Aalto University

Algorithm for calculating HNF of a matrix $M \in \mathbb{Z}^{n \times m}$ in C++

Kristian Wasastjerna
kristian.wasastjerna@aalto.fi

Pseudo code

- You create a matrix L to change the leftmost non lower triangular columns lowest (positionally) non zero element into zero such that det(L)=1
- Proceed upwards until the column is in lower triangular form
- Move onto the next column
- Continue until the matrix is lower triangular form

Algorithm 1: HNF(M)

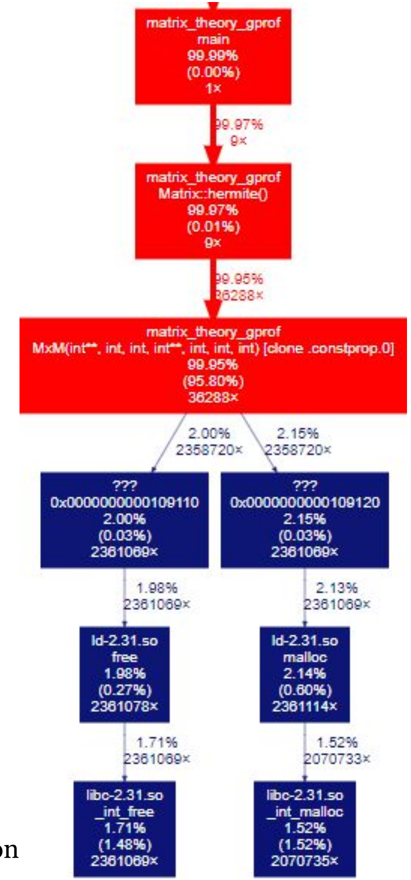
[n,m] $\leftarrow$ size(M)
$U \leftarrow IMat(n)$
$L \leftarrow IMat(n)$
$H \leftarrow M$
for $j \leftarrow 1...m$ **do**
  for $i \leftarrow n...j+1$ **do**
    if $H_{i,j} = 0$
      Nothing
    else if $H_{i-1,j} = 0$
      Swap rows $i-1$ and i in L
    else
      $a \leftarrow H_{i-1,j}, b \leftarrow H_{i,j}$
      $g \leftarrow gcd(a,b)$
      $[x,y] = \text{DiophantesSolver}(a,b,g)$
      $\alpha \leftarrow \frac{a}{g}, \beta \leftarrow \frac{b}{g}$
      $L_{i,i} \leftarrow \alpha,\ L_{i,i-1} \leftarrow -\beta,\ L_{i-1,i} \leftarrow y,\ L_{i-1,i-1} \leftarrow x$
    end
    $U \leftarrow LU, H \leftarrow LH$
    $L \leftarrow IMat(n)$
  end
end
return $[U,H]$

Pseudo code complexity

- Current complexity is $O(n^5)$
  - Since Matrix multiplication is $O(n^3)$
- Everything else is negligible time
  - O(log(max(A, B))) time complexity for Diophante solver
  - O(n) time for swapping rows of L
  - O(log(min(a, b)) time for gcd

Optimizing the algorithm

- Running profiling we can detect bottlenecks
  - Clearly only relevant bottleneck is matrix multiplication

