8.5 General rectangular systems

Concepts

Singular or nonsquare systems
Reduced echelon systems
Gauss-Jordan elimination
Inconsistent systems
MSP function **`GaussJordan`**
Rank of a matrix
Homogeneous systems

Nonsquare or singular systems

The discussion of Gauss elimination in Sections 8.2 and 8.4 applied only to square nonsingular systems. What about nonsquare systems or square ones with one or more diagonal zeros in the equivalent upper triangular system? Systems of the last type are called *singular*. Instead of proceeding with the upward pass, you can perform further elimination operations (interchanging equations or subtracting a multiple of one equation from another) to convert the upper triangular system to an equivalent *reduced echelon* system. Such a system has two defining properties:

i) if a_{ij} is the first nonzero coefficient in Equation $[i]$, then $a_{kl} = 0$ whenever $k > i$ and $l \le j$, or $k < i$ and $l = j$;
ii) any equations whose a coefficients are all zero come last.

To construct an equivalent system satisfying (i), just use the diagonal coefficient in each equation of the upper triangular system in turn, if it's not zero, to eliminate the corresponding variable from all preceding equations. Here's pseudocode for that process:

```
for (k = 1; k ≤ min(m,n); ++k)
  if (a_kk ≠ 0)
    for (i = 1; i < k; ++i)
      { M = a_ik/a_kk;
        subtract M times Equation [k] from Equation [i]; }
```

To meet criterion (ii), reorder equations as necessary. Sometimes, criterion (i) is strengthened by requiring that the first nonzero coefficient in any equation be 1. That's easy to achieve by dividing the equation by the coefficient.

The process of eliminating unknowns above the diagonal as well as below is called *Gauss-Jordan elimination*. It could be carried out during the downward pass, simply by altering the loop in Line (5) of the pseudocode in Figure 8.2.1 for Gauss elimination. However, because it's used here only for nonquare or singular systems, the Gauss-Jordan step is treated separately. Now, as an example, it's applied to the upper triangular system considered earlier in Section 8.2:

$$\begin{cases} 7x_1 + 3x_2 + 8x_3 - 15x_5 - 12x_6 = -18 & [21] = [16] \\ 3x_2 + 2x_3 - 2x_4 - 2x_6 = -5 & [22] = [17] \\ -4x_4 - 6x_6 = -8 & [23] = [18] \\ -2x_4 + 5x_5 + 3x_6 = 3 & [24] = [19] \\ -5x_5 - 6x_6 = -7 & [25] = [20] + [19] \end{cases}$$

$$\begin{cases} 7x_1 + 6x_3 + 2x_4 - 15x_5 + 10x_6 = -13 & [26] = [21] - [22] \\ 3x_2 + 2x_3 - 2x_4 - 2x_6 = -5 & [27] = [22] \\ -4x_4 - 6x_6 = -8 & [28] = [23] \\ -2x_4 + 5x_5 + 3x_6 = 3 & [29] = [24] \\ -5x_5 - 6x_6 = -7 & [30] = [25] \end{cases}$$

$$\begin{cases} 7x_1 + 6x_3 - 10x_5 - 7x_6 = -10 & [31] = [26] + [29] \\ 3x_2 + 2x_3 - 5x_5 - 5x_6 = -8 & [32] = [27] - [29] \\ -10x_5 - 12x_6 = -14 & [33] = [28] - 2\cdot[29] \\ -2x_4 + 5x_5 + 3x_6 = 3 & [34] = [29] \\ -5x_5 - 6x_6 = -7 & [35] = [30] \end{cases}$$

$$\begin{cases} 7x_1 + 6x_3 + 5x_6 = 4 & [36] = [31] - 2\cdot[35] \\ 3x_2 + 2x_3 + x_6 = -1 & [37] = [32] - [35] \\ 0 = 0 & [38] = [33] - 2\cdot[35] \\ -2x_4 - 3x_6 = -4 & [39] = [34] + [35] \\ -5x_5 - 6x_6 = -7 & [40] = [35]\ . \end{cases}$$

To satisfy requirement (ii) for a reduced echelon system, just move the $0 = 0$ equation [38] to the end.

Equations like [38] of the form $0 = 0$ can be called *nonrestrictive*, since they have no effect on the solutions. But you should realize that the occurrences of the two zeros are almost accidental. For example, a slight change in the right-hand sides of the original equations would have yielded instead of [38] an equation

$$0 = b \quad [38^*]$$

where $b \neq 0$. Since no values of the unknowns can satisfy a system containing [38*], neither the reduced echelon nor the original system would have any solution. They would be *inconsistent*.

