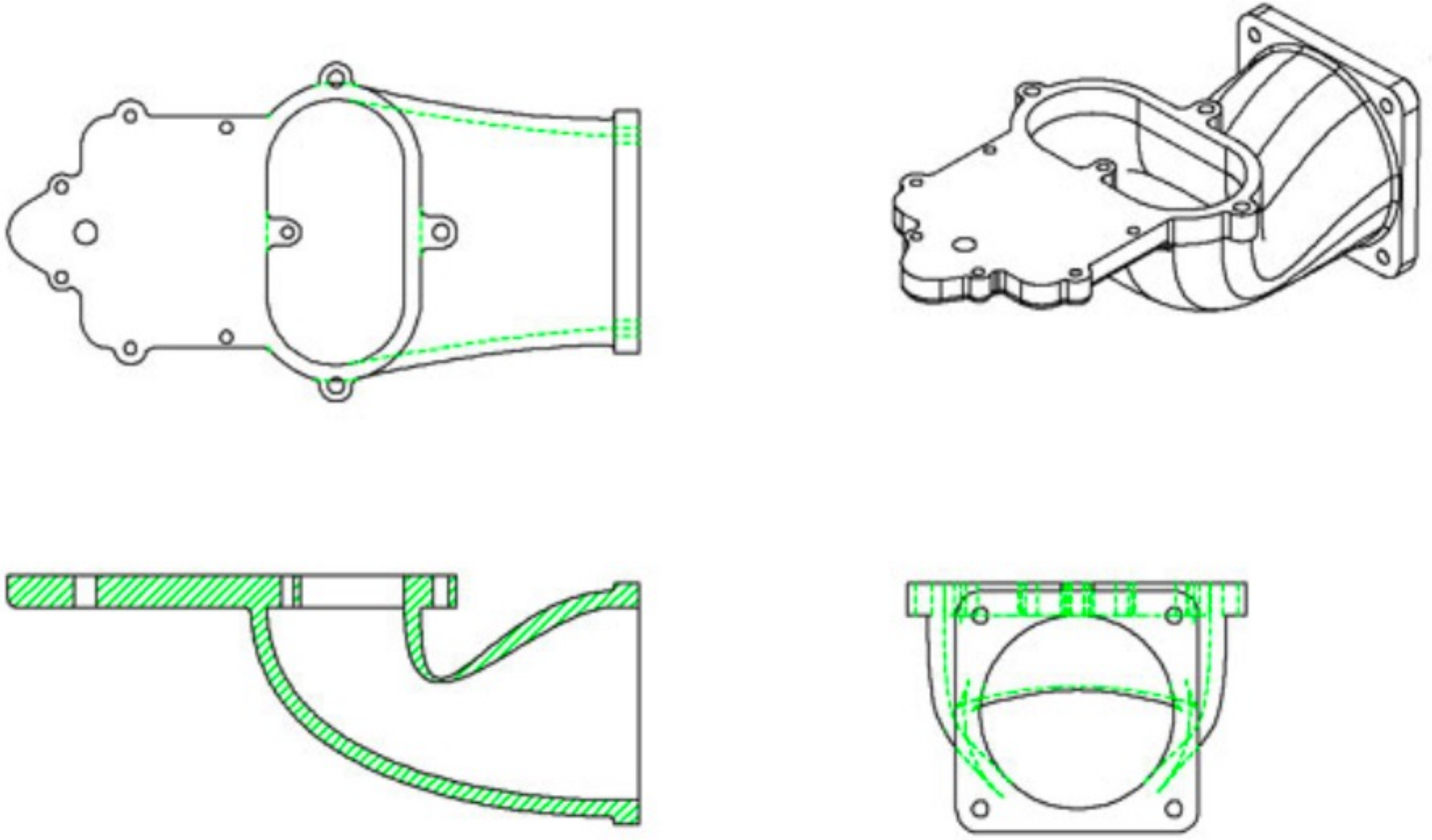
$$|V| \leq \sqrt{|V_x| \cdot |V_y| \cdot |V_z|}.$$



Lemma (Loomis-Whitney Inequality)

Let $V \in Z[3]$ be a finite set, and let V_x, V_y, and V_z be orthogonal projections of V onto the coordinate planes. The cardinality of V, $|V|$, satisfies

$$|V| \leq \sqrt{|V_x| \cdot |V_y| \cdot |V_z|}.$$

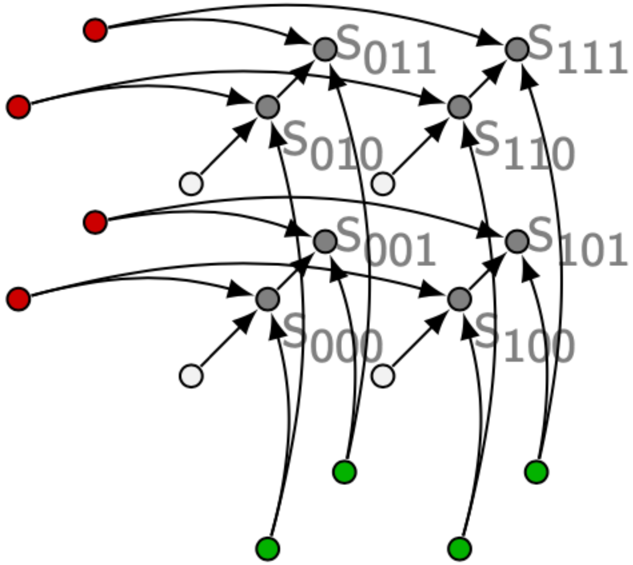


We consider a segmentation of the execution every T IOs. For each segment, we have at most K = S + T input data to perform the statements in the segment.

We will use a generalization: the Brascamp-Lieb inequality.

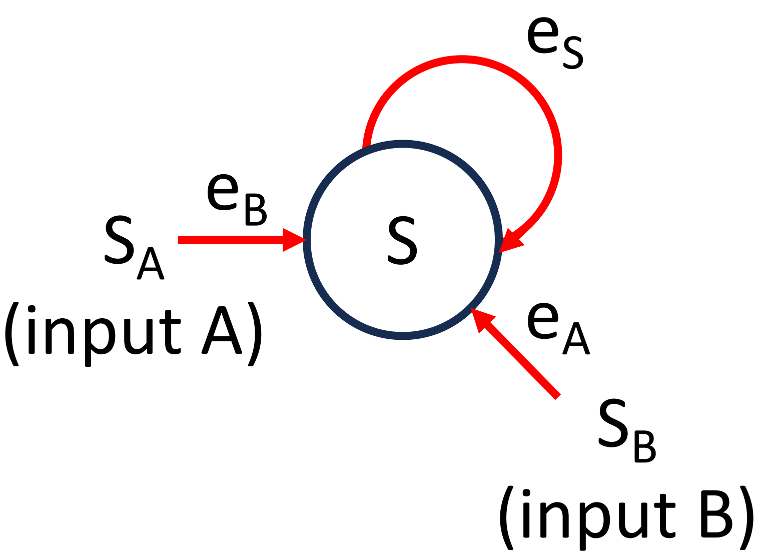
```
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < p; k++)
      C[i][j] += A[i][k] * B[k][j];
```

```
C[0][0] += A[0][0] * B[0][0];  // S000
C[0][0] += A[0][1] * B[1][0];  // S001
C[0][0] += A[0][2] * B[2][0];  // S002
C[0][0] += A[0][3] * B[3][0];  // S003
…
C[0][1] += A[0][0] * B[0][1];  // S010
C[0][1] += A[0][1] * B[1][1];  // S011
C[0][1] += A[0][2] * B[2][1];  // S012
C[0][1] += A[0][3] * B[3][1];  // S013
```

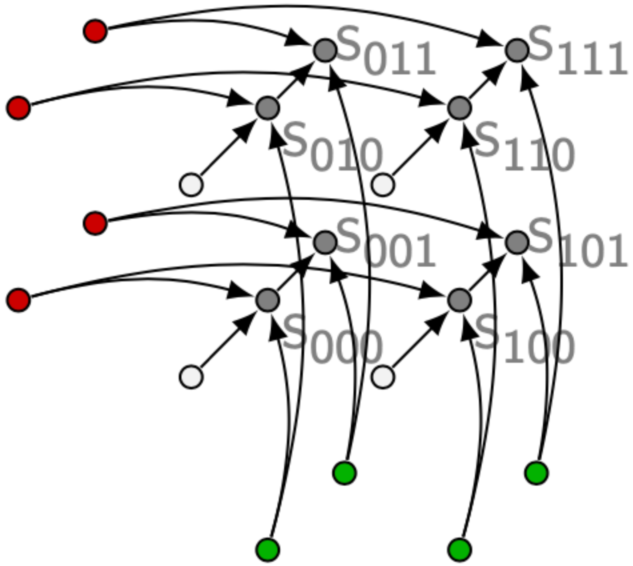


gemm

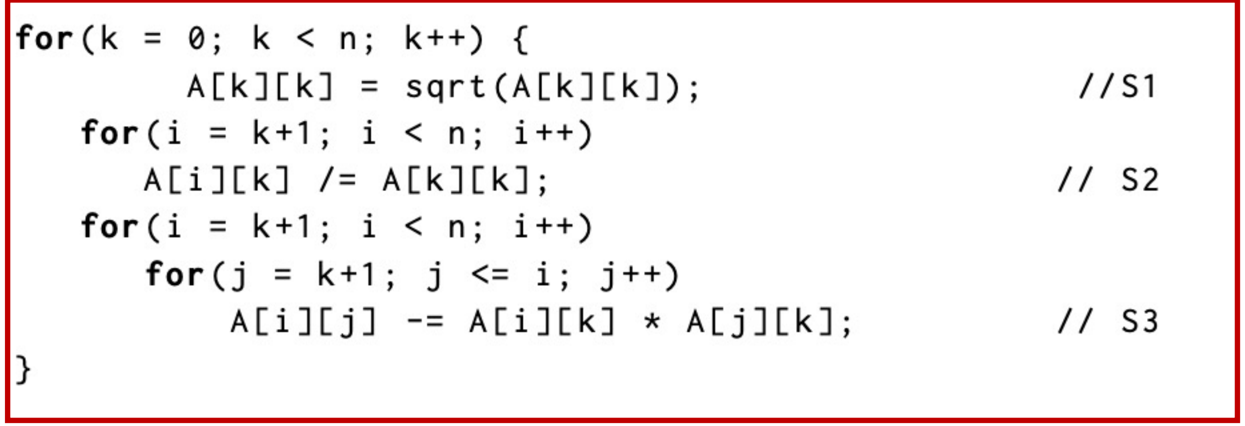
```
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < p; k++)
      C[i][j] += A[i][k] * B[k][j];
```



```
C[0][0] += A[0][0] * B[0][0];  // S000
C[0][0] += A[0][1] * B[1][0];  // S001
C[0][0] += A[0][2] * B[2][0];  // S002
C[0][0] += A[0][3] * B[3][0];  // S003
…
C[0][1] += A[0][0] * B[0][1];  // S010
C[0][1] += A[0][1] * B[1][1];  // S011
C[0][1] += A[0][2] * B[2][1];  // S012
C[0][1] += A[0][3] * B[3][1];  // S013
```



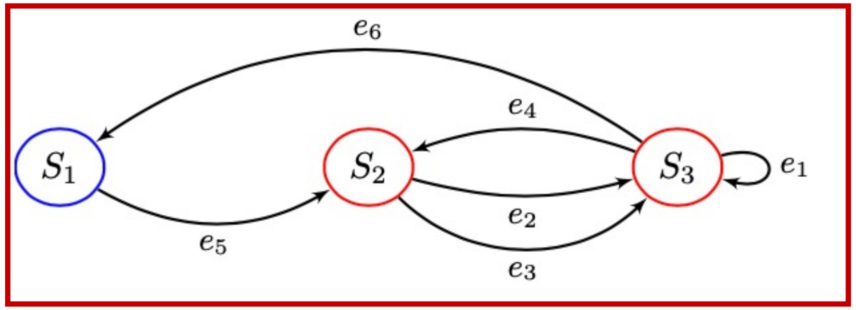
gemm



```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                  //S1
   for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                       // S2
   for(i = k+1; i < n; i++)
      for(j = k+1; j <= i; j++)
          A[i][j] -= A[i][k] * A[j][k];         // S3
}
```

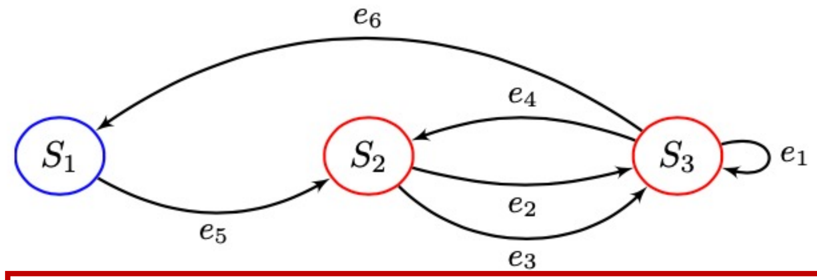
Cholesky

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                 //S1
   for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                       // S2
   for(i = k+1; i < n; i++)
      for(j = k+1; j <= i; j++)
          A[i][j] -= A[i][k] * A[j][k];         // S3
}
```



Cholesky

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                     //S1
   for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                           // S2
   for(i = k+1; i < n; i++)
      for(j = k+1; j <= i; j++)
          A[i][j] -= A[i][k] * A[j][k];             // S3
}
```



$$R_{e_1} = \{S_3[k-1,i,j] \longrightarrow S_3[k,i,j]:\ \ 1 \le k < N \ \wedge\ k+1 \le i < N \ \wedge\ k+1 \le j \le i\}$$
$$R_{e_2} = \{S_2[k,j] \longrightarrow S_3[k,i,j]:\ \ 0 \le k < N \ \wedge\ k+1 \le i < N \ \wedge\ k+1 \le j \le i\}$$
$$R_{e_3} = \{S_2[k,i] \longrightarrow S_3[k,i,j]:\ \ 0 \le k < N \ \wedge\ k+1 \le i < N \ \wedge\ k+1 \le j \le i\}$$
$$R_{e_4} = \{S_3[k-1,i,k] \longrightarrow S_2[k,i]:\ \ 1 \le k < N \ \wedge\ k+1 \le i < N\}$$
$$R_{e_5} = \{S_1[k] \longrightarrow S_2[k,i]:\ \ 0 \le k < N \ \wedge\ k+1 \le i < N\}$$
$$R_{e_6} = \{S_1[k-1,k,k] \longrightarrow S_1[k]:\ \ 1 \le k < N \ \wedge\ k+1 \le i < N\}$$

Cholesky

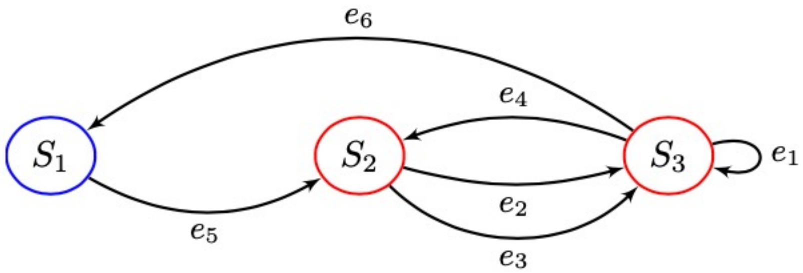
```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                    //S1
   for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                          // S2
   for(i = k+1; i < n; i++)
      for(j = k+1; j <= i; j++)
          A[i][j] -= A[i][k] * A[j][k];            // S3
}
```

$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j] : \; 1 \leq k < N \;\; \wedge \;\; k+1 \leq i < N \;\; \wedge \;\; k+1 \leq j \leq i\}$$

$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j] : \; 0 \leq k < N \;\; \wedge \;\; k+1 \leq i < N \;\; \wedge \;\; k+1 \leq j \leq i\}$$

$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j] : \; 0 \leq k < N \;\; \wedge \;\; k+1 \leq i < N \;\; \wedge \;\; k+1 \leq j \leq i\}$$

$$\begin{aligned} D_S &:= \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3) \\ &= \{S_3[k,i,j] : \; 1 \leq k < N \;\; \wedge \;\; k+1 \leq i < N \;\; \wedge \;\; k+1 \leq j \leq i\} \end{aligned}$$



$$R_{e_1} = \{S_3[k-1,i,j] \longrightarrow S_3[k,i,j] : \; 1 \leq k < N \;\; \wedge \;\; k+1 \leq i < N \;\; \wedge \;\; k+1 \leq j \leq i\}$$

$$R_{e_2} = \{S_2[k,j] \longrightarrow S_3[k,i,j] : \; 0 \leq k < N \;\; \wedge \;\; k+1 \leq i < N \;\; \wedge \;\; k+1 \leq j \leq i\}$$

$$R_{e_3} = \{S_2[k,i] \longrightarrow S_3[k,i,j] : \; 0 \leq k < N \;\; \wedge \;\; k+1 \leq i < N \;\; \wedge \;\; k+1 \leq j \leq i\}$$

$$R_{e_4} = \{S_3[k-1,i,k] \longrightarrow S_2[k,i] : \; 1 \leq k < N \;\; \wedge \;\; k+1 \leq i < N\}$$

$$R_{e_5} = \{S_1[k] \longrightarrow S_2[k,i] : \; 0 \leq k < N \;\; \wedge \;\; k+1 \leq i < N\}$$

$$R_{e_6} = \{S_1[k-1,k,k] \longrightarrow S_1[k] : \; 1 \leq k < N \;\; \wedge \;\; k+1 \leq i < N\}$$

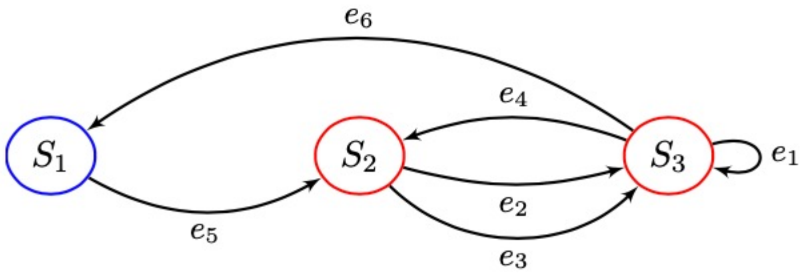
$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

Cholesky

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                    //S1
   for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                          // S2
   for(i = k+1; i < n; i++)
      for(j = k+1; j <= i; j++)
          A[i][j] -= A[i][k] * A[j][k];            // S3
}
```



$$R_{e_1} = \{S_3[k-1,i,j] \rightarrow S_3[k,i,j] : \; 1 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$R_{e_2} = \{S_2[k,j] \rightarrow S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$R_{e_3} = \{S_2[k,i] \rightarrow S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$R_{e_4} = \{S_3[k-1,i,k] \rightarrow S_2[k,i] : \; 1 \le k < N \;\wedge\; k+1 \le i < N\}$$
$$R_{e_5} = \{S_1[k] \rightarrow S_2[k,i] : \; 0 \le k < N \;\wedge\; k+1 \le i < N\}$$
$$R_{e_6} = \{S_1[k-1,k,k] \rightarrow S_1[k] : \; 1 \le k < N \;\wedge\; k+1 \le i < N\}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

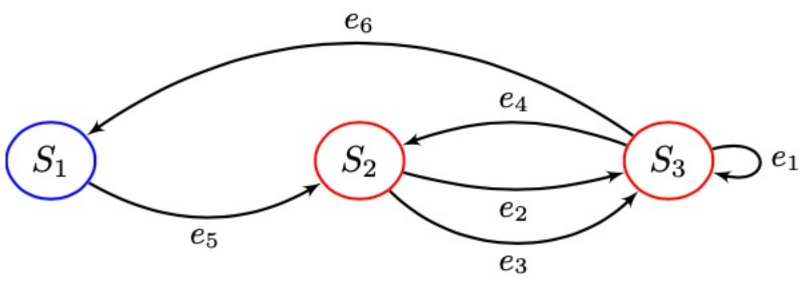
$P_3 = (e_3)$ is a broadcast path

Cholesky

$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j] : \; 1 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$\begin{aligned} D_S &:= \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3) \\ &= \{S_3[k,i,j] : \; 1 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\} \end{aligned}$$

$$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}(k,i,j) = (0,i,j) \qquad \text{kernel } k_1 = \mathrm{Ker}(\phi_1) = \langle(1,0,0)\rangle$$
$$\phi_2(k,i,j) = (k,j) \qquad \text{kernel } k_2 = \mathrm{Ker}(\phi_2) = \langle(0,1,0)\rangle$$
$$\phi_3(k,i,j) = (k,i) \qquad \text{kernel } k_3 = \mathrm{Ker}(\phi_3) = \langle(0,0,1)\rangle$$

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                    //S1
    for(i = k+1; i < n; i++)
       A[i][k] /= A[k][k];                         // S2
    for(i = k+1; i < n; i++)
       for(j = k+1; j <= i; j++)
           A[i][j] -= A[i][k] * A[j][k];           // S3
}
```



$$R_{e_1} = \{S_3[k-1,i,j] \rightarrow S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$R_{e_2} = \{S_2[k,j] \rightarrow S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$R_{e_3} = \{S_2[k,i] \rightarrow S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$R_{e_4} = \{S_3[k-1,i,k] \rightarrow S_2[k,i] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$

$$R_{e_5} = \{S_1[k] \rightarrow S_2[k,i] : \ 0 \le k < N \ \wedge \ k+1 \le i < N\}$$

$$R_{e_6} = \{S_1[k-1,k,k] \rightarrow S_1[k] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

$$\text{Dom}\,(P_1) = \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$\text{Dom}\,(P_2) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

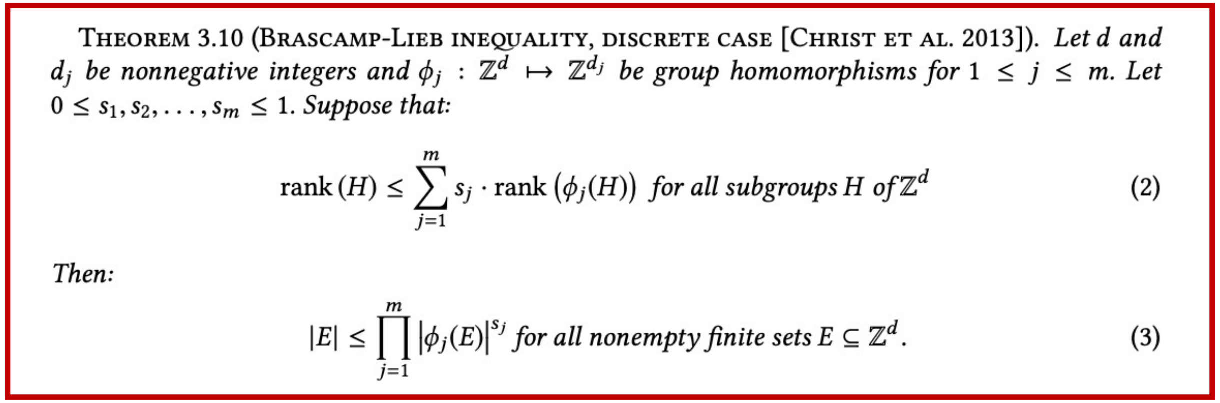
$$\text{Dom}\,(P_3) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$D_S := \text{Dom}\,(P_1) \cap \text{Dom}\,(P_2) \cap \text{Dom}\,(P_3)$$
$$= \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$\phi_1(k,i,j) = \text{proj}_{(1,0,0)}\,(k,i,j) = (0,i,j)$ kernel $k_1 = \text{Ker}(\phi_1) = \langle(1,0,0)\rangle$

$\phi_2(k,i,j) = (k,j)$ kernel $k_2 = \text{Ker}(\phi_2) = \langle(0,1,0)\rangle$

$\phi_3(k,i,j) = (k,i)$ kernel $k_3 = \text{Ker}(\phi_3) = \langle(0,0,1)\rangle$



Theorem 3.10 (Brascamp-Lieb inequality, discrete case [Christ et al. 2013]). *Let d and d_j be nonnegative integers and $\phi_j : \mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ be group homomorphisms for $1 \le j \le m$. Let $0 \le s_1, s_2, \ldots, s_m \le 1$. Suppose that:*

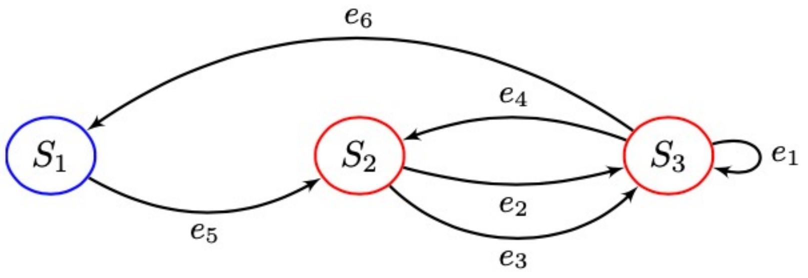
$$\text{rank}\,(H) \le \sum_{j=1}^{m} s_j \cdot \text{rank}\,(\phi_j(H)) \ \textit{for all subgroups } H \textit{ of } \mathbb{Z}^d \tag{2}$$

Then:

$$|E| \le \prod_{j=1}^{m} |\phi_j(E)|^{s_j} \ \textit{for all nonempty finite sets } E \subseteq \mathbb{Z}^d. \tag{3}$$

Cholesky

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                    //S1
    for(i = k+1; i < n; i++)
       A[i][k] /= A[k][k];                          // S2
    for(i = k+1; i < n; i++)
       for(j = k+1; j <= i; j++)
           A[i][j] -= A[i][k] * A[j][k];            // S3
}
```



$$R_{e_1} = \{S_3[k-1,i,j] \rightarrow S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_2} = \{S_2[k,j] \rightarrow S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_3} = \{S_2[k,i] \rightarrow S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_4} = \{S_3[k-1,i,k] \rightarrow S_2[k,i] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$
$$R_{e_5} = \{S_1[k] \rightarrow S_2[k,i] : \ 0 \le k < N \ \wedge \ k+1 \le i < N\}$$
$$R_{e_6} = \{S_1[k-1,k,k] \rightarrow S_1[k] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

Cholesky

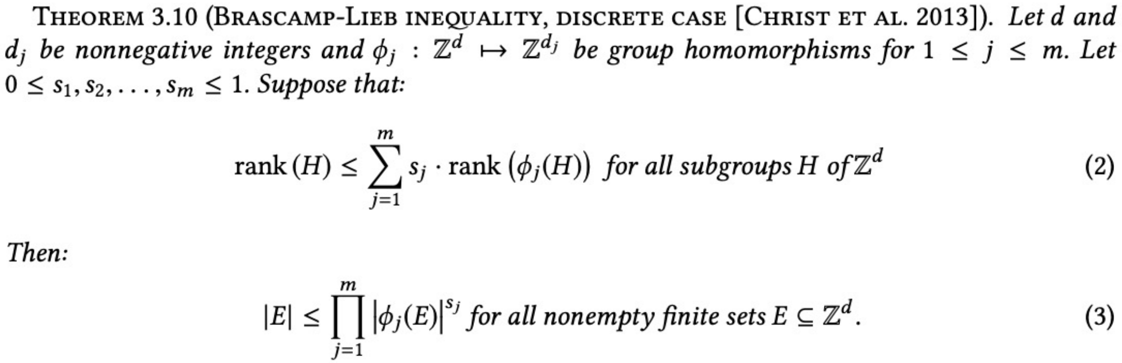
$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$D_S := \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3)$$
$$= \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}(k,i,j) = (0,i,j)$ kernel $k_1 = \mathrm{Ker}(\phi_1) = \langle(1,0,0)\rangle$

$\phi_2(k,i,j) = (k,j)$ kernel $k_2 = \mathrm{Ker}(\phi_2) = \langle(0,1,0)\rangle$

$\phi_3(k,i,j) = (k,i)$ kernel $k_3 = \mathrm{Ker}(\phi_3) = \langle(0,0,1)\rangle$



Theorem 3.10 (Brascamp-Lieb inequality, discrete case [Christ et al. 2013]). *Let d and d_j be nonnegative integers and $\phi_j : \mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ be group homomorphisms for $1 \le j \le m$. Let $0 \le s_1, s_2, \ldots, s_m \le 1$. Suppose that:*

$$\mathrm{rank}\,(H) \le \sum_{j=1}^{m} s_j \cdot \mathrm{rank}\,(\phi_j(H)) \ \textit{for all subgroups } H \textit{ of } \mathbb{Z}^d \qquad (2)$$

Then:

$$|E| \le \prod_{j=1}^{m} |\phi_j(E)|^{s_j} \ \textit{for all nonempty finite sets } E \subseteq \mathbb{Z}^d. \qquad (3)$$

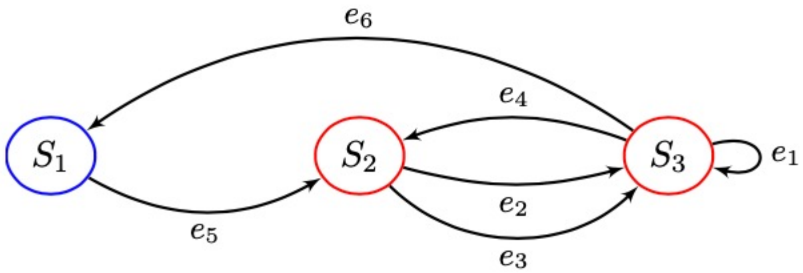
$0 \le s_1, s_2, s_3 \le 1$

$1 \le s_2 + s_3$

$1 \le s_1 + s_3$

$1 \le s_1 + s_2$

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                    //S1
   for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                          // S2
   for(i = k+1; i < n; i++)
      for(j = k+1; j <= i; j++)
          A[i][j] -= A[i][k] * A[j][k];            // S3
}
```



$$R_{e_1} = \{S_3[k-1,i,j] \longrightarrow S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_2} = \{S_2[k,j] \longrightarrow S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_3} = \{S_2[k,i] \longrightarrow S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_4} = \{S_3[k-1,i,k] \longrightarrow S_2[k,i] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$
$$R_{e_5} = \{S_1[k] \longrightarrow S_2[k,i] : \ 0 \le k < N \ \wedge \ k+1 \le i < N\}$$
$$R_{e_6} = \{S_1[k-1,k,k] \longrightarrow S_1[k] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$D_S := \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3)$$
$$= \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}\,(k,i,j) = (0,i,j)$ kernel $k_1 = \mathrm{Ker}(\phi_1) = \langle(1,0,0)\rangle$

$\phi_2(k,i,j) = (k,j)$ kernel $k_2 = \mathrm{Ker}(\phi_2) = \langle(0,1,0)\rangle$

$\phi_3(k,i,j) = (k,i)$ kernel $k_3 = \mathrm{Ker}(\phi_3) = \langle(0,0,1)\rangle$

Theorem 3.10 (Brascamp-Lieb inequality, discrete case [Christ et al. 2013]). *Let d and d_j be nonnegative integers and $\phi_j : \mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ be group homomorphisms for $1 \le j \le m$. Let $0 \le s_1, s_2, \ldots, s_m \le 1$. Suppose that:*

$$\mathrm{rank}\,(H) \le \sum_{j=1}^{m} s_j \cdot \mathrm{rank}\,(\phi_j(H)) \ \textit{for all subgroups } H \textit{ of } \mathbb{Z}^d \qquad (2)$$

Then:

$$|E| \le \prod_{j=1}^{m} |\phi_j(E)|^{s_j} \ \textit{for all nonempty finite sets } E \subseteq \mathbb{Z}^d. \qquad (3)$$