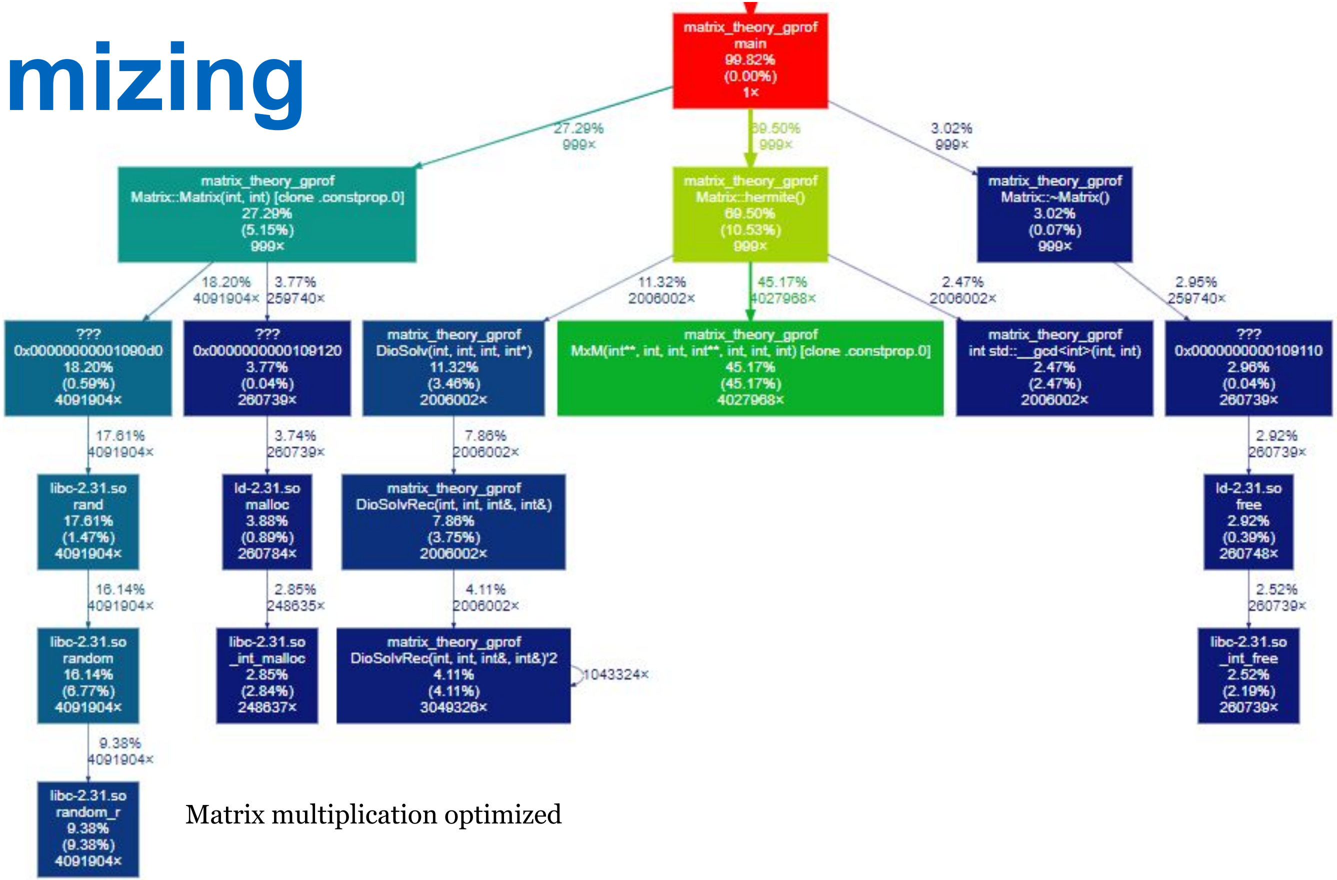


Naive implementation

Optimizing the algorithm

- Since L is a sparse matrix, with the form $I \oplus G \oplus I = L$, were $G \in \mathbb{Z}^{2\times 2}$, we see that in LU, L only acts on the rows where G is located
- This means that $LU = [U_1, GU_2, U_3]^T$ were $U_2 \in \mathbb{Z}^{2\times m}$
- Hence we can change the matrix multiplication used in the algorithm to a specific case with complexity O(m)
- Also resetting L to a identity matrix can be done by replacing G with a identity matrix $I \in \mathbb{Z}^{2\times 2}$

