Algebraic Complexity Theory

Lecture 2: Circuits for the Determinant; Parallel computation of rank

Department of Computer Science,
Indian Institute of Science

Recap

- In the last lecture, we saw examples of problems wherein the output is a *polynomial* (*or a rational function*) in the input variables.
- Several of these problems involve computation of the *determinant* of a matrix.
- We also defined a natural model of computation, namely *arithmetic circuits* (a.k.a *straight-line programs*).

Circuits for the Determinant

The determinant

- Let $X = (x_{ij})_{i,j\in[n]}$. Then,

$$\mathrm{Det}_n := \det(X) = \sum_{\sigma\in S_n} (-1)^{\mathrm{sign}(\sigma)} \prod_{i\in[n]} x_{i\,\sigma(i)},$$

- Question. How fast can we compute Det_n?
- The above formula gives an $O(n^n)$-size, depth-2 circuit for Det_n. This circuit has only $+$ and $\times$ gates.

The determinant

- Let $X = (x_{ij})_{i,j \in [n]}$. Then,

$$Det_n := det(X) = \sum_{\sigma \in S_n} (-1)^{sign(\sigma)} \prod_{i \in [n]} x_{i\,\sigma(i)} ,$$

- Question. How fast can we compute Det_n?
- The above formula gives an $O(n^n)$-size, depth-2 circuit for Det_n. This circuit has only + and × gates.
- The classical *Gaussian elimination* method yields a circuit of size $O(n^3)$ and depth $O(n)$. But the circuit has +, ×, and ÷ gates. Also, division by 0 is forbidden!

The determinant

- Let $X = (x_{ij})_{i,j\in[n]}$. Then,

$$\mathrm{Det}_n := \det(X) = \sum_{\sigma\in S_n} (-1)^{\mathrm{sign}(\sigma)} \prod_{i\in[n]} x_{i\,\sigma(i)}\ ,$$

- Question. How fast can we compute Det_n?
- The above formula gives an $O(n^n)$-size, depth-2 circuit for Det_n. This circuit has only $+$ and $\times$ gates.
- The classical <u>*Gaussian elimination*</u> method yields a circuit of size $O(n^3)$ and depth $O(n)$. But the circuit has $+$, $\times$, and $\div$ gates. Also, division by 0 is forbidden!
- Question. Can we remove $\div$ gates? If yes, we can avoid division by 0.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $\mathrm{poly}(sd)$ that uses only $+$ and $\times$ gates.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size poly(sd) that uses only $+$ and $\times$ gates.

- Corollary. The n^2-variate, degree-n determinant polynomial Det_n is computable by a *poly(n)* size circuit having only $+$ and $\times$ gates.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $\mathrm{poly}(sd)$ that uses only $+$ and $\times$ gates.
- *Proof sketch.* Assume that every gate of C has fan-in at most 2. If not, transform the circuit appropriately (using binary trees) to ensure that this condition is satisfied. The process increases the size of C by a constant factor.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $poly(sd)$ that uses only $+$ and $\times$ gates.
- *Proof sketch.* Observe that a gate of C computes a rational function. The idea is to *keep track of the numerators and denominators of these rational functions separately* using the following relations:
  - $h_1/g_1 + h_2/g_2 = (h_1g_2 + h_2g_1)/(g_1g_2)$
  - $h_1/g_1 \times h_2/g_2 = (h_1h_2)/(g_1g_2)$

  Only the o/p gate of the resulting circuit is a $\div$ gate.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size poly(sd) that uses only $+$ and $\times$ gates.
- *Proof sketch.* The o/p $\div$ gate computes $f = h/g$, for some $g \neq 0$. Observe that the degree of h and g could be as high as $D = 2^{O(s)}$. Suppose, $|\mathbb{F}| > D$.

