Analysis of Algorithms

Matrix algorithms

Andres Mendez-Vazquez

November 24, 2015

Outline

Outline

Cinvestav

Basic definitions

A matrix is a rectangular array of numbers

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

A transpose matrix is the matrix obtained by exchanging the rows and columns

$$A^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

Cinvestav

Basic definitions

A matrix is a rectangular array of numbers

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

A transpose matrix is the matrix obtained by exchanging the rows and columns

$$A^T = \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix}$$

Outline

Cinvestav

Several cases of matrices

Zero matrix

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The diagonal matrix

$$\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

Several cases of matrices

Zero matrix

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The diagonal matrix

$$\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

Several cases of matrices

Upper triangular matrix

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}$$

Cinvestav

Outline

Cinvestav

Operations on matrices

They Define a Vectorial Space

- Matrix addition.
- Multiplication by scalar.
- The existence of zero.

Operations on matrices

They Define a Vectorial Space

- Matrix addition.
- Multiplication by scalar.
- The existence of zero.

Operations on matrices

They Define a Vectorial Space

- Matrix addition.
- Multiplication by scalar.
- The existence of zero.

Outline

Cinvestav

Matrix Multiplication

What is Matrix Multiplication?

Given A, B matrices with dimensions $n \times n$, the multiplication is defined as

$$
\begin{aligned}
C &= AB \\
c_{ij} &= \sum_{k=1}^{n} a_{ik} b_{kj}
\end{aligned}
$$

Cinvestav

Complexity and Algorithm

Algorithm: Complexity $\Theta\left(n^3\right)$

Square-Matrix-Multiply(A, B)

$n = A.rows$
let C be a new matrix $n \times n$
for $i = 1$ to n
    for $j = 1$ to n
        $C\left[i, j\right] = 0$
        for $k = 1$ to n
            $C\left[i, j\right] = C\left[i, j\right] + A\left[i, j\right] * B\left[i, j\right]$
return C

Matrix multiplication properties

Properties of the Multiplication

- The Identity exist for a matrix A of $m \times n$:

$$I_m A = A I_n = A.$$

- The multiplication is associative:

$$A(BC) = (AB)C.$$

In addition, multiplication is distibutive

- $A(B+C) = AB + AC$
- $(B+C)D = BD + CD$

Cinvestav

Matrix multiplication properties

Properties of the Multiplication

- The Identity exist for a matrix A of $m \times n$:

$$I_m A = A I_n = A.$$

- The multiplication is associative:

$$A(BC) = (AB)C.$$

In addition, multiplication is distibutive

- $A(B+C) = AB + AC$
- $(B+C)D = BD + CD$

Matrix multiplication properties

Properties of the Multiplication

- The Identity exist for a matrix A of $m \times n$:

$$I_m A = A I_n = A.$$

- The multiplication is associative:

$$A(BC) = (AB)C.$$

In addition, multiplication is distibutive

- $A(B + C) = AB + AC$
- $(B + C)D = BD + CD$

Cinvestav

In addition

Definition

The inner product between vectors is defied as

$$x^T y = \sum_{i=1}^{n} x_i y_i$$

Outline

Cinvestav

Matrix inverses

The inverse is defined as the vector A^{-1} such that

$$AA^{-1} = A^{-1}A = I_n$$

Example

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1\cdot 0 + 1 \cdot 1 & 1 \cdot 1 - 1 \cdot 1 \\ 1 \cdot 0 + 1 \cdot 0 & 1 \cdot 1 + 0 \cdot -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Remark

A matrix that is invertible is called non-singular.

Matrix inverses

The inverse is defined as the vector A^{-1} such that

$$AA^{-1} = A^{-1}A = I_n$$

Example

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} = \begin{pmatrix} 1\cdot 0 + 1\cdot 1 & 1\cdot 1 - 1\cdot 1 \\ 1\cdot 0 + 1\cdot 0 & 1\cdot 1 + 0\cdot -1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Remark

A matrix that is invertible is called non-singular.

Cinvestav

Matrix inverses

The inverse is defined as the vector A^{-1} such that

$$AA^{-1} = A^{-1}A = I_n$$

Example

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \Longrightarrow \left(\begin{array}{cc} 1 & 1 \\ 1 & 0 \end{array} \right) \left(\begin{array}{cc} 0 & 1 \\ 1 & -1 \end{array} \right) = \left(\begin{array}{cc} 1 \cdot 0 + 1 \cdot 1 & 1 \cdot 1 - 1 \cdot 1 \\ 1 \cdot 0 + 1 \cdot 0 & 1 \cdot 1 + 0 \cdot -1 \end{array} \right) = \left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array} \right)$$

Remark

A matrix that is invertible is called non-singular.

Cinvestav

Properties of an inverse

Some properties are

- $(BA)^{-1} = A^{-1}B^{-1}$
- $\left(A^{-1}\right)^{T} = \left(A^{T}\right)^{-1}$

The Rank of A

Rank of A

A collection of vectors is $x_1, x_2, ..., x_n$ such that $c_1x_1 + c_2x_2 + ... + c_nx_n \neq 0$. The rank of a matrix is the number of linear independent rows.

Theorem 1

A square matrix has full rank if and only if it is nonsingular.

The Rank of A

Rank of A

A collection of vectors is $x_1, x_2, ..., x_n$ such that $c_1 x_1 + c_2 x_2 + ... + c_n x_n \neq 0$. The rank of a matrix is the number of linear independent rows.

Theorem 1

A square matrix has full rank if and only if it is nonsingular.

Other Theorems

A null vector x is such that $Ax = 0$

- Theorem 2: A matrix A has full column rank if and only if it does not have a null vector.

Then, for squared matrices, we have

- Corollary 3: A square matrix A is singular if and only if it has a null vector.

Other Theorems

A null vector x is such that $Ax = 0$

- Theorem 2: A matrix A has full column rank if and only if it does not have a null vector.

Then, for squared matrices, we have

- Corollary 3: A square matrix A is singular if and only if it has a null vector.

Cinvestav

Outline

Determinants

A determinant can be defined recursively as follows

$$det(A) = \begin{cases} a_1 1 & \text{if } n = 1 \\ \sum_{j=1}^{n} (-1)^{1+j} a_{1j} det\left(A_{[1j]}\right) & \text{if } n > 1 \end{cases} \quad (1)$$

Where $(-1)^{i+j} det\left(A_{[ij]}\right)$ is called a cofactor and $A_{[1j]}$ is the matrix formed when eliminating row 1 and column j from A

Determinants

A determinant can be defined recursively as follows

$$det(A) = \begin{cases} a_1 1 & \text{if } n = 1 \\ \sum_{j=1}^{n} (-1)^{1+j} a_{1j} det\left(A_{[1j]}\right) & \text{if } n > 1 \end{cases} \quad (1)$$

Where $(-1)^{i+j} det\left(A_{[ij]}\right)$ is called a cofactor and $A_{[1j]}$ is the matrix formed when eliminating row 1 and column j from A

Cinvestav

Theorems

Theorem 4(determinant properties).

The determinant of a square matrix A has the following properties:

- If any row or any column A is zero, then $det(A) = 0$.
- The determinant of A is multiplied by λ if the entries of any one row (or any one column) of A are all multiplied by λ.
- The determinant of A is unchanged if the entries in one row (respectively, column) are added to those in another row (respectively, column).
- The determinant of A equals the determinant of A^T.
- The determinant of A is multiplied by -1 if any two rows (or any two columns) are exchanged.

Theorems

Theorem 4(determinant properties).

The determinant of a square matrix A has the following properties:

- If any row or any column A is zero, then $det(A) = 0$.
- The determinant of A is multiplied by λ if the entries of any one row (or any one column) of A are all multiplied by λ.
- The determinant of A is unchanged if the entries in one row (respectively, column) are added to those in another row (respectively, column).
- The determinant of A equals the determinant of A^T.
- The determinant of A is multiplied by -1 if any two rows (or any two columns) are exchanged.

Theorem 5

An $n \times n$ matrix A is singular if and only if $det(A) = 0$.

Theorems

Theorem 4(determinant properties).

The determinant of a square matrix A has the following properties:

- If any row or any column A is zero, then $det(A) = 0$.
- The determinant of A is multiplied by λ if the entries of any one row (or any one column) of A are all multiplied by λ.
- The determinant of A is unchanged if the entries in one row (respectively, column) are added to those in another row (respectively, column).
- The determinant of A equals the determinant of A^T.
- The determinant of A is multiplied by -1 if any two rows (or any two columns) are exchanged.

Theorem 5

An $n \times n$ matrix A is singular if and only if $det(A) = 0$.

Theorems

Theorem 4(determinant properties).

The determinant of a square matrix A has the following properties:

- If any row or any column A is zero, then $det(A) = 0$.
- The determinant of A is multiplied by λ if the entries of any one row (or any one column) of A are all multiplied by λ.
- The determinant of A is unchanged if the entries in one row (respectively, column) are added to those in another row (respectively, column).
- The determinant of A equals the determinant of A^T.
- The determinant of A is multiplied by -1 if any two rows (or any two columns) are exchanged.

Theorem 5

An $n \times n$ matrix A is singular if and only if $det(A) = 0$.

Theorems

Theorem 4(determinant properties).

The determinant of a square matrix A has the following properties:

- If any row or any column A is zero, then $det(A) = 0$.
- The determinant of A is multiplied by λ if the entries of any one row (or any one column) of A are all multiplied by λ.
- The determinant of A is unchanged if the entries in one row (respectively, column) are added to those in another row (respectively, column).
- The determinant of A equals the determinant of A^T.
- The determinant of A is multiplied by -1 if any two rows (or any two columns) are exchanged.

Theorem 5

An $n \times n$ matrix A is singular if and only if $det(A) = 0$.

Theorems

Theorem 4(determinant properties).

The determinant of a square matrix A has the following properties:

- If any row or any column A is zero, then $det(A) = 0$.
- The determinant of A is multiplied by λ if the entries of any one row (or any one column) of A are all multiplied by λ.
- The determinant of A is unchanged if the entries in one row (respectively, column) are added to those in another row (respectively, column).
- The determinant of A equals the determinant of A^T.
- The determinant of A is multiplied by -1 if any two rows (or any two columns) are exchanged.

Theorem 5

An $n \times n$ matrix A is singular if and only if $det(A) = 0$.

Theorems

Theorem 4(determinant properties).

The determinant of a square matrix A has the following properties:

- If any row or any column A is zero, then $det(A) = 0$.
- The determinant of A is multiplied by λ if the entries of any one row (or any one column) of A are all multiplied by λ.
- The determinant of A is unchanged if the entries in one row (respectively, column) are added to those in another row (respectively, column).
- The determinant of A equals the determinant of A^T.
- The determinant of A is multiplied by -1 if any two rows (or any two columns) are exchanged.

