

Matrices and Linear Algebra

Matrices and Linear Algebra	4-2
Matrices in MATLAB	4-4
Addition and Subtraction	4-6
Vector Products and Transpose	4-7
Matrix Multiplication	4-8
The Identity Matrix	4-10
Vector and Matrix Norms	4-12
Solving Linear Equations	4-13
Square Systems	4-14
Overdetermined Systems	4-15
Undetermined Systems	4-17
Inverses and Determinants	4-20
Pseudoinverses	4-21
LU, QR, and Cholesky Factorizations	4-24
Cholesky Factorization	4-24
LU Factorization	4-25
QR Factorization	4-27
Matrix Powers and Exponentials	4-31
Eigenvalues	4-34
Singular Value Decomposition	4-38

Matrices and Linear Algebra

A *matrix* is a two-dimensional array of real or complex numbers. *Linear algebra* defines many matrix operations that are directly supported by MATLAB. Matrix arithmetic, linear equations, eigenvalues, singular values, and matrix factorizations are included.

The linear algebra functions are located in the `matfun` directory in the MATLAB Toolbox.

Category	Function	Description
Matrix analysis	<code>norm</code>	Matrix or vector norm.
	<code>normest</code>	Estimate the matrix 2-norm.
	<code>rank</code>	Matrix rank.
	<code>det</code>	Determinant.
	<code>trace</code>	Sum of diagonal elements.
	<code>null</code>	Null space.
	<code>orth</code>	Orthogonalization.
	<code>rref</code>	Reduced row echelon form.
	<code>subspace</code>	Angle between two subspaces.
Linear equations	<code>\</code> and <code>/</code>	Linear equation solution.
	<code>inv</code>	Matrix inverse.
	<code>cond</code>	Condition number for inversion.
	<code>condest</code>	1-norm condition number estimate.
	<code>chol</code>	Cholesky factorization.
	<code>cholinc</code>	Incomplete Cholesky factorization.
	<code>lu</code>	LU factorization.
	<code>luinc</code>	Incomplete LU factorization.

Category	Function	Description
Eigenvalues and singular values	qr	Orthogonal-triangular decomposition.
	nls	Nonnegative least-squares.
	pinv	Pseudoinverse.
	lsqov	Least squares with known covariance.
	eig	Eigenvalues and eigenvectors.
	svd	Singular value decomposition.
	eigs	A few eigenvalues.
	svds	A few singular values.
	poly	Characteristic polynomial.
	polyeig	Polynomial eigenvalue problem.
	condeig	Condition number for eigenvalues.
	hess	Hessenberg form.
	qz	QZ factorization.
Matrix functions	schur	Schur decomposition.
	expm	Matrix exponential.
	logm	Matrix logarithm.
	sqrtn	Matrix square root.
funm	Evaluate general matrix function.	

Matrices in MATLAB

Informally, the terms matrix and array are often used interchangeably. More precisely, a matrix is a two-dimensional rectangular array of real or complex numbers that represents a linear transformation. The linear algebraic operations defined on matrices have found applications in a wide variety of technical fields. (The Symbolic Math Toolboxes extend MATLAB's capabilities to operations on various types of nonnumeric matrices.)

MATLAB has dozens of functions that create different kinds of matrices. Two of them can be used to create a pair of 3-by-3 example matrices for use throughout this chapter. The first example is symmetric.

```
A = pascal (3)
```

```
A =
```

```
1 1 1
1 2 3
1 3 6
```

The second example is not symmetric.

```
B = magic(3)
```

```
B =
```

```
8 1 6
3 5 7
4 9 2
```

Another example is a 3-by-2 rectangular matrix of random integers.

```
C = fix(10*rand(3,2))
```

```
C =
```

```
9 4
2 8
6 7
```

A *column vector* is an m -by-1 matrix, a *row vector* is a 1-by- n matrix and a *scalar* is a 1-by-1 matrix. The statements

```
u = [3; 1; 4]
```

```
v = [2 0 -1]
```

```
s = 7
```

produce a column vector, a row vector, and a scalar.

```
u =
```

```
3  
1  
4
```

```
v =
```

```
2    0   -1
```

```
s =
```

```
7
```

Addition and Subtraction

Addition and subtraction of matrices is defined just as it is for arrays, element-by-element. Adding A to B and then subtracting A from the result recovers B.

$$X = A + B$$

$$X =$$

$$\begin{array}{ccc} 9 & 2 & 7 \\ 4 & 7 & 10 \\ 5 & 12 & 8 \end{array}$$

$$Y = X - A$$

$$Y =$$

$$\begin{array}{ccc} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{array}$$

Addition and subtraction require both matrices to have the same dimension, or one of them be a scalar. If the dimensions are incompatible, an error results.

$$X = A + C$$

Error using ==> +
Matrix dimensions must agree.

$$w = v + s$$

$$w =$$

$$\begin{array}{ccc} 9 & 7 & 6 \end{array}$$

Vector Products and Transpose

A row vector and a column vector of the same length can be multiplied in either order. The result is either a scalar, the *inner* product, or a matrix, the *outer* product.

$$x = v * u$$

$$x =$$

2

$$X = u * v$$

$$X =$$

6	0	-3
2	0	-1
8	0	-4

For real matrices, the *transpose* operation interchanges a_{ij} and a_{ji} . MATLAB uses the apostrophe (or single quote) to denote transpose. Our example matrix A is *symmetric*, so A' is equal to A. But B is not symmetric.

$$X = B'$$

$$X =$$

8	3	4
1	5	9
6	7	2

Transposition turns a row vector into a column vector.

$$x = v'$$

$$x =$$

2
0
-1

If x and y are both real column vectors, the product $x*y$ is not defined, but the two products

$$x' * y$$

and

$$y' * x$$

are the same scalar. This quantity is used so frequently, it has three different names: *inner product*, *scalar product*, or *dot product*.