- We'll handle the small field size case later.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $\text{poly}(sd)$ that uses only $+$ and $\times$ gates.
- *Proof sketch.* The o/p $\div$ gate computes $f = h/g$, for some $g \neq 0$. Observe that the degree of h and g could be as high as $D = 2^{O(s)}$. Suppose, $|\mathbb{F}| > D$. Then, there's a point $\alpha \in \mathbb{F}^{|\mathbf{x}|}$ s.t. $c = g(\alpha) \neq 0$, and $g(\mathbf{x}+\alpha) = c.(1+\mathcal{g})$ for some *constant-term-free* $\mathcal{g} \in \mathbb{F}[\mathbf{x}]$. We'll focus on getting a circuit for $f(\mathbf{x}+\alpha)$ first and then translate it back by $-\alpha$ to compute f.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $\mathrm{poly}(sd)$ that uses only $+$ and $\times$ gates.
- *Proof sketch.* The o/p $\div$ gate computes $f = h/g$, for some $g \neq 0$. Observe that the degree of h and g could be as high as $D = 2^{O(s)}$. Suppose, $|\mathbb{F}| > D$. Then, there's a point $\alpha \in \mathbb{F}^{|\mathbf{x}|}$ s.t. $c = g(\alpha) \neq 0$, and $g(\mathbf{x}+\alpha) = c.(1+\mathcal{g})$ for some *constant-term-free* $\mathcal{g} \in \mathbb{F}[\mathbf{x}]$. We'll focus on getting a circuit for $f(\mathbf{x}+\alpha)$ first and then translate it back by $-\alpha$ to compute f. Reusing symbols, let's denote $f(\mathbf{x}+\alpha)$ by f, $c^{-1}h(\mathbf{x}+\alpha)$ by h and $g(\mathbf{x}+\alpha)$ by g.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $\mathrm{poly}(sd)$ that uses only $+$ and $\times$ gates.
- *Proof sketch.* Then, $f = h/(1+g) = h(1-g+g^2-g^3+\ldots)$.
- Notice that the RHS has a power series expression; cancellation of terms "shrinks" it to a polynomial.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $\mathrm{poly}(sd)$ that uses only $+$ and $\times$ gates.
- *Proof sketch.* Then, $f = h/(1+g) = h(1-g+g^2-g^3+\ldots)$.
- Note, $\deg(f) \le d$ & $\deg(g^i) \ge i$, as g is *constant-term-free*.
- So, it is sufficient to truncate the above series after g^d.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $\mathrm{poly}(sd)$ that uses only $+$ and $\times$ gates.
- *Proof sketch.* Then, $f = h/(1+g) = h(1-g+g^2-g^3+\ldots)$.
- Note, $\deg(f) \le d$ & $\deg(g^i) \ge i$, as g is *constant-term-free*.
- So, it is sufficient to truncate the above series after g^d.
- Notation. Denote the i^{th} *homogeneous component* of a polynomial p by $p^{[i]}$, i.e., $p^{[i]}$ is the sum of the degree-i monomials of p.

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $\text{poly}(sd)$ that uses only $+$ and $\times$ gates.
- *Proof sketch.* Then, $f = h/(1+g) = h(1-g+g^2-g^3+\ldots)$.
- Let $p = h(1-g+g^2-g^3+\ldots+(-1)^d g^d)$. Then,

$$f = p^{[0]} + p^{[1]} + \ldots + p^{[d]}$$

Removing division gates

- Theorem. *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size $poly(sd)$ that uses only $+$ and $\times$ gates.
- *Proof sketch.* Then, $f = h/(1+g) = h(1-g+g^2-g^3+\ldots)$.
- Let $p = h(1-g+g^2-g^3+\ldots+(-1)^d g^d)$. Then,

$$f = p^{[0]} + p^{[1]} + \ldots + p^{[d]}$$

- As h and g are computable by circuits of size $O(s)$, p is computable by a circuit of size $poly(sd)$.
- Can we compute the homogeneous components of p?

Computing homogeneous components

- Lemma. *(Strassen 1973)* Let $p \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit having $+$ and $\times$ gates. Then, $p^{[0]}, p^{[1]}, \ldots, p^{[d]}$ are computable by a circuit of size $O(d^2 s)$.
- *Proof sketch.* For every gate computing a polynomial q, create $d+1$ gates computing $q^{[0]}, q^{[1]}, \ldots, q^{[d]}$. ∎

Computing homogeneous components

- Lemma. *(Strassen 1973)* Let $p \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit having $+$ and $\times$ gates. Then, $p^{[0]}, p^{[1]}, \ldots, p^{[d]}$ are computable by a circuit of size $O(d^2 s)$.
- *Proof sketch.* For every gate computing a polynomial q, create $d+1$ gates computing $q^{[0]}, q^{[1]}, \ldots, q^{[d]}$. ▪

- *Homework.* Fill in the details. Also, prove a *black-box* version of the above lemma (using interpolation).

Removing division gates

- **Theorem.** *(Strassen 1973)* Let $f \in \mathbb{F}[\mathbf{x}]$ be a degree-d polynomial that is computable by a size-s circuit C having $+$, $\times$, and $\div$ gates. Then, f is also computable by a circuit of size poly(sd) that uses only $+$ and $\times$ gates.
- *Proof sketch.* Then, $f = h/(1+g) = h(1-g+g^2-g^3+\ldots)$.
- Let $p = h(1-g+g^2-g^3+\ldots+(-1)^d g^d)$. Then,

$$f = p^{[0]} + p^{[1]} + \ldots + p^{[d]}$$

- Compute $p^{[0]}, p^{[1]}, \ldots, p^{[d]}$ using the previous lemma and then compute f using the above equation. ∎

- How to handle small fields?

Handling small fields

- Obs. Let $\mathbb{F}$ be a finite field and $\mathbb{K}$ be a field extension of $\mathbb{F}$ of degree k. If $f \in \mathbb{F}[\mathbf{x}]$ is computable by a size-s circuit over $\mathbb{K}$, then f is also computable by a circuit of size $O(k^2 s)$ over $\mathbb{F}$.

