Linear Algebra on a Computer
W. J. Turner
Symbolic Computation
Symbolic Algorithms
Black Box Matrix Model
Wiedemann Method
Background
The Method

Linear Algebra on a Computer

An Introduction to Black Box Methods

William J. Turner

Department of Mathematics & Computer Science

6 February 2007

Outline

1. Symbolic Computation
2. Symbolic Algorithms
3. Black Box Matrix Model
4. Wiedemann Method
   - Background
   - The Method

Symbolic Computation

Superset of computer algebra

- Symbols or exact arithmetic
- Based on exact finite representation of finite or infinite mathematical objects
- Abstract mathematical structures (groups, rings, fields, etc.)

Computar Algebra Systems

General Purpose Systems

- AXIOM, MAGMA, Maple, *Mathematica*, REDUCE

Special Purpose Systems

- CoCoA (Computations in Commutative Algebra)
- GAP (Groups, Algorithms, and Programming)
- NTL (Number Theory Library)
- SINGULAR (polynomial computations)

Long History

Ancient Algorithms

- Euclidean algorithm for finding the greatest common divisor
- Chinese remainder algorithm

Isaac Newton's *The Universal Arithmetic* (1728)

Systematically discusses rules for manipulating universal mathematical expressions, that is, formulae containing symbolic indeterminates, and algorithms for solving equations built with these expressions.

Symbolic Computation vs. Numerical Analysis

Numerical Analysis

- Floating point numbers (approximate real values)
- Find approximation quickly
- Error propogation important

  Condition Numbers: amplification factors of relative errors

  Stability: whether all roundoff errors of an algorithm are harmless

Symbolic Computation

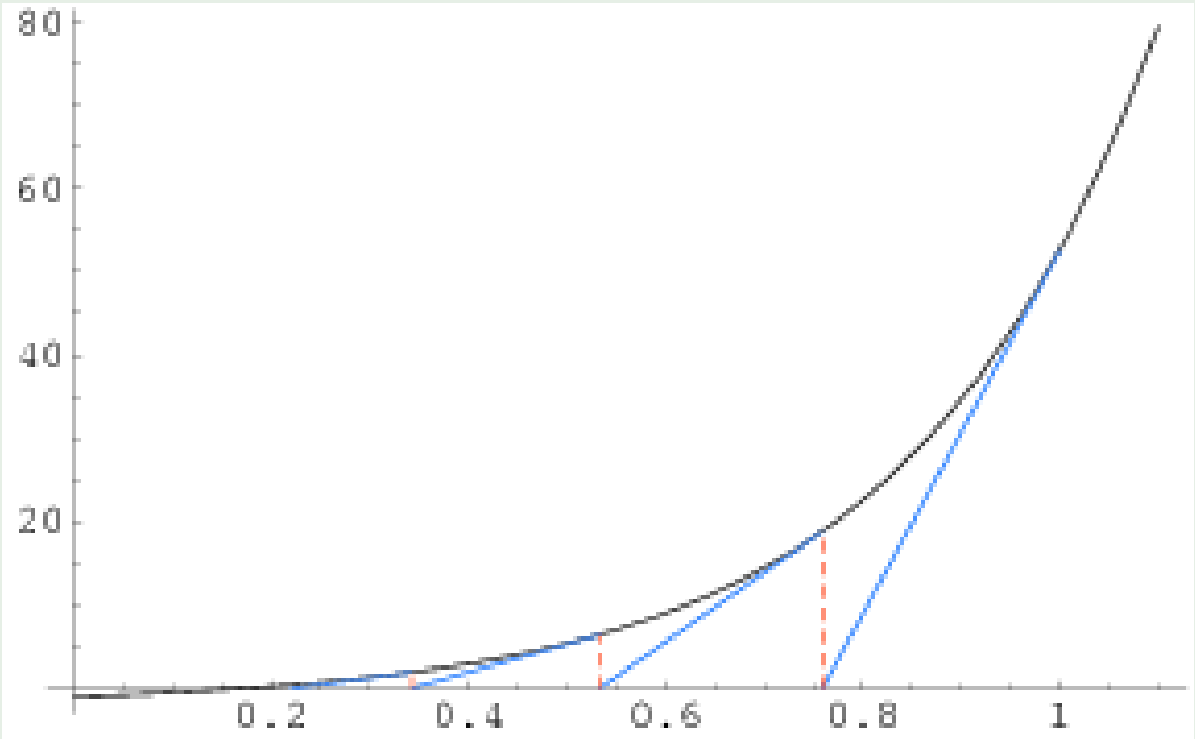
- Find exact solution quickly
- May never approximate (may have no metric)

