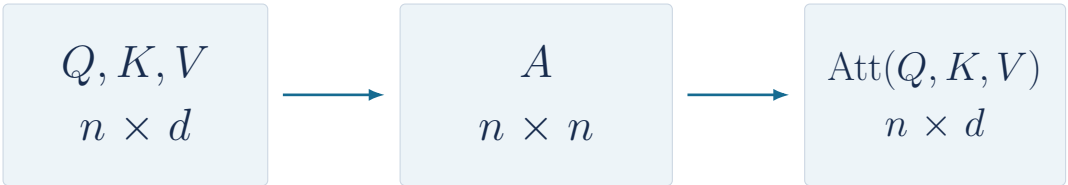
Fast Attention Requires Bounded Entries

When can we compute attention without forming the $n \times n$ matrix?

Josh Alman and Zhao Song, NeurIPS 2023

Teaching path: definitions, upper bound, lower bound

The main question



The input and output are thin. The usual intermediate matrix is quadratic.

Paper Sections 1 and 1.1.

Exact attention

$$M = \frac{QK^\top}{d}$$

$$A = \exp(M), \qquad A_{ij} = e^{M_{ij}}$$

$$D = \mathrm{diag}(A\mathbf{1}_n)$$

$$Y = \mathrm{Att}(Q, K, V) = D^{-1}AV$$

$\exp(M)$ means entry-wise exponentiation.

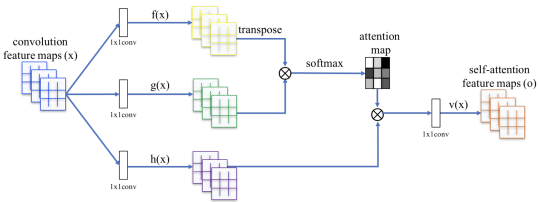
Attention mechanism

Matrix view of the same mechanism.

$$Q, K \rightarrow M = \frac{QK^\top}{d}$$

$$A = e^M \rightarrow D^{-1}A$$

$$Y = D^{-1}AV \leftarrow V$$



Scores choose the weights. Values supply the content being averaged.

Definition 1.1 and user-provided attention mechanism figure.

Why exact A loses low rank

$$M = \frac{QK^\top}{d} \in \mathbb{R}^{n\times n}, \qquad \operatorname{rank}(M) \le d \ll n$$
$$A = \exp(M), \qquad A_{ij} = e^{M_{ij}}$$

Entry-wise exponentiation can make exact A full rank.

$A = U_1U_2^\top, \quad U_1, U_2 \in \mathbb{R}^{n\times r}, \quad r < n$ is not guaranteed

Therefore the proof introduces a low-rank approximation to the exponential: $\tilde{A} = P(M) \approx \exp(M)$.

One score entry

$$M_{ij} = \frac{\langle q_i, k_j \rangle}{d}$$

$$M_{ij} = \frac{1}{d} \sum_{\ell=1}^{d} Q_{i\ell} K_{j\ell}$$

i and j choose rows. The index ℓ runs over feature coordinates.

Bounded entries give bounded scores

$$\|Q\|_\infty \le B, \qquad \|K\|_\infty \le B, \qquad \|V\|_\infty \le B$$

equivalently: $|Q_{i\ell}|,\ |K_{j\ell}|,\ |V_{j\ell}| \le B$ for all valid indices

$$|M_{ij}| \le \frac{1}{d}\sum_{\ell=1}^{d} |Q_{i\ell}K_{j\ell}| \le \frac{dB^2}{d} = B^2$$

$$M_{ij} \in [-B^2, B^2]$$

Upper-bound theorem setup

Conditions are assumptions. Targets are what the algorithm must achieve.

CONDITIONS ON THE INPUT

$$d = O(\log n)$$

$$\|Q\|_\infty, \|K\|_\infty, \|V\|_\infty \leq B$$

$$B = o(\sqrt{\log n})$$

$$\epsilon_a = 1/\operatorname{poly}(n)$$

TARGET TO PROVE

Output a matrix T.

$$\|T - Y\|_\infty \leq \epsilon_a$$

runtime $= n^{1+o(1)}$

The upper-bound proof constructs the algorithm under these conditions.

Theorem 1.3, separated into assumptions and guarantees.

