Lecture 22: Black-Box Linear Algebra & Wiedemann's Algorithm for Linear System Solving

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

April 4, 2021

Overview

- Administrivia
- Black-Box Linear Algebra
- Wiedemann's Algorithm
- Computing Minimal Polynomials of Krylov Sequences
- Conclusion

Rate this course!

Please log in to

https://evaluate.uwaterloo.ca/

Today is the **last day** to provide us (and the school) with your evaluation and feedback on the course!

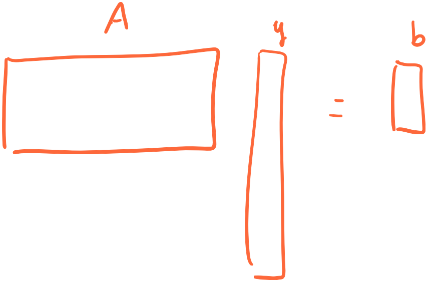
- This would really help me figuring out what worked and what didn't for the course
- And let the school know if I was a good boy this term!
- Teaching this course is also a learning experience for me :)

- Administrivia
- Black-Box Linear Algebra
- Wiedemann's Algorithm
- Computing Minimal Polynomials of Krylov Sequences
- Conclusion

Generic Approach to Linear Algebra

- **Problem:** given an input matrix $A \in \mathbb{F}^{n \times m}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find a (all) solution(s) $\mathbf{y} \in \mathbb{F}^m$ to

$$A\mathbf{y} = \mathbf{b}$$



Generic Approach to Linear Algebra

- **Problem:** given an input matrix $A \in \mathbb{F}^{n \times m}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find a (all) solution(s) $\mathbf{y} \in \mathbb{F}^m$ to

$$A\mathbf{y} = \mathbf{b}$$

- Naive solution: *Gaussian elimination*
- Running time: $O(\max\{m, n\}^3)$

Generic Approach to Linear Algebra

- **Problem:** given an input matrix $A \in \mathbb{F}^{n \times m}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find a (all) solution(s) $\mathbf{y} \in \mathbb{F}^m$ to

$$A\mathbf{y} = \mathbf{b}$$

- Naive solution: *Gaussian elimination*
- Running time: $O(\max\{m, n\}^3)$
- If A is a square and invertible matrix (for simplicity), above problem amounts to inverting A

Generic Approach to Linear Algebra

- **Problem:** given an input matrix $A \in \mathbb{F}^{n \times m}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find a (all) solution(s) $\mathbf{y} \in \mathbb{F}^m$ to

$$A\mathbf{y} = \mathbf{b}$$

- Naive solution: *Gaussian elimination*
- Running time: $O(\max\{m, n\}^3)$
- If A is a square and invertible matrix (for simplicity), above problem amounts to inverting A
- Can invert matrices $O(n^\omega)$ time!

Generic Approach to Linear Algebra

- **Problem:** given an input matrix $A \in \mathbb{F}^{n \times m}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find a (all) solution(s) $\mathbf{y} \in \mathbb{F}^m$ to

$$A\mathbf{y} = \mathbf{b}$$

- Naive solution: *Gaussian elimination*
- Running time: $O(\max\{m, n\}^3)$
- If A is a square and invertible matrix (for simplicity), above problem amounts to inverting A
- Can invert matrices $O(n^\omega)$ time!
- Can we do better?

Black-Box Model

- An ubiquitous problem in scientific computing is to solve system of linear equations $A\mathbf{y} = \mathbf{b}$
  1. linear programming
  2. optimization
  3. polynomial multiplication
  4. factoring
  5. polynomial interpolation (DFT)
  6. computing GCD of polynomials (Resultants)
  7. many more

Black-Box Model

- An ubiquitous problem in scientific computing is to solve system of linear equations $A\mathbf{y} = \mathbf{b}$
  1. linear programming
  2. optimization
  3. polynomial multiplication
  4. factoring
  5. polynomial interpolation (DFT)
  6. computing GCD of polynomials (Resultants)
  7. many more
- Often times, the input matrix A has very *special structure*, can exploit this structure to obtain *faster algorithms*

Black-Box Model

- An ubiquitous problem in scientific computing is to solve system of linear equations $A\mathbf{y} = \mathbf{b}$
  1. linear programming
  2. optimization
  3. polynomial multiplication
  4. factoring
  5. polynomial interpolation (DFT)
  6. computing GCD of polynomials (Resultants)
  7. many more
- Often times, the input matrix A has very *special structure*, can exploit this structure to obtain *faster algorithms*
- Often times, given a vector $\mathbf{c}$, we can evaluate $A\mathbf{c}$ much faster than the naive $O(n^2)$ algorithm. Can use it to get faster algorithms for linear system solving.

