European Journal of Applied Sciences 3 (4): 162-168, 2011
ISSN 2079-2077

A Finite Element Java Program for Stability Analysis of Pre-Cracked Beam-Columns

Peter N. Jiki

Department of Civil Engineering, University of Agriculture, Makurdi, Benue State, Nigeria

Abstract: The paper reports the findings of a study on the stability analysis of pre-cracked beam-columns using the finite element method and a newly developed java code. First a stiffness reduction parameter α_1 which reduces the stiffness of the beam-column due to pre-crack is calculated. The calculated stiffness parameter is then used to modify the stiffness of existing beam element as $EI(1-\alpha_1)$, so that when $\alpha_1=1$, the beam-column has completely failed. For now a well known beam element for flexural buckling analysis is modified by the introduction of the stiffness parameter. Here only the elastic stiffness is modified. The geometric stiffness is not affected therefore it is not modified. The elastic and geometric stiffness matrices are assembled into a global eigen value matrix equation which is solved using an inverse power method of matrix iteration. An object oriented program in java is developed for the solution of matrix eigenvalue equations by first transforming the matrix equations into a standard form ready for iteration by the inverse power method. The results of the present study using the proposed java code compare well with those using analytical or finite element methods using procedural codes mainly fortran codes. It is concluded that java programming language holds a great potential as a candidate for future engineering analysis tool and that a stiffness reduction parameter α_1 proposed herein can be used as a structural health monitoring parameter as a failure indicator.

Key words: Stability · Finite element · Pre-cracked · Java code · Programming · Object-oriented · Beam-columns

INTRODUCTION

In the stability and vibration analyses of engineering structures, the quantities EI,GJ and EI_w are often used to represent flexural, torsion and warping stiffness respectively [1-9]. These quantities are used universally to represent the elastic stiffness of components without cracks. For cracked components however, various forms of representing the cracked stiffness of the components are expressed either as $k_T h$ as in Dimarogonas[2] or as $2k_T l$ as in Okamura *et al*. [4], in which k_T is the torsion spring constant of the cracked component. As can be seen from the above review so far there is no universal or standard way of expressing the stiffness of a cracked or pre-cracked component. In a recent work by Jiki [9], the effect of an edge crack on the reduced stiffness of a pre-cracked beam-column was reported. A stiffness reduction(decay) parameter k due to the presence of an edge crack in a column or a beam-column was proposed and was used to study the effect of stiffness decay on the stability characteristics of pre-cracked thin-walled beam-columns. It was found and concluded that the use of a stiffness decay parameter k proposed by Jiki [9] can be universally used to rapidly calculate stiffness degradation due to pre-crack. This rapid calculation can be very useful in fatigue analyses of structural components as the calculations would reduce to evaluations of either EI(1-k) or GJ(1-k) for flexural or torsion stiffness decay respectively. In Jiki [9] an analytical model was used to study the stability characteristics of pre-cracked beam-columns. Apart from rapid calculation of stiffness decay and attempt to quantify failure of components through the use of parameter k, it was also concluded by Jiki [9] that cracks reduce load carrying capacity of pre-cracked structural components as well as alter the vibration characteristics of such components [6,7]. For rapid modeling of pre-crack effects in beam-columns, the finite element method and an object-oriented code are needed as modern tools for analysis [14-19].

The purpose of the present work is to use a newly developed java code for finite element stability analysis of pre-cracked beams and beam-columns.

Corresponding Author: Peter N. Jiki, Department of Civil Engineering, University of Agriculture, Makurdi, Benue State, Nigeria.