Symbolic Computation vs. Numerical Analysis

Algorithms may not be compatible

- Numerical algorithms may never find exact solution
- Symbolic algorithms may be ill conditioned or unstable

Example (Newton's Method)



Infinite Mathematical Objects

General Approach

- Computer has finite memory
- Cannot compute exactly over reals, rationals, integers, etc.
- Compute bound on desired solution
- Use finite field methods to find solution modulo p_i
- Reconstruct solution (e.g., Chinese remainder algorithm)

Probabilistic Algorithms

Monte Carlo Algorithm

- Always fast; Probably correct

Las Vegas Algorithm

- Probably fast; Always correct

Bounded-error, Probabilistic, Polynomial time (BPP)

- Probably fast; Probably correct
- Atlantic City?

Probabilistic Algorithms

Schwartz-Zippel Lemma

Let R be an integeral domain, $n \in \mathbb{N}$, $S \subset R$ finite with $s = |S|$ elements, and $f \in R[\lambda_1, \ldots, \lambda_n]$ a polynomial of total degree at most $d \in \mathbb{N}$.

1. If f is not the zero polynomial, then f has at most ds^{n-1} zeros in S^n.
2. If $s > d$ and f vanishes on S^n, then $f = 0$.

Converting Algorithms

Monte Carlo to Las Vegas

- Requires certificate to check if solution is correct
- If not correct, run Monte Carlo Algorithm again

Las Vegas to Monte Carlo

- Start Las Vegas algorithm and let run for set period of time
- Will stop early with given probability p
  - Solution guaranteed to be correct
  - Return correct answer
- Will not stop early with probability $1 - p$
  - Halt algorithm
  - Return some answer
  - Need not be correct

Problems of Interest in Symbolic Linear Algebra

Solving a Linear System of Equations

- Solve $Ax = b$ where $A \in \mathbb{F}^{n \times n}$ and $b \in \mathbb{F}^n$
- A nonsingular or singular

Matrix Determinant

- $\det(A)$ where $A \in \mathbb{F}^{n \times n}$
- A nonsingular

Matrix Rank

- $\text{rank}(A)$ where $A \in \mathbb{F}^{n \times n}$
- A singular

Solving Linear Systems

Linear Algebra on a Computer

W. J. Turner

Symbolic Computation

Symbolic Algorithms

Black Box Matrix Model

Wiedemann Method

Background

The Method

Example (Nonsingular system)

$$\begin{bmatrix} 1 & 3 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 3 & 3 & -2 & 3 \\ 0 & -3 & 0 & -2 \end{bmatrix} x = \begin{bmatrix} 4 \\ 2 \\ 6 \\ -4 \end{bmatrix}$$

Gaussian Elimination

Example

$$\begin{bmatrix} 1 & 3 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -2 & -6 \\ 0 & 0 & 0 & -2 \end{bmatrix} x = \begin{bmatrix} 4 \\ 2 \\ 6 \\ 2 \end{bmatrix} \implies x = \begin{bmatrix} 1 \\ 2 \\ 0 \\ -1 \end{bmatrix}$$

Gaussian Elimination

- Row echelon form and back substitution
- Must know how matrix stored
- Matrix changed in calculation
- Sparse matrices may become dense

Black Box Matrix Model

Linear Algebra on a Computer
W. J. Turner
Symbolic Computation
Symbolic Algorithms
Black Box Matrix Model
Wiedemann Method
Background
The Method