Approximate attention

The algorithm outputs T, not a new attention definition.

$$\|T - Y\|_\infty \leq \epsilon_a$$

$$\epsilon_a = 1/\operatorname{poly}(n)$$

$$\epsilon_a = \frac{1}{n^c} = n^{-c}$$

$\|\cdot\|_\infty$ is the largest entry-wise absolute error.

Approximate the exponential only where needed

The polynomial only needs to work on the score interval.

$$P(x) \approx e^x \qquad x \in [-B^2, B^2]$$

$$\widetilde{A} = P(M), \qquad \widetilde{A}_{ij} = P(M_{ij})$$

This is entry-wise. It is not the matrix product power M^2.

Corollary 2.2 and Lemma 3.4.

Two epsilons

INTERNAL ERROR

$$|\widetilde{A}_{ij} - A_{ij}| \leq \epsilon_A A_{ij}$$

FINAL ERROR

$$\|T - Y\|_\infty \leq \epsilon_a$$

We choose ϵ_A small enough to guarantee the requested ϵ_a.

Corollary 2.2.

How g and r are chosen

g is the polynomial degree needed to approximate e^x on $[-B^2, B^2]$.

$$g = O\left(\max\left\{\frac{\log(1/\epsilon_A)}{\log(\log(1/\epsilon_A)/B^2)},\ B^2\right\}\right)$$

r is the number of lifted polynomial features up to degree g.

$$r \leq \binom{2(d+g)}{2g}$$

Small g gives small r, and small r gives fast multiplication by $\widetilde{A}$.

Corollary 2.2 and Lemma 3.2.

Low-degree polynomial means low-rank features

$$P(x) = c_0 + c_1 x + \cdots + c_g x^g$$

$$P\left(\frac{\langle q_i, k_j \rangle}{d}\right) = \langle u_i, v_j \rangle$$

$$\widetilde{A} = U_1 U_2^\top$$

$$U_1, U_2 \in \mathbb{R}^{n \times r}, \qquad r \leq \binom{2(d+g)}{2g}$$

Lemma 3.2 and Appendix A.1.

Use the factors without forming $\widetilde{A}$

$$\begin{aligned} \widetilde{A}V &= U_1(U_2^\top V) \\ \widetilde{A}\mathbf{1}_n &= U_1(U_2^\top \mathbf{1}_n) \\ T &= \widetilde{D}^{-1}U_1(U_2^\top V) \end{aligned}$$

$\widetilde{D} = \mathrm{diag}(U_1(U_2^\top \mathbf{1}_n))$.

Algorithm 1.

Row sums keep the same relative error

$$|\widetilde{D}_{ii} - D_{ii}| = \left|\sum_j (\widetilde{A}_{ij} - A_{ij})\right|$$

$$|\widetilde{D}_{ii} - D_{ii}| \leq \sum_j \epsilon_A A_{ij} = \epsilon_A D_{ii}$$

The denominator is stable because the error is relative to each row's own mass.

Lemma 3.5.

Final error after normalization

$$T - Y = \widetilde{D}^{-1}\widetilde{A}V - D^{-1}AV$$

$$T - Y = (\widetilde{D}^{-1}\widetilde{A}V - D^{-1}\widetilde{A}V) + (D^{-1}\widetilde{A}V - D^{-1}AV)$$

$$\|T - Y\|_\infty \leq \|\widetilde{D}^{-1}\widetilde{A}V - D^{-1}\widetilde{A}V\|_\infty + \|D^{-1}(\widetilde{A} - A)V\|_\infty$$

denominator error

$$\|\widetilde{D}^{-1}\widetilde{A}V - D^{-1}\widetilde{A}V\|_\infty \leq \epsilon_D B$$

numerator error

$$\|D^{-1}(\widetilde{A} - A)V\|_\infty \leq \epsilon_A B$$

Lemma 3.6 and Appendix A.3.

Choose the internal error

$$\|\widetilde{D}^{-1}\widetilde{A}V - D^{-1}AV\|_\infty \leq (\epsilon_A + \epsilon_D)B$$

$$\epsilon_D = \epsilon_A \implies \|T - Y\|_\infty \leq 2\epsilon_A B$$

$$\epsilon_A = \frac{\epsilon_a}{2B}$$

$$\|T - Y\|_\infty \leq \epsilon_a$$

Lemma 3.6 and Appendix A.3.