Theorem 5

An $n \times n$ matrix A is singular if and only if $det(A) = 0$.

Positive definite matrix

Definition

A positive definite matrix A is called positive definite if and only if $x^T A x > 0$ for all $x \neq 0$

Theorem 6

For any matrix A with full column rank, the matrix $A^T A$ is positive definite.

Positive definite matrix

Definition

A positive definite matrix A is called positive definite if and only if $x^T A x > 0$ for all $x \neq 0$

Theorem 6

For any matrix A with full column rank, the matrix $A^T A$ is positive definite.

Cinvestav

Outline

Cinvestav

Matrix Multiplication

Problem description

Given $n \times n$ matrices A,B and C:

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

Thus, you could compute r, s, t and u using recursion!!!

$$r = ae + bg$$

$$s = af + bh$$

$$t = ce + dg$$

$$u = cf + dh$$

Matrix Multiplication

Problem description

Given $n \times n$ matrices A,B and C:

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

Thus, you could compute r, s, t and u using recursion!!!

$$r = ae + bg$$

$$s = af + bh$$

$$t = ce + dg$$

$$u = cf + dh$$

Problem

Complexity of previous approach

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2)$$

Thus

$$T(n) = \Theta(n^3)$$

Therefore

You need to use a different type of products.

Cinvestav

Problem

Complexity of previous approach

$$T(n) = 8\,T\left(\frac{n}{2}\right) + \Theta(n^2)$$

Thus

$$T(n) = \Theta(n^3)$$

Therefore

You need to use a different type of products.

Cinvestav

Problem

Complexity of previous approach

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2)$$

Thus

$$T(n) = \Theta(n^3)$$

Therefore

You need to use a different type of products.

Outline

Cinvestav

The Strassen's Algorithm

It is a divide and conquer algorithm

Given A, B, C matrices with dimensions $n \times n$, we recursively split the matrices such that we finish with 12 $\frac{n}{2} \times \frac{n}{2}$ sub matrices

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

Remember the Gauss Trick?

Imagine the same for Matrix Multiplication.

Cinvestav

The Strassen's Algorithm

It is a divide and conquer algorithm

Given A, B, C matrices with dimensions $n \times n$, we recursively split the matrices such that we finish with 12 $\frac{n}{2} \times \frac{n}{2}$ sub matrices

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

Remember the Gauss Trick?

Imagine the same for Matrix Multiplication.

Outline

Cinvestav

Algorithm

Strassen's Algorithm

1. Divide the input matrices A and B into $\frac{n}{2} \times \frac{n}{2}$ sub matrices.
2. Using $\Theta\left(n^2\right)$ scalar additions and subtractions, compute 14 matrices $A_1, B_1, ..., A_7, B_7$ each of which is $\frac{n}{2} \times \frac{n}{2}$.
3. Recursively compute the seven matrices products $P_i = A_i B_i$ for $i = 1, 2, 3, ..., 7.$
4. Compute the desired matrix

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

by adding and or subtracting various combinations of the P_i matrices, using only $\Theta\left(n^2\right)$ scalar additions and subtractions

Algorithm

Strassen's Algorithm

1. Divide the input matrices A and B into $\frac{n}{2} \times \frac{n}{2}$ sub matrices.
2. Using $\Theta\left(n^2\right)$ scalar additions and subtractions, compute 14 matrices $A_1, B_1, ..., A_7, B_7$ each of which is $\frac{n}{2} \times \frac{n}{2}$.
3. Recursively compute the seven matrices products $P_i = A_i B_i$ for $i = 1, 2, 3, ..., 7$.
4. Compute the desired matrix

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

by adding and or subtracting various combinations of the P_i matrices, using only $\Theta\left(n^2\right)$ scalar additions and subtractions

Algorithm

Strassen's Algorithm

1. Divide the input matrices A and B into $\frac{n}{2} \times \frac{n}{2}$ sub matrices.
2. Using $\Theta\left(n^2\right)$ scalar additions and subtractions, compute 14 matrices $A_1, B_1, ..., A_7, B_7$ each of which is $\frac{n}{2} \times \frac{n}{2}$.
3. Recursively compute the seven matrices products $P_i = A_i B_i$ for $i = 1, 2, 3, ..., 7$.
4. Compute the desired matrix

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

by adding and or subtracting various combinations of the P_i matrices, using only $\Theta\left(n^2\right)$ scalar additions and subtractions

Algorithm

Strassen's Algorithm

1. Divide the input matrices A and B into $\frac{n}{2} \times \frac{n}{2}$ sub matrices.
2. Using $\Theta\left(n^2\right)$ scalar additions and subtractions, compute 14 matrices $A_1, B_1, ..., A_7, B_7$ each of which is $\frac{n}{2} \times \frac{n}{2}$.
3. Recursively compute the seven matrices products $P_i = A_i B_i$ for $i = 1, 2, 3, ..., 7$.
4. Compute the desired matrix

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

by adding and or subtracting various combinations of the P_i matrices, using only $\Theta\left(n^2\right)$ scalar additions and subtractions

Algorithm

Strassen's Algorithm

1. Divide the input matrices A and B into $\frac{n}{2} \times \frac{n}{2}$ sub matrices.
2. Using $\Theta\left(n^2\right)$ scalar additions and subtractions, compute 14 matrices $A_1, B_1, ..., A_7, B_7$ each of which is $\frac{n}{2} \times \frac{n}{2}$.
3. Recursively compute the seven matrices products $P_i = A_i B_i$ for $i = 1, 2, 3, ..., 7$.
4. Compute the desired matrix

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

by adding and or subtracting various combinations of the P_i matrices, using only $\Theta\left(n^2\right)$ scalar additions and subtractions

Cinvestav

Algorithm

Strassen's Algorithm

1. Divide the input matrices A and B into $\frac{n}{2} \times \frac{n}{2}$ sub matrices.
2. Using $\Theta\left(n^2\right)$ scalar additions and subtractions, compute 14 matrices $A_1, B_1, ..., A_7, B_7$ each of which is $\frac{n}{2} \times \frac{n}{2}$.
3. Recursively compute the seven matrices products $P_i = A_i B_i$ for $i = 1, 2, 3, ..., 7$.
4. Compute the desired matrix

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix}$$

by adding and or subtracting various combinations of the P_i matrices, using only $\Theta\left(n^2\right)$ scalar additions and subtractions

Cinvestav

Outline

Cinvestav

Strassen Observed that

Trial and Error

First , he generated

$$P_i = A_i B_i = (\alpha_{i1} a + \alpha_{i2} b + \alpha_{i3} c + \alpha_{i4} d) \cdot (\beta_{i1} e + \beta_{i2} f + \beta_{i3} g + \beta_{i4} h)$$

Where $\alpha_{ij}, \beta_{ij} \in \{-1, 0, 1\}$

Cinvestav

Then

r

$$r = ae + bg = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

s

$$s = af + bh = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

Then

r

$$r = ae + bg = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

s

$$s = af + bh = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

Then

r

$$r = ae + bg = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

s

$$s = af + bh = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} +1 & 0 & 0 & 0 \\ 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

Therefore

t

$$r = ce + dg = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

u

$$u = cf + dh = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & +1 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

Therefore

t

$$r = ce + dg = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ +1 & 0 & 0 & 0 \\ 0 & 0 & +1 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

u

$$u = cf + dh = \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & +1 & 0 & 0 \\ 0 & 0 & 0 & +1 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}$$

Example Compute the s from P_1 and P_2 matrices

Compute

- $s = P_1 + P_2$

Example Compute the s from P_1 and P_2 matrices

Compute

- $s = P_1 + P_2$

Where P_1

$$
\begin{aligned}
P_1 &= A_1 B_1 \\
&= a\left(f - h\right) \\
&= af - ah \\
&= \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & +1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}
\end{aligned}
$$

Example Compute the s from P_1 and P_2 matrices

Compute

- $s = P_1 + P_2$

Where P_1

$$
\begin{aligned}
P_1 &= A_1 B_1 \\
&= a\left(f - h\right) \\
&= af - ah \\
&= \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & +1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}
\end{aligned}
$$

Example Compute the s from P_1 and P_2 matrices

Compute

- $s = P_1 + P_2$

Where P_1

$$
\begin{aligned}
P_1 &= A_1 B_1 \\
&= a\left(f - h\right) \\
&= af - ah
\end{aligned}
$$

$$
= \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & +1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}
$$

Example Compute the s from P_1 and P_2 matrices

Compute

- $s = P_1 + P_2$

Where P_1

$$
\begin{aligned}
P_1 &= A_1 B_1 \\
&= a\left(f - h\right) \\
&= af - ah \\
&= \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & +1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}
\end{aligned}
$$

Cinvestav

Example Compute the s from P_1 and P_2 matrices

Where P_2

$$P_2 = A_2 B_2$$

Example Compute the s from P_1 and P_2 matrices

Where P_2

$$
\begin{aligned}
P_2 &= A_2 B_2 \\
&= (a+b)\,h \\
&= ah + bh \\
&= \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}
\end{aligned}
$$

Cinvestav

Example Compute the s from P_1 and P_2 matrices

Where P_2

$$
\begin{aligned}
P_2 &= A_2 B_2 \\
&= (a+b)\,h \\
&= ah + bh
\end{aligned}
$$

$$
= \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}
$$

Cinvestav

Example Compute the s from P_1 and P_2 matrices

Where P_2

$$
\begin{aligned}
P_2 &= A_2 B_2 \\
&= (a+b)\,h \\
&= ah + bh \\
&= \begin{pmatrix} a & b & c & d \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & +1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} e \\ f \\ g \\ h \end{pmatrix}
\end{aligned}
$$

Outline

Cinvestav

Complexity

Because we are only computing 7 matrices

- $T(n) = 7T\left(\frac{n}{2}\right) + \Theta\left(n^2\right) = \Theta\left(n^{\lg 7}\right) = O\left(n^{2.81}\right)$.

Nevertheless

We do not use Strassen's because

- A constant factor hidden in the running of the algorithm is larger than the constant factor of the naive $\Theta\left(n^3\right)$ method.

Nevertheless

We do not use Strassen's because

- A constant factor hidden in the running of the algorithm is larger than the constant factor of the naive $\Theta\left(n^3\right)$ method.
- When matrices are sparse, there are faster methods.
- Strassen's is not a numerically stable as the naive method.
- The sub matrices formed at the levels of the recursion consume space.

Cinvestav

Nevertheless

We do not use Strassen's because

- A constant factor hidden in the running of the algorithm is larger than the constant factor of the naive $\Theta\left(n^3\right)$ method.
- When matrices are sparse, there are faster methods.
- Strassen's is not a numerically stable as the naive method.
- The sub matrices formed at the levels of the recursion consume space.

