# Polylog-depth subcubic-work algorithms for Singular Value Decomposition & Related Problems

Alexander Bistagne

February 6, 2023

**Abstract**

This paper presents its algorithms as a web of reductions. Some of which are well known but included for completeness and analysis. The subcubic work relies on fast matrix multiplication in time $O(n^\omega)$ which might not be practical. Some major sources of numerical stability are discussed but not analyzed. This paper works in the algebraic circuit model in order to move focus on number of field ops instead of precision. Within the model, computing the singular value decomposition can be achieved in ??? depth and ??? work; finding an orthogonal basis of a matrix can be achieved in ??? depth and ??? work; computing the pseudoinverse of a matrix can be achieved in ??? depth and ??? work.

# 1 Introduction

This paper is a rough draft; therefore no background/base introduction.

## 1.1 Overview

As is standard in theoretical computer science, the preliminaries will introduce the mathematics necessary but not widely familiar. Sections 3 to 15 will each be about a different reduction with the sections for the problem, its algorithm, its correctness, its depth, its work, a note on numerical stability. A diagram of the reduction web is below. Section 16 will combine the analysis in the previous sections so that the results can be stated merely in terms of matrix multiplication instead of the reduction chain. Section 17 will conclude the paper.

Possible reformat sections, subsections -¿ subsections, theorems .

This paper is a rough draft; therefore no reduction web

# 2 Preliminaries

This paper is a rough draft; therefore no preliminaries

### 2.1 Basic Linear Algebra

### 2.2 Computational complexity

### 2.3 Numerical Methods

### 2.4 Advanced Linear Algebra

## 3 Partial Powers of a Matrix Reduces to Matrix Multiplication

### 3.1 Problem

The partial powers of a matrix is just a fancy name for the the first $\sqrt{2^{\lceil log(n) \rceil}}$ powers of a square matrix, and the first $\sqrt{2^{\lceil lg(n) \rceil}}$ powers of the $\sqrt{2^{\lceil lg(n) \rceil}}$ power of the matrix. This problem is not of historical note, but occurs twice in the reduction web.

### 3.2 Algorithm

The algorithm has two phases. In Phase one, we generate the powers which are powers of two by squaring the matrix. Phase two is a divide and conquer algorithm which we run twice. Lower powers = F(1,$\lceil lgn/2 \rceil$) ; Upper Powers = F( $\lceil lgn/2 \rceil$, $\lceil lgn \rceil$) Base Case: F(a,a) = Identity matrix concatenated to the $A^{2^a}$ Induction Case: F(a,b) = BlockVectorize( BlockTranspose(F(a,$\lceil \frac{a+b}{2} \rceil - 1$)) * F( $\lceil \frac{a+b}{2} \rceil$, b ) ) Where block vectorize and block transpose treat the nxn matrices as elements which are left undisturbed, and behave as the standard vectorize and transpose operations.

### 3.3 Correctness

This trivial proof is left as an exercise to the reader or editor.

### 3.4 Depth

Phase one takes $lgn$ matrix multiplications in sequence. Phase Two is a divide and conquer algorithm with depth recursion $T(m) = T(m/2)+O(MM_{depth}(n))$ and so is O($(loglogn)MM_{depth}(n)$). Note that the depth at each stage is not asymptotically changed because we multiply sub$n^2$ sized matrices and $Poly(2) * MM_depth(n) = MM_depth(n^2)$ which is not an asymptotic change. Thus the first phase dominates with depth $PP_{depth} \in O((logn)MM_{depth})$