Black-Box Model

- An ubiquitous problem in scientific computing is to solve system of linear equations $A\mathbf{y} = \mathbf{b}$
  1. linear programming
  2. optimization
  3. polynomial multiplication
  4. factoring
  5. polynomial interpolation (DFT)
  6. computing GCD of polynomials (Resultants)
  7. many more
- Often times, the input matrix A has very *special structure*, can exploit this structure to obtain *faster algorithms*
- Often times, given a vector $\mathbf{c}$, we can evaluate $A\mathbf{c}$ much faster than the naive $O(n^2)$ algorithm. Can use it to get faster algorithms for linear system solving.
- We have already done that many times!

Black-Box Model: Example

- Often times, the input matrix $A \in \mathbb{F}^{n \times n}$ has very *special structure*, can exploit this structure to obtain *faster algorithms*

Black-Box Model: Example

- Often times, the input matrix $A \in \mathbb{F}^{n \times n}$ has very *special structure*, can exploit this structure to obtain *faster algorithms*
- Discrete Fourier Transform: $n = 2^k$, $z = e^{2\pi i/n}$.

$$A = V(1, z, z^2, \ldots, z^{n-1})$$

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & z & z^2 & \cdots & z^{n-1} \\ 1 & z^2 & z^4 & \cdots & z^{2(n-1)} \\ 1 & z^3 & z^6 & \cdots & z^{3(n-1)} \\ \vdots & & \vdots & & \vdots \\ 1 & z^{n-1} & \cdots & \cdots & z^{(n-1)^2} \end{pmatrix} = A$$

Black-Box Model: Example

- Often times, the input matrix $A \in \mathbb{F}^{n \times n}$ has very *special structure*, can exploit this structure to obtain *faster algorithms*
- Discrete Fourier Transform: $n = 2^k$, $z = e^{2\pi i/n}$.

$$A = V(1, z, z^2, \ldots, z^{n-1})$$

- We know that

$$A^{-1} = V(1, z^{-1}, z^{-2}, \ldots, z^{1-n})^T \cdot \frac{1}{n}$$

Black-Box Model: Example

- Often times, the input matrix $A \in \mathbb{F}^{n \times n}$ has very *special structure*, can exploit this structure to obtain *faster algorithms*
- Discrete Fourier Transform: $n = 2^k$, $z = e^{2\pi i/n}$.

$$A = V(1, z, z^2, \ldots, z^{n-1})$$

- We know that

$$A^{-1} = V(1, z^{-1}, z^{-2}, \ldots, z^{1-n})$$

- Know that $\mathbf{y} = A^{-1}\mathbf{b}$, so

$$\text{cost to solve the system} = \text{cost to multiply DFT matrix by vector } A^{-1}\mathbf{b}$$

Black-Box Model: Example

- Often times, the input matrix $A \in \mathbb{F}^{n \times n}$ has very *special structure*, can exploit this structure to obtain *faster algorithms*
- Discrete Fourier Transform: $n = 2^k$, $z = e^{2\pi i/n}$.

$$A = V(1, z, z^2, \ldots, z^{n-1})$$

- We know that

$$A^{-1} = V(1, z^{-1}, z^{-2}, \ldots, z^{1-n})$$

- Know that $\mathbf{y} = A^{-1}\mathbf{b}$, so

$$\text{cost to solve the system} = \text{cost to multiply DFT matrix by vector } A^{-1}\mathbf{b}$$

- Saw that this cost is $O(n \log n)$, which is much faster than $O(n^\omega)$

Cost of Evaluation

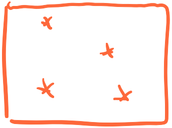
- Often times, given a vector $\mathbf{b}$, we can evaluate $A\mathbf{b}$ much faster than the naive $O(n^2)$ algorithm. Can use it to get faster algorithms

Cost of Evaluation

- Often times, given a vector $\mathbf{b}$, we can evaluate $A\mathbf{b}$ much faster than the naive $O(n^2)$ algorithm. Can use it to get faster algorithms
- Let $c(A)$ be the cost of multiplying A by any vector $\mathbf{b}$, and $M(n)$ the cost of multiplying two degree n polynomials

natural

Class of matrices	$c(A)$
general	$2n^2 - 2$
Sylvester Matrix	$O(M(n))$
DFT	$O(n \log n)$
Vandermonde matrix	$O(M(n) \log n)$
Berlekamp matrix over $\mathbb{F}_q$	$O(M(n) \log q)$
Sparse matrix with s non-zero entries	$2s$
Toeplitz matrix	$O(M(n))$