For a complex vector or matrix, z , the quantity z' denotes the *complex conjugate transpose*. The unconjugated complex transpose is denoted by $z.'$, in analogy with the other array operations. So if

$$z = [1+2i \quad 3+4i]$$

then z' is

$$\begin{bmatrix} 1-2i \\ 3-4i \end{bmatrix}$$

while $z.'$ is

$$\begin{bmatrix} 1+2i \\ 3+4i \end{bmatrix}$$

For complex vectors, the two scalar products $x' * y$ and $y' * x$ are complex conjugates of each other and the scalar product $x' * x$ of a complex vector with itself is real.

Matrix Multiplication

Multiplication of matrices is defined in a way that reflects composition of the underlying linear transformations and allows compact representation of systems of simultaneous linear equations. The matrix product $C = AB$ is defined when the column dimension of A is equal to the row dimension of B , or when one of them is a scalar. If A is m -by- p and B is p -by- n , their product C is m -by- n . The product can actually be defined using MATLAB's for loops, colon notation, and vector dot products.

```

for i = 1:m
    for j = 1:n
        C(i,j) = A(i,:)*B(:,j);
    end
end

```

MATLAB uses a single asterisk to denote matrix multiplication. The next two examples illustrate the fact that matrix multiplication is not commutative; AB is usually not equal to BA .

$X = A*B$

$X =$

15	15	15
26	38	26
41	70	39

$Y = B*A$

$Y =$

15	28	47
15	34	60
15	28	43

A matrix can be multiplied on the right by a column vector and on the left by a row vector.

$x = A*u$

$x =$

8
17
30

$y = v*B$

$y =$

12	-7	10
----	----	----

Rectangular matrix multiplications must satisfy the dimension compatibility conditions.

$$X = A * C$$

$$X =$$

$$\begin{array}{cc} 17 & 19 \\ 31 & 41 \\ 51 & 70 \end{array}$$

$$Y = C * A$$

Error using ==> *
Inner matrix dimensions must agree.

Anything can be multiplied by a scalar.

$$w = s * v$$

$$w =$$

$$\begin{array}{ccc} 14 & 0 & -7 \end{array}$$

The Identity Matrix

Generally accepted mathematical notation uses the capital letter I to denote *identity* matrices, matrices of various sizes with ones on the main diagonal and zeros elsewhere. These matrices have the property that $AI = A$ and $IA = A$ whenever the dimensions are compatible. The original version of MATLAB could not use I for this purpose because it did not distinguish between upper and lowercase letters and i already served double duty as a subscript and as the complex unit. So an English language pun was introduced. The function

`eye(m, n)`

returns an m -by- n rectangular identity matrix and `eye(n)` returns an n -by- n square identity matrix.

The Kronecker Tensor Product

The Kronecker product, $\text{kron}(X, Y)$, of two matrices is the larger matrix formed from all possible products of the elements of X with those of Y . If X is m -by- n and Y is p -by- q , then $\text{kron}(X, Y)$ is mp -by- nq . The elements are arranged in the order

$$\begin{bmatrix} X(1, 1) * Y & X(1, 2) * Y & \dots & X(1, n) * Y \\ & & \ddots & \\ X(m, 1) * Y & X(m, 2) * Y & \dots & X(m, n) * Y \end{bmatrix}$$

The Kronecker product is often used with matrices of zeros and ones to build up repeated copies of small matrices. For example, if X is the 2-by-2 matrix

$X =$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

and $I = \text{eye}(2, 2)$ is the 2-by-2 identity matrix, then the two matrices

$$\text{kron}(X, I)$$

and

$$\text{kron}(I, X)$$

are

$$\begin{bmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 3 & 0 & 4 & 0 \\ 0 & 3 & 0 & 4 \end{bmatrix}$$

and

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 3 & 4 \end{bmatrix}$$

Vector and Matrix Norms

The p -norm of a vector x

$$\|x\|_p = \left(\sum |x_i|^p \right)^{1/p}$$

is computed by `norm(x, p)`. This is defined by any value of $p > 1$, but the most common values of p are 1, 2, and ∞ . The default value is $p = 2$, which corresponds to *Euclidean length*.

```
[ norm(v, 1) norm(v) norm(v, inf) ]
```

```
ans =
```

```
3.0000    2.2361    2.0000
```

The p -norm of a matrix A ,

$$\|A\|_p = \max_x \|Ax\|_p / \|x\|_p$$

can be computed for $p = 1, 2$, and ∞ by `norm(A, p)`. Again, the default value is $p = 2$.

```
[ norm(C, 1) norm(C) norm(C, inf) ]
```

```
ans =
```

```
19.0000    14.8015    13.0000
```

Solving Linear Equations

One of the most important problems in technical computing is the solution of simultaneous linear equations. In matrix notation, this problem can be stated as follows:

Given two matrices A and B , does there exist a unique matrix X so that $AX = B$ or $XA = B$?

It is instructive to consider a 1-by-1 example.

Does the equation

$$7x = 21$$

have a unique solution ?

The answer, of course, is yes. The equation has the unique solution $x = 3$. The solution is easily obtained by *division*:

$$x = 21/7 = 3$$

The solution is *not* ordinarily obtained by computing the inverse of 7, that is $7^{-1} = 0.142857\dots$, and then multiplying 7^{-1} by 21. This would be more work and, if 7^{-1} is represented to a finite number of digits, less accurate. Similar considerations apply to sets of linear equations with more than one unknown; MATLAB solves such equations without computing the inverse of the matrix.