Cinvestav

Nevertheless

We do not use Strassen's because

- A constant factor hidden in the running of the algorithm is larger than the constant factor of the naive $\Theta\left(n^3\right)$ method.
- When matrices are sparse, there are faster methods.
- Strassen's is not a numerically stable as the naive method.
- The sub matrices formed at the levels of the recursion consume space.

The Holy Grail of Matrix Multiplications $O\left(n^2\right)$

In a method by Virginia Vassilevska Williams (2012) Assistant Professor at Stanford

- The computational complexity of her method is $\omega < 2.3727$ or $O\left(n^{2.3727}\right)$
- Better than Coppersmith and Winograd (1990) $O\left(n^{2.375477}\right)$

The Holy Grail of Matrix Multiplications $O\left(n^2\right)$

In a method by Virginia Vassilevska Williams (2012) Assistant Professor at Stanford

- The computational complexity of her method is $\omega < 2.3727$ or $O\left(n^{2.3727}\right)$
- Better than Coppersmith and Winograd (1990) $O\left(n^{2.375477}\right)$

Many Researchers Believe that

- Coppersmith, Winograd and Cohn et al. conjecture could lead to $O\left(n^2\right)$, contradicting a variant of the widely believed *sun flower* conjecture of Erdos and Rado.

Cinvestav

The Holy Grail of Matrix Multiplications $O\left(n^2\right)$

In a method by Virginia Vassilevska Williams (2012) Assistant Professor at Stanford

- The computational complexity of her method is $\omega < 2.3727$ or $O\left(n^{2.3727}\right)$
- Better than Coppersmith and Winograd (1990) $O\left(n^{2.375477}\right)$

Many Researchers Believe that

- Coppersmith, Winograd and Cohn et al. conjecture could lead to $O\left(n^2\right)$, contradicting a variant of the widely believed *sun flower* conjecture of Erdos and Rado.

Exercises

- 28.1-3
- 28.1-5
- 28.1-8
- 28.1-9
- 28.2-2
- 28.2-5

Outline

Cinvestav

In Many Fields

From Optimization to Control

We are required to solve systems of simultaneous equations.

For Example

For Polynomial Curve Fitting, we are given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$

We want

To find a polynomial of degree $n-1$ with structure

$$p(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1}$$

In Many Fields

From Optimization to Control

We are required to solve systems of simultaneous equations.

For Example

For Polynomial Curve Fitting, we are given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$

We want

To find a polynomial of degree $n-1$ with structure

$$p(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1}$$

In Many Fields

From Optimization to Control

We are required to solve systems of simultaneous equations.

For Example

For Polynomial Curve Fitting, we are given $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$

We want

To find a polynomial of degree $n-1$ with structure

$$p(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1}$$

Thus

We can build a system of equations

$$
\begin{aligned}
a_0 + a_1 x_1 + a_2 x_1^2 + \ldots + a_{n-1} x_1^{n-1} &= y_1 \\
a_0 + a_1 x_2 + a_2 x_2^2 + \ldots + a_{n-1} x_2^{n-1} &= y_2 \\
&\vdots \\
a_0 + a_1 x_n + a_2 x_n^2 + \ldots + a_{n-1} x_n^{n-1} &= y_n
\end{aligned}
$$

We have n unknowns

$$a_0, a_1, a_2, \ldots, a_{n-1}$$

Thus

We can build a system of equations

$$a_0 + a_1 x_1 + a_2 x_1^2 + ... + a_{n-1} x_1^{n-1} = y_1$$
$$a_0 + a_1 x_2 + a_2 x_2^2 + ... + a_{n-1} x_2^{n-1} = y_2$$
$$\vdots$$
$$a_0 + a_1 x_n + a_2 x_n^2 + ... + a_{n-1} x_n^{n-1} = y_n$$

We have n unknowns

$$a_0, a_1, a_2, ..., a_{n-1}$$

Cinvestav

Solving Systems of Linear Equations

Proceed as follows

- We start with a set of linear equations with n unknowns:

$$x_1, x_2, ..., x_n \begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n & = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n & = b_2 \\ \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n & = b_n \end{cases}$$

Solving Systems of Linear Equations

Proceed as follows

- We start with a set of linear equations with n unknowns:

$$x_1, x_2, ..., x_n \begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n & = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n & = b_2 \\ \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n & = b_n \end{cases}$$

Something Notable

- A set of values for $x_1, x_2, ..., x_n$ that satisfy all of the equations simultaneously is said to be a solution to these equations.
- In this section, we only treat the case in which there are exactly n equations in n unknowns.

Solving Systems of Linear Equations

Proceed as follows

- We start with a set of linear equations with n unknowns:

$$x_1, x_2, ..., x_n \begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n & = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n & = b_2 \\ \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n & = b_n \end{cases}$$

Something Notable

- A set of values for $x_1, x_2, ..., x_n$ that satisfy all of the equations simultaneously is said to be a solution to these equations.
- In this section, we only treat the case in which there are exactly n equations in n unknowns.

Solving Systems of Linear Equations

Proceed as follows

- We start with a set of linear equations with n unknowns:

$$x_1, x_2, ..., x_n \begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n & = b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n & = b_2 \\ \vdots & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n & = b_n \end{cases}$$

Something Notable

- A set of values for $x_1, x_2, ..., x_n$ that satisfy all of the equations simultaneously is said to be a solution to these equations.
- In this section, we only treat the case in which there are exactly n equations in n unknowns.

Solving systems of linear equations

continuation

- We can conveniently rewrite the equations as the matrix-vector equation:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

or, equivalently, letting $A = (a_{ij})$, $x = (x_j)$, and $b = (b_i)$, as

$$Ax = b$$

- In this section, we shall be concerned predominantly with the case of which A is nonsingular, after all we want to invert A.

Solving systems of linear equations

continuation

- We can conveniently rewrite the equations as the matrix-vector equation:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

or, equivalently, letting $A = (a_{ij})$, $x = (x_j)$, and $b = (b_i)$, as

$$Ax = b$$

- In this section, we shall be concerned predominantly with the case of which A is nonsingular, after all we want to invert A.

Solving systems of linear equations

continuation

- We can conveniently rewrite the equations as the matrix-vector equation:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

or, equivalently, letting $A = (a_{ij})$, $x = (x_j)$, and $b = (b_i)$, as

$$Ax = b$$

- In this section, we shall be concerned predominantly with the case of which A is nonsingular, after all we want to invert A.

Solving systems of linear equations

continuation

- We can conveniently rewrite the equations as the matrix-vector equation:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

or, equivalently, letting $A = (a_{ij})$, $x = (x_j)$, and $b = (b_i)$, as

$$Ax = b$$

- In this section, we shall be concerned predominantly with the case of which A is nonsingular, after all we want to invert A.

Solving systems of linear equations

continuation

- We can conveniently rewrite the equations as the matrix-vector equation:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

or, equivalently, letting $A = (a_{ij})$, $x = (x_j)$, and $b = (b_i)$, as

$$Ax = b$$

- In this section, we shall be concerned predominantly with the case of which A is nonsingular, after all we want to invert A.

Outline

Cinvestav

Overview of Lower Upper (LUP) Decomposition

Intuition

The idea behind LUP decomposition is to find three $n \times n$ matrices L, U, and P such that:

$$PA = LU$$

where:

- L is a unit lower triangular matrix.
- U is an upper triangular matrix.
- P is a permutation matrix.

Overview of Lower Upper (LUP) Decomposition

Intuition

The idea behind LUP decomposition is to find three $n \times n$ matrices L, U, and P such that:

$$PA = LU$$

where:

- L is a unit lower triangular matrix.
- U is an upper triangular matrix.
- P is a permutation matrix.

Where

We call matrices L, U, and P satisfying the above equation a LUP decomposition of the matrix A.

Overview of Lower Upper (LUP) Decomposition

Intuition

The idea behind LUP decomposition is to find three $n \times n$ matrices L, U, and P such that:

$$PA = LU$$

where:

- L is a unit lower triangular matrix.
- U is an upper triangular matrix.
- P is a permutation matrix.

Where

We call matrices L, U, and P satisfying the above equation a LUP decomposition of the matrix A.

Overview of Lower Upper (LUP) Decomposition

Intuition

The idea behind LUP decomposition is to find three $n \times n$ matrices L, U, and P such that:

$$PA = LU$$

where:

- L is a unit lower triangular matrix.
- U is an upper triangular matrix.
- P is a permutation matrix.

Where

We call matrices L, U, and P satisfying the above equation a LUP decomposition of the matrix A.

Overview of Lower Upper (LUP) Decomposition

Intuition

The idea behind LUP decomposition is to find three $n \times n$ matrices L, U, and P such that:

$$PA = LU$$

where:

- L is a unit lower triangular matrix.
- U is an upper triangular matrix.
- P is a permutation matrix.

Where

We call matrices L, U, and P satisfying the above equation a LUP decomposition of the matrix A.

Overview of Lower Upper (LUP) Decomposition

Intuition

The idea behind LUP decomposition is to find three $n \times n$ matrices L, U, and P such that:

$$PA = LU$$

where:

- L is a unit lower triangular matrix.
- U is an upper triangular matrix.
- P is a permutation matrix.

Where

We call matrices L, U, and P satisfying the above equation a LUP decomposition of the matrix A.

Overview of Lower Upper (LUP) Decomposition

Intuition

The idea behind LUP decomposition is to find three $n \times n$ matrices L, U, and P such that:

$$PA = LU$$

where:

- L is a unit lower triangular matrix.
- U is an upper triangular matrix.
- P is a permutation matrix.

Where

We call matrices L, U, and P satisfying the above equation a LUP decomposition of the matrix A.

What is a Permutation Matrix

Basically

We represent the permutation P compactly by an array $\pi[1..n]$. For $i = 1, 2, ..., n$, the entry $\pi[i]$ indicates that $P_{i\pi[i]} = 1$ and $P_{ij} = 0$ for $j \neq \pi[i]$.

Thus

- PA has $a_{\pi[i],j}$ in row i and a column j.
- Pb has $b_{\pi[i]}$ as its ith element.

What is a Permutation Matrix

Basically

We represent the permutation P compactly by an array $\pi[1..n]$. For $i = 1, 2, ..., n$, the entry $\pi[i]$ indicates that $P_{i\pi[i]} = 1$ and $P_{ij} = 0$ for $j \neq \pi[i]$.

Thus

- PA has $a_{\pi[i],j}$ in row i and a column j.
- Pb has $b_{\pi[i]}$ as its ith element.

Cinvestav

How can we use this in our advantage?

Lock at this

$$Ax = b \Longrightarrow PAx = Pb \tag{2}$$

Therefore

$$LUx = Pb \tag{3}$$

Now, if we make $Ux = y$

$$Ly = Pb \tag{4}$$

How can we use this in our advantage?

