Improving Approximations of the Mohr-Coulomb Failure Envelope for 2-Dimensional States of Stress

Jesse Amundsen

November 2008 (Revised April 2009)

Abstract

In order to more accurately find the Mohr-Coulomb failure envelope, differential calculus is used to find the angle of internal friction and the cohesion of a material. Further methods for solving for the failure envelope are explored using iterative methods in MATLAB. Determining an accurate failure envelope is important to risk assessment, engineering geology and a variety of other disciplines. A comparison of the two primary methods of explicit differential calculus and iterative approximation is discussed. In order to illustrate the explicit method a simple problem containing only two stress samples is completed. The iterative method is revealed to be a more preferable and realistic approach when approximating the failure envelope due to its scalability, speed and acceptable accuracy. Using the automation of the iterative approximation method it is possible to solve for best-fit failure envelopes for a large volume of data which could prove useful when assessing the potential failure of slopes and faults.

Contents

1 Introduction

1.1 Overview and Explanation of the Problem

Given two Mohr's circles, calculate the exact failure envelope to reveal the cohesion and the angle of internal friction of the sampled material. In the following text, a pair of principal stress values (σ_3, σ_1) is referred to simply as a "stress sample". Being able to solve for the exact failure envelope for two stress samples allows for the calculation of a failure envelope for an infinite amount of stress samples. When calculating a failure envelope for more than two stress samples, the failure envelope becomes a "best fit" case where the cohesion (c) and angle of internal friction (ϕ) for every combination of two pairs of stress samples is taken and averaged. For the intents and purposes of this text, all examples will focus on solutions where only two stress samples are taken into account.

Mohr's circles can be displayed in a traditional 2-dimensional Cartesian coordinate system by considering the relationship of σ_3 and σ_1 to the radius of the circle and its centroid. Using the formula for a circle (Equation 1) a formula can be created for each stress sample in terms of their principal stress values. The Mohr Circle will always be on on the x-axis so b will always be equal to 0.

$$(x-a)^2 + (y-b)^2 = r^2 \tag{1}$$

$$r = \frac{1}{2}(\sigma_1 - \sigma_3)$$

$$a = r + \sigma_3$$

Substitute into the equation for a circle and solve for positive values of y to arrive at the equation for a stress sample that represents one Mohr Circle (Equation 2).

$$[x - (\sigma_3 + \frac{1}{2}(\sigma_1 - \sigma_3)]^2 + y^2 = [\frac{1}{2}(\sigma_1 - \sigma_3)]^2$$

$$y = \sqrt{[\frac{1}{2}(\sigma_1 - \sigma_3)]^2 - [x - (\sigma_3 + \frac{1}{2}(\sigma_1 - \sigma_3))]^2} \tag{2}$$

It is much easier to express the equations in terms of σ_3 and σ_1, so when the normal and shear stress along with an angle (σ_n, σ_s and θ) are available they should be rearranged into terms of σ_3 and σ_1 using equations 3 and 4. These equations assume that 2θ is being measured from the centroid of the circle in a counter-clockwise manner.

$$\sigma_3 = \sigma_n - \frac{1}{\sigma_s}\tan 2\theta - \sigma_s \frac{1}{\sin 2\theta} \tag{3}$$

$$\sigma_1 = \sigma_n - \frac{1}{\sigma_s}\tan 2\theta + \sigma_s \frac{1}{\sin 2\theta} \tag{4}$$

1.2 Illustration and a Simplified Example

Shown on Figure 1 is a failure envelope and labels indicating that the y-intercept of the failure envelope is the cohesion of the material. The angle of internal friction is also shown and is identified as the angle formed between a horizontal line and the failure envelope.