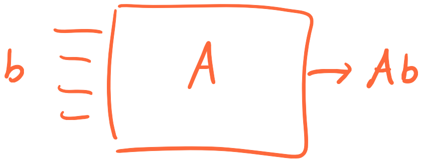
Cost of Evaluation

- Often times, given a vector $\mathbf{b}$, we can evaluate $A\mathbf{b}$ much faster than the naive $O(n^2)$ algorithm. Can use it to get faster algorithms
- Let $c(A)$ be the cost of multiplying A by any vector $\mathbf{b}$, and $M(n)$ the cost of multiplying two degree n polynomials

Class of matrices	$c(A)$
general	$2n^2 - 2$
Sylvester Matrix	$O(M(n))$
DFT	$O(n \log n)$
Vandermonde matrix	$O(M(n) \log n)$
Berlekamp matrix over $\mathbb{F}_q$	$O(M(n) \log q)$
Sparse matrix with s non-zero entries	$2s$
Toeplitz matrix	$O(M(n))$

- And these matrices appear quite often in practical applications!

input:



black-box

$c(A)$ field operations

task: solve linear system $Ay = b$ by using black-box.

- Administrivia
- Black-Box Linear Algebra
- Wiedemann's Algorithm
- Computing Minimal Polynomials of Krylov Sequences
- Conclusion

Wiedemann's Idea

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b} \qquad y = A^{-1}b$$

- Cost of multiplying A by *any vector*: $c(A)$

Wiedemann's Idea

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b}$$

- Cost of multiplying A by *any vector*: $c(A)$
- We need to compute $A^{-1}b$.

Wiedemann's Idea

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b}$$

- Cost of multiplying A by *any vector*: $c(A)$
- We need to compute $A^{-1}b$.
- Let $p_A(x)$ be the characteristic polynomial of A

$$p_A(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_1 x + \underbrace{(-1)^n \det(A)}_{\neq 0}$$

$p_A(x) = \det(xI - A)$

Wiedemann's Idea

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b}$$

- Cost of multiplying A by *any vector*: $c(A)$
- We need to compute $A^{-1}b$.
- Let $p_A(x)$ be the characteristic polynomial of A

$$p_A(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_1 x + (-1)^n \det(A)$$

- By Cayley-Hamilton, we know that $p_A(A) = 0$

$$p_A(A) = 0 = A^n + f_{n-1}A^{n-1} + \cdots + f_1 A + (-1)^n \det(A) \cdot I$$

$$0 \cdot b = \vec{0} = A^n \vec{b} + f_{n-1} A^{n-1} \vec{b} + \cdots + f_1 A \vec{b} + \underline{(-1)^n \det(A) \cdot \vec{b}}$$

Wiedemann's Idea

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b}$$

- Cost of multiplying A by *any vector*: $c(A)$
- We need to compute $A^{-1}b$.
- Let $p_A(x)$ be the characteristic polynomial of A

$$p_A(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_1 x + (-1)^n \det(A)$$

- By Cayley-Hamilton, we know that $p_A(A) = 0$

$$p_A(A) = 0 = A^n + f_{n-1}A^{n-1} + \cdots + f_1 A + (-1)^n \det(A) \cdot I$$

- Multiplying by $\mathbf{b}$, we get:

$$(-1)^{n+1} \det(A) \cdot \mathbf{b} = A \cdot \underbrace{(A^{n-1} + f_{n-1}A^{n-2} + \cdots + f_1 I)}_{M} \cdot \mathbf{b}$$

Wiedemann's Idea

$$(-1)^{n+1} \cdot \det(A) \cdot \vec{b} = A \cdot M \cdot \vec{b}$$

$$(-1)^{n+1} \det(A) \cdot A^{-1} \vec{b} = M \cdot b$$

$$\boxed{\frac{M}{(-1)^{n+1} \det(A)} = A^{-1}}$$

M can be computed from characteristic polynomial of A!

Wiedemann's Algorithm

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b}$$

- A's characteristic polynomial can be used to compute the inverse!

Wiedemann's Algorithm

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b}$$

- A's characteristic polynomial can be used to compute the inverse!
- If we pay closer attention, *any annihilating polynomial* of the Krylov sequence $(A^i\mathbf{b})_{i \geq 0}$ works!

Can use the *minimal polynomial* of $(A^i\mathbf{b})_{i \geq 0}$

$$m(x) = x^d + m_{d-1}x^{d-1} + \cdots + m_1 x + m_0$$

$$m(x) \mid \underbrace{p_A(x)}_{\text{also characteristic polynomial of } (A^i b)_{i \geq 0}}$$

$$m_0 \mid (-1)^n \det(A) \Rightarrow \boxed{m_0 \neq 0}$$

Wiedemann's Algorithm

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b}$$

- A's characteristic polynomial can be used to compute the inverse!
- If we pay closer attention, *any annihilating polynomial* of the Krylov sequence $(A^i\mathbf{b})_{i \geq 0}$ works!

