

# Matrices

## 1 Announcement

Dear Dr. Zhao:

Please announce in class that in assignment number 4 in problem number three the definition should be ... expr[4] [46] ...

## 2 Matrix Computations

The remaining assignments will use matrices.

1. **matrix product:**  $C = AB$  where  $A$  is a  $m \times n$  matrix,  $B$  is a  $n \times l$  matrix.

$$C_{ij} = \sum_k A_{ik} B_{kj}.$$

The column number of  $A$  is equal to the row number of  $B$ .

2. **eigenvalues:**  $\lambda$  is called an eigenvalue and  $\xi \neq 0$  is called an eigenvector of  $A$  if

$$A\xi = \lambda\xi.$$

Keep in mind that  $\det A = \lambda_1 \lambda_2 \cdots \lambda_n$ , where  $\lambda_i$ 's are eigenvalues of  $A$ . In fact, eigenvalues are roots of the polynomial equation

$$\det(\lambda I - A) = 0.$$

The identity matrix  $I$  is nonsingular and  $\det I = 1$ .

3. **rank** the rank of a matrix is the same as its row rank, and also its column rank. The row rank of a matrix  $A$  is defined as the maximum number of linear independent row vectors of  $A$ .
4. **inverse matrix**  $A$  is called singular if  $\det A = 0$ , and nonsingular otherwise. We can define the inverse matrix for a nonsingular matrix. The inverse matrix  $A^{-1}$  is such a matrix that

$$A^{-1}A = AA^{-1} = I.$$

5. **System of linear equations:**

$$Ax = b.$$

If  $A$  is nonsingular, then there is a unique solution  $x = A^{-1}b$ . Otherwise, there are infinitely many solutions or there is no solution.

6. **structured matrices** 1.  $A$  is called a symmetric matrix if the transpose of  $A$  is itself  $A^T = A$ . In practice, a Hessian matrix is symmetric. All eigenvalues of a real symmetric matrix are real. A matrix is said to be banded if all of the nonzero elements lie in a band close to the main diagonal. A matrix is said to be sparse if the number of nonzero elements is few. There matrices occur when we numerically solve a partial differential equations. For example

$$-au_{xx} = f, 0 < x < 1, u(0) = 0, u(1) = 1.$$

Denote  $u = (u_0, u_1, \dots, u_n)$ , then the finite difference scheme is

$$-a \frac{u_{i+1} - 2u_i + u_{i-1}}{dx^2} = f_i.$$

We will have a linear system of equations where  $A$  is sparse and banded and symmetric.

7. **positive definite:** A symmetric matrix  $A$  is positive definite if  $x^T Ax > 0$  whenever  $x \neq 0$ .
8. **partitioned matrices** The elements of the matrix are matrices. For example  $Ax = b$ , where

$$A = (a_1, a_2, \dots, a_n) \text{ and } x = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}.$$

9. **numpy matrix operations:** dot, inner, outer, identity, diagonal.
10. **Efficient matrix operations**

### 3 Triangular Factorizations

1. **Gauss-Jordan elimination** The primary application of a triangular factorization is to solve  $Ax = b$ . A *flop* is one floating-point operation. You will see this many times.
2. **comparison of cost of different approaches solving  $Ax = b$ :** The total cost for Gauss-Jordan elimination is  $n^3 + O(n^2)$  flops. The total cost by triangular factorization is  $\frac{2}{3}n^3 + O(n^2)$  flops. The total cost using matrix inversion is  $2n^3 + O(n^2)$  flops.
3. **LU factorization:**  $A = LU$  where  $U$  is upper triangular and  $L$  is unit lower triangular. For example:

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

The LU factorization may not always exist, for example

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

The LU factorization always exists if  $A$  is square and *strictly diagonally dominant*, meaning that

$$|a_{ii}| > |a_{i1}| + \dots + |a_{i,i-1}| + |a_{i,i+1}| + \dots + |a_{in}| \text{ for } i = 1, 2, \dots, n$$