Lock at this

$$Ax = b \Longrightarrow PAx = Pb \tag{2}$$

Therefore

$$LUx = Pb \tag{3}$$

Now, if we make $Ux = y$

$$Ly = Pb \tag{4}$$

How can we use this in our advantage?

Lock at this

$$Ax = b \Longrightarrow PAx = Pb \tag{2}$$

Therefore

$$LUx = Pb \tag{3}$$

Now, if we make $Ux = y$

$$Ly = Pb \tag{4}$$

Cinvestav

Thus

We first obtain y

Then, we obtain x.

Outline

Cinvestav

Forward and Back Substitution

Forward substitution

Forward substitution can solve the lower triangular system $Ly = Pb$ in $\Theta(n^2)$ time, given L, P and b.

Then

Since L is unit lower triangular, equation $Ly = Pb$ can be rewritten as:

$$
\begin{aligned}
y_1 &= b_{\pi[1]} \\
l_{21} y_1 + y_2 &= b_{\pi[2]} \\
l_{31} y_1 + l_{32} + y_3 &= b_{\pi[3]} \\
&\vdots \\
l_{n1} y_1 + l_{n2} y_2 + l_{n3} y_3 + \ldots + y_n &= b_{\pi[n]}
\end{aligned}
$$

Forward and Back Substitution

Forward substitution

Forward substitution can solve the lower triangular system $Ly = Pb$ in $\Theta(n^2)$ time, given L, P and b.

Then

Since L is unit lower triangular, equation $Ly = Pb$ can be rewritten as:

$$\begin{aligned} y_1 &= b_{\pi[1]} \\ l_{21}y_1 + y_2 &= b_{\pi[2]} \\ l_{31}y_1 + l_{32} + y_3 &= b_{\pi[3]} \\ &\vdots \\ l_{n1}y_1 + l_{n2}y_2 + l_{n3}y_3 + ... + y_n &= b_{\pi[n]} \end{aligned}$$

Cinvestav

Forward and Back Substitution

Back substitution

Back substitution is similar to forward substitution. Like forward substitution, this process runs in $\Theta(n^2)$ time. Since U is upper-triangular, we can rewrite the system $Ux = y$ as

$$
\begin{aligned}
u_{11}x_1 + u_{12}x_2 + ... + u_{1n-2}x_{n-2} + u_{1n-1}x_{n-1} + u_{1n}x_n &= y_1 \\
u_{22}x_2 + ... + u_{2n-2}x_{n-2} + u_{2n-1}x_{n-1} + u_{2n}x_n &= y_2 \\
&\vdots \\
u_{n-2n-2}x_{n-2} + u_{n-2n-1}x_{n-1} + u_{n-2n}x_n &= y_{n-2} \\
u_{n-1n-1}x_{n-1} + u_{n-1n}x_n &= y_{n-1} \\
u_{nn}x_n &= y_n
\end{aligned}
$$

Cinvestav

Example

We have

$$Ax = \begin{pmatrix} 1 & 2 & 0 \\ 3 & 4 & 4 \\ 5 & 6 & 3 \end{pmatrix} x = \begin{pmatrix} 3 \\ 7 \\ 8 \end{pmatrix} = b$$

Example

The L, U and P matrix

$$L=\begin{pmatrix}1 & 0 & 0\\ 0.2 & 1 & 0\\ 0.6 & 0.5 & 1\end{pmatrix}, U=\begin{pmatrix}5 & 6 & 3\\ 0 & 0.8 & -0.6\\ 0 & 0 & 2.5\end{pmatrix}, P=\begin{pmatrix}0 & 0 & 1\\ 1 & 0 & 0\\ 0 & 1 & 0\end{pmatrix}$$

Cinvestav

Example

Using forward substitution, $Ly = Pb$ for y

$$Ly = \begin{pmatrix} 1 & 0 & 0 \\ 0.2 & 1 & 0 \\ 0.6 & 0.5 & 1 \end{pmatrix} y = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 3 \\ 7 \\ 8 \end{pmatrix} = Pb$$

Example

Using forward substitution, we get y

$$y = \begin{pmatrix} 8 \\ 1.4 \\ 1.5 \end{pmatrix}$$

Example

Now, we use the back substitution, $Ux = y$ for x

$$Ux = \begin{pmatrix} 5 & 6 & 3 \\ 0 & 0.8 & -0.6 \\ 0 & 0 & 2.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 1.4 \\ 1.5 \end{pmatrix},$$

Cinvestav

Example

Finally, we get

$$x = \begin{pmatrix} -1.4 \\ 2.2 \\ 0.6 \end{pmatrix}$$

Forward and Back Substitution

Given P, L, U, and b, the procedure LUP- SOLVE solves for x by combining forward and back substitution

LUP-SOLVE(L, U, π, b)

1. $n = L.rows$
2. Let x be a new vector of length n
3. **for** $i = 1$ **to** n
4. $\quad y_i = b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij} y_j$
5. **for** $i = n$ **downto** 1
6. $\quad x_i = \frac{\left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j\right)}{u_{ii}}$
7. **return** x

Forward and Back Substitution

Given P, L, U, and b, the procedure LUP- SOLVE solves for x by combining forward and back substitution

LUP-SOLVE(L, U, π, b)

1. $n = L.rows$
2. Let x be a new vector of length n
3. **for** $i = 1$ **to** n
4. $\qquad y_i = b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij} y_j$
5. **for** $i = n$ **downto** 1
6. $\qquad x_i = \frac{\left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j\right)}{u_{ii}}$
7. **return** x

Complexity

The running time is $\Theta(n^2)$.

Forward and Back Substitution

Given P, L, U, and b, the procedure LUP- SOLVE solves for x by combining forward and back substitution

LUP-SOLVE(L, U, π, b)

$n = L.rows$

Let x be a new vector of length n

for $i = 1$ **to** n

  $y_i = b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij} y_j$

for $i = n$ **downto** 1

  $x_i = \frac{\left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j\right)}{u_{ii}}$

return x

Complexity

The running time is $\Theta(n^2)$.

Forward and Back Substitution

Given P, L, U, and b, the procedure LUP- SOLVE solves for x by combining forward and back substitution

LUP-SOLVE(L, U, π, b)

$n = L.rows$

Let x be a new vector of length n

for $i = 1$ **to** n

$\quad y_i = b_{\pi[i]} - \sum_{j=1}^{i-1} l_{ij} y_j$

for $i = n$ **downto** 1

$\quad x_i = \frac{\left(y_i - \sum_{j=i+1}^{n} u_{ij} x_j\right)}{u_{ii}}$

return x

Complexity

The running time is $\Theta(n^2)$.

Outline

Cinvestav

Ok, if we have the L, U and P!!!

Thus

We need to find those matrices

How, we do it?

We are going to use something called the Gaussian Elimination.

Ok, if we have the L, U and P!!!

Thus

We need to find those matrices

How, we do it?

We are going to use something called the Gaussian Elimination.

For this

We assume that A is a $n \times n$

Such that A is not singular

We use a process known as Gaussian elimination to create LU decomposition

This algorithm is recursive in nature.

Properties

Clearly if $n = 1$, we are done for $L = I_1$ and $U = A$.

For this

We assume that A is a $n \times n$

Such that A is not singular

We use a process known as Gaussian elimination to create LU decomposition

This algorithm is recursive in nature.

Properties

Clearly if $n = 1$, we are done for $L = I_1$ and $U = A$.

For this

We assume that A is a $n \times n$

Such that A is not singular

We use a process known as Gaussian elimination to create LU decomposition

This algorithm is recursive in nature.

Properties

Clearly if $n = 1$, we are done for $L = I_1$ and $U = A$.

Outline

Cinvestav

Computing LU decomposition

For $n > 1$, we break A into four parts

$$A = \left(\begin{array}{c|ccc} a_{11} & a_{12} & \cdots & a_{1n} \\ \hline a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{array}\right) = \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ \boldsymbol{v} & A' \end{pmatrix} \tag{5}$$

Cinvestav

Where

We have

- $\boldsymbol{v}$ is a column $(n-1)$ −vector.
- $\boldsymbol{w}^T$ is a row $(n-1)$ −vector.
- A' is an $(n-1) \times (n-1)$.

Where

We have

- $\boldsymbol{v}$ is a column $(n-1)$ −vector.
- $\boldsymbol{w}^T$ is a row $(n-1)$ −vector.
- A' is an $(n-1) \times (n-1)$.

Where

We have

- $\boldsymbol{v}$ is a column $(n-1)$ −vector.
- $\boldsymbol{w}^T$ is a row $(n-1)$ −vector.
- A' is an $(n-1) \times (n-1)$.

Where

We have

- $\boldsymbol{v}$ is a column $(n-1)$ −vector.
- $\boldsymbol{w}^T$ is a row $(n-1)$ −vector.
- A' is an $(n-1) \times (n-1)$.

Computing a LU decomposition

Thus, we can do the following

$$A = \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ \boldsymbol{v} & A' \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & \underbrace{A' - \frac{\boldsymbol{v}\boldsymbol{w}^T}{a_{11}}}_{\text{Schur Complement}} \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & L'U' \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & L' \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & U' \end{pmatrix}$$

$$= LU$$

Computing a LU decomposition

Thus, we can do the following

$$
\begin{aligned}
A &= \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ \boldsymbol{v} & A' \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & \underbrace{A' - \frac{\boldsymbol{v}\boldsymbol{w}^T}{a_{11}}}_{\text{Schur Complement}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & L'U' \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & L' \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Computing a LU decomposition

Thus, we can do the following

$$
\begin{aligned}
A &= \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ \boldsymbol{v} & A' \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & \underbrace{A' - \frac{\boldsymbol{v}\boldsymbol{w}^T}{a_{11}}}_{\text{Schur Complement}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & L'U' \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & L' \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Computing a LU decomposition

Thus, we can do the following

$$
\begin{aligned}
A &= \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ \boldsymbol{v} & A' \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & \underbrace{A' - \frac{\boldsymbol{v}\boldsymbol{w}^T}{a_{11}}}_{\text{Schur Complement}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & L'U' \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & L' \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Computing a LU decomposition

Thus, we can do the following

$$
\begin{aligned}
A &= \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ \boldsymbol{v} & A' \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & \underbrace{A' - \frac{\boldsymbol{v}\boldsymbol{w}^T}{a_{11}}}_{\text{Schur Complement}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & L'U' \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{\boldsymbol{v}}{a_{11}} & L' \end{pmatrix} \begin{pmatrix} a_{11} & \boldsymbol{w}^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Computing a LU decomposition

Pseudo-Code running in $\Theta\left(n^3\right)$

LU-Decomposition(A)

$n = A.rows$

Let L and U be new $n \times n$ matrices

Initialize U with 0's below the diagonal

Initialize L with 1's on the diagonal and 0's above the diagonal.