Handling small fields

- Obs. Let $\mathbb{F}$ be a finite field and $\mathbb{K}$ be a field extension of $\mathbb{F}$ of degree k. If $f \in \mathbb{F}[\mathbf{x}]$ is computable by a size-s circuit over $\mathbb{K}$, then f is also computable by a circuit of size $O(k^2 s)$ over $\mathbb{F}$.
- *Proof sketch*. Field $\mathbb{K} \cong \mathbb{F}[y]/(h(y))$, where $h(y) \in \mathbb{F}[y]$ is an irreducible polynomial of degree k. A polynomial $g(\mathbf{x}) \in \mathbb{K}[\mathbf{x}]$ can be naturally expressed as
$$g(\mathbf{x}) = g_0(\mathbf{x}) + g_1(\mathbf{x})y + \ldots + g_{k-1}(\mathbf{x})y^{k-1}$$
where each $g_i(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$.

Handling small fields

- Obs. Let $\mathbb{F}$ be a finite field and $\mathbb{K}$ be a field extension of $\mathbb{F}$ of degree k. If $f \in \mathbb{F}[\mathbf{x}]$ is computable by a size-s circuit over $\mathbb{K}$, then f is also computable by a circuit of size $O(k^2 s)$ over $\mathbb{F}$.
- *Proof sketch*. For every gate computing $g(\mathbf{x}) \in \mathbb{K}[\mathbf{x}]$, create k gates computing $g_0(\mathbf{x}), g_1(\mathbf{x}), \ldots, g_{k-1}(\mathbf{x})$ using the polynomial $h(y)$. ∎
- *Homework.* Fill in the details. (Similar to the proof of the last lemma)

Handling small fields

- Obs. Let $\mathbb{F}$ be a finite field and $\mathbb{K}$ be a field extension of $\mathbb{F}$ of degree k. If $f \in \mathbb{F}[\mathbf{x}]$ is computable by a size-s circuit over $\mathbb{K}$, then f is also computable by a circuit of size $O(k^2 s)$ over $\mathbb{F}$.

- Note. In the proof of Strassen's theorem, we may have to work with a field extension of $\mathbb{F}$ of degree $O(s)$.

The determinant

- Let $X = (x_{ij})_{i,j \in [n]}$. Then,

$$\mathrm{Det}_n := \det(X) = \sum_{\sigma \in S_n} (-1)^{\mathrm{sign}(\sigma)} \prod_{i \in [n]} x_{i\,\sigma(i)} ,$$

- Question. How fast can we compute Det_n?
- The *Gaussian elimination* method yields a circuit (having only + and × gates) of size and depth poly(n).

The determinant

- Let $X = (x_{ij})_{i,j\in[n]}$. Then,

$$\mathrm{Det}_n := \det(X) = \sum_{\sigma\in S_n} (-1)^{\mathrm{sign}(\sigma)} \prod_{i\in[n]} x_{i\,\sigma(i)} \,,$$

- Question. How fast can we compute Det_n?
- The *Gaussian elimination* method yields a circuit (having only + and × gates) of size and depth poly(n).
- *Valiant, Skyum, Berkowitz, Rackoff '83* gave a general *depth-reduction result* for circuits. (We'll discuss this later)
- *Borodin, von zur Gathen, Hopcroft '82.* $O(n^{15})$-size circuit of fan-in 2 and depth $O((\log n)^2)$ over any field, using the above depth reduction result.

The determinant

- Let $X = (x_{ij})_{i,j\in[n]}$. Then,

$$\mathrm{Det}_n := \det(X) = \sum_{\sigma\in S_n} (-1)^{\mathrm{sign}(\sigma)} \prod_{i\in[n]} x_{i\,\sigma(i)} ,$$

- Question. How fast can we compute Det_n?
- The *Gaussian elimination* method yields a circuit (having only + and × gates) of size and depth poly(n).

- But there are *low-depth circuits* for Det_n of significantly *smaller size* than what Gaussian elimination provides.

Low depth circuits for $\det_n$

- History:
  1. *Csanky '76.* $O(n^4)$-size circuit of depth $O((\log n)^2)$ over fields of characteristic 0 or $> n$.

- We'll discuss this algorithm in details.

Low depth circuits for $\det_n$

- History:
  1. *Csanky '76.* $O(n^4)$-size circuit of depth $O((\log n)^2)$ over fields of characteristic 0 or > n.
  2. *Berkowitz '84.* $O(n^4 \log n)$-size circuit of depth $O((\log n)^2)$ over *any* field.
  3. *Chistov '84; Pippenger 2022.* $O(n^4 \log n)$-size circuit of depth $O((\log n)^2)$ over *any* field.

- These circuits have fanin bounded by 2.

