BMEN/ENGR 636 Introduction to the Finite Element Method
Hart

# SUMMARY OF MATRIX MANIPULATIONS

Primary Reference: Bathe, K-J., and Wilson, E.L. *Numerical Methods in Finite Element Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

## I. MATRIX

*Definition:* A matrix is an array of ordered elements (numbers). A general matrix consists of $m$ by $n$ numbers arranged in $m$ rows and $n$ columns as follows:

$$[\mathbf{A}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

if $n = 1 \Rightarrow$ column vector $\{\mathbf{A}\}$
if $m = 1 \Rightarrow$ row vector $\lfloor \mathbf{A} \rfloor$

### *A. REAL MATRIX: All elements are real numbers. This restricted class is what we will always imply.*

### *B. TRANSPOSE OPERATION*

The transposed matrix [**A**] is written $[\mathbf{A}]^T$ and is found by interchanging the rows and columns of [**A**].

Example:

$$[\mathbf{A}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \quad (2\text{x}3)$$

$$[\mathbf{A}]^T = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \\ a_{13} & a_{23} \end{bmatrix} \quad (3\text{x}2)$$

Note: $\lfloor \mathbf{A} \rfloor = \{\mathbf{A}\}^T$

IF $[\mathbf{A}] = [\mathbf{A}]^T$ (they are "equal" if and only if all elements in corresponding locations are same) THEN $m = n$ (square) and $[\mathbf{A}]$ is said to be **symmetric**

Note: Symmetric implies the matrix is square, but (of course!) a square matrix does not imply symmetry.

### C. IDENTITY OR UNIT MATRIX $[\mathbf{I}]$ or $[\mathbf{1}]$

*Definition:* Square matrix of order $n$ with diagonal elements = 1, all other elements = 0.
Example: Identity matrix of order 3

$$[\mathbf{I}]_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A diagonal matrix has all off-diagonal elements = 0. (The identity matrix is a special case of a diagonal matrix).

### D. NULL MATRIX $[\mathbf{0}]$

*Definition:* Matrix with all elements = 0.

## II. MATRIX ALGEBRA

### A. EQUALITY

$[\mathbf{A}] = [\mathbf{B}] \Leftrightarrow mA = mB,\ nA = nB$ (number of rows and columns same),
and $a_{ij} = b_{ij},\quad i = 1,2,\ldots m,\ j = 1,2,\ldots n$

### B. ADDITION

$[\mathbf{A}] + [\mathbf{B}]$ exists $\Leftrightarrow mA = mB,\ nA = nB$
and if $[\mathbf{A}] + [\mathbf{B}] = [\mathbf{C}]$, then $c_{ij} = a_{ij} + b_{ij}$
Note: $[\mathbf{A}] + [\mathbf{B}] = [\mathbf{B}] + [\mathbf{A}]$

### C. SUBTRACTION

$[\mathbf{A}] - [\mathbf{B}]$ exists $\Leftrightarrow mA = mB,\ nA = nB$
and if $[\mathbf{A}] - [\mathbf{B}] = [\mathbf{C}]$, then $c_{ij} = a_{ij} - b_{ij}$

### D. MULTIPLICATION BY A SCALAR, b

$b[\mathbf{A}] = [\mathbf{C}]$ where $c_{ij} = b \cdot a_{ij}$
Example:

$$b = 0.5,\ [\mathbf{A}] = \begin{bmatrix} 1 & 0 & 2 \\ 1 & 3 & 1 \end{bmatrix}$$

$$[\mathbf{C}] = b[\mathbf{A}] = \begin{bmatrix} 0.5 & 0 & 1 \\ 0.5 & 1.5 & 0.5 \end{bmatrix}$$