Using the aforementioned equation (Equation 1) and two hypothetical stress samples of ($\sigma_3 = 1$, $\sigma_1 = 3$) and ($\sigma_3 = 4$, $\sigma_1 = 8$) two Mohr's circles are constructed.

$$y_1 = \sqrt{(\frac{1}{2}(3-1))^2 - (x - (1 + \frac{1}{2}(3-1)))^2}$$

$$y_1 = \sqrt{1 - (x-2)^2} \tag{5}$$

$$y_2 = \sqrt{(\frac{1}{2}(8-4))^2 - (x - (4 + \frac{1}{2}(8-4)))^2}$$

$$y_2 = \sqrt{4 - (x-6)^2} \tag{6}$$

The relationship of the failure envelope to the two Mohr's circles can be described as a line that is tangent to both of the Mohr's circles (Figure 1). The line is unique given the constraints that

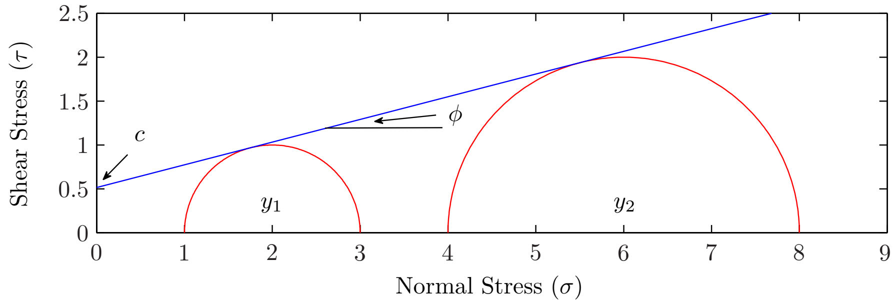

Figure 1: Simplified example identifying that the y-intercept of the failure envelope is the cohesion (c) of the material and that the angle of internal friction (ϕ) is the angle formed between the line tangent to both Mohr Circles and a horizontal line.

- Any circle with a larger differential stress be located further along the x-axis.
- Only values in the first quadrant are considered.

Being able to confidently solve for an accurate failure envelope is important because it can reveal how cohesive the sampled material is, how much stress it can withstand before failure, and other properties of the material being examined. Each of the previously mentioned pieces of data are important in risk assessment, engineering geology and a variety of other disciplines.

2 Explicit Solution to the Coulomb Failure Envelope

2.1 Method Accompanied by an Example

Finding the failure envelope can be done using differential calculus once two half-circles are constructed since they are continuous functions. The method can be time consuming and difficult and as a result poses formidable challenges to automation in a computational environment such as MATLAB. Other methods that were developed with automation in mind are discussed in later sections.

To find the line that is tangent to both half-circles, at least one point of tangency on either half-circle must be obtained. Typically, the stress pairs will be under certain constraints that guarantee that the first circle (y_1) must be smaller than the second (y_2) (a greater differential stress) and that neither σ_3 or σ_1 from y_2 can be larger than σ_3 or σ_1 from y_1. Due to these constraints, it is usually simpler to solve for the point of tangency on the smaller circle. The derivative of the function that represents the half-circle that was chosen will provide the instantaneous slope at any given value of x, thus it is unnecessary to calculate a corresponding point on the second circle and solve for the slope.

In order to setup the problem, two theoretical coordinate sets are identified as $(x_f,\ f(x_f))$ and $(x_g,\ g(x_g))$. These two points represent the points of tangency on the two half-circles. To solve for the failure envelope explicitly, the equation for each circle must first be differentiated in terms of its corresponding point of tangency.

$$f(x) = y_1$$

$$g(x) = y_2$$

$$f(x) = \sqrt{1 - (x-2)^2}$$

$$f'(x_f) = \frac{-(x_f - 2)}{\sqrt{-x_f^2 + 4x_f - 3}}$$

$$g(x) = \sqrt{4 - (x-6)^2}$$

$$g'(x_g) = \frac{-(x_g - 6)}{\sqrt{-x_g^2 + 12x_g - 32}}$$

The standard form for the derivative of the equation of the half-circle can be expressed as seen in Equation 7.