### 3.5 Work

Phase one takes $lgn$ matrix multiplications in total. Phase Two's recursion relation is $W(m) = 2 * W(m/2) + 2^m * O(MM_{work}(n))$. This is a very nonstandard recursion relation, but turning it into a summation over levels and multiplying number of subproblems at

each level by number of matrix multiplications at that subproblem:

$$W(m) = \sum_{i=0}^{lglgn} 2^i * 2^{2^i} * MM_{work}(n) = MM_{work}(n) * \sum_{i=0}^{lglgn} 2^i * 2^{2^{lglgn-i}} < MM_{work}(n)2\sqrt{n}$$

$$PP_{work}(n) \in O(\sqrt{n}MM_{work}(n))$$

**Remark 3.1.** *Notice that this relies on Square matrix multiplication, but might be improved by analyzing it using rectangular matrix multiplication.*

FORMAT SHIFT ERROR

## 3.6 Trace of powers of a matrix reduces to partial powers and matrix multiplication

**Definition 3.2.** *Trace of powers of a matrix is a list of n values of i=1 ... n of* $tr(A^i)$

This Problem is not particularly special, but does highlight an interesting way to generate a lot of traces in parallel. It is also useful to separate it from the problem of solving a triangular system in parallel which is of independent interest.

**Algorithm 3.3.** *Given a matrix A, First compute the lower and upper partial powers L and U. antiblock transpose the matrices in L -¿ . Then AntiblockVectorize L' and U. Then BlockTranspose L' -¿ L". Then Multiply L' by U. This new* $\sqrt{n}$ *sized square matrix will contain all the traces in column major order.*

So while the only work done is one matrix multiplication, You might not implicitly guess what the data manipulation is doing. This algorithm exploits the property that both a single row times a single column is a dot product and that the trace of the product of two matrices is a dot product. We are reducing the later to the former.

**Theorem 3.4.** *The above algorithm correctly solves the above problem*

*Proof.* $L = [A^1 ... A^{\sqrt{n}}]$ then $L' = [A^{1T} ... A^{\sqrt{n}T}]$ then we are vectorizing each block matrix while preserving the ration between blocks corresponding to each power of A then we are putting this vectors in rows. U is handled similarly. Then we use the common fact that $tr(A^T B) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} b_{ij}$ to get the answer via matrix multiplication. □

**Theorem 3.5.** *The above algorithm* $TP_{depth} \in O(PP_{depth}(n) + MM_{depth}(\sqrt{n}) + log(n))$

**Theorem 3.6.** *The above algorithm* $TP_{work} \in O(PP_{work}(n) + n^{1.5} MM_{work}(\sqrt{n}))$

*Proof.* After calling the Partial Powers algorithm, we do a lot of data movement but no actual field operations, which is the thing being counted in this analysis. Then we do one matrix multiplication of a $\sqrt{n}$ by $n^2$ matrix by a $n^2$ by $\sqrt{n}$ matrix. This can be done as $n^{1.5}$ square matrix multiplications of size $\sqrt{n}$ followed by adding all those products together. □

**Remark 3.7.** *I do not see any source of numerical instability in this algorithm.*

## 3.7 Solving an invertible Triangular system reduces to Matrix Multiplication

There are already good algorithms for solving a triangular system but they do not fit into my target intersection of subcubic work and poly log depth.

**Definition 3.8.** *Given Ax=b where A is an n by n matrix, and b is an n by 1 matrix, find x with the promise that* $(A)_{ij} = 0$ *for all* $i > j$ *and* $(A)_{ii} \neq 0$.

**Algorithm 3.9.** *The algorithm has 3 phases. In Phase 1, Take the main diagonal of A to get D, and* $A' = D^{-1}A$ *;*$b' = D^{-1}b$. *Now remove the main diagonal of A' to get A". Then negate the matrix to get N =- A". In Phase2, Now construct the powers of N' which are powers of two by repeated squaring. Last phase compute the following as logn matrix additions and then logn matrix-vector multiplications.*

$$x = (\prod_{i=0}^{\lceil logn \rceil} (I + N^{2^i}))b'$$

**Theorem 3.10.** *The above algorithm is correct.*

*Proof.* So the main claim of correctness is that $x = A^{-1}b$. Because $A = D(I - N)$, then $A^{-1} = (I + N + N^2...N^{n-1} + N^n)D^{-1}$ by a well known property of Nilpotent matrices.The sum of those N matrices can be computed by $(I+N)(I+N^2)...(I+N^{2^{\lceil lgn \rceil - 1}})(I+N^{2^{\lceil lgn \rceil}})$ □