## E. MULTIPLICATION OF MATRICES

$[\mathbf{A}][\mathbf{B}] = [\mathbf{C}]$

$[\mathbf{A}]$ is post - multiplied by $[\mathbf{B}]$

$[\mathbf{B}]$ is pre - multiplied by $[\mathbf{A}]$

if $[\mathbf{A}]$ is $m$ x $p$, and $[\mathbf{B}]$ is $q$ x $n$, then $[\mathbf{A}][\mathbf{B}]$ exists $\Leftrightarrow$ $p = q$

($i.e.$, the number of columns of $[\mathbf{A}] =$ the number of rows of $[\mathbf{B}]$)

$$c_{ij} = \sum_{r=1}^{p} a_{ir} \cdot b_{rj}, \begin{cases} i = 1,2,\ldots m \\ j = 1,2,\ldots n \end{cases}$$

where $[\mathbf{C}]$ is $m$ x $n$.

1. Matrix multiplication is <u>not</u> commutative
   Example: $[\mathbf{A}][\mathbf{B}] \neq [\mathbf{B}][\mathbf{A}]$
2. Matrix multiplication is distributive and associative
   Example 1: $[\mathbf{E}] = ([\mathbf{A}] + [\mathbf{B}])[\mathbf{C}] = [\mathbf{A}][\mathbf{C}] + [\mathbf{B}][\mathbf{C}]$
   Example 2: $[\mathbf{G}] = ([\mathbf{A}][\mathbf{B}])[\mathbf{C}] = [\mathbf{A}]([\mathbf{B}][\mathbf{C}]) = [\mathbf{A}][\mathbf{B}][\mathbf{C}]$

   Note: If $[\mathbf{A}][\mathbf{B}] = [\mathbf{C}][\mathbf{B}]$, $[\mathbf{A}]$ does not necessairly equal $[\mathbf{C}]$

***SPECIAL RULES***

1. The transpose of the product of matrices equals the product of the transposed matrices taken in reverse order.
   Thus: $([\mathbf{A}][\mathbf{B}])^T = [\mathbf{B}]^T[\mathbf{A}]^T$

2. If $[\mathbf{A}]$ is symmetric, and although $[\mathbf{A}][\mathbf{B}]$ is not necessarily symmetric, $[\mathbf{B}]^T[\mathbf{A}][\mathbf{B}]$is always symmetric.

## F. INVERSE (for m x m only) $[\mathbf{A}]^{-1}$

Assume the inverse exists. Then the elements of [A] are such that:

$[\mathbf{A}]^{-1}[\mathbf{A}] = [\mathbf{I}]$ and $[\mathbf{A}][\mathbf{A}]^{-1} = [\mathbf{I}]$

A matrix for which an inverse exists is <u>non</u>-<u>singular</u>.
A matrix for which no inverse exists is <u>singular</u>.

## G. TRACE OF A MATRIX (for m x m only)

$$tr[\mathbf{A}] = \sum_{i=1}^{m} a_{ii}$$

### *H. DETERMINANT (for* m *x* m *only)* $\det[\mathbf{A}]$ or $|\mathbf{A}|$

$$\det[\mathbf{A}] = |\mathbf{A}| = \sum_{j=1}^{m} (-1)^{1+j} a_{1j} \det A_{1j}$$

where $A_{1j}$ is the ($m$ - 1) x ($m$ - 1) matrix obtained
by eliminating the 1st row and jth column from $[\mathbf{A}]$.

### *I. BANDWIDTH (for symmetric,* m *x* m *only)*

Definition of BANDEDNESS: All elements beyond the bandwidth = 0.

Since $[\mathbf{A}] = [\mathbf{A}]^T$ (symmetric)
then $a_{ij} = 0$ for $j > 1 + b$
where $2b + 1$ = bandwidth of $[\mathbf{A}]$
and $b$ = half-bandwidth of $[\mathbf{A}]$
Note: If $b = 0$, [A] is diagonal.

## III. NUMBER OF OPERATIONS

### *A. ADDITION of 2* m *x* n *matrices:*

*mn* additions.