$$h(x) = \sqrt{c - (x-d)^2}$$

$$h'(x) = \frac{-(x-d)}{\sqrt{-x^2 + 2dx - (d^2 - c)}} \tag{7}$$

The first derivative of a function describes the slope of the function, so to find where the slope of each half-circle is equivalent the two equations are set equal to eachother and solved for x_f and x_g. This is most easily done using an algebraic solver such as the `solve()` function available on the TI-89 calculator or the `solve()` function available in MATLAB's symbolic math toolbox.

$$f'(x_f) = g'(x_g)$$

$$\frac{-(x_f - 2)}{\sqrt{-x_f^2 + 4x_f - 3}} = \frac{-(x_g - 6)}{\sqrt{-x_g^2 + 12x_g - 32}}$$

$$x_f = \frac{(x_g - 2)}{2}$$

$$x_f = \frac{-(x_g - 10)}{2}$$

$$x_g = 2(x_f + 1)$$

$$x_g = -2(x_f - 5)$$

Identifying that the slope of the failure envelope will be equal to either $f'(x_f)$ or $g'(x_g)$, choose either $f'(x_f)$ or $g'(x_g)$ and set it equal to the slope m. In this example, $f'(x_f)$ was chosen.

$$m = \frac{g(x_g) - f(x_f)}{x_g - x_f}$$

$$f'(x_f) = m$$

$$\frac{-(x_f - 2)}{\sqrt{-x_f^2 + 4x_f - 3}} = \frac{g(x_g) - f(x_f)}{x_g - x_f}$$

At this point another decision has to be made. Either all x_f have to be replaced in terms of x_g or vice versa. This example replaces all x_g in terms of x_f. Either value for x_f or x_g can be substituted and the opposite solved for.

$$\frac{-(x_f - 2)}{\sqrt{-x_f^2 + 4x_f - 3}} = \frac{\sqrt{4 - (x_g - 6)^2} - \sqrt{1 - (x_f - 2)^2}}{2(x_f + 1) - x_f}$$

$$\frac{-(x_f-2)}{\sqrt{-x_f^2+4x_f-3}} = \frac{\sqrt{4-((2(x_f+1))-6)^2}-\sqrt{1-(x_f-2)^2}}{2(x_f+1)-x_f}$$

$$x_f = \frac{7}{4}$$

The value of x_f represents the x-value on the half-circle $f(x_f)$ where the slope of the line tangent to it at that point is also tangent to the half circle $g(x_g)$. Knowing the point of tangency on one of the half-circles is enough to calculate the failure envelope since $f'(x_f)$ can be solved at x_f to find the slope (m_c) of the tangent line and (since x_f is an x-value) $f(x_f)$ can be solved at x_f to find a corresponding y-value and thus provide an (x, y) coordinate. To calculate the failure envelope, find the slope (m_c) of the line using the value of $f'(x_f)$ at x_f.

$$m_c = f'(\frac{7}{4}) = \frac{-(\frac{7}{4}-2)}{\sqrt{-(\frac{7}{4})^2+4\frac{7}{4}-3}} = \frac{\sqrt{15}}{15}$$

Next, find a point along the line. Use the x-value (x_f) to find a corresponding y-value ($f(x_f)$).

$$f(\frac{7}{4}) = \sqrt{1-(\frac{7}{4}-2)^2} = \frac{\sqrt{15}}{4}$$

$$(x_f, f(x_f)) = (\frac{7}{4}, \frac{\sqrt{15}}{4})$$

With the slope of the failure envelope and a point located on the failure envelope, use the point-slope form for a linear equation to find the cohesion (y-intercept) and the equation for the failure envelope.

$$y_c - f(x_f) = m_c(x_c - x_f)$$

$$y_c - \frac{\sqrt{15}}{4} = \frac{\sqrt{15}}{15}(x_c - \frac{7}{4})$$

To find the cohesion (c) find the y-intercept of the equation.