Although it is not standard mathematical notation, MATLAB uses the division terminology familiar in the scalar case to describe the solution of a general system of simultaneous equations. The two division symbols, *slash*, $/$, and *backslash*, \backslash , are used for the two situations where the unknown matrix appears on the left or right of the coefficient matrix.

$X = A \backslash B$ denotes the solution to the matrix equation $AX = B$.

$X = B / A$ denotes the solution to the matrix equation $XA = B$.

You can think of “dividing” both sides of the equation $AX = B$ or $XA = B$ by A . The coefficient matrix A is always in the “denominator”.

The dimension compatibility conditions for $X = A \backslash B$ require the two matrices A and B to have the same number of rows. The solution X then has the same number of columns as B and its row dimension is equal to the column dimension of A . For $X = B / A$, the roles of rows and columns are interchanged.

In practice, linear equations of the form $AX = B$ occur more frequently than those of the form $XA = B$. Consequently, backslash is used far more frequently than slash. The remainder of this section concentrates on the backslash operator; the corresponding properties of the slash operator can be inferred from the identity

$$(B/A)' = (A' \setminus B')$$

The coefficient matrix A need not be square. If A is m -by- n , there are three cases.

$m = n.$	Square system. Seek an exact solution.
$m > n.$	Overdetermined system. Find a least squares solution.
$m < n.$	Underdetermined system. Find a basic solution with at most m nonzero components.

The backslash operator employs different algorithms to handle different kinds of coefficient matrices. The various cases, which are diagnosed automatically by examining the coefficient matrix, include:

- Permutations of triangular matrices
- Symmetric, positive definite matrices
- Square, nonsingular matrices
- Rectangular, overdetermined systems
- Rectangular, underdetermined systems

Square Systems

The most common situation involves a square coefficient matrix A and a single right-hand side column vector b . The solution, $x = A \setminus b$, is then the same size as b . For example

$$x = A \setminus u$$

$$x =$$

$$\begin{array}{c} 10 \\ -12 \\ 5 \end{array}$$

It can be confirmed that $A*x$ is exactly equal to u .

If A and B are square and the same size, then $X = A \setminus B$ is also that size.

$$X = A \setminus B$$

$$X =$$

$$\begin{array}{ccc} 19 & -3 & -1 \\ -17 & 4 & 13 \\ 6 & 0 & -6 \end{array}$$

It can be confirmed that $A*X$ is exactly equal to B .

Both of these examples have exact, integer solutions. This is because the coefficient matrix was chosen to be pascal (3), which has a determinant equal to one. A later section considers the effects of roundoff error inherent in more realistic computation.

A square matrix A is *singular* if it does not have linearly independent columns. If A is singular, the solution to $AX = B$ either does not exist, or is not unique. The backslash operator, $A \setminus B$, issues a warning if A is nearly singular and raises an error condition if exact singularity is detected.

Overdetermined Systems

Overdetermined systems of simultaneous linear equations are often encountered in various kinds of curve fitting to experimental data. Here is a hypothetical example. A quantity y is measured at several different values of time, t , to produce the following observations:

t	y
0.0	0.82
0.3	0.72
0.8	0.63
1.1	0.60
1.6	0.55
2.3	0.50

This data can be entered into MATLAB with the statements

```
t = [0 .3 .8 1.1 1.6 2.3]';
y = [.82 .72 .63 .60 .55 .50]';
```

It is believed that the data can be modeled with a decaying exponential function.

$$y(t) \approx c_1 + c_2 e^{-t}$$

This equation says that the vector y should be approximated by a linear combination of two other vectors, one the constant vector containing all ones and the other the vector with components e^{-t} . The unknown coefficients, c_1 and c_2 , can be computed by doing a *least squares fit*, which minimizes the sum of the squares of the deviations of the data from the model. There are six equations in two unknowns, represented by the 6-by-2 matrix.

$$E = [\text{ones}(\text{size}(t)) \quad \exp(-t)]$$

E =

```

1.0000    1.0000
1.0000    0.7408
1.0000    0.4493
1.0000    0.3329
1.0000    0.2019
1.0000    0.1003
    
```

The least squares solution is found with the backslash operator.

$$c = E \backslash y$$

c =

```

0.4760
0.3413
    
```

In other words, the least squares fit to the data is

$$y(t) \approx 0.4760 + 0.3413 e^{-t}$$

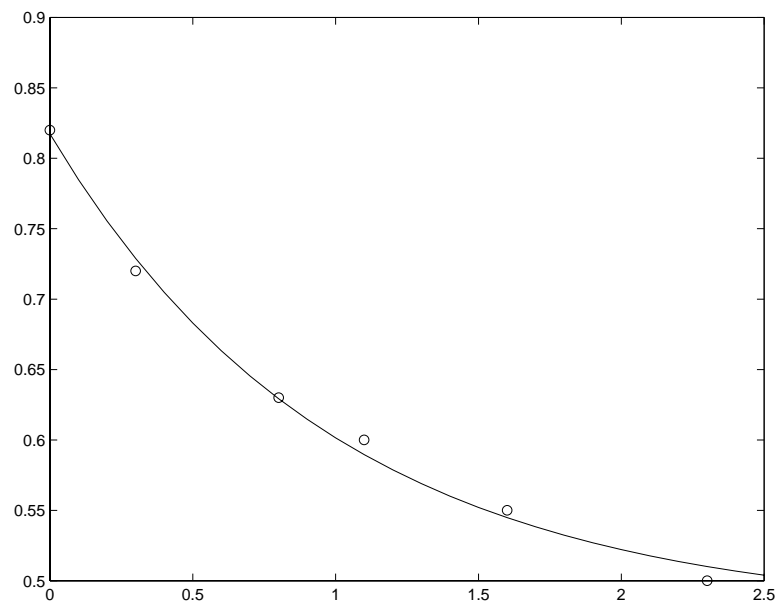
The following statements evaluate the model at regularly spaced increments in t , and then plot the result, together with the original data.

```

T = (0:0.1:2.5)';
Y = [ones(size(T)) exp(-T)]*c;
plot(T, Y, '- ', t, y, 'o')
    
```