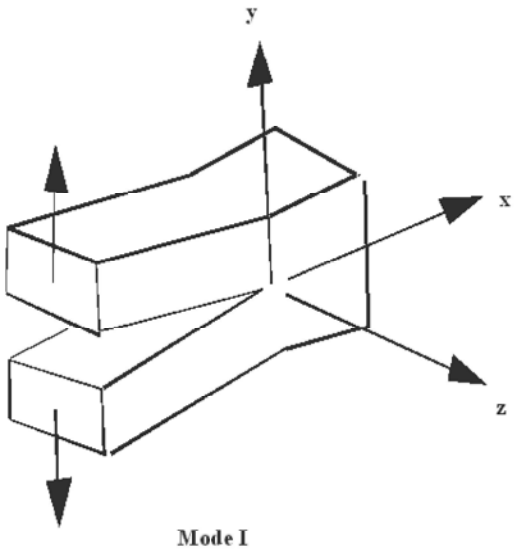

Fig. 1: Failure mode for calculation of statess intensity factor.

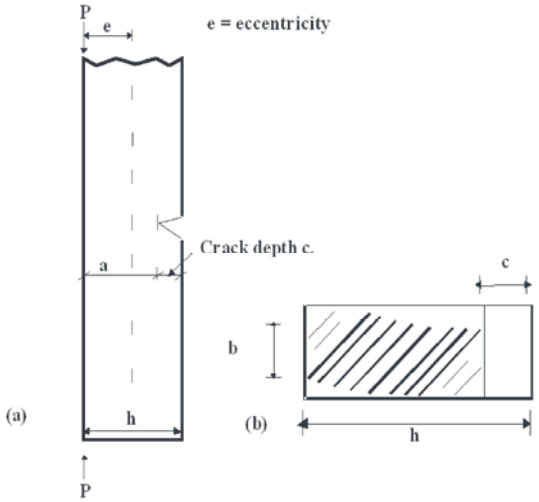

Fig. 2: Pre-Cracked beam -column : (a) Edge creak in a loaded by axial load peccentricity e. (b) section of the cracded bar.

Derivation of Stiffness Reduction Parameter: According to Parker [8] and Sih [1] fast fracture prediction depends on knowledge of critical strain energy density factor S which for mode 1 opening it can be shown that in the direction of fracture we have [8]:

$$S = \frac{k_1^2(1-2\upsilon)}{4\pi\mu} \tag{1}$$

When $k_1 \rightarrow k_{1c}$

$$S_c = \frac{k_{1c}^2(1-2\upsilon)}{4\pi\mu} \tag{2}$$

For the pre-cracked beam-column problem considered here for mode 1 opening (Figs. 1 and 2), we assume that $k_1<k_{1b}<k_{1c}$ and $S<S_b<S_c$. Then at buckling failure, we have the buckling strain energy density for mode1 opening as:

$$S_b = \frac{k_{1b}^2(1-2\upsilon)}{4\pi\mu} \tag{3}$$

Here k_1, k_{1b} and k_{1c} are stress intensity factors, μ is shear modulus and υ is poisons ratio. Another factor for characterizing stress field around a cracked body is the strain energy release rate G_1 (which is related to the compliance of a component) given as[1]:

$$G_1 = \frac{k_1^2(1-\upsilon^2)}{E} \quad (\forall\ plain\ strain) \tag{3}$$

It seems from equations (1) and (4) that for some problems both S and G_1 are similar. This we can write as:

$$S=G_1=Ck_1^2 \tag{5}$$

From equation (5) we can also use the strain energy density factor S to calculate the compliance of a cracked body. Without going into detailed derivation, we propose our stiffness reduction parameter due to crack in terms strain energy density as:

$$\alpha_1^2 = \frac{\frac{k_1^2(1-2\upsilon)}{4\pi\mu}}{\frac{k_{1b}^2(1-2\upsilon)}{4\pi\mu}} = \frac{k_1^2}{k_{1b}^2} \tag{6}$$

Therefore $\alpha_1 = \sqrt{\frac{S}{S_b}} = \frac{k_1}{k_{1b}}$

For mode1 opening of a crack due to bending moment M, a well known result is (see Anifantis and Dimarogonas[5]):

$$k_1 = \frac{6M}{bh^2}\sqrt{\pi c_1}\, Y_1\left(\frac{c_1}{h}\right) \tag{8}$$

In which

$$Y_1\left(\frac{c_1}{h}\right)=\sqrt{\frac{2h}{\pi c_1}\tan\frac{\pi c_1}{2h}}\;.\;\frac{0.923+0.199\left(1-\sin\frac{\pi c_1}{2h}\right)^4}{\cos\frac{\pi c_1}{2h}} \quad (9)$$