Low depth circuits for $\det_n$

- History:
  1. *Csanky '76.* $O(n^4)$-size circuit of depth $O((\log n)^2)$ over fields of characteristic 0 or $> n$.
  2. *Berkowitz '84.* $O(n^4 \log n)$-size circuit of depth $O((\log n)^2)$ over *any* field.
  3. *Chistov '84; Pippenger 2022.* $O(n^4 \log n)$-size circuit of depth $O((\log n)^2)$ over *any* field.
  4. *Gupta, Kamath, Kayal, Saptharishi 2013; Tavenas 2013*. $n^{O(\sqrt{n})}$-size (*unbounded fanin*) depth 3 circuit over any field. (We'll discuss this result later)

Csanky's algorithm: The idea

- Focus on $n = 2$, i.e., $X = (x_{ij})_{i,j\in[2]}$. Let y_1, y_2 be the eigenvalues of X. Then, $\det(X) = y_1 y_2$.
- Also, $tr(X) = y_1+y_2$ and $tr(X^2) = y_1^2 + y_2^2$, where $tr(X)$ is the *trace* of X.

Csanky's algorithm: The idea

- Focus on $n = 2$, i.e., $X = (x_{ij})_{i,j\in[2]}$. Let y_1, y_2 be the eigenvalues of X. Then, $\det(X) = y_1 y_2$.
- Also, $tr(X) = y_1+y_2$ and $tr(X^2) = y_1^2 + y_2^2$, where $tr(X)$ is the *trace* of X.
- Hence, $\det(X) = ½\cdot(tr(X)^2 – tr(X^2))$.
- We can compute $\det$ by computing X, X^2 and then computing their traces, provided $char(\mathbb{F}) \neq 2$.

Csanky's algorithm: The idea

- Focus on $n = 2$, i.e., $X = (x_{ij})_{i,j \in [2]}$. Let y_1, y_2 be the eigenvalues of X. Then, $\det(X) = y_1 y_2$.
- Also, $tr(X) = y_1 + y_2$ and $tr(X^2) = y_1^2 + y_2^2$, where $tr(X)$ is the *trace* of X.
- Hence, $\det(X) = ½ \cdot (tr(X)^2 - tr(X^2))$.
- We can compute $\det$ by computing X, X^2 and then computing their traces, provided $char(\mathbb{F}) \neq 2$.
- Question. For any n, can $\det(X)$ be expressed as a polynomial in $tr(X), \ldots, tr(X^n)$?

Csanky's algorithm: The idea

- Focus on $n = 2$, i.e., $X = (x_{ij})_{i,j\in[2]}$. Let y_1, y_2 be the eigenvalues of X. Then, $\det(X) = y_1 y_2$.
- Also, $\mathrm{tr}(X) = y_1 + y_2$ and $\mathrm{tr}(X^2) = y_1^2 + y_2^2$, where $\mathrm{tr}(X)$ is the <u>*trace*</u> of X.
- Hence, $\det(X) = ½\cdot(\mathrm{tr}(X)^2 – \mathrm{tr}(X^2))$.
- We can compute $\det$ by computing X, X^2 and then computing their traces, provided $\mathrm{char}(\mathbb{F}) \neq 2$.
- Question. For any n, can $\det(X)$ be expressed as a polynomial in $\mathrm{tr}(X), \ldots, \mathrm{tr}(X^n)$? **Yes!**
- Although the eigenvalues of X are *not* polynomials in the entries of X, $\mathrm{tr}(X^i)$ is.

Csanky's algorithm

- Let $X = (x_{ij})_{i,j\in[n]}$ & $\mathbf{y} = \{y_1,\ldots, y_n\}$ the eigenvalues of X.
- The *characteristic polynomial* of X,

  $h(y) = \det(yI_n - X) = (y - y_1)\cdot\ldots\cdot(y - y_n)$

  $= y^n+s_1y^{n-1}+\ldots+s_n$, where $s_i = (-1)^i\cdot ESym_{n,i}(\mathbf{y})$.
- Each s_i is also a polynomial in $\mathbf{x} = \{x_{ij}\}_{i,j\in[n]}$ of degree i.
- Notice that $s_n = (-1)^n\det(X)$. Goal: Circuit for $s_1,\ldots, s_n$.

Csanky's algorithm

- Let $X = (x_{ij})_{i,j\in[n]}$ & $\mathbf{y} = \{y_1,\ldots, y_n\}$ the eigenvalues of X.
- The *characteristic polynomial* of X,

  $h(y) = \det(yI_n - X) = (y - y_1)\cdot\ldots\cdot(y - y_n)$

  $= y^n+s_1y^{n-1}+\ldots+s_n$, where $s_i = (-1)^i\cdot ESym_{n,i}(\mathbf{y})$.
- Each s_i is also a polynomial in $\mathbf{x} = \{x_{ij}\}_{i,j\in[n]}$ of degree i.
- Notice that $s_n = (-1)^n\det(X)$. Goal: Circuit for $s_1,\ldots, s_n$.
- Further, $tr(X^i) = y_1^i +\ldots+ y_n^i = PSym_{n,i}(\mathbf{y})$, the i^{th} *power symmetric polynomial*. Denote $PSym_{n,i}(\mathbf{y})$ by p_i.