$0 \le s_1, s_2, s_3 \le 1$

$1 \le s_2 + s_3$

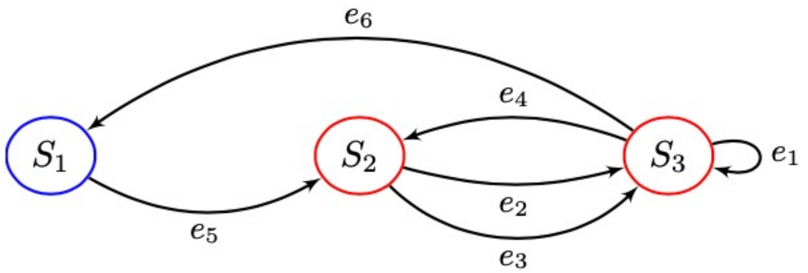
$1 \le s_1 + s_3$

$1 \le s_1 + s_2$

$$\boxed{s_1 = s_2 = s_3 = \tfrac{1}{2}}$$

Cholesky

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                    //S1
    for(i = k+1; i < n; i++)
       A[i][k] /= A[k][k];                         // S2
    for(i = k+1; i < n; i++)
       for(j = k+1; j <= i; j++)
           A[i][j] -= A[i][k] * A[j][k];           // S3
}
```



$$R_{e_1} = \{S_3[k-1,i,j] \rightarrow S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_2} = \{S_2[k,j] \rightarrow S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_3} = \{S_2[k,i] \rightarrow S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_4} = \{S_3[k-1,i,k] \rightarrow S_2[k,i] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$
$$R_{e_5} = \{S_1[k] \rightarrow S_2[k,i] : \ 0 \le k < N \ \wedge \ k+1 \le i < N\}$$
$$R_{e_6} = \{S_1[k-1,k,k] \rightarrow S_1[k] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

Cholesky

$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$\begin{aligned} D_S &:= \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3) \\ &= \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\} \end{aligned}$$

$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}\,(k,i,j) = (0,i,j)$ kernel $k_1 = \mathrm{Ker}(\phi_1) = \langle(1,0,0)\rangle$

$\phi_2(k,i,j) = (k,j)$ kernel $k_2 = \mathrm{Ker}(\phi_2) = \langle(0,1,0)\rangle$

$\phi_3(k,i,j) = (k,i)$ kernel $k_3 = \mathrm{Ker}(\phi_3) = \langle(0,0,1)\rangle$

THEOREM 3.10 (BRASCAMP-LIEB INEQUALITY, DISCRETE CASE [CHRIST ET AL. 2013]). *Let d and d_j be nonnegative integers and $\phi_j : \mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ be group homomorphisms for $1 \le j \le m$. Let $0 \le s_1, s_2, \ldots, s_m \le 1$. Suppose that:*

$$\mathrm{rank}\,(H) \le \sum_{j=1}^{m} s_j \cdot \mathrm{rank}\,(\phi_j(H)) \ \ \textit{for all subgroups } H \textit{ of } \mathbb{Z}^d \qquad (2)$$

Then:

$$|E| \le \prod_{j=1}^{m} |\phi_j(E)|^{s_j} \ \textit{for all nonempty finite sets } E \subseteq \mathbb{Z}^d. \qquad (3)$$

$0 \le s_1, s_2, s_3 \le 1$

$1 \le s_2 + s_3$

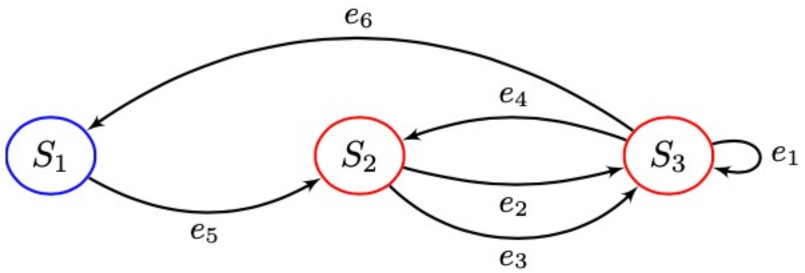
$1 \le s_1 + s_3$

$1 \le s_1 + s_2$

$s_1 = s_2 = s_3 = \frac{1}{2}$

$$\boxed{|E| \le |\phi_1(E)|^{\frac{1}{2}} \cdot |\phi_2(E)|^{\frac{1}{2}} \cdot |\phi_3(E)|^{\frac{1}{2}}}$$

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                        //S1
    for(i = k+1; i < n; i++)
       A[i][k] /= A[k][k];                              // S2
    for(i = k+1; i < n; i++)
       for(j = k+1; j <= i; j++)
          A[i][j] -= A[i][k] * A[j][k];                 // S3
}
```



$$R_{e_1} = \{S_3[k-1,i,j] \rightarrow S_3[k,i,j] : \; 1 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$R_{e_2} = \{S_2[k,j] \rightarrow S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$R_{e_3} = \{S_2[k,i] \rightarrow S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$R_{e_4} = \{S_3[k-1,i,k] \rightarrow S_2[k,i] : \; 1 \le k < N \;\wedge\; k+1 \le i < N\}$$
$$R_{e_5} = \{S_1[k] \rightarrow S_2[k,i] : \; 0 \le k < N \;\wedge\; k+1 \le i < N\}$$
$$R_{e_6} = \{S_1[k-1,k,k] \rightarrow S_1[k] : \; 1 \le k < N \;\wedge\; k+1 \le i < N\}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j] : \; 1 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$\begin{aligned} D_S &:= \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3) \\ &= \{S_3[k,i,j] : \; 1 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\} \end{aligned}$$

$$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}(k,i,j) = (0,i,j) \qquad \text{kernel } k_1 = \mathrm{Ker}(\phi_1) = \langle(1,0,0)\rangle$$
$$\phi_2(k,i,j) = (k,j) \qquad \text{kernel } k_2 = \mathrm{Ker}(\phi_2) = \langle(0,1,0)\rangle$$
$$\phi_3(k,i,j) = (k,i) \qquad \text{kernel } k_3 = \mathrm{Ker}(\phi_3) = \langle(0,0,1)\rangle$$

Theorem 3.10 (Brascamp-Lieb inequality, discrete case [Christ et al. 2013]). *Let d and d_j be nonnegative integers and $\phi_j : \mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ be group homomorphisms for $1 \le j \le m$. Let $0 \le s_1, s_2, \ldots, s_m \le 1$. Suppose that:*

$$\mathrm{rank}\,(H) \le \sum_{j=1}^{m} s_j \cdot \mathrm{rank}\,(\phi_j(H)) \;\; \textit{for all subgroups } H \textit{ of } \mathbb{Z}^d \tag{2}$$

Then:

$$|E| \le \prod_{j=1}^{m} |\phi_j(E)|^{s_j} \; \textit{for all nonempty finite sets } E \subseteq \mathbb{Z}^d. \tag{3}$$

$0 \le s_1, s_2, s_3 \le 1$

$1 \le s_2 + s_3$

$1 \le s_1 + s_3$

$1 \le s_1 + s_2$

$$s_1 = s_2 = s_3 = \frac{1}{2}$$

$$\boxed{\begin{array}{c} |E| \le |\phi_1(E)|^{\frac{1}{2}} \cdot |\phi_2(E)|^{\frac{1}{2}} \cdot |\phi_3(E)|^{\frac{1}{2}} \\ |\phi_i(E)| \le K, \; i = 1,2,3 \\ |E| \le K^{3/2} \end{array}}$$

Cholesky

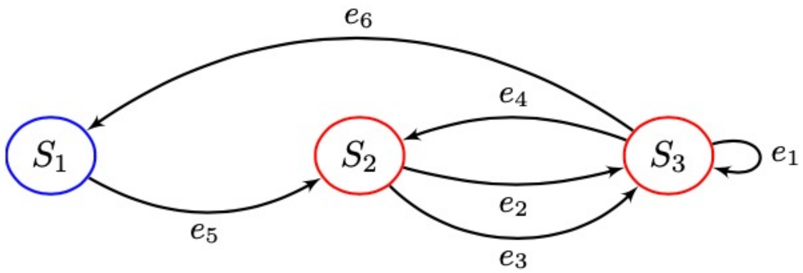
```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                    //S1
    for(i = k+1; i < n; i++)
       A[i][k] /= A[k][k];                         // S2
    for(i = k+1; i < n; i++)
       for(j = k+1; j <= i; j++)
           A[i][j] -= A[i][k] * A[j][k];           // S3
}
```

$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j]: \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j]: \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j]: \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$\begin{aligned} D_S &:= \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3) \\ &= \{S_3[k,i,j]: \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\} \end{aligned}$$



$$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}\,(k,i,j) = (0,i,j) \qquad \text{kernel } k_1 = \mathrm{Ker}(\phi_1) = \langle(1,0,0)\rangle$$

$$\phi_2(k,i,j) = (k,j) \qquad \text{kernel } k_2 = \mathrm{Ker}(\phi_2) = \langle(0,1,0)\rangle$$

$$\phi_3(k,i,j) = (k,i) \qquad \text{kernel } k_3 = \mathrm{Ker}(\phi_3) = \langle(0,0,1)\rangle$$

Theorem 3.10 (Brascamp-Lieb inequality, discrete case [Christ et al. 2013]). *Let d and d_j be nonnegative integers and $\phi_j : \mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ be group homomorphisms for $1 \le j \le m$. Let $0 \le s_1, s_2, \ldots, s_m \le 1$. Suppose that:*

$$\mathrm{rank}\,(H) \le \sum_{j=1}^{m} s_j \cdot \mathrm{rank}\,(\phi_j(H)) \ \textit{for all subgroups } H \textit{ of } \mathbb{Z}^d \qquad (2)$$

Then:

$$|E| \le \prod_{j=1}^{m} |\phi_j(E)|^{s_j} \textit{ for all nonempty finite sets } E \subseteq \mathbb{Z}^d. \qquad (3)$$

$$R_{e_1} = \{S_3[k-1,i,j] \to S_3[k,i,j]: \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$R_{e_2} = \{S_2[k,j] \to S_3[k,i,j]: \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$R_{e_3} = \{S_2[k,i] \to S_3[k,i,j]: \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$

$$R_{e_4} = \{S_3[k-1,i,k] \to S_2[k,i]: \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$

$$R_{e_5} = \{S_1[k] \to S_2[k,i]: \ 0 \le k < N \ \wedge \ k+1 \le i < N\}$$

$$R_{e_6} = \{S_1[k-1,k,k] \to S_1[k]: \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$

$$0 \le s_1, s_2, s_3 \le 1 \qquad s_1 = s_2 = s_3 = \frac{1}{2}$$

$$1 \le s_2 + s_3$$

$$1 \le s_1 + s_3 \qquad |E| \le |\phi_1(E)|^{\frac{1}{2}} \cdot |\phi_2(E)|^{\frac{1}{2}} \cdot |\phi_3(E)|^{\frac{1}{2}}$$

$$1 \le s_1 + s_2 \qquad \boxed{|\phi_i(E)| \le K, \ i = 1,2,3}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

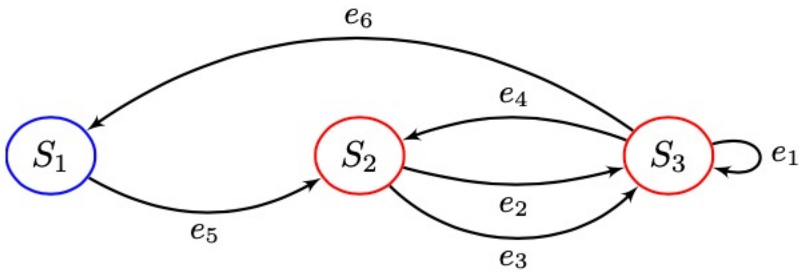
$$|E| \le K^{3/2}$$

$$\boxed{\begin{aligned} &|\phi_1(E)| + |\phi_2(E)| \le K \\ &|\phi_1(E)| + |\phi_3(E)| \le K \qquad |\phi_1(E)| + \frac{1}{2}\,|\phi_2(E)| + \frac{1}{2}\,|\phi_3(E)| \le K \end{aligned}}$$

Cholesky

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                    //S1
    for(i = k+1; i < n; i++)
       A[i][k] /= A[k][k];                          // S2
    for(i = k+1; i < n; i++)
       for(j = k+1; j <= i; j++)
           A[i][j] -= A[i][k] * A[j][k];            // S3
}
```

$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j]:\ 1 \le k < N\ \wedge\ k+1 \le i < N\ \wedge\ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j]:\ 0 \le k < N\ \wedge\ k+1 \le i < N\ \wedge\ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j]:\ 0 \le k < N\ \wedge\ k+1 \le i < N\ \wedge\ k+1 \le j \le i\}$$
$$D_S := \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3)$$
$$= \{S_3[k,i,j]:\ 1 \le k < N\ \wedge\ k+1 \le i < N\ \wedge\ k+1 \le j \le i\}$$



$$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}\,(k,i,j) = (0,i,j) \qquad \text{kernel } k_1 = \mathrm{Ker}(\phi_1) = \langle(1,0,0)\rangle$$
$$\phi_2(k,i,j) = (k,j) \qquad \text{kernel } k_2 = \mathrm{Ker}(\phi_2) = \langle(0,1,0)\rangle$$
$$\phi_3(k,i,j) = (k,i) \qquad \text{kernel } k_3 = \mathrm{Ker}(\phi_3) = \langle(0,0,1)\rangle$$

Theorem 3.10 (Brascamp-Lieb inequality, discrete case [Christ et al. 2013]). *Let d and d_j be nonnegative integers and $\phi_j : \mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ be group homomorphisms for $1 \le j \le m$. Let $0 \le s_1, s_2, \ldots, s_m \le 1$. Suppose that:*

$$\mathrm{rank}\,(H) \le \sum_{j=1}^{m} s_j \cdot \mathrm{rank}\,(\phi_j(H))\ \ \textit{for all subgroups } H \textit{ of } \mathbb{Z}^d \tag{2}$$

Then:

$$|E| \le \prod_{j=1}^{m} |\phi_j(E)|^{s_j} \textit{ for all nonempty finite sets } E \subseteq \mathbb{Z}^d. \tag{3}$$

$$R_{e_1} = \{S_3[k-1,i,j] \to S_3[k,i,j]:\ 1 \le k < N\ \wedge\ k+1 \le i < N\ \wedge\ k+1 \le j \le i\}$$
$$R_{e_2} = \{S_2[k,j] \to S_3[k,i,j]:\ 0 \le k < N\ \wedge\ k+1 \le i < N\ \wedge\ k+1 \le j \le i\}$$
$$R_{e_3} = \{S_2[k,i] \to S_3[k,i,j]:\ 0 \le k < N\ \wedge\ k+1 \le i < N\ \wedge\ k+1 \le j \le i\}$$
$$R_{e_4} = \{S_3[k-1,i,k] \to S_2[k,i]:\ 1 \le k < N\ \wedge\ k+1 \le i < N\}$$
$$R_{e_5} = \{S_1[k] \to S_2[k,i]:\ 0 \le k < N\ \wedge\ k+1 \le i < N\}$$
$$R_{e_6} = \{S_1[k-1,k,k] \to S_1[k]:\ 1 \le k < N\ \wedge\ k+1 \le i < N\}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

$$0 \le s_1, s_2, s_3 \le 1$$
$$1 \le s_2 + s_3$$
$$1 \le s_1 + s_3$$
$$1 \le s_1 + s_2$$

$$s_1 = s_2 = s_3 = \frac{1}{2}$$

$$|E| \le |\phi_1(E)|^{\frac{1}{2}} \cdot |\phi_2(E)|^{\frac{1}{2}} \cdot |\phi_3(E)|^{\frac{1}{2}}$$

$$|\phi_i(E)| \le K,\ i = 1,2,3$$

$$|E| \le K^{3/2}$$

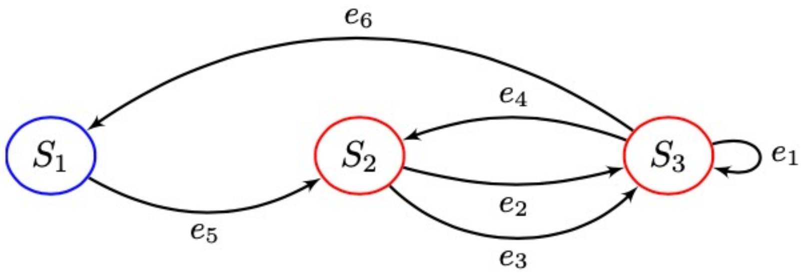
$$|\phi_1(E)| + |\phi_2(E)| \le K$$
$$|\phi_1(E)| + |\phi_3(E)| \le K$$

$$|\phi_1(E)| + \frac{1}{2}\,|\phi_2(E)| + \frac{1}{2}\,|\phi_3(E)| \le K$$

$$\boxed{|E| \le 2 \cdot (K/3)^{3/2}}$$

Cholesky

```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                     //S1
   for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                           // S2
   for(i = k+1; i < n; i++)
      for(j = k+1; j <= i; j++)
          A[i][j] -= A[i][k] * A[j][k];             // S3
}
```



$$R_{e_1} = \{S_3[k-1,i,j] \to S_3[k,i,j] : \; 1 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$R_{e_2} = \{S_2[k,j] \to S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$R_{e_3} = \{S_2[k,i] \to S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$R_{e_4} = \{S_3[k-1,i,k] \to S_2[k,i] : \; 1 \le k < N \;\wedge\; k+1 \le i < N\}$$
$$R_{e_5} = \{S_1[k] \to S_2[k,i] : \; 0 \le k < N \;\wedge\; k+1 \le i < N\}$$
$$R_{e_6} = \{S_1[k-1,k,k] \to S_1[k] : \; 1 \le k < N \;\wedge\; k+1 \le i < N\}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

Cholesky

$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j] : \; 1 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j] : \; 0 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\}$$

$$\begin{aligned} D_S &:= \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3) \\ &= \{S_3[k,i,j] : \; 1 \le k < N \;\wedge\; k+1 \le i < N \;\wedge\; k+1 \le j \le i\} \end{aligned}$$

$$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}\,(k,i,j) = (0,i,j) \qquad \text{kernel } k_1 = \mathrm{Ker}(\phi_1) = \langle(1,0,0)\rangle$$
$$\phi_2(k,i,j) = (k,j) \qquad \text{kernel } k_2 = \mathrm{Ker}(\phi_2) = \langle(0,1,0)\rangle$$
$$\phi_3(k,i,j) = (k,i) \qquad \text{kernel } k_3 = \mathrm{Ker}(\phi_3) = \langle(0,0,1)\rangle$$

THEOREM 3.10 (BRASCAMP-LIEB INEQUALITY, DISCRETE CASE [CHRIST ET AL. 2013]). *Let d and d_j be nonnegative integers and $\phi_j : \mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ be group homomorphisms for $1 \le j \le m$. Let $0 \le s_1, s_2, \ldots, s_m \le 1$. Suppose that:*

$$\mathrm{rank}\,(H) \le \sum_{j=1}^{m} s_j \cdot \mathrm{rank}\,(\phi_j(H)) \;\; \textit{for all subgroups } H \textit{ of } \mathbb{Z}^d \tag{2}$$

Then:

$$|E| \le \prod_{j=1}^{m} |\phi_j(E)|^{s_j} \; \textit{for all nonempty finite sets } E \subseteq \mathbb{Z}^d. \tag{3}$$

$$0 \le s_1, s_2, s_3 \le 1$$
$$1 \le s_2 + s_3$$
$$1 \le s_1 + s_3$$
$$1 \le s_1 + s_2$$

$$s_1 = s_2 = s_3 = \frac{1}{2}$$

$$|E| \le |\phi_1(E)|^{\frac{1}{2}} \cdot |\phi_2(E)|^{\frac{1}{2}} \cdot |\phi_3(E)|^{\frac{1}{2}}$$

$$|\phi_i(E)| \le K, \; i = 1, 2, 3$$

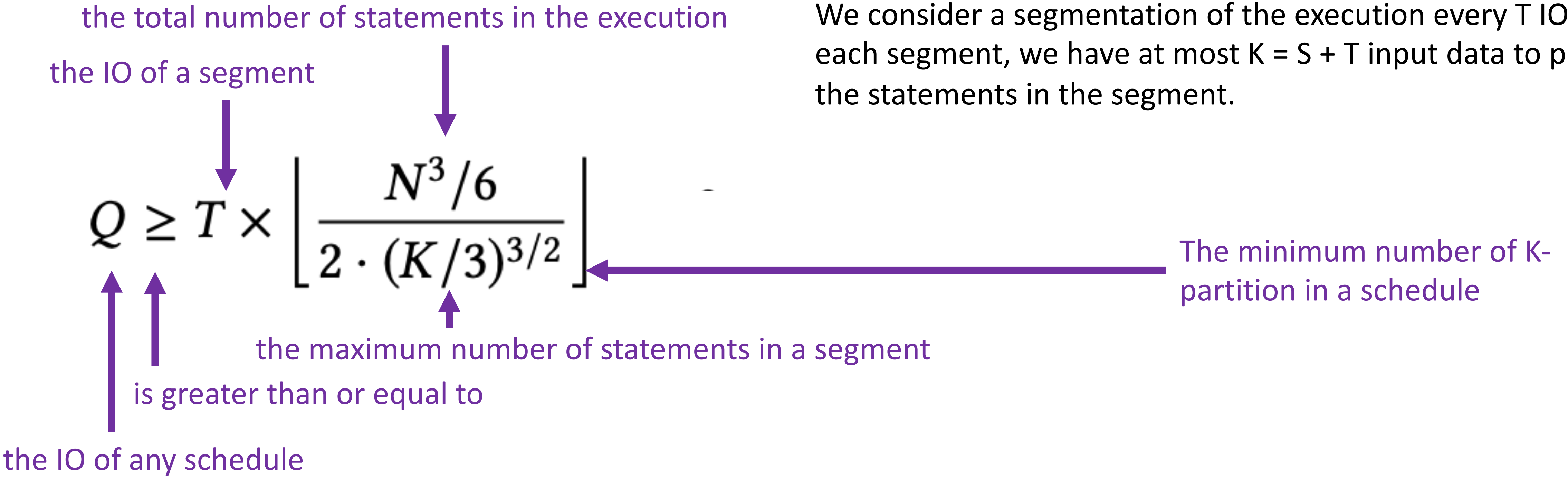
$$|E| \le K^{3/2}$$

$$|\phi_1(E)| + |\phi_2(E)| \le K$$
$$|\phi_1(E)| + |\phi_3(E)| \le K$$

$$|\phi_1(E)| + \frac{1}{2}\,|\phi_2(E)| + \frac{1}{2}\,|\phi_3(E)| \le K$$

$$\boxed{|E| \le 2 \cdot (K/3)^{3/2}}$$

We consider a segmentation of the execution every T IOs. For each segment, we have at most K = S + T input data to perform the statements in the segment.



the total number of statements in the execution

the IO of a segment

$$Q \geq T \times \left\lfloor \frac{N^3/6}{2 \cdot (K/3)^{3/2}} \right\rfloor$$

The minimum number of K-partition in a schedule

the maximum number of statements in a segment

is greater than or equal to

the IO of any schedule

Cholesky

the total number of statements in the execution

the IO of a segment

We consider a segmentation of the execution every T IOs. For each segment, we have at most K = S + T input data to perform the statements in the segment.

$$Q \geq T \times \left\lfloor \frac{N^3/6}{2 \cdot (K/3)^{3/2}} \right\rfloor$$

The minimum number of K-partition in a schedule

the maximum number of statements in a segment

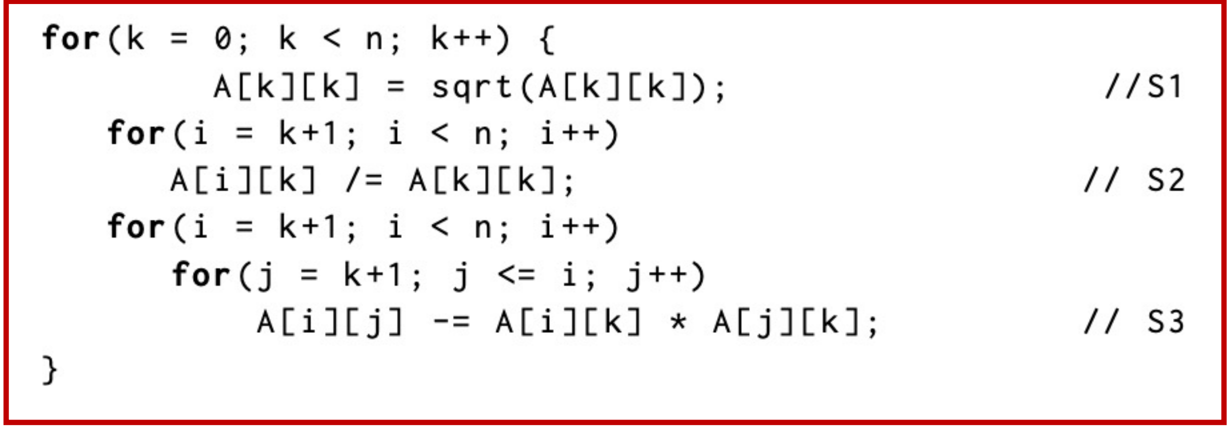
is greater than or equal to

the IO of any schedule

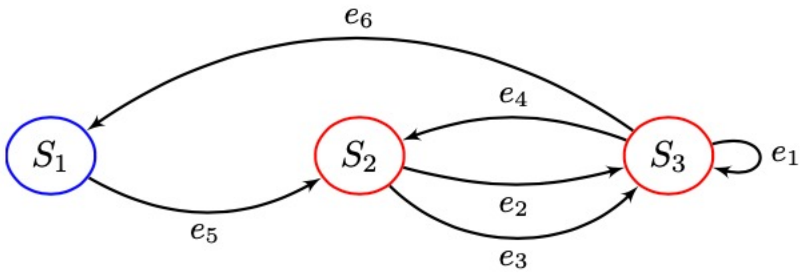
Setting $T = 2S$ (so $K = S + T = 3S$)

$$Q \geq (2S) \times \frac{N^3/6}{2S^{3/2}} = \boxed{\frac{N^3}{6\sqrt{S}}}$$

Cholesky



```
for(k = 0; k < n; k++) {
        A[k][k] = sqrt(A[k][k]);                        //S1
    for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                               // S2
    for(i = k+1; i < n; i++)
      for(j = k+1; j <= i; j++)
          A[i][j] -= A[i][k] * A[j][k];                 // S3
}
```



$$R_{e_1} = \{S_3[k-1,i,j] \to S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_2} = \{S_2[k,j] \to S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_3} = \{S_2[k,i] \to S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$R_{e_4} = \{S_3[k-1,i,k] \to S_2[k,i] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$
$$R_{e_5} = \{S_1[k] \to S_2[k,i] : \ 0 \le k < N \ \wedge \ k+1 \le i < N\}$$
$$R_{e_6} = \{S_1[k-1,k,k] \to S_1[k] : \ 1 \le k < N \ \wedge \ k+1 \le i < N\}$$

$P_1 = (e_1)$ is a chain path

$P_2 = (e_2)$ is a broadcast path

$P_3 = (e_3)$ is a broadcast path

Cholesky

$$|\phi_1(E)| + |\phi_2(E)| \le K$$
$$|\phi_1(E)| + |\phi_3(E)| \le K$$

$|\phi_1$

$$\mathrm{Dom}\,(P_1) = \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_2) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$\mathrm{Dom}\,(P_3) = \{S_3[k,i,j] : \ 0 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\}$$
$$\begin{aligned} D_S &:= \mathrm{Dom}\,(P_1) \cap \mathrm{Dom}\,(P_2) \cap \mathrm{Dom}\,(P_3) \\ &= \{S_3[k,i,j] : \ 1 \le k < N \ \wedge \ k+1 \le i < N \ \wedge \ k+1 \le j \le i\} \end{aligned}$$

$$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}(k,i,j) = (0,i,j) \qquad \text{kernel } k_1 = \mathrm{Ker}(\phi_1) = \langle(1,0,0)\rangle$$
$$\phi_2(k,i,j) = (k,j) \qquad \text{kernel } k_2 = \mathrm{Ker}(\phi_2) = \langle(0,1,0)\rangle$$
$$\phi_3(k,i,j) = (k,i) \qquad \text{kernel } k_3 = \mathrm{Ker}(\phi_3) = \langle(0,0,1)\rangle$$

THEOREM 3.10 (BRASCAMP-LIEB INEQUALITY, DISCRETE CASE [CHRIST ET AL. 2013]). *Let d and d_j be nonnegative integers and $\phi_j : \mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ be group homomorphisms for $1 \le j \le m$. Let $0 \le s_1, s_2, \ldots, s_m \le 1$. Suppose that:*

$$\mathrm{rank}\,(H) \le \sum_{j=1}^{m} s_j \cdot \mathrm{rank}\,(\phi_j(H)) \ \textit{for all subgroups } H \textit{ of } \mathbb{Z}^d \qquad (2)$$

Then:

We consider a segmentation of the execution every T IOs. For each segment, we have at most K = S + T input data to perform the statements in the segment.

$$0 \le s_1, s_2, s_3 \le 1$$
$$1 \le s_2 + s_3$$
$$1 \le s_1 + s_3$$
$$1 \le s_1 + s_2$$

$$Q \ge T \times \left\lfloor \frac{N^3/6}{2\cdot(K/3)^{3/2}} \right\rfloor$$

the total number of statements in the execution

the IO of a segment

The minimum number of K-partition in a schedule

the maximum number of statements in a segment

is greater than or equal to

the IO of any schedule

Setting $T = 2S$ (so $K = S + T = 3S$)

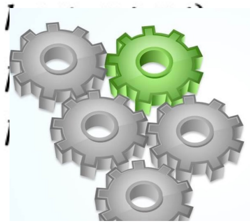
$$Q \ge (2S) \times \frac{N^3/6}{2S^{3/2}} = \boxed{\frac{N^3}{6\sqrt{S}}}$$

```
for(k = 0; k < n; k++) {
     A[k][k] = sqrt(A[k][k]);                 //S1
  for(i = k+1; i < n; i++)
    A[i][k] /= A[k][k];                      // S2
  for(i = k+1; i < n; i++)
    for(j = k+1; j <= i; j++)
       A[i][j] -= A[i][k] * A[j][k];         // S3
}
```

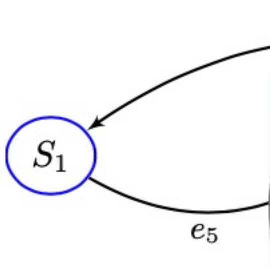
$\mathrm{Dom}(P_1) = \{S_3[k,i,j] :\ 1 < k < N\ \wedge\ k+1 \le i < N\ \wedge\ i$

$+1 \le i < N\ \wedge\ i$

$< N\ \wedge\ i$



$< j \le i\}$



$R_{e_1} = \{S_3[k-1, i,$ $\le i < N\ \wedge\ k+1 \le j \le i\}$

$R_{e_2} = \{S_2[k,j] \rightarrow$ $N\ \wedge\ k+1 \le j \le i\}$

$R_{e_3} = \{S_2[k,i] \rightarrow$ $N\ \wedge\ k+1 \le j \le i\}$

$R_{e_4} = \{S_3[k-1, i,$ $i < N\}$

$R_{e_5} = \{S_1[k] \rightarrow S$

$R_{e_6} = \{S_1[k-1,k,k] \rightarrow S_1[k] :\ 1 \le k < N\ \wedge\ k+1 \le i < N\}$

$\phi_1(k,i,j) = \mathrm{proj}_{(1,0,0)}(k,i,j) = (0,i,j)$ $(1,0,0))$

$\phi_2(k$ k $1,0))$

$\phi_3(k$ ker $0,1))$

THEO INEQUALITY, DISCRETE CASE [CHRIS

d_j *be n* $\mathbb{Z}^d \mapsto \mathbb{Z}^{d_j}$ *be group homomorphi*

$0 \le s_1, s$

$\mathrm{rank}(H) \le \sum_{j=1} s_j \cdot \mathrm{rank}(\phi_j(H))$ *for all subgroups H o*

Then:

the total number of statements in the execution

the IO of a segment

We consider a y T IOs. For each segment a to perform the statement

$$0 \le s_1, s_2, s_3 \le 1$$
$$1 \le s_2 + s_3$$
$$1 \le s_1 + s_3$$
$$1 \le s_1 + s_2$$

$$Q \ge T \times \left\lfloor \frac{N^3/6}{2 \cdot (K/3)^{3/2}} \right\rfloor$$

f K-

the maximum number of statements in a segment

is greater than or equal to

the IO of any schedule

IOLB: I/O Lower Bound
https://iocomplexity.corse.inria.fr/iolb

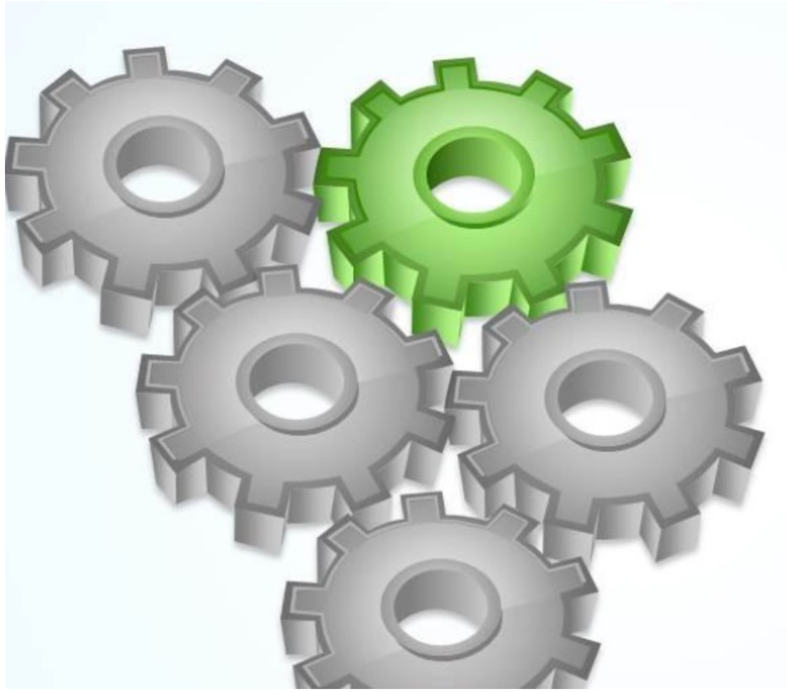
Setting $T = 2S$ (so $K = S + T = 3S$)

$$|\phi_1(E)| + |\phi_2(E)| \le K$$
$$|\phi_1(E)| + |\phi_3(E)| \le K$$

$|\phi_1$

$$Q \ge (2S) \times \frac{N^3/6}{2S^{3/2}} = \boxed{\frac{N^3}{6\sqrt{S}}}$$

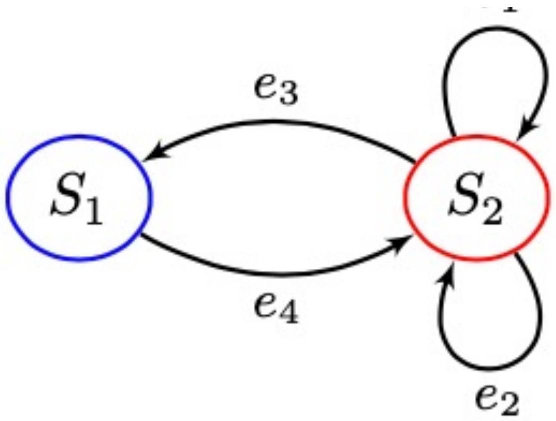
Cholesky



IOLB: I/O Lower Bound https://iocomplexity.corse.inria.fr/iolb

- PET: parse the C code (PET = Polyhedral Extraction Tool)
- DFG (in house) Data Flow Graph
- Piplib: for the ILP (PIP = Parametric Integer Programming)
- GINAC: manipulate symbolic expressions
- Lib ISL: (Integer Set Library) Barvinok: omputes cardinalities of Z polyhedron. Main tool for polyhedral compiling.