### *B. SUBTRACTION of 2* m *x* n *matrices:*

*mn* subtractions.

### *C. SCALAR MULTIPLICATION of a* m *x* n *matrix:*

*mn* multiplications

### *D. MATRIX MULTIPLICATION of an* m *x* n *matrix with an* n *x* p *matrix:*

*mnp* multiplications (this can often be reduced by taking advantage of known zero elements)

## IV. MATLAB

***The following pages, taken from the MATLAB 7 on-line Help system provides an overview of MATLAB matrix operations.***

Note MATLAB has specific capabilities for working with sparse matrices, common in finite element analyses.

# Arithmetic Operators + - * / \ ^ '

Matrix and array arithmetic

## Syntax

```
A+B
A-B
A*B      A.*B
A/B      A./B
A\B      A.\B
A^B      A.^B
A'       A.'
```

## Description

MATLAB has two different types of arithmetic operations. Matrix arithmetic operations are defined by the rules of linear algebra. Array arithmetic operations are carried out element by element, and can be used with multidimensional arrays. The period character (`.`) distinguishes the array operations from the matrix operations. However, since the matrix and array operations are the same for addition and subtraction, the character pairs `.+` and `.-` are not used.

| | |
|---|---|
| `+` | Addition or unary plus. `A+B` adds `A` and `B`. `A` and `B` must have the same size, unless one is a scalar. A scalar can be added to a matrix of any size. |
| `-` | Subtraction or unary minus. `A-B` subtracts `B` from `A`. `A` and `B` must have the same size, unless one is a scalar. A scalar can be subtracted from a matrix of any size. |
| `*` | Matrix multiplication. `C = A*B` is the linear algebraic product of the matrices `A` and `B`. More precisely, $$C(i,j) = \sum_{k=1}^{n} A(i,k)B(k,j)$$ For nonscalar `A` and `B`, the number of columns of `A` must equal the number of rows of `B`. A scalar can multiply a matrix of any size. |
| `.*` | Array multiplication. `A.*B` is the element-by-element product of the arrays `A` and `B`. `A` and `B` must have the same size, unless one of them is a scalar. |
| `/` | Slash or matrix right division. `B/A` is roughly the same as `B*inv(A)`. More precisely, `B/A = (A'\B')'`. See `\`. |
| `./` | Array right division. `A./B` is the matrix with elements `A(i,j)/B(i,j)`. `A` and `B` must have the same size, unless one of them is a scalar. |
| `\` | Backslash or matrix left division. If `A` is a square matrix, `A\B` is roughly the same as `inv(A)*B`, except it is computed in a different way. If `A` is an `n`-by-`n` matrix and `B` is a column vector with `n` components, or a matrix with several such columns, then `X = A\B` is the solution to the equation AX = B computed by Gaussian elimination (see Algorithm for details). A warning message is displayed if `A` is badly scaled or nearly singular. |

| | |
|---|---|
| | If `A` is an `m`-by-`n` matrix with `m ~= n` and `B` is a column vector with `m` components, or a matrix with several such columns, then `X = A\B` is the solution in the least squares sense to the under- or overdetermined system of equations AX = B. The effective rank, `k`, of `A` is determined from the QR decomposition with pivoting (see "Algorithm" for details). A solution `X` is computed that has at most `k` nonzero components per column. If `k < n`, this is usually not the same solution as `pinv(A)*B`, which is the least squares solution with the smallest norm, $\|X\|$. |
| `.\` | Array left division. `A.\B` is the matrix with elements `B(i,j)/A(i,j)`. `A` and `B` must have the same size, unless one of them is a scalar. |
| `^` | Matrix power. `X^p` is `X` to the power `p`, if `p` is a scalar. If `p` is an integer, the power is computed by repeated squaring. If the integer is negative, `X` is inverted first. For other values of `p`, the calculation involves eigenvalues and eigenvectors, such that if `[V,D] = eig(X)`, then `X^p = V*D.^p/V`.<br><br>If `x` is a scalar and `P` is a matrix, `x^P` is `x` raised to the matrix power `P` using eigenvalues and eigenvectors. `X^P`, where `X` and `P` are both matrices, is an error. |
| `.^` | Array power. `A.^B` is the matrix with elements `A(i,j)` to the `B(i,j)` power. `A` and `B` must have the same size, unless one of them is a scalar. |
| `'` | Matrix transpose. `A'` is the linear algebraic transpose of `A`. For complex matrices, this is the complex conjugate transpose. |
| `.'` | Array transpose. `A.'` is the array transpose of `A`. For complex matrices, this does not involve conjugation. |