Csanky's algorithm

- Let $X = (x_{ij})_{i,j\in[n]}$ & $\mathbf{y} = \{y_1,\ldots, y_n\}$ the eigenvalues of X.
- The *characteristic polynomial* of X,

  $h(y) = \det(yI_n - X) = (y - y_1)\cdot\ldots\cdot(y - y_n)$

  $= y^n+s_1y^{n-1}+\ldots+s_n$, where $s_i = (-1)^i\cdot ESym_{n,i}(\mathbf{y})$.
- Each s_i is also a polynomial in $\mathbf{x} = \{x_{ij}\}_{i,j\in[n]}$ of degree i.
- Notice that $s_n = (-1)^n\det(X)$. Goal: Circuit for $s_1,\ldots, s_n$.
- Further, $tr(X^i) = y_1^i +\ldots+ y_n^i = PSym_{n,i}(\mathbf{y})$, the i^{th} *power symmetric polynomial*. Denote $PSym_{n,i}(\mathbf{y})$ by p_i.
- Easy to compute p_i from X^i.
- Question. Can we compute $s_1,\ldots, s_n$ from $p_1,\ldots,p_n$?

Csanky's algorithm

- Let $X = (x_{ij})_{i,j\in[n]}$ & $\mathbf{y} = \{y_1,\ldots, y_n\}$ the eigenvalues of X.
- The *characteristic polynomial* of X,

  $h(y) = \det(yI_n - X) = (y - y_1)\cdot\ldots\cdot(y - y_n)$

  $= y^n+s_1y^{n-1}+\ldots+s_n$, where $s_i = (-1)^i\cdot ESym_{n,i}(\mathbf{y})$.
- Each s_i is also a polynomial in $\mathbf{x} = \{x_{ij}\}_{i,j\in[n]}$ of degree i.
- Notice that $s_n = (-1)^n\det(X)$. Goal: Circuit for $s_1,\ldots, s_n$.
- Further, $tr(X^i) = y_1^i +\ldots+ y_n^i = PSym_{n,i}(\mathbf{y})$, the i^{th} *power symmetric polynomial*. Denote $PSym_{n,i}(\mathbf{y})$ by p_i.
- Easy to compute p_i from X^i.
- Question. Can we compute $s_1,\ldots, s_n$ from $p_1,\ldots,p_n$? Yes! using **Newton Identities**

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$ks_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$k s_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- *Proof.* Polynomial $h(y) = s_0 y^n + s_1 y^{n-1} + \ldots + s_n$.
- Case $k = n$: As $h(y_i) = s_0 y_i^n + s_1 y_i^{n-1} + \ldots + s_n = 0$, summing over $i \in [n]$, we get $n s_n + \sum_{i \in [n]} s_{n-i} p_i = 0$.

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$ks_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- *Proof.* Polynomial $h(y) = s_0 y^n + s_1 y^{n-1} + \ldots + s_n$.
- Case $k < n$: Every monomial in $ks_k + \sum_{i \in [k]} s_{k-i} p_i$ has *support* at most k in $\mathbf{y} = \{y_1, \ldots, y_n\}$.
- *Support* of a monomial is the number of variables with nonzero exponents appearing in the monomial.

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$ks_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- *Proof.* Polynomial $h(y) = s_0 y^n + s_1 y^{n-1} + \ldots + s_n$.
- Case $k < n$: Every monomial in $ks_k + \sum_{i \in [k]} s_{k-i} p_i$ has *<u>support</u>* at most k in $\mathbf{y} = \{y_1, \ldots, y_n\}$.
- W.l.o.g let m be a monomial in $ks_k + \sum_{i \in [k]} s_{k-i} p_i$ in the variables $\mathbf{y}_k := \{y_1, \ldots, y_k\}$. Let $\mathbf{z} = \mathbf{y} \backslash \mathbf{y}_k$.
- Obs. The coefficient of m in $ks_k + \sum_{i \in [k]} s_{k-i} p_i$ is the same as that in $[ks_k + \sum_{i \in [k]} s_{k-i} p_i]_{\mathbf{z}=0}$.

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$k s_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- *Proof.* Polynomial $h(y) = s_0 y^n + s_1 y^{n-1} + \ldots + s_n$.
- Case $k < n$: Every monomial in $k s_k + \sum_{i \in [k]} s_{k-i} p_i$ has *support* at most k in $\mathbf{y} = \{y_1, \ldots, y_n\}$.
- W.l.o.g let m be a monomial in $k s_k + \sum_{i \in [k]} s_{k-i} p_i$ in the variables $\mathbf{y}_k := \{y_1, \ldots, y_k\}$. Let $\mathbf{z} = \mathbf{y} \setminus \mathbf{y}_k$.
- Obs. The coefficient of m in $k s_k + \sum_{i \in [k]} s_{k-i} p_i$ is the same as that in $k[s_k]_{\mathbf{z}=0} + \sum_{i \in [k]} [s_{k-i}]_{\mathbf{z}=0} [p_i]_{\mathbf{z}=0}$.

