Automated Search for Optimal Standard Sections

S. GINSBURG and G. KAAZ

Microcomputers provide the engineer with practical tools for structural analysis and design. Beyond the obvious analysis, usually performed by using canned programs, there are many useful operations for which the microcomputer can be employed. This paper describes one such application, the search for minimum-weight cross sections. Done manually, this is a tedious job for the designer. Even when automated optimization is performed,[1,2] the task of choosing a standard section still remains.

An efficient search algorithm is described here, which can easily be adopted by practitioners, especially if they use BASIC for programming. The method is demonstrated by several examples where a minimum area is chosen from the table of W shapes in the AISC *Manual of Steel Construction*.[3]

THE METHOD

First, it is assumed an axially loaded column is under consideration. The objective is to find a cross section which satisfies the stress constraints.

for $Kl/r<C_c$

$$F_a = \frac{\left[1 - \frac{(Kl/r)^2}{2C_c^2}\right]F_y}{\frac{5}{3} + \frac{3(Kl/r)}{8C_c} - \frac{(Kl/r)^3}{8C_c}} \tag{1}$$

for $Kl/r>C_c$

$$F_a = \frac{12\pi^2 E}{23(Kl/r)^2} \tag{2}$$

where

$$C_c = \sqrt{2\pi^2 E/F_y} \tag{3}$$

S. Ginsburg is an Assistant Professor, Department of Civil Engineering, The University of Kansas, Lawrence, Kansas.

G. Kaaz is an undergraduate student, Department of Civil Engineering, The University of Kansas, Lawrence, Kansas.

where Kl/r is the largest effective slenderness ratio, F_y is the yield stress, and E is the modulus of elasticity. The algorithm requires that the cross section will be of minimum area.

Theory

The simplest case possible is a hypothetical one where the allowable stress is independent of the radius of gyration r. Here, it is necessary to find the smallest area which satisfies the stress constraint without referring to r. For such a column, it is necessary to rearrange the table of W shapes according to increasing values of areas. Assuming there are n (about 200) available sections, the worst case is a search over all n values. But, it is possible to divide the entire table (arranged now by order of increasing areas) into two almost equal parts. If the area at the middle of the table satisfies the stress constraint, then only the lower values should now be searched for the minimum area. If the middle value does not satisfy the stress constraint, the upper half of the table should be scanned. So, the worst case now is about $n/2$ values to be checked. But, once it is known the minimum area lies within the lower (or upper) half of the table, it is possible to divide that half into two almost equal parts. Now the search proceeds as before, but on a smaller interval with about $n/2$ values. The $n/2$ interval is divided into two parts to obtain an $n/4$ portion, etc. Thus, instead of searching all n values, or at least $n/2$ values when we divide the table once, we can find the optimum by merely checking $\log_2 n$ values, if we repeat the process of dividing the reduced intervals by 2. Of course, this result is an approximation for large n. It is quite accurate for typical numbers of standard sections, especially if combinations consisting of several sections are possible. A conservative estimate for the number of points to be checked is $\log_2 n + 1$.

Practice

When the radius of gyration is included in the calculation, as is the case in most practical applications, the

situation becomes more complicated. First, it will be assumed that r_y alone is considered, e.g., when the x direction is braced. The table of standard sections is still arranged by order of increasing areas. Now, a set of points is determined with the following properties: every point has an area which is larger than that of all previous points, and its r_y is larger too. This is illustrated in Table 1, which shows a small portion of the new arrangement, together with several points of the set. There are 30 such points altogether, and the search algorithm described is applied to this set first.

Once the minimum area has been found for the 30-point set, it remains to determine whether the true optimum lies within an interval defined by two such points. The largest number of points within such an interval is 10 (Table 1). It is possible to find a local point in that interval which serves as a middle point. If the stress constraint is violated by this point, the optimum lies within the upper half of the interval, or at the upper limit established by the 30-point search. Otherwise, the optimum is a value within the lower half of the interval, or again the upper limit established by the 30-point search. Thus, the *worst case* requires one to check 11 points only. If, for the sake of simple programming, the middle points in the large interval of 10 points, or some smaller ones, are not taken into account, the worst case will require 16 points to be checked.

Table 1. Arrangement for r_y Search

Number	Designation	Area	r_y	
.				
.				
115	W30×108	31.7	2.15	
116	W14×109	32.9	3.73[a]	
117	W21×111	32.7	2.90	
118	W10×112	32.9	2.68	
119	W27×114	33.5	2.18	
120	W30×116	34.2	2.19	8-point interval
121	W24×117	34.4	2.94	
122	W33×118	34.7	2.32	
123	W18×119	35.1	2.69	
124	W12×120	35.3	3.13	
125	W14×120	35.3	3.74[a]	
126	W21×122	35.9	2.92	
127	W30×124	36.5	2.23	
128	W33×130	38.3	2.39	
129	W24×131	38.5	2.97	
130	W14×132	38.8	3.76	← middle point for
131	W21×132	38.8	2.93	10-point interval
132	W30×132	38.9	2.25	
133	W36×135	39.7	2.38	
134	W12×136	39.9	3.16	
135	W33×141	41.6	2.43	
136	W14×145	42.7	3.98[a]	
137	W27×146	42.9	3.21	
.				
.				

[a]Part of 30-point global set

From the programmer's standpoint, a vector (pointer) is to be used, which contains the positions of the 30 points to be checked in the global search. Another pointer contains the positions of intermediate points within intervals defined by the global points.