At buckling, the buckling moment M_b is known and the critical stress intensity factor k_{1b} is given in terms of the buckling moment M_b as:

$$k_{1b}=\frac{6M_b}{bh^2}\sqrt{\pi c_b}\;Y_b\left(\frac{c_b}{h}\right) \quad (10)$$

In which

$$Y_b\left(\frac{c_b}{h}\right)=\sqrt{\frac{2h}{\pi c_b}\tan\frac{\pi c_b}{2h}}\;.\;\frac{0.923+0.199\left(1-\sin\frac{\pi c_b}{2h}\right)^4}{\cos\frac{\pi c_F}{2h}} \quad (11)$$

Substitution of equations (8)-(11) into equation (7) and simplifying, we have:

$$\alpha_1=\frac{k_1}{k_{1b}}=\frac{M}{M_b}\sqrt{\frac{c_1}{c_b}}\;\frac{Y_1}{Y_b} \quad (12)$$

For constant eccentricity of loading considered in the present work equation (13) reduces to

$$\alpha_1=\frac{P}{P_b}\sqrt{\frac{c_1}{c_b}}\;\frac{Y_1}{Y_b} \quad (13)$$

Finite Element Analysis Model: Analysis of the buckling strength of a pre-cracked is formulated as a generalized eigen value problem of the form [12]:

$$\left(\left[K\right]-\lambda\left[G\right]\right)\left\{R\right\}=\left\{0\right\} \quad (14)$$

By introduction of the stiffness parameter α_1 into the assembled elastic stiffness of equation (14) we have :

$$\left(\left[\overline{K}\right]-\lambda\left[G\right]\right)\left\{R\right\}=\left\{0\right\} \quad (15)$$

In which

$$\left[\overline{K}\right]=(1-\alpha_1)\left[K\right] \quad (16)$$

Equation (15) is transformed to standard eigen value equation and solved by inverse power matrix iteration using our newly developed java code.

Structure of the Finite Element Java Program: The java code we have developed and used for the present work is reported in Jiki [19]. The architecture of the code is simple. We use a beam element to carry out the coding of the elastic and geometric element stiffness class. Actually, the package mainApplicationCalculatej forms the elastic and geometric element stiffness class. The two are combined into one class. Then we have the assembly class and finally the boundary condition class making a total of three classes in that package. Since matrix eigenvalue equation for the problem at hand uses consistent elastic and geometric beam element stiffness matrices, the assembled matrix equation is also consistent and to use the inverse power method we have to transform the eigenvalue equation into a standard form ready for use in the vector iteration. To do this transformation, we develop another package called calculate which uses Gauss Jordan method to invert the elastic stiffness matrix and multiply with the geometric stiffness matrix to form another matrix C. The package calculate also carries out the iteration to convergence and returns the required maximum eigenvalue, the inverse of which gives the lowest load factor. The third and the last package is the mainApplication package which has the main method signature, the input class, the output class and all the necessary constructors that initialize the objects needed to produce results. We have first tested the packages in the various procedures before putting all together, that is, we tested the Gauss Jordan matrix inversion separately using well known published works by Bathe [16], Paz [17] and Rajasekeran [18]. We also tested matrix iteration routine using the break and the continue control structures where necessary. Having satisfied ourselves that the test results from the program are satisfactory, we tested the program with known stability problems. We have used the program to study column buckling problems with pin-pin, fixed-fixed, fixed-free and we are convinced that the code gives good and acceptable results. Besides, we have developed similar programs using fortran codes which are ready for comparison with the new java code. Because java language has a very flexible control structure and with careful design of the program to avoid indefinite loops, convergence in the iterations are very good. In solving the problems or the

examples presented in the present work, we observed numerical instability when we applied the stiffness reduction parameter to the assembled matrices. To avoid such instabilities we solved the eigenvalue problems without the crack parameter and after reaching convergence to the required tolerance,we then apply the factor to the critical load factor. This problem arises because java was not designed as a number crunching language, fortran was. The results of the present study using the proposed java code compare well with those using analytical or finite element methods using procedural codes mainly fortran codes.