Can use the *minimal polynomial* of $(A^i\mathbf{b})_{i \geq 0}$

- Wiedemann's Algorithm: on input $A \in \mathbb{F}^{n \times n}$ invertible and $\mathbf{b} \in \mathbb{F}^n$
  1. Compute the minimal polynomial $m(x)$ of the Krylov sequence $(A^i\mathbf{b})_{i \geq 0}$

$$m(x) = m_d x^d + \cdots + m_1 x + m_0$$

Wiedemann's Algorithm

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b}$$

- A's characteristic polynomial can be used to compute the inverse!
- If we pay closer attention, *any annihilating polynomial* of the Krylov sequence $(A^i\mathbf{b})_{i \geq 0}$ works!

Can use the *minimal polynomial* of $(A^i\mathbf{b})_{i \geq 0}$

- Wiedemann's Algorithm: on input $A \in \mathbb{F}^{n \times n}$ invertible and $\mathbf{b} \in \mathbb{F}^n$
  1. Compute the minimal polynomial $m(x)$ of the Krylov sequence $(A^i\mathbf{b})_{i \geq 0}$

$$m(x) = m_d x^d + \cdots + m_1 x + m_0$$

  2. compute $h(x) = -\dfrac{m(x) - m_0}{m_0 \cdot x}$
  3. compute $\mathbf{y} = h(A) \cdot \mathbf{b}$ using Horner's rule

} well-defined $m_0 \neq 0$ and black box evaluation

Wiedemann's Algorithm

- **Problem:** given an input invertible matrix $A \in \mathbb{F}^{n \times n}$ and a vector $\mathbf{b} \in \mathbb{F}^n$, find the solution $\mathbf{y} \in \mathbb{F}^n$ to

$$A\mathbf{y} = \mathbf{b}$$

- A's characteristic polynomial can be used to compute the inverse!
- If we pay closer attention, *any annihilating polynomial* of the Krylov sequence $(A^i\mathbf{b})_{i \geq 0}$ works!

Can use the *minimal polynomial* of $(A^i\mathbf{b})_{i \geq 0}$

- Wiedemann's Algorithm: on input $A \in \mathbb{F}^{n \times n}$ invertible and $\mathbf{b} \in \mathbb{F}^n$
  1. Compute the minimal polynomial $m(x)$ of the Krylov sequence $(A^i\mathbf{b})_{i \geq 0}$
  $$m(x) = m_d x^d + \cdots + m_1 x + m_0$$
  2. compute $h(x) = -\dfrac{m(x) - m_0}{m_0 \cdot x}$
  3. compute $\mathbf{y} = h(A) \cdot \mathbf{b}$ using Horner's rule
  4. return $\mathbf{y}$

$$m(x) = x^d + m_{d-1}x^{d-1} + \cdots + m_1 x + m_0$$

$$b, Ab, A^2 b, \ldots, A^d b$$

$$0 = A^d b + m_{d-1}A^{d-1} b + \cdots + m_1 A b + m_0 \cdot b$$

$$-m_0 b = A(\underbrace{A^{d-1} b + m_{d-1} A^{d-2} b + \cdots + m_1 b}_{x^{d-1} + m_{d-1} x^{d-2} + \cdots + m_2 x + m_1 \,=\, \frac{m(x) - m_0}{x}})$$

$$A^{-1} b = \frac{1}{-m_0} \cdot (\underbrace{A^{d-1} b + m_{d-1} A^{d-2} b + \cdots + m_1 b}_{\frac{m(x) - m_0}{-m_0 x} \,=\, h(x)}) \rightarrow h(A) \cdot b$$

Wiedemann's Algorithm: Correctness

- A is invertible, then $\underbrace{p_A(0)}_{\text{constant term of characteristic polynomial}} = (-1)^n \det(A) \neq 0$

Wiedemann's Algorithm: Correctness

- A is invertible, then $p_A(0) = (-1)^n \det(A) \neq 0$
- Minimal polynomial $m(x)$ divides $p_A(x)$, so $m_0 \neq 0$

$$\text{Thus, we can always compute } h(x) = -\frac{m(x) - m_0}{m_0 x}$$

Wiedemann's Algorithm: Correctness

- A is invertible, then $p_A(0) = (-1)^n \det(A) \neq 0$
- Minimal polynomial $m(x)$ divides $p_A(x)$, so $m_0 \neq 0$

$$\text{Thus, we can always compute } h(x) = -\frac{m(x) - m_0}{m_0 x}$$

- $m(x)$ is minimal polynomial then

$$A^d \mathbf{b} + m_{d-1} A^{d-1} \mathbf{b} + \cdots + m_1 A \mathbf{b} + m_0 \mathbf{b} = \mathbf{0}$$