for $k = 1$ **to** n

    $u_{kk} = a_{kk}$

    for $i = k + 1$ **to** n

        $l_{ik} = \frac{a_{ik}}{u_{kk}}$ ◁ l_{ik} holds v_i

        $u_{ki} = a_{ki}$ ◁ u_{ki} holds w_i^T

    for $i = k + 1$ **to** n

        for $j = k + 1$ **to** n

            $a_{ij} = a_{ij} - l_{ik} u_{kj}$

return L and U

Computing a LU decomposition

Pseudo-Code running in $\Theta\left(n^3\right)$

LU-Decomposition(A)

1. $n = A.rows$
2. Let L and U be new $n \times n$ matrices
3. Initialize U with 0's below the diagonal
4. Initialize L with 1's on the diagonal and 0's above the diagonal.
5. **for** $k = 1$ **to** n
6. $\quad u_{kk} = a_{kk}$
7. $\quad$ **for** $i = k + 1$ **to** n
8. $\qquad l_{ik} = \frac{a_{ik}}{u_{kk}} \triangleleft l_{ik}$ holds v_i
9. $\qquad u_{ki} = a_{ki} \triangleleft u_{ki}$ holds w_i^T
10. $\quad$ **for** $i = k + 1$ **to** n
11. $\qquad$ **for** $j = k + 1$ **to** n
12. $\qquad\quad a_{ij} = a_{ij} - l_{ik} u_{kj}$
13. **return** L and U

Computing a LU decomposition

Pseudo-Code running in $\Theta\left(n^3\right)$

LU-Decomposition(A)

1. $n = A.rows$
2. Let L and U be new $n \times n$ matrices
3. Initialize U with 0's below the diagonal
4. Initialize L with 1's on the diagonal and 0's above the diagonal.
5. for $k = 1$ to n
6. $\quad u_{kk} = a_{kk}$
7. $\quad$ for $i = k + 1$ to n
8. $\quad\quad l_{ik} = \frac{a_{ik}}{u_{kk}} \triangleleft\ l_{ik}$ holds v_i
9. $\quad\quad u_{ki} = a_{ki} \triangleleft\ u_{ki}$ holds w_i^T
10. $\quad$ for $i = k + 1$ to n
11. $\quad\quad$ for $j = k + 1$ to n
12. $\quad\quad\quad a_{ij} = a_{ij} - l_{ik} u_{kj}$
13. return L and U

Computing a LU decomposition

Pseudo-Code running in $\Theta\left(n^3\right)$

LU-Decomposition(A)

1. $n = A.rows$
2. Let L and U be new $n \times n$ matrices
3. Initialize U with 0's below the diagonal
4. Initialize L with 1's on the diagonal and 0's above the diagonal.
5. for $k = 1$ to n
6. $\quad u_{kk} = a_{kk}$
7. $\quad$ for $i = k + 1$ to n
8. $\quad\quad l_{ik} = \frac{a_{ik}}{u_{kk}} \triangleleft l_{ik}$ holds v_i
9. $\quad\quad u_{ki} = a_{ki} \triangleleft u_{ki}$ holds w_i^T
10. $\quad$ for $i = k + 1$ to n
11. $\quad\quad$ for $j = k + 1$ to n
12. $\quad\quad\quad a_{ij} = a_{ij} - l_{ik} u_{kj}$
13. return L and U

Computing a LU decomposition

Pseudo-Code running in $\Theta\left(n^3\right)$

LU-Decomposition(A)

1. $n = A.rows$
2. Let L and U be new $n \times n$ matrices
3. Initialize U with 0's below the diagonal
4. Initialize L with 1's on the diagonal and 0's above the diagonal.
5. **for** $k = 1$ **to** n
6. $\quad u_{kk} = a_{kk}$
7. $\quad$ **for** $i = k + 1$ **to** n
8. $\quad\quad l_{ik} = \frac{a_{ik}}{u_{kk}}$ ◁ l_{ik} holds v_i
9. $\quad\quad u_{ki} = a_{ki}$ ◁ u_{ki} holds w_i^T
10. $\quad$ **for** $i = k + 1$ **to** n
11. $\quad\quad$ **for** $j = k + 1$ **to** n
12. $\quad\quad\quad a_{ij} = a_{ij} - l_{ik} u_{kj}$
13. **return** L and U

Computing a LU decomposition

Pseudo-Code running in $\Theta\left(n^3\right)$

LU-Decomposition(A)

1. $n = A.rows$
2. Let L and U be new $n \times n$ matrices
3. Initialize U with 0's below the diagonal
4. Initialize L with 1's on the diagonal and 0's above the diagonal.
5. **for** $k = 1$ **to** n
6. $\quad u_{kk} = a_{kk}$
7. $\quad$ **for** $i = k + 1$ **to** n
8. $\quad\quad l_{ik} = \frac{a_{ik}}{u_{kk}}$ ◁ l_{ik} holds v_i
9. $\quad\quad u_{ki} = a_{ki}$ ◁ u_{ki} holds w_i^T
10. $\quad$ **for** $i = k + 1$ **to** n
11. $\quad\quad$ **for** $j = k + 1$ **to** n
12. $\quad\quad\quad a_{ij} = a_{ij} - l_{ik} u_{kj}$
13. **return** L and U

Computing a LU decomposition

Pseudo-Code running in $\Theta\left(n^3\right)$

LU-Decomposition(A)

- $n = A.rows$
- Let L and U be new $n \times n$ matrices
- Initialize U with 0's below the diagonal
- Initialize L with 1's on the diagonal and 0's above the diagonal.
- **for** $k = 1$ **to** n
    - $u_{kk} = a_{kk}$
    - **for** $i = k + 1$ **to** n
        - $l_{ik} = \frac{a_{ik}}{u_{kk}}$ $\triangleleft$ l_{ik} holds v_i
        - $u_{ki} = a_{ki}$ $\triangleleft$ u_{ki} holds w_i^T
    - **for** $i = k + 1$ **to** n
        - **for** $j = k + 1$ **to** n
            - $a_{ij} = a_{ij} - l_{ik} u_{kj}$
- **return** L and U

Computing a LU decomposition

Pseudo-Code running in $\Theta\left(n^3\right)$

LU-Decomposition(A)

```
n = A.rows
Let L and U be new n × n matrices
Initialize U with 0's below the diagonal
Initialize L with 1's on the diagonal and 0's above the diagonal.
for k = 1 to n
    u_kk = a_kk
    for i = k + 1 to n
    l_ik = a_ik / u_kk ◁ l_ik holds v_i
    u_ki = a_ki ◁ u_ki holds w_i^T
    for i = k + 1 to n
        for j = k + 1 to n
            a_ij = a_ij − l_ik u_kj
return L and U
```

Computing a LU decomposition

Pseudo-Code running in $\Theta\left(n^3\right)$

LU-Decomposition(A)

$n = A.rows$

Let L and U be new $n \times n$ matrices

Initialize U with 0's below the diagonal

Initialize L with 1's on the diagonal and 0's above the diagonal.

for $k = 1$ **to** n

    $u_{kk} = a_{kk}$

    for $i = k + 1$ **to** n

    $l_{ik} = \frac{a_{ik}}{u_{kk}} \triangleleft l_{ik}$ holds v_i

    $u_{ki} = a_{ki} \triangleleft u_{ki}$ holds w_i^T

    for $i = k + 1$ **to** n

        for $j = k + 1$ **to** n

            $a_{ij} = a_{ij} - l_{ik} u_{kj}$

return L and U

Example

Here, we have this example

2	3	1	5
6	**13**	**5**	**19**
2	**19**	**10**	**23**
4	**10**	**11**	**31**

$$\Rightarrow \begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 6 \\ 2 \\ 4 \end{pmatrix}\begin{pmatrix} 3 & 1 & 5 \end{pmatrix} =$$

$$\begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 18 & 6 & 30 \\ 6 & 2 & 10 \\ 12 & 4 & 20 \end{pmatrix} \Rightarrow$$

2	3	1	5
3	4	2	4
1	16	9	18
2	4	9	21

$$\Rightarrow \begin{pmatrix} 9 & 18 \\ 9 & 11 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 16 \\ 4 \end{pmatrix}\begin{pmatrix} 2 & 4 \end{pmatrix} = \begin{pmatrix} 9 & 18 \\ 9 & 11 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 32 & 64 \\ 8 & 16 \end{pmatrix} =$$

$$\begin{pmatrix} 9 & 18 \\ 9 & 11 \end{pmatrix} - \begin{pmatrix} 8 & 16 \\ 2 & 4 \end{pmatrix} \Rightarrow$$

2	3	1	5
3	4	2	4
1	4	1	2
2	1	7	17

$\Rightarrow$

2	3	1	5
3	4	2	4
1	4	1	2
2	1	7	3

Example

Here, we have this example

$$\left|\begin{array}{cccc} 2 & 3 & 1 & 5 \\ 6 & \mathbf{\color{red}{13}} & \mathbf{\color{red}{5}} & \mathbf{\color{red}{19}} \\ 2 & \mathbf{\color{red}{19}} & \mathbf{\color{red}{10}} & \mathbf{\color{red}{23}} \\ 4 & \mathbf{\color{red}{10}} & \mathbf{\color{red}{11}} & \mathbf{\color{red}{31}} \end{array}\right| \Rightarrow \begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 6 \\ 2 \\ 4 \end{pmatrix}\begin{pmatrix} 3 & 1 & 5 \end{pmatrix} =$$

$$\begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 18 & 6 & 30 \\ 6 & 2 & 10 \\ 12 & 4 & 20 \end{pmatrix}$$

Example

Here, we have this example

2	3	1	5
6	**13**	**5**	**19**
2	**19**	**10**	**23**
4	**10**	**11**	**31**

$$\Rightarrow \begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 6 \\ 2 \\ 4 \end{pmatrix}\begin{pmatrix} 3 & 1 & 5 \end{pmatrix} =$$

$$\begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 18 & 6 & 30 \\ 6 & 2 & 10 \\ 12 & 4 & 20 \end{pmatrix} \Rightarrow$$

2	**3**	**1**	**5**
3	4	2	4
1	16	**9**	**18**
2	4	**9**	**21**

Example

Here, we have this example

$$\left|\begin{array}{cccc} 2 & 3 & 1 & 5 \\ 6 & \mathbf{13} & \mathbf{5} & \mathbf{19} \\ 2 & \mathbf{19} & \mathbf{10} & \mathbf{23} \\ 4 & \mathbf{10} & \mathbf{11} & \mathbf{31} \end{array}\right| \Rightarrow \begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 6 \\ 2 \\ 4 \end{pmatrix}\begin{pmatrix} 3 & 1 & 5 \end{pmatrix} =$$

$$\begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 18 & 6 & 30 \\ 6 & 2 & 10 \\ 12 & 4 & 20 \end{pmatrix} \Rightarrow \begin{array}{c|ccc} \mathbf{2} & \mathbf{3} & \mathbf{1} & \mathbf{5} \\ \hline \mathbf{3} & 4 & 2 & 4 \\ \mathbf{1} & 16 & \mathbf{9} & \mathbf{18} \\ \mathbf{2} & 4 & \mathbf{9} & \mathbf{21} \end{array}$$

$$\Rightarrow \begin{pmatrix} 9 & 18 \\ 9 & 11 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 16 \\ 4 \end{pmatrix}\begin{pmatrix} 2 & 4 \end{pmatrix} = \begin{pmatrix} 9 & 18 \\ 9 & 11 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 32 & 64 \\ 8 & 16 \end{pmatrix} =$$

$$\begin{pmatrix} 9 & 18 \\ 9 & 11 \end{pmatrix} - \begin{pmatrix} 8 & 16 \\ 2 & 4 \end{pmatrix}$$