$$y_c = \frac{\sqrt{15}}{15}(x_c - \frac{7}{4}) + \frac{\sqrt{15}}{4}$$

$$y_c = \frac{\sqrt{15}}{15}x_c + \frac{-\sqrt{15}}{15}\frac{7}{4} + \frac{\sqrt{15}}{4}$$

$$c = \frac{-\sqrt{15}}{15}\frac{7}{4} + \frac{\sqrt{15}}{4} = \frac{2\sqrt{15}}{15}$$

To find the angle of internal friction (ϕ) trigonometry is used (Figure 2). In the example provided, the line is chosen as one that started at the y-intercept $(0, c)$ and extended out 2 units (Figure 2).

$$\sin\phi = \frac{f(2)-c}{2} = \frac{\frac{4\sqrt{15}}{15} - \frac{2\sqrt{15}}{15}}{2}$$

$$\phi = \arcsin\left(\frac{2\frac{\sqrt{15}}{15}}{2}\right) = 14.9632^\circ$$

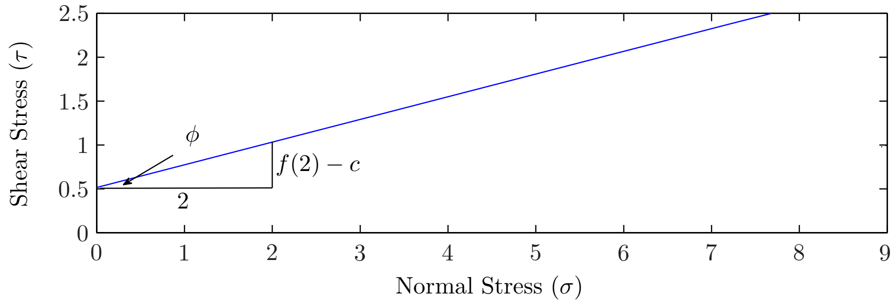

Figure 2: Illustration indicating that the angle of internal friction (ϕ) can be found using simple trigonometry. A vertical line is drawn from the y-intercept (c) out 2 units and then the value of the function representing the internal friction is evaluated at $x = 2$. It is important to note that the cohesion is subtracted from the value of the function at the chosen x-value.

2.2 Example MATLAB Function

To solve for the failure envelope explicitly as described above, the symbolic toolbox plugin for MATLAB is necessary. Without this toolbox, the `solve()` and `subs()` functions will not be available. The method works exactly as described previously. Note that the provided function is not a fully working example, it only shows the code that actually solves for the failure envelope.

```
% Use the format for a circle
%  (x-a)^2+(y-b)^2=r^2
%  b will always be zero, a will
%  be s3 + .5(s1-s3)
a_a = s3a + (1/2)*(s1a - s3a);
a_b = s3b + (1/2)*(s1b - s3b);

%  r will be .5(s1-s3)
r_a = (1/2)*(s1a - s3a);
r_b = (1/2)*(s1b - s3b);

%  put the equation together
syms xf;
syms xg;

% equation f(x)
fxf = sqrt(r_a^2 - (xf - a_a)^2);

% equation g(x)
gxg = sqrt(r_b^2 - (xg - a_b)^2);

% Differentiate each equation
%  and solve for x_a (xf) and x_b (xg)
fxfp = diff(fxf);
gxgp = diff(gxg);
[xge] = solve(fxfp - gxgp, xg);

% fxfp = (gxg - fxf)/(xg - xf);
gxg2 = subs(gxg, xg, xge(1));
```

```
% solve for xf; pt = point of tangency
%  Uses solve() function in symbolic math toolbox
pt = solve(fxfp - (gxg2 - fxf)/(xge(1) - xf),xf);

% Calculate slope of line:
%  put pt into xf in fxfp
m = subs(fxfp, xf, pt(1));

%  point of tangency: at pt in fxf what is
%   intercept
pty = subs(fxf, xf, pt(1));

%  create equation of line using
%   (pt,pty)=(h,k) with m as slope
%   in form y-k=m(x-h)
syms x;
y = m * (x - pt) + pty;

%  calculate cohesion c = (y-intercept)
c = simplify(subs(y, x, 0));

%  calculate phi (angle of internal friction)
phical = subs(y, x, 2);
phi = asin((phical-c)/2)*180/pi;
```

