

September 25, 2026

Today:

- More parallel linear algebra
- Faster parallel algo for resultant

Last time: parallel algos for

- determinant
- inverse
- char polynomial

Today we'll extend these to matrix rank and solving $Ax = b$ when A is not square.

We'll also see a faster parallel algorithm to compute resultants

$A \in \mathbb{C}^{n \times n}$ matrix.

$$p_A(t) = \prod_i (t - \lambda_i) = \text{char poly of } A$$

$$\begin{aligned} \text{rank}(A) &= \dim \text{im}(A) \\ \text{nullity}(A) &= \dim \ker(A). \end{aligned}$$

$$\text{Rank-nullity thm: } \text{rank}(A) + \text{nullity}(A) = n$$

$$\text{Suppose } p_A(t) = t^n + c_{n-1}t^{n-1} + \dots + c_m t^m$$

Then zero is an eigenvalue of algebraic multiplicity m . Can we conclude $\text{nullity}(A) = m$?

No! Jordan blocks can be an issue

e.g.

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \text{ has rank } 1 \\ \text{but } p_A(t) = t^2$$

Def $A \in \mathbb{C}^{n \times n}$ is Hermitian if $A = A^*$

$\uparrow$ conjugate
transpose

Theorem (Spectral thm)

If A Hermitian, then $\mathbb{C}^n$ has an orthonormal basis of eigenvectors of A , i.e., A is diagonalizable.

Lemma A diagonalizable

$\Leftrightarrow$ alg multiplicity = geom multiplicity
for eigenvalues

$\Rightarrow$ nullity(A) = multiplicity of
0 as eigenvalue.

How to reduce to Hermitian case?

Lemma.

$$\text{rank} \begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix} = 2 \text{rank}(A).$$

Thm Can compute $\text{rank}(A)$ in $O(\log^2 n)$ parallel time.

Proof Let $B = \begin{pmatrix} 0 & A \\ A^* & 0 \end{pmatrix}$.

- Compute $p_B(t) = t^{2n} + c_{2n-1}t^{2n-1} + \dots + c_m t^m$
- Return $\text{rank}(A) = \frac{2n - m}{2}$

Complexity: clear.

Correctness:

$$\begin{aligned} B \text{ Hermitian} &\Rightarrow m = \text{nullity}(B) \\ &\Rightarrow 2n - m = \text{rank}(B) \end{aligned}$$

$$\text{rank}(A) = \frac{1}{2} \text{rank}(B) = \frac{2n - m}{2} . \quad \square$$

Next, we'll compute a basis for the row space (resp. column space) of A .

Lemma Can compute basis for row space of A in $O(\log^2 n)$ parallel time.

Proof Pick the basis greedily.

Let $v_1, \dots, v_n$ = rows of A

$$S = \{i : \text{rank}(v_1, \dots, v_{i-1}) < \text{rank}(v_1, \dots, v_i)\}$$

Claim: $\{v_i : i \in S\}$ is a basis for row space of A .

Proof let $k \notin S$. Then v_k spanned by $\{v_i : i \in S, i < k\}$. $\square$

Can compute S in $O(\log^2 n)$ parallel time: for each i in parallel, decide if $i \in S$ using two rank computations $\square$

Next, we compute basis for $\ker(A)$.

Lemma. Can compute basis for $\ker(A)$ in $O(\log^2 n)$ parallel time.

Proof Using previous algo, find a maximal invertible submatrix of A .

Suppose $A = \begin{pmatrix} M & B \\ C & D \end{pmatrix}$, M maximal invertible submatrix.

Claim Columns of $\begin{pmatrix} M^{-1}B \\ -I \end{pmatrix}$ are basis for $\ker(A)$.

Proof Clearly $\dim(\ker(A))$ nonzero independent vectors. To see why they're in $\ker(A)$, write $C = \hat{C}M$, $D = \hat{C}B$.

$$\begin{pmatrix} M & B \\ C & D \end{pmatrix} \begin{pmatrix} M^{-1}B \\ -I \end{pmatrix} = \begin{pmatrix} B - B \\ \hat{C}B - \hat{C}B \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

□

Finally, we solve rectangular systems.