**Theorem 3.11.** *The above algorithm takes depth* $TS_{depth} \in O(log(n)MM_{depth}(n))$ *and* $TS_{work} \in O(log(n)MM_{work}(n))$

*Proof.* The first phase of the algorithm does some simple arithmetic and one matrix multiplication. The second phase of the algorithm does O(logn) matrix multiplications in sequence. The third phase then does O(logn) matrix additions in parallel and O(logn) matrix-vector multiplications in sequence □

FORMAT CHANGE COMBINE DEPTH AND WORK MOVE NUMERICAL ANALYSIS TO OTHER SECTION

## 3.8 Csanksy's Algorithms

Csanksy's Algorithm for the inverse of matrix can be seen as a reduction from inverse to characteristic polynomial and polynomial of a matrix, and characteristic polynomial of a matrix to traces of powers of a matrices and solving a triangular system. With also NC algorithms for the mentioned problems. PROPER CITATION NEEDED HOW DO I REFRAME AN ALGORITHM AS A REDUCTION. SHOULD I RESTATE THE PROOF?

## 3.9 The polynomial of a Matrix reduces to Matrix Multiplication

There is an obvious algorithm for this problem with $O(nMM_{work}(n))$ work, but that isn't subcubic because matrix multiplication takes $\Omega(n^2)$.

**Definition 3.12.** *Given a polynomial p of degree n via its coefficients, and a n by n matrix A. Compute p(A)*

**Algorithm 3.13.** *First compute the partial powers of A to get L, U. Transpose(AntiBlockVectorize(L)) to get L'. Arrange the coefficients into a $\sqrt{n}$ by $\sqrt{n}$ square matrix C. Multiply C by L' to get P'. Then AntiBlockUnVectorize(P') P". Then Multiply U by P" to get p(A).*

Note this might have off by one errors....

**Theorem 3.14.** *The above algorithm is correct.*

*Proof.* The Algorithm is built off noticing that you can build the polynomial in $\sqrt{n}$ sized sections from first combining them as linear combinations of the lower partial powers, then mutiplying each by the appropriate upper power before summing them together the sections. □

**Theorem 3.15.** *For the above algorithm, $PM_{depth} \in O(PP_{depth}(n) + MM_{depth}(n) + logn)$ and $PM_{work}(n) \in O(PP_{work}(n) + n^{1.5}MM_{work}(\sqrt{n}) + n^{.5}MM_{work}(n))$*

*Proof.* In the first computation step we call the PP subroutine and so take on its work and depth, Then we do two matrix multiplications one of size $\sqrt{n}$ by $\sqrt{n}$ matrix by a $\sqrt{n}$ by $n^2$ matrix. The second matrix multiplication is the multiplication of a $n$ by $n^{1.5}$ matrix by a $n^{1.5}$ by $n$ matrix. The former of these rectangular matrix multiplications can be analyzed as $n^{1.5}$ many $\sqrt{n}$ by $\sqrt{n}$ square matrix multiplications, followed by many matrix additions which take logn depth and at most $n^{2.5}$ work but that is dominated by the matrix multiplications by lower bounds on matrix multiplications. The latter of these matrix multiplications can be analyzed as $\sqrt{n}$ many $n$ by $n$ multiplications. □

CITE lower bounds

## 3.10 Eigenvalues of a Hermitian reduces to Characteristic Polynomial and Finding real zeros of a polynomial

THIS SECTION FEELS TRIVIAL Its essentially just compute characteristic polynomial then find its zeros with a parallel algorithm.