You can see that $E*c$ is not exactly equal to y , but that the difference might well be less than measurement errors in the original data.

A rectangular matrix A is *rank deficient* if it does not have linearly independent columns. If A is rank deficient, the least squares solution to $AX = B$ is not unique. The backslash operator, $A \setminus B$, issues a warning if A is rank deficient and produces a *basic* solution that has as few nonzero elements as possible.



Undetermined Systems

Underdetermined linear systems involve more unknowns than equations. When they are accompanied by additional constraints, they are the purview of *linear programming*. By itself, the backslash operator deals only with the unconstrained system. The solution is never unique. MATLAB finds a *basic* solution, which has at most m nonzero components, but even this may not be unique. The particular solution actually computed is determined by the QR factorization with column pivoting (see a later section on the QR factorization).

Here is a small, random example.

```
R = fix(10*rand(2, 4))
```

```
R =
```

```
     6     8     7     3
     3     5     4     1
```

```
b = fix(10*rand(2, 1))
```

```
b =
```

```
     1
     2
```

The linear system $Rx = b$ involves two equations in four unknowns. Since the coefficient matrix contains small integers, it is appropriate to display the solution in *rational* format. The particular solution is obtained with

```
format rat
```

```
p = R\b
```

```
p =
```

```
     0
     5/7
     0
    -11/7
```

One of the nonzero components is $p(2)$ because $R(:, 2)$ is the column of R with largest norm. The other nonzero component is $p(4)$ because $R(:, 4)$ dominates after $R(:, 2)$ is eliminated.

The complete solution to the overdetermined system can be characterized by adding an arbitrary vector from the null space, which can be found using the `nul l` function with an option requesting a “rational” basis.

$$Z = \text{nul l}(R, 'r')$$

$$Z =$$

$$\begin{array}{cc} -1/2 & -7/6 \\ -1/2 & 1/2 \\ 1 & 0 \\ 0 & 1 \end{array}$$

It can be confirmed that $A*Z$ is zero and that any vector of the form

$$x = p + Z*q$$

for an arbitrary vector q satisfies $R*x = b$.

Inverses and Determinants

If A is square and nonsingular, the equations $AX = I$ and $XA = I$ have the same solution, X . This solution is called the *inverse* of A , is denoted by A^{-1} , and is computed by the function `inv`. The determinant of a matrix is useful in theoretical considerations and some types of symbolic computation, but its scaling and roundoff error properties make it far less satisfactory for numeric computation. Nevertheless, the function `det` computes the determinant of a square matrix.

```
d = det(A)
```

```
X = inv(A)
```

```
d =
```

```
1
```

```
X =
```

```
3   -3   1
-3   5  -2
1   -2   1
```

Again, because A is symmetric, has integer elements, and has determinant equal to one, so does its inverse. On the other hand,

```
d = det(B)
```

```
X = inv(B)
```

```
d =
```

```
-360
```

```
X =
```

```
0.1472  -0.1444  0.0639
-0.0611  0.0222  0.1056
-0.0194  0.1889 -0.1028
```

Closer examination of the elements of X , or use of `format rat`, would reveal that they are integers divided by 360.

If A is square and nonsingular, then without roundoff error, $X = \text{inv}(A) * B$ would theoretically be the same as $X = A \setminus B$ and $Y = B * \text{inv}(A)$ would theoretically be the same as $Y = B / A$. But the computations involving the backslash and slash operators are preferable because they require less computer time, less memory, and have better error detection properties.

Pseudoinverses

Rectangular matrices do not have inverses or determinants. At least one of the equations $AX = I$ and $XA = I$ does not have a solution. A partial replacement for the inverse is provided by the *Moore-Penrose pseudoinverse*, which is computed by the `pinv` function.

$$X = \text{pinv}(C)$$

$$X =$$

$$\begin{array}{ccc} 0.1159 & -0.0729 & 0.0171 \\ -0.0534 & 0.1152 & 0.0418 \end{array}$$

The matrix

$$Q = X * C$$

$$Q =$$

$$\begin{array}{cc} 1.0000 & 0.0000 \\ 0.0000 & 1.0000 \end{array}$$

is the 2-by-2 identity, but the matrix

$$P = C * X$$

$$P =$$

$$\begin{array}{ccc} 0.8293 & -0.1958 & 0.3213 \\ -0.1958 & 0.7754 & 0.3685 \\ 0.3213 & 0.3685 & 0.3952 \end{array}$$

is not the 3-by-3 identity. However, P acts like an identity on a portion of the space in the sense that P is symmetric, $P * C$ is equal to C and $X * P$ is equal to X .

If A is m -by- n with $m > n$ and full rank n , then each of the three statements

$$\begin{aligned} x &= A \backslash b \\ x &= \text{pinv}(A) * b \\ x &= \text{inv}(A' * A) * A' * b \end{aligned}$$

theoretically computes the same least squares solution x , although the backslash operator does it faster.

