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

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

$$Ay = b$$

Naive solution: Gaussian elimination

Running time: $O(\max\{m, n\}^3)$
Problem: given an input matrix $A \in \mathbb{F}^{n \times m}$ and a vector $b \in \mathbb{F}^n$, find a (all) solution(s) $y \in \mathbb{F}^m$ to

$$Ay = b$$

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

$$Ay = b$$

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

$$Ay = b$$

- 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 $Ay = b$
  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 $Ay = b$
  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*
An ubiquitous problem in scientific computing is to solve system of linear equations $Ay = b$.

- Linear programming
- Optimization
- Polynomial multiplication
- Factoring
- Polynomial interpolation (DFT)
- Computing GCD of polynomials (Resultants)
- 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 $c$, we can evaluate $Ac$ 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 $Ay = b$
  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 $c$, we can evaluate $Ac$ 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: $n = 2^k$, $z = e^{2\pi i/n}$.

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

\[
\begin{pmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & z & z^2 & \cdots & z^{n-1} \\
1 & z^2 & z^4 & \cdots & z^{2(n-1)} \\
1 & z^3 & z^6 & \cdots & z^{3(n-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & z^{n-1} & z^{2(n-1)} & \cdots & z^{(n-1)(n-1)}
\end{pmatrix} = A
\]
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 $y = A^{-1}b$, so

  cost to solve the system $= \text{cost to multiply DFT matrix by vector } A^{-1}b$
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 $y = A^{-1}b$, so
  
  cost to solve the system = cost to multiply DFT matrix by vector $A^{-1}b$
- 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 \( b \), we can evaluate \( Ab \) 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.

<table>
<thead>
<tr>
<th>Class of matrices</th>
<th>$c(A)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>general</td>
<td>$2n^2 - 2$</td>
</tr>
<tr>
<td>Sylvester Matrix</td>
<td>$O(M(n))$</td>
</tr>
<tr>
<td>DFT</td>
<td>$O(n \log n)$</td>
</tr>
<tr>
<td>Vandermonde matrix</td>
<td>$O(M(n) \log n)$</td>
</tr>
<tr>
<td>Berlekamp matrix over $\mathbb{F}_q$</td>
<td>$O(M(n) \log q)$</td>
</tr>
<tr>
<td>Sparse matrix with $s$ non-zero entries</td>
<td>$2s$</td>
</tr>
<tr>
<td>Toeplitz matrix</td>
<td>$O(M(n))$</td>
</tr>
</tbody>
</table>
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.

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

- And these matrices appear quite often in practical applications!
input: \( b \rightarrow A \rightarrow Ab \)

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 $b \in \mathbb{F}^n$, find the solution $y \in \mathbb{F}^n$ to $Ay = b$

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

Solution:

Let $p_A(x)$ be the characteristic polynomial of $A$:

$$p_A(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_1x + (-1)^n \det(A)$$

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

Multiplying by $b$, we get:

$$(-1)^n \det(A) \cdot b = A \cdot (A^{n-1} + f_{n-1}A^{n-2} + \cdots + f_1I) \cdot b$$
Wiedemann's Idea

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

\[
Ay = b
\]

- 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 $b \in \mathbb{F}^n$, find the solution $y \in \mathbb{F}^n$ to

$$Ay = b$$

- 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 $A$

$$p_A(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_1x + (-1)^n \det(A)$$

$$p_A(x) = \det(xI - A) = 0$$
Wiedemann’s Idea

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

\[
Ay = b
\]

- 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 \( A \)

\[
p_A(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_1x + (-1)^n \det(A)
\]

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

\[
p_A(A) = 0 = A^n + f_{n-1}A^{n-1} + \cdots + f_1A + (-1)^n \det(A) \cdot I
\]

\[
0 \cdot b = \hat{0} = A^n \hat{b} + f_{n-1}A^{n-1} \hat{b} + \cdots + f_1A \hat{b} + (-1)^n \det(A) \cdot \hat{b}
\]
Wiedemann’s Idea