Wiedemann's Algorithm: Correctness

- A is invertible, then $p_A(0) = (-1)^n \det(A) \neq 0$
- Minimal polynomial $m(x)$ divides $p_A(x)$, so $m_0 \neq 0$

$$\text{Thus, we can always compute } h(x) = -\frac{m(x) - m_0}{m_0 x}$$

- $m(x)$ is minimal polynomial then

$$A^d \mathbf{b} + m_{d-1} A^{d-1} \mathbf{b} + \cdots + m_1 A \mathbf{b} + m_0 \mathbf{b} = \mathbf{0}$$

- Rearranging, we get

$$h(A) \cdot \mathbf{b} = A^{-1} \mathbf{b}$$

Wiedemann's Algorithm: Runtime Analysis

- For Wiedemann's algorithm, all we need to do is:
  1. Compute minimal polynomial of $(A^i\mathbf{b})_{i\geq 0}$
  2. Compute $h(x)$ from the minimal polynomial
  3. Use Horner's rule and matrix-vector multiplication to compute $h(A)\mathbf{b}$

$$\boxed{A^i b}$$

$$b \to \boxed{A} \to Ab \to \boxed{A} \to A^2 b \to \boxed{A} \to A^3 b$$

Wiedemann's Algorithm: Runtime Analysis

- For Wiedemann's algorithm, all we need to do is:
  1. Compute minimal polynomial of $A^i\mathbf{b}$
  2. Compute $h(x)$ from the minimal polynomial
  3. Use Horner's rule and matrix-vector multiplication to compute $h(A)\mathbf{b}$
- Running time of each step:
  1. Next part of lecture
  2. $O(n)$ time
  3. This takes d matrix-vector multiplications and $O(d)$ vector additions, so running time $O(d \cdot c(A) + dn)$

Since $d \leq n$, the above is in the worst case $O(n \cdot c(A) + n^2)$

$d :=$ degree of minimal polynomial of $(A^i b)_{i \geq 0}$.

Wiedemann's Algorithm: Runtime Analysis

- For Wiedemann's algorithm, all we need to do is:
  1. Compute minimal polynomial of $A^i\mathbf{b}$
  2. Compute $h(x)$ from the minimal polynomial
  3. Use Horner's rule and matrix-vector multiplication to compute $h(A)\mathbf{b}$
- Running time of each step:
  1. Next part of lecture
  2. $O(n)$ time
  3. This takes d matrix-vector multiplications and $O(d)$ vector additions, so running time $O(d \cdot c(A) + dn)$

     Since $d \leq n$, the above is in the worst case $O(n \cdot c(A) + n^2)$
- Much better than n^ω for all the cases that we discussed!

Wiedemann's Algorithm: Runtime Analysis

- For Wiedemann's algorithm, all we need to do is:
  1. Compute minimal polynomial of $A^i\mathbf{b}$
  2. Compute $h(x)$ from the minimal polynomial
  3. Use Horner's rule and matrix-vector multiplication to compute $h(A)\mathbf{b}$
- Running time of each step:
  1. Next part of lecture
  2. $O(n)$ time
  3. This takes d matrix-vector multiplications and $O(d)$ vector additions, so running time $O(d \cdot c(A) + dn)$

     Since $d \leq n$, the above is in the worst case $O(n \cdot c(A) + n^2)$
- Much better than n^ω for all the cases that we discussed!
- Well, only if we can compute the minimal polynomial really fast...

Minimal Polynomials of Krylov Subspaces

- Last lecture: minimal polynomials of sequences of *field elements*

$(a_i)_{i \geq 0}$ $a_i \in \mathbb{F}$

[1] If $\mathbb{F}$ has enough elements.

Minimal Polynomials of Krylov Subspaces

- Last lecture: minimal polynomials of sequences of *field elements*
- Krylov subspaces are sequences in $\mathbb{F}^n$. How to compute minimal polynomials in this case?

[1] If $\mathbb{F}$ has enough elements.

Minimal Polynomials of Krylov Subspaces

- Last lecture: minimal polynomials of sequences of *field elements*
- Krylov subspaces are sequences in $\mathbb{F}^n$. How to compute minimal polynomials in this case?
- **Idea:** convert sequences in $\mathbb{F}^n$ to sequences over $\mathbb{F}$!

[1] If $\mathbb{F}$ has enough elements.