Thm Can solve $Ax = b$ in $O(\log^2 n)$ parallel time

Proof. Again write $A = \begin{pmatrix} M & B \\ C & D \end{pmatrix}$,
 M maximal invertible

Suffices to solve $(M \ B) x = \hat{b}$

If a solution exists, then there is one that is zero on vars corresponding to columns of B .

So compute $\hat{x} = M^{-1} \hat{b}$ & check if it is a solution.

If not, no solutions.

If so, all solutions lie in $\hat{x} + \ker(A)$ □

Can we compute determinants in $O(\log n)$ parallel time?

Don't know! Important open problem

For structured matrices, we can do better.

Thm Can compute $\text{res}(f, g)$ in $O(\log n)$ parallel time.

Fact (Poisson formula)

$$\text{If } f = \prod_{i=1}^n (x - a_i) \quad g = \prod_{i=1}^m (x - b_i)$$

$$\begin{aligned} \text{then } \text{res}(f, g) &= \prod_{i,j} (a_i - b_j) \\ &= \prod_i g(a_i) \\ &= (-1)^{nm} \prod_j f(b_j). \end{aligned}$$

We'll combine the Poisson formula with fast parallel algos for Newton's identities.

$$\begin{aligned} \text{let } f &= x^n + f_{n-1}x^{n-1} + \dots + f_1x + f_0 \\ &= \prod_{i=1}^n (x - a_i) \end{aligned}$$

$$p_d := a_1^d + \dots + a_n^d.$$

Lemma The following can be done in $O(\log n)$ parallel time

- Given $f_0, \dots, f_{n-1}$, compute $p_1, p_2, \dots, p_m$ for $m \leq \text{poly}(n)$.
- Given $p_1, \dots, p_n$, compute $f_0, \dots, f_{n-1}$.

Proof Homework 2. □

Assuming the lemma, let's compute $\text{res}(f, g)$.

Lemma. Given $f, g \in \mathbb{C}[x]$, can compute
 $g(a_1) + \dots + g(a_n)$

in $O(\log n)$ parallel time,
where $f = \prod_i (x - a_i)$.

Proof let $g = g_n x^n + g_{n-1} x^{n-1} + \dots$

$$\sum_{i=1}^n g(a_i) = \sum_{i=1}^n \sum_{j=0}^n g_j a_i^j$$

$$= \sum_{j=0}^n g_j \sum_{i=1}^n a_i^j$$

$$= \sum_{j=0}^n g_j \cdot p_j$$

$$p_j = a_1^j + \dots + a_n^j$$

Compute $p_1, \dots, p_n$ in parallel
using lemma.

Compute inner product with $g_0, \dots, g_n$
in $O(\log n)$ parallel time by adding
in binary tree. $\square$

Lemma. For $k \leq n$, can compute
 $g(a_1)^k + \dots + g(a_n)^k$
in $O(\log n)$ parallel time.

Proof By previous lemma, suffices to
compute coefficients of $g(x)^k$ in
parallel.

Use evaluation + interpolation:

- Evaluate $g(i)^k$ for $0 \leq i \leq n^2$
in parallel
- Interpolate coeffs of $g(x)^k$.

Eval: $O(\log n)^{\text{parallel}}$ ops to compute $g(i)$
+ $O(\log n)$ ops to compute k^{th}
power using repeated squaring

Interp: linear combination of $n^2 + 1$
evals, sum in $O(\log n)$
parallel steps. $\square$

Theorem Compute $\text{res}(f, g)$ in $O(\log n)$ parallel ops.

Proof

- For $k = 1, \dots, n$, compute $g(a_1)^k + \dots + g(a_n)^k$

These are power sums of the polynomial $\prod_i (x - g(a_i))$.

Constant term is $\pm \prod_i g(a_i) = \text{res}(f, g)$

- Use parallel algo for Girard-Newton to compute $\text{res}(f, g)$ $\square$

Next week:

- No Wed lecture
- Start fast lin alg Friday.