## Nondouble Data Type Support

This section describes the arithmetic operators' support for data types other than `double`.

### Data Type single

You can apply any of the arithmetic operators to arrays of type `single` and MATLAB returns an answer of type `single`. You can also combine an array of type `double` with an array of type `single`, and the result has type `single`.

### Integer Data Types

You can apply most of the arithmetic operators to real arrays of the following integer data types:

- `int8` and `uint8`
- `int16` and `uint16`
- `int32` and `uint32`

All operands must have the same integer data type and MATLAB returns an answer of that type.

> Note The arithmetic operators do not support operations on the data types `int64` or `uint64`. Except for the unary operators `+A` and `A.'`, the arithmetic operators do not support operations on complex arrays of any integer data type.

For example,

```
x = int8(3) + int8(4);
class(x)

ans =

int8
```

The following table lists the binary arithmetic operators that you can apply to arrays of the same integer data type. In the table, `A` and `B` are arrays of the same integer data type and `c` is a scalar of type `double` or the same type as `A` and `B`.

| Operation | Support when A and B Have Same Integer Type |
|---|---|
| `+A`, `-A` | Yes |
| `A+B`, `A+c`, `c+B` | Yes |
| `A-B`, `A-c`, `c-B` | Yes |
| `A.*B` | Yes |
| `A*c`, `c*B` | Yes |
| `A*B` | No |
| `A/c`, `c/B` | Yes |
| A.\B, A./B | Yes |
| A\B, A/B | No |
| `A.^B` | Yes, if B has nonnegative integer values. |
| `c^k` | Yes, for a scalar `c` and a nonnegative scalar integer `k`, which have the same integer data type or one of which has type `double` |
| `A.'`, `A'` | Yes |

## Combining Integer Data Types with Type Double

For the operations that support integer data types, you can combine a scalar or array of an integer data type with a scalar, but not an array, of type `double` and the result has the same integer data type as the input of integer type. For example,

```
y = 5 + int32(7);
class(y)

ans =

int32
```

However, you cannot combine an array of an integer data type with either of the following:

- A scalar or array of a different integer data type
- A scalar or array of type `single`

Nondouble Data Types, in the online MATLAB documentation, provides more information about operations on nondouble data types.

## Remarks

The arithmetic operators have M-file function equivalents, as shown:

| | | |
|---|---|---|
| Binary addition | `A+B` | `plus(A,B)` |
| Unary plus | `+A` | `uplus(A)` |
| Binary subtraction | `A-B` | `minus(A,B)` |
| Unary minus | `-A` | `uminus(A)` |
| Matrix multiplication | `A*B` | `mtimes(A,B)` |
| Arraywise multiplication | `A.*B` | `times(A,B)` |
| Matrix right division | `A/B` | `mrdivide(A,B)` |
| Arraywise right division | `A./B` | `rdivide(A,B)` |
| Matrix left division | `A\B` | `mldivide(A,B)` |
| Arraywise left division | `A.\B` | `ldivide(A,B)` |
| Matrix power | `A^B` | `mpower(A,B)` |
| Arraywise power | `A.^B` | `power(A,B)` |
| Complex transpose | `A'` | `ctranspose(A)` |
| Matrix transpose | `A.'` | `transpose(A)` |

Note For some toolboxes, the arithmetic operators are overloaded, that is, they perform differently in the context of that toolbox. To see the toolboxes that overload a given operator, type `help` followed by the operator name. For example, type `help plus`. The toolboxes that overload `plus` (+) are listed. For information about using the operator in that toolbox, see the documentation for the toolbox.