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \le n$,

$$ks_k + \sum_{i\in[k]} s_{k-i}p_i = 0; \quad \text{here } s_0 = 1.$$

- *Proof.* Polynomial $h(y) = s_0y^n + s_1y^{n-1} + \ldots + s_n$.
- Case $k < n$: Every monomial in $ks_k+\sum_{i\in[k]} s_{k-i}p_i$ has *<u>support</u>* at most k in $\mathbf{y} = \{y_1,\ldots,y_n\}$.
- Notice that $[s_i]_{\mathbf{z}=0} = (-1)^i\cdot ESym_{k,i}(\mathbf{y}_k) = s_i(\mathbf{y}_k)$, and $[p_i]_{\mathbf{z}=0} = PSym_{k,i}(\mathbf{y}_k) = p_i(\mathbf{y}_k)$.
- Obs. The coefficient of m in $ks_k+\sum_{i\in[k]} s_{k-i}p_i$ is the same as that in $k[s_k]_{\mathbf{z}=0}+\sum_{i\in[k]} [s_{k-i}]_{\mathbf{z}=0}[p_i]_{\mathbf{z}=0}$.

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \le n$,

$$k s_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- *Proof.* Polynomial $h(y) = s_0 y^n + s_1 y^{n-1} + \ldots + s_n$.
- Case $k < n$: Every monomial in $k s_k + \sum_{i \in [k]} s_{k-i} p_i$ has ***support*** at most k in $\mathbf{y} = \{y_1, \ldots, y_n\}$.
- Notice that $[s_i]_{\mathbf{z}=0} = (-1)^i \cdot \mathrm{ESym}_{k,i}(\mathbf{y}_k) = s_i(\mathbf{y}_k)$, and $[p_i]_{\mathbf{z}=0} = \mathrm{PSym}_{k,i}(\mathbf{y}_k) = p_i(\mathbf{y}_k)$.
- Obs. The coefficient of m in $k s_k + \sum_{i \in [k]} s_{k-i} p_i$ is the same as that in $k s_k(\mathbf{y}_k) + \sum_{i \in [k]} s_{k-i}(\mathbf{y}_k) p_i(\mathbf{y}_k)$.

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \le n$,

$$ks_k + \sum_{i\in[k]} s_{k-i}p_i = 0; \quad \text{here } s_0 = 1.$$

- *Proof.* Polynomial $h(y) = s_0y^n + s_1y^{n-1} + \ldots + s_n$.
- Case $k < n$: Every monomial in $ks_k+\sum_{i\in[k]} s_{k-i}p_i$ has ***support*** at most k in $\mathbf{y} = \{y_1,\ldots, y_n\}$.
- Notice that $[s_i]_{\mathbf{z}=0} = (-1)^i\cdot ESym_{k,i}(\mathbf{y}_k) = s_i(\mathbf{y}_k)$, and $[p_i]_{\mathbf{z}=0} = PSym_{k,i}(\mathbf{y}_k) = p_i(\mathbf{y}_k)$.
- Obs. The coefficient of m in $ks_k+\sum_{i\in[k]} s_{k-i}p_i$ is the same as that in $ks_k(\mathbf{y}_k)+\sum_{i\in[k]} s_{k-i}(\mathbf{y}_k)p_i(\mathbf{y}_k)$.

= 0 by Case k = n

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \le n$,

$$ks_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- *Proof.* Polynomial $h(y) = s_0 y^n + s_1 y^{n-1} + \ldots + s_n$.
- Case $k < n$: Every monomial in $ks_k + \sum_{i \in [k]} s_{k-i} p_i$ has *support* at most k in $\mathbf{y} = \{y_1, \ldots, y_n\}$.
- Therefore, the coefficient of m in $ks_k + \sum_{i \in [k]} s_{k-i} p_i$ is 0. As m is chosen arbitrarily,

$$ks_k + \sum_{i \in [k]} s_{k-i} p_i = 0.$$

∎

Newton Identities

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$ks_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- Ref. See the wiki page on Newton Identities for more on the *power symmetric*, the *elementary symmetric*, and the *complete homogeneous symmetric polynomial*.

Csanky's algorithm

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$k s_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- $$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ p_1 & 2 & 0 & \cdots & 0 \\ p_2 & p_1 & 3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n-1} & p_{n-2} & \cdots & p_1 & n \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{pmatrix} = - \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_n \end{pmatrix}$$

Csanky's algorithm

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$k s_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- $$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ p_1/2 & 1 & 0 & \cdots & 0 \\ p_2/3 & p_1/3 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n-1}/n & p_{n-2}/n & \cdots & p_1/n & 1 \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{pmatrix} = - \begin{pmatrix} p_1 \\ p_2/2 \\ p_3/3 \\ \vdots \\ p_n/n \end{pmatrix}$$

Provided $\mathrm{char}(\mathbb{F}) > n$.

