Substitutional Method in Structural Mechanics

J. Goodman
Bechtel Power Corporation, 12400 E. Imperial Highway, Norwalk, California 90650, U.S.A.

Abstract

A new method for the solution of the system of equations characterizing linear structural response is proposed. This method reduces the time complexity of the computational algorithm from $n^{2.81}$ to n^2 where n is a number of degrees of freedom. The method also can be generalized for use in nonlinear response analysis.

1. Introduction

The equation of linear structural response to external force vibration of frequency ω for finite element approximation can be presented in the form:

$$[A]\{x\} = \{P\} \tag{1}$$

where $\{x\}$ is the n-vector of relative displacements, $\{P\}$ is the n-vector of external forces and $[A]$ is a matrix determined by formula:

$$[A] = \omega^2[M] + i\omega[C] + [K] \tag{2}$$

where $[M]$ is the mass matrix, $[C]$ is the damping matrix and $[K]$ is the stiffness matrix. To make the matrix $[A]$ real we may assume $[C] = 0$.

The solution to eq. (1) can be found be inverting the matrix $[A]$:

$$\{x\} = [A]^{-1}\{P\} \tag{3}$$

This procedure is time-consuming for a large number of nodes. Even the best algorithm of solution (see Aho et. al., [1]) has a time complexity T(n) of the order:

$$T(n) \sim n^{2.81} \tag{4}$$

where n is a number of degrees of freedom. For some stress analysis problems typical for nuclear power plants, n may assume values as large as 20,000 - 30,000.

Further progress in seismic PRA analysis depends on the success in simplifying the seismic response analysis, which usually consumes about 90 percent of all computational time. In this paper we propose the method which can significantly reduce the computational time without reducing the accuracy.

2. Matrix Size Reduction

Only components of the load n-vector {P} applied to the external nodes could be nonzero. The number of nonzero components, m, typically is much smaller than the total number of equations n:

$$m << n \tag{5}$$

The assumption (5) is not limiting. Due to linearity of the system of equations (1) we can present a solution of this system as a superposition of solutions provided that for each of them assumption (5) is met. The optimum way of the decomposition of the original problem into simpler problems satisfying the inequality (5) will be considered later.

Let us denote the nonzero components of the vector {P} as $F_\alpha (\alpha = 1, 2, \ldots m)$. Then expression (3) can be reduced to the form:

$$x_i = \sum_{\alpha=1}^{m} B_{i\alpha} F_\alpha, \quad (i = 1, 2, \ldots n) \tag{6}$$

where $B_{i\alpha}$ is a reduced rectangular matrix which can be obtained from the square matrix $[A^{-1}]$ by eliminating columns which have numbers coinciding with numbers of zero elements in the n-vector of external loads {P}.

Assume that m different solutions $x_i^{(\beta)} (i = 1, 2, \ldots n; \beta = 1, 2, \ldots m)$ of the system (1) corresponding to m different sets $F_{(\alpha)}^{(\beta)}$ of external forces are known. In our notation the superscript (β) numbers solutions. Generally, none of the sets $F_\alpha^{(\beta)}$ coincide with F_α from eq. (6).

If we insert $x_i^{(\beta)}$ and $F_\alpha^{(\beta)}$ into eq. (3) we can obtain the system of equations for elements of the matrix $B_{i\alpha}$.

$$\sum_{\alpha=1}^{m} F_\alpha^{(\beta)} B_{i\alpha} = x_i^{(\beta)}, \quad (\beta = 1, 2, \ldots m) \tag{7}$$

For every fixed number i there is a system of m equations with m variables. Because the matrix of coefficients $F_\alpha^{(\beta)}$ does not depend on the number i we can readily solve this system for any i by inverting the m x m matrix $F_\alpha^{(\beta)}$:

$$B_{i\alpha} = \sum_{\beta=1}^{m} H_{\alpha\beta} x_i^{(\beta)} \tag{8}$$

where $H_{\alpha\beta}$ is an inverted matrix relative to $F_\alpha^{(\beta)}$.

Putting the expression $B_{i\alpha}$ from eq. (8) into eq. (6) we obtain the solution of eq. (1):

$$x_i = \sum_{\alpha=1}^{m} \sum_{\beta=1}^{m} H_{\alpha\beta} x_i^{(\beta)} F_\alpha \tag{9}$$

To find this solution we have to invert the matrix of significantly lesser dimension. The reduction coefficient of the computational time can be defined as:

$$R = \left(\frac{n}{m}\right)^{2.81} \tag{10}$$

For example, if n/m = 10 then $R \approx 645$. However, to utilize this advantage we have to know m arbitrary solutions with arbitrary $F_\alpha^{(\beta)}$. Because of arbitrariness of $F_\alpha^{(\beta)}$ there is a possibility to find these solutions without a real solution of Eq. (1).

3. Substitutional Algorithm

Let us split the system of equations (1) into two systems of n-m homogeneous equations and m nonhomogeneous equations:

$$\sum_{k=1}^{n} A_{ik}\, x_k = 0 \qquad (i = 1, 2, \ldots n-m) \tag{11}$$

$$\sum_{k=1}^{n} A_{\alpha k}\, x_k = F_\alpha \qquad (\alpha = 1, 2, \ldots m) \tag{12}$$

We may consider eq. (11) as a linear system of n-m equations for n arguments and eq. (12) as a definition of m functions F_α of n arguments x_k. In this case the system (11) has an infinite number of solutions. Picking up any solution $x_k^{(\beta)}$ of the system (11) and inserting it into eq. (12) we estimate m functions $F_\alpha^{(\beta)}$. Therefore, the solution $x_k^{(\beta)}$ is a solution of the system (1) with $F_\alpha = F_\alpha^{(\beta)}$.