However, if A does not have full rank, the solution to the least squares problem is not unique. There are many vectors x that minimize

$$\text{norm}(A * x - b)$$

The solution computed by $x = A \backslash b$ is a *basic* solution; it has at most r nonzero components, where r is the rank of A . The solution computed by $x = \text{pinv}(A) * b$ is the *minimal norm* solution; it also minimizes $\text{norm}(x)$. An attempt to compute a solution with $x = \text{inv}(A' * A) * A' * b$ fails because $A' * A$ is singular.

Here is an example to illustrate the various solutions.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

does not have full rank. Its second column is the average of the first and third columns. If

$$b = A(:, 2)$$

is the second column, then an obvious solution to $A * x = b$ is $x = [0 \ 1 \ 0]'$. But none of the approaches computes that x . The backslash operator gives

$$x = A \backslash b$$

Warning: Rank deficient, rank = 2.

$$x =$$

$$\begin{bmatrix} 0.5000 \\ 0 \\ 0.5000 \end{bmatrix}$$

This solution has two nonzero components. The pseudoinverse approach gives

$$y = \text{pinv}(A) * b$$

$$y =$$

0.3333

0.3333

0.3333

There is no warning about rank deficiency. But $\text{norm}(y) = 0.5774$ is less than $\text{norm}(x) = 0.7071$. Finally

$$z = \text{inv}(A' * A) * A' * b$$

fails completely.

Warning: Matrix is singular to working precision.

$$z =$$

Inf

Inf

Inf

LU, QR, and Cholesky Factorizations

MATLAB's linear equation capabilities are based on three basic matrix factorizations.

- Cholesky factorization for symmetric, positive definite matrices
- Gaussian elimination for general square matrices
- Orthogonalization for rectangular matrices

These three factorizations are available through the `chol`, `lu`, and `qr` functions.

All three of these factorizations make use of *triangular* matrices where all the elements either above or below the diagonal are zero. Systems of linear equations involving triangular matrices are easily and quickly solved using either *forward* or *back substitution*.

Cholesky Factorization

The Cholesky factorization expresses a symmetric matrix as the product of a triangular matrix and its transpose.

$$A = R'R$$

where R is an upper triangular matrix.

Not all symmetric matrices can be factored in this way; the matrices that have such a factorization are said to be *positive definite*. This implies that all the diagonal elements of A are positive and that the offdiagonal elements are "not too big." The Pascal matrices provide an interesting example. Throughout this chapter, our example matrix A has been the 3-by-3 Pascal matrix. Let's temporarily switch to the 6-by-6.

$$A = \text{pascal}(6)$$

$$A =$$

1	1	1	1	1	1
1	2	3	4	5	6
1	3	6	10	15	21
1	4	10	20	35	56
1	5	15	35	70	126
1	6	21	56	126	252

The elements of A are binomial coefficients. Each element is the sum of its north and west neighbors. The Cholesky factorization is

$$R = \text{chol}(A)$$

$$R =$$

$$\begin{array}{cccccc} 1 & & & & & \\ 0 & 1 & & & & \\ 0 & 0 & 1 & & & \\ 0 & 0 & 0 & 1 & & \\ 0 & 0 & 0 & 0 & 1 & \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}$$

The elements are again binomial coefficients. The fact that $R' * R$ is equal to A demonstrates an identity involving sums of products of binomial coefficients.

The Cholesky factorization also applies to complex matrices. Any complex matrix which has a Cholesky factorization satisfies $A' = A$ and is said to be *Hermitian positive definite*.

The Cholesky factorization allows the linear system

$$A * x = b$$

to be replaced by

$$R' * R * x = b$$

Because the backslash operator recognizes triangular systems, this can be solved quickly with

$$x = R \setminus (R' \setminus b)$$

If A is n -by- n , the computational complexity of $\text{chol}(A)$ is $O(n^3)$, but the complexity of the subsequent backslash solutions is only $O(n^2)$.

LU Factorization

Gaussian elimination, or LU factorization, expresses any square matrix as the product of a permutation of a lower triangular matrix and an upper triangular matrix

$$A = L U$$

where L is a permutation of a lower triangular matrix with ones on its diagonal and U is an upper triangular matrix.

The permutations are necessary for both theoretical and computational reasons. The matrix

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

cannot be expressed as the product of triangular matrices without interchanging its two rows. Although the matrix

$$\begin{bmatrix} \varepsilon & 1 \\ 1 & 0 \end{bmatrix}$$

can be expressed as the product of triangular matrices, when ε is small the elements in the factors are large and magnify errors, so even though the permutations are not strictly necessary, they are desirable. *Partial pivoting* ensures that the elements of L are bounded by one in magnitude and that the elements of U are not much larger than those of A .

For example

$$[L, U] = \text{l u}(B)$$

$L =$

$$\begin{array}{ccc} 1.0000 & 0 & 0 \\ 0.3750 & 0.5441 & 1.0000 \\ 0.5000 & 1.0000 & 0 \end{array}$$

$U =$

$$\begin{array}{ccc} 8.0000 & 1.0000 & 6.0000 \\ 0 & 8.5000 & -1.0000 \\ 0 & 0 & 5.2941 \end{array}$$

The LU factorization of A allows the linear system

$$A\mathbf{x} = \mathbf{b}$$

to be solved quickly with

$$\mathbf{x} = U \setminus (L \setminus \mathbf{b})$$

Determinants and inverses are computed from the LU factorization using

$$\det(A) = \det(L) * \det(U) = \pm \prod(\text{diag}(U))$$

and

$$\text{inv}(A) = \text{inv}(U) * \text{inv}(L)$$

QR Factorization

An *orthogonal* matrix, or a matrix with *orthonormal columns*, is a real matrix whose columns all have unit length and are perpendicular to each other. If Q is orthogonal, then

$$Q' Q = I$$

The simplest orthogonal matrices are two-dimensional coordinate rotations.