Minimal Polynomials of Krylov Subspaces

- Last lecture: minimal polynomials of sequences of *field elements*
- Krylov subspaces are sequences in $\mathbb{F}^n$. How to compute minimal polynomials in this case?
- **Idea:** convert sequences in $\mathbb{F}^n$ to sequences over $\mathbb{F}$!
- *Key Lemma*: Given a *random* $\mathbf{u} \in \mathbb{F}^n$, the sequences

$$\boxed{(A^i \mathbf{b})_{i \geq 0},} \quad \text{and} \quad \boxed{(\mathbf{u}^T A^i \mathbf{b})_{i \geq 0}}$$

have the *same minimal polynomial* with high probability![1]

$u^T A^i b \in \mathbb{F}$

[1] If $\mathbb{F}$ has enough elements.

Minimal Polynomials of Krylov Subspaces

- Last lecture: minimal polynomials of sequences of *field elements*
- Krylov subspaces are sequences in $\mathbb{F}^n$. How to compute minimal polynomials in this case?
- **Idea:** convert sequences in $\mathbb{F}^n$ to sequences over $\mathbb{F}$!
- *Key Lemma*: Given a *random* $\mathbf{u} \in \mathbb{F}^n$, the sequences

$$(A^i \mathbf{b})_{i \geq 0}, \qquad \text{and} \qquad (\mathbf{u}^T A^i \mathbf{b})_{i \geq 0}$$

  have the *same minimal polynomial* with high probability![1]
- Algorithm: input $A \in \mathbb{F}^{n \times n}$, $\mathbf{b} \in \mathbb{F}^n$
  1. If $\mathbf{b} = \mathbf{0}$, return 1

[1] If $\mathbb{F}$ has enough elements.

Minimal Polynomials of Krylov Subspaces

- Last lecture: minimal polynomials of sequences of *field elements*
- Krylov subspaces are sequences in $\mathbb{F}^n$. How to compute minimal polynomials in this case?
- **Idea:** convert sequences in $\mathbb{F}^n$ to sequences over $\mathbb{F}$!
- *Key Lemma*: Given a *random* $\mathbf{u} \in \mathbb{F}^n$, the sequences

$$(A^i \mathbf{b})_{i \geq 0}, \qquad \text{and} \qquad (\mathbf{u}^T A^i \mathbf{b})_{i \geq 0}$$

  have the *same minimal polynomial* with high probability![1]
- Algorithm: input $A \in \mathbb{F}^{n \times n}$, $\mathbf{b} \in \mathbb{F}^n$
  1. If $\mathbf{b} = \mathbf{0}$, return 1
  2. $\mathbf{u} \in U^n$ uniformly at random, $U \subset \mathbb{F}$ is a large enough finite set

[1] If $\mathbb{F}$ has enough elements.

Minimal Polynomials of Krylov Subspaces

- Last lecture: minimal polynomials of sequences of *field elements*
- Krylov subspaces are sequences in $\mathbb{F}^n$. How to compute minimal polynomials in this case?
- **Idea:** convert sequences in $\mathbb{F}^n$ to sequences over $\mathbb{F}$!
- *Key Lemma*: Given a *random* $\mathbf{u} \in \mathbb{F}^n$, the sequences

$$(A^i \mathbf{b})_{i \geq 0}, \qquad \text{and} \qquad (\mathbf{u}^T A^i \mathbf{b})_{i \geq 0}$$

  have the *same minimal polynomial* with high probability![1]
- Algorithm: input $A \in \mathbb{F}^{n \times n}$, $\mathbf{b} \in \mathbb{F}^n$
  1. If $\mathbf{b} = \mathbf{0}$, return 1
  2. $\mathbf{u} \in U^n$ uniformly at random, $U \subset \mathbb{F}$ is a large enough finite set
  3. Use algorithm from previous lecture to compute $m(x)$ minimal polynomial of $(\mathbf{u}^T A^i \mathbf{b})_{i \geq 0}$

[1] If $\mathbb{F}$ has enough elements.

Minimal Polynomials of Krylov Subspaces

- Last lecture: minimal polynomials of sequences of *field elements*
- Krylov subspaces are sequences in $\mathbb{F}^n$. How to compute minimal polynomials in this case?
- **Idea:** convert sequences in $\mathbb{F}^n$ to sequences over $\mathbb{F}$!
- *Key Lemma*: Given a *random* $\mathbf{u} \in \mathbb{F}^n$, the sequences

$$(A^i \mathbf{b})_{i \geq 0}, \qquad \text{and} \qquad (\mathbf{u}^T A^i \mathbf{b})_{i \geq 0}$$

  have the *same minimal polynomial* with high probability![1]
- Algorithm: input $A \in \mathbb{F}^{n \times n}$, $\mathbf{b} \in \mathbb{F}^n$
  1. If $\mathbf{b} = \mathbf{0}$, return 1
  2. $\mathbf{u} \in U^n$ uniformly at random, $U \subset \mathbb{F}$ is a large enough finite set
  3. Use algorithm from previous lecture to compute $m(x)$
     minimal polynomial of $\mathbf{u}^T A^i \mathbf{b}$.
  4. If $m(A)\mathbf{b} = \mathbf{0}$ return $m(x)$, else return to step (2) } checking that our guess is correct