Black Box Matrix Model

- External view of matrix
- Only matrix-vector products allowed
- Independent of implementation
- Implementations may be efficient in time or space
- Not unique to symbolic computation
- Can compute Krylov sequence $(A^i v)_{i \in \mathbb{Z}_{\geq 0}}$

Possible Black Box Matrices

Linear Algebra on a Computer
W. J. Turner
Symbolic Computation
Symbolic Algorithms
Black Box Matrix Model
Wiedemann Method
Background
The Method

Arbitrary Matrix

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,n} \end{bmatrix}$$

Storage: n^2 (store every entry)

Time: $O(n^2)$

Sparse Matrix

Only η nonzero entries

Storage: $O(\eta)$ (store nonzero entries and indices)

Time: $O(\eta)$

Possible Black Box Matrices

Hilbert Matrix

$$A = \begin{bmatrix} \frac{1}{1} & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+1} & \cdots & \frac{1}{2n} \end{bmatrix}$$

Storage: $O(1)$ (store only dimension n)

Time: $O(n)$

Possible Black Box Matrices

Toeplitz Matrix

$$A = \begin{bmatrix} a_n & a_{n+1} & a_{n+2} & \cdots & a_{2n-1} \\ a_{n-1} & a_n & a_{n+1} & \cdots & a_{2n-2} \\ a_{n-2} & a_{n-1} & a_n & \cdots & a_{2n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_n \end{bmatrix}$$

Storage: $O(n)$

Time: $O(n \log(n) \operatorname{loglog}(n))$ (implement via fast polynomial multiplication)

Linearly Recurrent Sequences

Definition

- Let $\mathbb{V}$ be a vector space over field $\mathbb{F}$.
- A sequence

$$a = (a_i)_{i \in \mathbb{Z}_{\geq 0}} \in \mathbb{V}^{\mathbb{Z}_{\geq 0}}$$

is *linearly recurrent* if and only if there exist $m \in \mathbb{Z}_{\geq 0}$ and

$$c_0, \ldots, c_m \in \mathbb{F}, \quad c_m \neq 0$$

such that for all $i \geq 0$

$$\sum_{j=0}^{m} c_j \; a_{i+j} = 0 \text{ or } a_{i+m} = -\frac{1}{c_m} \left(\sum_{j=0}^{m-1} c_j \; a_{i+j} \right)$$

Linearly Recurrent Sequences

Example (Fibonacci Numbers)

- $(a_i)_{i \in \mathbb{Z}_{\geq 0}} = (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \dots)$
- $\mathbb{V} = \mathbb{F} = \mathbb{R}$
- $a_{i+2} = a_{i+1} + a_i$
- $a_{i+2} - a_{i+1} - a_i = 0$

Generating Polynomials

Definition

The polynomial $f(\lambda) = \sum_{j=0}^{m} c_j \ \lambda^j$ generates a

Example (Fibonacci Numbers)

- $(a_i)_{i \in \mathbb{Z}_{\geq 0}} = (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \dots)$
- $a_{i+2} - a_{i+1} - a_i = 0$
- $f = \lambda^2 - \lambda - 1$ generates the Fibonacci sequence a

Generating Polynomials

Module

- Define $f \bullet a = \left(\sum_{j=0}^{m} c_j \; a_{i+j} \right)_{i \in \mathbb{Z}_{\geq 0}} = (0)_{i \in \mathbb{Z}_{\geq 0}} = 0 \in \mathbb{V}^{\mathbb{Z}_{\geq 0}}$
- $\mathbb{V}^{\mathbb{Z}_{\geq 0}}$ is an $\mathbb{F}[\lambda]$-module
- If $f \bullet a = 0$ and $g \in \mathbb{F}[\lambda]$, then

$$(g \; f) \bullet a = g \bullet (f \bullet a) = g \bullet 0 = 0$$

Generating Polynomials

Example (Fibonacci Numbers)