Program Validation: To test accuracy of solution of linear stability analysis program proposed here a few numerical examples with known solutions are solved using the new program and the results are compared as presented in Table 1.The results compare well with those using procedural codes. Appendix1 is attached here to show the developed kclass used for rapid calculation of the proposed parameter α_1.

Application to Crack Problems

Example 1: The first application example is taken from Okamula *et al* [4] who have studied the deflection of pre-cracked beam-columns. Their closed form solution is given as:

$$\frac{\delta}{e} = \frac{1}{\cos\lambda L - \beta\lambda L \sin\lambda L} - 1 \tag{17}$$

$$\frac{\delta}{e} + 1 = \frac{1}{\cos\lambda L - \beta\lambda L \sin\lambda L} \tag{18}$$

In which β is crack parameter, δ is deflection, e is eccentricity of loading and L=effective length.

$$\lambda = \sqrt{\frac{P}{EI}} \tag{19}$$

To use our parameter in equation (18), we let

$$\beta\lambda L \tan\lambda L = \alpha_1 \tag{20}$$

Using equation (20) into equation (18) we have:

$$\frac{\delta}{e} + 1 = \frac{1}{\cos\lambda L(1-\alpha_1)} \tag{21}$$

Equations (21) and (18) are compared in Figure 3 for values of α_1.The comparison is good.

Example 2: The second example to validate our new java program considers published work by Attard [15] who has studied the lateral buckling strength of a rectangular cantilever beam using the finite element method. He has compared his finite element solutions with Michell's closed form solutions for beams with initial curvature as:

$$\frac{P_{cr}L^2}{\sqrt{EI_y GJ}} = \frac{4.0126}{\sqrt{1-\frac{I_{yy}}{I_{xx}}}} \tag{24}$$

Using our proposed stiffness reduction parameter α_1 into equation (24) initially proposed by Michell [15] we have:

$$\frac{P_{cr}L^2}{\sqrt{EI_y GJ}} = \frac{4.0126(1-\alpha_1)}{\sqrt{1-\frac{I_{yy}}{I_{xx}}}} \tag{25}$$

Equation (25) is plotted in Figure 4 for α_1=0; 0.1 and 0.2. When α_1=0, we have the solution given by Attard [15]. For α_1=0.1, we have the pre-cracked analytical solution by Jiki [9]. Our java program reported here gives the cantilever parameter as 4.0125462275 after ten iterations. We see that the program gives satisfactory solutions for elastic stability solutions considered herein.

Table 1: Comparison of critical load factors for axially loaded columns

Column model	Euler [10]	Hartz [11]	Present
Factor	k	k	k
Pin-Pin	1.0	1.005(0.5%)	1.0055(.55%)
Fixed-Fixed	4.0	4.052(1.3%)	4.052(1.3%)
Fixed-Free	1.0	1.055(5.5%)	1.005(0.5%)
Fixed-Free	$Đ^2/(0.699L)^2$		$Đ^2/(0.710L)^2$(1.6%)