- **Problem:** given an input invertible matrix \( A \in \mathbb{F}^{n \times n} \) and a vector \( b \in \mathbb{F}^n \), find the solution \( y \in \mathbb{F}^n \) to
  \[
  Ay = b
  \]

- 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 \( A \)
  \[
  p_A(x) = x^n + f_{n-1}x^{n-1} + \cdots + f_1x + (-1)^n \det(A)
  \]
- By Cayley-Hamilton, we know that \( p_A(A) = 0 \)
  \[
  p_A(A) = 0 = A^n + f_{n-1}A^{n-1} + \cdots + f_1A + (-1)^n \det(A) \cdot I
  \]
- Multiplying by \( b \), we get:
  \[
  (-1)^{n+1} \det(A) \cdot b = A \cdot (A^{n-1} + f_{n-1}A^{n-2} + \cdots + f_1I) \cdot b
  \]
Wiedemann’s Idea

\[ (-1)^{n+1} \cdot \det(A) \cdot \hat{b} = A \cdot M \cdot \hat{b} \]

\[ (-1)^{n+1} \cdot \det(A) \cdot A^{-1} \hat{b} = M \cdot \hat{b} \]

\[ \frac{M}{(-1)^{n+1} \det(A)} = A^{-1} \]

\[ M \text{ can be computed from characteristic polynomial of } A. \]
Wiedemann’s Algorithm

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

$$Ay = b$$

- $A$'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 $b \in \mathbb{F}^n$, find the solution $y \in \mathbb{F}^n$ to

  $$Ay = b$$

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

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

  $$m(x) \mid p_A(x)$$

  **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 \( b \in \mathbb{F}^n \), find the solution \( y \in \mathbb{F}^n \) to

\[ Ay = b \]

\( A \)'s characteristic polynomial can be used to compute the inverse!

If we pay closer attention, *any annihilating polynomial* of the Krylov sequence \( (A^i b)_{i \geq 0} \) works!

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

Wiedemann’s Algorithm: on input \( A \in \mathbb{F}^{n \times n} \) invertible and \( b \in \mathbb{F}^n \)

1. Compute the minimal polynomial \( m(x) \) of the Krylov sequence \( (A^i b)_{i \geq 0} \)

\[ m(x) = m_d x^d + \cdots + m_1 x + m_0 \]
Wiedemann’s Algorithm

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

$$Ay = b$$

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

Wiedemann’s Algorithm: on input $A \in \mathbb{F}^{n \times n}$ invertible and $b \in \mathbb{F}^n$

1. Compute the minimal polynomial $m(x)$ of the Krylov sequence $(A^i b)_{i \geq 0}$

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

2. compute $h(x) = -\frac{m(x) - m_0}{m_0 \cdot x}$ \{ well-defined m.t.o \}

3. compute $y = h(A) \cdot b$ using Horner’s rule \{ and block bu evaluation \}
Wiedemann’s Algorithm

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

\[ Ay = b \]

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

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

\[ b, A b, A^2 b, \ldots, A^d b \]

\[ O = A^d b + m_{d-1} A^{d-1} b + \ldots + m_1 A b + m_0 b \]

\[ -m_0 b = A \left( A^{d-1} b + m_{d-1} A^{d-2} b + \ldots + m_1 b \right) \]

\[ \chi^{d-1} + m_{d-1} \chi^{d-2} + \ldots + m_2 \chi + m_1 = m(x) - m_0 \]

\[ \chi \]

\[ \frac{m(x) - m_0}{-m_0 \chi} = h(x) \]

\[ A^{-1} b = \frac{1}{-m_0} \left( A^{d-1} b + m_{d-1} A^{d-2} b + \ldots + m_1 b \right) \]

\[ h(A) \cdot b \]
Wiedemann’s Algorithm: Correctness

- $A$ is invertible, then $p_A(0) = (-1)^n \det(A) \neq 0$
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 $m_0 \neq 0$

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 \( m_0 \neq 0 \)
  
  Thus, we can always compute \( h(x) = -\frac{m(x) - m_0}{m_0x} \)

- \( m(x) \) is minimal polynomial then
  
  \[
  A^d b + m_{d-1} A^{d-1} b + \cdots + m_1 A b + m_0 b = 0
  \]
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 $m_0 \neq 0$

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

- $m(x)$ is minimal polynomial then

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

- Rearranging, we get

  $$h(A) \cdot b = A^{-1} b$$
Wiedemann’s Algorithm: Runtime Analysis

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

Let's go through the steps of the algorithm:

1. **Compute minimal polynomial of \( (A^i b)_{i \geq 0} \)**
   - This step involves finding the minimal polynomial of the sequence \( (A^i b) \).
   - The minimal polynomial is the polynomial of least degree that annihilates the sequence.