## 3.11 PseudoInverse reduces to Eigenvalues of Hermitian, Polynomial of a Matrix

DO I NEED TO DEFINE THIS PROBLEM Let m be greater than n.

**Algorithm 3.16.** *Multiply A* by A to get B. Compute the eigenvalues of B to get e. Then invert nonzero eigenvalues e to get e'. Construct polynomial p such that $p(e_i) = e'_i$. Compute p(B) to get P. Multiply P by A* to get the desired result $A^+$.*

**Theorem 3.17.** *The above algorithm is correct.*

*Proof.* So the pseudo inverse is first reduced to the hermitian case, then That case is easily solved by simply inverting all the nonzero eignevalues because the hermitian matrix is diagonalizable. In the process you need to compute a polynomial interpolator. □

CITE AUTHORS FOR POLYNOMIAL INTERPOLATION or analyze a simple algorithm?$n^2$ work

**Theorem 3.18.** *The above algorithm takes* $PI_{depth} \in O(MM_{depth}(n,m) + EH_{depth}(n) + PM_{depth}(n) + log^2(n))$ *and* $PI_{work} \in O(EH_{work}(n) + PM_{work}(n) + n^2 + MM_{work}(n,m))$

*Proof.* Follows from breaking the algorithm into its steps each of which is a separate subroutine □

## 3.12 PseudoInverse reduces to Characteristic Polynomial and Polynomial of a matrix

This is both a more numerically stable equivalent to the previous algorithm, and a mild generalization of ?Csanky? algorithm for the inverse to the pseudoinverse.

**Algorithm 3.19.** *Given a m by n matrix A where m ¿= n and* $A \neq 0$*, Construct* $B = A^*A$*. Compute the characteristic polynomial of B to get cp. If zero is an eigenvalue of B then cp will have form* $rp * x^k$ *for some polynomial rp where* $rp(0) \neq 0$*(Edge case zero is not an eigenvalue of B then set rp to cp). Compute a third polynomial* $xp = \frac{rp - rp(0)}{rp(0)x}$*. Compute xp(B) to get C. Multiply C by (I -xp(0)A) and subtract xp(0)I where I is the n by n identity matrix to get* $B^+$ *Then multiply* $B^+$ *by A to get* $A^+$

**Theorem 3.20.** *The above algorithm is correct.*

*Proof.* The reduction from computing the pseudoinverse of A to B is the standard reduction from the general case to the hermitian case over the complex numbers. The second claim is that xp maps all the nonzero eigenvalues to their inverses. $xp(\lambda) = \frac{rp(\lambda) - rp(0)}{-rp(0)\lambda} = \frac{0 - rp(0)}{rp(0)\lambda} = \frac{1}{\lambda}$. Note we substract $rp(0)$ to guarantee that xp remains a polynomial. Next we claim $xp(A) = A^+ + xp(0)(I - A^+A)$ because polynomials act as polynomials on the eigenvalues of a matrix, and $P = (I - A^+A)$ has $nullspace(P) = rowspace(A)$ and $nullspace(P) = rowspace(A)$ that Then $xp(A)A = A^+A$ because for hermitian matrices $A^+A = AA^+$ and for all matrices $AA^+A = A$. Thus, $A^+ = xp(A) - xp(0)(I - xp(A)A)$ rearranging to $A^+ = xp(A)(I - xp(0)A) - xp(0)I$ which is what the algorithm computes. □

**Theorem 3.21.** *The algorithm takes depth* $O(MM_{depth}(n) + logn + CP_{depth}(n) + PM_{depth}(n))$ *and work* $O(\frac{m}{n}MM_{work}(n) + CP_{work}(n) + n + PM_{work}(n))$

*Proof.* Other than doing a matrix multiplication at the beginning and end with shape m,n,n and calling the characteristic polynomial subroutine and polynomial of a matrix subroutine we do O(n) divisions and O(n) additions to the main diagonal of a matrix. □

