1 The Inverse of a Matrix and Matrix Pathology

- Division by a matrix is not defined but the equivalent is obtained from the *inverse* of the matrix.
- If the product of two square matrices, $A*B$, equals the identity matrix,I, B is said to be the inverse of A (and A is the inverse of B).
- Matrices do not commute on multiplication but inverses are an exception: $A * A^{-1} = A^{-1} * A$.
- To find the inverse of matrix A, use an elimination method. We augment the A matrix with the identity matrix of the same size and solve. The solution is A^{-1}.
- i.e.,

$$A = \left[\begin{array}{ccc} 1 & -1 & 2 \\ 3 & 0 & 1 \\ 1 & 0 & 2 \end{array}\right], \left[\begin{array}{cccccc} 1 & -1 & 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 & 1 \end{array}\right],$$

$$\begin{array}{r} R_2 - (3/1)R_1 \rightarrow \\ R_3 - (1/1)R_1 \rightarrow \end{array} \left[\begin{array}{cccccc} 1 & -1 & 2 & 1 & 0 & 0 \\ 0 & 3 & -5 & -3 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 \end{array}\right], \left[\begin{array}{cccccc} 1 & -1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 3 & -5 & -3 & 1 & 0 \end{array}\right],$$

$$R_3 - (3/1)R_2 \rightarrow \left[\begin{array}{cccccc} 1 & -1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & -5 & 0 & 1 & -3 \end{array}\right], \quad R_3/(-5) \quad ,$$

$$R_1 - (2/1)R_3 \rightarrow \quad , \left[\begin{array}{cccccc} 1 & -1 & 0 & 1 & 2/5 & -6/5 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1/5 & 3/5 \end{array}\right],$$

$$R_2 - (1/-1)R_1 \rightarrow \, , \left[\begin{array}{cccccc} 1 & 0 & 0 & 0 & 2/5 & -1/5 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1/5 & 3/5 \end{array}\right]$$

We confirm the fact that we have found the inverse by multiplication:

$$\left[\begin{array}{ccc} 1 & -1 & 2 \\ 3 & 0 & 1 \\ 1 & 0 & 2 \end{array}\right] \left[\begin{array}{ccc} 0 & 2/5 & -1/5 \\ -1 & 0 & 1 \\ 0 & -1/5 & 3/5 \end{array}\right] = \left[\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

It is more efficient to use Gaussian elimination. We show only the final triangular matrix; we used pivoting:

$$\begin{bmatrix} 1 & -1 & 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 2 & 0 & 0 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 3 & 0 & 1 & 0 & 1 & 0 \\ (0.333) & -1 & 1.667 & 1 & -0.333 & 0 \\ (0.333) & (0) & 1.667 & 0 & -0.333 & 1 \end{bmatrix}$$

After doing the back-substitutions, we get

$$\begin{bmatrix} 3 & 0 & 1 & 0 & 0.4 & -0.2 \\ (0.333) & -1 & 1.667 & -1 & 0 & 1 \\ (0.333) & (0) & 1.667 & 0 & -0.2 & 0.6 \end{bmatrix}$$

If we have the inverse of a matrix, we can use it to solve a set of equations, $Ax = b$, because multiplying by A^{-1} gives the answer:

$$\begin{aligned} A^{-1}Ax &= A^{-1}b \\ x &= A^{-1}b \end{aligned}$$

1.1 Pathological Systems

- When a real physical situation is modeled by a set of linear equations, we can anticipate that the set of equations will have a solution that matches the values of the quantities in the physical problem, at least as far as the equations truly do represent it.
- Because of round-off errors, the solution vector that is calculated may imperfectly predict the physical quantity, but there is assurance that a solution exists, at least in principle.
- Consequently, it must always be theoretically possible to avoid divisions by zero when the set of equations has a solution.
- Here is an example of a matrix that has no inverse: What is the LU equivalent of

$$A = \begin{bmatrix} 1 & -2 & 3 \\ 2 & 4 & -1 \\ -1 & -14 & 11 \end{bmatrix}$$

```
>> lu(A)
ans =
 2.0000      4.0000     -1.0000
-0.5000    -12.0000     10.5000
 0.5000      0.3333      0
```

Element A(3,3) cannot be used as a divisor in the back-substitution. That means that we cannot solve.