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

For complex matrices, the corresponding term is *unitary*. Orthogonal and unitary matrices are desirable for numerical computation because they preserve length, preserve angles, and do not magnify errors.

The orthogonal, or QR, factorization expresses any rectangular matrix as the product of an orthogonal or unitary matrix and an upper triangular matrix. A column permutation may also be involved.

$$A = Q R$$

or

$$A P = Q R$$

where Q is orthogonal or unitary, R is upper triangular, and P is a permutation.

There are four variants of the QR factorization— full or economy size and with or without column permutation.

Overdetermined linear systems involve a rectangular matrix with more rows than columns, that is m -by- n with $m > n$. The *full* size QR factorization produces a square, m -by- m orthogonal Q and a rectangular m -by- n upper triangular R .

$$[Q, R] = \text{qr}(C)$$

$$Q =$$

$$\begin{array}{ccc} -0.8182 & 0.3999 & -0.4131 \\ -0.1818 & -0.8616 & -0.4739 \\ -0.5455 & -0.3126 & 0.7777 \end{array}$$

$$R =$$

$$\begin{array}{cc} -11.0000 & -8.5455 \\ 0 & -7.4817 \\ 0 & 0 \end{array}$$

In many cases, the last $m - n$ columns of Q are not needed because they are multiplied by the zeros in the bottom portion of R . So the *economy* size QR factorization produces a rectangular, m -by- n Q with orthonormal columns and a square n -by- n upper triangular R . For our 3-by-2 example, this is not much of a saving, but for larger, highly rectangular matrices, the savings in both time and memory can be quite important.

$$[Q, R] = \text{qr}(C, 0)$$

$$Q =$$

$$\begin{array}{cc} -0.8182 & 0.3999 \\ -0.1818 & -0.8616 \\ -0.5455 & -0.3126 \end{array}$$

$$R =$$

$$\begin{array}{cc} -11.0000 & -8.5455 \\ 0 & -7.4817 \end{array}$$

In contrast to the LU factorization, the QR factorization does not require any pivoting or permutations. But an optional column permutation, triggered by the presence of a third output argument, is useful for detecting singularity or rank deficiency. At each step of the factorization, the column of the remaining unfactored matrix with largest norm is used as the basis for that step. This ensures that the diagonal elements of R occur in decreasing order and that any linear dependence among the columns will almost certainly be revealed by examining these elements. For our small example, the second column of C has a larger norm than the first, so the two columns are exchanged.

$$[Q, R, P] = \text{qr}(C)$$

$$Q =$$

$$\begin{array}{ccc} -0.3522 & 0.8398 & -0.4131 \\ -0.7044 & -0.5285 & -0.4739 \\ -0.6163 & 0.1241 & 0.7777 \end{array}$$

$$R =$$

$$\begin{array}{cc} -11.3578 & -8.2762 \\ 0 & 7.2460 \\ 0 & 0 \end{array}$$

$$P =$$

$$\begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array}$$

When the economy size and column permutations are combined, the third output argument is a permutation vector, rather than a permutation matrix.

$$[Q, R, p] = \text{qr}(C, 0)$$

$$Q =$$

$$\begin{bmatrix} -0.3522 & 0.8398 \\ -0.7044 & -0.5285 \\ -0.6163 & 0.1241 \end{bmatrix}$$

$$R =$$

$$\begin{bmatrix} -11.3578 & -8.2762 \\ 0 & 7.2460 \end{bmatrix}$$

$$p =$$

$$\begin{bmatrix} 2 & 1 \end{bmatrix}$$

The QR factorization transforms an overdetermined linear system into an equivalent triangular system. The expression

$$\text{norm}(A*x - b)$$

is equal to

$$\text{norm}(Q*R*x - b)$$

Multiplication by orthogonal matrices preserves the Euclidean norm, so this expression is also equal to

$$\text{norm}(R*x - y)$$

where $y = Q' * b$. Since the last $m - n$ rows of R are zero, this expression breaks into two pieces

$$\text{norm}(R(1:n, 1:n)*x - y(1:n))$$

and

$$\text{norm}(y(n+1:m))$$

When A has full rank, it is possible to solve for x so that the first of these expressions is zero. Then the second expression gives the norm of the residual. When A does not have full rank, the triangular structure of R makes it possible to find a basic solution to the least squares problem.

Matrix Powers and Exponentials

If A is a square matrix and p is a positive integer, then A^p multiplies A by itself p times.

$$X = A^2$$

$$X =$$

$$\begin{bmatrix} 3 & 6 & 10 \\ 6 & 14 & 25 \\ 10 & 25 & 46 \end{bmatrix}$$

If A is square and nonsingular, then $A^{(-p)}$ multiplies $\text{inv}(A)$ by itself p times.

$$Y = B^{(-3)}$$

$$Y =$$

$$\begin{bmatrix} 0.0053 & -0.0068 & 0.0018 \\ -0.0034 & 0.0001 & 0.0036 \\ -0.0016 & 0.0070 & -0.0051 \end{bmatrix}$$

Fractional powers, like $A^{(2/3)}$, are also permitted; the results depend upon the distribution of the eigenvalues of the matrix.