**Remark 3.22.** *I believe this algorithm is more numerically stable than the previous one, but I don't know how to prove that right now*

## 3.13 Reduction from Orthonormal Basis to Pseudoinverse

This reduction is roughly both a parallelization and work decrease of the Gram-Schmit Process and relies heavily on the following lemma:

**Lemma 3.23.** *Given a matrix $[AB]$ we can construct a matrix $[AC]$ with the same span but with the additional property $CA^\dagger = 0$*

*Proof.* Construction: $C = B - A^{\dagger +}A^\dagger B$ Proof of latter property: $A^\dagger C = A^\dagger B - A^\dagger A^{\dagger +}A^\dagger B = A^\dagger B - A^\dagger B$ by the property of the pseudo inverse $MM^+M = M$. Proof of the former property. The span of [A B] is a superset of the span of [A C] because the columns of $A^{\dagger +}A^\dagger B$ are firmly in the span of $A$ because $A^{\dagger +}A^\dagger B = A(A^\dagger A)^+A^\dagger B$ by the reduction to the hermitian case, so $[AC]$ is a linear recombination of $[AB]$. □

**Definition 3.24.** *Given a Matrix M, we define a useful orthogonal basis B to have three properties. 1,span(B) = span(M). 2, $M^\dagger M$ is a diagonal matrix. 3, If you take the indexes of the nonzero columns of M, the columns of B with those indexes form a basis of B.*

We will define the algorithm when n is a power of two and square. Polishing the algorithm for other input sizes is left as an exercise for the audience.

**Algorithm 3.25.** *Given B, we will construct a valid M. This happens in log(n) phases where the ith phase has $2^(i-1)$ sections (those sections divide up the columns of the current matrix in the most obvious way) and $A_{i,j}$ is the 1st half of the jth section of the ith phase, and $B_{i,j}$ is the 2nd half of the jth section of the ith phase. (Step 1) of phase i has all j sections do the following in parallel: $C_{i,j} = (A_{i,j}^\dagger A_{i,j})$. (Step 2) of phase i has all the j sections do the following in parallel: $D_{i,j} = (A_{i,j}^\dagger B_{i,j})$. (step 3) of phase i has all j sections do the following in parallel: $M_{i,j} = C_{i,j}^+ D_{i,j}$.. (Step 4) of phase i has all j sections do the following in parallel: $[A_{i+1,2j+1}B_{i+1,2j+1}] = B_{i,j} - A_{i,j}M_{i,j}$. Output when each section is a single column, all columns in order.*

**Theorem 3.26.** *The above algorithm solves the above problem.*

*Proof.* By substituting the steps together, we get $[A_{i+1,2j+1}B_{i+1,2j+1}] = B_{i,j} - A_{i,j}M_{i,j} = B_{i,j} - A_{i,j}C_{i,j}^+ D_{i,j} = B_{i,j} - A_{i,j}(A_{i,j}^\dagger A_{i,j})^+(A_{i,j}^\dagger B_{i,j})$ which by reversing the hermitian reduction is equal to $B_{i,j} - A_{i,j}^{\dagger +}A_{i,j}^\dagger B_{i,j}$.Therefore, the algorithm is repeatedly apply the above lemma on thinner and thinner matricies. Clearly then, span(B) = span(M). The argument for orthogonality ( $[A_{i+1,2j+1}B_{i+1,2j+1}]^\dagger [A_{i+1,2j+1}B_{i+1,2j+1}]$ being diagonal) follows from the fact that after the ith phase $[A_{i+1,2j+1}B_{i+1,2j+1}]^\dagger [A_{i+1,2j+1}B_{i+1,2j+1}]$ has an upper right and lower left quarters of all zeros. When the algorithm terminates, this is true at all levels and thus is diagonal. (MAYBE INCLUDE VISUALIZATION). The last property doesn't have a good name, but can be proved by the nature of the lemma. Essentially, on each step, some columns have some components which are in the span of earlier vectors removed (This follows from $A_{i,j}^{\dagger +}A_{i,j}^\dagger$ being an orthogonal projector) At the end all such components are removed. (clear from orthogonality). Because we were only removing components, if a vector changed the span at the beginning of the algorithm it is nonzero at the end of the algorithm; therefore the extra property is part of the answer. □