2.3 Brief Summary and Other Notes

Explicitly solving for the failure envelope is difficult, time consuming and the solution provided above for MATLAB is very unreliable. This is due to the limitations of the symbolic math toolbox. Other methods for solving the complicated equation to find the value of x_f or x_g involve estimations by graphing or using a separate function that estimates the intersections of two functions. When using graphing estimations or intersect estimations, the accuracy of the result depends on the accuracy of the x-value and y-value vectors; the accuracy of the computed failure envelope decreases as the numeric size of each stress pair grows. This poses an issue for computing time and storage in memory since, to obtain satisfactory accuracy, using intervals smaller than .01 is often necessary. Since these vectors are duplicated, calculated and modified at multiple times in the method, memory quickly runs out with problems that have stress pairs that exceed 60-80 units in differential stress.

3 An Alternative Solution to the Coulomb Failure Envelope

3.1 Recognizing a Key Relationship

The following method was developed by Elige Grant of CERI (Center for Earthquake Research and Information) at the University of Memphis. This algorithm is much quicker than solving the problem explicitly and maintains a more than satisfactory amount of accuracy. It also has no dependencies on the symbolic math toolbox. While the answers are not exact in the sense that the explicit method guarantees, they are still extremely accurate and consequently the algorithm is a viable and recommended alternative to the explicit solution in nearly all situations.

The key to this method is to recognize that the angle θ in each half-circle will always be equal when a line is drawn from the centroid of the circle to the point of tangency on each half-circle. A

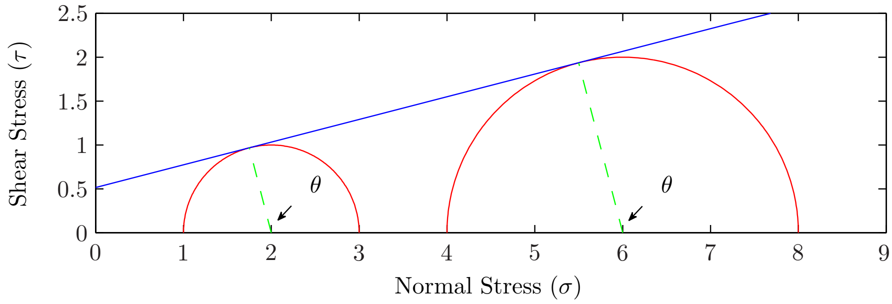

Figure 3: Identifying that the angle θ will be identical on each circle when the line tangent to both half-circles is found allows for the use of an iterative algorithm that tests large numbers of possible angles to find the highest cohesion value.

diagram shows this relationship based on the two stress pairs that were used to demonstrate the technique of explicitly solving for the failure envelope (Figure 3).

Since the angle θ will be equal at the correct points of tangency, it is possible to approximate the angle of θ where the points of tangency create the most optimal failure envelope. Since constraints have been put in place to force it so that the second stress pair will always have a larger differential stress and have both σ_3 and σ_1 values that are greater than those in the first stress pair, the y-intercept of the set of points of tangency that are on both half-circles for the same angle θ will be greatest when the most optimal pair is found.

Using this knowledge, it is then possible to divide the circle into increments and test each dividing point to find its y-intercept. In the method provided as an example, the `find_theta_range()` function divides the range it is given into 100 sections and returns the point-boundaries for the most optimal range (the range that returns the highest y-intercept value). The accuracy of the approximation will increase as the `find_theta_range()` function is called more often. Once the maximum y-intercept is found, the points that yielded that y-intercept and the y-intercept itself are enough to determine the slope of the failure envelope.