Example

Here, we have this example

$$\left|\begin{array}{cccc} 2 & 3 & 1 & 5 \\ 6 & \color{red}{\mathbf{13}} & \color{red}{\mathbf{5}} & \color{red}{\mathbf{19}} \\ 2 & \color{red}{\mathbf{19}} & \color{red}{\mathbf{10}} & \color{red}{\mathbf{23}} \\ 4 & \color{red}{\mathbf{10}} & \color{red}{\mathbf{11}} & \color{red}{\mathbf{31}} \end{array}\right| \Rightarrow \begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 6 \\ 2 \\ 4 \end{pmatrix}\begin{pmatrix} 3 & 1 & 5 \end{pmatrix} =$$

$$\begin{pmatrix} 13 & 5 & 19 \\ 19 & 10 & 23 \\ 10 & 11 & 31 \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 18 & 6 & 30 \\ 6 & 2 & 10 \\ 12 & 4 & 20 \end{pmatrix} \Rightarrow \begin{array}{c|ccc} \mathbf{2} & \color{blue}{\mathbf{3}} & \color{blue}{\mathbf{1}} & \color{blue}{\mathbf{5}} \\ \hline \color{blue}{\mathbf{3}} & 4 & 2 & 4 \\ \color{blue}{\mathbf{1}} & 16 & \color{red}{\mathbf{9}} & \color{red}{\mathbf{18}} \\ \color{blue}{\mathbf{2}} & 4 & \color{red}{\mathbf{9}} & \color{red}{\mathbf{21}} \end{array}$$

$$\Rightarrow \begin{pmatrix} 9 & 18 \\ 9 & 11 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 16 \\ 4 \end{pmatrix}\begin{pmatrix} 2 & 4 \end{pmatrix} = \begin{pmatrix} 9 & 18 \\ 9 & 11 \end{pmatrix} - \frac{1}{4}\begin{pmatrix} 32 & 64 \\ 8 & 16 \end{pmatrix} =$$

$$\begin{pmatrix} 9 & 18 \\ 9 & 11 \end{pmatrix} - \begin{pmatrix} 8 & 16 \\ 2 & 4 \end{pmatrix} \Rightarrow \begin{array}{c|ccc} 2 & 3 & 1 & 5 \\ \hline 3 & \mathbf{4} & \color{blue}{\mathbf{2}} & \color{blue}{\mathbf{4}} \\ \cline{2-4} 1 & \multicolumn{1}{c|}{\color{blue}{\mathbf{4}}} & \color{red}{\mathbf{1}} & \color{red}{\mathbf{2}} \\ 2 & \multicolumn{1}{c|}{\color{blue}{\mathbf{1}}} & \color{red}{\mathbf{7}} & \color{red}{\mathbf{17}} \end{array}$$

Example

Here, we have this example

$$\left|\begin{array}{cccc} 2 & 3 & 1 & 5\\ 6 & \mathbf{13} & \mathbf{5} & \mathbf{19}\\ 2 & \mathbf{19} & \mathbf{10} & \mathbf{23}\\ 4 & \mathbf{10} & \mathbf{11} & \mathbf{31}\end{array}\right| \Rightarrow \begin{pmatrix} 13 & 5 & 19\\ 19 & 10 & 23\\ 10 & 11 & 31\end{pmatrix} - \frac{1}{2}\begin{pmatrix} 6\\ 2\\ 4\end{pmatrix}\begin{pmatrix} 3 & 1 & 5\end{pmatrix} =$$

$$\begin{pmatrix} 13 & 5 & 19\\ 19 & 10 & 23\\ 10 & 11 & 31\end{pmatrix} - \frac{1}{2}\begin{pmatrix} 18 & 6 & 30\\ 6 & 2 & 10\\ 12 & 4 & 20\end{pmatrix} \Rightarrow$$

2	**3**	**1**	**5**
3	4	2	4
1	16	**9**	**18**
2	4	**9**	**21**

$$\Rightarrow \begin{pmatrix} 9 & 18\\ 9 & 11\end{pmatrix} - \frac{1}{4}\begin{pmatrix} 16\\ 4\end{pmatrix}\begin{pmatrix} 2 & 4\end{pmatrix} = \begin{pmatrix} 9 & 18\\ 9 & 11\end{pmatrix} - \frac{1}{4}\begin{pmatrix} 32 & 64\\ 8 & 16\end{pmatrix} =$$

$$\begin{pmatrix} 9 & 18\\ 9 & 11\end{pmatrix} - \begin{pmatrix} 8 & 16\\ 2 & 4\end{pmatrix} \Rightarrow$$

2	3	1	5
3	**4**	**2**	**4**
1	**4**	**1**	**2**
2	**1**	**7**	**17**

$\Rightarrow$

2	3	1	5
3	4	2	4
1	4	**1**	**2**
2	1	**7**	**3**

Thus

We get the following decomposition

$$\begin{pmatrix} 2 & 3 & 1 & 5 \\ 6 & 13 & 5 & 19 \\ 2 & 19 & 10 & 23 \\ 4 & 10 & 11 & 31 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 2 & 1 & 7 & 1 \end{pmatrix} \begin{pmatrix} 2 & 3 & 1 & 5 \\ 0 & 4 & 2 & 4 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

Outline

Cinvestav

Observations

Something Notable

- The elements by which we divide during LU decomposition are called pivots.
- They occupy the diagonal elements of the matrix U.

Observations

Something Notable

- The elements by which we divide during LU decomposition are called pivots.
- They occupy the diagonal elements of the matrix U.

Why the permutation P

It allows us to avoid dividing by 0.

Observations

Something Notable

- The elements by which we divide during LU decomposition are called pivots.
- They occupy the diagonal elements of the matrix U.

Why the permutation P

It allows us to avoid dividing by 0.

Cinvestav

Thus, What do we want?

We want P, L and U

$$PA = LU$$

However, we move a non-zero element, a_{k1}

From somewhere in the first column to the $(1, 1)$ position of the matrix.

In addition

a_{k1} as the element in the first column with the greatest absolute value.

Thus, What do we want?

We want P, L and U

$$PA = LU$$

However, we move a non-zero element, a_{k1}

From somewhere in the first column to the $(1, 1)$ position of the matrix.

In addition

a_{k1} as the element in the first column with the greatest absolute value.

Cinvestav

Thus, What do we want?

We want P, L and U

$$PA = LU$$

However, we move a non-zero element, a_{k1}

From somewhere in the first column to the $(1,1)$ position of the matrix.

In addition

a_{k1} as the element in the first column with the greatest absolute value.

Cinvestav

Exchange Rows

Thus

We exchange row 1 with row k, or multiplying A by a permutation matrix Q on the left

$$QA = \begin{pmatrix} a_{k1} & w^T \\ v & A' \end{pmatrix}$$

With

- $v = (a_{21}, a_{31}, ..., a_{n1})^T$ with a_{11} replaces a_{k1}.
- $w^T = (a_{k2}, a_{k3}, ..., a_{kn})$.
- A' is a $(n-1) \times (n-1)$

Exchange Rows

Thus

We exchange row 1 with row k, or multiplying A by a permutation matrix Q on the left

$$QA = \begin{pmatrix} a_{k1} & w^T \\ v & A' \end{pmatrix}$$

With

- $v = (a_{21}, a_{31}, ..., a_{n1})^T$ with a_{11} replaces a_{k1}.
- $w^T = (a_{k2}, a_{k3}, ..., a_{kn})$.
- A' is a $(n-1) \times (n-1)$

Cinvestav

Now, $a_{k1} \neq 0$

We have then

$$QA = \begin{pmatrix} a_{k1} & w^T \\ v & A' \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 \\ \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix}$$

Now, $a_{k1} \neq 0$

We have then

$$
\begin{aligned}
QA &= \begin{pmatrix} a_{k1} & w^T \\ v & A' \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix}
\end{aligned}
$$

Cinvestav

Important

Something Notable

if A is nonsingular, then the Schur complement $A' - \frac{vw^T}{a_{k1}}$ is nonsingular, too.

Now, we can find recursively an LUP decomposition for it

$$P'\left(A' - \frac{vw^T}{a_{k1}}\right) = L'U'$$

Then, we define a new permutation matrix

$$P = \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} Q$$

Important

Something Notable

if A is nonsingular, then the Schur complement $A' - \frac{vw^T}{a_{k1}}$ is nonsingular, too.

Now, we can find recursively an LUP decomposition for it

$$P'\left(A' - \frac{vw^T}{a_{k1}}\right) = L'U'$$

Then, we define a new permutation matrix

$$P = \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} Q$$

Important

Something Notable

if A is nonsingular, then the Schur complement $A' - \frac{vw^T}{a_{k1}}$ is nonsingular, too.