[1] If $\mathbb{F}$ has enough elements.

$$\left(A^i\right)_{i\geq 0} \qquad A \in \mathbb{F}^{n\times n}$$

$$A = \begin{pmatrix} 1 & & \\ & 2 & \\ & & 3 \end{pmatrix}$$

$$(x-1)(x-2)(x-3) = \underbrace{p_A(x)}_{\text{minimal polynomial}}$$

$$\left(A^i \bar{0}\right)_{i\geq 0} = \left(\bar{0}\right)_{i\geq 0}$$

minimal polynomial 1

$$\left(A^i \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\right)_{i\geq 0} = \left(\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\right)_{i\geq 0}$$

minimal polynomial $x-1$

$$\left(m(x)\right) \ni p_A(x)$$

Example

- $\mathbb{F} = \mathbb{F}_5$ $= \mathbb{Z}/5\mathbb{Z}$

$$A = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 0 & 3 \\ 1 & 2 & -1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

Example

- $\mathbb{F} = \mathbb{F}_5$

$$A = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 0 & 3 \\ 1 & 2 & -1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

- Minimal polynomial for $(A^i\mathbf{b})_{i \geq 0}$ is $m(x) = x^3 + 3x + 1$

$$Ab = \begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix} \qquad A^2b = A(Ab) = \begin{pmatrix} -1 \\ -1 \\ 3 \end{pmatrix}$$

$$A^3b = A(A^2b) = \begin{pmatrix} -3 \\ 0 \\ -1 \end{pmatrix}$$

$$A^3b + 3Ab + b = \begin{pmatrix} -3 \\ 0 \\ -1 \end{pmatrix} + 3\begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Example

- $\mathbb{F} = \mathbb{F}_5$

$$A = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 0 & 3 \\ 1 & 2 & -1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

- Minimal polynomial for $(A^i\mathbf{b})_{i\geq 0}$ is $m(x) = x^3 + 3x + 1$
- If we picked $\mathbf{u} = (1, 0, 0)^T \in \mathbb{F}_5^3$ we would get the sequence

$$(\mathbf{u}^T A^i \mathbf{b})_{i\geq 0} = (\underbrace{3, 0, 4, 2, 3, 0}_{\text{6 values enough for our algorithm}}, \cdots)$$

Example

- $\mathbb{F} = \mathbb{F}_5$

$$A = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 0 & 3 \\ 1 & 2 & -1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

- Minimal polynomial for $(A^i\mathbf{b})_{i\geq 0}$ is $m(x) = x^3 + 3x + 1$
- If we picked $\mathbf{u} = (1,0,0)^T \in \mathbb{F}_5^3$ we would get the sequence

$$(\mathbf{u}^T A^i \mathbf{b})_{i\geq 0} = (3,0,4,2,3,0,\cdots)$$

- Minimal polynomial for this sequence is $m(x) = x^2 + 2x + 2$

NOT minimal polynomial of $(A^i\mathbf{b})_{i\geq 0}$, but divides it (as it must happen)

$$(x^2+2x+2)(x-2) = x^3 - 2x^2 + 2x^2 - 4x + 2x - 4$$
$$= x^3 + 3x + 1$$

$$A^2b + 2Ab + 2b \neq 0$$

Example

- $\mathbb{F} = \mathbb{F}_5$

$$A = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 0 & 3 \\ 1 & 2 & -1 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

- Minimal polynomial for $(A^i\mathbf{b})_{i\geq 0}$ is $m(x) = x^3 + 3x + 1$
- If we picked $\mathbf{u} = (1,0,0)^T \in \mathbb{F}_5^3$ we would get the sequence

$$(\mathbf{u}^T A^i \mathbf{b})_{i\geq 0} = (3,0,4,2,3,0,\cdots)$$

- Minimal polynomial for this sequence is $m(x) = x^2 + 2x + 2$

  NOT minimal polynomial of $(A^i\mathbf{b})_{i\geq 0}$, but divides it (as it must happen)
- Picking $\mathbf{u} = (1,2,0)^T$, we get minimal polynomial $m(x) = x^3 + 3x + 1$

Probability of Success

- Why would most $\mathbf{u} \in \mathbb{F}^n$ work?
- Assume that the degree of the minimal polynomial of $(A^i\mathbf{b})_{i \geq 0}$ is d

