Numerical Solution of a Set of Linear Algebraic Equations - Gaussian Elimination

At this point in our numerical sol'n to the static plate problem we have expressed the original integral eqn. is the form of a set of linear, constant coefficient equations:

$$[l_{mn}][\alpha_n] = [g_m].$$

From our previous discussions, we can assume that the fill routines for $[l_{mn}]$ & $[g_m]$ have been implemented.

The remaining task is to numerically solve for the unknown coeffs α_n which represent the amplitudes of all pulse basis fcts for the ρ_s on the plate.

Numerical solutions to this set of equations can be separated into two general classes -

(1) Direct methods : Gaussian elimination
Gauss - Jordan elimination
L-U decomposition

(2) Indirect (or iterative) methods : Gauss - Seidel
Conjugate Gradients

We will use only direct methods in this course. Of these, only the Gaussian elimination scheme will be discussed.

The basic concepts of the Gaussian elimination are used throughout most Direct Methods – although each method has its own variants.

(*) The motivating philosophy of the Gaussian elimination scheme is to reduce the original matrix equation to an equivalent set of eqns. which are upper triangular (i.e., the lower triangle is filled w/ zeros). From this, the unknowns can be easily solved for by back-substitution.

This solution is accompolished using nothing more than the elementary row operations on the augmented matrix eqn. (augmented = include RHS):

1.) Can multiply any row by a constant,
2.) Can add (or subtract) a multiple of one row to a multiple of any other row,
3.) Can interchange the order of any two rows

without changing the results of the original matrix equation.

Augmented matrix $\Rightarrow$ $\begin{bmatrix} l_{11} & l_{12} \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$

$$\left[\begin{array}{cc|c} l_{11} & l_{12} & g_1 \\ l_{21} & l_{22} & g_2 \end{array}\right]$$

To minimize numerical round-off error, we must guard against the following occurances:

(i) large numbers resulting from multiplications which could result in overflow or underflow errors.

(ii) division by zero.

(iii) equations with drastically larger coefficients.
↳ will not be a problem in this course.

Outline of the Gaussian elimination method –

(1) Augment the N×N coefficient matrix with the R.H.S. vector to form an $N \times (N+1)$ matrix.

(2) Interchange rows, if necessary, to make the first diagonal element (l_{11}) of largest magnitude in column 1 (called partial pivoting and is explained below).

(3) Create zeros in the second through N^{th} rows of the first column by subtracting l_{i1}/l_{11} times the first row from the i^{th} row of $[l_{mn}]$.

(4) Repeat (2) & (3) for the second through the $(N-1)^{th}$ rows, using partial pivoting (considering only elements in the j^{th} column in rows j through N). Then subtract l_{ij}/l_{jj} times the j^{th} row from the i^{th} row so as to create zeros in all positions of the j^{th} column below the diagonal.

At this point, the matrix is now upper triangular :

$$\left[\begin{array}{ccccc|c} l'_{11} & l'_{12} & l'_{13} & \cdots & l'_{1N} & g'_1 \\ 0 & l'_{22} & & & \vdots & g'_2 \\ 0 & 0 & \ddots & & \vdots & \vdots \\ \vdots & \vdots & & l'_{N-1,N-1} & l'_{N-1,N} & g'_{N-1} \\ 0 & 0 & \cdots & 0 & l'_{NN} & g'_N \end{array}\right]$$

Now backsubstitute

(5) Solve for α_N from the N^{th} equation -

$$\alpha_N = \frac{g'_N}{l'_{NN}}$$

(6) Solve for the remaining unknowns as

$$\alpha_i = \frac{g'_i}{l'_{ii}} - \sum_{j=i+1}^{N} \frac{l'_{ij}}{l'_{ii}} \alpha_j \qquad i = N-1, \cdots, 1$$

example, to find α_{N-1} once α_N has been calculated - using the $(N-1)^{th}$ eqn from the upper diagonal matrix

$$l'_{N-1,N-1}\alpha_{N-1} + l'_{N-1,N}\alpha_N = g'_{N-1}$$

$$\Rightarrow \quad \alpha_{N-1} = \frac{g'_{N-1}}{l'_{N-1,N-1}} - \frac{l'_{N-1,N}}{l'_{N-1,N-2}}\alpha_N \qquad (j = N-1)$$

This Gaussian elimination method is numerically unstable if the pivoting step is not used → round-off errors.

Full-pivoting is to perform elementary column & row operations such that the largest element (of a column & row corresponding to the j^{th} element in step (4)) lies on the diagonal. This diagonal element is called the pivot element.

Full-pivoting is typically not done. Instead, partial pivoting is used & is numerically stable. Partial pivoting uses elementary row interchanges to place the coeff. of largest magnitude in a column on the diagonal.

Example, solve the system of eqns. using Gaussian elimination -

$$\begin{aligned} \alpha_2 + 3\alpha_3 &= 4 \\ 4\alpha_1 + 6\alpha_2 + 2\alpha_3 &= 7 \\ \alpha_1 + \alpha_2 &= 9 \end{aligned}$$

Augmented matrix equation

$$\left[\begin{array}{ccc:c} 0 & 1 & 3 & 4 \\ 4 & 6 & 2 & 7 \\ 1 & 1 & 0 & 9 \end{array}\right]$$

Partial pivot

$$\left[\begin{array}{ccc|c} 4 & 6 & 2 & 7 \\ 0 & 1 & 3 & 4 \\ 1 & 1 & 0 & 9 \end{array}\right]$$

This row o.k. $\rightarrow$

Row 3 $-\frac{1}{4}$ Row 1 $\rightarrow$

$$\left[\begin{array}{ccc|c} 4 & 6 & 2 & 7 \\ 0 & 1 & 3 & 4 \\ 0 & 1-\frac{6}{4}=-\frac{1}{2} & -\frac{1}{2} & 9-\frac{7}{4}=\frac{29}{4} \end{array}\right]$$

$\downarrow$

Don't need to pivot since $|1| > |-\frac{1}{2}|$

Row 3 $-(-\frac{1}{2})$ Row 2 $\rightarrow$

$$\left[\begin{array}{ccc|c} 4 & 6 & 2 & 7 \\ 0 & 1 & 3 & 4 \\ 0 & 0 & -\frac{1}{2}+\frac{3}{2}=1 & \frac{29}{4}+\frac{4}{2}=\frac{37}{4} \end{array}\right]$$

Now, backsubstitute —

$$\alpha_3 = \frac{37}{4}$$

$$\alpha_2 = 4 - 3 \cdot \frac{37}{4} = -\frac{95}{4}$$

$$\alpha_1 = \frac{7}{4} - \frac{2 \cdot \frac{37}{4} + 6 \cdot \left(-\frac{95}{4}\right)}{4} = \frac{131}{4}$$

For a matrix with N eqns. and N unknowns, the time required to solve for the unknowns α_m is on the order of N^3!

The total time to solve for the unknowns is approximately

$$t \approx t_1 N^2 + t_2 N^3$$

where t_1 = time to compute 1 element of $[l_{mn}]$

$t_2 N^3$ = time to solve matrix eqn. $[l_{mn}][\alpha_m] = [g_n]$

[For large problems, efficiently written codes, the solve time will completely dominate a mm solver.]

A good source of linear equation solvers (plus a whole host of other excellent codes) is Linpack and Lapack.

Access through the web at www.netlib.org.

You can Browse or Search the Netlib repository for specific subroutines or entire packages.

Real matrices ← [Gaussian Elimination] → Complex matrices.

factor →	sgeco.f	cgeco.f
solve →	sgesl.f	cgesl.f