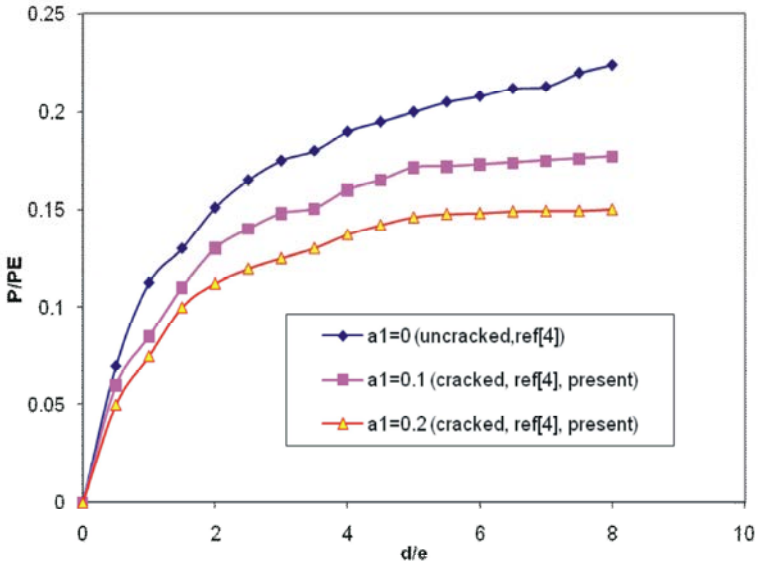

Fig. 3: The deflection of a cracked colume subjected to an eccentric compression load.

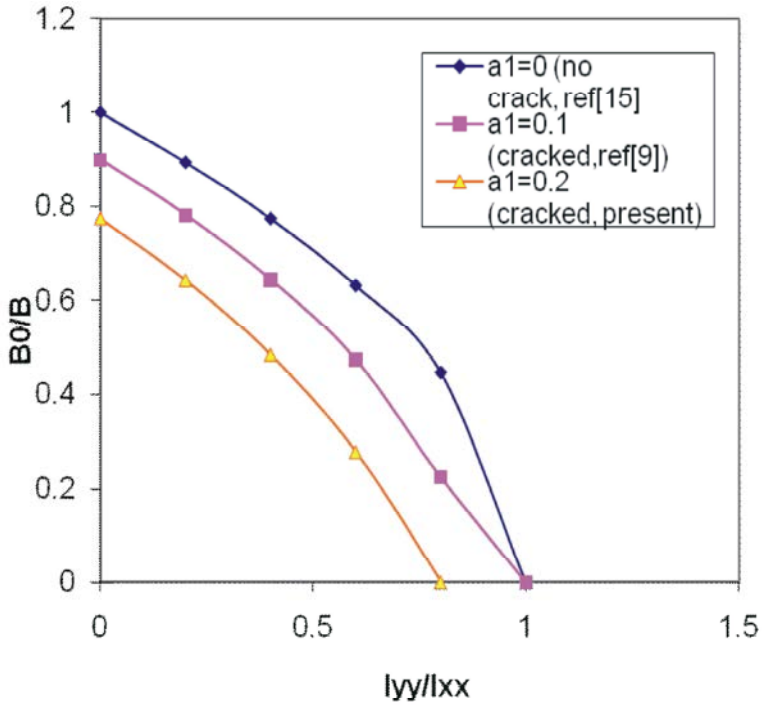

Fig. 4: Reduced lateral buckling strengths of pre-cracked beams using modified Michell's formula[15]

Example 3: The third example considers the effect of pre-crack on end shortening of eccentrically loaded compression members. Tsai [13] has studied the force-displacement relationship for compression members with initial curvatures and those with eccentricity of loading. He came to the conclusion that force displacement relationship for the two cases considered is nonlinear and is of the form [13]:

$$\frac{\Delta L}{r^2} = \frac{P}{P_{cr}}\pi^2\left[1 + C\left(\frac{e}{r}\right)^2\right] \tag{26}$$

In which the nonlinearity is contained in C. For the present model the column is eccentrically loaded to open up the crack. In this case the parameter C is defined as:

Table 2: Effect of pre-crack on end shortening $\Delta L/r^2$ of pre-cracked columns for e/r=1.0, α_1=0.1, 0.2.