2. **Compute \( h(x) \) from the minimal polynomial**
   - Once the minimal polynomial is found, \( h(x) \) is derived from it.
   - The degree of \( h(x) \) is equal to the degree of the minimal polynomial.

3. **Use Horner’s rule and matrix-vector multiplication to compute \( h(A)b \)**
   - Horner’s rule is applied to evaluate \( h(x) \) at \( A \).
   - Then, matrix-vector multiplication is used to compute \( h(A)b \).

The running time of each step is as follows:

- **Step 1**: \( O(n) \) time
- **Step 2**: This step involves \( d \) matrix-vector multiplications and \( O(d) \) vector additions.
  - Running time: \( O(d \cdot c(A) + dn) \)

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

This runtime is much better than \( n \omega \) for all the cases we discussed!

However, only if we can compute the minimal polynomial really fast...
Wiedemann’s Algorithm: Runtime Analysis

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

- 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 $O(n \cdot c(A) + n^2)$

$d := \text{degree of minimal polynomial of } (A^i b)_{i \geq 0}$. 
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 $n^\omega$ 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 $A^i b$
  2. Compute $h(x)$ from the minimal polynomial
  3. Use Horner’s rule and matrix-vector multiplication to compute $h(A)b$

- 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 $n^\omega$ for all the cases that we discussed!

- Well, only if we can compute the minimal polynomial really fast...
• Administrivia

• Black-Box Linear Algebra

• Wiedemann’s Algorithm

• Computing Minimal Polynomials of Krylov Sequences

• Conclusion
Minimal Polynomials of Krylov Subspaces

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

\[(a_i)_{i \geq 0}, \quad a_i \in F\]

---

1. If $F$ 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 $u \in \mathbb{F}^n$, the sequences $(A^i b)_{i \geq 0}$, and $(u^T A^i b)_{i \geq 0}$ have the same minimal polynomial with high probability!\(^1\)

\[^1\text{If } \mathbb{F} \text{ has enough elements.}\]
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
  \[ (A^i \mathbf{b})_{i \geq 0}, \quad \text{and} \quad (\mathbf{u}^T A^i \mathbf{b})_{i \geq 0} \]
  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* $u \in \mathbb{F}^n$, the sequences $(A^i b)_{i \geq 0}$, and $(u^T A^i b)_{i \geq 0}$ have the *same minimal polynomial* with high probability!\(^1\)

**Algorithm:** input $A \in \mathbb{F}^{n \times n}$, $b \in \mathbb{F}^n$

1. If $b = 0$, return 1
2. $u \in U^n$ 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 $F^n$. How to compute minimal polynomials in this case?
- Idea: convert sequences in $F^n$ to sequences over $F$!
- Key Lemma: Given a random $u \in F^n$, the sequences $(A^i b)_{i \geq 0}$, and $(u^T A^i b)_{i \geq 0}$ have the same minimal polynomial with high probability!\(^1\)
- Algorithm: input $A \in F^{n \times n}$, $b \in F^n$
  1. If $b = 0$, return 1
  2. $u \in U^n$ uniformly at random, $U \subset F$ is a large enough finite set
  3. Use algorithm from previous lecture to compute $m(x)$ minimal polynomial of $(u^T A^i b)_{i \geq 0}$

---

\(^1\)If $F$ 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 $u \in \mathbb{F}^n$, the sequences $(A^i b)_{i \geq 0}$ and $(u^T A^i b)_{i \geq 0}$ have the same minimal polynomial with high probability!\(^1\)

