

The Polar Express Algorithm

Mikkel Paltorp (mpasc@dtu.dk), 10 October 2026

Department of Applied Mathematics and Computer Science, Technical University of Denmark

In this note we look at how the sign function can be approximated by compositions of polynomials, as in the Polar Express algorithm [1]. This simple idea has a number of applications in modern machine learning and optimization, as it can be used to approximate matrix functions using only matrix-matrix products, which are very efficient to compute on modern hardware such as GPUs. Applied to the singular values of a matrix, the approximation of the sign function gives the polar factor $UV^\top$. This is, e.g., the basis of the update of the Muon optimizer [2], which would otherwise require an SVD, an expensive computation that is not very GPU friendly. Muon is explained as steepest descent in the spectral norm in Which Way Is Down?¹ It is furthermore possible to approximate the ReLU function in a similar way, which can be used to project matrices onto the cone of positive semidefinite matrices.

Contents

1. The Singular Value Transformation
 2. Approximating the Polar Normalization
 3. The Polar Express Algorithm
 4. Projection onto the semidefinite cone
- Bibliography

¹ Later post: *Which Way Is Down? First-Order Optimizers as a Choice of Norm.*
mipals.github.io/blog/2026/10-13-steepest-descent/

1. The Singular Value Transformation

The singular value transformation is defined for any odd function f , as

$$f(M) := Uf(\Sigma)V^\top, \quad M = U\Sigma V^\top. \quad (1)$$

When M is square and symmetric, the singular value transformation of an odd f coincides with the matrix function $f(M) = Qf(\Lambda)Q^\top$, which applies f to the eigenvalues. Writing $M = Q\Lambda Q^\top = (Q \operatorname{sign}(\Lambda)) |\Lambda| Q^\top$ shows that $U = Q \operatorname{sign}(\Lambda)$, $\Sigma = |\Lambda|$ and $V = Q$, and since f is odd,

$$f(M) = Qf(\Lambda)Q^\top = Q \operatorname{sign}(\Lambda)f(|\Lambda|)Q^\top = Uf(\Sigma)V^\top, \quad (2)$$

where the sign function is defined as the odd function

$$\operatorname{sign}(x) := \begin{cases} 1, & x > 0, \\ 0, & x = 0, \\ -1, & x < 0, \end{cases} \quad (3)$$

2. Approximating the Polar Normalization

Polar normalization corresponds to the singular value transformation when f itself is the sign function, i.e.

$$\operatorname{polar}(M) = U \operatorname{diag}(\operatorname{sign}(\sigma_1), \dots, \operatorname{sign}(\sigma_k))V^\top = UV^\top, \quad (4)$$

where the last equality takes U and V from the compact SVD, which keeps only the singular vectors of the nonzero singular values. Zero singular values stay zero, as $\operatorname{sign}(0) = 0$.

Computed directly, the polar factor requires an SVD, which is computationally expensive and not very GPU friendly (it is primarily memory bound rather than compute bound). The Polar Express algorithm instead approximates the matrix sign function using just matrix-matrix products [1]. The key idea is rather simple: Approximate the function f using a composition of polynomials, e.g.,