- The definition of a *singular matrix* is a matrix that does not have an inverse.

1.2 Redundant Systems

- Even though a matrix is singular, it may still have a solution. Consider again the same singular matrix:

$$A = \begin{bmatrix} 1 & -2 & 3 \\ 2 & 4 & -1 \\ -1 & -14 & 11 \end{bmatrix}$$

Suppose we solve the system $Ax = b$ where the right-hand side is $b = [5, 7, 1]^T$.

```
>> lu ( Ab )
ans =
 2.0000      4.0000     -1.0000      7.0000
-0.5000    -12.0000     10.5000 4.5000
 0.5000      0.3333      0 0
```

and the back-substitution cannot be done.

  - The output suggests that x_3 can have any value.
  - Suppose we set it equal to 0. We can solve the first two equations with that substitution, that gives $[17/4, -3/8, 0]^T$.
  - Suppose we set x_3 to 1 and repeat. This gives $[3, 1/2, 1]^T$, and this is another solution.
  - We have found a solution, actually, an infinity of them. The reason for this is that the system is redundant.

- What we have here is not truly three linear equations but only two independent ones. The system is called *redundant*.

- Here is a comparison of singular and nonsingular matrices:

For Singular matrix A:	For Nonsingular Matrix A:
It has no inverse, A^{-1}	It has an inverse, A^{-1}
Its determinant is zero	The determinant is nonzero
There is no unique solution	There is a unique solution
to the system $Ax = b$	to the system $Ax = b$
Gaussian elimination cannot avoid	Gaussian elimination does not
a zero on the diagonal	encounter a zero on the diagonal
The rank is less than n	The rank equals n
Rows are linearly dependent	Rows are linearly independent
Columns are linearly dependent	Columns are linearly independent

Table 1: A comparison of singular and nonsingular matrices

2 Ill-Conditioned Systems

- A system whose coefficient matrix is singular has no unique solution. What if the matrix is *almost* singular?

$$A = \begin{bmatrix} 3.02 & -1.05 & 2.53 \\ 4.33 & 0.56 & -1.78 \\ -0.83 & -0.54 & 1.47 \end{bmatrix}$$

The LU equivalent has a very small element in position (3, 3), and the inverse has elements very large in comparison to A:

$$LU = \begin{bmatrix} 4.33 & 0.56 & -1.78 \\ 0.6975 & -1.4406 & 3.7715 \\ -0.1917 & 0.3003 & -0.0039 \end{bmatrix},$$

$$inv(A) = \begin{bmatrix} 5.6611 & -7.2732 & -18.5503 \\ 200.5046 & -268.2570 & -66669.9143 \\ 76.8511 & -102.6500 & -255.8846 \end{bmatrix}$$

- Matrix is nonsingular but is *almost* singular
- Suppose we solve the system $Ax = b$, with b equal to $[-1.61, 7.23, -3.38]^T$. The solution is $x = [1.0000, 2.0000, -1.0000]^T$.
- Now suppose that we make a small change in just the first element of the b-vector : $[-1.60, 7.23, -3.38]^T$. We get $x = [1.0566, 4.0051, -0.2315]^T$
- $b = [-1.61, 7.22, -3.38]^T$. The solution now is $x = [1.07271, 4.6826, 0.0265]^T$ which also differs.

- A system whose coefficient matrix is nearly singular is called **ill-conditioned**. When a system is ill-conditioned, the solution is very sensitive to changes in the right-hand vector. It is also sensitive to small changes in the coefficients.
- $A(1,1)$ is changed from 3.02 to 3.00, original b-vector, a large change in the solution $x = [1.1277, 6.5221, 0.7333]^T]$. This means that it is also very sensitive to round-off error.