However, when no equation of the form [38*] occurs in the reduced echelon system, you *can* find a solution. This process is best described in terms of the "new" unknowns that you see in each equation, as you climb from bottom to top through the reduced echelon system. Here are the "new" unknowns in the equations of the reduced echelon example just considered:

[36] x_1
[37] x_2, x_3
[39] x_4
[40] x_5, x_6.

Except for the equations $0 = 0$, each equation will have at least one new unknown. To construct a solution, proceed from bottom to top, skipping the equations $0 = 0$. If an equation has more than one "new" unknown, assign arbitrary values to all but one of them, and solve the equation for the remaining one. This process is also called an *upward pass*. Here, then, is a solution for the earlier example:

$$\begin{cases} 7x_1 + 6x_3 + 5x_6 = 4 & [41] = [36] \\ 3x_2 + 2x_3 + x_6 = -1 & [42] = [37] \\ -2x_4 - 3x_6 = -4 & [43] = [39] \\ -5x_5 - 6x_6 = -7 & [44] = [40] \\ 0 = 0 & [45] = [38] \end{cases}$$

$$[44] \quad x_6 = \text{arbitrary value}$$

$$x_5 = \frac{-7 + 6x_6}{-5} = \frac{7}{5} - \frac{6}{5}x_6$$

$$[43] \quad x_4 = \frac{-4 + 3x_6}{-2} = 2 - \frac{3}{2}x_6$$

$$[42] \quad x_3 = \text{arbitrary value}$$

$$x_2 = \frac{-1 - 2x_3 - x_6}{3} = -\frac{1}{3} - \frac{2}{3}x_3 - \frac{1}{3}x_6$$

$$[41] \quad x_1 = \frac{4 - 6x_3 - 5x_6}{7} = \frac{4}{7} - \frac{6}{7}x_3 - \frac{5}{7}x_6 .$$

Here's a summary of the preceding discussion of the solution of nonsquare or singular systems:

> Suppose that the downward pass of Gauss elimination, then Gauss-Jordan elimination have been performed on a linear system, resulting in a reduced echelon system. If an equation of the form $0 = b$, where b is not zero, occurs in the reduced echelon system, then it and the original system are inconsistent: they have no solution. On the other hand, if no equation of that form occurs, then the system has at least one solution, and perhaps infinitely many, which can be computed by an upward pass.

Function **GaussJordan**

For completeness, MSP contains routine **GaussJordan** for computing reduced echelon forms:

```
                                  // Return  rank A  and the reduced
template<class Scalar>            // echelon form of  AX = B.  A  is
  int GaussJordan(                // mxn  and  X , B  are  nx1 .
    Matrix<Scalar>& A,            // Scalars  x  with  |x| < T  are
    Vector<Scalar>& B,            // regarded as zero.  Set status
           double   T,            // Code = 1  :  many solutions,
               int& Code);        //        0  :  one solution,
                                  //       -1  :  no solution.
```

The *rank* of a matrix is the number of nonzero rows in its reduced echelon form. The tolerance parameter is necessary to recognize unrestrictive rows of the form $0 = 0$ and inconsistent rows of the form $0 = b$ where $b \neq 0$. The source code for **GaussJordan** appears in Figures 8.5.1 and 8.5.2. The code resembles that of the basic Gauss elimination routine **GaussElim** through the downward pass: there's a minor adjustment to implement Gauss-Jordan elimination. Since only the reduced echelon form is computed, not the solution itself, there's no upward pass. However, some messy code is required to inspect and process the equation to get the rank and the final form of the output. Here are the high points:

- After the elimination steps are complete, coefficients with norm < **T** are replaced by zeros. (The appropriate tolerance for roundoff error may change from one application to another, so the client is given control.)
- Equations corresponding to nonzero rows of **A** are normalized by dividing by their first nonzero coefficients. This makes it easier to construct the solutions from the reduced echelon form. At the same time, the number of these equations is tallied to get the rank of **A**.
- Equations are shuffled so that nonrestrictive $0 = 0$ equations come last, preceded by any inconsistent equations.