Csanky's algorithm

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$k s_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- $$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ p_1/2 & 1 & 0 & \cdots & 0 \\ p_2/3 & p_1/3 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{n-1}/n & p_{n-2}/n & \cdots & p_1/n & 1 \end{pmatrix} \cdot \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{pmatrix} = - \begin{pmatrix} p_1 \\ p_2/2 \\ p_3/3 \\ \vdots \\ p_n/n \end{pmatrix}$$

||

$I_n + P$, where P is a nilpotent matrix, i.e., $P^n = 0$

Csanky's algorithm

- Theorem. *(Girard 1629, Newton 1666)* For $k \leq n$,

$$k s_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- $$\begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{pmatrix} = -\,(I_n + P)^{-1} \cdot \begin{pmatrix} p_1 \\ p_2/2 \\ p_3/3 \\ \vdots \\ p_n/n \end{pmatrix}$$

Csanky's algorithm

- **Theorem.** *(Girard 1629, Newton 1666)* For $k \leq n$,

$$k s_k + \sum_{i \in [k]} s_{k-i} p_i = 0; \quad \text{here } s_0 = 1.$$

- $$\begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_n \end{pmatrix} = -\left(I_n - P + P^2 - \ldots + (-1)^{n-1} P^{n-1}\right) \cdot \begin{pmatrix} p_1 \\ p_2/2 \\ p_3/3 \\ \vdots \\ p_n/n \end{pmatrix}$$ -- Equation 1

Csanky's algorithm

- Claim. Given an $n \times n$ matrix A, we can compute A^2, …, A^n using a circuit of size $O(n^4)$ & depth $O((\log n)^2)$.
- *Proof sketch.* The circuit has $O(\log n)$ stages. In the first stage compute A^2, in the second compute A^3, A^4, in the third compute A^5, A^6, A^7 A^8, and so on. The i^{th} stage involves 2^{i-1} matrix multiplications.

Csanky's algorithm

- Claim. Given an $n \times n$ matrix A, we can compute $A^2, \ldots, A^n$ using a circuit of size $O(n^4)$ & depth $O((\log n)^2)$.
- *Proof sketch.* The circuit has $O(\log n)$ stages. In the first stage compute A^2, in the second compute A^3, A^4, in the third compute A^5, A^6, A^7 A^8, and so on. The i^{th} stage involves 2^{i-1} matrix multiplications.
- Multiplication of two $n \times n$ matrices can be computed using a circuit of size $O(n^3)$ and depth $O(\log n)$. So, the overall size of the circuit is $O(n^4)$ and its depth is $O((\log n)^2)$.

Csanky's algorithm

- Claim. Given an $n \times n$ matrix A, we can compute $A^2, \ldots, A^n$ using a circuit of size $O(n^4)$ & depth $O((\log n)^2)$.
- Algorithm/Circuit. Input: $X = (x_{ij})_{i,j \in [n]}$

1. Compute $X^2, \ldots, X^n$.
2. Compute $p_1, \ldots, p_n$.
3. Compute $P^2, \ldots, P^{n-1}$.
4. Compute $(I_n + P)^{-1}$ and $s_1, \ldots, s_n$ using Equation 1.

Csanky's algorithm

- Claim. Given an $n \times n$ matrix A, we can compute $A^2, \ldots, A^n$ using a circuit of size $O(n^4)$ & depth $O((\log n)^2)$.
- Algorithm/Circuit. Input: $X = (x_{ij})_{i,j \in [n]}$

1. Compute $X^2, \ldots, X^n$. (circuit size $O(n^4)$ & depth $O((\log n)^2)$)
2. Compute $p_1, \ldots, p_n$. (circuit size $O(n^2)$ & depth $O(\log n)$)
3. Compute $P^2, \ldots, P^{n-1}$. (circuit size $O(n^4)$ & depth $O((\log n)^2)$)
4. Compute $(I_n + P)^{-1}$ and $s_1, \ldots, s_n$ using Equation 1.

   (circuit size $O(n^3)$ & depth $O(\log n)$)

Csanky's algorithm

- Claim. Given an $n \times n$ matrix A, we can compute $A^2, \ldots, A^n$ using a circuit of size $O(n^4)$ & depth $O((\log n)^2)$.
- Algorithm/Circuit. Input: $X = (x_{ij})_{i,j \in [n]}$
  1. Compute $X^2, \ldots, X^n$.
  2. Compute $p_1, \ldots, p_n$.
  3. Compute $P^2, \ldots, P^{n-1}$.
  4. Compute $(I_n + P)^{-1}$ and $s_1, \ldots, s_n$ using Equation 1.

- Corollary. Det_n can be computed by a circuit of size $O(n^4)$ and depth $O((\log n)^2)$, provided $\mathrm{char}(\mathbb{F}) > n$.

Computing inverse of a matrix

- As $y^n+s_1y^{n-1}+\ldots+s_n$ is the *characteristic polynomial* of X, by the <u>*Cayley-Hamilton theorem*</u>, $X^n + s_1X^{n-1}+\ldots+s_n = 0$.

- Hence, $X^{-1} = -s_n^{-1}(X^{n-1} + s_1X^{n-2} + \ldots + s_{n-1})$.

- Corollary. X^{-1} can be computed by a circuit of size $O(n^4)$ and depth $O((\log n)^2)$, provided $\mathrm{char}(\mathbb{F}) > n$.