```
for(k = 0; k < n; k++) {
   for(i = k+1; i < n; i++)
      A[i][k] /= A[k][k];                                 // S1
   for(i = k+1; i < n; i++)
      for(j = k+1; j < n; j++)
          A[i][j] += A[i][k] * A[k][j];                   // S2
}
```



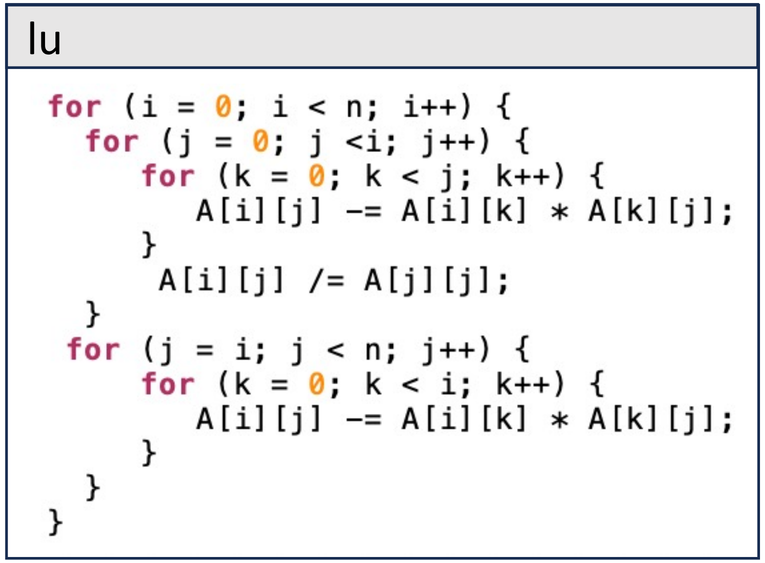
$$Q \geq (2S)\frac{N^3/3}{S^{3/2}} = \boxed{\frac{2N^3}{3\sqrt{S}}}$$

Operational intensity of LU is at most sqrt(S)

LU

LU

similar algorithm as Béreux’s Cholesky (SIMAX 2008)

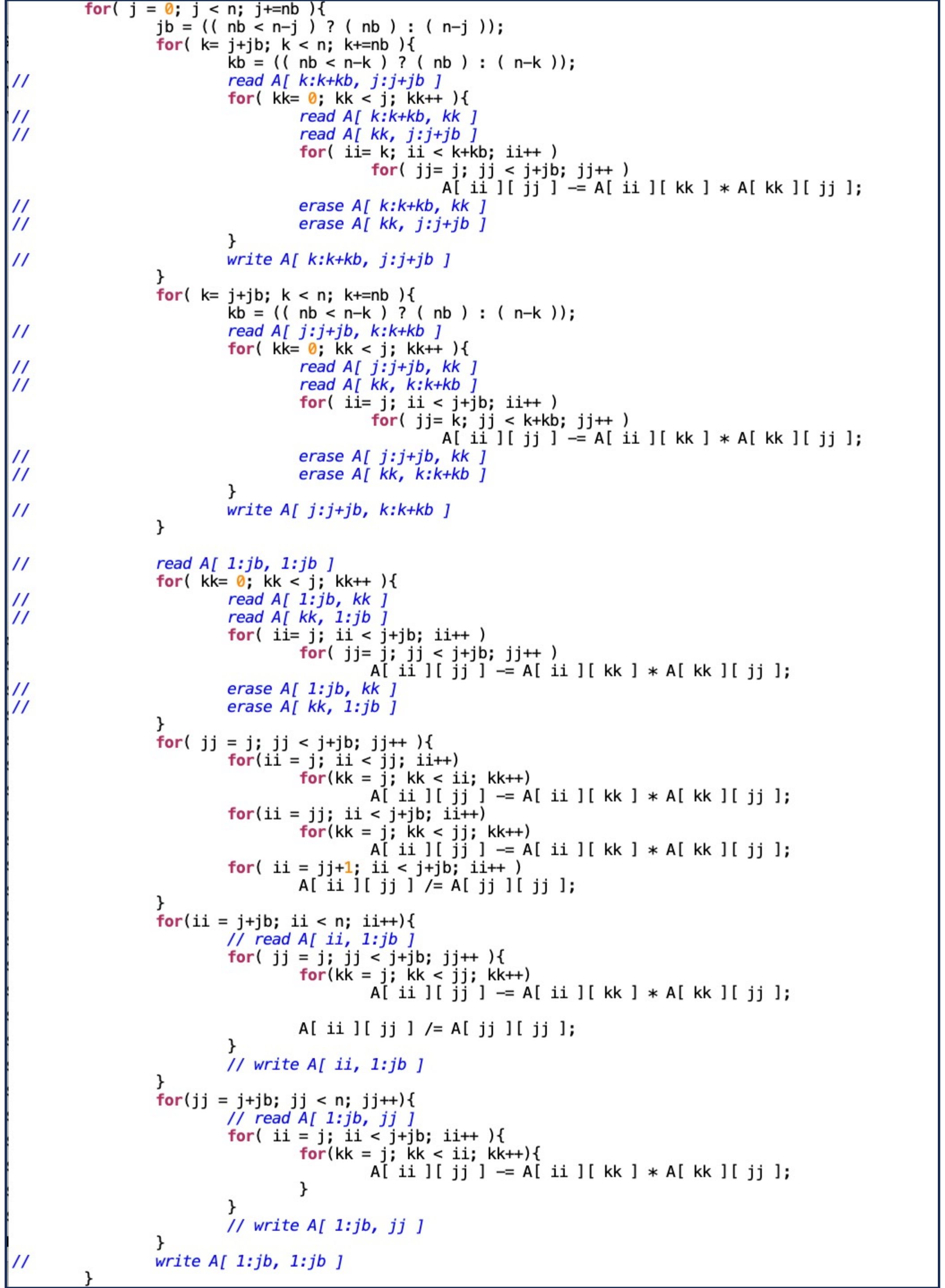


lu

```
for (i = 0; i < n; i++) {
  for (j = 0; j <i; j++) {
     for (k = 0; k < j; k++) {
        A[i][j] -= A[i][k] * A[k][j];
     }
      A[i][j] /= A[j][j];
  }
 for (j = i; j < n; j++) {
     for (k = 0; k < i; k++) {
        A[i][j] -= A[i][k] * A[k][j];
     }
  }
}
```

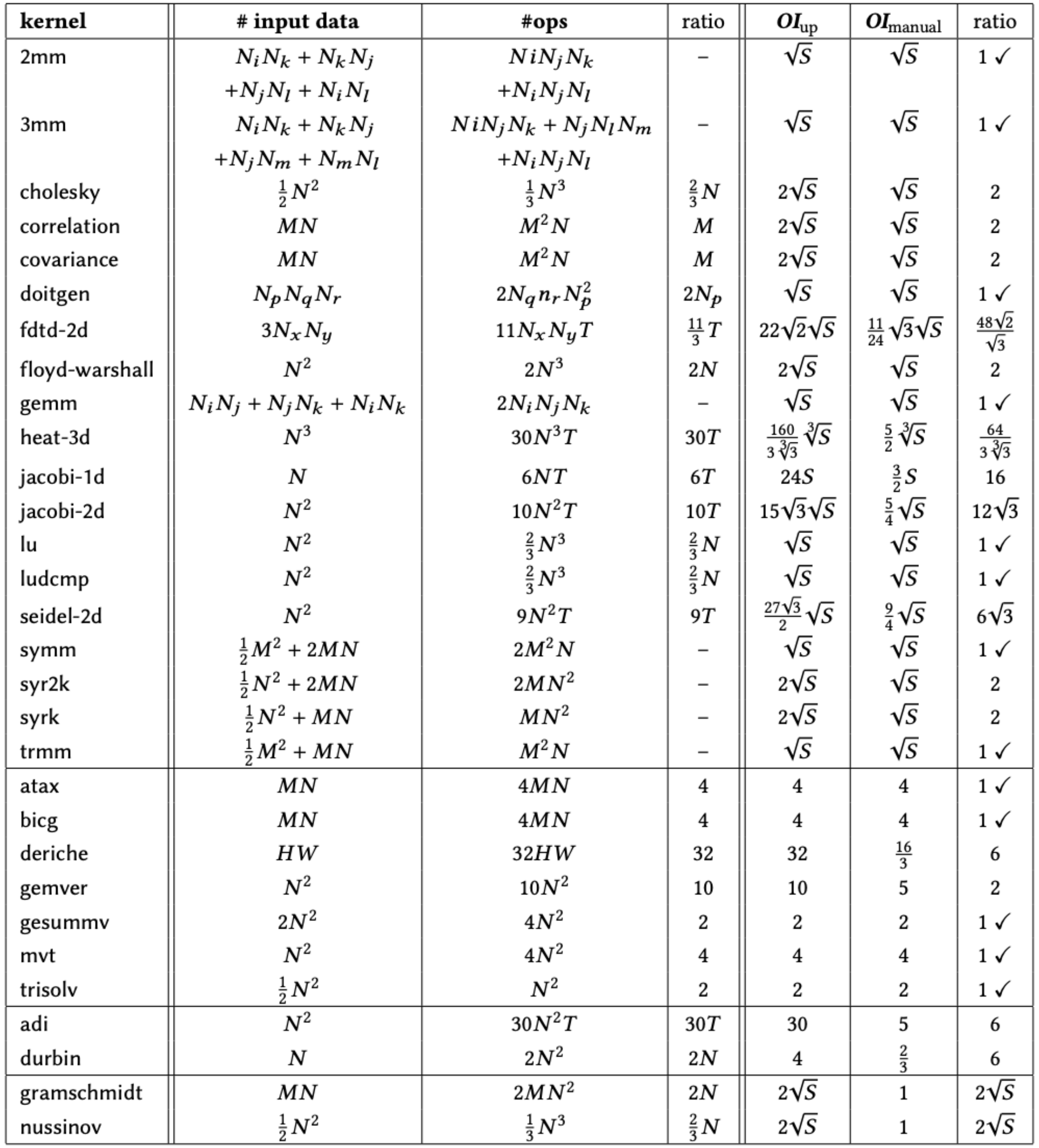
Reorder statements to add blocking in the update

explicit cache directives



```
        for( j = 0; j < n; j+=nb ){
            jb = (( nb < n-j ) ? ( nb ) : ( n-j ));
            for( k= j+jb; k < n; k+=nb ){
                kb = (( nb < n-k ) ? ( nb ) : ( n-k ));
//              read A[ k:k+kb, j:j+jb ]
                for( kk= 0; kk < j; kk++ ){
//                  read A[ k:k+kb, kk ]
//                  read A[ kk, j:j+jb ]
                    for( ii= k; ii < k+kb; ii++ )
                        for( jj= j; jj < j+jb; jj++ )
                            A[ ii ][ jj ] -= A[ ii ][ kk ] * A[ kk ][ jj ];
//                  erase A[ k:k+kb, kk ]
//                  erase A[ kk, j:j+jb ]
                }
//              write A[ k:k+kb, j:j+jb ]
            }
            for( k= j+jb; k < n; k+=nb ){
                kb = (( nb < n-k ) ? ( nb ) : ( n-k ));
//              read A[ j:j+jb, k:k+kb ]
                for( kk= 0; kk < j; kk++ ){
//                  read A[ j:j+jb, kk ]
//                  read A[ kk, k:k+kb ]
                    for( ii= j; ii < j+jb; ii++ )
                        for( jj= k; jj < k+kb; jj++ )
                            A[ ii ][ jj ] -= A[ ii ][ kk ] * A[ kk ][ jj ];
//                  erase A[ j:j+jb, kk ]
//                  erase A[ kk, k:k+kb ]
                }
//              write A[ j:j+jb, k:k+kb ]
            }

//          read A[ 1:jb, 1:jb ]
            for( kk= 0; kk < j; kk++ ){
//              read A[ 1:jb, kk ]
//              read A[ kk, 1:jb ]
                for( ii= j; ii < j+jb; ii++ )
                    for( jj= j; jj < j+jb; jj++ )
                        A[ ii ][ jj ] -= A[ ii ][ kk ] * A[ kk ][ jj ];
//              erase A[ 1:jb, kk ]
//              erase A[ kk, 1:jb ]
            }
            for( jj = j; jj < j+jb; jj++ ){
                for(ii = j; ii < jj; ii++)
                    for(kk = j; kk < ii; kk++)
                        A[ ii ][ jj ] -= A[ ii ][ kk ] * A[ kk ][ jj ];
                for(ii = jj; ii < j+jb; ii++)
                    for(kk = j; kk < jj; kk++)
                        A[ ii ][ jj ] -= A[ ii ][ kk ] * A[ kk ][ jj ];
                for( ii = jj+1; ii < j+jb; ii++ )
                    A[ ii ][ jj ] /= A[ jj ][ jj ];
            }
            for(ii = j+jb; ii < n; ii++){
                // read A[ ii, 1:jb ]
                for( jj = j; jj < j+jb; jj++ ){
                    for(kk = j; kk < jj; kk++)
                        A[ ii ][ jj ] -= A[ ii ][ kk ] * A[ kk ][ jj ];

                    A[ ii ][ jj ] /= A[ jj ][ jj ];
                }
                // write A[ ii, 1:jb ]
            }
            for(jj = j+jb; jj < n; jj++){
                // read A[ 1:jb, jj ]
                for( ii = j; ii < j+jb; ii++ ){
                    for(kk = j; kk < ii; kk++){
                        A[ ii ][ jj ] -= A[ ii ][ kk ] * A[ kk ][ jj ];
                    }
                }
                // write A[ 1:jb, jj ]
            }
//          write A[ 1:jb, 1:jb ]
        }
```

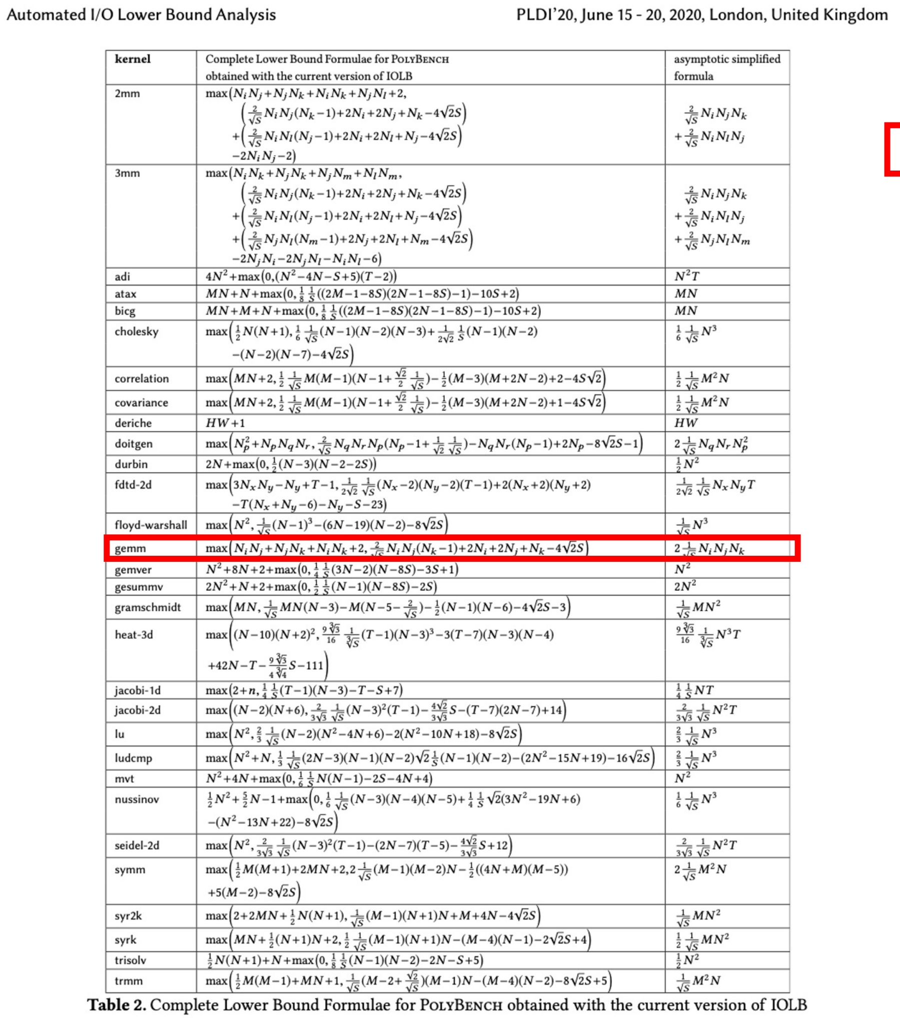
Operational intensity of LU is at least sqrt(S) polybench test suite



kernel	# input data	#ops	ratio	OI_{up}	OI_{manual}	ratio
2mm	$N_iN_k + N_kN_j + N_jN_l + N_iN_l$	$NiN_jN_k + N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
3mm	$N_iN_k + N_kN_j + N_jN_m + N_mN_l$	$NiN_jN_k + N_jN_lN_m + N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
cholesky	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	$\sqrt{S}$	2
correlation	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
covariance	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
doitgen	$N_pN_qN_r$	$2N_qn_rN_p^2$	$2N_p$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
fdtd-2d	$3N_xN_y$	$11N_xN_yT$	$\frac{11}{3}T$	$22\sqrt{2}\sqrt{S}$	$\frac{11}{24}\sqrt{3}\sqrt{S}$	$\frac{48\sqrt{2}}{\sqrt{3}}$
floyd-warshall	N^2	$2N^3$	$2N$	$2\sqrt{S}$	$\sqrt{S}$	2
gemm	$N_iN_j + N_jN_k + N_iN_k$	$2N_iN_jN_k$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
heat-3d	N^3	$30N^3T$	$30T$	$\frac{160}{3\sqrt[3]{3}}\sqrt[3]{S}$	$\frac{5}{2}\sqrt[3]{S}$	$\frac{64}{3\sqrt[3]{3}}$
jacobi-1d	N	$6NT$	$6T$	$24S$	$\frac{3}{2}S$	16
jacobi-2d	N^2	$10N^2T$	$10T$	$15\sqrt{3}\sqrt{S}$	$\frac{5}{4}\sqrt{S}$	$12\sqrt{3}$
lu	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
ludcmp	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
seidel-2d	N^2	$9N^2T$	$9T$	$\frac{27\sqrt{3}}{2}\sqrt{S}$	$\frac{9}{4}\sqrt{S}$	$6\sqrt{3}$
symm	$\frac{1}{2}M^2 + 2MN$	$2M^2N$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
syr2k	$\frac{1}{2}N^2 + 2MN$	$2MN^2$	–	$2\sqrt{S}$	$\sqrt{S}$	2
syrk	$\frac{1}{2}N^2 + MN$	MN^2	–	$2\sqrt{S}$	$\sqrt{S}$	2
trmm	$\frac{1}{2}M^2 + MN$	M^2N	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
atax	MN	$4MN$	4	4	4	1 ✓
bicg	MN	$4MN$	4	4	4	1 ✓
deriche	HW	$32HW$	32	32	$\frac{16}{3}$	6
gemver	N^2	$10N^2$	10	10	5	2
gesummv	$2N^2$	$4N^2$	2	2	2	1 ✓
mvt	N^2	$4N^2$	4	4	4	1 ✓
trisolv	$\frac{1}{2}N^2$	N^2	2	2	2	1 ✓
adi	N^2	$30N^2T$	$30T$	30	5	6
durbin	N	$2N^2$	$2N$	4	$\frac{2}{3}$	6
gramschmidt	MN	$2MN^2$	$2N$	$2\sqrt{S}$	1	$2\sqrt{S}$
nussinov	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	1	$2\sqrt{S}$

A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)

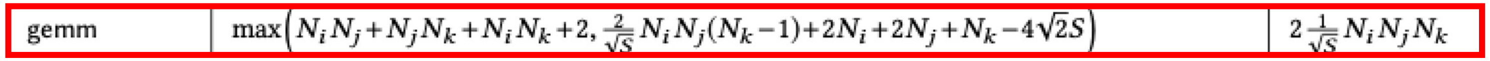
A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)



Automated I/O Lower Bound Analysis PLDI'20, June 15 - 20, 2020, London, United Kingdom

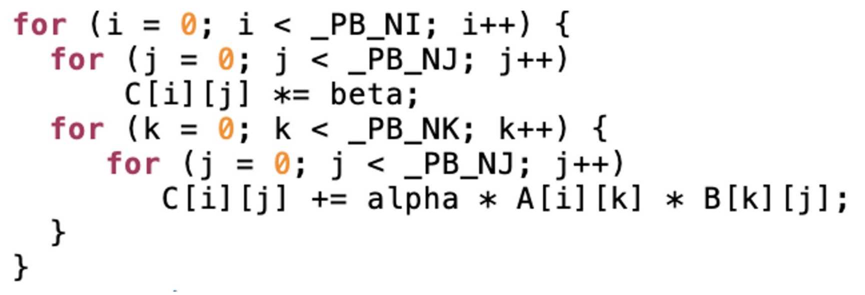
kernel	Complete Lower Bound Formulae for PolyBench obtained with the current version of IOLB	asymptotic simplified formula
2mm	$\max\Big(N_iN_j+N_jN_k+N_iN_k+N_jN_l+2,$ $\Big(\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\Big)$ $+\Big(\frac{2}{\sqrt{S}}N_iN_l(N_j-1)+2N_i+2N_l+N_j-4\sqrt{2}S\Big)$ $-2N_iN_j-2\Big)$	$\frac{2}{\sqrt{S}}N_iN_jN_k$ $+\frac{2}{\sqrt{S}}N_iN_lN_j$
3mm	$\max\Big(N_iN_k+N_jN_k+N_jN_m+N_lN_m,$ $\Big(\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\Big)$ $+\Big(\frac{2}{\sqrt{S}}N_iN_l(N_j-1)+2N_i+2N_l+N_j-4\sqrt{2}S\Big)$ $+\Big(\frac{2}{\sqrt{S}}N_jN_l(N_m-1)+2N_j+2N_l+N_m-4\sqrt{2}S\Big)$ $-2N_jN_i-2N_jN_l-N_iN_l-6\Big)$	$\frac{2}{\sqrt{S}}N_iN_jN_k$ $+\frac{2}{\sqrt{S}}N_iN_lN_j$ $+\frac{2}{\sqrt{S}}N_jN_lN_m$
adi	$4N^2+\max\big(0,(N^2-4N-S+5)(T-2)\big)$	N^2T
atax	$MN+N+\max\big(0,\frac{1}{8}\frac{1}{S}((2M-1-8S)(2N-1-8S)-1)-10S+2\big)$	MN
bicg	$MN+M+N+\max\big(0,\frac{1}{8}\frac{1}{S}((2M-1-8S)(2N-1-8S)-1)-10S+2\big)$	MN
cholesky	$\max\Big(\frac{1}{2}N(N+1),\frac{1}{6}\frac{1}{\sqrt{S}}(N-1)(N-2)(N-3)+\frac{1}{2\sqrt{2}}\frac{1}{S}(N-1)(N-2)$ $-(N-2)(N-7)-4\sqrt{2}S\Big)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$
correlation	$\max\Big(MN+2,\frac{1}{2}\frac{1}{\sqrt{S}}M(M-1)(N-1+\frac{\sqrt{2}}{2}\frac{1}{\sqrt{S}})-\frac{1}{2}(M-3)(M+2N-2)+2-4S\sqrt{2}\Big)$	$\frac{1}{2}\frac{1}{\sqrt{S}}M^2N$
covariance	$\max\Big(MN+2,\frac{1}{2}\frac{1}{\sqrt{S}}M(M-1)(N-1+\frac{\sqrt{2}}{2}\frac{1}{\sqrt{S}})-\frac{1}{2}(M-3)(M+2N-2)+1-4S\sqrt{2}\Big)$	$\frac{1}{2}\frac{1}{\sqrt{S}}M^2N$
deriche	$HW+1$	HW
doitgen	$\max\Big(N_p^2+N_pN_qN_r,\frac{2}{\sqrt{S}}N_qN_rN_p(N_p-1+\frac{1}{\sqrt{2}}\frac{1}{\sqrt{S}})-N_qN_r(N_p-1)+2N_p-8\sqrt{2}S-1\Big)$	$2\frac{1}{\sqrt{S}}N_qN_rN_p^2$
durbin	$2N+\max\big(0,\frac{1}{2}(N-3)(N-2-2S)\big)$	$\frac{1}{2}N^2$
fdtd-2d	$\max\Big(3N_xN_y-N_y+T-1,\frac{1}{2\sqrt{2}}\frac{1}{\sqrt{S}}(N_x-2)(N_y-2)(T-1)+2(N_x+2)(N_y+2)$ $-T(N_x+N_y-6)-N_y-S-23\Big)$	$\frac{1}{2\sqrt{2}}\frac{1}{\sqrt{S}}N_xN_yT$
floyd-warshall	$\max\Big(N^2,\frac{1}{\sqrt{S}}(N-1)^3-(6N-19)(N-2)-8\sqrt{2}S\Big)$	$\frac{1}{\sqrt{S}}N^3$
gemm	$\max\Big(N_iN_j+N_jN_k+N_iN_k+2,\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\Big)$	$2\frac{1}{\sqrt{S}}N_iN_jN_k$
gemver	$N^2+8N+2+\max\big(0,\frac{1}{4}\frac{1}{S}(3N-2)(N-8S)-3S+1\big)$	N^2
gesummv	$2N^2+N+2+\max\big(0,\frac{1}{2}\frac{1}{S}(N-1)(N-8S)-2S\big)$	$2N^2$
gramschmidt	$\max\Big(MN,\frac{1}{\sqrt{S}}MN(N-3)-M(N-5-\frac{2}{\sqrt{S}})-\frac{1}{2}(N-1)(N-6)-4\sqrt{2}S-3\Big)$	$\frac{1}{\sqrt{S}}MN^2$
heat-3d	$\max\Big((N-10)(N+2)^2,\frac{9\sqrt[3]{3}}{16}\frac{1}{\sqrt[3]{S}}(T-1)(N-3)^3-3(T-7)(N-3)(N-4)$ $+42N-T-\frac{9\sqrt[3]{3}}{4\sqrt[3]{4}}S-111\Big)$	$\frac{9\sqrt[3]{3}}{16}\frac{1}{\sqrt[3]{S}}N^3T$
jacobi-1d	$\max\big(2+n,\frac{1}{4}\frac{1}{S}(T-1)(N-3)-T-S+7\big)$	$\frac{1}{4}\frac{1}{S}NT$
jacobi-2d	$\max\Big((N-2)(N+6),\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}(N-3)^2(T-1)-\frac{4\sqrt{2}}{3\sqrt{3}}S-(T-7)(2N-7)+14\Big)$	$\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}N^2T$
lu	$\max\Big(N^2,\frac{2}{3}\frac{1}{\sqrt{S}}(N-2)(N^2-4N+6)-2(N^2-10N+18)-8\sqrt{2}S\Big)$	$\frac{2}{3}\frac{1}{\sqrt{S}}N^3$
ludcmp	$\max\Big(N^2+N,\frac{1}{3}\frac{1}{\sqrt{S}}(2N-3)(N-1)(N-2)\sqrt{2}\frac{1}{S}(N-1)(N-2)-(2N^2-15N+19)-16\sqrt{2}S\Big)$	$\frac{2}{3}\frac{1}{\sqrt{S}}N^3$
mvt	$N^2+4N+\max\big(0,\frac{1}{6}\frac{1}{S}N(N-1)-2S-4N+4\big)$	N^2
nussinov	$\frac{1}{2}N^2+\frac{5}{2}N-1+\max\Big(0,\frac{1}{6}\frac{1}{\sqrt{S}}(N-3)(N-4)(N-5)+\frac{1}{4}\frac{1}{S}\sqrt{2}(3N^2-19N+6)$ $-(N^2-13N+22)-8\sqrt{2}S\Big)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$
seidel-2d	$\max\Big(N^2,\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}(N-3)^2(T-1)-(2N-7)(T-5)-\frac{4\sqrt{2}}{3\sqrt{3}}S+12\Big)$	$\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}N^2T$
symm	$\max\Big(\frac{1}{2}M(M+1)+2MN+2,2\frac{1}{\sqrt{S}}(M-1)(M-2)N-\frac{1}{2}((4N+M)(M-5))$ $+5(M-2)-8\sqrt{2}S\Big)$	$2\frac{1}{\sqrt{S}}M^2N$
syr2k	$\max\Big(2+2MN+\frac{1}{2}N(N+1),\frac{1}{\sqrt{S}}(M-1)(N+1)N+M+4N-4\sqrt{2}S\Big)$	$\frac{1}{\sqrt{S}}MN^2$
syrk	$\max\Big(MN+\frac{1}{2}(N+1)N+2,\frac{1}{2}\frac{1}{\sqrt{S}}(M-1)(N+1)N-(M-4)(N-1)-2\sqrt{2}S+4\Big)$	$\frac{1}{2}\frac{1}{\sqrt{S}}MN^2$
trisolv	$\frac{1}{2}N(N+1)+N+\max\big(0,\frac{1}{8}\frac{1}{S}(N-1)(N-2)-2N-S+5\big)$	$\frac{1}{2}N^2$
trmm	$\max\Big(\frac{1}{2}M(M-1)+MN+1,\frac{1}{\sqrt{S}}(M-2+\frac{\sqrt{2}}{\sqrt{S}})(M-1)N-(M-4)(N-2)-8\sqrt{2}S+5\Big)$	$\frac{1}{\sqrt{S}}M^2N$

Table 2. Complete Lower Bound Formulae for PolyBench obtained with the current version of IOLB



gemm	$\max\Big(N_iN_j+N_jN_k+N_iN_k+2,\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\Big)$	$2\frac{1}{\sqrt{S}}N_iN_jN_k$

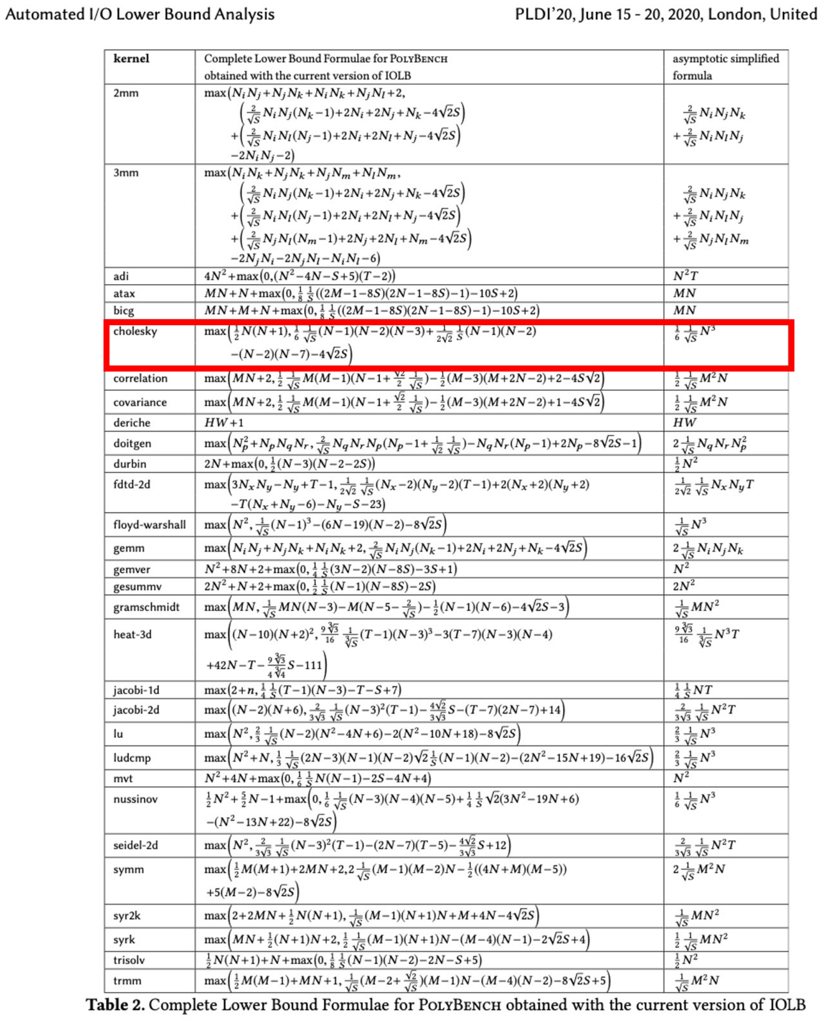
gemm