3.2 Implementation of Algorithm in MATLAB

The code provided below was written by Elige Grant and modified by Jesse Amundsen. It is not meant to be used in its current state, as part of the functionality has been removed for simplicity. The parts of the algorithm that remain are intended to illustrate the method.

```
function coulomb_approx

%   Radius of Half-Circles
r1 = (1/2)*(s1a-s3a);
r2 = (1/2)*(s1b-s3b);

%   Center of Half-Circles
c1 = s3a + r1;
c2 = s3b + r2;

%   Initialize theta range between 0 and 180 degrees
theta1=0;
theta2=180;
```

```
%   Zero in on theta over 10 iterations
for i = 1 : 10
    %   Function "find_theta_range" is located
    %   at the bottom of this script
    [theta1,theta2] =
     find_theta_range(theta1, theta2, c1, c2, r1, r2);
end

%   Take average of final theta1 and theta2
theta = (theta1+theta2)/2;

%   Define temp_theta, x1, x2, y1, and y2
temp_theta = theta*pi/180;
x1 = c1 + r1*cos(temp_theta);
y1 = r1*sin(temp_theta);
x2 = c2 + r2*cos(temp_theta);
y2 = r2*sin(temp_theta);

%   Find temporary y-int vector based on current theta
slope = (y2 - y1)/(x2 - x1);
y_int = y1 - x1*slope;


function [new_theta1,new_theta2] = ...
find_theta_range(theta1, theta2, c1, c2, r1, r2)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   FIND BIGGEST Y-INT VALUE METHOD   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%   Define increment in theta
delta_theta = (theta2-theta1)/100;

% Define max_y_int to be very large negative number
max_y_int = -1e6;

%   Loop over all angles in range
%   Exit loop when y-int values start to decrease - it means
%   we have already found the best possible theta for
%   this range/increment.
for i = 1 : 101
    %   Define temp_theta, x1, x2, y1, and y2
    temp_theta = (theta1+(i-1)*delta_theta)*pi/180;
    x1 = c1 + r1*cos(temp_theta);
    y1 = r1*sin(temp_theta);
    x2 = c2 + r2*cos(temp_theta);
    y2 = r2*sin(temp_theta);

    %   Find temporary y-int vector based on current theta
    temp_y_int = y1 - x1*(y2 - y1)/(x2 - x1);
```

```
    if max_y_int < temp_y_int
max_y_int = temp_y_int;
new_theta1=(theta1+(i-2)*delta_theta);
new_theta2=(theta1+(i)*delta_theta);
    else
break;
    end
end

return
```

4 Comparison of the Two Methods

Solving for the failure envelope explicitly is a much more mathematically elegant solution while the iterative method of approximating the failure envelope is far more realistic from an algorithmic perspective. In nearly all cases, the approximation made by the iterative method will fall well within acceptable margins of error. If the calculation must be done by hand, the explicit method is favored since it is unfeasible for a human to perform the iterative method. Yet, when we take into account that most of the uses for calculating the failure envelope will involve many stress samples (more than 2), it becomes clear that a computational solution is necessary.

While a working example of the explicit solution written in MATLAB was provided, the function is plagued with issues and errors. The dependency on the symbolic math toolbox is unnecessary and the `solve()` function within the toolbox often reports incorrect non-real answers. Explicitly solving the problem also takes a longer time than the iterative method, which is equally as fast for extremely large stress values. The clearest advantages to the iterative approximation method are its speed and its consistency.

The explicit method may lose to the iterative approximation method in most categories, but it is mathematically unbreakable in the sense that no matter what the stress values are, it will always find a tangent line. To guarantee that the tangent line being solved for is the failure envelope, a series of constraints is necessary no matter which method is chosen. Constraints are necessary for any deployable solution that would solve many stress samples for an optimal failure envelope.

Due to the reality that the most optimal failure envelope will be found when larger amounts of stress samples are considered, the scalability of the iterative approximation method makes it a clear choice. Considering that the explicit method has the potential to cause the computing system to run out of memory or return an incorrect x-value for the point of tangency the only realistic choice for solving for the failure envelope is the iterative approximation method.