- The above circuit has <u>*only one*</u> $\div$ gate at the top.

Parallel computation of rank

Schwartz-Zippel lemma

- Lemma. *(Schwartz 1980, Zippel 1979)* Let $f(x_1, \ldots, x_n) \neq 0$ be a multivariate polynomial of (total) degree at most d over a field $\mathbb{F}$. Let $S \subseteq \mathbb{F}$ be finite, and $(a_1, \ldots, a_n) \in S^n$ such that each a_i is chosen independently and uniformly at random from S. Then,

$$\Pr_{(a_1, \ldots, a_n) \in_r S^n} [f(a_1, \ldots, a_n) = 0] \leq d/|S|.$$

- *Proof sketch.* Roots are far fewer than non-roots. Use induction on the number of variables.

(Homework)

Linear independence of polynomials

- Lemma 1. Let $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be $\mathbb{F}$-linearly independent n-variate, deg-d polynomials. Then, the determinant of the following matrix is *non-zero*.

$f_1(\mathbf{x}_1)$	$f_2(\mathbf{x}_1)$	…	$f_m(\mathbf{x}_1)$
$f_1(\mathbf{x}_2)$	$f_2(\mathbf{x}_2)$	…	$f_m(\mathbf{x}_2)$
$f_1(\mathbf{x}_m)$	$f_2(\mathbf{x}_m)$	…	$f_m(\mathbf{x}_m)$

  Here, $\mathbf{x}_1, \ldots, \mathbf{x}_m$ are disjoint sets of n variables.

- *Proof sketch.* Use induction on m and the Schwartz-Zippel lemma. *(Homework)*

Linear independence of polynomials

- Corollary 1. Let $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be $\mathbb{F}$-linearly independent n-variate, deg-d polynomials. Then, the determinant of the following matrix is _non-zero_ w.h.p.

$f_1(\mathbf{a}_1)$	$f_2(\mathbf{a}_1)$	…	$f_m(\mathbf{a}_1)$
$f_1(\mathbf{a}_2)$	$f_2(\mathbf{a}_2)$	…	$f_m(\mathbf{a}_2)$
$f_1(\mathbf{a}_m)$	$f_2(\mathbf{a}_m)$	…	$f_m(\mathbf{a}_m)$

  Here, $\mathbf{a}_1, \ldots, \mathbf{a}_m \in_r S^n$, where $S \subseteq \mathbb{F}$ and $|S| = 10md$.

- *Proof sketch.* Use the Schwartz-Zippel lemma.

Rank computation in NC

- Remark. If the input matrix is rectangular, pad it up with zeroes to make it a square matrix.
- Notation. Let $[M]_i$ be the *principal i x i submatrix* of a matrix M. Let $S \subseteq \mathbb{F}$ and $|S| = 20n$.

- Algorithm. Input: $A = (a_{ij})_{i,j \in [n]}$
  1. Pick $X \in_r S^{nxn}$ and $Y \in_r S^{nxn}$.
  2. Compute $d_i = \det([XAY]_i)$ for $i \in [n]$.
  3. Output $r = \max\{\{i : d_i \neq 0\}, 0\}$.

Rank computation in NC

- Remark. If the input matrix is rectangular, pad it up with zeroes to make it a square matrix.
- Notation. Let $[M]_i$ be the *principal i x i submatrix* of a matrix M. Let $S \subseteq \mathbb{F}$ and $|S| = 20n$.

- Algorithm. Input: $A = (a_{ij})_{i,j \in [n]}$
  1. Pick $X \in_r S^{n \times n}$ and $Y \in_r S^{n \times n}$.
  2. Compute $d_i = \det([XAY]_i)$ for $i \in [n]$.
  3. Output $r = \max\{\{i : d_i \neq 0\}, 0\}$.

} all in NC

Rank computation in NC

- Theorem. *(Borodin, Gathen, Hopcroft '82)* The algorithm outputs the rank of A with probability at least 0.9.
- *Proof sketch.* Define linear forms $f_1(\mathbf{x}), \ldots, f_n(\mathbf{x})$ whose coefficient vectors are the columns of A. Use Corollary 1 to argue that the first r rows of XA are linearly independent w.p. ≥ 0.95, where $r = \mathrm{rank}(A)$.

Rank computation in NC

- Theorem. *(Borodin, Gathen, Hopcroft '82)* The algorithm outputs the rank of A with probability at least 0.9.
- *Proof sketch.* Define linear forms $f_1(\mathbf{x}), \ldots, f_n(\mathbf{x})$ whose coefficient vectors are the columns of A. Use Corollary 1 to argue that the first r rows of XA are linearly independent w.p. ≥ 0.95, where $r = \mathrm{rank}(A)$.
- Define linear forms $g_1(\mathbf{x}), \ldots, g_r(\mathbf{x})$ whose coefficient vectors are the first r rows of XA. Use Corollary 1 to argue that $[XAY]_r$ has rank r w.p. ≥ 0.95.