Why small B gives almost-linear time

$$d = O(\log n), \quad B = o(\sqrt{\log n}), \quad \epsilon_a = 1/\operatorname{poly}(n)$$

$$g = o(\log n)$$

$$r = n^{o(1)}$$

$$\text{time} = n^{1+o(1)}$$

Theorem 3.8.

Upper-bound proof path

1. Bounded entries force scores into $[-B^2, B^2]$.
2. A low-degree polynomial approximates e^x on that interval.
3. The polynomial gives $\widetilde{A} = U_1 U_2^\top$.
4. Relative error survives row normalization.
5. The feature count is $n^{o(1)}$ when $B = o(\sqrt{\log n})$.

Summary of Section 3.

What hardness means here

A hardness result says a fast algorithm would break a widely believed assumption.

Assume fast attention exists.
Use it to solve a known hard problem.

This paper uses SETH through approximate nearest neighbor.

Imported theorem: SETH

SETH is not about attention.

It says SAT cannot be solved much faster than brute force.

$$\forall \delta > 0, \exists k :$$
$$k\text{-SAT} \notin O(2^{(1-\delta)n})$$

For this presentation: SETH is the starting hardness assumption.



Journal of Computer and System Sciences 62, 367–375 (2001)
doi:10.1006/jcss.2000.1727, available online at http://www.idealibrary.com on IDEAL

On the Complexity of k-SAT

Russell Impagliazzo[1] and Ramamohan Paturi[2]

University of California San Diego, La Jolla, California

Received June 22, 1998; revised June 8, 2000
published online January 31, 2001

The k-(s)SAT problem is to determine if a given k-CNF has a satisfying assignment. It is a celebrated open question as to whether it requires exponential time to solve k-SAT for $k \geq 3$. Here exponential time means 2^{cn} for some $c > 0$. In this paper, we define $s_k = \inf\{\delta : k\text{-SAT} \in \text{DTIME}(2^{\delta n})\}$. Using complexity, we show that the complexity of k-SAT increases as k increases. More precisely, for $k \geq 3$, define $c_k = \inf\{\delta$ there exists $2^{\delta n}$ algorithm for solving k-SAT$\}$. Then $c_k < c_{k+1}$ unless $c_k = 1$. Similarly, $s_k < s_{k+1}$ unless $s_k = 1$. For $k \geq 4$, $s_k > 0$. In this paper, we show that s_k is increasing infinitely often assuming ETH(or k-SAT). Let c_∞ be the limit of c_k. We will in fact show that $c_\infty = \inf\{\delta$ there exists $2^{\delta n}$ algorithm for k-SAT for all $k\}$. We also combine together the ideas of critical clauses and the Sparsification Lemma to define the satisfiability of a k-CNF to the satisfiability of a disjunction of $2^{\varepsilon n}$ k-CNFs in those variables for some $k \geq 3$ and arbitrarily small $\varepsilon > 0$. We also show that such a disjunction can be computed in time $2^{\varepsilon n}$ for arbitrarily small $\varepsilon > 0$. — 1998 Academic Press

Although all NP-complete problems are equivalent as far as the existence of polynomial-time algorithms is concerned, there is wide variation in the worst-case complexity of known algorithms for these problems. For example, there have been several algorithms for maximum independent set [6, 13, 15, 16], and the best of these takes time 1.2190^n in the worst-case [12]. Recently, a 3-coloring algorithm has been announced which takes time 1.3289^n [11]. On the other hand, 3-SAT and 4-coloring can be solved in 2.4423^n time [3]. However, it is not known what, if any, relationships exist among the worst-case complexities of various problems. In this paper, we examine the complexity of k-SAT, and derive a relationship that governs

[1] This research is supported by NSF Grant CCR-9734911 from the Theory of Computing Program and INT-9600919/ME103 of the NSF and Czech Republic of Education.
[2] This research is supported by NSF Grant CCR-9734911 from the Theory of Computing Program.

367

0022-0000/01 $35.00
Copyright © 2001 by Academic Press
All rights of reproduction in any form reserved.

Impagliazzo and Paturi, Journal of Computer and System Sciences, 2001.