```
for (i = 0; i < _PB_NI; i++) {
  for (j = 0; j < _PB_NJ; j++)
      C[i][j] *= beta;
  for (k = 0; k < _PB_NK; k++) {
     for (j = 0; j < _PB_NJ; j++)
        C[i][j] += alpha * A[i][k] * B[k][j];
  }
}
```

Operational intensity of GEMM is at sqrt(S)

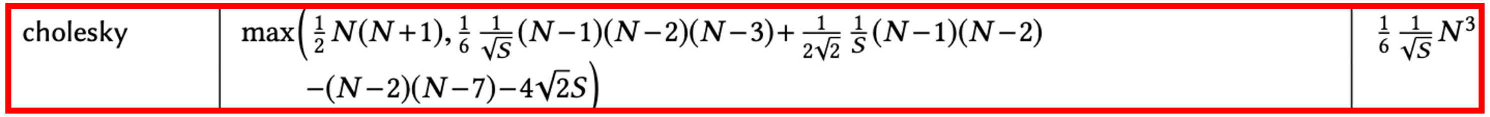
A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)



Automated I/O Lower Bound Analysis PLDI'20, June 15 - 20, 2020, London, United Kingdom

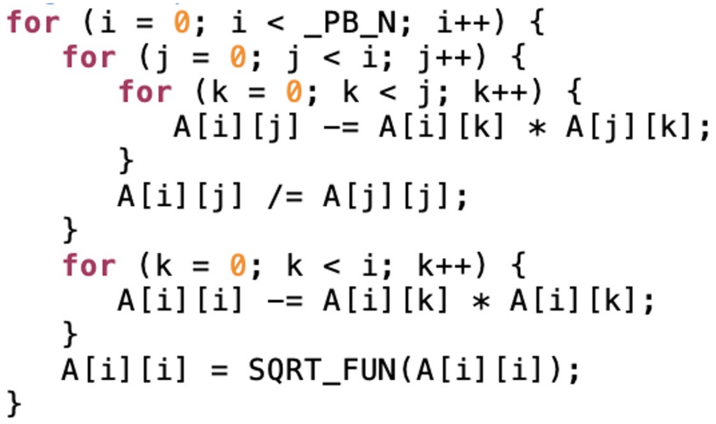
kernel	Complete Lower Bound Formulae for PolyBench obtained with the current version of IOLB	asymptotic simplified formula
2mm	$\max\big(N_iN_j+N_jN_k+N_iN_k+N_jN_l+2,$ $\big(\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\big)$ $+\big(\frac{2}{\sqrt{S}}N_iN_l(N_j-1)+2N_i+2N_l+N_j-4\sqrt{2}S\big)$ $-2N_iN_j-2\big)$	$\frac{2}{\sqrt{S}}N_iN_jN_k$ $+\frac{2}{\sqrt{S}}N_iN_lN_j$
3mm	$\max\big(N_iN_k+N_jN_k+N_jN_m+N_lN_m,$ $\big(\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\big)$ $+\big(\frac{2}{\sqrt{S}}N_iN_l(N_j-1)+2N_i+2N_l+N_j-4\sqrt{2}S\big)$ $+\big(\frac{2}{\sqrt{S}}N_jN_l(N_m-1)+2N_j+2N_l+N_m-4\sqrt{2}S\big)$ $-2N_jN_i-2N_jN_l-N_iN_l-6\big)$	$\frac{2}{\sqrt{S}}N_iN_jN_k$ $+\frac{2}{\sqrt{S}}N_iN_lN_j$ $+\frac{2}{\sqrt{S}}N_jN_lN_m$
adi	$4N^2+\max\big(0,(N^2-4N-S+5)(T-2)\big)$	N^2T
atax	$MN+N+\max\big(0,\frac{1}{8}\frac{1}{S}((2M-1-8S)(2N-1-8S)-1)-10S+2\big)$	MN
bicg	$MN+M+N+\max\big(0,\frac{1}{8}\frac{1}{S}((2M-1-8S)(2N-1-8S)-1)-10S+2\big)$	MN
cholesky	$\max\Big(\frac{1}{2}N(N+1),\frac{1}{6}\frac{1}{\sqrt{S}}(N-1)(N-2)(N-3)+\frac{1}{2\sqrt{2}}\frac{1}{S}(N-1)(N-2)$ $-(N-2)(N-7)-4\sqrt{2}S\Big)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$
correlation	$\max\Big(MN+2,\frac{1}{2}\frac{1}{\sqrt{S}}M(M-1)(N-1+\frac{\sqrt{2}}{2}\frac{1}{\sqrt{S}})-\frac{1}{2}(M-3)(M+2N-2)+2-4S\sqrt{2}\Big)$	$\frac{1}{2}\frac{1}{\sqrt{S}}M^2N$
covariance	$\max\Big(MN+2,\frac{1}{2}\frac{1}{\sqrt{S}}M(M-1)(N-1+\frac{\sqrt{2}}{2}\frac{1}{\sqrt{S}})-\frac{1}{2}(M-3)(M+2N-2)+1-4S\sqrt{2}\Big)$	$\frac{1}{2}\frac{1}{\sqrt{S}}M^2N$
deriche	$HW+1$	HW
doitgen	$\max\Big(N_p^2+N_pN_qN_r,\frac{2}{\sqrt{S}}N_qN_rN_p(N_p-1+\frac{1}{\sqrt{2}}\frac{1}{\sqrt{S}})-N_qN_r(N_p-1)+2N_p-8\sqrt{2}S-1\Big)$	$2\frac{1}{\sqrt{S}}N_qN_rN_p^2$
durbin	$2N+\max\big(0,\frac{1}{2}(N-3)(N-2-2S)\big)$	$\frac{1}{2}N^2$
fdtd-2d	$\max\Big(3N_xN_y-N_y+T-1,\frac{1}{2\sqrt{2}}\frac{1}{\sqrt{S}}(N_x-2)(N_y-2)(T-1)+2(N_x+2)(N_y+2)$ $-T(N_x+N_y-6)-N_y-S-23\Big)$	$\frac{1}{2\sqrt{2}}\frac{1}{\sqrt{S}}N_xN_yT$
floyd-warshall	$\max\Big(N^2,\frac{1}{\sqrt{S}}(N-1)^3-(6N-19)(N-2)-8\sqrt{2}S\Big)$	$\frac{1}{\sqrt{S}}N^3$
gemm	$\max\Big(N_iN_j+N_jN_k+N_iN_k+2,\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\Big)$	$2\frac{1}{\sqrt{S}}N_iN_jN_k$
gemver	$N^2+8N+2+\max\big(0,\frac{1}{4}\frac{1}{S}(3N-2)(N-8S)-3S+1\big)$	N^2
gesummv	$2N^2+N+2+\max\big(0,\frac{1}{2}\frac{1}{S}(N-1)(N-8S)-2S\big)$	$2N^2$
gramschmidt	$\max\Big(MN,\frac{1}{\sqrt{S}}MN(N-3)-M(N-5-\frac{2}{\sqrt{S}})-\frac{1}{2}(N-1)(N-6)-4\sqrt{2}S-3\Big)$	$\frac{1}{\sqrt{S}}MN^2$
heat-3d	$\max\Big((N-10)(N+2)^2,\frac{9\sqrt[3]{3}}{16}\frac{1}{\sqrt[3]{S}}(T-1)(N-3)^3-3(T-7)(N-3)(N-4)$ $+42N-T-\frac{9\sqrt[3]{3}}{4\sqrt[3]{4}}S-111\Big)$	$\frac{9\sqrt[3]{3}}{16}\frac{1}{\sqrt[3]{S}}N^3T$
jacobi-1d	$\max\big(2+n,\frac{1}{4}\frac{1}{S}(T-1)(N-3)-T-S+7\big)$	$\frac{1}{4}\frac{1}{S}NT$
jacobi-2d	$\max\Big((N-2)(N+6),\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}(N-3)^2(T-1)-\frac{4\sqrt{2}}{3\sqrt{3}}S-(T-7)(2N-7)+14\Big)$	$\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}N^2T$
lu	$\max\Big(N^2,\frac{2}{3}\frac{1}{\sqrt{S}}(N-2)(N^2-4N+6)-2(N^2-10N+18)-8\sqrt{2}S\Big)$	$\frac{2}{3}\frac{1}{\sqrt{S}}N^3$
ludcmp	$\max\Big(N^2+N,\frac{1}{3}\frac{1}{\sqrt{S}}(2N-3)(N-1)(N-2)\sqrt{2}\frac{1}{S}(N-1)(N-2)-(2N^2-15N+19)-16\sqrt{2}S\Big)$	$\frac{2}{3}\frac{1}{\sqrt{S}}N^3$
mvt	$N^2+4N+\max\big(0,\frac{1}{6}\frac{1}{S}N(N-1)-2S-4N+4\big)$	N^2
nussinov	$\frac{1}{2}N^2+\frac{5}{2}N-1+\max\Big(0,\frac{1}{6}\frac{1}{\sqrt{S}}(N-3)(N-4)(N-5)+\frac{1}{4}\frac{1}{S}\sqrt{2}(3N^2-19N+6)$ $-(N^2-13N+22)-8\sqrt{2}S\Big)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$
seidel-2d	$\max\Big(N^2,\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}(N-3)^2(T-1)-(2N-7)(T-5)-\frac{4\sqrt{2}}{3\sqrt{3}}S+12\Big)$	$\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}N^2T$
symm	$\max\Big(\frac{1}{2}M(M+1)+2MN+2,2\frac{1}{\sqrt{S}}(M-1)(M-2)N-\frac{1}{2}((4N+M)(M-5))$ $+5(M-2)-8\sqrt{2}S\Big)$	$2\frac{1}{\sqrt{S}}M^2N$
syr2k	$\max\Big(2+2MN+\frac{1}{2}N(N+1),\frac{1}{\sqrt{S}}(M-1)(N+1)N+M+4N-4\sqrt{2}S\Big)$	$\frac{1}{\sqrt{S}}MN^2$
syrk	$\max\Big(MN+\frac{1}{2}(N+1)N+2,\frac{1}{2}\frac{1}{\sqrt{S}}(M-1)(N+1)N-(M-4)(N-1)-2\sqrt{2}S+4\Big)$	$\frac{1}{2}\frac{1}{\sqrt{S}}MN^2$
trisolv	$\frac{1}{2}N(N+1)+N+\max\big(0,\frac{1}{8}\frac{1}{S}(N-1)(N-2)-2N-S+5\big)$	$\frac{1}{2}N^2$
trmm	$\max\Big(\frac{1}{2}M(M-1)+MN+1,\frac{1}{\sqrt{S}}(M-2+\frac{\sqrt{2}}{\sqrt{S}})(M-1)N-(M-4)(N-2)-8\sqrt{2}S+5\Big)$	$\frac{1}{\sqrt{S}}M^2N$

Table 2. Complete Lower Bound Formulae for PolyBench obtained with the current version of IOLB



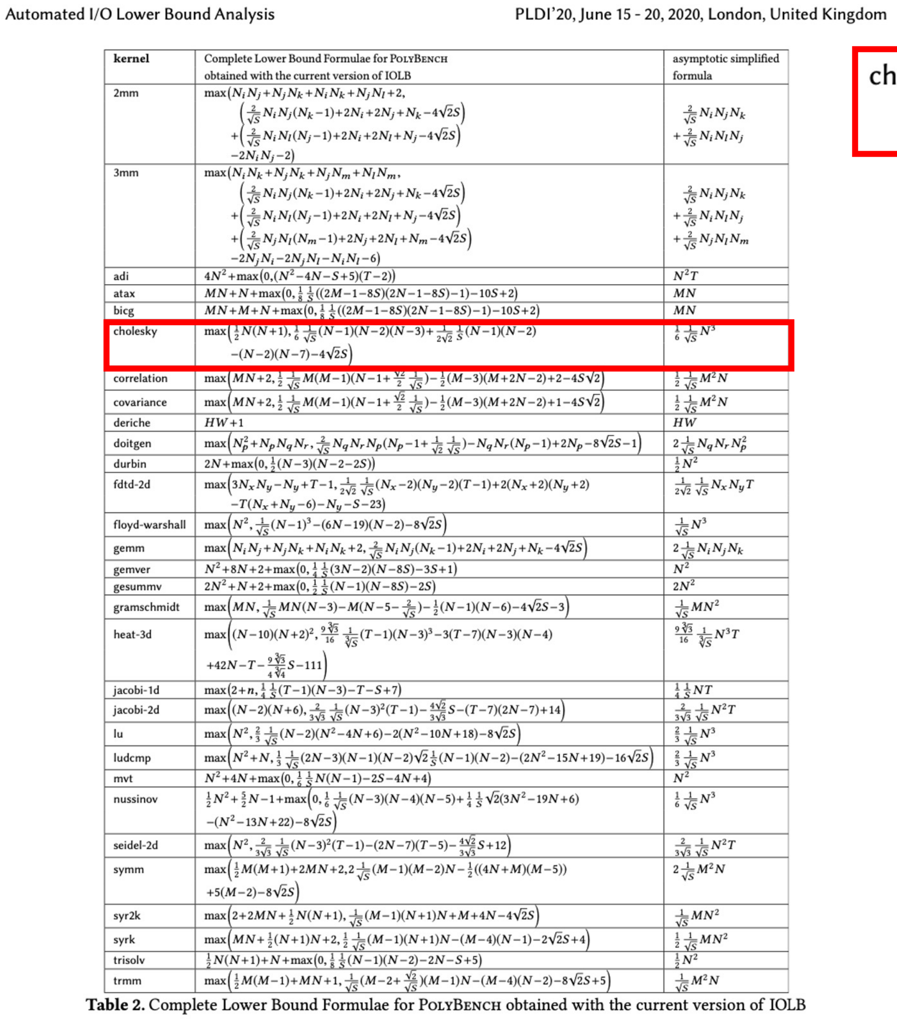
cholesky	$\max\Big(\frac{1}{2}N(N+1),\frac{1}{6}\frac{1}{\sqrt{S}}(N-1)(N-2)(N-3)+\frac{1}{2\sqrt{2}}\frac{1}{S}(N-1)(N-2)$ $-(N-2)(N-7)-4\sqrt{2}S\Big)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$

cholesky



```
for (i = 0; i < _PB_N; i++) {
   for (j = 0; j < i; j++) {
      for (k = 0; k < j; k++) {
         A[i][j] -= A[i][k] * A[j][k];
      }
      A[i][j] /= A[j][j];
   }
   for (k = 0; k < i; k++) {
      A[i][i] -= A[i][k] * A[i][k];
   }
   A[i][i] = SQRT_FUN(A[i][i]);
}
```

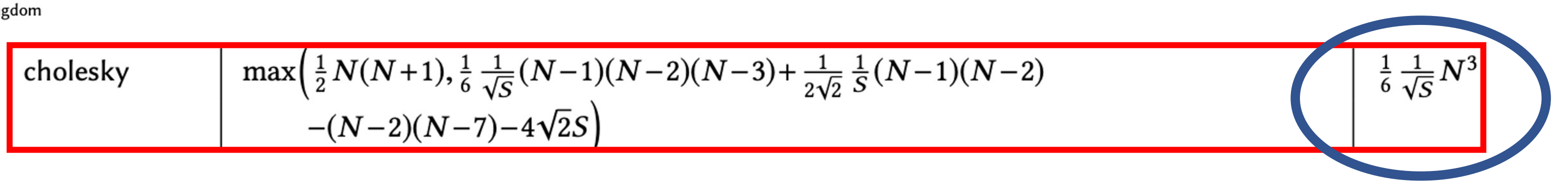
A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)



Automated I/O Lower Bound Analysis PLDI'20, June 15 - 20, 2020, London, United Kingdom

kernel	Complete Lower Bound Formulae for PolyBench obtained with the current version of IOLB	asymptotic simplified formula
2mm	$\max\Big(N_iN_j+N_jN_k+N_iN_k+N_jN_l+2, \left(\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\right) +\left(\frac{2}{\sqrt{S}}N_iN_l(N_j-1)+2N_i+2N_l+N_j-4\sqrt{2}S\right) -2N_iN_j-2\Big)$	$\frac{2}{\sqrt{S}}N_iN_jN_k +\frac{2}{\sqrt{S}}N_iN_lN_j$
3mm	$\max\Big(N_iN_k+N_jN_k+N_jN_m+N_lN_m, \left(\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\right) +\left(\frac{2}{\sqrt{S}}N_iN_l(N_j-1)+2N_i+2N_l+N_j-4\sqrt{2}S\right) +\left(\frac{2}{\sqrt{S}}N_jN_l(N_m-1)+2N_j+2N_l+N_m-4\sqrt{2}S\right) -2N_jN_i-2N_jN_l-N_iN_l-6\Big)$	$\frac{2}{\sqrt{S}}N_iN_jN_k +\frac{2}{\sqrt{S}}N_iN_lN_j +\frac{2}{\sqrt{S}}N_jN_lN_m$
adi	$4N^2+\max(0,(N^2-4N-S+5)(T-2))$	N^2T
atax	$MN+N+\max(0,\frac{1}{8}\frac{1}{S}((2M-1-8S)(2N-1-8S)-1)-10S+2)$	MN
bicg	$MN+M+N+\max(0,\frac{1}{8}\frac{1}{S}((2M-1-8S)(2N-1-8S)-1)-10S+2)$	MN
cholesky	$\max\left(\frac{1}{2}N(N+1),\frac{1}{6}\frac{1}{\sqrt{S}}(N-1)(N-2)(N-3)+\frac{1}{2\sqrt{2}}\frac{1}{S}(N-1)(N-2) -(N-2)(N-7)-4\sqrt{2}S\right)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$
correlation	$\max\left(MN+2,\frac{1}{2}\frac{1}{\sqrt{S}}M(M-1)(N-1+\frac{\sqrt{2}}{2}\frac{1}{\sqrt{S}})-\frac{1}{2}(M-3)(M+2N-2)+2-4S\sqrt{2}\right)$	$\frac{1}{2}\frac{1}{\sqrt{S}}M^2N$
covariance	$\max\left(MN+2,\frac{1}{2}\frac{1}{\sqrt{S}}M(M-1)(N-1+\frac{\sqrt{2}}{2}\frac{1}{\sqrt{S}})-\frac{1}{2}(M-3)(M+2N-2)+1-4S\sqrt{2}\right)$	$\frac{1}{2}\frac{1}{\sqrt{S}}M^2N$
deriche	$HW+1$	HW
doitgen	$\max\left(N_p^2+N_pN_qN_r,\frac{2}{\sqrt{S}}N_qN_rN_p(N_p-1+\frac{1}{\sqrt{2}}\frac{1}{\sqrt{S}})-N_qN_r(N_p-1)+2N_p-8\sqrt{2}S-1\right)$	$2\frac{1}{\sqrt{S}}N_qN_rN_p^2$
durbin	$2N+\max(0,\frac{1}{2}(N-3)(N-2-2S))$	$\frac{1}{2}N^2$
fdtd-2d	$\max\left(3N_xN_y-N_y+T-1,\frac{1}{2\sqrt{2}}\frac{1}{\sqrt{S}}(N_x-2)(N_y-2)(T-1)+2(N_x+2)(N_y+2) -T(N_x+N_y-6)-N_y-S-23\right)$	$\frac{1}{2\sqrt{2}}\frac{1}{\sqrt{S}}N_xN_yT$
floyd-warshall	$\max\left(N^2,\frac{1}{\sqrt{S}}(N-1)^3-(6N-19)(N-2)-8\sqrt{2}S\right)$	$\frac{1}{\sqrt{S}}N^3$
gemm	$\max\left(N_iN_j+N_jN_k+N_iN_k+2,\frac{2}{\sqrt{S}}N_iN_j(N_k-1)+2N_i+2N_j+N_k-4\sqrt{2}S\right)$	$2\frac{1}{\sqrt{S}}N_iN_jN_k$
gemver	$N^2+8N+2+\max(0,\frac{1}{4}\frac{1}{S}(3N-2)(N-8S)-3S+1)$	N^2
gesummv	$2N^2+N+2+\max(0,\frac{1}{2}\frac{1}{S}(N-1)(N-8S)-2S)$	$2N^2$
gramschmidt	$\max\left(MN,\frac{1}{\sqrt{S}}MN(N-3)-M(N-5-\frac{2}{\sqrt{S}})-\frac{1}{2}(N-1)(N-6)-4\sqrt{2}S-3\right)$	$\frac{1}{\sqrt{S}}MN^2$
heat-3d	$\max\left((N-10)(N+2)^2,\frac{9\sqrt[3]{3}}{16}\frac{1}{\sqrt[3]{S}}(T-1)(N-3)^3-3(T-7)(N-3)(N-4) +42N-T-\frac{9\sqrt[3]{3}}{4\sqrt[3]{4}}S-111\right)$	$\frac{9\sqrt[3]{3}}{16}\frac{1}{\sqrt[3]{S}}N^3T$
jacobi-1d	$\max(2+n,\frac{1}{4}\frac{1}{S}(T-1)(N-3)-T-S+7)$	$\frac{1}{4}\frac{1}{S}NT$
jacobi-2d	$\max\left((N-2)(N+6),\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}(N-3)^2(T-1)-\frac{4\sqrt{2}}{3\sqrt{3}}S-(T-7)(2N-7)+14\right)$	$\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}N^2T$
lu	$\max\left(N^2,\frac{2}{3}\frac{1}{\sqrt{S}}(N-2)(N^2-4N+6)-2(N^2-10N+18)-8\sqrt{2}S\right)$	$\frac{2}{3}\frac{1}{\sqrt{S}}N^3$
ludcmp	$\max\left(N^2+N,\frac{1}{3}\frac{1}{\sqrt{S}}(2N-3)(N-1)(N-2)\sqrt{2}\frac{1}{S}(N-1)(N-2)-(2N^2-15N+19)-16\sqrt{2}S\right)$	$\frac{2}{3}\frac{1}{\sqrt{S}}N^3$
mvt	$N^2+4N+\max(0,\frac{1}{6}\frac{1}{S}N(N-1)-2S-4N+4)$	N^2
nussinov	$\frac{1}{2}N^2+\frac{5}{2}N-1+\max\left(0,\frac{1}{6}\frac{1}{\sqrt{S}}(N-3)(N-4)(N-5)+\frac{1}{4}\frac{1}{S}\sqrt{2}(3N^2-19N+6) -(N^2-13N+22)-8\sqrt{2}S\right)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$
seidel-2d	$\max\left(N^2,\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}(N-3)^2(T-1)-(2N-7)(T-5)-\frac{4\sqrt{2}}{3\sqrt{3}}S+12\right)$	$\frac{2}{3\sqrt{3}}\frac{1}{\sqrt{S}}N^2T$
symm	$\max\left(\frac{1}{2}M(M+1)+2MN+2,2\frac{1}{\sqrt{S}}(M-1)(M-2)N-\frac{1}{2}((4N+M)(M-5)) +5(M-2)-8\sqrt{2}S\right)$	$2\frac{1}{\sqrt{S}}M^2N$
syr2k	$\max\left(2+2MN+\frac{1}{2}N(N+1),\frac{1}{\sqrt{S}}(M-1)(N+1)N+M+4N-4\sqrt{2}S\right)$	$\frac{1}{\sqrt{S}}MN^2$
syrk	$\max\left(MN+\frac{1}{2}(N+1)N+2,\frac{1}{2}\frac{1}{\sqrt{S}}(M-1)(N+1)N-(M-4)(N-1)-2\sqrt{2}S+4\right)$	$\frac{1}{2}\frac{1}{\sqrt{S}}MN^2$
trisolv	$\frac{1}{2}N(N+1)+N+\max(0,\frac{1}{8}\frac{1}{S}(N-1)(N-2)-2N-S+5)$	$\frac{1}{2}N^2$
trmm	$\max\left(\frac{1}{2}M(M-1)+MN+1,\frac{1}{\sqrt{S}}(M-2+\frac{\sqrt{2}}{\sqrt{S}})(M-1)N-(M-4)(N-2)-8\sqrt{2}S+5\right)$	$\frac{1}{\sqrt{S}}M^2N$

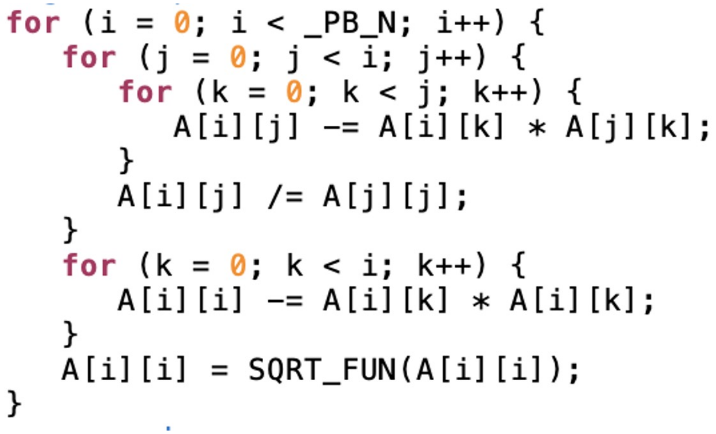
Table 2. Complete Lower Bound Formulae for PolyBench obtained with the current version of IOLB



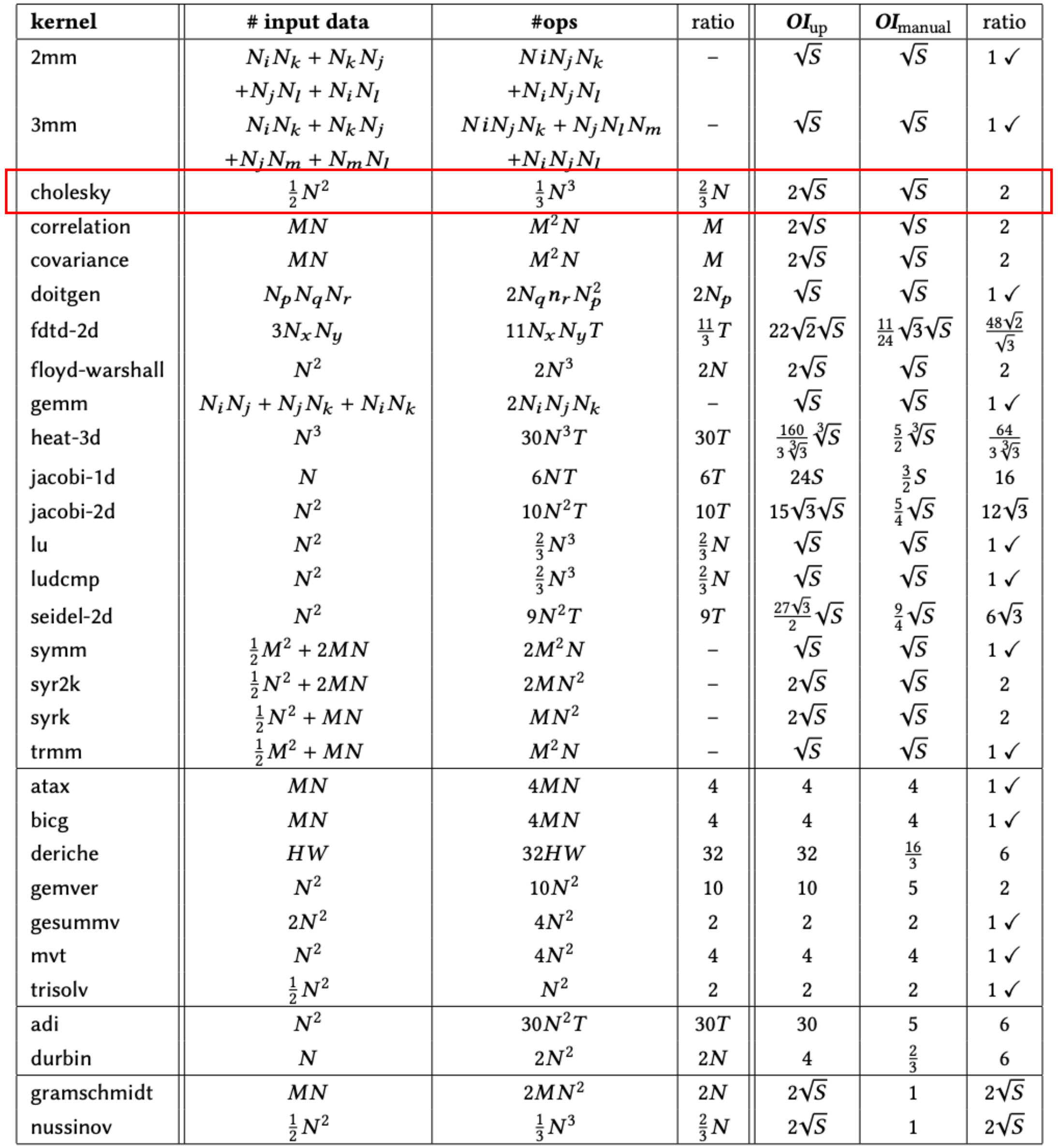
cholesky	$\max\left(\frac{1}{2}N(N+1),\frac{1}{6}\frac{1}{\sqrt{S}}(N-1)(N-2)(N-3)+\frac{1}{2\sqrt{2}}\frac{1}{S}(N-1)(N-2) -(N-2)(N-7)-4\sqrt{2}S\right)$	$\frac{1}{6}\frac{1}{\sqrt{S}}N^3$

Off by a factor of 2 from the lowest known upper bound

cholesky

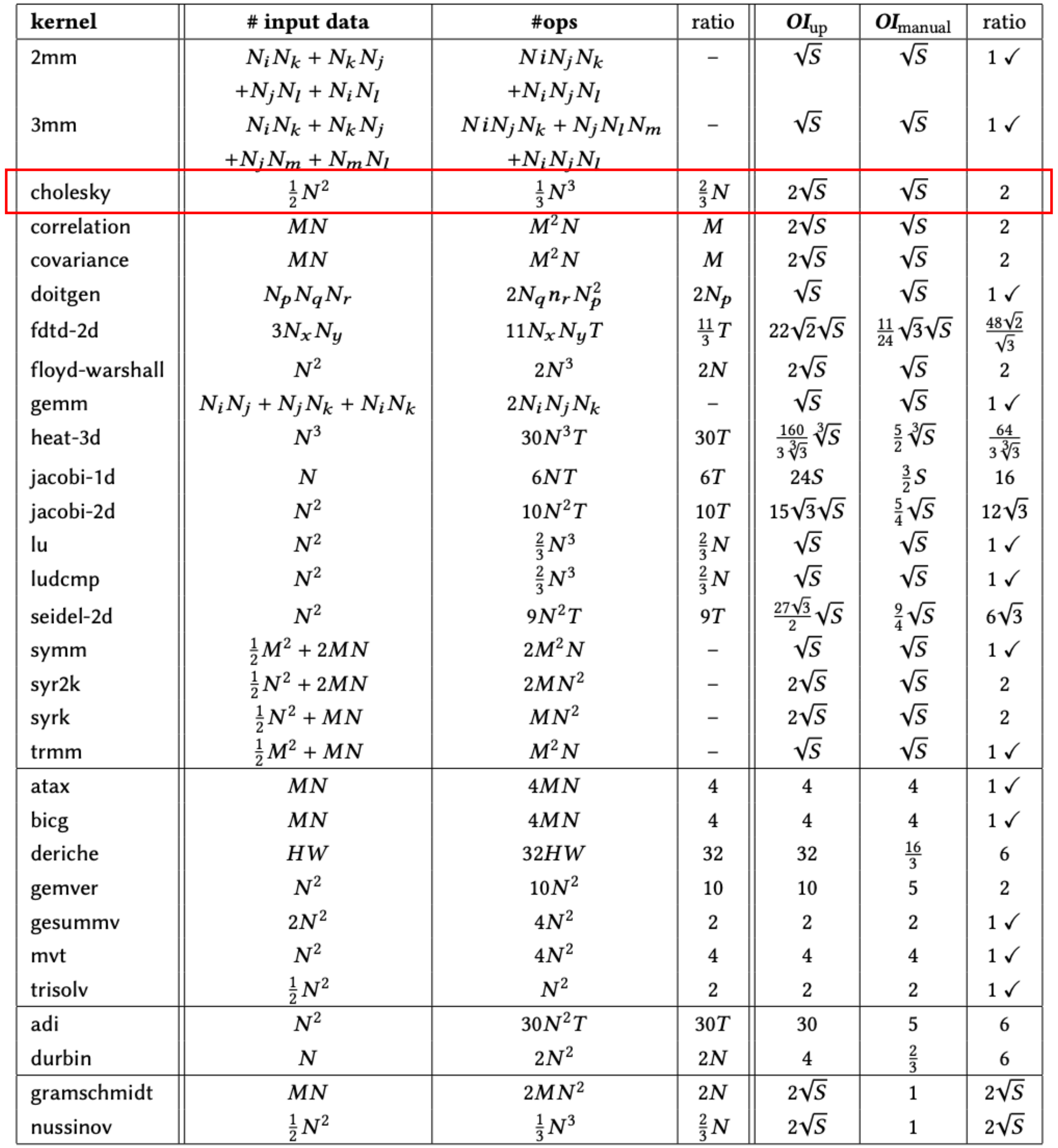