4. **Gaussian elimination** The unknowns  $l_{21}$  and  $u_{11}$  et. al are 9 unknowns and there are totally 9 equations. It seems that LU factorization becomes of a solving a linear system.

$$\begin{matrix} m_{21} \\ m_{31} \end{matrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \longrightarrow \begin{matrix} \\ m_{32} \end{matrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & a'_{32} & a'_{33} \end{pmatrix} \longrightarrow \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ 0 & a'_{22} & a'_{23} \\ 0 & 0 & a''_{33} \end{pmatrix}$$

Then

$$L = \begin{pmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{pmatrix}$$

5. algorithm

```

for k in range(n-1):
    for i in range(k+1, n):
        m = a[i,k]/a[k,k]
        for j in range(k+1, n):
            *****
            a[i,j] = a[i,j] - m*a[k,j]
            *****
            * a[i,j] = a[i,j] - (a[i,k]/a[k,k])*a[k,j]
            *****
            a[i,k] = m

```

The cost of this algorithm is  $(2/3)n^3 + O(n^2)$  flops.

6. Gaussian elimination viewed as a recursive algorithm

$$\begin{pmatrix} 1 & 0^T \\ l & \hat{L} \end{pmatrix} \begin{pmatrix} \omega & u^T \\ 0 & \hat{U} \end{pmatrix} = \begin{pmatrix} \alpha & b^T \\ a & \hat{A} \end{pmatrix}$$

Thus,

$$\begin{aligned} \omega &= \alpha \\ u^T &= b^T \\ l &= a/\omega \\ \hat{L}\hat{U} &= \hat{A} - lu^T \end{aligned}$$

7. **Gaussian elimination with partial pivoting:** Gaussian elimination is in general numerically unstable. This can be fixed by incorporating row interchanges—partial pivoting.

```

p = range(n)
for k in range(n-1):
    ell = k
    for i in range(k+1,n):
        if abs(a[i,k]) > abs(a[ell,k]): ell = i
    p[k], p[ell] = p[ell], p[k]
    for j in range(n):

```

```

    a[k,j], a[ell,j] = a[ell,j], a[k,j]
for i in range(k+1,n):
    m = a[i,k]/a[k,k]
    for j in range(k+1,n):
        a[i,j] = a[i,j] - m*a[k,j]
    a[i, k] = m

```

8. Division by a triangular matrix:  $Ax = b$  becomes  $LUx = b$ . Then  $x = (LU)^{-1}b = U^{-1}L^{-1}b = U \setminus (L \setminus b)$ . Performing  $x = U \setminus y$  is equivalent to solve  $Ux = y$ . This can be done by *back substitution*.

$$\begin{aligned}
 x_n &= y_n/u_{nn} \\
 x_i &= (y_i - u_{i,i+1}x_{i+1} - \dots - u_{i,n}x_n)/u_{ii}, \text{ for } i = n-1, n-2, \dots, 1.
 \end{aligned}$$

Python code

```

for i in range(n-1, -1, -1): # n-1, n-2, ..., 0
    temp = y[i]
    *****
    for j in range(i+1, n): # i+1, i+2, ..., n-1
        temp = temp - u[i, j]*x[j]
    *****
    * if i < n-1: temp =temp - dot(u[i, i+1:n], x[i+1:n])
    *****
    x[i] = temp/u[i, i]

```

9. Cholesky factorization Gaussian elimination without pivoting is stable for a symmetric positive definite matrix. Cholesky factorization is stated as

$$A = GG^T$$

where  $G$  is lower triangular with positive diagonal elements. The cost of a Cholesky factorization is  $\frac{1}{3}n^3 + O(n^2)$  flops.

10. extensive example

$$\begin{pmatrix} 2 & 1 & -1 & 8 \\ -3 & -1 & 2 & -11 \\ -2 & 1 & 2 & -3 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 2 & 1 & 5 \end{pmatrix}$$

$$\begin{pmatrix} 2 & 1 & -1 & 8 \\ 0 & 1/2 & 1/2 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$