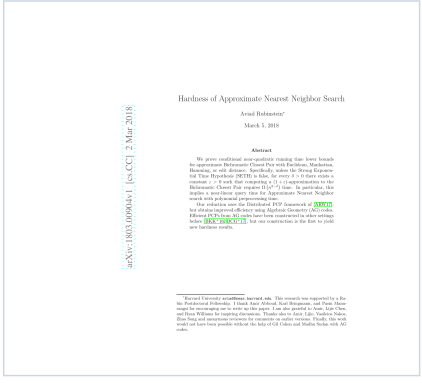
Imported theorem: SAT to approximate NN

Prior work gives this black box:

$$\text{SETH} \implies$$
$$\text{ANN needs } n^{2-o(1)} \text{ time}$$

For $d = C \log n$, gap approximate Hamming nearest neighbor is hard.

So this paper can start from ANN instead of redoing the SAT reduction.



arXiv:1803.00904v1 [cs.CC] 2 Mar 2018

Hardness of Approximate Nearest Neighbor Search

Aviad Rubinstein*

March 5, 2018

Abstract

We prove conditional near-quadratic running time lower bounds for approximate Bichromatic Closest Pair with Euclidean, Manhattan, Hamming, or edit distance. Specifically, unless the Strong Exponential Time Hypothesis (SETH) is false, for every $\delta > 0$ there exists a constant $\varepsilon > 0$ such that computing a $(1+\varepsilon)$-approximation to the Bichromatic Closest Pair requires $\Omega(n^{2-\delta})$ time. In particular, this implies a near-linear query time for Approximate Nearest Neighbor search with polynomial preprocessing time.

Our reduction uses the Orthogonal Vectors Conjecture (OVC) as a starting point, which is implied by SETH. We use the framework of [AWW15], but obtain improved efficiency using Algebraic Geometry (AG) codes. Efficient PCPs from AG codes have been constructed in other settings before [BKKS+16, BCGT13], but our construction is the first to yield new hardness results.

*Harvard University **aviad@mit.harvard.edu**. This research was supported by a Rothschild Postdoctoral Fellowship, and by NSF Grant CCF-1408635. Thanks to Oded Regev for encouraging me to write up this paper. I am also grateful to Amit Lahiri, Yuzhou Gu, and Ryan Williams for inspiring discussions. Thanks also to Arnab Bhattacharyya, Zhao Song and anonymous reviewers for comments on earlier versions. Finally, this work would not have been possible without the help of Gil Cohen and Madhu Sudan with AG codes.

Rubinstein, STOC 2018; cited as the approximate nearest-neighbor hardness input.

This paper: approximate NN to attention

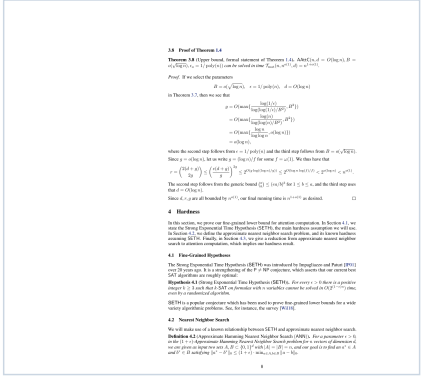
This paper adds the second translation:

fast attention $\Longrightarrow$ fast approximate NN

Build Q, K, V from binary vectors.

Close pairs become larger attention weights.

Direction: nearest neighbor is encoded inside attention.



3.8 Proof of Theorem 1.4

Theorem 3.8 (Upper bound, formal statement of Theorem 1.4)**.** *After* $\Pi_{A,B} = O(\log n)$*,* $B = \alpha\sqrt{\log(n)}$*,* $\varepsilon_a = 1$ *and* $\alpha(\min(a))$ *can be solved in time* $T_{\mathrm{mult}}(n, n^{o(1)}, d) = n^{1+o(1)}$*.*

Proof. If we select the parameters

$$B = a\sqrt{\log n}, \quad \epsilon = 1/\mathrm{poly}(n), \quad d = O(\log n)$$

in Theorem 3.7, then we see that

$$\begin{aligned} p &= O(\max(\frac{\log(1/\epsilon)}{\log(\log(1/\epsilon)/B^2)}, B^2)) \\ &= O(\max(\frac{\log(n)}{\log(\log(n)/B^2)}, B^2)) \\ &= O(\max(\frac{\log n}{\log\log n}, a(\log n))) \\ &= a(\log n), \end{aligned}$$

where the second step follows from $\epsilon = 1/\mathrm{poly}(n)$ and the third step follows from $B = a\sqrt{\log(n)}$. Since $q = o(\log n)$, let us write $q = (\log n)/f$ for some $f = \omega(1)$. We thus have that

$$r = \left(\frac{Bd+q}{2p}\right)^2 \leq \left(\frac{a(d+q)}{2}\right)^2 \leq g^2(a+ba(1/n)) \leq g^2(\log n + ba(1/n) \leq g^2(\log n \cdot \omega(1)) < g^2(\log n \cdot \omega(1)).$$

The second step follows from the generic bound $\binom{n}{k} \leq (en/k)^k$ for $1 \leq k \leq n$, and the third step uses that $d = O(\log n)$.