## Examples

Here are two vectors, and the results of various matrix and array operations on them, printed with `format rat`.

| Matrix Operations | | Array Operations | |
|---|---|---|---|
| `x` | 1<br>2<br>3 | `y` | 4<br>5<br>6 |
| `x'` | 1 2 3 | `y'` | 4 5 6 |
| `x+y` | 5<br>7<br>9 | `x-y` | -3<br>-3<br>-3 |

| | | | |
|---|---|---|---|
| `x + 2` | 3<br>4<br>5 | `x-2` | -1<br>0<br>1 |
| `x * y` | Error | `x.*y` | 4<br>10<br>18 |
| `x'*y` | 32 | `x'.*y` | Error |
| `x*y'` | 4 5 6<br>8 10 12<br>12 15 18 | `x.*y'` | Error |
| `x*2` | 2<br>4<br>6 | `x.*2` | 2<br>4<br>6 |
| `x\y` | 16/7 | `x.\y` | 4<br>5/2<br>2 |
| `2\x` | 1/2<br>1<br>3/2 | `2./x` | 2<br>1<br>2/3 |
| `x/y` | 0 0 1/6<br>0 0 1/3<br>0 0 1/2 | `x./y` | 1/4<br>2/5<br>1/2 |
| `x/2` | 1/2<br>1<br>3/2 | `x./2` | 1/2<br>1<br>3/2 |
| `x^y` | Error | `x.^y` | 1<br>32<br>729 |
| `x^2` | Error | `x.^2` | 1<br>4<br>9 |
| `2^x` | Error | `2.^x` | 2<br>4<br>8 |
| `(x+i*y)'` | 1 - 4i 2 - 5i 3 - 6i | | |
| `(x+i*y).'` | 1 + 4i 2 + 5i 3 + 6i | | |

## Algorithm

The specific algorithm used for solving the simultaneous linear equations denoted by `X = A\B` and `X = B/A` depends upon the structure of the coefficient matrix `A`. To determine the structure of `A` and select the appropriate algorithm, MATLAB follows this precedence:

1. If A is sparse, square, and banded, then banded solvers are used. Band density is (# nonzeros in the band)/(# nonzeros in a full band). Band density = `1.0` if there are no zeros on any of the three diagonals.
    - If `A` is real and tridiagonal, i.e., band density = `1.0`, and `B` is real with only one column, `X` is computed quickly using Gaussian elimination without pivoting.

   - If the tridiagonal solver detects a need for pivoting, or if `A` or `B` is not real, or if `B` has more than one column, but `A` is banded with band density greater than the spparms parameter `'bandden'` (default = `0.5`), then `X` is computed using LAPACK.
2. If A is an upper or lower triangular matrix, then `X` is computed quickly with a backsubstitution algorithm for upper triangular matrices, or a forward substitution algorithm for lower triangular matrices. The check for triangularity is done for full matrices by testing for zero elements and for sparse matrices by accessing the sparse data structure.
3. If A is a permutation of a triangular matrix, then `X` is computed with a permuted backsubstitution algorithm.
4. If A is symmetric, or Hermitian, and has real positive diagonal elements, then a Cholesky factorization is attempted (see chol). If `A` is found to be positive definite, the Cholesky factorization attempt is successful and requires less than half the time of a general factorization. Nonpositive definite matrices are usually detected almost immediately, so this check also requires little time. If successful, the Cholesky factorization is

   ```
   A = R'*R
   ```

   where `R` is upper triangular. The solution `X` is computed by solving two triangular systems,