```
for (i = 0; i < _PB_N; i++) {
   for (j = 0; j < i; j++) {
      for (k = 0; k < j; k++) {
         A[i][j] -= A[i][k] * A[j][k];
      }
      A[i][j] /= A[j][j];
   }
   for (k = 0; k < i; k++) {
      A[i][i] -= A[i][k] * A[i][k];
   }
   A[i][i] = SQRT_FUN(A[i][i]);
}
```



kernel	**# input data**	**#ops**	ratio	OI_{up}	OI_{manual}	ratio
2mm	$N_iN_k + N_kN_j$ $+N_jN_l + N_iN_l$	NiN_jN_k $+N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
3mm	$N_iN_k + N_kN_j$ $+N_jN_m + N_mN_l$	$NiN_jN_k + N_jN_lN_m$ $+N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
cholesky	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	$\sqrt{S}$	2
correlation	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
covariance	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
doitgen	$N_pN_qN_r$	$2N_qn_rN_p^2$	$2N_p$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
fdtd-2d	$3N_xN_y$	$11N_xN_yT$	$\frac{11}{3}T$	$22\sqrt{2}\sqrt{S}$	$\frac{11}{24}\sqrt{3}\sqrt{S}$	$\frac{48\sqrt{2}}{\sqrt{3}}$
floyd-warshall	N^2	$2N^3$	$2N$	$2\sqrt{S}$	$\sqrt{S}$	2
gemm	$N_iN_j + N_jN_k + N_iN_k$	$2N_iN_jN_k$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
heat-3d	N^3	$30N^3T$	$30T$	$\frac{160}{3\sqrt[3]{3}}\sqrt[3]{S}$	$\frac{5}{2}\sqrt[3]{S}$	$\frac{64}{3\sqrt[3]{3}}$
jacobi-1d	N	$6NT$	$6T$	$24S$	$\frac{3}{2}S$	16
jacobi-2d	N^2	$10N^2T$	$10T$	$15\sqrt{3}\sqrt{S}$	$\frac{5}{4}\sqrt{S}$	$12\sqrt{3}$
lu	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
ludcmp	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
seidel-2d	N^2	$9N^2T$	$9T$	$\frac{27\sqrt{3}}{2}\sqrt{S}$	$\frac{9}{4}\sqrt{S}$	$6\sqrt{3}$
symm	$\frac{1}{2}M^2 + 2MN$	$2M^2N$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
syr2k	$\frac{1}{2}N^2 + 2MN$	$2MN^2$	–	$2\sqrt{S}$	$\sqrt{S}$	2
syrk	$\frac{1}{2}N^2 + MN$	MN^2	–	$2\sqrt{S}$	$\sqrt{S}$	2
trmm	$\frac{1}{2}M^2 + MN$	M^2N	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
atax	MN	$4MN$	4	4	4	1 ✓
bicg	MN	$4MN$	4	4	4	1 ✓
deriche	HW	$32HW$	32	32	$\frac{16}{3}$	6
gemver	N^2	$10N^2$	10	10	5	2
gesummv	$2N^2$	$4N^2$	2	2	2	1 ✓
mvt	N^2	$4N^2$	4	4	4	1 ✓
trisolv	$\frac{1}{2}N^2$	N^2	2	2	2	1 ✓
adi	N^2	$30N^2T$	$30T$	30	5	6
durbin	N	$2N^2$	$2N$	4	$\frac{2}{3}$	6
gramschmidt	MN	$2MN^2$	$2N$	$2\sqrt{S}$	1	$2\sqrt{S}$
nussinov	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	1	$2\sqrt{S}$

A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)



kernel	# input data	#ops	ratio	OI_{up}	OI_{manual}	ratio
2mm	$N_iN_k + N_kN_j$ $+N_jN_l + N_iN_l$	NiN_jN_k $+N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
3mm	$N_iN_k + N_kN_j$ $+N_jN_m + N_mN_l$	$NiN_jN_k + N_jN_lN_m$ $+N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
cholesky	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	$\sqrt{S}$	2
correlation	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
covariance	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
doitgen	$N_pN_qN_r$	$2N_qn_rN_p^2$	$2N_p$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
fdtd-2d	$3N_xN_y$	$11N_xN_yT$	$\frac{11}{3}T$	$22\sqrt{2}\sqrt{S}$	$\frac{11}{24}\sqrt{3}\sqrt{S}$	$\frac{48\sqrt{2}}{\sqrt{3}}$
floyd-warshall	N^2	$2N^3$	$2N$	$2\sqrt{S}$	$\sqrt{S}$	2
gemm	$N_iN_j + N_jN_k + N_iN_k$	$2N_iN_jN_k$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
heat-3d	N^3	$30N^3T$	$30T$	$\frac{160}{3\sqrt[3]{3}}\sqrt[3]{S}$	$\frac{5}{2}\sqrt[3]{S}$	$\frac{64}{3\sqrt[3]{3}}$
jacobi-1d	N	$6NT$	$6T$	$24S$	$\frac{3}{2}S$	16
jacobi-2d	N^2	$10N^2T$	$10T$	$15\sqrt{3}\sqrt{S}$	$\frac{5}{4}\sqrt{S}$	$12\sqrt{3}$
lu	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
ludcmp	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
seidel-2d	N^2	$9N^2T$	$9T$	$\frac{27\sqrt{3}}{2}\sqrt{S}$	$\frac{9}{4}\sqrt{S}$	$6\sqrt{3}$
symm	$\frac{1}{2}M^2 + 2MN$	$2M^2N$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
syr2k	$\frac{1}{2}N^2 + 2MN$	$2MN^2$	–	$2\sqrt{S}$	$\sqrt{S}$	2
syrk	$\frac{1}{2}N^2 + MN$	MN^2	–	$2\sqrt{S}$	$\sqrt{S}$	2
trmm	$\frac{1}{2}M^2 + MN$	M^2N	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
atax	MN	$4MN$	4	4	4	1 ✓
bicg	MN	$4MN$	4	4	4	1 ✓
deriche	HW	$32HW$	32	32	$\frac{16}{3}$	6
gemver	N^2	$10N^2$	10	10	5	2
gesummv	$2N^2$	$4N^2$	2	2	2	1 ✓
mvt	N^2	$4N^2$	4	4	4	1 ✓
trisolv	$\frac{1}{2}N^2$	N^2	2	2	2	1 ✓
adi	N^2	$30N^2T$	$30T$	30	5	6
durbin	N	$2N^2$	$2N$	4	$\frac{2}{3}$	6
gramschmidt	MN	$2MN^2$	$2N$	$2\sqrt{S}$	1	$2\sqrt{S}$
nussinov	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	1	$2\sqrt{S}$

A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)

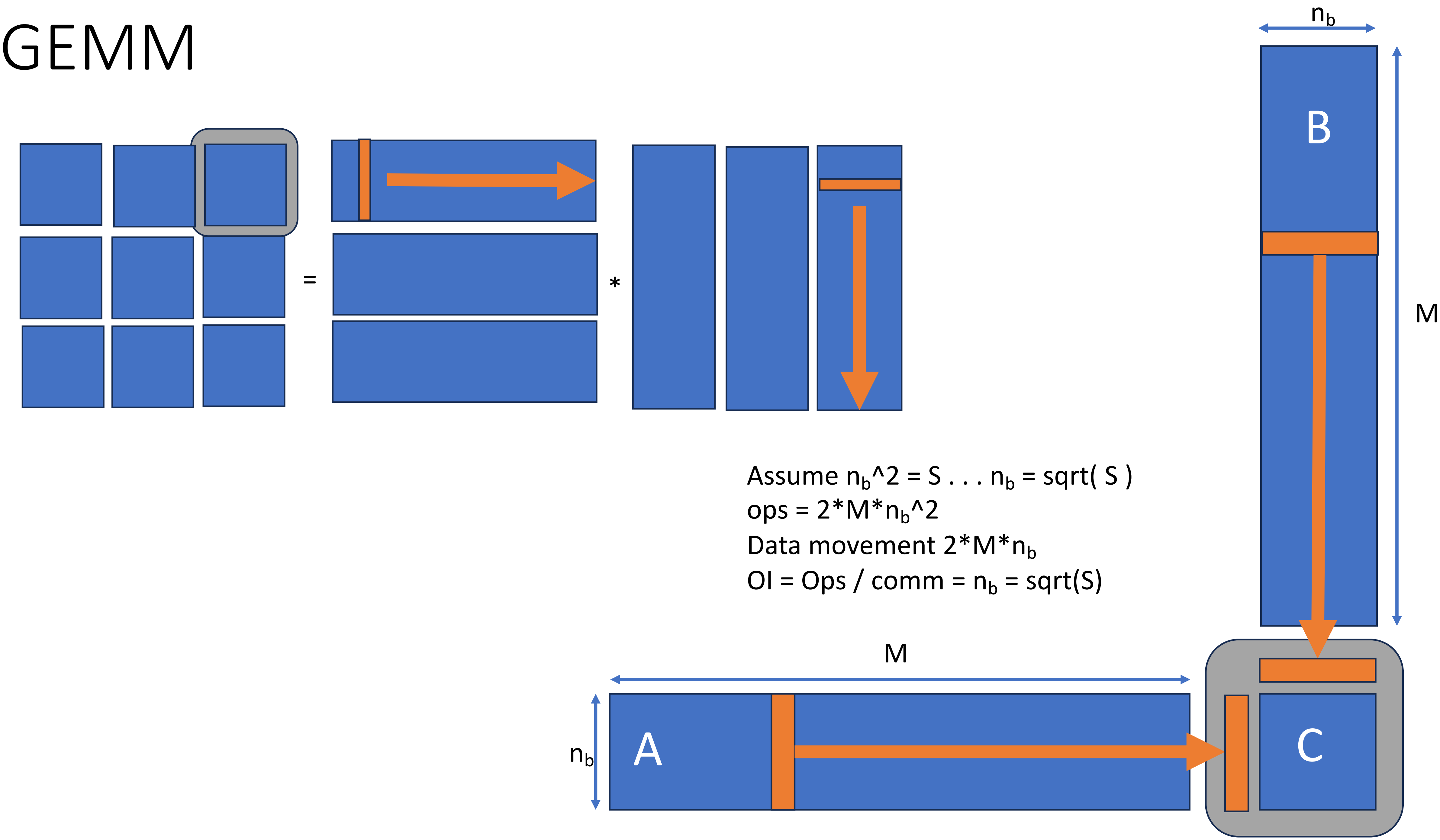
(1) Olivier Beaumont, Lionel Eyraud-Dubois, Julien Langou, and Mathieu Vérité. I/O-optimal algorithms for symmetric linear algebra kernels. In the 34th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '22), Philadelphia, PA, USA, July 11–14, 2022. DOI information: 10.1145/3490148.3538587.

(2) Olivier Beaumont, Philippe Duchon, Lionel Eyraud-Dubois, Julien Langou, and Mathieu Vérité.
Symmetric Block-Cyclic Distribution: Fewer Communications leads to Faster Dense Cholesky Factorization.
In ACM/IEEE SC 2022 Conference (SC'22), Dallas, TX, USA, November 13-18, 2022. DOI information: 10.1109/SC41404.2022.00034. Nominee for Best Paper Award.

(3) Emmanuel Agullo, Alfredo Buttari, Olivier Coulaud, Lionel Eyraud-Dubois, Mathieu Faverge, Alain Franc, Abdou Guermouche, Antoine Jego, Romain Peressoni, and Florent Pruvost. On the Arithmetic Intensity of Distributed-Memory Dense Matrix Multiplication Involving a Symmetric Input Matrix (SYMM). In 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS’23), St. Petersburg, FL, USA, 15-19 May 2023. DOI information: 10.1109/IPDPS54959.2023.00044

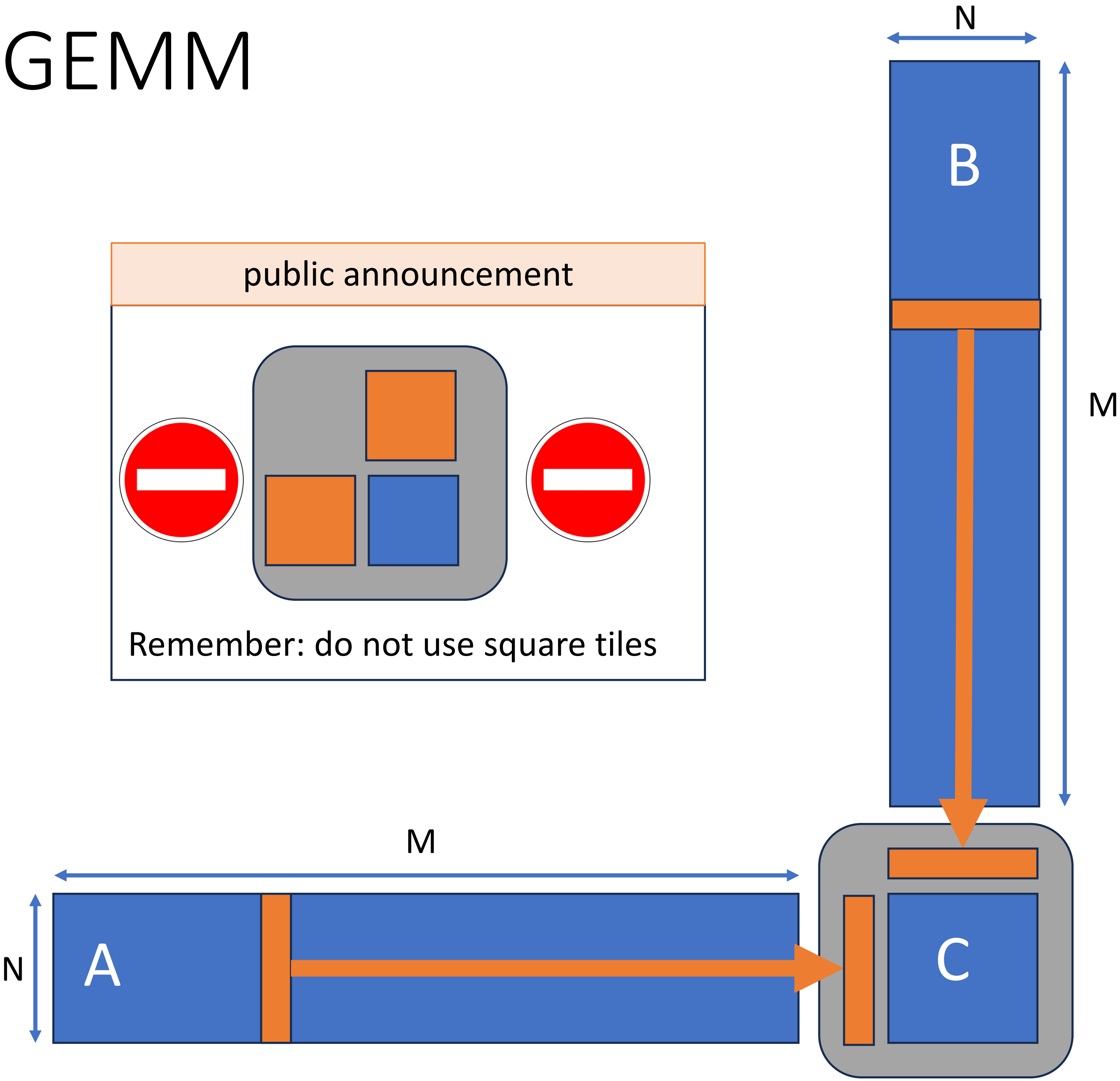
Operational intensity of LU is at sqrt(2) * sqrt(S)

GEMM



Assume n_b^2 = S . . . n_b = sqrt(S)
ops = 2*M*n_b^2
Data movement 2*M*n_b
OI = Ops / comm = n_b = sqrt(S)

GEMM



public announcement

Remember: do not use square tiles

SYRK

Assume (N^2)/2 = S . . . N = sqrt(S) * sqrt(2)

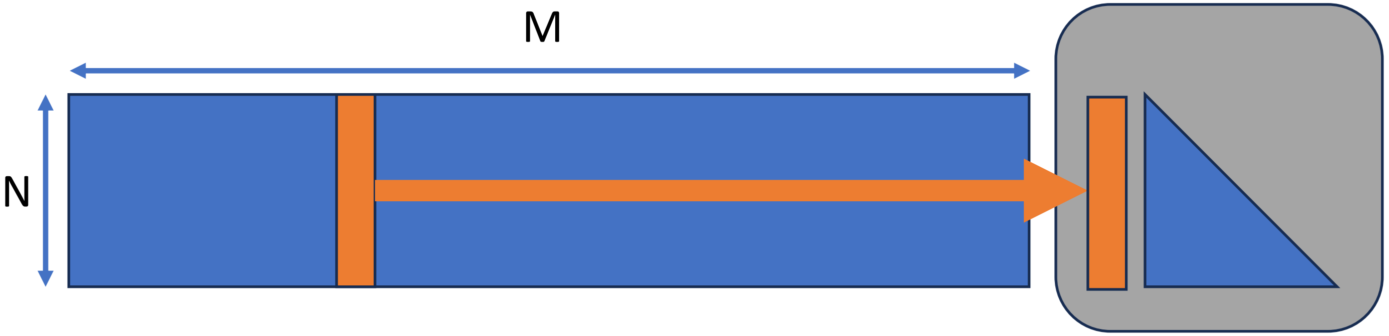
ops = M*N^2

Data movement M*N

OI = Ops / comm = N = sqrt(S)*sqrt(2)

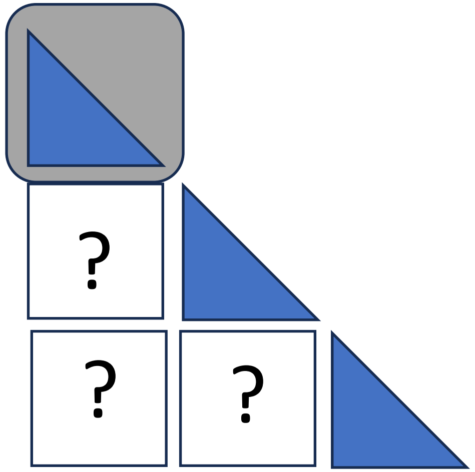
<u>Observation:</u>

SYRK has an OI sqrt(2) better than MM when a “triangle fit in cache”

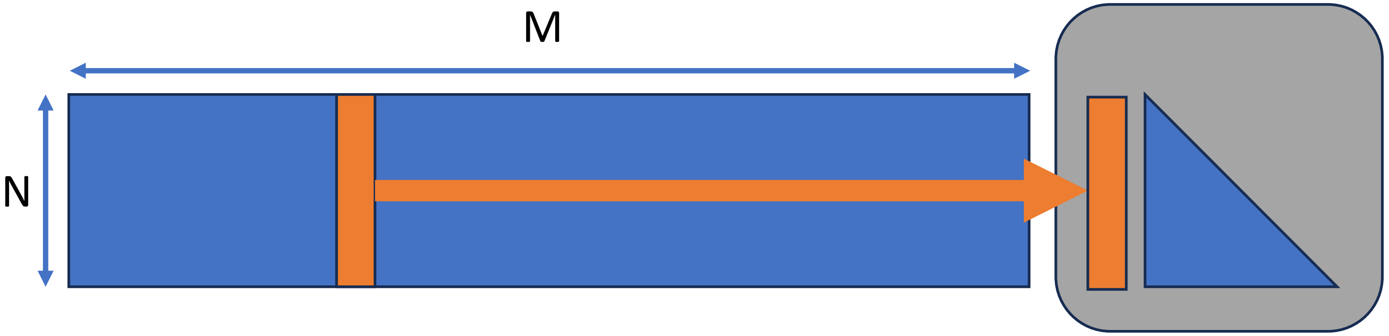


SYRK

Assume (N^2)/2 = S . . . N = sqrt(S) * sqrt(2)
ops = M*N^2
Data movement M*N
OI = Ops / comm = N = sqrt(S)*sqrt(2)



SYRK has an OI sqrt(2) better than MM when a “triangle fit in cache”



- Building an optimal GEMM is easy.
- But how to build an optimal SYRK?

SYRK

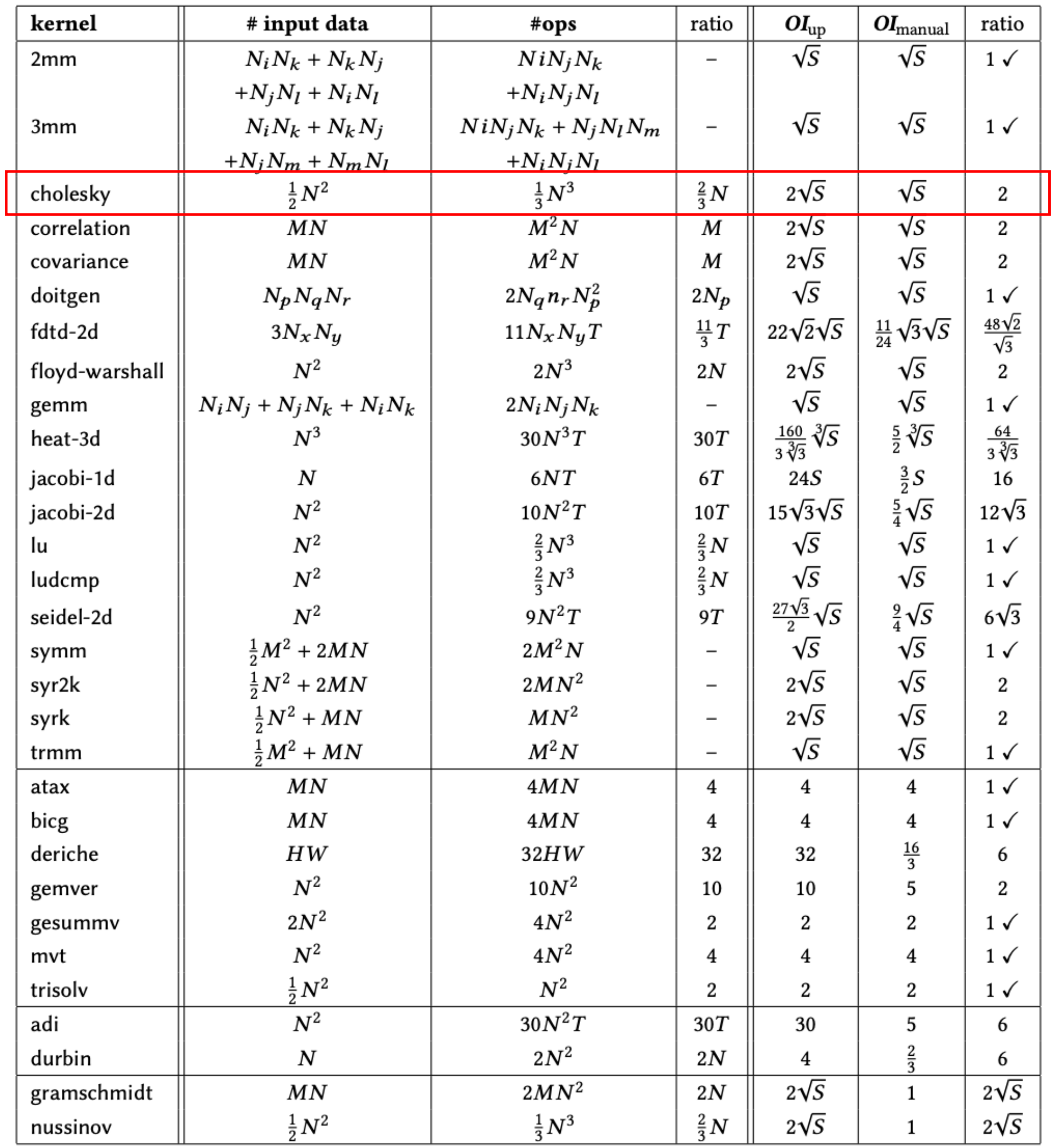
- IOLB says OI is less than 2 sqrt(S)

- Béreux’s algorithm has an OI of sqrt(S)

- This begs the question what is the OI of SYRK?

SYRK

- IOLB says OI is less than 2 sqrt(S)
- Béreux’s algorithm has an OI of sqrt(S)
- The OI upper bound from IOLB of 2 sqrt(S) is too optimistic, we can find a lower OI upper bound: sqrt(2) * sqrt(S).
- And we can improve Béreux’s algorithm. Instead of an OI of sqrt(S), we can improve to sqrt(2) * sqrt(S).



kernel	# input data	#ops	ratio	OI_{up}	OI_{manual}	ratio
2mm	$N_iN_k + N_kN_j$ $+N_jN_l + N_iN_l$	NiN_jN_k $+N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
3mm	$N_iN_k + N_kN_j$ $+N_jN_m + N_mN_l$	$NiN_jN_k + N_jN_lN_m$ $+N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
cholesky	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	$\sqrt{S}$	2
correlation	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
covariance	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
doitgen	$N_pN_qN_r$	$2N_qn_rN_p^2$	$2N_p$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
fdtd-2d	$3N_xN_y$	$11N_xN_yT$	$\frac{11}{3}T$	$22\sqrt{2}\sqrt{S}$	$\frac{11}{24}\sqrt{3}\sqrt{S}$	$\frac{48\sqrt{2}}{\sqrt{3}}$
floyd-warshall	N^2	$2N^3$	$2N$	$2\sqrt{S}$	$\sqrt{S}$	2
gemm	$N_iN_j + N_jN_k + N_iN_k$	$2N_iN_jN_k$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
heat-3d	N^3	$30N^3T$	$30T$	$\frac{160}{3\sqrt[3]{3}}\sqrt[3]{S}$	$\frac{5}{2}\sqrt[3]{S}$	$\frac{64}{3\sqrt[3]{3}}$
jacobi-1d	N	$6NT$	$6T$	$24S$	$\frac{3}{2}S$	16
jacobi-2d	N^2	$10N^2T$	$10T$	$15\sqrt{3}\sqrt{S}$	$\frac{5}{4}\sqrt{S}$	$12\sqrt{3}$
lu	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
ludcmp	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
seidel-2d	N^2	$9N^2T$	$9T$	$\frac{27\sqrt{3}}{2}\sqrt{S}$	$\frac{9}{4}\sqrt{S}$	$6\sqrt{3}$
symm	$\frac{1}{2}M^2 + 2MN$	$2M^2N$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
syr2k	$\frac{1}{2}N^2 + 2MN$	$2MN^2$	–	$2\sqrt{S}$	$\sqrt{S}$	2
syrk	$\frac{1}{2}N^2 + MN$	MN^2	–	$2\sqrt{S}$	$\sqrt{S}$	2
trmm	$\frac{1}{2}M^2 + MN$	M^2N	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
atax	MN	$4MN$	4	4	4	1 ✓
bicg	MN	$4MN$	4	4	4	1 ✓
deriche	HW	$32HW$	32	32	$\frac{16}{3}$	6
gemver	N^2	$10N^2$	10	10	5	2
gesummv	$2N^2$	$4N^2$	2	2	2	1 ✓
mvt	N^2	$4N^2$	4	4	4	1 ✓
trisolv	$\frac{1}{2}N^2$	N^2	2	2	2	1 ✓
adi	N^2	$30N^2T$	$30T$	30	5	6
durbin	N	$2N^2$	$2N$	4	$\frac{2}{3}$	6
gramschmidt	MN	$2MN^2$	$2N$	$2\sqrt{S}$	1	$2\sqrt{S}$
nussinov	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	1	$2\sqrt{S}$

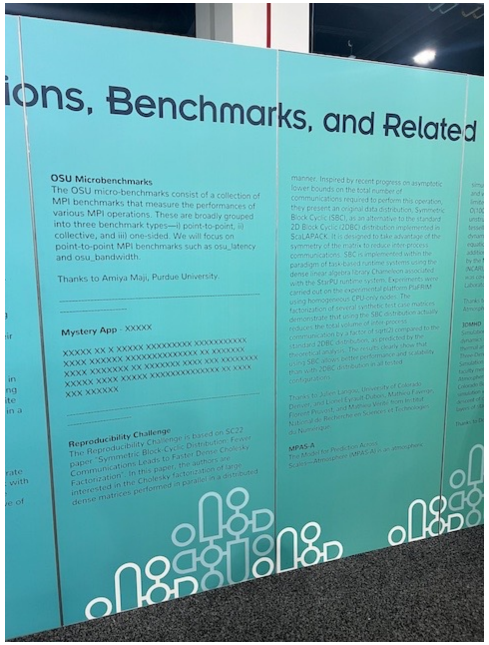
A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)

(1) Olivier Beaumont, Lionel Eyraud-Dubois, Julien Langou, and Mathieu Vérité. I/O-optimal algorithms for symmetric linear algebra kernels. In the 34th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '22), Philadelphia, PA, USA, July 11–14, 2022. DOI information: 10.1145/3490148.3538587.

(2) Olivier Beaumont, Philippe Duchon, Lionel Eyraud-Dubois, Julien Langou, and Mathieu Vérité.
Symmetric Block-Cyclic Distribution: Fewer Communications leads to Faster Dense Cholesky Factorization.
In ACM/IEEE SC 2022 Conference (SC'22), Dallas, TX, USA, November 13-18, 2022. DOI information: 10.1109/SC41404.2022.00034. Nominee for Best Paper Award.

(3) Emmanuel Agullo, Alfredo Buttari, Olivier Coulaud, Lionel Eyraud-Dubois, Mathieu Faverge, Alain Franc, Abdou Guermouche, Antoine Jego, Romain Peressoni, and Florent Pruvost. On the Arithmetic Intensity of Distributed-Memory Dense Matrix Multiplication Involving a Symmetric Input Matrix (SYMM). In 2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS’23), St. Petersburg, FL, USA, 15-19 May 2023. DOI information: 10.1109/IPDPS54959.2023.00044

SIGHPC Reproducibility Award
SCC @ SC’23



ions, Benchmarks, and Related

OSU Microbenchmarks
The OSU micro-benchmarks consist of a collection of MPI benchmarks that measure the performances of various MPI operations. These are broadly grouped into three benchmark types—i) point-to-point, ii) collective, and iii) one-sided. We will focus on point-to-point MPI benchmarks such as osu_latency and osu_bandwidth.

Thanks to Amiya Maji, Purdue University.

Mystery App - XXXXX

XXXXX XX X XXXXX XXXXXXXXX XXXXXXXXXX XXXXX XX X XXXXX XXXXXXXXXXXXX XX XXXXXXX XXXX XXXXXX XXXXXXXXXXXXXX XXXX XXX XXXXXXXX XXXX XXXXXXX XX XXXXXXX XXXX XXXXXXXXXXXXX XX XXXX XXXXX XXXX XXXXX XXXXXXXXXXXXXX XX XXXX XXX XXXXXX

Reproducibility Challenge
The Reproducibility Challenge is based on SC22 paper "Symmetric Block-Cyclic Distribution: Fewer Communications Leads to Faster Dense Cholesky Factorization". In this paper, the authors are interested in the Cholesky factorization of large dense matrices performed in parallel in a distributed manner. Inspired by recent progress on asymptotic lower bounds on the total number of communications required to perform this operation, they present an original data distribution, Symmetric Block Cyclic (SBC), as an alternative to the standard 2D Block Cyclic (2DBC) distribution implemented in ScaLAPACK. It is designed to take advantage of the symmetry of the matrix to reduce inter-process communications. SBC is implemented within the paradigm of task-based runtime systems using the dense linear algebra library Chameleon associated with the StarPU runtime system. Experiments were carried out on the experimental platform PlaFRIM using homogeneous CPU-only nodes. The factorization of several synthetic test case matrices demonstrate that using the SBC distribution actually reduces the total volume of inter-process communication by a factor of sqrt(2) compared to the standard 2DBC distribution, as predicted by the theoretical analysis. The results clearly show that using SBC allows better performance and scalability than with 2DBC distribution in all tested configurations.

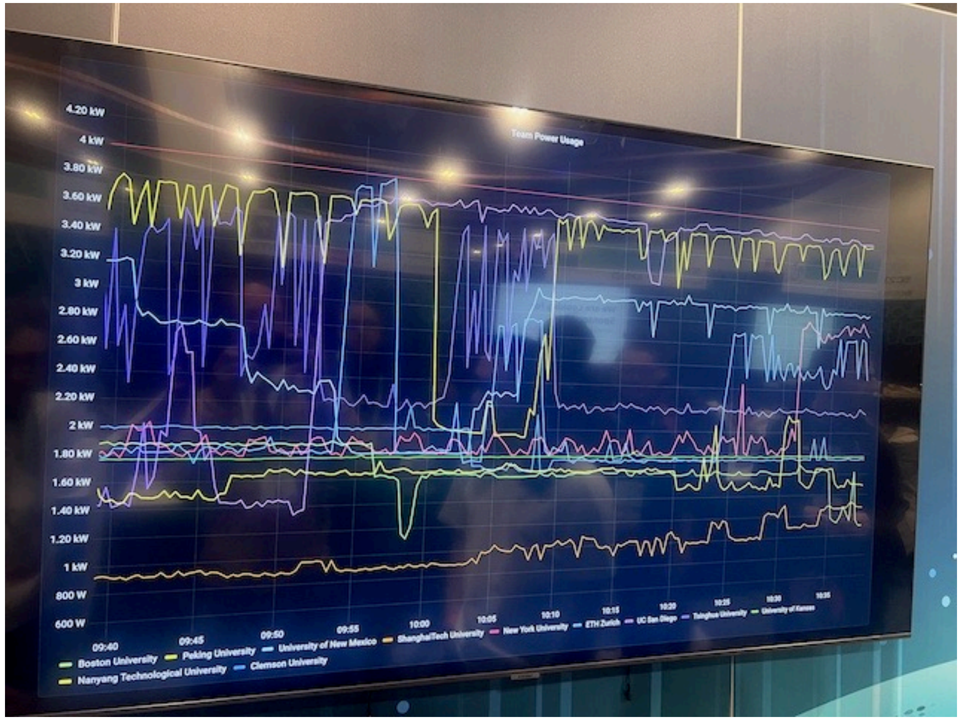
Thanks to Julien Langou, University of Colorado Denver, and Lionel Eyraud-Dubois, Mathieu Faverge, Florent Pruvost, and Mathieu Vérité from Institut National de Recherche en Sciences et Technologies du Numérique.

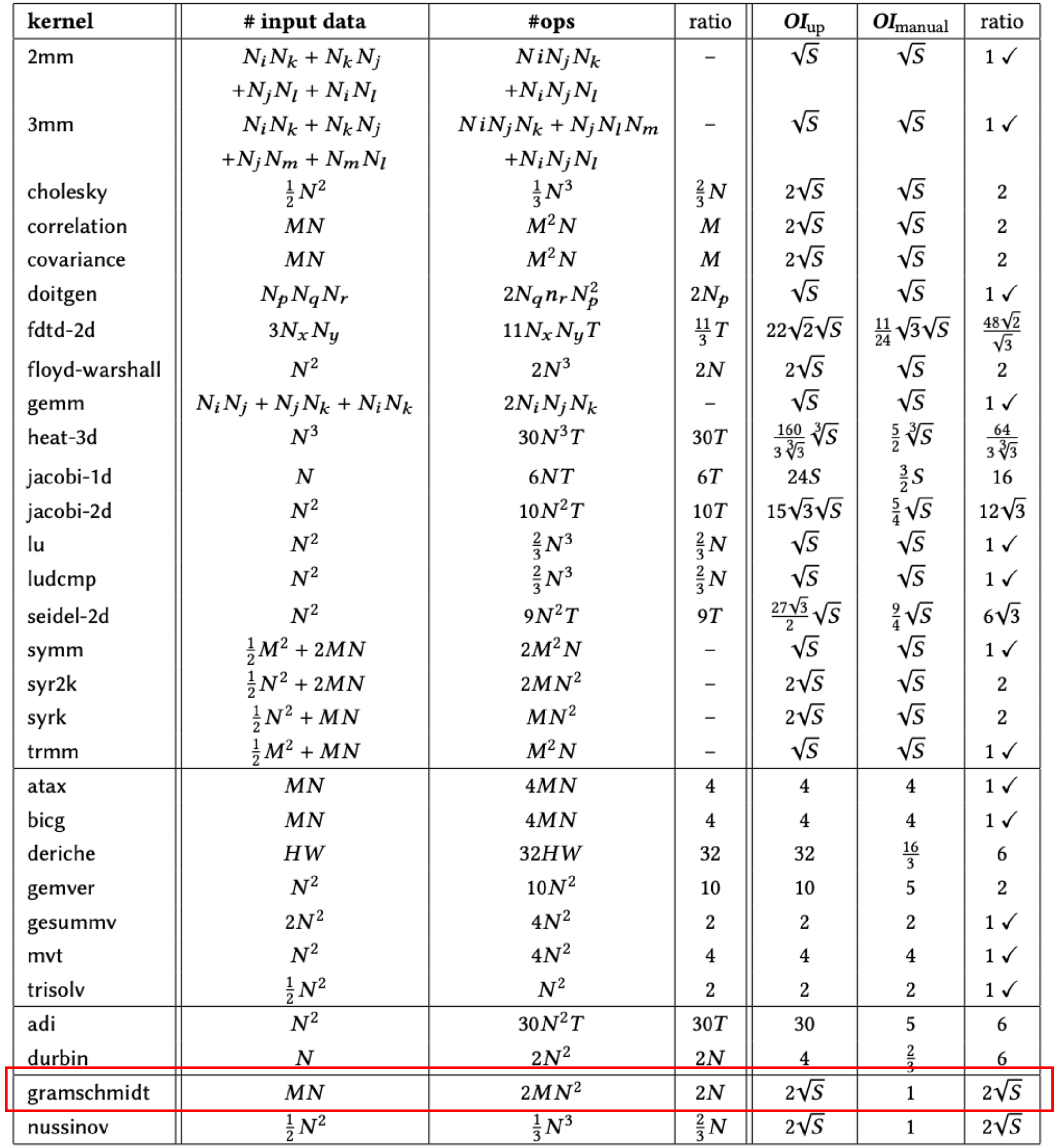
MPAS-A
The Model for Prediction Across Scales—Atmosphere (MPAS-A) is an atmospheric

Thanks to Mathieu Faverge and Florent Pruvost

Thanks to Thomas Herault, George Bosilca, and Weslley Pereira

Olivier Beaumont, Philippe Duchon, Lionel Eyraud-Dubois, Julien Langou, and Mathieu Vérité. Symmetric Block-Cyclic Distribution: Fewer Communications leads to Faster Dense Cholesky Factorization. In ACM/IEEE SC 2022 Conference (SC'22), Dallas, TX, USA, November 13-18, 2022. DOI information: 10.1109/SC41404.2022.00034. Nominee for Best Paper Award.





kernel	**# input data**	**#ops**	ratio	$\boldsymbol{OI}_{\text{up}}$	$\boldsymbol{OI}_{\text{manual}}$	ratio
2mm	$N_iN_k + N_kN_j$ $+N_jN_l + N_iN_l$	NiN_jN_k $+N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
3mm	$N_iN_k + N_kN_j$ $+N_jN_m + N_mN_l$	$NiN_jN_k + N_jN_lN_m$ $+N_iN_jN_l$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
cholesky	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	$\sqrt{S}$	2
correlation	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
covariance	MN	M^2N	M	$2\sqrt{S}$	$\sqrt{S}$	2
doitgen	$N_pN_qN_r$	$2N_qn_rN_p^2$	$2N_p$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
fdtd-2d	$3N_xN_y$	$11N_xN_yT$	$\frac{11}{3}T$	$22\sqrt{2}\sqrt{S}$	$\frac{11}{24}\sqrt{3}\sqrt{S}$	$\frac{48\sqrt{2}}{\sqrt{3}}$
floyd-warshall	N^2	$2N^3$	$2N$	$2\sqrt{S}$	$\sqrt{S}$	2
gemm	$N_iN_j + N_jN_k + N_iN_k$	$2N_iN_jN_k$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
heat-3d	N^3	$30N^3T$	$30T$	$\frac{160}{3\sqrt[3]{3}}\sqrt[3]{S}$	$\frac{5}{2}\sqrt[3]{S}$	$\frac{64}{3\sqrt[3]{3}}$
jacobi-1d	N	$6NT$	$6T$	$24S$	$\frac{3}{2}S$	16
jacobi-2d	N^2	$10N^2T$	$10T$	$15\sqrt{3}\sqrt{S}$	$\frac{5}{4}\sqrt{S}$	$12\sqrt{3}$
lu	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
ludcmp	N^2	$\frac{2}{3}N^3$	$\frac{2}{3}N$	$\sqrt{S}$	$\sqrt{S}$	1 ✓
seidel-2d	N^2	$9N^2T$	$9T$	$\frac{27\sqrt{3}}{2}\sqrt{S}$	$\frac{9}{4}\sqrt{S}$	$6\sqrt{3}$
symm	$\frac{1}{2}M^2 + 2MN$	$2M^2N$	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
syr2k	$\frac{1}{2}N^2 + 2MN$	$2MN^2$	–	$2\sqrt{S}$	$\sqrt{S}$	2
syrk	$\frac{1}{2}N^2 + MN$	MN^2	–	$2\sqrt{S}$	$\sqrt{S}$	2
trmm	$\frac{1}{2}M^2 + MN$	M^2N	–	$\sqrt{S}$	$\sqrt{S}$	1 ✓
atax	MN	$4MN$	4	4	4	1 ✓
bicg	MN	$4MN$	4	4	4	1 ✓
deriche	HW	$32HW$	32	32	$\frac{16}{3}$	6
gemver	N^2	$10N^2$	10	10	5	2
gesummv	$2N^2$	$4N^2$	2	2	2	1 ✓
mvt	N^2	$4N^2$	4	4	4	1 ✓
trisolv	$\frac{1}{2}N^2$	N^2	2	2	2	1 ✓
adi	N^2	$30N^2T$	$30T$	30	5	6
durbin	N	$2N^2$	$2N$	4	$\frac{2}{3}$	6
gramschmidt	MN	$2MN^2$	$2N$	$2\sqrt{S}$	1	$2\sqrt{S}$
nussinov	$\frac{1}{2}N^2$	$\frac{1}{3}N^3$	$\frac{2}{3}N$	$2\sqrt{S}$	1	$2\sqrt{S}$

A. Olivry, J. Langou, L-N Pouchet, P. Sadayappan, and F. Rastello. **Automated derivation of parametric data movement lower bounds for affine programs.** In the Proceedings of PLDI 2020: Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation, page 808–822, June 2020. (link)

Classical Gram-Schmidt (CGS)