Since δ, r, p are all bounded by $n^{o(1)}$, our final running time is $n^{1+o(1)}$ as desired. □

4 Hardness

In this section, we prove our fine-grained lower bound for attention computation. In Section 4.1, we state the Strong Exponential Time Hypothesis (SETH), the main hardness assumption we will use. In Section 4.2, we define the approximate nearest neighbor search problem, and its known hardness assuming SETH. Finally, in Section 4.3, we give a reduction from approximate nearest neighbor search to attention computation, which implies our hardness result.

4.1 Fine-Grained Hypothesis

The Strong Exponential Time Hypothesis (SETH) was introduced by Impagliazzo and Paturi [IP01] over 20 years ago. It is a strengthening of the $P \neq NP$ conjecture, which asserts that our current best SAT algorithms are roughly optimal.

Hypothesis 4.1 (Strong Exponential Time Hypothesis (SETH))**.** *For every* $\varepsilon > 0$ *there is a positive integer* $k \geq 3$ *such that* k*-SAT on formulas with* n *variables cannot be solved in* $O(2^{(1-\varepsilon)n})$ *time, even on a randomized algorithm.*

SETH is a popular conjecture which has been used to prove fine-grained lower bounds for a wide variety algorithmic problems. See, for instance, the survey [Vas18].

4.2 Nearest Neighbor Search

We will make use of a known relationship between SETH and approximate nearest neighbor search.

Definition 4.2 (Approximate Harmming Nearest Neighbor Search (ANN))**.** *For a parameter* $c > 0$ *(the "approximation factor"), parameters* $d = d(n)$ *and* $r = r(n)$*, and a dimension* $d = d(n)$*, the problem* c*-ANN can given as input two sets* $A, B \subset \{0,1\}^d$ *with* $|A| = |B| = n$*, and (we need it to find an* $a^* \in A$ *and* $b^* \in B$ *satisfying* $\|a^* - b^*\|_0 \leq (1+\varepsilon) \cdot \min_{a \in A, b \in B} \|a - b\|_0$*.*

Alman and Song, Theorem 4.6 and Lemma 4.7.

How the chain becomes a lower bound

SETH $\Rightarrow$ approximate NN is hard

$\downarrow$

fast attention at $B = \Theta(\sqrt{\log n}) \Rightarrow$ fast approximate NN

$\downarrow$

fast attention would contradict SETH

So, assuming SETH, truly subquadratic attention is ruled out at this entry scale.

How the imported theorems combine with Theorem 4.6.

Approximate nearest neighbor

Input: two sets of binary vectors

$$A = \{a_1, \ldots, a_n\}, \qquad B = \{b_1, \ldots, b_n\}$$

$$a_i, b_j \quad \in \quad \{0,1\}^d$$

Goal: find vectors that are close in Hamming distance.

$$\|a_i - b_j\|_0 \quad = \quad \text{number of mismatches}$$

Definition 4.2.

The hard gap problem

NEAR CASE

$$\exists j \,:\, \|a_i - b_j\|_0 < t$$

FAR CASE

$$\forall j : \|a_i - b_j\|_0 \geq (1+\epsilon)t$$

Under SETH, this cannot be solved in truly subquadratic time for $d = C \log n$.

Definition 4.5 and Lemma 4.3.

Reduction direction

Nearest neighbor is encoded inside attention.

$$\text{fast attention} \quad \Longrightarrow \quad \text{fast nearest neighbor}$$

This is the reduction: turn each hard nearest-neighbor instance into an attention instance.
If attention were easy, the hard problem would become easy.

Section 1.2 and Lemma 4.7.

Close vectors become large scores

small Hamming distance

↓

large inner product $\langle q_i, k_j \rangle$

↓

large attention weight $e^{\langle q_i, k_j \rangle / d}$

The formal proof adds balancing gadgets so row normalization does not hide the signal.