2.1 Norms

- **Norms**, a measure of the magnitude of the matrix.
- The magnitude of a single number is just its distance from zero: $|-4.2| = 4.2$.
- using $||A||$ to represent the norm of matrix A
  1. $||A|| \geq 0$ and $||A|| = 0$ if and only if $A = 0$
  2. $||kA|| = ||k||||A||$
  3. $||A + B|| \leq ||A|| + ||B||$
  4. $||AB|| \leq ||A||||B||$
- For vectors in two- or three space, norm is called the *Euclidean norm*, and is computed by $\sqrt{x_1^2 + x_2^2 + x_3^2}$. We compute the Euclidean norm of vectors with more than three components by

$$||x||_e = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2} = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}$$

- defining the p-norm as

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

$||x||_1 = \sum_{i=1}^{n} |x_i| = sum\ of\ magnitudes$
$||x||_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2} = Euclidean\ norm$
$||x||_\infty = max_{1 \leq i \leq n} |x_i| = maximum - magnitude\ norm$

- i.e., Compute the 1-, 2-, and ∞-norms of the vector $x = (1.25, 0.02, -5.15, 0)$

$||x||_1 = |1.25| + |0.02| + |-5.15| + |0| = 6.42$
$||x||_2 = 5.2996$
$||x||_\infty = 5.15$

2.1.1 Matrix Norms

- The norms of a matrix are similar to the norms of a vector.

$$||A||_1 = max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| = maximum\ column\ sum$$
$$||A||_\infty = max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}| = maximum\ row\ sum$$

- Suppose r is the largest eigenvalue of $A^T * A$. Then $||A||_2 = r^{1/2}$. This is called the *spectral norm* of A, and $||A||_2$ is always less than (or equal to) $||A||_1$ and $||A||_\infty$.

- For an $m \times n$ matrix, the Frobenius norm is defined as

$$||A||_f = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 \right)^{1/2}$$

  The Frobenius norm is a good measure of the magnitude of a matrix.

- The spectral norm is usually the most *expensive*. Which norm is best? In most instances, we want the norm that puts the smallest upper bound on the magnitude of the matrix. In this sense, the spectral norm is usually the "best".

```
A =
5 -5 -7
 -4  2 -4
 -7 -4  5
>> norm(A,'fro')
ans =
15
>> norm(A,inf)
ans =
17
>> norm(A,1)
ans=
16
>> norm(A)
ans =
12.0301
>> norm (A,2)
ans =
12.0301
```

we observe that the 2-norm, the spectral norm, is the norm we get if we just ask for the norm. The smallest norm of the matrix is the spectral norm, it is the tightest measure.

3 Iterative Methods

- Gaussian elimination and its variants are called *direct methods*. An entirely different way to solve many systems is through *iteration*. In this, we start with an initial estimate of the solution vector and proceed to refine this estimate.
- The two methods for solving $Ax = b$ that we shall discuss in this section are the *Jacobi Method* and the *Gauss-Seidel Method*.
- An $n \times n$ matrix A is *diagonally dominant* if and only if;

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, i = l, 2, \ldots, n$$

- Although this may seem like a very restrictive condition, it turns out that there are very many applied problems that have this property.
- i.e.,

$$\begin{aligned} 6x_1 - 2x_2 + x_3 &= 11 \\ x_l + 2x_2 - 5x_3 &= -1 \\ -2x_1 + 7x_2 + 2x_3 &= 5 \end{aligned}$$

- The solution is $x_1 = 2, x_2 = 1, x_3 = 1$. However, before we begin our iterative scheme we must first reorder the equations so that the coefficient matrix is diagonally dominant

$$\begin{aligned} 6x_1 - 2x_2 + x_3 &= 11 \\ -2x_1 + 7x_2 + 2x_3 &= 5 \\ x_l + 2x_2 - 5x_3 &= -1 \end{aligned}$$