In Section 8.8, function **GaussJordan** is tested on an eigenspace analysis problem.

```
template<class Scalar>                  // Return  rank A  and the
  int GaussJordan(                      // reduced echelon form of
      Matrix<Scalar>& A,                // AX = B.  A  is  mxn
      Vector<Scalar>& B,                // and  X , B  are  nx1 .
            double   T,                 // Regard  Scalars  x
               int&  Code) {            // with  |x| < T  as  0 .
    try {                               // Set  Code =
      int Low  = A.LowIndex(),          //  2 :  many solutions,
          m    = A.HighRowIndex(),      //  1 :  one solution,
          n    = A.HighColIndex();      //  0 :  no solution.
      if (Low != B.LowIndex()  ||       // If the index ranges are
            m != B.HighIndex() ||                // inconsistent,
            m  < Low || n < Low ) {              // report and
        cerr << "\nException  IndexError";       // throw an MSP
        throw(Exception(IndexError)); }          // exception.
      Code = (m < n ? 2 : 1);
      for (int k = Low; k <= min(m,n); ++k) {         // Downward
        for (int i  = k;                              // pass.
                 i <= m && A[i][k] ==
                           Scalar(0); ++i);
        if (i <= m) {
            if (i > k) {
              Vector<Scalar> T;  Scalar t;            // Swap
              T    = A[i];       t    = B[i];         // Equa-
              A[i] = A[k];       B[i] = B[k];         // tions
              A[k] = T;          B[k] = t; }          // i & k .
            for (i = Low; i <= m; ++i) if (i != k) {  // Gauss-
              Scalar M = A[i][k]/A[k][k];             // Jordan
              A[i] -= A[k]*M;                         // elimina-
              B[i] -= B[k]*M; }}                      // tion.
          else Code = 2; }                 // 0  on diagonal.
```

Figure 8.5.1 **GaussEl** function **GaussJordan**
(Part 1 of 2. Continued in Figure 8.5.2.)

```
  int Rank  = 0,                          // Rank  will be the
      Old_m = m;                           // number of nonzero
  for (k = Low; k <= m; ++k) {             // rows  A[k] .
    int L = Low - 1;                       // L  will indicate
    for (int j = k; j <= n; ++j) {         // the first nonzero
      if (abs(A[k][j]) < T)                // A[k]  entry.
          A[k][j] = 0;
        else if (L < Low) L = j; }         // If there is one,
    if (L >= Low) {                        // increment  Rank
        ++Rank;                            // and divide  Eq. k
        Scalar t = 1/A[k][L];              // by  A[k][L]  to
        A[k] *= t;                         // normalize it.  If
        B[k] *= t; }                       // not, move  Eq. k
      else {                               // to the end.  If
        Scalar Bk = B[k];                  // it's inconsistent,
        if (abs(Bk) < T) {                 // report that with
            Bk = 0;  L = Old_m; }          // Code  and use
          else {                           // Old_m  and  L  to
            Code = 0;  L = m; }            // make it precede
        for (int i = k; i < L; ++i) {      // any  0 = 0  equa-
          A[i] = A[i+1];                   // tions.  Arrange to
          B[i] = B[i+1]; }                 // avoid reprocessing
        A[L].MakeZero();                   // any equation just
        B[L] = Bk;                         // moved, and to redo
        --k;  --m; }}                      // Eq. k  if a new
  return Rank; }                           // one was moved up.
catch(...) {
  cerr << "\nwhile solving  AX = B  by Gauss-Jordan elimina-"
          "\ntion, where for  A ,  Low,HighRow,HighCol = "
       << A.LowIndex() << ',' << A.HighRowIndex()
                       << ',' << A.HighColIndex()
       << "\nand   for  B ,  Low,High = "
       << B.LowIndex() << ',' << B.HighIndex() ;
  throw; }}
```

Figure 8.5.2 **GaussEl** function **GaussJordan**
(Part 2 of 2. Continued from Figure 8.5.1.)

Homogeneous linear systems

One type of linear system is particularly important in theoretical considerations needed later. A *homogeneous* linear system has the form $A\xi = O$: that is,

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = 0 \\ \quad\vdots \qquad\qquad\quad \vdots \quad\ \vdots \\ a_{m1}x_1 + \cdots + a_{mn}x_n = 0 \end{cases}.$$

A homogeneous system always has the trivial solution $x_1 = x_2 = \cdots = x_n = 0$. To search for nontrivial solutions, perform the downward pass of Gauss elimination. Suppose $m < n$, or $m = n$ and a zero falls on the diagonal of the resulting equivalent upper triangular system. Now perform Gauss-Jordan elimination. In these two cases, the resulting reduced echelon system will contain at least one equation with more than one "new" unknown—that is, more than one unknown that does not occur in lower equations. Solution of that equation during the upward pass involves at least one arbitrary choice. Therefore, the system has infinitely many solution vectors. In summary,

> Every homogeneous system of m linear equations in n unknowns $x_1, \ldots, x_n$ has at least one solution, the trivial one $x_1 = \cdots = x_n = 0$. If $m < n$, or if $m = n$ and a zero falls on the diagonal of the upper triangular system resulting from the downward pass of Gauss elimination, then the system has infinitely many solutions.