- $(a_i)_{i \in \mathbb{Z}_{\geq 0}} = (0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \dots)$
- $f = \lambda^2 - \lambda - 1$ generates the Fibonacci sequence a
- $(\lambda + 1)f = \lambda^3 - 2\lambda - 1$ also generates a
  - $a_{i+3} - 2a_{i+1} - a_i = 0$
  - $a_{i+3} = 2a_{i+1} + a_i$
- $\lambda^k f = \lambda^{k+2} - \lambda^{k+1} - \lambda^k$ also generates a
  - $a_{i+k+2} - a_{i+k+1} - a_{i+k} = 0$
  - $a_{i+k+2} = a_{i+k+1} + a_{i+k}$
  - Skips first k elements of a

Minimal Generating Polynomial

Existence

- $\{f \in \mathbb{F}[\lambda] \mid f \bullet a = 0\}$ is an ideal
- $\mathbb{F}[\lambda]$ is a principal ideal domain
- There exists a unique monic generator of minimal degree, the *minimal generating polynomial* of sequence
- Minimal polynomial divides all generating polynomials

Example (Fibonacci Numbers)

The polynomial $f = \lambda^2 - \lambda - 1$ is the minimal generating polynomial of the Fibonacci sequence a.

Matrix Power Sequence

Matrix Power Sequence

- $f \bullet (A^i)_{i \in \mathbb{Z}_{\geq 0}}$ if and only if $f(A) = 0$
- $\det(\lambda I - A)$ generates $(A^i)_{i \in \mathbb{Z}_{\geq 0}}$
  - Cayley-Hamilton Theorem
- Let f^A be the minimal polynomial of $(A^i)_{i \in \mathbb{Z}_{\geq 0}}$.
  - $f^A \mid \det(\lambda I - A)$
  - $\deg(f^A) \leq \deg(\det(\lambda I - A)) = n$

Krylov Sequence

Krylov Sequence

- $f \bullet (A^i v)_{i \in \mathbb{Z}_{\geq 0}}$ if and only if $f(A)\ v = 0$
- $f(A) = 0$ means $f(A)\ v = 0$
- $\{f \mid f \bullet (A^i)_{i \in \mathbb{Z}_{\geq 0}} = 0\} \subset \{f \mid f \bullet (A^i v)_{i \in \mathbb{Z}_{\geq 0}} = 0\}$
- Let $f^{A,v}$ be the minimal polynomial of $(A^i v)_{i \in \mathbb{Z}_{\geq 0}}$.
  - $f^{A,v} \mid f^A \mid \det(\lambda I - A)$
  - $\deg(f^{A,v}) \leq \deg(f^A) \leq \deg(\det(\lambda I - A)) = n$

Solving a Linear System

Using Linear Recurrences to Solve a Linear System

- Suppose $g = \sum_{j=0}^{m} c_j \ \lambda^j$, $g(0) = c_0 \neq 0$, and $g \bullet (A^i b)_{i \in \mathbb{Z}_{\geq 0}}$
- If g exists, then $f^{A,b}$ satisfies the requirements.
- Consider linear combination
  $c_0 \ b + c_1 \ A \ b + \cdots + c_m \ A^m \ b = 0$
- $b = -\frac{1}{c_0} \sum_{j=1}^{m} (A^j \ b) = A \left(-\frac{1}{c_0} \sum_{j=1}^{m} (c_j \ A^{j-1} \ b) \right)$
- $x = -\frac{1}{c_0} \sum_{j=1}^{m} (c_j \ A^{j-1} \ b)$ is a solution

Nonsingular Matrix

Nonsingular A

- $\det(A) \neq 0$
- $\det(\lambda I - A)|_{\lambda=0} \neq 0$
- $f^{A,b}(0) \neq 0$
- $x = -\frac{1}{c_0} \sum_{j=1}^{m} (c_j \ A^{j-1} \ b)$ is the unique solution