3.1 Jacobi Method

The iterative methods depend on the rearrangement of the equations in this manner:

$$x_i = \frac{b_i}{a_{ii}} - \sum_{j=1, j\neq i}^{n} \frac{a_{ij}}{a_{ii}} x_j, i = l, 2, \ldots, n, \mapsto x_1 = \frac{11}{6} - \left(\frac{-2}{6}x_2 + \frac{1}{6}x_3\right)$$

	First	Second	Third	Fourth	Fifth	Sixth	...	Ninth
x_1	0	1.833	2.038	2.085	2.004	1.994	...	2.000
x_2	0	0.7l4	1.181	l.053	l.001	0.990	...	l.000
x_3	0	0.200	0.852	1.080	l.038	l.00l	...	1.000

Table 2: Successive estimates of solution (Jacobi method)

Each equation now solved for the variables in succession:

$$\begin{aligned} x_1 &= 1.8333 + 0.3333x_2 - 0.1667x_3 \\ x_2 &= 0.7143 + 0.2857x_1 - 0.2857x_3 \\ x_3 &= 0.2000 + 0.2000x_1 + 0.4000x_2 \end{aligned}$$

- We begin with some initial approximation to the value of the variables. (Each component might be taken equal to zero if no better initial estimates are at hand.)

- The new values are substituted in the right-hand sides to generate a second approximation, and the process is repeated until successive values of each of the variables are sufficiently alike.

$$\begin{aligned} x_1^{(n+1)} &= 1.8333 + 0.3333x_2^{(n)} - 0.l667x_3^{(n)} \\ x_2^{(n+1)} &= 0.7143 + 0.2857x_1^{(n)} - 0.2857x_3^{(n)} \\ x_3^{(n+1)} &= 0.2000 + 0.2000x_1^{(n)} + 0.4000x_2^{(n)} \end{aligned} \tag{1}$$

  Starting with an initial vector of $x^{(0)} = (0, 0, 0,)$, we get

- Note that this method is exactly the same as the method of fixed-point iteration for a single equation that was discussed in Section **??**, but it is now applied to a set of equations; we see this if we write Eq. 1 in the form of

$$x^{(n+1)} = G(x^{(n)}) = b' - Bx^n$$

  which is identical to $x_{n+1} = g(x_n)$ as used in Section **??**.

- In the present context, $x^{(n)}$ and $x^{(n+1)}$ refer to the nth and $(n+1)$st iterates of a vector rather than a simple variable, and g is a linear transformation rather than a nonlinear function.

$$Ax = b, \begin{bmatrix} 6 & -2 & 1 \\ -2 & 7 & 2 \\ 1 & 2 & -5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 5 \\ -1 \end{bmatrix}$$

Now, let $A = L + D + U$, where

$$L = \begin{bmatrix} 0 & 0 & 0 \\ -2 & 0 & 0 \\ 1 & 2 & 0 \end{bmatrix}, D = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & -5 \end{bmatrix}, U = \begin{bmatrix} 0 & -2 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}$$

rewritten as

$$\begin{aligned} Ax &= (L + D + U)x = b \\ Dx &= -(L + U)x + b \\ x &= -D^{-1}(L + U)x + D^{-1}b \end{aligned}$$

From this we have, identifying x on the left as the new iterate,

$$x^{(n+1)} = -D^{-1}(L + U)x^{(n)} + D^{-1}b$$

In Eq. 1,

$$b' = D^{-1}b = \begin{bmatrix} 1.8333 \\ 0.7143 \\ 0.2000 \end{bmatrix}$$

$$D^{-1}(L + U) = \begin{bmatrix} 0 & -0.3333 & 0.1667 \\ -0.2857 & 0 & 0.2857 \\ -0.2000 & -0.4000 & 0 \end{bmatrix}$$

- The procedure we have just described is known as the Jacobi method, also called "the method of simultaneous displacements", because each of the equations is simultaneously changed by using the most recent set of x-values. See Table 2.