**Algorithm:** input $A \in \mathbb{F}^{n \times n}$, $b \in \mathbb{F}^n$

1. If $b = 0$, return 1
2. $u \in U^n$ 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 $u^T A^i b$.
4. If $m(A)b = 0$ return $m(x)$, else return to step (2)

\(^1\)If $\mathbb{F}$ has enough elements.
\((A^i)_{i \geq 0}\)

\(A \in \mathbb{F}^{n \times n}\)

\(A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}\)

\((x-1)(x-2)(x-3) = p_A(x)\)

\(\overset{\text{minimal polynomial}}{\Rightarrow}\)

\((A^i 0)_{i \geq 0} = (0)_{i \geq 0}\)

\(\overset{\text{minimal polynomial}}{\Rightarrow}\ 1\)

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

\(\overset{\text{minimal polynomial}}{\Rightarrow}\ x-1\)

\((m(x)) \in p_A(x)\)
Example

\[ F = \mathbb{F}_5 \]

\[ A = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 0 & 3 \\ 1 & 2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix} \]
Example

- $F = \mathbb{F}_5$

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

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

$$\begin{align*}
A b &= \begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix} \\
A^2 b &= A(\lambda b) = \begin{pmatrix} -1 \\ 3 \\ 3 \end{pmatrix} \\
A^3 b &= A(A^2 b) = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} \\
A^4 b + 3A^3 b + b &= \begin{pmatrix} -3 \\ 0 \\ -1 \end{pmatrix} + 3 \begin{pmatrix} 0 \\ 3 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}
\end{align*}$$
Example

- $F = \mathbb{F}_5$

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

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

- If we picked $u = (1, 0, 0)^T \in \mathbb{F}_5^3$ we would get the sequence $(u^T A^i b)_{i \geq 0} = (3, 0, 4, 2, 3, 0, \cdots)$

\[\text{\underline{Note:}}\text{ value enough for an algorithm}\]
Example

- $\mathbb{F} = \mathbb{F}_5$
  
  $$A = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 0 & 3 \\ 1 & 2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}$$

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

- If we picked $u = (1, 0, 0)^T \in \mathbb{F}_5^3$ we would get the sequence
  
  $$(u^T A^i b)_{i \geq 0} = (3, 0, 4, 2, 3, 0, \cdots)$$

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

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

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

  $$= x^3 + 3x + 1$$
$A^2b + 2AAb + 2b \neq 0$
Example

- \( F = \mathbb{F}_5 \)

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

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

\[
(u^T A^i b)_{i \geq 0} = (3, 0, 4, 2, 3, 0, \ldots)
\]

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

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

- Picking \(u = (1, 2, 0)^T\), we get minimal polynomial \(m(x) = x^3 + 3x + 1\)
Probability of Success

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

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

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

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

- Proof is a bit involved
Probability of Success

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

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

- Proof is a bit involved
- Thus, by using the Schwarz-Zippel lemma, if $|U| > td$, then

$$\Pr_{u \in U^n}[R_{A,b}(u) = 0] \leq \frac{d}{|U|} = \frac{1}{t}$$

Prob. of failure $\geq 1 - \frac{1}{t}$
For Wiedemann’s algorithm, all we need to do is:

1. Compute minimal polynomial of $A^i b$
2. Compute $h(x)$ from the minimal polynomial
3. Use Horner’s rule and matrix-vector multiplication to compute $h(A)b$

Running time of each step:
1. Same number of operations as it takes to compute minimal polynomial of $(u^T A^i b)$
   - 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 $O(n \cdot c(A) + n^2)$

Much better than $n^{\omega}$ 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 $A^i b$
  2. Compute $h(x)$ from the minimal polynomial
  3. Use Horner’s rule and matrix-vector multiplication to compute $h(A)b$

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

- Running time of each step:
  1. Same number of operations as it takes to compute minimal polynomial of $(u^T A^i b)_{i \geq 0}$
     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 $O(n \cdot c(A) + n^2)$

- Much better than $n^\omega$ for all the cases that we discussed!
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!
von zur Gathen, J. and Gerhard, J. 2013.
Modern Computer Algebra
Cambridge University Press

Chapter 12