Optimizing

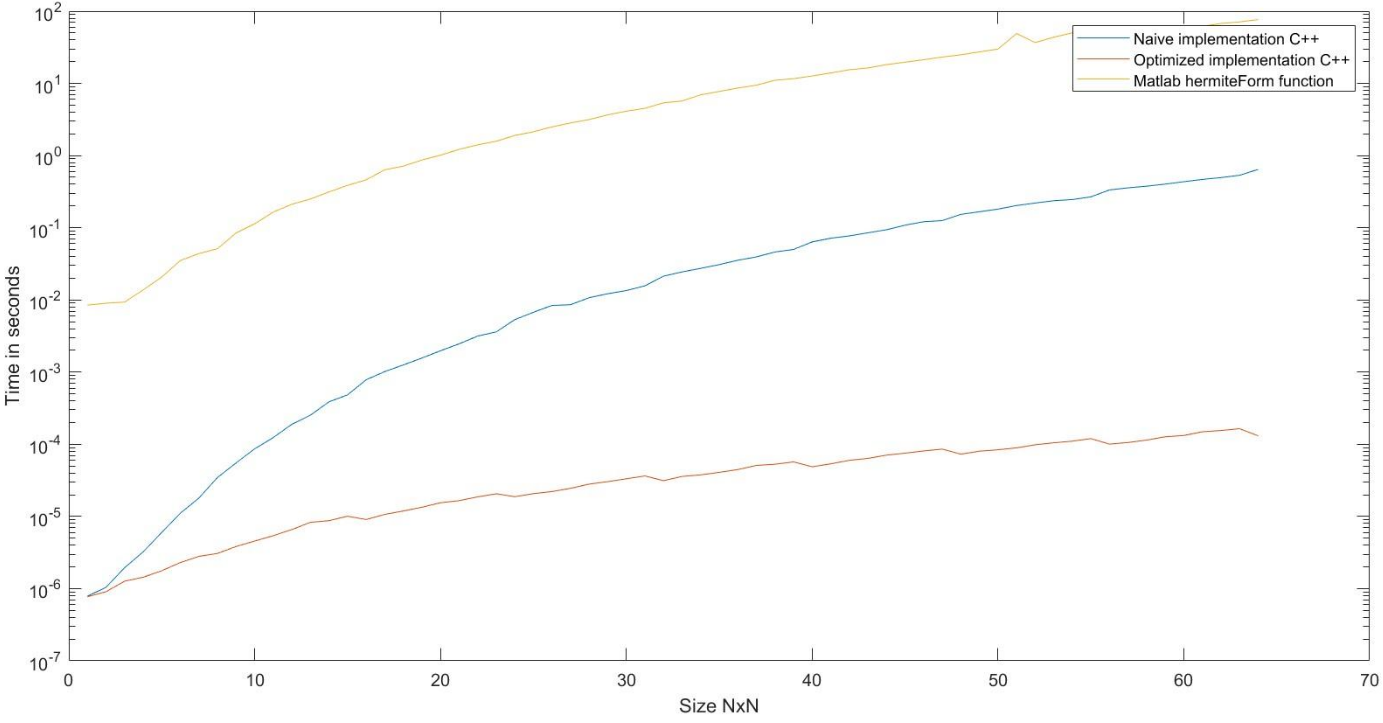


Matrix multiplication optimized

Optimized algorithm

- Still over half of the time used in calculating the HNF is taken by matrix multiplication
- Diophantine equation solver takes around 15% of the time
- And around 15% is spent inside the HNF algorithm itself
  - Mainly updating variables and if statements

Comparing algorithms to Matlab



Comparing algorithms to Matlab

- Clearly the optimized version is a lot more efficient in runtime
- Compared to the matlab version both were vastly superior, why?
  - Possibly since Matlab automatically stores values as double precision floats
  - Matlab is used for mostly matrices over real and complex numbers
    - HNF implementation might be of little importance
  - Overflow checking, see next slide

Shortcomings of algorithm in C++

- There is no integer overflow guard in the algorithm
  - Since the values, especially at the bottom right grow large extremely quickly overflow errors happen and the result is not correct
  - This starts happening when elements in $M \in \mathbb{Z}^{n \times n}$ being between -10 and 10 and n>5

Sources

- https://github.com/KristianWasas/HNF-calc
- Chapter 2 of lecture notes
- https://github.com/google/benchmark
    - For benchmarking
- https://github.com/jrfonseca/gprof2dot
    - For visualization of profiling data
- Matlab R2022b for plotting