3.2 Gauss-Seidel Iteration

- Even though we have $newx^1$ available, we do not use it to compute $newx^2$ even though in nearly all cases the new values are better than the old and ought to be used instead. When this done, the procedure known as *Gauss-Seidel* iteration.

- We proceed to improve each x-value in turn, using always the most recent approximations of the other variables. The rate of convergence is more rapid than for the Jacobi method. See Table 3. These values were computed by using this iterative scheme:

$$\begin{aligned} x_1^{(n+1)} &= 1.8333 + 0.3333x_2^{(n)} - 0.l667x_3^{(n)} \\ x_2^{(n+1)} &= 0.7143 + 0.2857x_1^{(n+1)} - 0.2857x_3^{(n)} \\ x_3^{(n+1)} &= 0.2000 + 0.2000x_1^{(n+1)} + 0.4000x_2^{(n+1)} \end{aligned}$$

beginning with $x^{(1)} = (0, 0, 0)^T$

	First	Second	Third	Fourth	Fifth	Sixth
x_1	0	1.833	2.069	1.998	1.999	2.000
x_2	0	1.238	1.002	0.995	l.000	l.000
x_3	0	1.062	1.015	0.998	1.000	1.000

Table 3: Successive estimates of solution (Gauss-Seidel method)

3.3 Sparse Matrices and Banded Matrices

- Many applied problems are solved with systems whose coefficient matrix is sparse-only a fraction of the elements are nonzero.
- In other applications the coefficient matrix may be sparse and have elements situated in selected positions.
- A banded matrix is one where the nonzero elements lie on diagonals parallel to the main diagonal.

4 Parallel Processing

We have mentioned that the operation of many numerical methods can be speeded up by the proper use of parallel processing or distributed systems.

- **Vector/Matrix Operations:**
  - For *inner products*, a very elementary case, we assume we have two vectors, v, u, of length n, and an equal number of parallel processors, $proc(i), i = 1, \ldots, n$, where each $proc(i)$ contains the components, v_i, u_i.
  - Then the multiplication of all the $v_i * u_i$ can be done in parallel in one time unit.
  - if the processors are connected suitably we can actually do the addition part in $log(n)$ time units.
  - This assumes a high degree of connectivity between the processors. The different designs are referred to as the topologies of the systems.
  - Suppose our processors were only connected as a linear array in which the communication send/receive is just between two adjacent processors.

$$P_1 \leftrightarrow P_2 \leftrightarrow \ldots \leftrightarrow P_{n-1} \leftrightarrow P_n$$

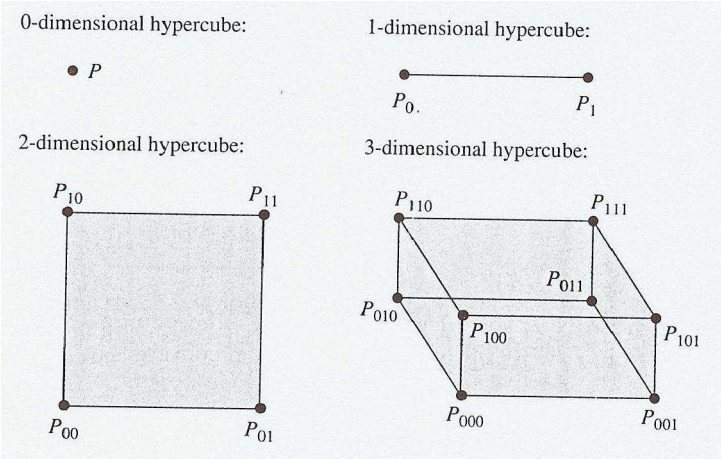

Figure 1: Connectivity between the processors (topologies).