   ```
   X = R\(R'\B)
   ```

   If `A` is sparse, a symmetric minimum degree preordering is applied first (see `symmmd` and `spparms`) before `X` is computed. The algorithm is

   ```
   perm = symmmd(A);          % Symmetric approximate minimum
                              % degree reordering
   R = chol(A(perm,perm));    % Cholesky factorization
   Y = R'\B(perm);            % Lower triangular solve
   X(perm,:) = R\Y;           % Upper triangular solve
   ```

5. If A is Hessenberg, but not sparse, it is reduced to an upper triangular matrix and that system is solved via substitution.
6. If A is square and does not satisfy criteria 1 through 5, then a general triangular factorization is computed by Gaussian elimination with partial pivoting (see `lu`). This results in

   ```
   A = L*U
   ```

   where `L` is a permutation of a lower triangular matrix and `U` is an upper triangular matrix. Then `X` is computed by solving two permuted triangular systems.

   ```
   X = U\(L\B)
   ```

   If `A` is sparse, then UMFPACK is used to compute `X`. The computations result in

   ```
   P*A*Q = L*U
   ```

   where `P` is a row permutation matrix and `Q` is a column reordering matrix. Then `X = Q*(U\L\(P*B))`.

7. If A is not square, then Householder reflections are used to compute an orthogonal-triangular factorization.

   ```
   A*P = Q*R
   ```

   where `P` is a permutation, `Q` is orthogonal and `R` is upper triangular (see `qr`). The least squares solution `X` is computed with

   ```
   X = P*(R\(Q'*B))
   ```

   If `A` is sparse, then MATLAB computes a least squares solution using the sparse qr factorization of `A`.

Note    For sparse matrices, to see information about choice of algorithm and storage allocation, set the spparms parameter `'spumoni' = 1`.

| Note | Backslash is not implemented for sparse matrices `A` that are complex but not square. |
|---|---|

MATLAB uses LAPACK routines to compute these matrix factorizations:

| Matrix | Real | Complex |
|---|---|---|
| Sparse square banded with band density > `'bandden'` | `DGBTRF, DGBTRS` | `ZGBTRF, ZGBTRS` |
| Full square, symmetric (Hermitian) positive definite | `DLANGE, DPOTRF, DPOTRS, DPOCON` | `ZLANGE, ZPOTRF, ZPOTRS ZPOCON` |
| Full square, general case | `DLANGE, DGESV, DGECON` | `ZLANGE, ZGESV, ZGECON` |
| Full nonsquare | `DGEQP3, DORMQR, DTRTRS` | `ZGEQP3, ZORMQR, ZTRTRS` |
| For other cases (sparse, triangular, and Hessenberg) MATLAB does not use LAPACK. | | |

## Diagnostics

- From matrix division, if a square `A` is singular,
  ```
  Warning: Matrix is singular to working precision.
  ```
- From elementwise division, if the divisor has zero elements,
  ```
  Warning: Divide by zero.
  ```
  Matrix division and elementwise division can produce `NaN`s or `Inf`s where appropriate.
- If the inverse was found, but is not reliable,
  ```
  Warning: Matrix is close to singular or badly scaled.
      Results may be inaccurate.  RCOND = xxx
  ```
- From matrix division, if a nonsquare `A` is rank deficient,
  ```
  Warning: Rank deficient, rank = xxx tol = xxx
  ```

## See Also

chol, det, inv, lu, orth, permute, ipermute, qr, rref

## References

[1] Anderson, E., Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, LAPACK User's Guide (http://www.netlib.org/lapack/lug/lapack_lug.html), Third Edition, SIAM, Philadelphia, 1999.

[2] Davis, T.A., UMFPACK Version 4.0 User Guide (http://www.cise.ufl.edu/research/sparse/umfpack/v4.0/UserGuide.pdf), Dept. of Computer and Information Science and Engineering, Univ. of Florida, Gainesville, FL, 2002.