$$f(x) \approx p_k(p_{k-1}(\dots p_2(p_1(x))\dots)). \quad (5)$$

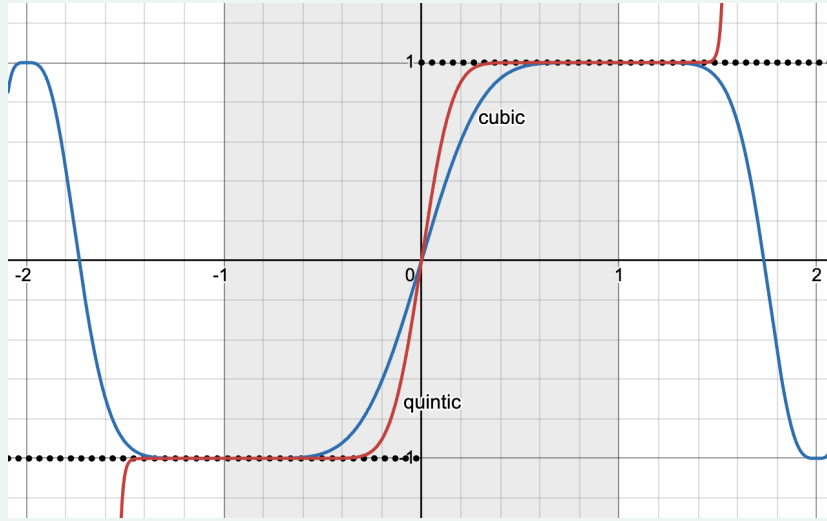
Since the above is a composition of polynomials, it can be evaluated using just additions and multiplications.

Example 1: Newton–Schulz

Example 1 (Newton–Schulz) The idea of approximating functions using repeated polynomial application is not new. An example is the Newton–Schulz algorithm which is based on repeatedly applying the same polynomial, i.e. $p_1 = p_2 = \dots = p_k$, in order to approximate the sign function. For order 3 and 5 the polynomials look as follows

$$p(x) = \frac{3}{2}x - \frac{1}{2}x^3, \quad p(x) = (15x - 10x^3 + 3x^5)/8. \quad (6)$$

While the Newton–Schulz method converges for all $x \in [-1, 1]$, the convergence is only fast for values close to ± 1 . Conversely, for values that are close to zero the convergence is slow. The slow convergence is exactly what the Polar Express method explained shortly aims to fix. The graph shows the cubic and quintic polynomials composed k times, together with the sign function (dotted). The shaded interval $[-1, 1]$ is where they approximate it.



Interactive version on the website →

A note here is that for non-square matrices we cannot simply do M^n . Instead, what is done is that $M^{2n+1} = M(M^\top M)^n = (MM^\top)^n M$.

3. The Polar Express Algorithm

The basic question that the Polar Express algorithm aims to solve is as follows: Given a set of T polynomials $\{p_i(x) = a_1^i x + a_3^i x^3 + a_5^i x^5\}_{i=1}^T$, can we find an optimal set of coefficients that minimizes the maximum error over a given interval $[\ell, u]$? Define their composition as $p = p_T \circ p_{T-1} \circ \dots \circ p_1$. We then solve

$$(p_1^*, \dots, p_T^*) = \arg \min_{p_1, \dots, p_T} \max_{\substack{M \in \mathbb{R}^{m \times n} \\ \sigma(M) \subset [\ell, u]}} \| \text{polar}(M) - p(M) \|_2 \quad (7)$$

subject to $p_t \in \mathbb{R}_5^{\text{odd}}[x]$, $t = 1, \dots, T$.

Here $\mathbb{R}_d^{\text{odd}}[x]$ denotes the space of real odd polynomials of degree at most d . Since p is odd, $p(M) = Up(\Sigma)V^\top$, so the error is $\max_i |1 - p(\sigma_i)|$, and the problem is the scalar one of approximating 1 on $[\ell, u]$, or by oddness the sign function on $[-u, -\ell] \cup [\ell, u]$.

In [1] they show that the above can be solved by a greedy approach, with a caveat that the error bound only holds if the smallest singular value is at least ℓ , which we do not know in advance. Smaller singular values still converge, but more slowly. A comment here is that the main focus in the paper is on bfloat16. Here the spacing between representable numbers at 1 is $2^{-7} = 0.0078125$, so singular values much smaller than this are at the level of rounding errors. They therefore suggest setting $\ell = 10^{-3}$, and $u = 1$ after scaling M by its Frobenius norm so that $\sigma_1 \leq 1$. In addition, they provide a few modifications in order to make the method numerically robust. These additions are skipped here, but can be found in the original paper.