Probability of Success

- Why would most $\mathbf{u} \in \mathbb{F}^n$ work?
- Assume that the degree of the minimal polynomial of $(A^i\mathbf{b})_{i\geq 0}$ is d
- There exists a polynomial $R_{A,\mathbf{b}}(x_1, \ldots, x_n)$ of degree d such that $(\mathbf{u}^T A^i \mathbf{b})_{i\geq 0}$ has the same minimal polynomial as $(A^i\mathbf{b})_{i\geq 0}$ iff

$$\boxed{R_{A,\mathbf{b}}(\mathbf{u}) \neq 0}$$

Probability of Success

- Why would most $\mathbf{u} \in \mathbb{F}^n$ work?
- Assume that the degree of the minimal polynomial of $(A^i\mathbf{b})_{i\geq 0}$ is d
- There exists a polynomial $R_{A,\mathbf{b}}(x_1, \ldots, x_n)$ of degree d such that $(\mathbf{u}^T A^i\mathbf{b})_{i\geq 0}$ has the same minimal polynomial as $(A^i\mathbf{b})_{i\geq 0}$ iff

$$R_{A,\mathbf{b}}(\mathbf{u}) \neq 0$$

- Proof is a bit involved

Probability of Success

- Why would most $\mathbf{u} \in \mathbb{F}^n$ work?
- Assume that the degree of the minimal polynomial of $(A^i\mathbf{b})_{i\geq 0}$ is d
- There exists a non-zero polynomial $R_{A,\mathbf{b}}(x_1, \ldots, x_n)$ of degree d such that $(\mathbf{u}^T A^i \mathbf{b})_{i\geq 0}$ has the same minimal polynomial as $(A^i\mathbf{b})_{i\geq 0}$ iff

$$\text{u is good} \iff R_{A,\mathbf{b}}(\mathbf{u}) \neq 0$$

- Proof is a bit involved
- Thus, by using the Schwarz-Zippel lemma, if $\boxed{|U| > td}$ (tn), then

$$\underbrace{\Pr_{\mathbf{u}\in U^n}[R_{A,\mathbf{b}}(\mathbf{u}) = 0] \leq \frac{d}{|U|}}_{\text{prob. of failure}} = 1/t$$

$\therefore$ Prob. of success in $\geq 1 - \frac{d}{|U|} = 1 - \frac{1}{t}$

Wiedemann's Algorithm: Runtime Analysis

- For Wiedemann's algorithm, all we need to do is:
  1. Compute minimal polynomial of $A^i\mathbf{b}$
  2. Compute $h(x)$ from the minimal polynomial
  3. Use Horner's rule and matrix-vector multiplication to compute $h(A)\mathbf{b}$

Wiedemann's Algorithm: Runtime Analysis

- For Wiedemann's algorithm, all we need to do is:
  1. Compute minimal polynomial of $A^i\mathbf{b}$
  2. Compute $h(x)$ from the minimal polynomial
  3. Use Horner's rule and matrix-vector multiplication to compute $h(A)\mathbf{b}$
- Running time of each step:
  1. Same number of operations as it takes to compute minimal polynomial of $(\mathbf{u}^T A^i \mathbf{b})_{i \geq 0}$

     From last class: $O(M(d) \log d + d \cdot c(A))$

Wiedemann's Algorithm: Runtime Analysis

- For Wiedemann's algorithm, all we need to do is:
  1. Compute minimal polynomial of $A^i\mathbf{b}$
  2. Compute $h(x)$ from the minimal polynomial
  3. Use Horner's rule and matrix-vector multiplication to compute $h(A)\mathbf{b}$
- Running time of each step:
  1. Same number of operations as it takes to compute minimal polynomial of $(\mathbf{u}^T A^i \mathbf{b})_{i\geq 0}$

     From last class: $O(M(d)\log d + d\cdot c(A))$
  2. $O(n)$ time
  3. This takes d matrix-vector multiplications and $O(d)$ vector additions, so running time $O(d\cdot c(A) + dn)$

     Since $d \leq n$, the above is in the worst case $\boxed{O(n\cdot c(A)+n^2)}$
- Much better than n^ω for all the cases that we discussed!

$O\left(\underline{M(n)\log n} + n\cdot c(A) + \underline{n^2}\right)$

Running time of Wiedemann's

Conclusion

- Today we learned about black-box model for linear algebra
- Very useful for linear system solving (ubiquitous in CS and scientific computing, ML, etc!)
- Saw how to use Krylov subspaces to solve linear systems - Wiedemann's algorithm
- Very fast algorithms for special classes of matrices of interest!

References I

von zur Gathen, J. and Gerhard, J. 2013.
Modern Computer Algebra
Cambridge University Press

Chapter 12