```
Input: $a_1, a_2, \dots, a_{n-1}, a_n$
Output: $q_1, q_2, \dots, q_{n-1}, q_n$

for j=1:n,
    $w = ( I - Q_{1:j-1} Q_{1:j-1}^T ) a_j$
    $q_j = w / || w ||_2$
end
```

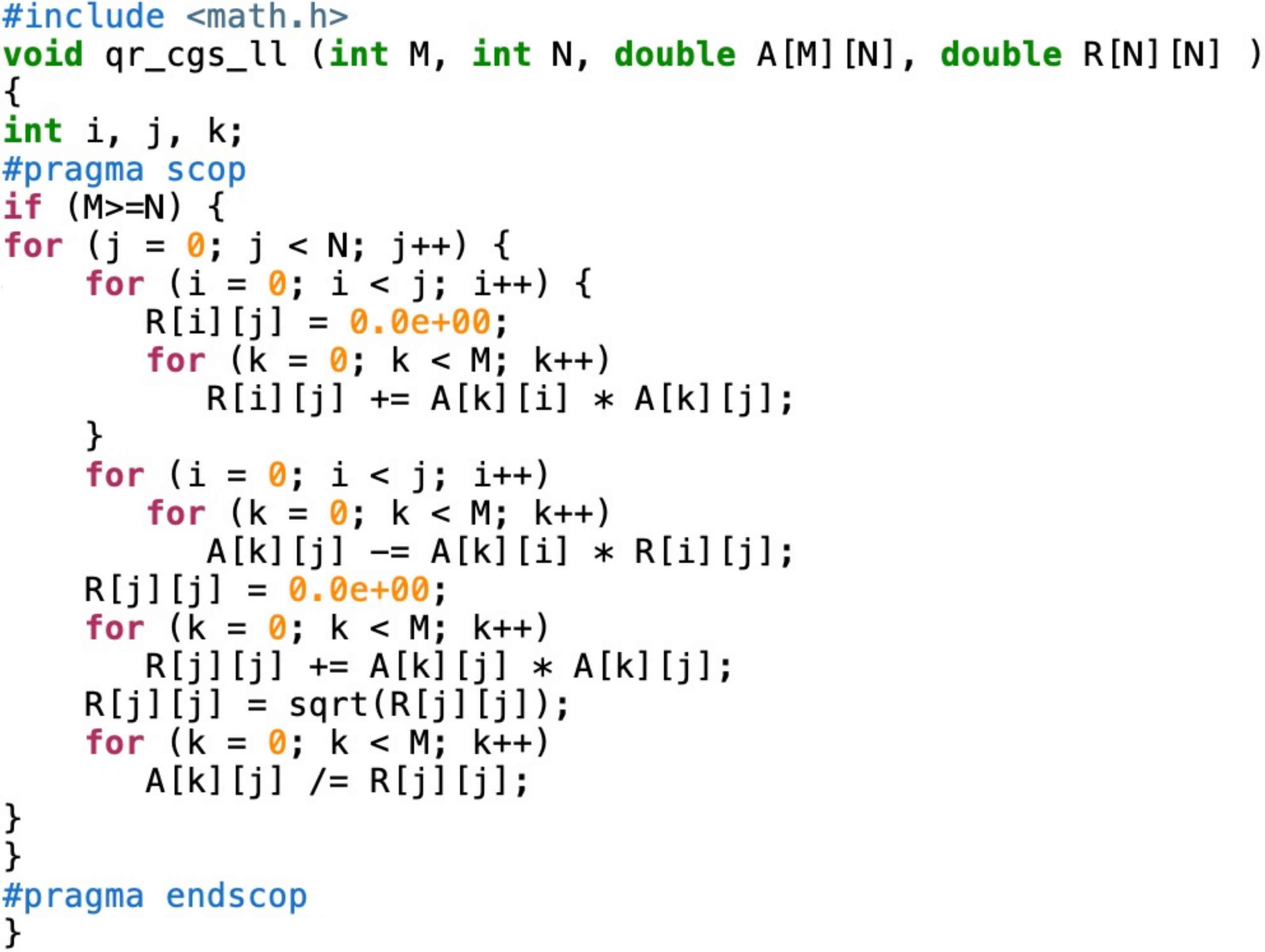
Modified Gram-Schmidt (MGS)

```
Input: $a_1, a_2, \dots, a_{n-1}, a_n$
Output: $q_1, q_2, \dots, q_{n-1}, q_n$

for j=1:n,
    $w = ( I - q_{j-1} q_{j-1}^T ) .. ( I - q_1 q_1^T ) a_j$
    $q_j = w / || w ||_2$
end
```

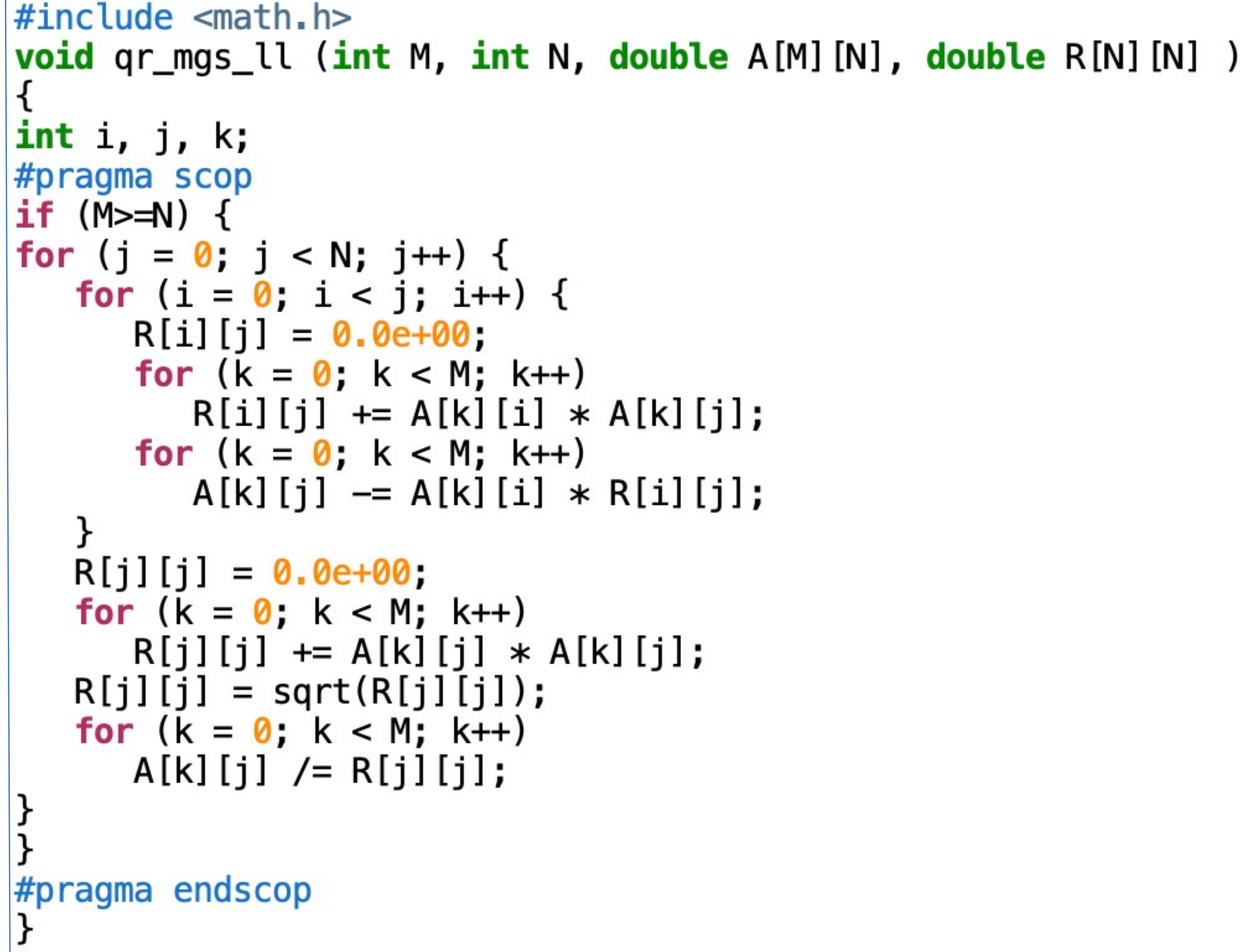
Rice, J. R., Experiments on Gram-Schmidt Orthogonalization, Math. Comp. 20 (1966), 325-328.

```
function [ Q, R ] = cgs( A )
   n = size(A,2);
   Q = zeros(size(A));
   R = zeros (n);
   for j=1:n,
      Q(:,j) = A(:,j) ;
      for i=1:j-1,
          R(i,j) =  Q(:,i)’ * A(:,j) ;
          Q(:,j)  = Q(:,j)  - Q(:,i) * R(i,j);
      end
      R(j,j) = norm( Q(:,j)  );
      Q(:,j) = Q(:,j)  / R(j,j);
   end
end
```

```
function [ Q, R ] = mgs( A )
   n = size(A,2);
   Q = zeros(size(A));
   R = zeros (n);
   for j=1:n,
      Q(:,j) = A(:,j) ;
      for i=1:j-1,
          R(i,j) =  Q(:,i)’ * Q(:,j) ;
          Q(:,j)  = Q(:,j)  - Q(:,i) * R(i,j);
      end
      R(j,j) = norm( Q(:,j)  );
      Q(:,j) = Q(:,j)  / R(j,j);
   end
end
```



```
#include <math.h>
void qr_cgs_ll (int M, int N, double A[M][N], double R[N][N] )
{
int i, j, k;
#pragma scop
if (M>=N) {
for (j = 0; j < N; j++) {
   for (i = 0; i < j; i++) {
     R[i][j] = 0.0e+00;
     for (k = 0; k < M; k++)
       R[i][j] += A[k][i] * A[k][j];
   }
   for (i = 0; i < j; i++)
     for (k = 0; k < M; k++)
       A[k][j] -= A[k][i] * R[i][j];
   R[j][j] = 0.0e+00;
   for (k = 0; k < M; k++)
     R[j][j] += A[k][j] * A[k][j];
   R[j][j] = sqrt(R[j][j]);
   for (k = 0; k < M; k++)
     A[k][j] /= R[j][j];
}
}
#pragma endscop
}
```



```
#include <math.h>
void qr_mgs_ll (int M, int N, double A[M][N], double R[N][N] )
{
int i, j, k;
#pragma scop
if (M>=N) {
for (j = 0; j < N; j++) {
  for (i = 0; i < j; i++) {
    R[i][j] = 0.0e+00;
    for (k = 0; k < M; k++)
      R[i][j] += A[k][i] * A[k][j];
    for (k = 0; k < M; k++)
      A[k][j] -= A[k][i] * R[i][j];
  }
  R[j][j] = 0.0e+00;
  for (k = 0; k < M; k++)
    R[j][j] += A[k][j] * A[k][j];
  R[j][j] = sqrt(R[j][j]);
  for (k = 0; k < M; k++)
    A[k][j] /= R[j][j];
}
}
#pragma endscop
}
```

CGS

MN input data (+ N^2 / 2 for R in output)

2 MN^2 operations

IOLB: operational intensity is at most $2\sqrt{S}$

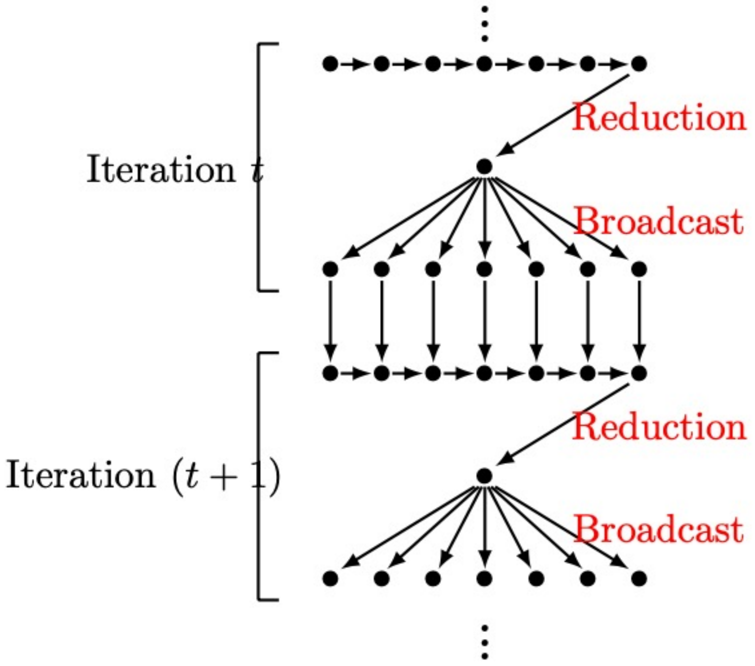
MGS

MN data (+ N^2 / 2 for R in output)

2 MN^2 operations

IOLB: operational intensity is at most $2\sqrt{S}$

Hourglass pattern



Standard “MM-way” of doing business

$$|E| \leq |\phi_{i,j}(E)|^{\frac{1}{2}}.|\phi_{i,k}(E)|^{\frac{1}{2}}.|\phi_{k,j}(E)|^{\frac{1}{2}} \leq \boxed{K^{3/2}}$$

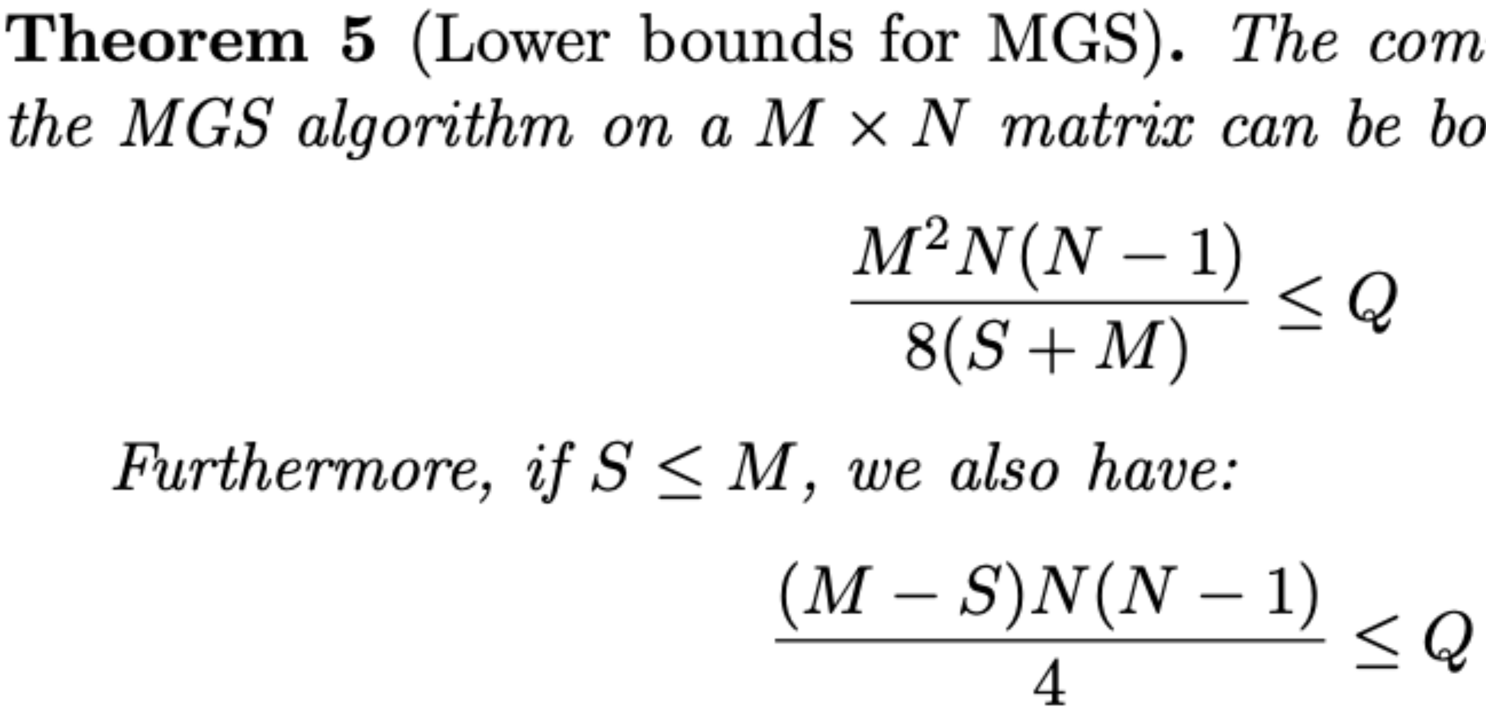
If Hourglass pattern is detected then

$$|I'| \leq |\phi_i(I')| \times |\phi_j(I')| \times |\phi_k(I')|.$$

$$|I'| \leq M \times \frac{K}{M} \times \frac{K}{M} = \boxed{\frac{K^2}{M}}$$

Modified Gram-Schmidt

Lionel Eyraud-Dubois, Guillaume Iooss, Julien Langou, and Fabrice Rastello. **Tightening I/O Lower Bounds through the Hourglass Dependency Pattern**. Accepted to the 36th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA ‘24), Nantes, France, June 17—21, 2024.



Theorem 5 (Lower bounds for MGS). *The communication volume Q for the MGS algorithm on a $M \times N$ matrix can be bounded as follows:*

$$\frac{M^2 N(N-1)}{8(S+M)} \leq Q$$

Furthermore, if $S \leq M$, we also have:

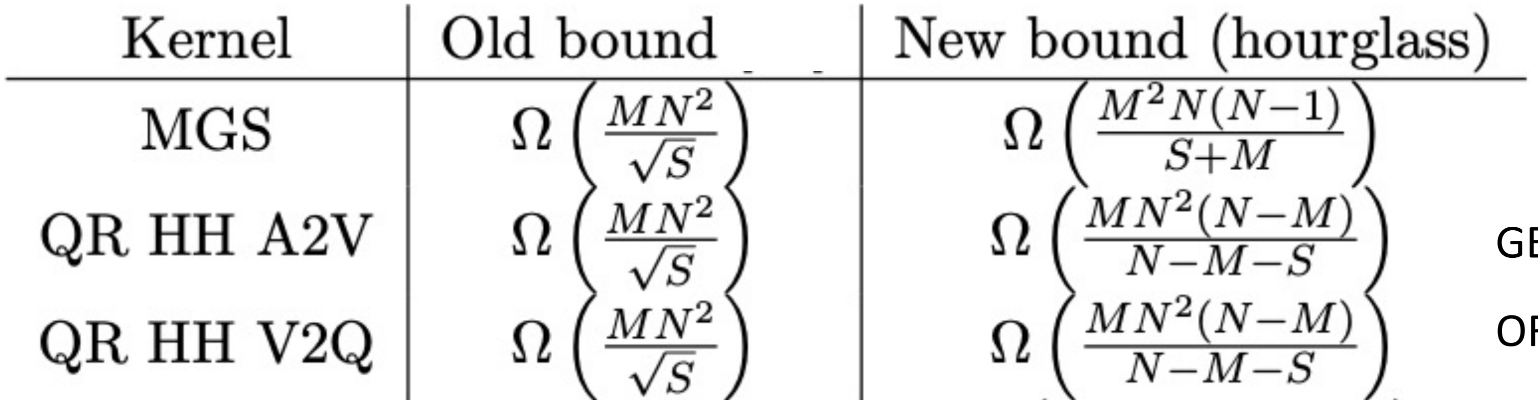
$$\frac{(M-S)N(N-1)}{4} \leq Q$$

Interpretation: (with approximations)

If M < N then IO is at least (1/8) * $M^2 N^2$ / S
or in another words (1/8) * (M / S) * M N^2

If M > N then IO is at least (1/4) * M N^2

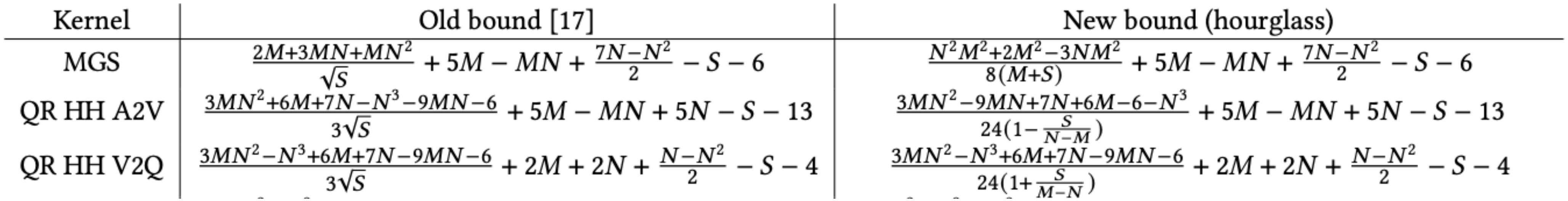
Interpretation:
Since M < sqrt(S), $M^2 N^2$ / S is a much better lower bound on IO than M N^2 / sqrt(S) – (larger is better)



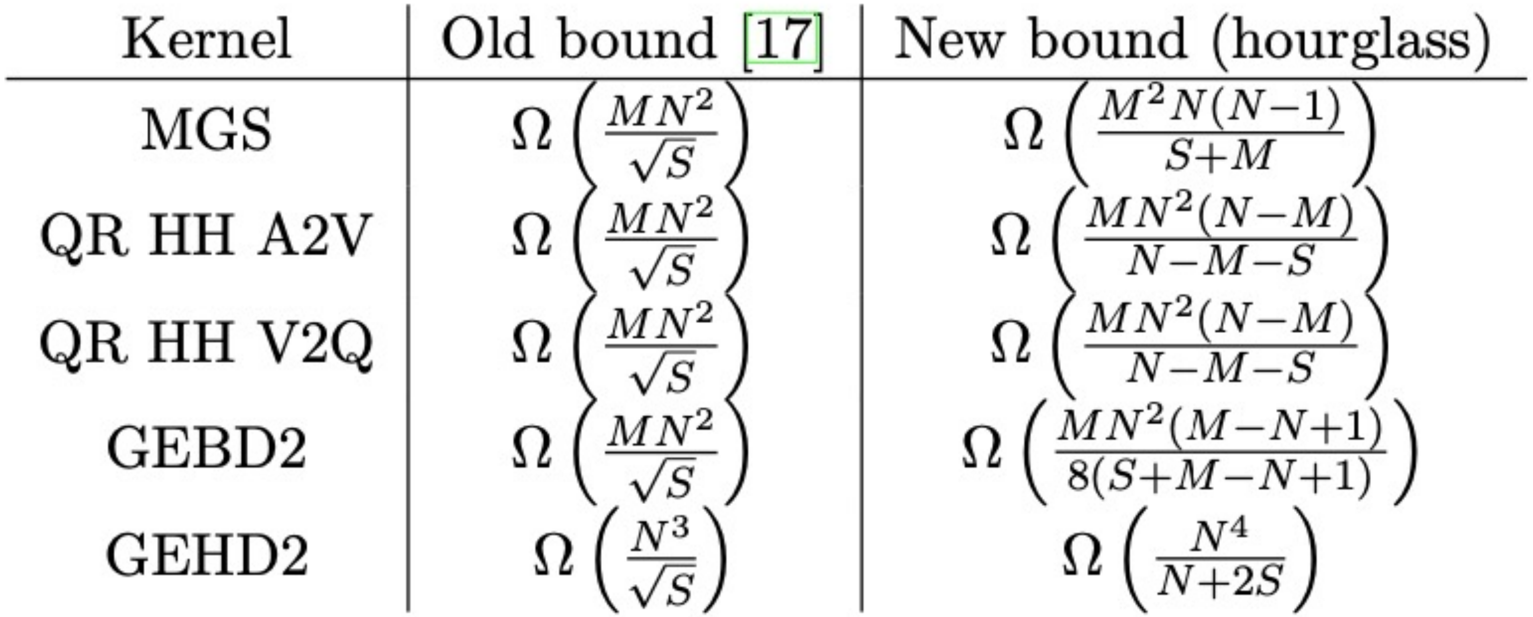
Kernel	Old bound	New bound (hourglass)
MGS	$\Omega\left(\frac{MN^2}{\sqrt{S}}\right)$	$\Omega\left(\frac{M^2N(N-1)}{S+M}\right)$
QR HH A2V	$\Omega\left(\frac{MN^2}{\sqrt{S}}\right)$	$\Omega\left(\frac{MN^2(N-M)}{N-M-S}\right)$
QR HH V2Q	$\Omega\left(\frac{MN^2}{\sqrt{S}}\right)$	$\Omega\left(\frac{MN^2(N-M)}{N-M-S}\right)$

GEQR2

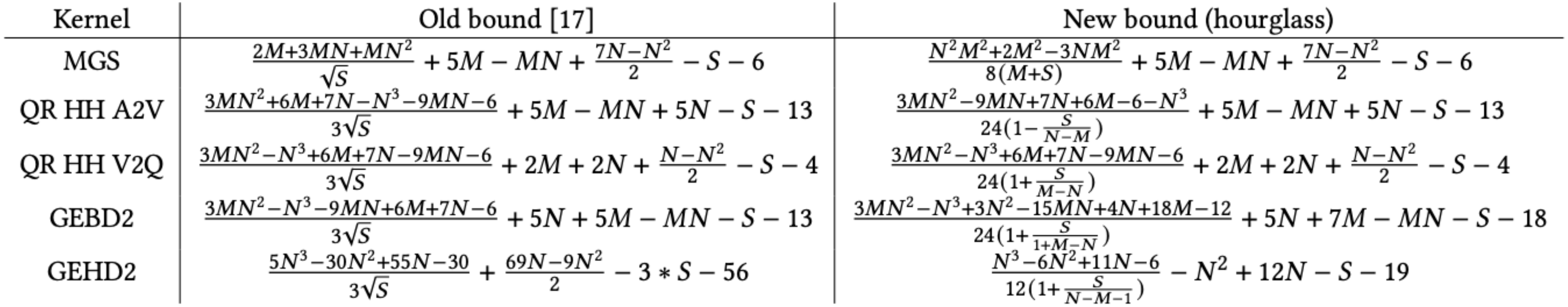
ORG2R



Kernel	Old bound [17]	New bound (hourglass)
MGS	$\frac{2M+3MN+MN^2}{\sqrt{S}} + 5M - MN + \frac{7N-N^2}{2} - S - 6$	$\frac{N^2M^2+2M^2-3NM^2}{8(M+S)} + 5M - MN + \frac{7N-N^2}{2} - S - 6$
QR HH A2V	$\frac{3MN^2+6M+7N-N^3-9MN-6}{3\sqrt{S}} + 5M - MN + 5N - S - 13$	$\frac{3MN^2-9MN+7N+6M-6-N^3}{24(1-\frac{S}{N-M})} + 5M - MN + 5N - S - 13$
QR HH V2Q	$\frac{3MN^2-N^3+6M+7N-9MN-6}{3\sqrt{S}} + 2M + 2N + \frac{N-N^2}{2} - S - 4$	$\frac{3MN^2-N^3+6M+7N-9MN-6}{24(1+\frac{S}{M-N})} + 2M + 2N + \frac{N-N^2}{2} - S - 4$



Kernel	Old bound [17]	New bound (hourglass)
MGS	$\Omega\left(\frac{MN^2}{\sqrt{S}}\right)$	$\Omega\left(\frac{M^2N(N-1)}{S+M}\right)$
QR HH A2V	$\Omega\left(\frac{MN^2}{\sqrt{S}}\right)$	$\Omega\left(\frac{MN^2(N-M)}{N-M-S}\right)$
QR HH V2Q	$\Omega\left(\frac{MN^2}{\sqrt{S}}\right)$	$\Omega\left(\frac{MN^2(N-M)}{N-M-S}\right)$
GEBD2	$\Omega\left(\frac{MN^2}{\sqrt{S}}\right)$	$\Omega\left(\frac{MN^2(M-N+1)}{8(S+M-N+1)}\right)$
GEHD2	$\Omega\left(\frac{N^3}{\sqrt{S}}\right)$	$\Omega\left(\frac{N^4}{N+2S}\right)$



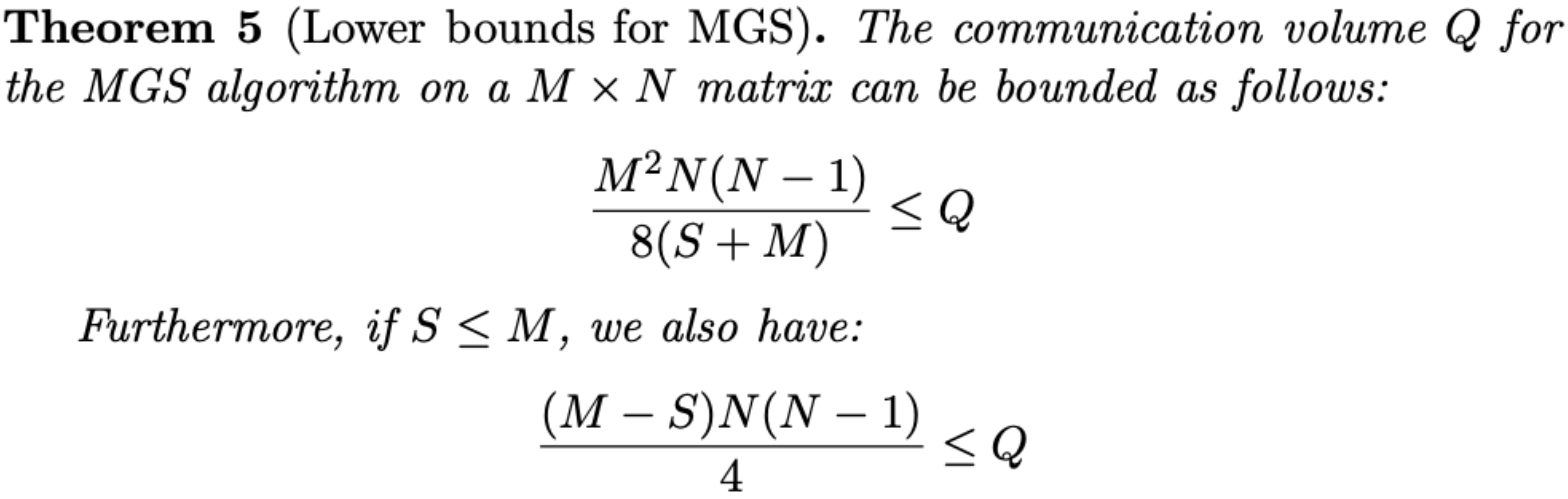
Kernel	Old bound [17]	New bound (hourglass)
MGS	$\frac{2M+3MN+MN^2}{\sqrt{S}} + 5M - MN + \frac{7N-N^2}{2} - S - 6$	$\frac{N^2M^2+2M^2-3NM^2}{8(M+S)} + 5M - MN + \frac{7N-N^2}{2} - S - 6$
QR HH A2V	$\frac{3MN^2+6M+7N-N^3-9MN-6}{3\sqrt{S}} + 5M - MN + 5N - S - 13$	$\frac{3MN^2-9MN+7N+6M-6-N^3}{24(1-\frac{S}{N-M})} + 5M - MN + 5N - S - 13$
QR HH V2Q	$\frac{3MN^2-N^3+6M+7N-9MN-6}{3\sqrt{S}} + 2M + 2N + \frac{N-N^2}{2} - S - 4$	$\frac{3MN^2-N^3+6M+7N-9MN-6}{24(1+\frac{S}{M-N})} + 2M + 2N + \frac{N-N^2}{2} - S - 4$
GEBD2	$\frac{3MN^2-N^3-9MN+6M+7N-6}{3\sqrt{S}} + 5N + 5M - MN - S - 13$	$\frac{3MN^2-N^3+3N^2-15MN+4N+18M-12}{24(1+\frac{S}{1+M-N})} + 5N + 7M - MN - S - 18$
GEHD2	$\frac{5N^3-30N^2+55N-30}{3\sqrt{S}} + \frac{69N-9N^2}{2} - 3 * S - 56$	$\frac{N^3-6N^2+11N-6}{12(1+\frac{S}{N-M-1})} - N^2 + 12N - S - 19$