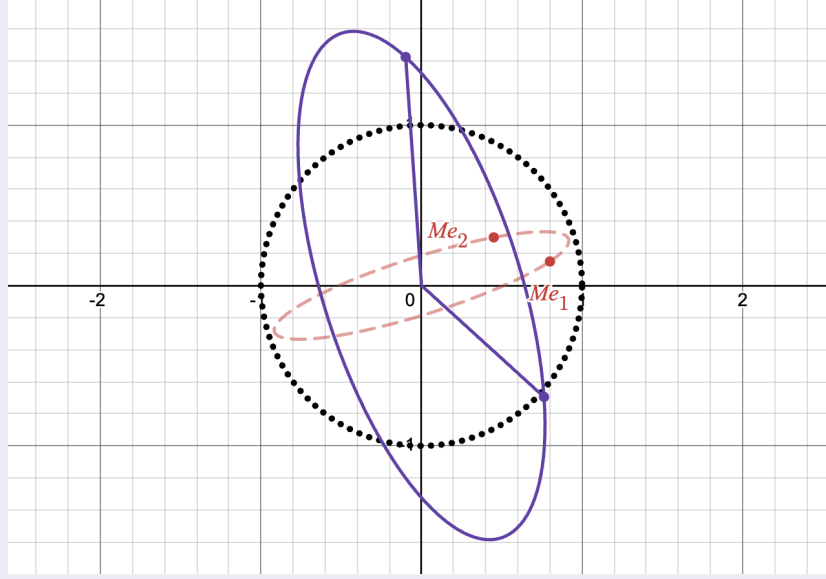
Interactive figure 1: Polar Express against Newton–Schulz

Interactive figure 1 (Polar Express against Newton–Schulz) The graph compares the first k Polar Express polynomials for $[\ell, u] = [10^{-3}, 1]$, composed, with the Newton–Schulz quintic composed k times, drawn as in Example 1. Newton–Schulz approaches ± 1 slowly from below, while Polar Express first lifts the small values and then oscillates around ± 1 , with every step shrinking the oscillation. The oscillation is the price for the smallest worst-case error. By the equioscillation theorem, the polynomial with the smallest maximum error on an interval reaches that error several times with alternating sign [3], so each optimal step has to oscillate around ± 1 . Newton–Schulz instead matches the sign function closely at ± 1 , where $p'(1) = 0$, which makes it accurate close to ± 1 but slow close to zero. After a few steps the oscillations become too dense to be told apart, and soon after that too small to be seen. The readout gives both compositions at ℓ , which is too close to zero to be seen in the graph.



Interactive version on the website →

Interactive figure 2 (Polar Express on a matrix) The same polynomials applied to a 2×2 matrix M . A matrix maps the unit circle (dotted) to an ellipse whose semi-axes are its singular values. The dashed ellipse belongs to M , scaled so that $\sigma_1 \leq 1$,² and its columns (the two points) can be dragged. The solid ellipse and arrows show the matrix after k steps. Each step applies the next polynomial to the singular values, so the ellipse turns into the unit circle, which is the ellipse of $UV^\top$.



Interactive version on the website →

² The scaling is a requirement: the polynomials are designed for singular values in $[\ell, 1]$, the positive half of $[-1, 1]$ in the previous figure, and larger singular values need not converge. Dividing M by its Frobenius norm guarantees $\sigma_1 \leq 1$, as the Frobenius norm is at least the largest singular value.

4. Projection onto the semidefinite cone

The positive semidefinite cone is the set of all real symmetric $n \times n$ positive semidefinite matrices. That is

$$\mathbb{S}_+^n := \{X \in \mathbb{R}^{n \times n} \mid X = X^\top, X \succeq 0\}. \quad (8)$$