**Theorem 3.27.** *The algorithm takes depth* $OB_{depth}(n) \in O(PI_{depth}(n)log(n)+MM_{depth}(n)log(n)+log^2(n))$ *and work* $OB_{work}(n) \in O(PI_{work}(n) + MM_{work}(n)log(n))$

*Proof.* For brevity we are assuming the input matrix is square, the first, second, and fourth step when it has k sections has k operations in parallel where each rectangular matrix multiplication can be broken into 2k many (n/2k) dimension square matrix multiplications. With $MM_{work}(n) \in \omega(n^2)$ then each phase does at most $O(MM_{work}(n))$ for $O(MM_{work}(n)log(n)$ total work. With depth $O(MM_{depth}(n)log(n) + log^2(n))$ from these same multiplications and matrix additions. Step three does one pseudo inverse and one matrix multiplication per section. And per phase $O(MM_{work}(n/2k)k + PI_{work}(n/2k)k)$ where k is the number of sections in a phase. Because both matrix multiplication and pseudoinverse are superquadratic operations ( The former has a clear proof the later is a mild conjecture), then the total work forms a geometric series so that it is $O(MM_{work}(n) + PI_{work}(n))$. Likewise the depth is $O(MM_{work}(n)log(n) + PI_{work}(n)log(n) + log^2(n))$. Combining step 3 with the other steps we get $OB_{work} \in O(MM_{work}(n)log(n) + PI_{work}(n))$ and $OB_{depth} \in O(MM_{depth}(n)log(n) + PI_{depth}(n)log(n) + log^2(n))$ □

**Remark 3.28.** *This algorithm is one of the layers that makes the algorithm an* $NC^3$ *as opposed to* $NC^2$ *algorithm.*

**Remark 3.29.** *This algorithm is sometimes used in the reduction web for the usefulness property, and sometimes for the orthogonality property but always for the basis property.*

NUMERICAL STABILITY? RENAME useful TO ordered AND BETTER DEFINE

## 3.14 Reduction from Nullspace to Basis and Inverse

The following reduction is fairly obvious yet elegant, but is included for analysis purposes. DO I NEED TO DEFINE THE NULLSPACE PROBLEM

**Algorithm 3.30.** *Given a matrix M. Compute the Ordered Orthogonal Basis of M -¿ T to separate out the lexicographically minimum basis of M -¿ B from the other vectors C. Now we will solve BX = C by computing the pseudoinverse of B (* $(B^\dagger B)^+ B^\dagger$*) and right multiplying it by C to get X. The columns of X correspond to a set of linearly independent vectors of the nullspace of M by permuting the indicies of rows of X from the column indices of B to the column indicies of M and filling the other rows of the new matrix with zeros -¿ Y. Now MY = C, and there is an obvious subset of standard basis vectors S such that MS = C and TS = 0. Return N = Y-S.*

**Theorem 3.31.** *The above algorithm is correct.*

*Proof.* One M(Y-S) = C - C = 0, so MN = 0 and N is a nullspace. Because B is a basis for M, by definition, BX = C is a consistent set of systems.The last claim worth defending is that N is a set linearly independent vectors. N can have its rows permuted into a matrix with X ontop of -I; therefore it is linearly independent. □