```
./iolb-affine-indocker.sh examples/lin-alg/gehd2_splitted.c
```

```
-1+n^2+max(0,-107+5/24*(S-(2+m-n)^(-1)*S^2)^(-1)*S*n^3-11/12*((1+m-
n)^(-1)*S^2-S)^(-1)*S*n-8*m*S^(-1/2)*n^2-5/4*(S-(2+m-n)^(-1)*S^2)^(-
1)*S*n^2-23*S^(-1/2)-27*m*S^(-1/2)-1/12*((1+m-n)^(-1)*S^2-S)^(-
1)*S*n^3+55/24*(S-(2+m-n)^(-1)*S^2)^(-1)*S*n-2*m^2-5/4*(S-(2+m-
n)^(-1)*S^2)^(-1)*S+1/2*((1+m-n)^(-1)*S^2-S)^(-1)*S*n^2+31*m*S^(-
1/2)*n-9*m+235/6*S^(-1/2)*n+7*m^2*S^(-1/2)*n-13*m^2*S^(-
1/2)+5*m*n-25/2*n^2-41/2*S^(-1/2)*n^2-10*S+145/2*n+1/2*((1+m-n)^(-
1)*S^2-S)^(-1)*S+10/3*S^(-1/2)*n^3-2*m^3*S^(-1/2))
```

Modified Gram-Schmidt

Lionel Eyraud-Dubois, Guillaume Ioos, Julien Langou, and Fabrice Rastello. **Tightening I/O Lower Bounds through the Hourglass Dependency Pattern**. Accepted to the 36th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA ‘24), Nantes, France, June 17—21, 2024.



Theorem 5 (Lower bounds for MGS). *The communication volume Q for the MGS algorithm on a $M \times N$ matrix can be bounded as follows:*

$$\frac{M^2 N(N-1)}{8(S+M)} \leq Q$$

Furthermore, if $S \leq M$, we also have:

$$\frac{(M-S)N(N-1)}{4} \leq Q$$

Interpretation:

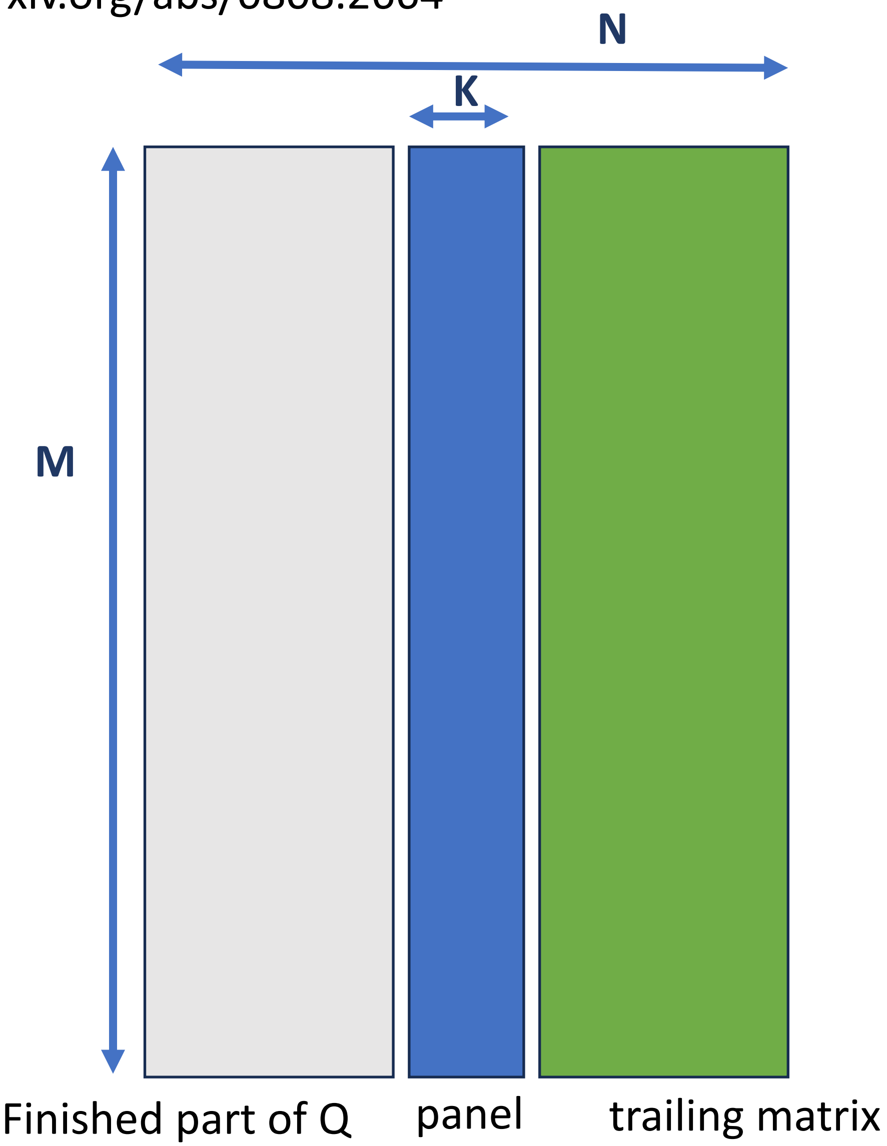
If M < N then IO is at least (1/8) * M^2 N^2 / S or (1/8) * (M / S) M N^2

If M > N then IO is at least (1/4) * M N^2

Interpretation:

Much better lower bound on IO than M N^2 / sqrt(S)

Communication-optimal parallel and sequential QR and LU factorizations
Demmel, Grigori, Hoemmen and Langou, 2008
https://arxiv.org/abs/0808.2664



We need at least a few columns to fit in cache!

Take K such that

$$M * K + M < S$$

panel fits in cache and stays in cache

At start of step, panel+trailing are updated and ready to go
Load panel
Do MGS on panel (panel factorization)
Update trailing matrix by loading one column at a time

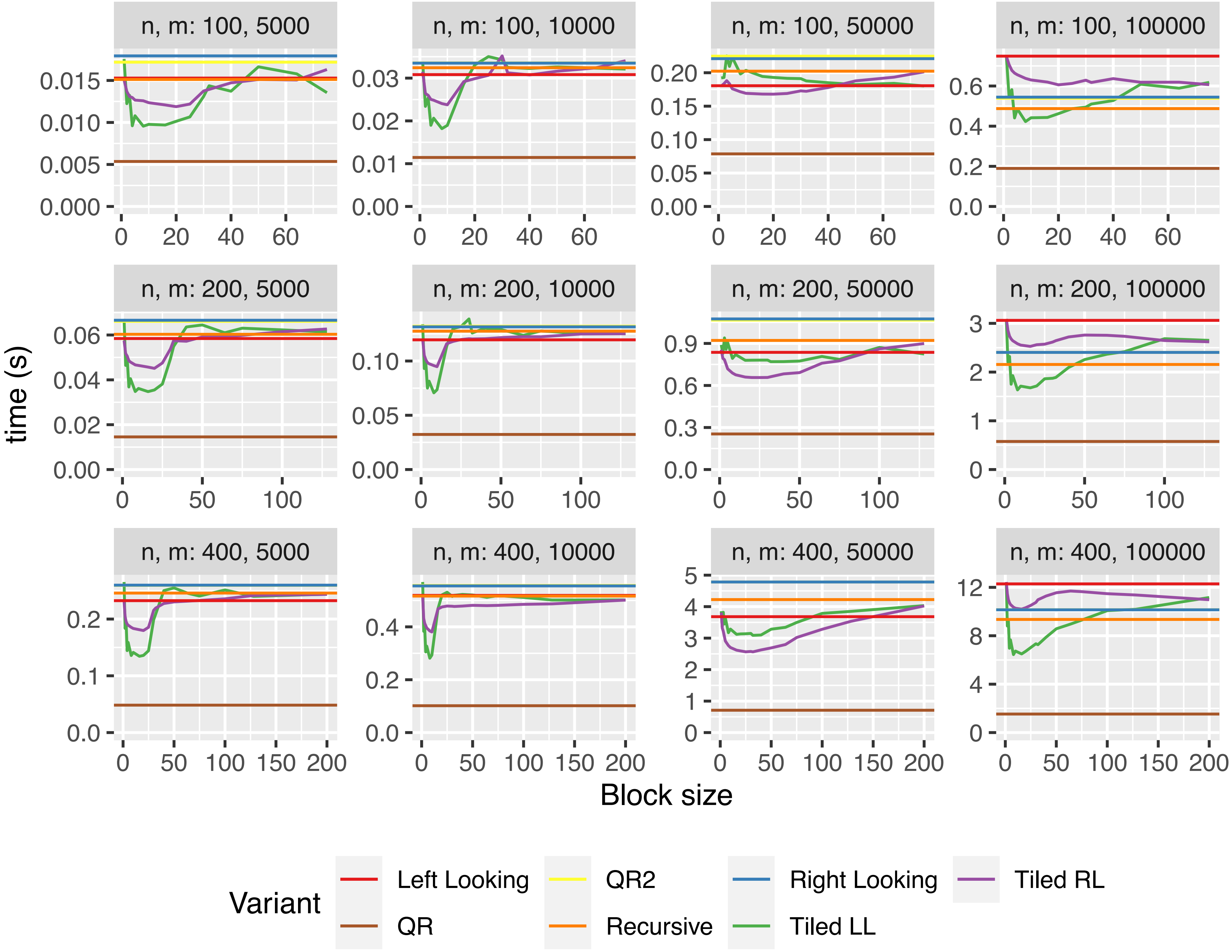
Analysis:
K is about M / S
Load about (less than) N columns of size M every N / K steps
So IO is about $M^2 N^2 / S$

Which is a factor of S/M better than $M N^2$
So a factor of K better than $M N^2$

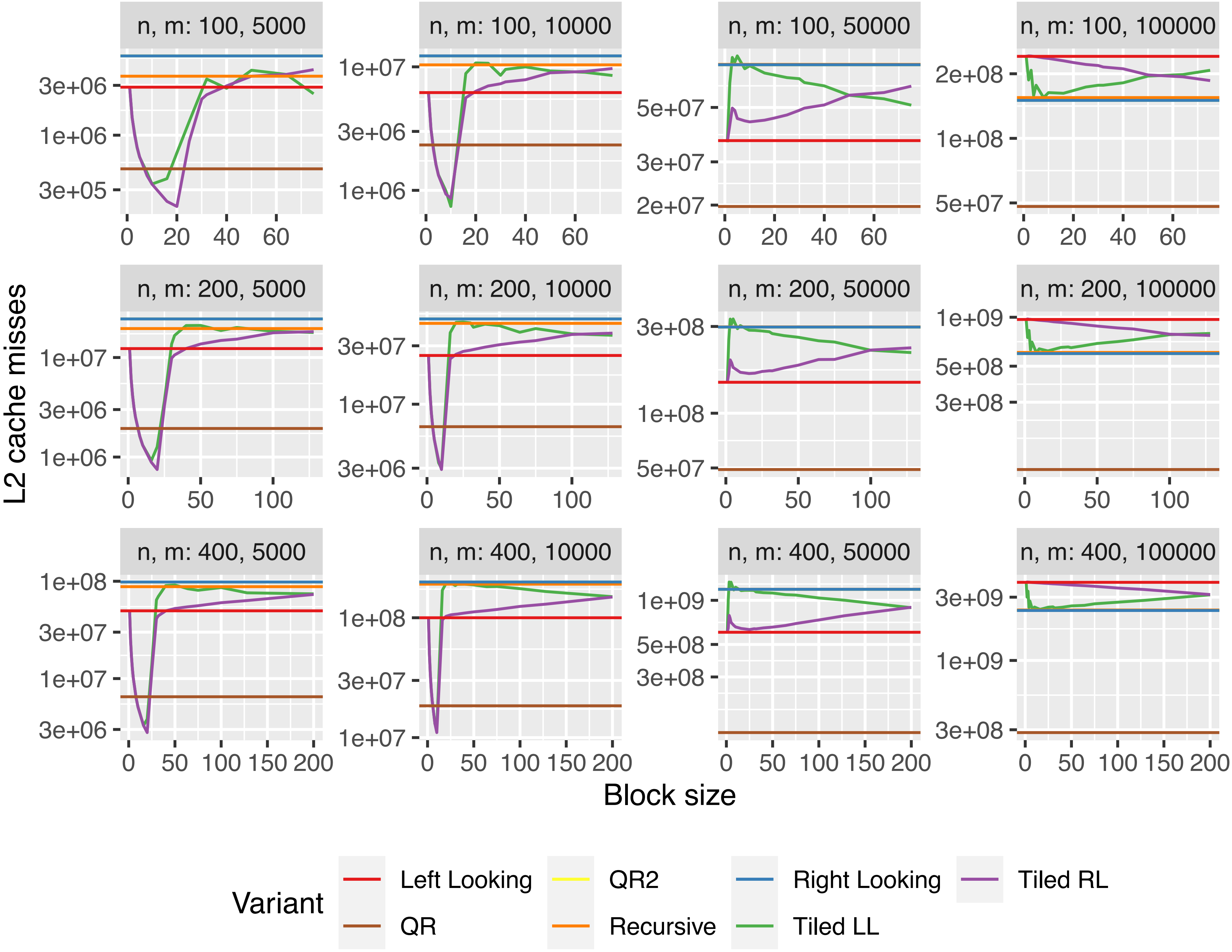
This is right looking, you can do a left looking variant too.

Modified Gram-Schmidt, GEQR2, ORGQR2

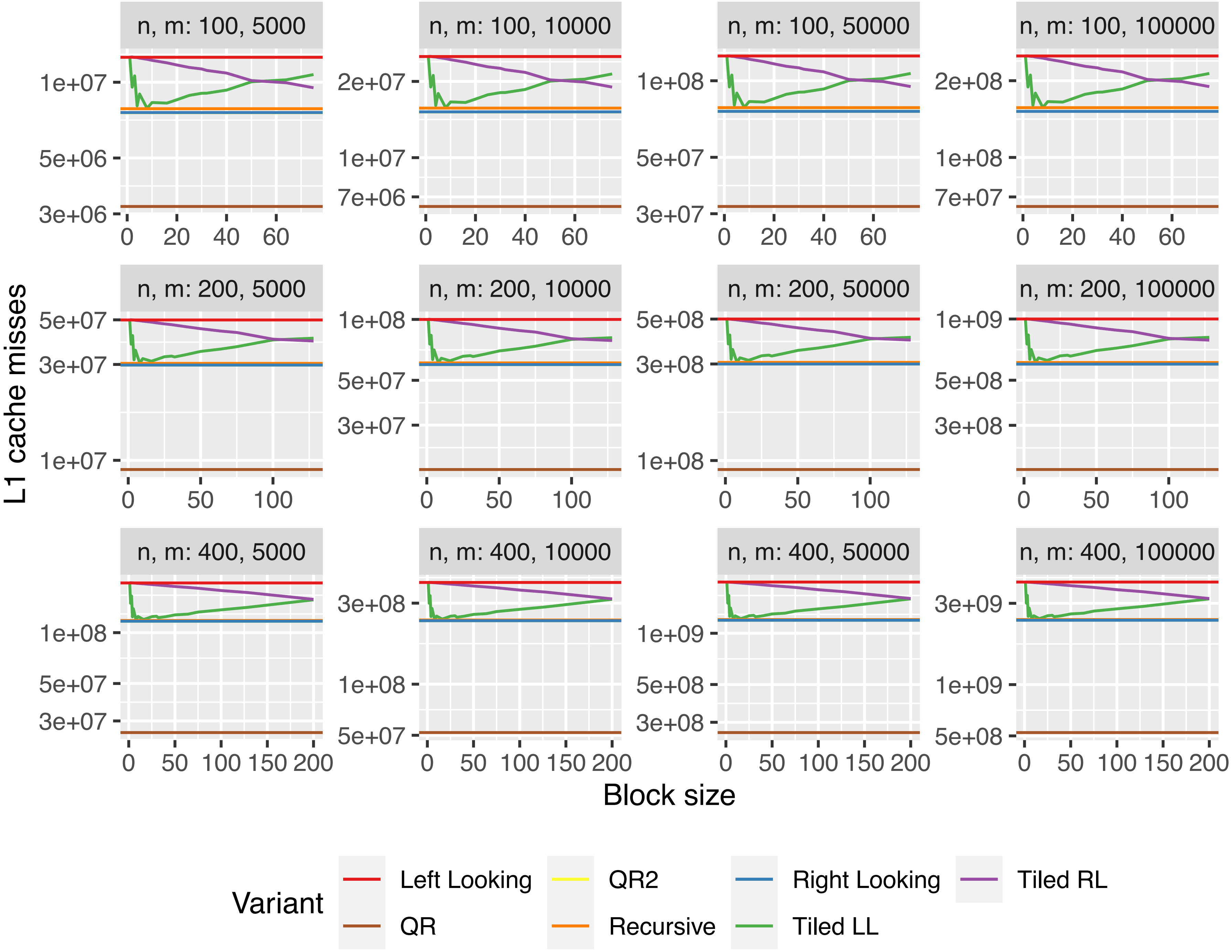
A2V (geqr2)
bora@plafrim
Intel CascadeLake



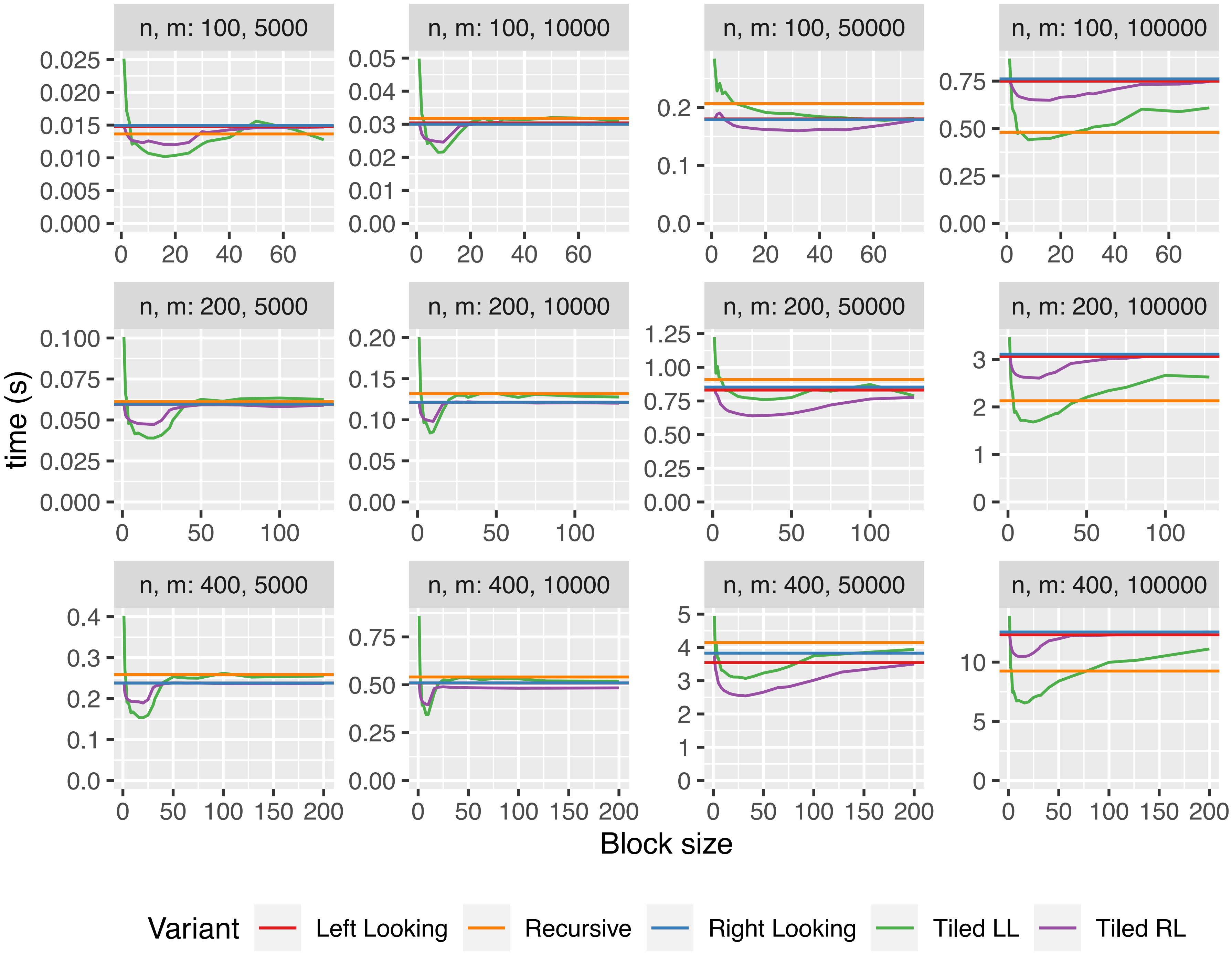
A2V (geqr2)
bora@plafrim
Intel CascadeLake



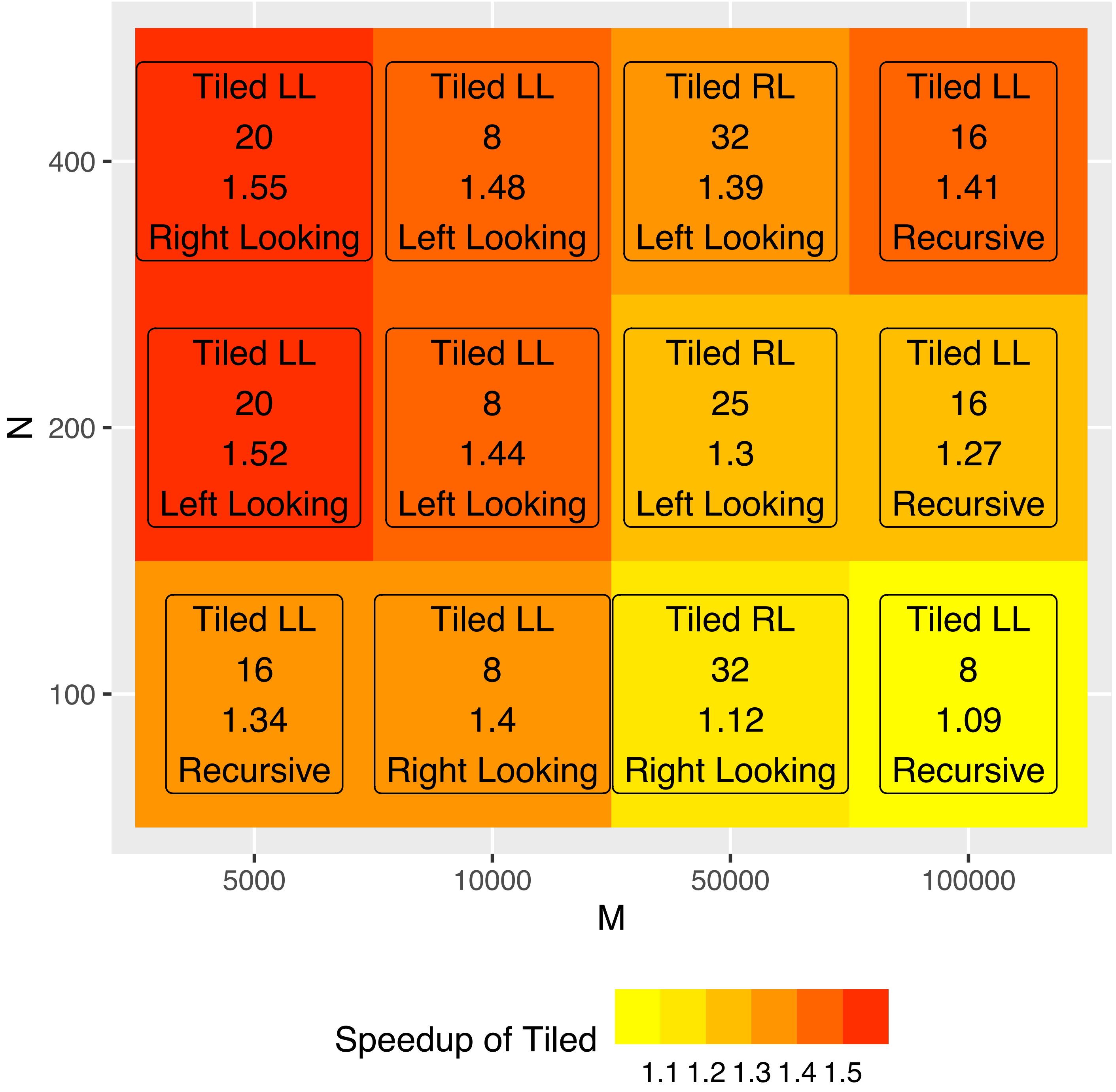
A2V (geqr2)
bora@plafrim
Intel CascadeLake



MGS
bora@plafrim
Intel CascadeLake

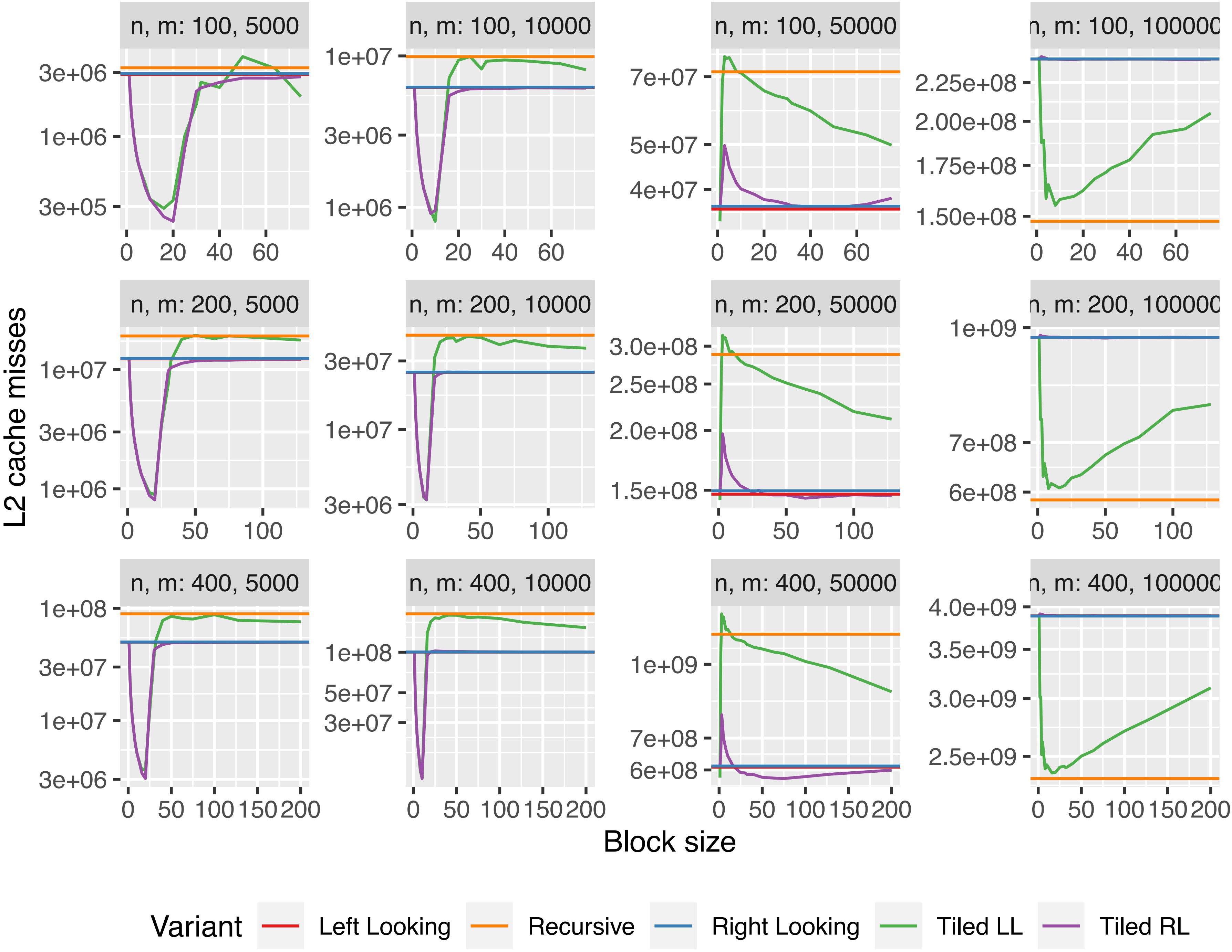


MGS
bora@plafrim
Intel CascadeLake

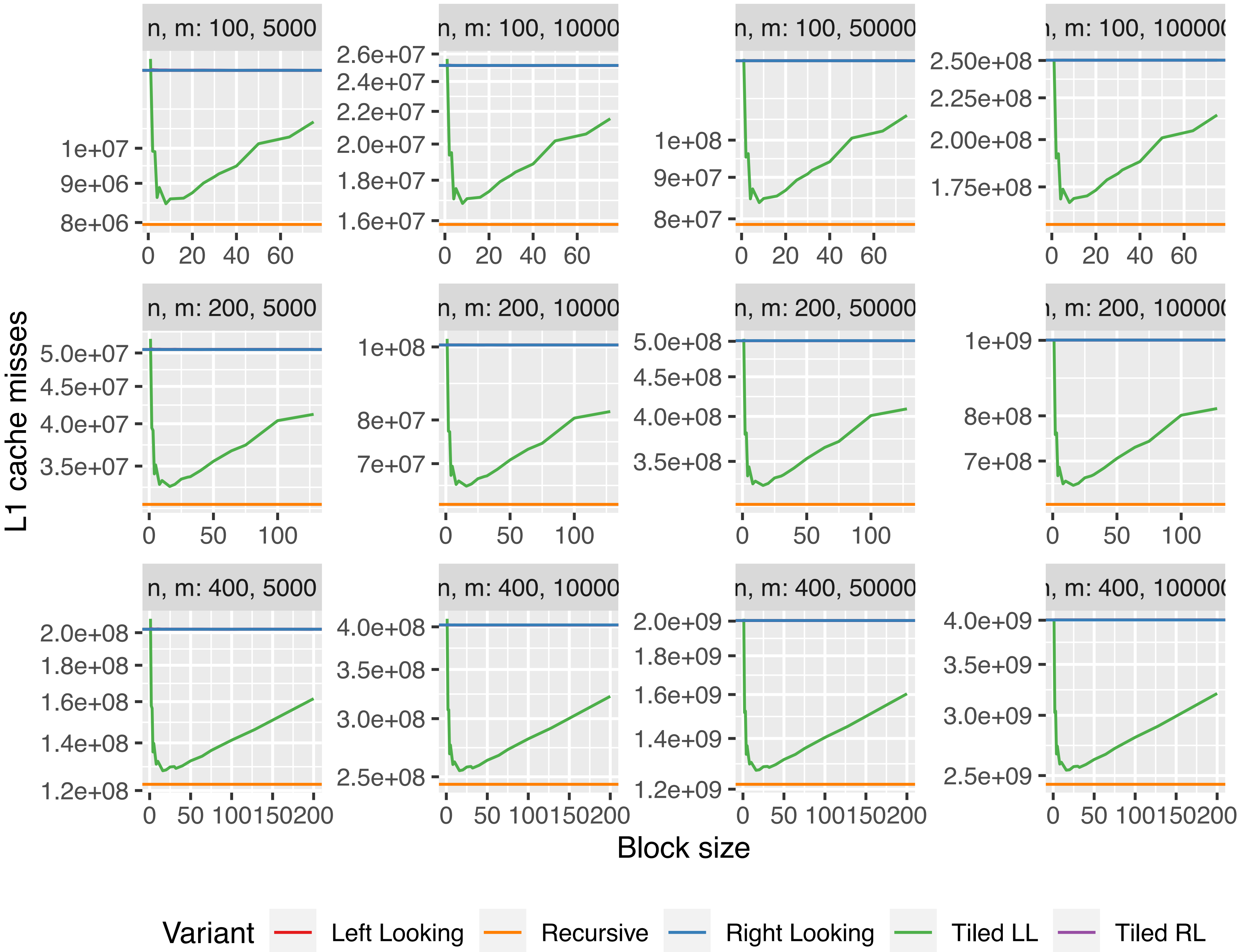


N \ M	5000	10000	50000	100000
400	Tiled LL 20 1.55 Right Looking	Tiled LL 8 1.48 Left Looking	Tiled RL 32 1.39 Left Looking	Tiled LL 16 1.41 Recursive
200	Tiled LL 20 1.52 Left Looking	Tiled LL 8 1.44 Left Looking	Tiled RL 25 1.3 Left Looking	Tiled LL 16 1.27 Recursive
100	Tiled LL 16 1.34 Recursive	Tiled LL 8 1.4 Right Looking	Tiled RL 32 1.12 Right Looking	Tiled LL 8 1.09 Recursive

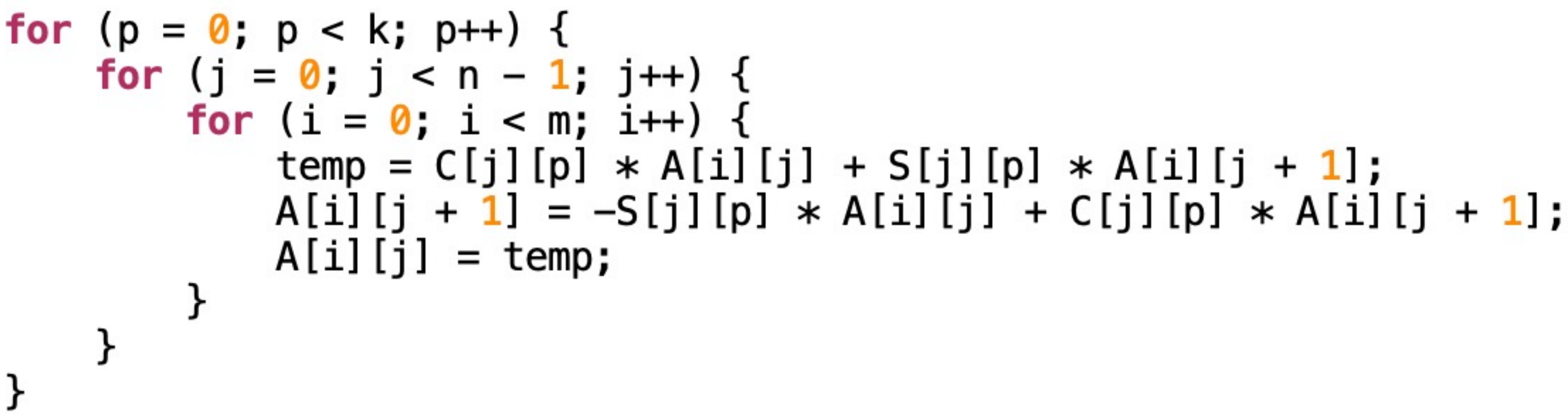
MGS
bora@plafrim
Intel CascadeLake



MGS
bora@plafrim
Intel CascadeLake

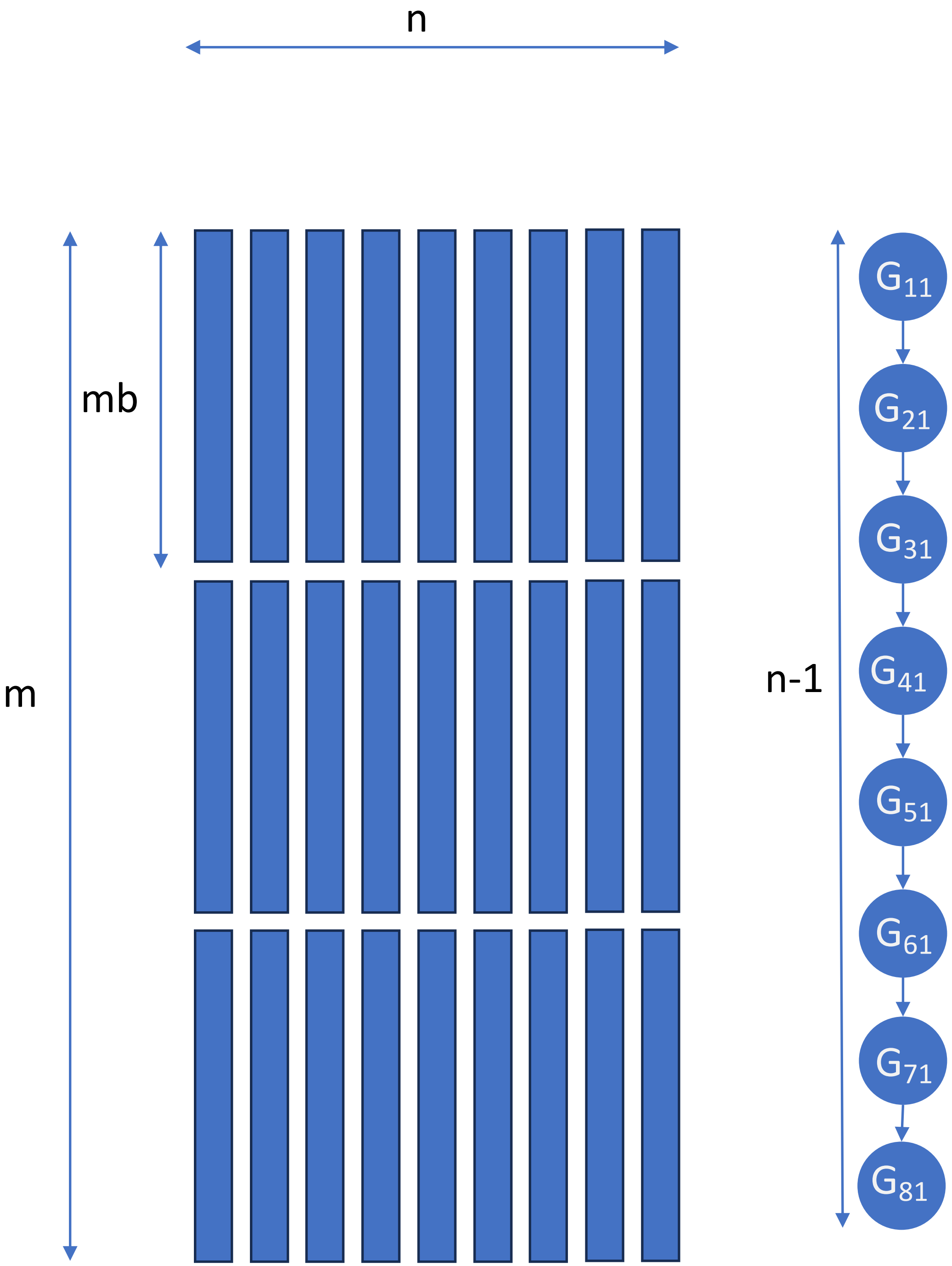


Current work: chains of Givens rotations with Thijs Steel