- Then our addition of the n elements would be $n/2$ time steps, because we could do an addition at each end in parallel and proceed to the middle.
- The n-dimensional hypercube is a graph with 2^n vertices in which each vertex has n edges (is connected to n other vertices). This graph can be easily defined recursively because there is an easy albalgorithm determine the order in which the vertices are connected. See Fig. 1.
- Two vertices are adjacent if and only if the indices differ in exactly one bit.
- We can get to the $(n+1)$-dimensional hypercube by making two copies of the n-dimensional hypercube and then adding a zero to the leftmost bit of the first n-dimensional cube and then doing the same with a l to the second cube.
- There are many other designs for connecting the processors. Such designs include names like **star, ring, torus, mesh**
- For the *matrix/vector product,* Ax, we assume that processor $proc(i)$ contains the ith row of A as well as the vector, x.
- Because one processor is performing this dot product of row i of A and of vector x, the whole operation will only take $O(n)$ units of time.
- For the *matrix/matrix product,* AB, for two $n \times n$ matrices, we suppose that we have n^2 processors. Before we start our compu-

tations each processor, P_{ij} will have received the values for row i of A and column j of B. Based on our previous discussion, we can expect the time to be $O(n)$.

- **Gaussian Elimination:**
  - To see how this can be done in a parallel-processing environment, we must examine the row-reduction phase.
  - If we are computing in a parallel processing environment, we can take advantage of the independence by assigning each row-reduction task to a different processor:

$$\begin{bmatrix} 1 & 2 & 1 & 3 & 4 \\ 2 & 5 & 4 & 3 & 4 \\ 1 & 4 & 2 & 3 & 3 \\ 3 & 2 & 4 & 1 & 8 \end{bmatrix} \begin{array}{l} \\ \rightarrow proc(1) : R_2 - (2/1)R_1 \\ \rightarrow proc(2) : R_3 - (1/1)R_1 \\ \rightarrow proc(3) : R_4 - (3/1)R_1 \end{array}$$

  - Suppose each row assignment statement requires 4 time units, one for each element in a row. Then the sequential algorithm performs this stage of the row reduction in 12 time units, whereas we need only 4 time units for the parallel algorithm.
  - This example of parallel processing on the first stage of row reduction of a 4×4 system matrix generalizes to any row reduction in stage j of an $n \times n$ system matrix.
  - Recall that there are $n-1$ row-reduction stages in Gaussian elimination, one for each of the n columns of the coefficient matrix except for the last column. This suggests that we need $n-1$ processors to do the reduction in parallel.

Algorithms for Row Reduction in Gaussian Elimination:

```
Sequential Processing (without pivoting)
For j = 1 To (n - l) /*n-1 row-reduction stages*/
For i= (j + 1) To n /* 1st row-reduction stage */
For k = j To (n + 1)
a[i, k] = a[i, k] - a[i,j]/a[j,j]*a[j,k]
End For k
End For i
End For j
Parallel Processing
For i = 1 To (n - 1)
For k = i To (n + 1)
a[i, k] = a[i, k] - a[i,j]/a[j,j]*a[j,k]
End For k
End For i
```

- Sequential algorithm requires $O(n^3)$ operations and that the parallel algorithm with n processors accomplishes the same task in $O(n^2)$ operations.

- **Problems in Using Parallel Processors:**
  - The algorithm described here does not pivot. Thus, our solution may not be as numerically stable as one obtained via a sequential algorithm with partial pivoting.
  - The coefficient matrix A is assumed to be nonsingular. It is easy to check for singularity at each stage of the row reduction, but such error-handling will more than double the running time of the algorithm.
  - We have ignored the communication and overhead time costs that are involved in parallelization. Because of these costs, it is probably more efficient to solve small systems of equation using a sequential algorithm.
  - Other, perhaps faster, parallel algorithms exist for solving systems of linear equations.

- **Iterative Solutions -The Jacobi Method**
  - Recall that at each iteration of the algorithm a new solution vector $x^{(n+1)}$ is computed using only the elements of the solution vector from the previous iteration, $x^{(n)}$.

– Because Gauss-Seidel iteration requires that the new iterates for each variable be used after they have been obtained, this method cannot be speeded up by parallel processing.