Computing the Minimal Polynomial of Krylov Sequence

Bilinear Projection Sequence

- $f \bullet (u^{\mathsf{T}} A^i v)_{i \in \mathbb{Z}_{\geq 0}}$ if and only if $u^{\mathsf{T}}\ f(A)\ v = 0$
- $f(A)\ v = 0$ means $u^{\mathsf{T}}\ f(A)\ v = 0$
- $\{f \mid f \bullet (A^i v)_{i \in \mathbb{Z}_{\geq 0}} = 0\} \subset \{f \mid f \bullet (u^{\mathsf{T}} A^i v)_{i \in \mathbb{Z}_{\geq 0}} = 0\}$
- Let $f_u^{A,v}$ be the minimal polynomial of $(u^{\mathsf{T}} A^i v)_{i \in \mathbb{Z}_{\geq 0}}$.
  - $f_u^{A,v} \mid f^{A,v} \mid f^A \mid \det(\lambda I - A)$
  - $\deg(f_u^{A,v}) \leq \deg(f^{A,v}) \leq \deg(f^A) \leq \deg(\det(\lambda I - A)) = n$

Computing the Minimal Polynomial of Krylov Sequence

Linear Algebra on a Computer

W. J. Turner

Symbolic Computation

Symbolic Algorithms

Black Box Matrix Model

Wiedemann Method

Background

The Method

Minimal Polynomials Probably Equal

If u chosen randomly from finite $S^n \subset \mathbb{F}^n$, then $f_u^{A,v} = f^{A,v}$ with probability at least

$$1 - \frac{\deg(f^{A,v})}{|S|}$$

Certificate

$$f_u^{A,v} \bullet (A^i v)_{i \in \mathbb{Z}_{\geq 0}} = 0$$

Computing the Minimal Polynomial of Scalar Sequence

Degree Bound

Must know bound $M \geq \deg(f)$

Extended Euclidean Algorithm

- $s_j\ f_{-1} + t_j\ f_0 = f_j$
- Inputs are $f_{-1} = \lambda^{2M}$ and $f_0 = \sum_{i=0}^{2M-1}(a_i\ \lambda^i)$
- Stop when $\deg(f_j) \leq M < \deg(f_{j-1})$
- Minimal polynomial is reversal of $t_j(\lambda)$

Berlekamp-Massey Algorithm

- From coding theory
- Interpolates elements of scalar sequence
- Related to Extended Euclidean Algorithm

Wiedemann's Algorithm

Degree Bounds

- $f_u^{A,v} \mid f^{A,v} \mid f^A \mid \det(\lambda I - A)$
- $\deg(f_u^{A,v}) \leq \deg(\det(\lambda I - A)) = n$
- Only need $(u^T A^i b)_{i=0}^{2n-1}$

Wiedemann's Algorithm

Require: nonsingular $A \in \mathbb{F}^{n \times n}$ and $b \in \mathbb{F}^n$

Ensure: $x \in \mathbb{F}^n$ such that $A\,x = b$

1: $u \leftarrow$ random vector in S^n where $S \subset \mathbb{F}$

2: use Berlekamp-Massey to compute $f^{A,b} = c_0 + c_1\lambda + \cdots + c_m\lambda^m$ {Store only $A^i b$}

3: $x \leftarrow -\frac{1}{c_0}(c_1 b + c_2 A b + \cdots c_m A^{m-1} b)$

Back to original problem

Example

- $\begin{bmatrix} 1 & 3 & 0 & 3 \\ 0 & 1 & 0 & 0 \\ 3 & 3 & -2 & 3 \\ 0 & -3 & 0 & -2 \end{bmatrix} x = \begin{bmatrix} 4 \\ 2 \\ 6 \\ -4 \end{bmatrix}$
- $n = 4$
- $u^{\mathsf{T}} = \begin{bmatrix} 2 & 1 & 1 & 2 \end{bmatrix}$
- $(u^{\mathsf{T}} A^i b)_{i=0}^{2n-1} = (8, -4, 20, -28, 68, -124, 260, -508)$
- $f^{A,b} = f_u^{A,b} = \lambda^2 + \lambda - 2$
- $x = \frac{1}{2}(Ab + b) = \frac{1}{2}\left(\begin{bmatrix} -2 \\ 2 \\ -6 \\ 2 \end{bmatrix} + \begin{bmatrix} 4 \\ 2 \\ 6 \\ -4 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 2 \\ 0 \\ -1 \end{bmatrix}$