P/P_{cr}, (e/r)=1.0 (1)	$\Delta L/r^2$, α_1=0. (2)(ref 13)	α_1=0.1 (3)(present)	α_1=0.2 (4)(present)	%difference [(2)-(3)]	% difference [(2)-(4)]
0.0	0.0	0.0	0.0	0.0	0.0
0.2	2.226	2.204	2.181	1.00	2.0
0.4	5.736	5.677	5.620	1.30	2.0
0.6	14.946	14.795	14.643	1.10	2.0
0.7	28.720	28.429	28.138	1.o1	2.0
0.75	43.441	43.000	42.561	1.00	2.0

$$C_e = \frac{1}{2}.\frac{1-\frac{(\sin\pi\ p)}{\pi\ p}}{1+\cos\pi\ p} \tag{27}$$

Introduction of our proposed parameter into equation (26) we have:

$$\frac{\Delta L}{r^2} = \frac{P}{P_{cr}}\pi^2(1-\alpha_1)\left[1+C_e\left(\frac{e}{r}\right)^2\right] \tag{28}$$

Equation (28) shows that the presence of pre-crack amplifies axial shortening or increases axial compressibility. Effects of pre-crack on end shortening $\frac{\Delta L}{r^2}$ are calculated using equation (28) for values of and $\left(\frac{e}{r}\right)=1$, α_1=0.1,0.2 and are presented in Table 2. It can be seen from the table that the effect of pre-crack on end shortening increases with increase in α_1. This is shown in columns 4 and 6 of the table in percentages. T he effect is remarkable.

Example 4: Calculation of the stiffness reduction parameter α_1 using equation (12) and the java code presented here in appendix 11.

Data: c1=20mm(initial crack length), cb=25 mm (crack length at buckling of column), h=60 mm (width of column), m =bending moment

When crack length is 20 mm, mb=buckling bending moment. When these data are loaded into the java code in appendix 11, the program returns the value of α_1 (a1 in the java code) =0.3185 which compares with manual calculation=0.32. Then the reserve strength due to crack is EI(1-α_1)=EI(1-0.32)=0.68 or 68%. This example is hypothetical. It is an assumption to test the program how ever.

CONCLUSIONS

The Present Study Has Shown That:

- The proposed stiffness reduction parameter can be used to study amplified axial compressibility or end shortening in columns and beam-columns. An object-oriented code in java programming language is presented here for rapid calculation of the parameter α_1.
- A finite element code in java has been developed and is documented in Jiki[19] for vibration and stability analyses of discrete structural systems. A java class for rapid calculation of the strength reduction parameter α_1 is attached here in Appendix 1 and has been used to extend the java program documented in Jiki[19].The results shown in tables 1 and 2 were computed using java kclass proposed herein. They compare well with exact manual calculations.
- Advantages of some of the object-oriented programming attributes such as: program extendibility, program reuse and write once and use all are captured in the present work.

REFERENCES

1. Bentham, J.P. and W.T. Koiter, 1973. Asymptotic approximations to crack problems. In methods of analysis and Solutions of crack problems.Ed. G. Sih, Noordhoff .
2. Dimarogonas, A.D. , 1981. Buckling of rings and tubes with longitudinal cracks. Mechanics Research Commentary, pp: 8.
3. Papadopoulos, C.A., 1995. Torsional vibration of rotors with transverse surface cracks. Computers and Structures, 51(6): 713-718.
4. Okamura, H., H.W. Liu and C.S. Chu, 1969. A cracked column under compression. Engineering Fracture Mechanics, 1: 547-564.