Element-by-element powers are obtained with `.^`. For example

$$X = A.^2$$

$$A =$$

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 9 \\ 1 & 9 & 36 \end{bmatrix}$$

The function

$$\text{sqrtm}(A)$$

computes $A^{(1/2)}$ by a more accurate algorithm. The `m` in `sqrtm` distinguishes this function from `sqrt(A)` which, like $A.^{(1/2)}$, does its job element-by-element.

A system of linear, constant coefficient, ordinary differential equations can be written

$$dx/dt = Ax$$

where $x = x(t)$ is a vector of functions of t and A is a matrix independent of t . The solution can be expressed in terms of the *matrix exponential*,

$$x(t) = e^{tA}x(0)$$

The function

`expm(A)`

computes the matrix exponential. An example is provided by the 3-by-3 coefficient matrix

`A =`

$$\begin{bmatrix} 0 & -6 & -1 \\ 6 & 2 & -16 \\ -5 & 20 & -10 \end{bmatrix}$$

and the initial condition, $x(0)$

`x0 =`

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

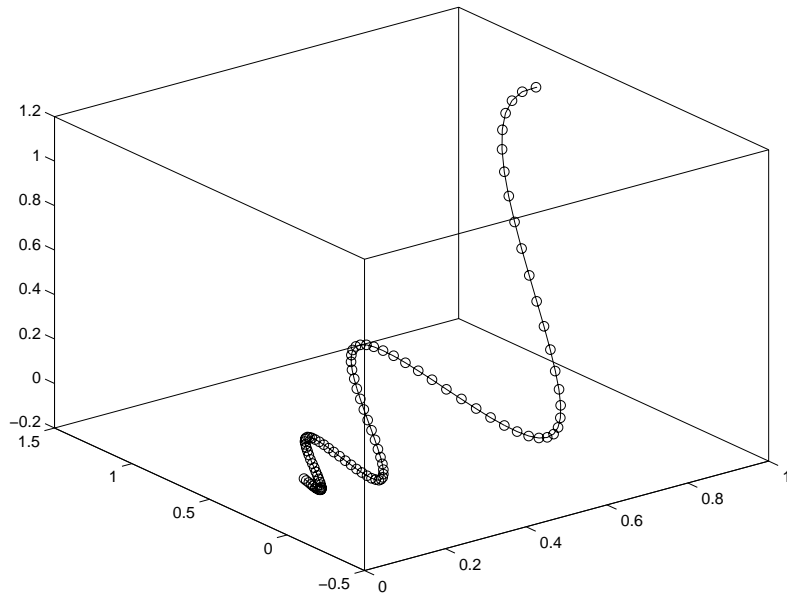
The matrix exponential is used to compute the solution, $x(t)$, to the differential equation at 101 points on the interval $0 \leq t \leq 1$ with

```
X = [];
for t = 0: .01: 1
    X = [X expm(t*A)*x0];
end
```

A three-dimensional phase plane plot obtained with

```
plot3(X(1,:), X(2,:), X(3,:), '-o')
```

shows the solution spiraling in towards the origin. This behavior is related to the eigenvalues of the coefficient matrix, which are discussed in the next section.



Eigenvalues

An *eigenvalue* and *eigenvector* of a square matrix A are a scalar λ and a vector v that satisfy

$$Av = \lambda v$$

With the eigenvalues on the diagonal of a diagonal matrix Λ and the corresponding eigenvectors forming the columns of a matrix V , we have

$$AV = V\Lambda$$

If V is nonsingular, this becomes the *eigenvalue decomposition*

$$A = V\Lambda V^{-1}$$

A good example is provided by the coefficient matrix of the ordinary differential equation in the previous section.

$$A =$$

$$\begin{array}{ccc} 0 & -6 & -1 \\ 6 & 2 & -16 \\ -5 & 20 & -10 \end{array}$$

The statement

$$\lambda = \text{eig}(A)$$

produces a column vector containing the eigenvalues. For this matrix, the eigenvalues are complex.

$$\lambda =$$

$$\begin{array}{l} -3.0710 \\ -2.4645 + 17.6008i \\ -2.4645 - 17.6008i \end{array}$$

The real part of each of the eigenvalues is negative, so $e^{\lambda t}$ approaches zero as t increases. The nonzero imaginary part of two of the eigenvalues, $\pm\omega$, contributes the oscillatory component, $\sin(\omega t)$, to the solution of the differential equation.

With two output arguments, `ei g` computes the eigenvectors and stores the eigenvalues in a diagonal matrix.

$$[V, D] = \text{ei g}(A)$$

$V =$

$$\begin{array}{ccc} -0.8326 & -0.1203 + 0.2123i & -0.1203 - 0.2123i \\ -0.3553 & 0.4691 + 0.4901i & 0.4691 - 0.4901i \\ -0.4248 & 0.6249 - 0.2997i & 0.6249 + 0.2997i \end{array}$$

$D =$

$$\begin{array}{ccc} -3.0710 & 0 & 0 \\ 0 & -2.4645 + 17.6008i & 0 \\ 0 & 0 & -2.4645 - 17.6008i \end{array}$$

The first eigenvector is real and the other two vectors are complex conjugates of each other. All three vectors are normalized to have Euclidean length, $\text{norm}(v, 2)$, equal to one.

The matrix $V * D * \text{inv}(V)$, which can be written more succinctly as $V * D / V$, is within roundoff error of A . And, $\text{inv}(V) * A * V$, or $V \backslash A * V$, is within roundoff error of D .

