Determinant and Matrix Inversion

The class *RMatrixInverseLinearSolver* in **Rts** contains the C++ code that implements the approach described in the following. The objective is to solve the system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1}$$

where **A** is a square known matrix, **b** is a known vector, and **x** is the unknown vector, is Cramer's rule and the use of determinants. This approach is inefficient compared with decomposition and iterative algorithms and it essentially establishes the inverse of the coefficient matrix **A** to obtain the solution as

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} \tag{2}$$

The approach is often referred to as Cramer's rule after Gabriel Cramer (1704–1752). Interestingly, it can be used to solve for individual values x_i leaving the other values of **x** unknown. One way to write Cramer's rule is

$$x_i = \frac{\det(\mathbf{A}_i)}{\det(\mathbf{A})} \tag{3}$$

where $\det(\mathbf{A}_i)$ is the determinant of $\mathbf{A}_i$, which is the matrix obtained by replacing column number *i* in **A** by **b**. Another way to write Cramer's rule is

$$\mathbf{A}^{-1} = \frac{\operatorname{adj}(\mathbf{A}_i)}{\det(\mathbf{A})} \tag{4}$$

where adj(**A**) is the adjoint matrix of **A**, i.e., the transposed matrix of cofactors. For a square matrix **A** the cofactor of a component is

$$C_{ij} = (-1)^{i+j} \cdot M_{ij} \tag{5}$$

where M_{ij} is the minor

$$M_{ij} = \det(\mathbf{A}_{\text{reduced}}) \tag{6}$$

where $\mathbf{A}_{\text{reduced}}$ is the reduced version of **A** with row *i* and column *j* removed. This leads to a recursive formula when the determinant is calculated as the sum over an arbitrary row or column:

$$\det(\mathbf{A}) = \sum \left(A_{ij} \cdot C_{ij} \right) \tag{7}$$

In the C++ implementation the sum is always taken over the first row, and the iteration takes place when the determinant-algorithm calls the cofactor-algorithm. Once the algorithms to calculate cofactors and determinant are in place the solution to **Ax**=**b** is calculated using Eq. (3). It can also be solved by establishing the inverse and calculating $\mathbf{x}=\mathbf{A}^{-1}\mathbf{b}$.

Other Algorithms and Applications for the Determinant

The algorithm above is only one of several to calculate the determinant. Liebniz and Laplace had their respective algorithms, and using LU decomposition is another approach. The determinant is defined for square matrices and was originally named by Gauss (Lagrange had earlier called it resultant) because of its ability to predict whether a solution to **Ax**=**b** exists. Chinese scholars had understood similar concepts several hundred years BC and people like Cardano and Leibniz had also used such tests. In passing, it is noted that for an inhomogeneous system of equations, i.e., when **b** is nonzero, a zero-valued determinant indicates there exist either infinite or no solutions, i.e., the determinant must be non-zero to obtain a unique solution. Conversely, for homogeneous systems (**b**=**0**) a zero determinant is the only way to get nontrivial solutions, which there will be an infinite number of. Also, a matrix **A** is said to be positive definite if the scalar $\mathbf{z}^T\mathbf{A}\mathbf{z}$ is positive for every non-zero column vector **z**. The determinant of a positive definite matrix is always positive; hence, a positive definite matrix is always nonsingular. A matrix is singular if its determinant is zero, i.e., a matrix with nonzero determinant is nonsingular. In practice the determinant is rarely calculated to check if **Ax**=**b** is solvable but it does indeed appear in Cramer's approach for solving that system of linear equations, as seen above.