5. Anifantis, N. and A.D. Dimarogonas, 1983. Stability of columns with a single crack subjected to follower and vertical loads. International J. solids and Structures, 19(4): 281-291.
6. Gunaris, G.D., C.A. Papadopoulos and A.D. Dimarogonas, 1995. Crack identification by coupled response measurements. Computers and Structures, 58(2): 299-305.
7. Dentsoras, A. and A.D. Dimarogonas, 1983. Resonance controlled fatigue crack propagation. Engineering Fracture Mechanics, 17(4): 381-386.
8. Parker, A.P., 1981. The mechanics of fracture and fatigue. An introduction. E and F.N.Spon Ltd. London.
9. Jiki, P.N., 2007. Buckling analysis of pre-cracked beam-columns by Liapunov's second method. European J. Mechanics. A /Solids. 26: 503-518.
10. Euler, L., 1744. Methodus Ikeniendi linear curves maximi minimive proprietate. Gandentes, Farlum Mchaelem Bosusquet, Lausanne and Geneva.
11. Hartz, B.J., 1965. Matrix formulation of structural stability problems. J. Struct. Div. ASCE. 91(ST6): 141-157.
12. Peabody, A.B. and J.W. Wekezer, 1994. Buckling strength of wood power poles using finite elements J. Struct Engineering. 120(6): 1893-1906.
13. Tsai W.T., 1977. Nonlinear behavior of compression members. J. Struct. Div ASCE. (ST7): 1484-1489.
14. Nikishkov, G.P., 2006. Object oriented design of finite element code in java. J. Computer Modeling in Engng and Sci., 11: 1-10.
15. Attard, M.M., 1986. Lateral buckling analysis of beams by FEM. Computers and Structures, 21: 286-296.
16. Bathe, K.J., 1990. Finite element procedures in engineering analysis.Prentice-Hall Ltd New Delhi India. pp: 610- 613.
17. Paz, M., 1980. Structural dynamics:Theory and applications.Van Nostrand Reinhold Ltd. New York. pp: 176-178.
18. Rajasekaran, S., 1992. Numerical methods in science and engineering. A practical approach. Wheeler Publishing Co Ltd, Allahabad. pp: 168-170.
19. Jiki, P.N., 2009. Usser Manual for a Java Program "mainApplicationCalculate" for vibration and stability analysis of discrete structural systems. Report no 005, Dept of Civil Engineering, University of Agriculture Makurdi, Benue Sate, Nigeria.

Appendix 1: A java class for calculation of parameter α_1

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package parameterCalculation;

import java.io.BufferedReader;
import java.io.InputStreamReader;
public class kclass {

        /**
         * @param args the command line arguments
         */
public static void main(String[] args) {
        // TODO code application logic here
        InputStreamReader sr =new Input tream eader (System.in);
        BufferedReader br =new BufferedReader(sr);
        double c1 =0.0, cb=0.0, h=0.0,m=0.0,mb=0.0;
        String txt =null;
        try{
                System.out.println("Enter the Value of c1 ");
                txt =br.readLine();
                c1=new Double(txt).doubleValue();
                System.out.println("Enter the Value of cb ");
                txt =br.readLine();
                cb=new Double(txt).doubleValue();
                System.out.println("Enter the Value of h ");
                txt =br.readLine();
                h=new Double(txt).doubleValue();
                System.out.println("Enter the Value of m ");
                txt =br.readLine();
                m=new Double(txt).doubleValue();
                System.out.println("Enter the Value of mb ");
                txt =br.readLine();
                mb=new Double(txt).doubleValue();

        }catch(Exception e){
        System.out.println("Error here "+e.getMessage());
        }
        System.out.println("The Answer is "+calculatek( c1, cb, h, m, mb));
        }
        public static double calculatek(double c1, double cb, double h, double
        m, double mb){

        double pi =3.1456;
        double z1 =pi*c1/(2*h);
        double zb =pi*cb/(2*h);
        double q1=1-Math.sin(z1) ;
        double qb=1-Math.sin(zb) ;
        double k1 = 2*h/(pi*c1)*Math.tan(z1);
        double x1 =1*Math.pow(k1, 0.5);
        double b1 =0.199*Math.pow(q1, 4)/Math.cos(z1);
        double y1 =x1*b1;
        double k2 =2*h/(pi*cb)*Math.tan(zb);
        double x2 =1*Math.pow(k2, 0.5);
        double b2 =0.199*Math.pow(qb, 4)/Math.cos(zb);;
        double yb =x2*b2;
        double a1 =m/mb*y1/yb*Math.pow(c1/cb, 0.5);
        return a1;
        }
        }
```