Some matrices do not have an eigenvector decomposition. These matrices are *defective*, or *not diagonalizable*. For example,

$A =$

$$\begin{array}{ccc} 6 & 12 & 19 \\ -9 & -20 & -33 \\ 4 & 9 & 15 \end{array}$$

For this matrix

$$[V, D] = \text{ei g}(A)$$

produces

V =

0. 4741	0. 4082	-0. 4082
-0. 8127	-0. 8165	0. 8165
0. 3386	0. 4082	-0. 4082

D =

-1. 0000	0	0
0	1. 0000	0
0	0	1. 0000

There is a double eigenvalue at $\lambda = 1$. The second and third columns of V are negatives of each other; they are merely different normalizations of the single eigenvector corresponding to $\lambda = 1$. For this matrix, a full set of linearly independent eigenvectors does not exist.

The optional Symbolic Math Toolbox extends MATLAB's capabilities by connecting to Maple, a powerful computer algebra system. One of the functions provided by the toolbox computes the Jordan Canonical Form. This is appropriate for matrices like our example, which is 3-by-3 and has exactly known, integer elements.

[X, J] = jordan(A)

X =

-1. 7500	1. 5000	2. 7500
3. 0000	-3. 0000	-3. 0000
-1. 2500	1. 5000	1. 2500

J =

-1	0	0
0	1	1
0	0	1

The Jordan Canonical Form is an important theoretical concept, but it is not a reliable computational tool for larger matrices, or for matrices whose elements are subject to roundoff errors and other uncertainties.

MATLAB's advanced matrix computations do not require eigenvalue decompositions. They are based, instead, on the *Schur decomposition*,

$$A = USU^T$$

where U is an orthogonal matrix and S is a block upper triangular matrix with 1-by-1 and 2-by-2 blocks on the diagonal. The eigenvalues are revealed by the diagonal elements and blocks of S , while the columns of U provide a basis with much better numerical properties than a set of eigenvectors. The Schur decomposition of our defective example is

$$[U, S] = \text{schur}(A)$$

$U =$

$$\begin{array}{ccc} 0.4741 & -0.6571 & 0.5861 \\ -0.8127 & -0.0706 & 0.5783 \\ 0.3386 & 0.7505 & 0.5675 \end{array}$$

$S =$

$$\begin{array}{ccc} -1.0000 & 21.3737 & 44.4161 \\ 0 & 1.0081 & 0.6095 \\ 0 & -0.0001 & 0.9919 \end{array}$$

The double eigenvalue is contained in the lower 2-by-2 block of S .

Singular Value Decomposition

A *singular value* and corresponding *singular vectors* of a rectangular matrix A are a scalar σ and a pair of vectors u and v that satisfy

$$\begin{aligned}Av &= \sigma u \\ A^T u &= \sigma v\end{aligned}$$

With the singular values on the diagonal of a diagonal matrix Σ and the corresponding singular vectors forming the columns of two orthogonal matrices U and V , we have

$$\begin{aligned}AV &= U\Sigma \\ A^T U &= V\Sigma\end{aligned}$$

Since U and V are orthogonal, this becomes the *singular value decomposition*

$$A = U\Sigma V^T$$

The full singular value decomposition of an m -by- n matrix involves an m -by- m U , an m -by- n Σ , and an n -by- n V . In other words, U and V are both square and Σ is the same size as A . If A has many more rows than columns, the resulting U can be quite large, but most of its columns are multiplied by zeros in Σ . In this situation, the *economysized* decomposition saves both time and storage by producing an m -by- n U , an n -by- n Σ and the same V .

The eigenvalue decomposition is the appropriate tool for analyzing a matrix when it represents a mapping from a vector space into itself, as it does for an ordinary differential equation. On the other hand, the singular value decomposition is the appropriate tool for analyzing a mapping from one vector space into another vector space, possibly with a different dimension. Most systems of simultaneous linear equations fall into this second category.

If A is square, symmetric, and positive definite, then its eigenvalue and singular value decompositions are the same. But, as A departs from symmetry and positive definiteness, the difference between the two decompositions increases. In particular, the singular value decomposition of a real matrix is always real, but the eigenvalue decomposition of a real, nonsymmetric matrix might be complex.

For the example matrix

A =

$$\begin{bmatrix} 9 & 4 \\ 6 & 8 \\ 2 & 7 \end{bmatrix}$$

the full singular value decomposition is

$$[U, S, V] = \text{svd}(A)$$

U =

$$\begin{bmatrix} 0.6105 & -0.7174 & 0.3355 \\ 0.6646 & 0.2336 & -0.7098 \\ 0.4308 & 0.6563 & 0.6194 \end{bmatrix}$$

S =

$$\begin{bmatrix} 14.9359 & & 0 \\ & 0 & 5.1883 \\ & 0 & 0 \end{bmatrix}$$

V =

$$\begin{bmatrix} 0.6925 & -0.7214 \\ 0.7214 & 0.6925 \end{bmatrix}$$

You can verify that U^*S^*V' is equal to A to within roundoff error. For this small problem, the economy size decomposition is only slightly smaller.

$$[U, S, V] = \text{svd}(A, 0)$$

$$U =$$

$$\begin{array}{cc} 0.6105 & -0.7174 \\ 0.6646 & 0.2336 \\ 0.4308 & 0.6563 \end{array}$$

$$S =$$

$$\begin{array}{cc} 14.9359 & 0 \\ 0 & 5.1883 \end{array}$$

$$V =$$

$$\begin{array}{cc} 0.6925 & -0.7214 \\ 0.7214 & 0.6925 \end{array}$$

Again, U^*S^*V' is equal to A to within roundoff error.