Semidefinite programming (SDP) problems are optimization problems for which matrix variables are constrained to be positive semidefinite, a special case of the conic programs in Conic Optimization³. As such, algorithms that aim to solve SDPs need to ensure that the solution satisfies the positive semidefinite constraint. In interior point methods this is done by only taking steps for which the next iterate remains inside the cone, while first-order methods most often rely on projecting back onto the set after taking a step, i.e., for a symmetric $X = U\Lambda U^\top$,

$$\Pi_{\mathbb{S}_+^n}(X) = \arg \min_{Y \in \mathbb{S}_+^n} \|Y - X\|_F = U \operatorname{diag}(\operatorname{ReLU}(\lambda_1), \dots, \operatorname{ReLU}(\lambda_n))U^\top. \quad (9)$$

While the idea of projecting back onto the semidefinite cone using just matrix-matrix products is not new (a similar fixed-point approach was suggested in [4]), the main takeaway is that the ideas of Polar Express can be used to find a faster converging method [5]. An initial idea would be to directly find polynomials that approximate the ReLU function, i.e., solving

³ Earlier post: *Conic Optimization*.
mipals.github.io/blog/2026/01-04-conic-optimization/

$$\begin{aligned} & \inf_{f_1, \dots, f_T} \max_{x \in [-1, 1]} |f_T \circ f_{T-1} \circ \dots \circ f_1(x) - \text{ReLU}(x)| \\ \text{subject to } & f_t \in \mathbb{R}_{d_t}[x], t = 1, \dots, T, \end{aligned} \quad (10)$$

where T is the total number of polynomials in the composition, d_t is the degree of the t 'th polynomial, and the degrees are fixed in advance as part of a “matrix-matrix” multiplication budget. Unfortunately, directly solving the above is challenging, and instead a two-stage approach is proposed. The idea is to realize that we can write the ReLU function as

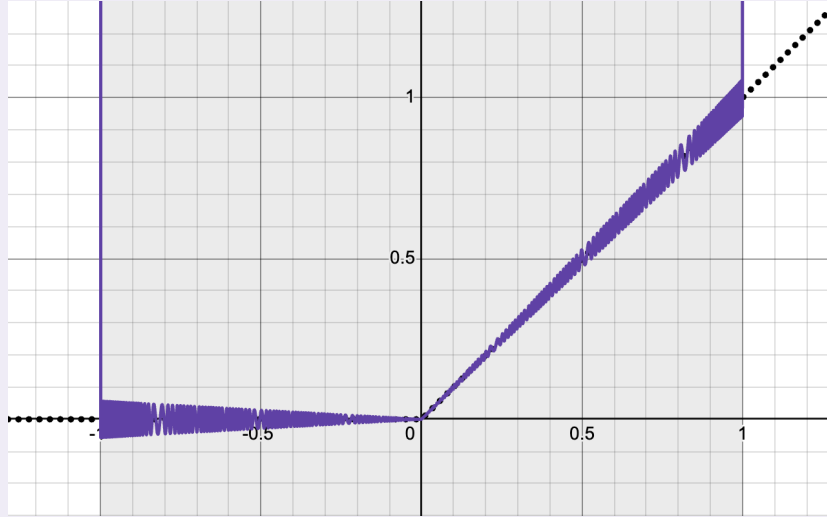
$$\text{ReLU}(x) = \frac{1}{2}x(1 + \text{sign}(x)). \quad (11)$$

The first step is to compute optimal coefficients for approximating the sign function using the Polar Express approach. The second step is then to refine the coefficients of the polynomials found in the first step by minimizing the error to the ReLU function on the entire interval $[-1, 1]$, i.e., minimizing the following loss function

$$L(f_T, \dots, f_1) := \max_{x \in [-1, 1]} \left| \frac{1}{2}x(1 + f_T \circ f_{T-1} \circ \dots \circ f_1(x)) - \text{ReLU}(x) \right|. \quad (12)$$

Interactive figure 3: From the sign function to the ReLU

Interactive figure 3 (From the sign function to the ReLU) The graph shows $\frac{1}{2}x(1 + p(x))$, with p the Polar Express composition from the previous figure, together with the ReLU (dotted). The shaded interval $[-1, 1]$ is where it approximates the ReLU. The oscillation is inherited from p , and the factor x damps the error of p where it is largest, close to zero.