**Theorem 3.32.** *The above algorithm has clear depth and work complexities???*

*Proof.* Given an m by n matrix. The two matrix multiplications have work of $(m/n)MM_{work}(n)$ and depth $log(n)+MM_{depth}(n)$. The algorithm also has work of $PI_{work}(n)$ and $OB_{work}(m,n)$ and work $PI_{depth}(n)$ and $OB_{depth}(m,n)$ because of the pseudoinverse and Ordered Orthoganal Basis subroutines. □

## 3.15 Reduction from intersection of spaces to nullspaces

**Definition 3.33.** *Given a pair of vector spaces A, B each represented as the span of a list of linearly independent vectors, finding the intersection of A and B is the problem*

**Algorithm 3.34.** *Construct a matrix C with columns first from vectors of A then B. Find its Nullspace N. The intersection is the product of [A 0] and N .*

Correctness: Trivial. Runtime: nullspace algorithm & one multiply. MERGE INTO PREVIOUS SUBSECTION?

## 3.16 Reduction from Diagonalizable Eigenvector to IoB and polynomial of a matrix

**Definition 3.35.** *Given a matrix A and its eigenvalues e with the promise of A being an diagonalizable matrix, find its eigenvectors.*

A future draft should explain how to handle eigenvalues of high multiplicicty.

**Algorithm 3.36.** *The following is a weird divide and conquer algorithm where we imagine each eigenvalue has an label between 0 and [n-1]. F(a,b) is a list of basis where each basis has a different bits in its eigenvalue label between bit position a and b. F(a,a) is the base case: let $p_0$ be a polynomial which is zero at every eigenvalue which has a zero in bit position a of its label. ( and similarly for $p_1$) compute $nullspace(p_0(A))$ and $nullspace(p_1(A))$ using the proper subroutines. F(a,b) is the divide case: Using the bases from F(a,floor((a+b)/2)) and F(ceil((a+b)/2),b), then for every pair of basis between the former and later list. Compute the IoB of them. F(1,lg(n)) is the start case.*

Correctness: Because the matrix is diagonalizable, a polynomial of the matrix acts directly on the eigenvalues. Thus the nullspace of the polynomial of the matrix spans all the eigenspaces intended to span. We then intersect smaller and smaller sets of eigenspaces until a single eigenspace is left. Depth: loglog(n) times the IoB subroutine. Work: AHHHHHH. Definitely polynomial. Analysis requires precise IoB analysis. STABILITY???

## 3.17 Reduction from SVD to Nullspace, Characteristic Polynomial, Real zeros, Diagonalizable Eigenvector

HOW DO I DEFINE THIS PROBLEM. This is roughly one of the standard computation strategies for this problem, but layed out for analysis purposes.

**Algorithm 3.37.** *Given a n by m matrix A with $m \geq n$ . (step 1) Compute $B = AA^\dagger$ . (step 2) Compute the characteristic polynomial of B to get p (step 3) Compute the zeros of p to get $\sigma^2$ ( CITATION NEEDED) (step 4) Compute the eigenvectors of the Diagonalizable matrix B using $\sigma^2$ to get U (step 5) Compute the other singular vectors by multiplying $U^\dagger A$ and normalizing the vectors but keeping track of the norm to get $\Sigma V^{`\dagger}$ (Step 6) Compute the remaining singular vectors of $V^\dagger$ via the nullspace(A) to now have $A = U\Sigma V^\dagger$.*

Correctness FEELS TRIVIAL.

RUNTIME work $(m/n)MM_{work}(n)+CP_{work}(n)+Zeros_{work}(n)+EV_{work}(n)+N_{work}(m,n)$ depth $MM_{depth}(n)+CP_{depth}(n)+Zeros_{depth}(n)+EV_{depth}(n)+N_{depth}(m,n)$ STABILITY ???

**Remark 3.38.** *A future draft should analyze the needed numerical stability in terms of changing an equality test to a*

# 4 Other reduction

Put pseudoinverse to basis and inverse reduction here, and discuss polynomial interpolation.

# 5 Conclusion

This paper is a rough draft; therefore no conclusion