```
for (p = 0; p < k; p++) {
    for (j = 0; j < n − 1; j++) {
        for (i = 0; i < m; i++) {
            temp = C[j][p] ∗ A[i][j] + S[j][p] ∗ A[i][j + 1];
            A[i][j + 1] = −S[j][p] ∗ A[i][j] + C[j][p] ∗ A[i][j + 1];
            A[i][j] = temp;
        }
    }
}
```

A Francis step



work on columns 1 and 2

work on columns 2 and 3

work on columns 3 and 4

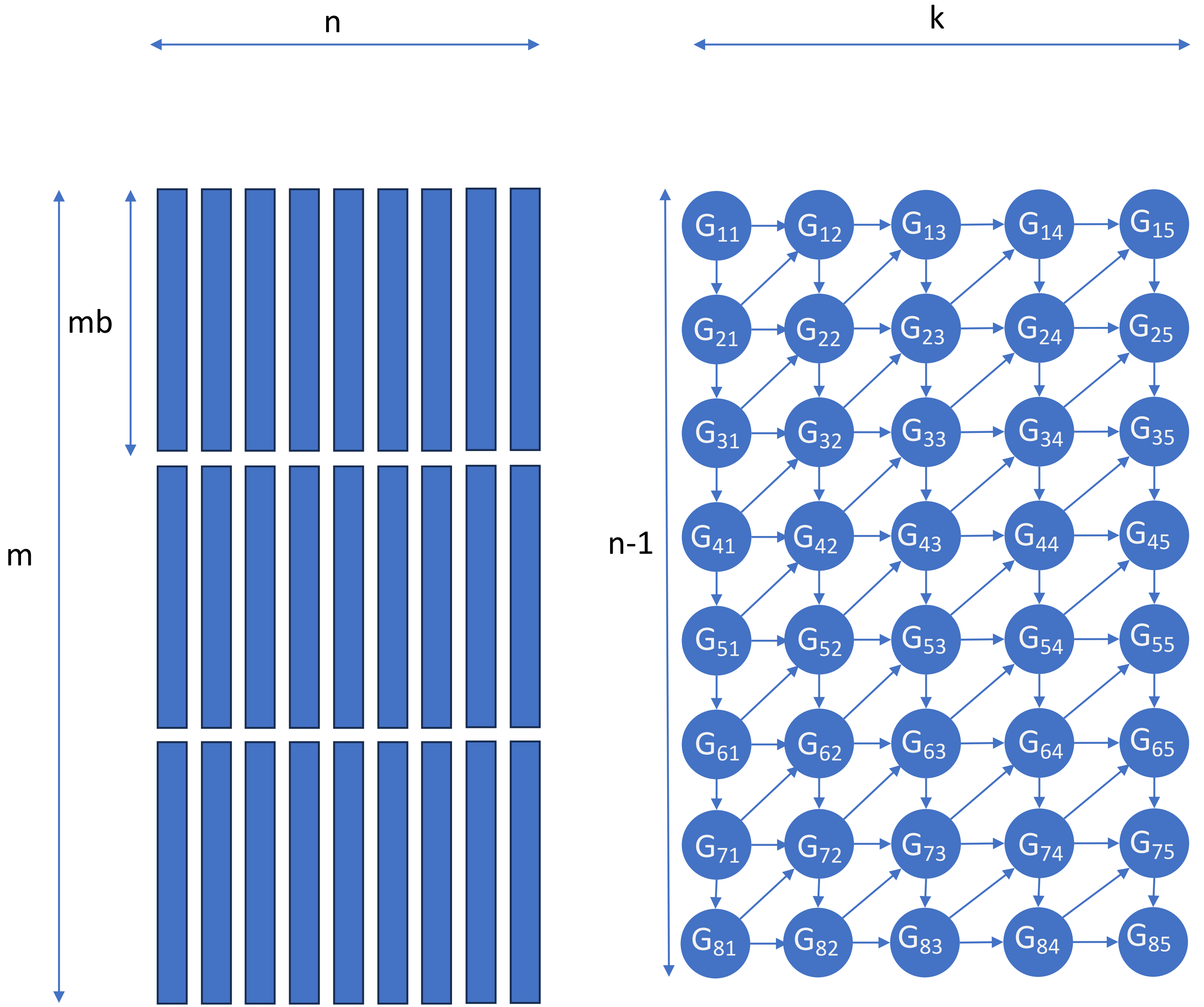
work on columns 4 and 5

work on columns 5 and 6

work on columns 6 and 7

work on columns 7 and 8

work on columns 8 and 9



A waves of five Francis steps

work on columns 1 and 2

work on columns 2 and 3

work on columns 3 and 4

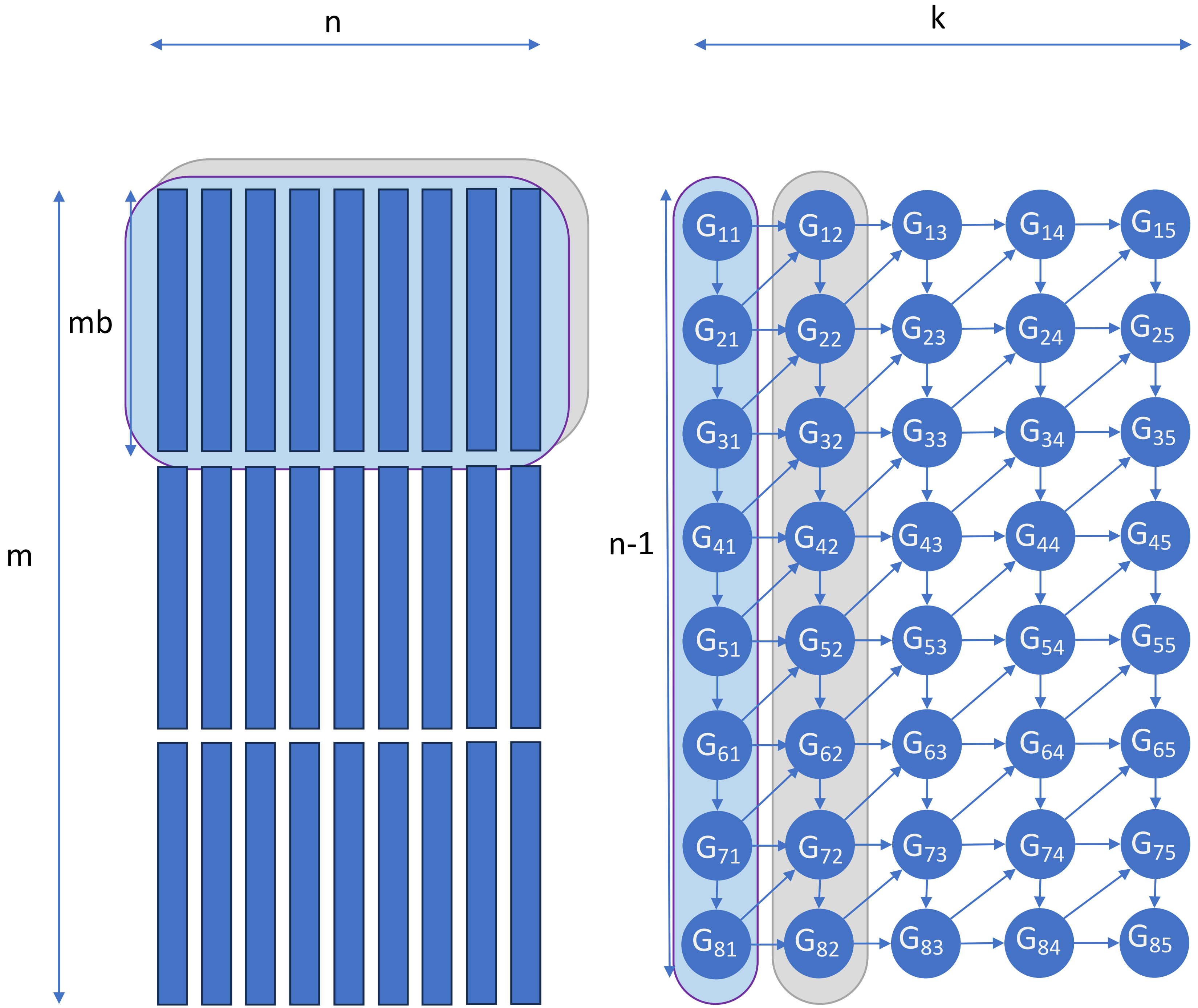
work on columns 4 and 5

work on columns 5 and 6

work on columns 6 and 7

work on columns 7 and 8

work on columns 8 and 9


n
k
mb
m
n-1
G_{11} G_{12} G_{13} G_{14} G_{15}
G_{21} G_{22} G_{23} G_{24} G_{25}
G_{31} G_{32} G_{33} G_{34} G_{35}
G_{41} G_{42} G_{43} G_{44} G_{45}
G_{51} G_{52} G_{53} G_{54} G_{55}
G_{61} G_{62} G_{63} G_{64} G_{65}
G_{71} G_{72} G_{73} G_{74} G_{75}
G_{81} G_{82} G_{83} G_{84} G_{85}

work on columns 1 and 2

work on columns 2 and 3

work on columns 3 and 4

work on columns 4 and 5

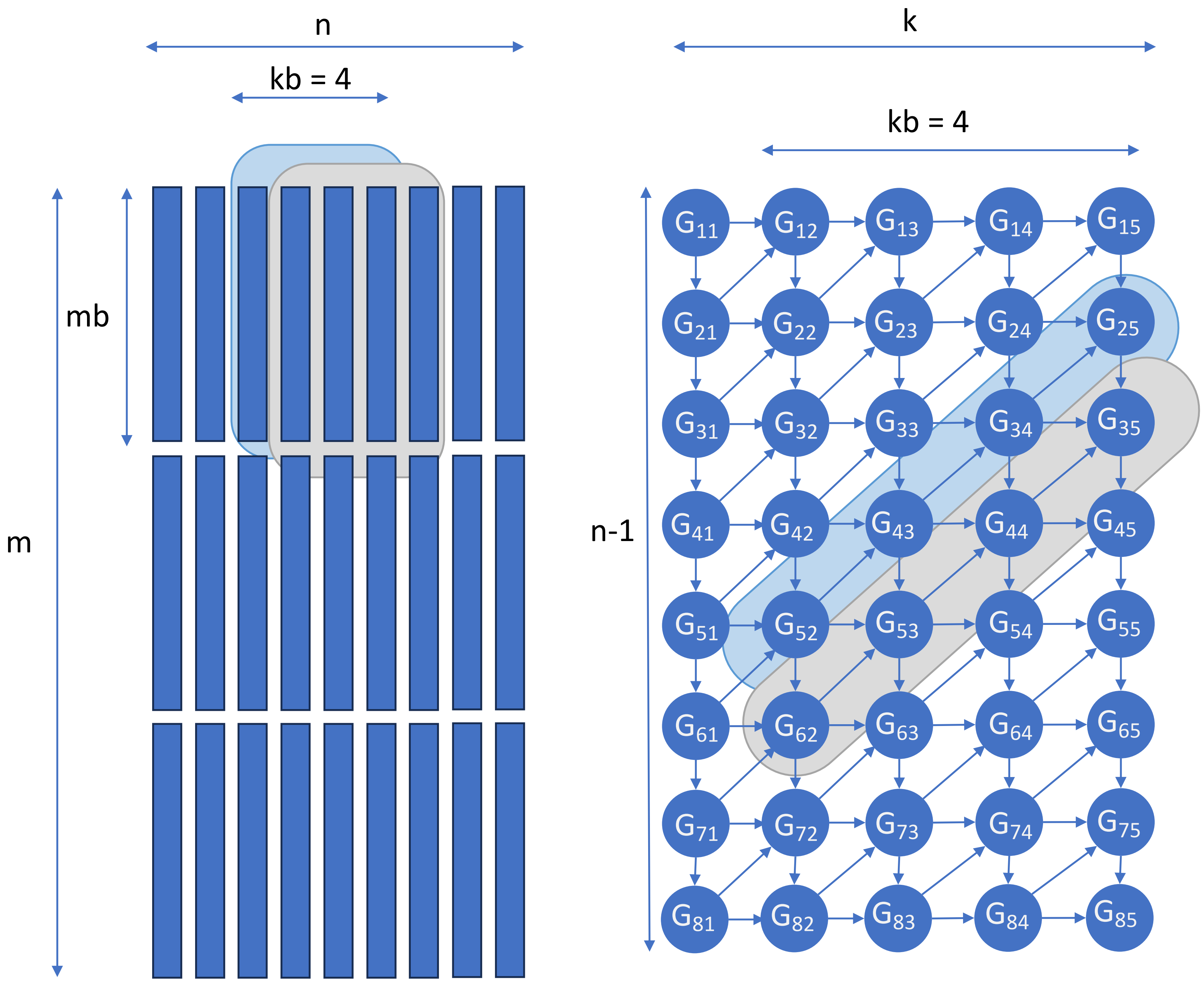
work on columns 5 and 6

work on columns 6 and 7

work on columns 7 and 8

work on columns 8 and 9

Field G. Van Zee, Robert A. van de Geijn, and Gregorio Quintana-Ortí. 2014. Restructuring the Tridiagonal and Bidiagonal QR Algorithms for Performance. ACM Trans. Math. Softw. 40, 3, Article 18 (April 2014), 34 pages. https://doi.org/10.1145/2535371


n
kb = 4
mb
m
k
kb = 4
n-1
G_{11} G_{12} G_{13} G_{14} G_{15}
G_{21} G_{22} G_{23} G_{24} G_{25}
G_{31} G_{32} G_{33} G_{34} G_{35}
G_{41} G_{42} G_{43} G_{44} G_{45}
G_{51} G_{52} G_{53} G_{54} G_{55}
G_{61} G_{62} G_{63} G_{64} G_{65}
G_{71} G_{72} G_{73} G_{74} G_{75}
G_{81} G_{82} G_{83} G_{84} G_{85}

work on columns 1 and 2

work on columns 2 and 3

work on columns 3 and 4

work on columns 4 and 5

work on columns 5 and 6

work on columns 6 and 7

work on columns 7 and 8

work on columns 8 and 9

Analysis

- Going fast.
- We need to apply mnk Givens rotations.
- We assume that $k_b * m_b \leq M$
- We move on piece of column of size m_b, write back one piece of size m_b, and load $2k_b$ rotations
- And we will do $k_b * m_b$ rotations in a segment
- Forgetting start and end clean up code, we need to do

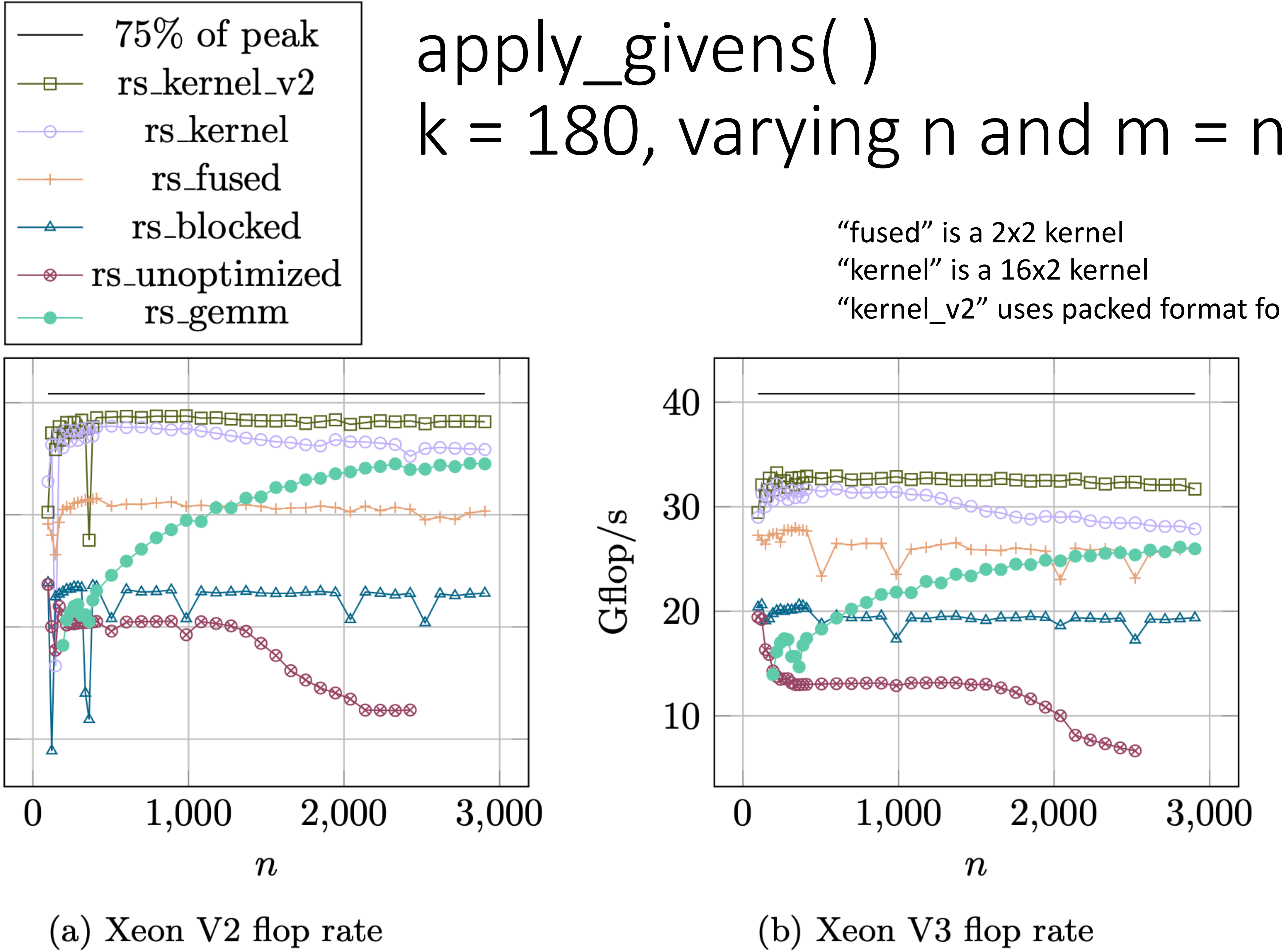
  $mnk / m_b / k_b$ segments
- And so the volume of communication of the algorithm is

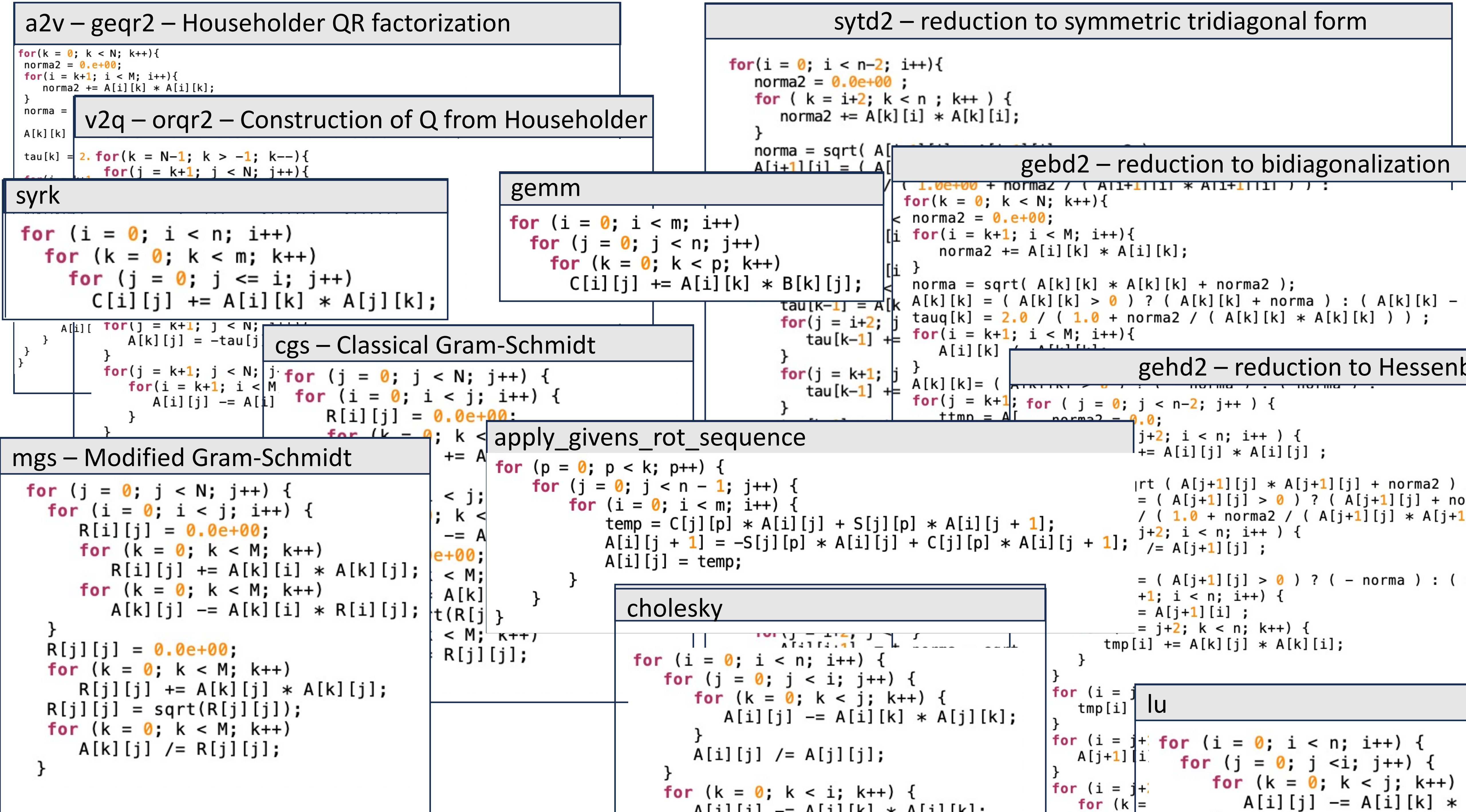
  $(mnk / m_b / k_b) * (2m_b + 2k_b)$
- For $k_b = \text{sqrt}(M)$ and $m_b = \text{sqrt}(M)$ we get

  $4\ mnk / \text{sqrt}(M)$
- Since the total number of operations of our code is 6mnk, this means that the operational intensity for the wavefront algorithm is **(3/2) sqrt{S}**
- IOLB returns **6 sqrt{S}** as an upper bound for the operational intensity
- So we are a factor of 4 off.

apply_givens( )
k = 180, varying n and m = n

“fused” is a 2x2 kernel
“kernel” is a 16x2 kernel
“kernel_v2” uses packed format for A





a2v – geqr2 – Householder QR factorization

```
for(k = 0; k < N; k++){
 norma2 = 0.e+00;
 for(i = k+1; i < M; i++){
    norma2 += A[i][k] * A[i][k];
 }
 norma =

 A[k][k]

 tau[k] =

    A[i][
   }
  }
 }
```

v2q – orqr2 – Construction of Q from Householder

```
2. for(k = N-1; k > -1; k--){
     for(j = k+1; j < N; j++){

     for(j = k+1; j < N;
        A[k][j] = -tau[j
     }
     for(j = k+1; j < N; j
        for(i = k+1; i < M
           A[i][j] -= A[i]
        }
     }
```

sytd2 – reduction to symmetric tridiagonal form

```
for(i = 0; i < n-2; i++){
   norma2 = 0.0e+00 ;
   for ( k = i+2; k < n ; k++ ) {
      norma2 += A[k][i] * A[k][i];
   }
   norma = sqrt( A[
   A[i+1][i] = ( A[
      ( 1.0e+00 + norma2 / ( A[i+1][i] * A[i+1][i] ) ) ;

      tau[k-1] = A[k
      for(j = i+2; j
         tau[k-1] +=
      }
      for(j = k+1; j
         tau[k-1] +=
      }
```

gebd2 – reduction to bidiagonalization

```
for(k = 0; k < N; k++){
 norma2 = 0.e+00;
 for(i = k+1; i < M; i++){
    norma2 += A[i][k] * A[i][k];
 }
 norma = sqrt( A[k][k] * A[k][k] + norma2 );
 A[k][k] = ( A[k][k] > 0 ) ? ( A[k][k] + norma ) : ( A[k][k] –
 tauq[k] = 2.0 / ( 1.0 + norma2 / ( A[k][k] * A[k][k] ) ) ;
 for(i = k+1; i < M; i++){
    A[i][k]
 }
 A[k][k]= (
 for(j = k+1
   ttmp = A[
```

gehd2 – reduction to Hessenb

```
for ( j = 0; j < n-2; j++ ) {
   norma2 = 0.0;
     j+2; i < n; i++ ) {
     += A[i][j] * A[i][j] ;

     rt ( A[j+1][j] * A[j+1][j] + norma2 )
     = ( A[j+1][j] > 0 ) ? ( A[j+1][j] + no
     / ( 1.0 + norma2 / ( A[j+1][j] * A[j+1
     j+2; i < n; i++ ) {
     /= A[j+1][j] ;

     = ( A[j+1][j] > 0 ) ? ( – norma ) : (
     +1; i < n; i++) {
     = A[j+1][i] ;
     = j+2; k < n; k++) {
     tmp[i] += A[k][j] * A[k][i];
   }
}
for (i = j
  tmp[i]
}
for (i = j+
  A[j+1][i
}
for (i = j+
  for (k
```

syrk

```
for (i = 0; i < n; i++)
  for (k = 0; k < m; k++)
    for (j = 0; j <= i; j++)
      C[i][j] += A[i][k] * A[j][k];
```

gemm

```
for (i = 0; i < m; i++)
  for (j = 0; j < n; j++)
    for (k = 0; k < p; k++)
      C[i][j] += A[i][k] * B[k][j];
```

cgs – Classical Gram-Schmidt

```
for (j = 0; j < N; j++) {
  for (i = 0; i < j; i++) {
     R[i][j] = 0.0e+00;
     for (k = 0; k <
        += A
     < j;
     ; k <
        -= A
     e+00;
     < M;
     A[k]
     t(R[j
     < M; k++)
     R[j][j];
```

apply_givens_rot_sequence

```
for (p = 0; p < k; p++) {
    for (j = 0; j < n - 1; j++) {
        for (i = 0; i < m; i++) {
            temp = C[j][p] * A[i][j] + S[j][p] * A[i][j + 1];
            A[i][j + 1] = -S[j][p] * A[i][j] + C[j][p] * A[i][j + 1];
            A[i][j] = temp;
        }
    }
}
```

mgs – Modified Gram-Schmidt

```
for (j = 0; j < N; j++) {
  for (i = 0; i < j; i++) {
     R[i][j] = 0.0e+00;
     for (k = 0; k < M; k++)
        R[i][j] += A[k][i] * A[k][j];
     for (k = 0; k < M; k++)
        A[k][j] -= A[k][i] * R[i][j];
  }
  R[j][j] = 0.0e+00;
  for (k = 0; k < M; k++)
     R[j][j] += A[k][j] * A[k][j];
  R[j][j] = sqrt(R[j][j]);
  for (k = 0; k < M; k++)
     A[k][j] /= R[j][j];
}
```

cholesky

```
for (i = 0; i < n; i++) {
  for (j = 0; j < i; j++) {
    for (k = 0; k < j; k++) {
      A[i][j] -= A[i][k] * A[j][k];
    }
    A[i][j] /= A[j][j];
  }
  for (k = 0; k < i; k++) {
    A[i][i] -= A[i][k] * A[i][k];
```

lu

```
for (i = 0; i < n; i++) {
  for (j = 0; j <i; j++) {
    for (k = 0; k < j; k++)
      A[i][j] -= A[i][k] *
```

The I/O Requirements of Various Numerical Linear Algebra Kernels

Julien Langou

Tuesday June 26th 2024