Now, we can find recursively an LUP decomposition for it

$$P'\left(A' - \frac{vw^T}{a_{k1}}\right) = L'U'$$

Then, we define a new permutation matrix

$$P = \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} Q$$

Thus

We have

$$
\begin{aligned}
PA &= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} QA \\
&= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & P' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & P'\left(A' - \frac{vw^T}{a_{k1}}\right) \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & L'U' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & L' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Thus

We have

$$
\begin{aligned}
PA &= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} QA \\
&= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & P' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & P'\left(A' - \frac{vw^T}{a_{k1}}\right) \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & L'U' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & L' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Thus

We have

$$
\begin{aligned}
PA &= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} QA \\
&= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P' \frac{v}{a_{k1}} & P' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P' \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & P' \left(A' - \frac{vw^T}{a_{k1}} \right) \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P' \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & L'U' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P' \frac{v}{a_{k1}} & L' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Thus

We have

$$
\begin{aligned}
PA &= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} QA \\
&= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & P' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & P'\left(A' - \frac{vw^T}{a_{k1}}\right) \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & L'U' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & L' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Thus

We have

$$
\begin{aligned}
PA &= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} QA \\
&= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & P' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & P'\left(A' - \frac{vw^T}{a_{k1}}\right) \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & L'U' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & L' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Thus

We have

$$
\begin{aligned}
PA &= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} QA \\
&= \begin{pmatrix} 1 & 0 \\ 0 & P' \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & P' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & A' - \frac{vw^T}{a_{k1}} \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & P'\left(A' - \frac{vw^T}{a_{k1}}\right) \end{pmatrix} \\
&= \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & L'U' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ P'\frac{v}{a_{k1}} & L' \end{pmatrix} \begin{pmatrix} a_{k1} & w^T \\ 0 & U' \end{pmatrix} \\
&= LU
\end{aligned}
$$

Computing a LUP decomposition

Algorithm

LUP-Decomposition(A)

```
n = A.rows
Let π[1..n] new array
for i = 1 to n
    π[i] = i
for k = 1 to n
    p = 0
    for i = k to n
        if |a_ik| > p
            p = |a_ik|
            k' = i
    if p == 0
        error "Singular Matrix"
    Exchange π[k] ⟷ π[k']
    for i = 1 to n
        Exchange a_ki ⟷ a_k'i
    for i = k + 1 to n
        a_ik = a_ik / a_kk
        for j = k + 1 to n
            a_ij = a_ij − a_ik a_kj
```

Computing a LUP decomposition

Algorithm

LUP-Decomposition(A)

$n = A.rows$

Let $\pi[1..n]$ new array

for $i = 1$ **to** n

    $\pi[i] = i$

for $k = 1$ **to** n

    $p = 0$

    for $i = k$ **to** n

        if $|a_{ik}| > p$

            $p = |a_{ik}|$

            $k' = i$

    if $p == 0$

        error "Singular Matrix"

    Exchange $\pi[k] \longleftrightarrow \pi[k']$

    for $i = 1$ **to** n

        Exchange $a_{ki} \longleftrightarrow a_{k'i}$

    for $i = k + 1$ **to** n

        $a_{ik} = \frac{a_{ik}}{a_{kk}}$

        for $j = k + 1$ **to** n

            $a_{ij} = a_{ij} - a_{ik} a_{kj}$

Cinvestav

Computing a LUP decomposition

Algorithm

LUP-Decomposition(A)

$n = A.rows$
Let $\pi[1..n]$ new array
for $i = 1$ **to** n
    $\pi[i] = i$
for $k = 1$ **to** n
    $p = 0$
    for $i = k$ **to** n
        if $|a_{ik}| > p$
            $p = |a_{ik}|$
            $k' = i$
    if $p == 0$
        error "Singular Matrix"
    Exchange $\pi[k] \longleftrightarrow \pi[k']$
    for $i = 1$ **to** n
        Exchange $a_{ki} \longleftrightarrow a_{k'i}$
    for $i = k + 1$ **to** n
        $a_{ik} = \frac{a_{ik}}{a_{kk}}$
        for $j = k + 1$ **to** n
            $a_{ij} = a_{ij} - a_{ik}a_{kj}$

Cinvestav

Computing a LUP decomposition

Algorithm

LUP-Decomposition(A)

$n = A.rows$
Let $\pi\left[1..n\right]$ new array
for $i = 1$ **to** n
    $\pi\left[i\right] = i$
for $k = 1$ **to** n
    $p = 0$
    for $i = k$ **to** n
        if $|a_{ik}| > p$
            $p = |a_{ik}|$
            $k' = i$
    if $p == 0$
        error "Singular Matrix"
    Exchange $\pi\left[k\right] \longleftrightarrow \pi\left[k'\right]$
    for $i = 1$ **to** n
        Exchange $a_{ki} \longleftrightarrow a_{k'i}$
    for $i = k + 1$ **to** n
        $a_{ik} = \frac{a_{ik}}{a_{kk}}$
        for $j = k + 1$ **to** n
            $a_{ij} = a_{ij} - a_{ik}a_{kj}$

Cinvestav

Computing a LUP decomposition

Algorithm

```
LUP-Decomposition(A)
n = A.rows
Let π[1..n] new array
for i = 1 to n
    π[i] = i
for k = 1 to n
    p = 0
    for i = k to n
        if |a_ik| > p
            p = |a_ik|
            k' = i
    if p == 0
        error "Singular Matrix"
    Exchange π[k] ⟷ π[k']
    for i = 1 to n
        Exchange a_ki ⟷ a_k'i
    for i = k + 1 to n
        a_ik = a_ik / a_kk
        for j = k + 1 to n
            a_ij = a_ij − a_ik a_kj
```

Computing a LUP decomposition

Algorithm

LUP-Decomposition(A)

$n = A.rows$
Let $\pi[1..n]$ new array
for $i = 1$ **to** n
    $\pi[i] = i$
for $k = 1$ **to** n
    $p = 0$
    for $i = k$ **to** n
        if $|a_{ik}| > p$
            $p = |a_{ik}|$
            $k' = i$
    if $p == 0$
        error "Singular Matrix"
    Exchange $\pi[k] \longleftrightarrow \pi[k']$
    for $i = 1$ **to** n
        Exchange $a_{ki} \longleftrightarrow a_{k'i}$
    for $i = k + 1$ **to** n
        $a_{ik} = \frac{a_{ik}}{a_{kk}}$
        for $j = k + 1$ **to** n
            $a_{ij} = a_{ij} - a_{ik}a_{kj}$

Computing a LUP decomposition

Algorithm

LUP-Decomposition(A)

```
n = A.rows
Let π[1..n] new array
for i = 1 to n
    π[i] = i
for k = 1 to n
    p = 0
    for i = k to n
        if |a_ik| > p
            p = |a_ik|
            k' = i
    if p == 0
        error "Singular Matrix"
    Exchange π[k] ⟷ π[k']
    for i = 1 to n
        Exchange a_ki ⟷ a_k'i
    for i = k + 1 to n
        a_ik = a_ik / a_kk
        for j = k + 1 to n
            a_ij = a_ij − a_ik a_kj
```

Computing a LUP decomposition

Algorithm

LUP-Decomposition(A)

$n = A.rows$
Let $\pi[1..n]$ new array
for $i = 1$ **to** n
  $\pi[i] = i$
for $k = 1$ **to** n
  $p = 0$
  for $i = k$ **to** n
    if $|a_{ik}| > p$
      $p = |a_{ik}|$
      $k' = i$
  if $p == 0$
    error "Singular Matrix"
  Exchange $\pi[k] \longleftrightarrow \pi[k']$
  for $i = 1$ **to** n
    Exchange $a_{ki} \longleftrightarrow a_{k'i}$
  for $i = k + 1$ **to** n
    $a_{ik} = \frac{a_{ik}}{a_{kk}}$
    for $j = k + 1$ **to** n
      $a_{ij} = a_{ij} - a_{ik}a_{kj}$

Computing a LUP decomposition

Algorithm

LUP-Decomposition(A)

$n = A.rows$
Let $\pi[1..n]$ new array
for $i = 1$ **to** n
    $\pi[i] = i$
for $k = 1$ **to** n
    $p = 0$
    for $i = k$ **to** n
        if $|a_{ik}| > p$
            $p = |a_{ik}|$
            $k' = i$
    if $p == 0$
        error "Singular Matrix"
    Exchange $\pi[k] \longleftrightarrow \pi[k']$
    for $i = 1$ **to** n
        Exchange $a_{ki} \longleftrightarrow a_{k'i}$
    for $i = k + 1$ **to** n
        $a_{ik} = \frac{a_{ik}}{a_{kk}}$
        for $j = k + 1$ **to** n
            $a_{ij} = a_{ij} - a_{ik}a_{kj}$

Computing a LUP decomposition

Example

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

⟹

3	5	5	4	2
2	3	3	4	-2
1	2	0	2	0.6
4	-1	-2	3.4	-1

⟹

3	5	5	4	2
2	3	3	4	-2
1	2	0	2	0.6
4	-1	-2	3.4	-1

⟹

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	-2	0.4	-0.2
4	-1	-1	4.2	-0.6

⟹

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	-2	0.4	-0.2
4	-1	-1	4.2	-0.6

⟹

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	-2	0.4	-0.2
4	-1	-1	4.2	-0.6

⟹

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

⟹

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

⟹

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	0.5	4	-0.5

Computing a LUP decomposition

Example

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	3	3	4	-2
1	**2**	**0**	**2**	**0.6**
4	-1	-2	3.4	-1

Computing a LUP decomposition

Example

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	3	3	4	-2
1	**2**	**0**	**2**	**0.6**
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**3**	3	4	-2
1	**2**	0	2	0.6
4	**-1**	-2	3.4	-1

Computing a LUP decomposition

Example

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	3	3	4	-2
1	**2**	**0**	**2**	**0.6**
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**3**	3	4	-2
1	**2**	0	2	0.6
4	**-1**	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**0.6**	0	1.6	-3.2
1	**0.4**	-2	0.4	-0.2
4	**-1**	-1	4.2	-0.6

Computing a LUP decomposition

Example

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	3	3	4	-2
1	**2**	**0**	**2**	**0.6**
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**3**	3	4	-2
1	**2**	0	2	0.6
4	**-1**	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**0.6**	0	1.6	-3.2
1	**0.4**	-2	0.4	-0.2
4	**-1**	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	-2	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	0.5	4	-0.5

Computing a LUP decomposition

Example

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	3	3	4	-2
1	**2**	**0**	**2**	**0.6**
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**3**	3	4	-2
1	**2**	0	2	0.6
4	**-1**	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**0.6**	0	1.6	-3.2
1	**0.4**	-2	0.4	-0.2
4	**-1**	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	0.5	4	-0.5

Computing a LUP decomposition

Example

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	3	3	4	-2
1	**2**	**0**	**2**	**0.6**
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**3**	3	4	-2
1	**2**	0	2	0.6
4	**-1**	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**0.6**	0	1.6	-3.2
1	**0.4**	-2	0.4	-0.2
4	**-1**	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	0.5	4	-0.5

Computing a LUP decomposition

Example

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	3	3	4	-2
1	**2**	**0**	**2**	**0.6**
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**3**	3	4	-2
1	**2**	0	2	0.6
4	**-1**	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**0.6**	0	1.6	-3.2
1	**0.4**	-2	0.4	-0.2
4	**-1**	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	0.5	4	-0.5

Computing a LUP decomposition

Example

1	2	0	2	0.6
2	3	3	4	-2
3	**5**	5	4	2
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	3	3	4	-2
1	**2**	**0**	**2**	**0.6**
4	-1	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**3**	3	4	-2
1	**2**	0	2	0.6
4	**-1**	-2	3.4	-1

$\Longrightarrow$

3	**5**	**5**	**4**	**2**
2	**0.6**	0	1.6	-3.2
1	**0.4**	-2	0.4	-0.2
4	**-1**	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
2	0.6	0	1.6	-3.2
1	0.4	**-2**	0.4	-0.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	-1	4.2	-0.6

$\Longrightarrow$

3	5	5	4	2
1	0.4	-2	0.4	-0.2
2	0.6	0	1.6	-3.2
4	-1	0.5	4	-0.5

Finally, you get

The Permutation and Decomposition

$$\underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}}_{P}\underbrace{\begin{pmatrix} 2 & 0 & 2 & 0.6 \\ 3 & 3 & 4 & -2 \\ 5 & 5 & 4 & 2 \\ -1 & -2 & 3.4 & -1 \end{pmatrix}}_{A} = \dots$$

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0.4 & 1 & 0 & 0 \\ -0.2 & 0.5 & 1 & 0 \\ 0.6 & 0 & 0.4 & 1 \end{pmatrix}}_{L}\underbrace{\begin{pmatrix} 5 & 5 & 4 & 2 \\ 0 & -2 & 0.4 & -0.2 \\ 0 & 0 & 4 & -0.5 \\ 0 & 0 & 0 & -3 \end{pmatrix}}_{U}$$

Outline

Cinvestav

Symmetric positive-definite matrices

Lemma 28.9

Any symmetric positive-definite matrix is nonsingular.