Now, we have to find the method to obtain an arbitrary solution of eq. (11) without really solving this equation. For this purpose a substitutional algorithm was developed.

Because every internal node has ν neighboring nodes, any equation in the system (1) has no more than $\nu + 1$ nonzero coefficients A_{ik}. There are no two equations with the same sets of variables x_k having nonzero coefficients A_{ik}. Due to these properties we can formulate a substitutional algorithm to obtain $x_i^{(\beta)}$. This algorithm consists of several steps.

Step 1. Select equations of the lowest size from the system (11) where the size of an equation is a number of nonzero coefficients A_{ik} in the left side of the equation.

Step 2. If the size of equations is equal to 1 then calculate corresponding variables x_k by dividing the right side of the equation by the coefficient of this variable. Otherwise, go to Step 3.

Step 3. Assign an arbitrary value to the variable x_k included, at least, in one of the lowest size equations.

Step 4. Insert an adopted value of a selected variable from Step 3 in all equations which contain this variable reducing the size of these equations by one, then go to Step 1.

After exhausting this algorithm all values x_k will be specified. Applying this procedure m times and using different arbitrary values we can obtain in different solutions $x_i^{(\beta)}$ corresponding to different $F_\alpha^{(\beta)}$.

4. Design and Time-Complexity of the Substitutional Algorithm

The least time-consuming algorithm for the substitutional method has a time-complexity proportional to mn. Four arrays are used to implement this algorithm. The first array has n rows and $2\nu + m + 2$ columns. It is designed to store current information about the system of equations. The first $\nu + 1$ columns of every row are reserved to store variable numbers for those variables which have nonzero coefficients. The next $\nu + 1$ columns are reserved for their coefficients and the last m columns for the absolute terms P_i for m different solutions.

The second array consists of $\nu + 1$ rows and $n + 1$ columns and has to store current information about sizes of equations in the system. The first row is allocated to store information about all equations with one term at the left side, the second row for two term equations, and so on. The first column is assigned for a counter which shows how many equations have a specified size. Other columns are reserved for equation numbers for those equations which have this size. This arrangement allows one to read directly the last equation number of any size without reading the entire row.

The third array has dimension $n \times (\nu + 1)$ to store information about the location of every variable. The fourth array has dimension $m \times n$ to store m solutions for n variables.

The performance of our substitutional algorithm starts from reading the first column in the second array to find the first nonzero counter. After finding this counter we locate and read the last equation number, erase this number from the array and reduce the counter by one. Following the equation number we read a corresponding row in the first array. We assign arbitrary values for all variables but one, calculate the value of the last variable using this equation and place the results in the last array. This procedure should be repeated m times to find all solutions for the variables under consideration simultaneously. After that we delete information from this row and read from the third array locations of all equations containing these variables. Then we go to corresponding equations, substitute all already determined variables with their values and subtract corresponding terms from absolute terms. We do this m times and place results into m different columns. Finally, we update the second array because the sizes of some equations will reduce. After that we repeat the entire algorithm again till the last array of solutions will be completely filled out.

It is easy to see that every step of our substitutional algorithm contains the finite number of operations which does not depend on n or m and is proportional to the

number of neighboring nodes ν. Therefore, the time complexity of algorithm is proportional to the total number of steps, i.e. mn.

To find the time complexity of the method we have to choose the greatest among two values: time complexity of the matrix inversion $m^{2.81}$ and time complexity of the substitutional algorithm mn:

$$T(m,n) \sim \max \{m^{2.81}, mn\} \tag{13}$$

If the following inequality holds

$$m \leq n^{0.5525} \tag{14}$$

then the leading term is eq. (13) is mn. If $m > n^{0.5525}$ then we can break m nonzero absolute terms into k groups, solve k independent systems with m/k different nonzero absolute terms in each and find the sum of these solutions. According to the superposition theorem (see, for example, Korn and Korn [2]), the sum is the solution of the original system.

The time complexity of the substitutional algorithm does not change but the time complexity of the matrix inversion part of the program will be reduced by $K^{1.81}$ times. Therefore, it is always possible to make the term mn in eq. (13) dominant. Hence, the computational time reduction coefficient k can be estimated as:

$$R = \frac{n^{1.81}}{m} \tag{15}$$

In the worst case m=n the reduction coefficient is:

$$R = n^{0.81} \tag{16}$$

This means that using the substitutional method we can construct the matrix inversion algorithm with the time complexity $T(n) \sim n^2$. However, this method is not universal. It can be applied only to matrices with no more than $\nu + 1$ nonzero elements in any row where ν is a constant number independent from n.

This method could be readily generalized for the case of nonlinear response analysis. Actually, the substitutional method will work even if the matrix [A] is a function of vector {x}. Assuming the response function of a polynomial type, we obtain the linear system of equations for response coefficients; however the dimension of this equation will be larger than that in linear case.

References:

[1] A.V. AHO, JOHN E. HOPCROFT, J. D. ULMAN, The Design and Analysis of Computer Algorithms, Reading, Massachusetts: Addison-Wesley Publishing Company, 1974

[2] KORN, G. A., and KORN, J. M., Mathematical Handbook for Scientists and Engineers, New York: McGraw-Hill Book Company, 1968.