The procedure used for r_y can be applied to r_x with minor modifications. In this case the set of points which serve for the global search consists of 36 points, requiring a second pointer-vector. In addition, there is a group of 14 sections (the large W12 and W14) which have large areas with relatively small r_x. This group requires another pointer, and although it will never provide a global minimum, these sections may be needed for cases with size constraints. This topic will be discussed later.

Generalization

The case where both r_x and r_y are to be checked needs to be handled in a different way. The two sets of global points (36 for r_x and 30 for r_y) are not common to both axes; thus, a minimum for one direction is in general not optimal for the other direction.

This case is solved by the following method. First, the table of standard cross sections is divided into two groups. The first group contains sections belonging to W4 through W14, and the second group consists of W16 through W36. Each group is arranged according to an increasing value of m, where m indicates the size (i.e.Wm). Within each Wm group the order of sections is by increasing value of n, where n indicates the second size parameter (W$m \times n$). This establishes two sets of global points, used in Table 2, each of which is arranged by order of increasing r_x and r_y. That is, these points are arranged (within each of the two groups) by order of increasing area and both r_x and r_y. The total number of these points is 15. Now the search for optimum section is conducted as before, i.e., the global points are examined first, and they establish upper and lower bounds. Moreover, the search begins with the first global group, and pointers to that group sometimes eliminate searching in the second group. For example, if a global search indicates a W10 section does not violate the constraints, there is no need to search sections in the second group above W30. Furthermore, once an acceptable section is found within a group, say W10×30, there is no need to search any section with a larger area; thus, all sections above W16×31 are skipped. Of course, within each group, such as W12, calculations are skipped for all sections having an area larger than that of W10×30. Once the upper and lower bounds have been determined, the sections lying in this interval are already arranged by order of increasing r_x and r_y. A local search (by halving) is now conducted within the interval defined by the lower and upper bounds.

Thus, for a search with both the x and y-directions active, three pointer vectors are used, one for global points (two sets), one for saving unnecessary searches

Table 2. Arrangement for r_x and r_y Search

Number	Designation	Area	r_x	r_y		
.						
.						
101	W14×500	147.	7.48	4.43		
102	W14×550	162.	7.63	4.49	Subset of	
103	W14×605	178.	7.80	4.55	Group I	
104	W14×665	196.	7.98	4.64		
105	W14×730	215.	8.17	4.69[a]		
106	W16×026	7.68	6.26	1.12		A i
107	W16×031	9.12	6.41	1.17		r n
108	W16×036	10.6	6.51	1.52		e c
109	W16×040	11.8	6.63	1.57		a r
110	W16×045	13.3	6.65	1.57	Subset of	e
111	W16×050	14.7	6.68	1.59	Group II	r_x a
112	W16×057	16.8	6.72	1.60		s
113	W16×067	19.7	6.69	2.46		r_y e
114	W16×077	22.6	7.00	2.47		
115	W16×089	26.2	7.05	2.49		
116	W16×100	29.4	7.10	2.51[a]		
117	W18×035	10.3	7.04	1.22		
.						
.						

[a]Part of 15-point global set

and one for local points within the upper and lower bounds.

There is another important advantage to the arrangement of the table described here. In case of size constraints, there is no difficulty in conducting the search, as sections violating the constraints (too small or too large) are automatically eliminated without creating any problem. That is, the global points pertaining to sections which are either too large or too small are skipped. Also, it is possible to use the method with constraints on the flange size, by eliminating internal points. Any other ordering method requires definition of additional pointer vectors which will enable elimination of sections that cause size violation. This task imposes a problem, as global points or pointer locations might be eliminated, and alternative ones need to be defined. Thus, even if a search in one direction is more efficient, and therefore should be employed in cases where one direction dominates, the more general approach seems to be the practical one as it allows handling of both directions and size constraints at the same time.

For the more general case there are some exceptions, such as the W8 × 24, with a value of r_x smaller than that of the preceeding section, W8 × 21. Actually, the following two sections also have r_x smaller than that of W8 × 21. The program handles these cases, so that nonoptimal sections are not chosen, unless of course a size constraint dictates differently.

CONCLUSION

The algorithm described in this paper demonstrates how, with little programming effort, the engineer can take advantage of microcomputers to solve practical problems in the design office. Dedicating a small machine for tasks such as the search for optimum cross sections is useful, as it saves a lot of effort associated with selection of values from tables, and numerous calculations.

The algorithm used for the program is simple, and the programming is straight-forward. Moreover, it is possible to write the program in a variety of high-level languages with no difficulties.

The method or algorithm used for selecting standard sections is useful for a variety of applications. The main advantage of the algorithm described here is the final result which, unlike some other optimizers, does not require any additional work, i.e., selecting a real section close to some "continuous" approximation. The latter might lead to nonoptimal solutions. It has been found that for all cases studied here, the solutions obtained from the program were lighter than those obtained from the approach outlined in Ref. 2. Therefore, the program can and should be included as a subroutine in a more comprehensive program. As a result of such integration, the user will be able to obtain optimal solutions, and at any stage of the design process stop the program and still have a feasible, locally optimal design.

REFERENCES

1. *Kirsch, U.* Optimum Structural Design *McGraw-Hill, 1981.*
2. *Pesquera, C. I.* Integrated Analysis and Design of Steel Frames with Interactive Computer Graphics *Department of Structural Engineering, Cornell University, March 1984.*
3. *American Institute of Steel Construction* Manual of Steel Construction *8th Ed., Chicago, Ill., 1980.*