Lemma 28.10

If A is a symmetric positive-definite matrix, then every leading submatrix of A is symmetric and positive-definite.

Symmetric positive-definite matrices

Lemma 28.9

Any symmetric positive-definite matrix is nonsingular.

Lemma 28.10

If A is a symmetric positive-definite matrix, then every leading submatrix of A is symmetric and positive-definite.

Symmetric positive-definite matrices

Definition: Schur complement

Let A be a symmetric positive-definite matrix, and let A_k be a leading $k \times k$ submatrix of A. Partition A as:

$$A = \begin{pmatrix} A_k & B^T \\ B & C \end{pmatrix}$$

Then, the Schur complement of A with respect to A_k is defined to be

$$S = C - BA_k^{-1}B^T$$

Symmetric positive-definite matrices

Definition: Schur complement

Let A be a symmetric positive-definite matrix, and let A_k be a leading $k \times k$ submatrix of A. Partition A as:

$$A = \begin{pmatrix} A_k & B^T \\ B & C \end{pmatrix}$$

Then, the Schur complement of A with respect to A_k is defined to be

$$S = C - BA_k^{-1}B^T$$

Symmetric positive-definite matrices

Definition: Schur complement

Let A be a symmetric positive-definite matrix, and let A_k be a leading $k \times k$ submatrix of A. Partition A as:

$$A = \begin{pmatrix} A_k & B^T \\ B & C \end{pmatrix}$$

Then, the Schur complement of A with respect to A_k is defined to be

$$S = C - BA_k^{-1}B^T$$

Symmetric positive-definite matrices

Definition: Schur complement

Let A be a symmetric positive-definite matrix, and let A_k be a leading $k \times k$ submatrix of A. Partition A as:

$$A = \begin{pmatrix} A_k & B^T \\ B & C \end{pmatrix}$$

Then, the Schur complement of A with respect to A_k is defined to be

$$S = C - BA_k^{-1}B^T$$

Symmetric positive-definite matrices

Lemma 28.11 (Schur complement lemma)

If A is a symmetric positive-definite matrix and A_k is a leading $k \times k$ submatrix of A, then the Schur complement of A with respect to A_k is symmetric and positive-definite.

Corollary 28.12

LU decomposition of a symmetric positive-definite matrix never causes a division by 0.

Symmetric positive-definite matrices

Lemma 28.11 (Schur complement lemma)

If A is a symmetric positive-definite matrix and A_k is a leading $k \times k$ submatrix of A, then the Schur complement of A with respect to A_k is symmetric and positive-definite.

Corollary 28.12

LU decomposition of a symmetric positive-definite matrix never causes a division by 0.

Outline

Cinvestav

Inverting matrices

LUP decomposition can be used to compute a matrix inverse

The computation of a matrix inverse can be speed up using techniques such as Strassen's algorithm for matrix multiplication.

Computing a matrix inverse from a LUP decomposition

Proceed as follows

- The equation $AX = I_n$ can be viewed as a set of n distinct equations of the form $A_{x_i} = e_i$, for $i = 1, ..., n$.
- We have a LUP decomposition of a matrix A in the form of three matrices L, U, and P such that $PA = LU$.
- Then we use the backward-forward to solve $AX_i = e_i$.

Complexity

First

- We can compute each X_i in time $\Theta\left(n^2\right)$.
- Thus, X can be computed in time $\Theta\left(n^3\right)$.
- LUP decomposition is computed in time $\Theta\left(n^3\right)$.

Finally

We can compute A^{-1} of a matrix A in time $\Theta\left(n^3\right)$.

Cinvestav

Complexity

First

- We can compute each X_i in time $\Theta\left(n^2\right)$.
- Thus, X can be computed in time $\Theta\left(n^3\right)$.
- LUP decomposition is computed in time $\Theta\left(n^3\right)$.

Finally

We can compute A^{-1} of a matrix A in time $\Theta\left(n^3\right)$.

Cinvestav

Matrix multiplication and matrix inversion

Theorem 28.7

If we can invert an $n \times n$ matrix in time $I(n)$, where $I(n) = \Omega(n^2)$ and $I(n)$ satisfies the regularity condition $I(3n) = O(I(n))$, then we can multiply two $n \times n$ matrices in time $O(I(n))$.

Matrix multiplication and matrix inversion

Theorem 28.8

If we can multiply two $n \times n$ real matrices in time $M(n)$, where $M(n) = \Omega(n^2)$ and $M(n) = O(M(n+k))$ for any k in range $0 \leq k \leq n$ and $M(\frac{n}{2}) \leq cM(n)$ for some constant $c < \frac{1}{2}$. Then we can compute the inverse of any real nonsingular $n \times n$ matrix in time $O(M(n))$.

Cinvestav

Outline

Cinvestav

Least-squares Approximation

Fitting curves to given sets of data points is an important application of symmetric positive-definite matrices.

Given

$$(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)$$

where the y_i are known to be subject to measurement errors. We would like to determine a function $F(x)$ such that:

$$y_i = F(x_i) + \eta_i$$

for $i = 1, 2, ..., m$

Cinvestav

Least-squares Approximation

Continuation

The form of the function F depends on the problem at hand.

$$F(x) = \sum_{j=1}^{n} c_j f_j(x)$$

A common choice is $f_j(x) = x^{j-1}$, which means that

$$F(x) = c_1 + c_2 x + c_3 x^2 + ... + c_n x^{n-1}$$

is a polynomial of degree $n-1$ in x.

Least-squares Approximation

Continuation

Let

$$A = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \dots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_m) & f_2(x_m) & \dots & f_n(x_m) \end{pmatrix}$$

denote the matrix of values of the basis functions at the given points; that is, $a_{ij} = f_j(x_i)$. Let $c = (c_k)$ denote the desired size-n vector of coefficients. Then,

$$A = \begin{pmatrix} f_1(x_1) & f_2(x_1) & \dots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_m) & f_2(x_m) & \dots & f_n(x_m) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} F(x_1) \\ F(x_2) \\ \vdots \\ F(x_m) \end{pmatrix}$$

Least-squares Approximation

Then

Thus, $\eta = Ac - y$ is the size of approximation errors. To minimize approximation errors, we choose to minimize the norm of the error vector , which gives us a least-squares solution.

$$||\eta||^2 = ||Ac - y||^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} c_j - y_i \right)^2$$

Thus

We can minimize $||\eta||$ by differentiating $||\eta||$ with respect to each c_k and then setting the result to 0:

$$\frac{d||\eta||^2}{dc_k} = \sum_{i=1}^{m} 2 \left(\sum_{j=1}^{n} a_{ij} c_j - y_i \right) a_{ik} = 0$$

Least-squares Approximation

Then

Thus, $\eta = Ac - y$ is the size of approximation errors. To minimize approximation errors, we choose to minimize the norm of the error vector , which gives us a least-squares solution.

$$||\eta||^2 = ||Ac - y||^2 = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} c_j - y_i \right)^2$$

Thus

We can minimize $||\eta||$ by differentiating $||\eta||$ with respect to each c_k and then setting the result to 0:

$$\frac{d||\eta||^2}{dc_k} = \sum_{i=1}^{m} 2 \left(\sum_{j=1}^{n} a_{ij} c_j - y_i \right) a_{ik} = 0$$

Least-squares Approximation

We can put all derivatives

The n equation for $k = 1, 2, ..., n$

$$(Ac - y)^T A = 0$$

or equivalently to

$$A^T (Ac - y) = 0$$

which implies

$$A^T Ac = A^T y$$

Least-squares Approximation

Continuation

The $A^T A$ is symmetric:

- If A has full column rank, then $A^T A$ is positive- definite as well.

Hence, $(A^T A)^{-1}$ exists, and the solution to equation $A^T A c = A^T y$ is

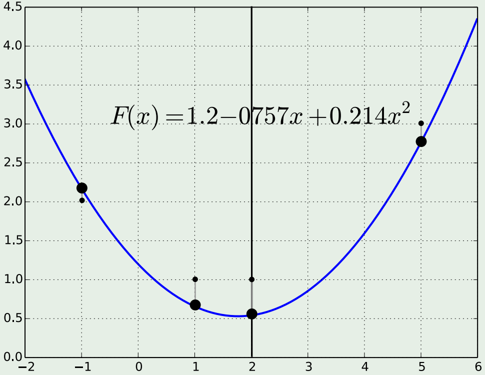
$$c = ((A^T A)^{-1} A^T) y = A^+ y$$

where the matrix $A^+ = ((A^T A)^{-1} A^T)$ is called the pseudoinverse of the matrix A.

Least-Square Approximation

Continuation

As an example of producing a least-squares fit, suppose that we have 5 data points (-1,2), (1,1),(2,1),(3,0),(5,3), shown as black dots in following figure



Least-squares Approximation

Continuation

We start with the matrix of basis-function values

$$A = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_5^2 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ 1 & 5 & 25 \end{pmatrix}$$

whose pseudoinverse is

$$A^{+} = \begin{pmatrix} 0.500 & 0.300 & 0.200 & 0.100 & -0.100 \\ -0.388 & 0.093 & 0.190 & 0.193 & -0.088 \\ 0.060 & -0.036 & -0.048 & -0.036 & 0.060 \end{pmatrix}$$

Matrix multiplication and matrix inversion

Continuation

Multiplying y by A^{+} , we obtain the coefficient vector

$$c = \begin{pmatrix} 1.200 \\ -0.757 \\ 0.214 \end{pmatrix}$$

which corresponds to the quadratic polynomial

$$F(x) = 1.200 - 0.757x + 0.214x^2$$

Cinvestav

Outline

Cinvestav

Exercises

From Cormen's book solve

- 34.5-1
- 34.5-2
- 34.5-3
- 34.5-4
- 34.5-5
- 34.5-7
- 34.5-8