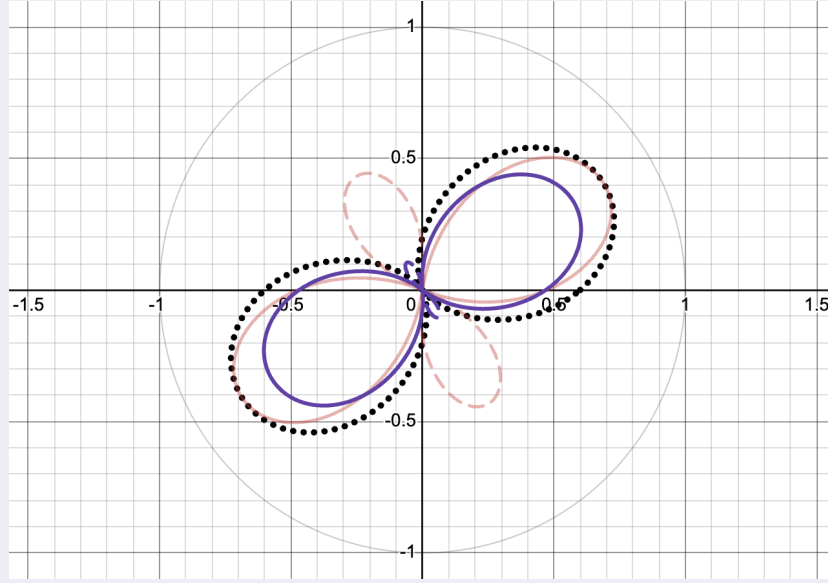


Interactive version on the website →

Interactive figure 4: Projecting a matrix onto the semidefinite cone

Interactive figure 4 (Projecting a matrix onto the semidefinite cone) The same idea for a symmetric 2×2 matrix A with eigenvalues λ_1, λ_2 and an eigenvector at angle φ . For each unit vector u the graph draws a point at distance $|u^\top A u|$ in the direction of u , solid where $u^\top A u > 0$ and dashed where it is negative, so the lobes point along the eigenvectors and have lengths $|\lambda_1|$ and $|\lambda_2|$. A is positive semidefinite exactly

when nothing is dashed.⁴ The faint lobes are A , the dotted curve its exact projection, and the bold curve the approximation $\frac{1}{2}A(I + p(A))$ after k steps. As k grows, the dashed lobes vanish and the solid ones grow slightly, since the negative eigenvalue no longer subtracts from them.



Interactive version on the website →

⁴ If A is already positive semidefinite, the curve first moves away from A and then returns. The first polynomials are tuned for singular values near $\ell = 10^{-3}$ and temporarily push values close to 1 far off.

Bibliography

- [1] N. Amsel, D. Persson, C. Musco, and R. M. Gower, “The Polar Express: Optimal Matrix Sign Methods and Their Application to the Muon Algorithm.” [Online]. Available: <https://arxiv.org/abs/2505.16932>
- [2] K. Jordan *et al.*, “Muon: An optimizer for hidden layers in neural networks.” [Online]. Available: <https://kellerjordan.github.io/posts/muon/>
- [3] L. N. Trefethen, *Approximation Theory and Approximation Practice, Extended Edition*. Society for Industrial, Applied Mathematics, 2019. doi: 10.1137/1.9781611975949.
- [4] J. B. Francisco and D. S. Gonçalves, “A fixed-point method for approximate projection onto the positive semidefinite cone,” *Linear Algebra and its Applications*, vol. 523, pp. 59–78, 2017, doi: 10.1016/j.laa.2017.02.014.
- [5] S. Kang, H. Han, A. Groudiev, and H. Yang, “Factorization-free Orthogonal Projection onto the Positive Semidefinite Cone with Composite Polynomial Filtering.” [Online]. Available: <https://arxiv.org/abs/2507.09165>