Approximate output needs a robust signal

$$\epsilon_a \quad = \quad n^{-C_a}$$

If near and far cases differ by less than ϵ_a, the approximate solver may blur them.

$$\text{signal gap} \quad \gg \quad n^{-C_a}$$

The reduction therefore creates a polynomially robust separation.

Lower-bound intuition from Lemma 4.7.

A logarithmic score gap gives a polynomial ratio

$$\Delta = M_{\text{near}} - M_{\text{far}}$$

$$\frac{e^{M_{\text{near}}}}{e^{M_{\text{far}}}} = e^{\Delta}$$

$$\Delta = c \log n \quad \Longrightarrow \quad e^{\Delta} = n^{c}$$

Reversing the ratio gives n^{-c}, which can dominate n^{-C_a} when constants are chosen.

Exponential amplification in the lower-bound construction.

Why $B = \Theta(\sqrt{\log n})$

$$M_{ij} = \frac{\langle q_i, k_j \rangle}{d}$$

$$\text{entry size } B \implies \text{score scale } B^2$$

$$\Delta = \Theta(\log n)$$

$$B^2 = \Theta(\log n) \implies B = \Theta(\sqrt{\log n})$$

Parameter scale behind Theorem 4.6.

Normalization can hide the largest entry

$$Y_{i,:} = \frac{\sum_j A_{ij} V_{j,:}}{\sum_j A_{ij}}$$

A large A_{ij} matters only relative to the whole row sum.

The reduction adds dummy structure to control denominators.

That is the most technical part of the lower-bound proof.

Section 1.2 and Appendix B.

The formal lower-bound statement

Assuming SETH, for every fixed exponent saving $q > 0$, suitable constants exist such that

$$d = C \log n, \qquad B = C_b \sqrt{\log n}, \qquad \epsilon_a = n^{-C_a}$$

No $O(n^{2-q})$ algorithm solves all such instances.

The constants may depend on q.

Theorem 4.6 and Lemma 4.7.

Lower-bound proof path

1. Prior work: SETH makes approximate nearest neighbor hard.
2. Convert nearest-neighbor vectors into attention inputs.
3. Close vectors create larger attention weights.
4. Scale entries so the gap survives inverse-polynomial error.
5. Fast attention would solve nearest neighbor too fast.
6. That contradicts SETH.

Proof structure of Theorem 4.6.

The two results meet at the same scale

$$B = o(\sqrt{\log n})$$

Fast approximation exists

$$B = \Theta(\sqrt{\log n})$$

Truly subquadratic approximation is ruled out under SETH

Theorems 1.3 and 1.4.

Notation to defend

$O(f(n))$: at most a constant times $f(n)$.

$o(f(n))$: negligible compared with $f(n)$.

$\Omega(f(n))$: at least a constant times $f(n)$.

$\Theta(f(n))$: both $O(f(n))$ and $\Omega(f(n))$.

$n^{1+o(1)}$ is almost linear. $O(n^{2-q})$ is truly subquadratic.

Asymptotic notation used throughout the paper.

Likely questions

T is just the algorithm's approximate output.

$P(M)$ means applying P to each scalar entry.

ϵ_A is internal. ϵ_a is the final output tolerance.

Nearest neighbor reduces to attention, not the reverse.

The lower bound is conditional on SETH.

Common questions from the proof path.

Three things to remember

1. Small entries keep scores in a short interval.

2. A low-degree polynomial gives a low-rank attention approximation.

3. At the $\sqrt{\log n}$ scale, attention can encode hard nearest-neighbor instances.

Fast attention requires bounded entries.

Backup: the error split

$$\widetilde{D}^{-1}\widetilde{A}V - D^{-1}AV = (\widetilde{D}^{-1}\widetilde{A}V - D^{-1}\widetilde{A}V) + (D^{-1}\widetilde{A}V - D^{-1}AV)$$

$$\begin{aligned} \|\widetilde{D}^{-1}\widetilde{A}V \;-\; D^{-1}\widetilde{A}V\|_\infty &\;\leq\; \epsilon_D B \\ \|D^{-1}\widetilde{A}V \;-\; D^{-1}AV\|_\infty &\;\leq\; \epsilon_A B \end{aligned}$$

Paper Sections 3.5–3.6.