Scott Moura
SID 15905638
ME C124
Professor Zohdi
Due: April 21, 2005

Project #1 Report

Given an object under various loads, we are to determine the maximum values of those loads that satisfy the Von Mises criteria. In order to perform this mathematically rigorous task, we shall employ the use of numerical methods. In particular, we will be using Newton's method to converge onto the critical values for the normalized cost function derived from the Von Mises criteria. Newton's method requires that we take derivatives of the function, which is a timely and monotonous task. However, MATLAB's symbolic toolbox can be used to find these derivatives easily. Yet, to calculate the Hessian we must use a numerical scheme. By picking various initial guesses for those loads, we obtain different final values. The results follow below...

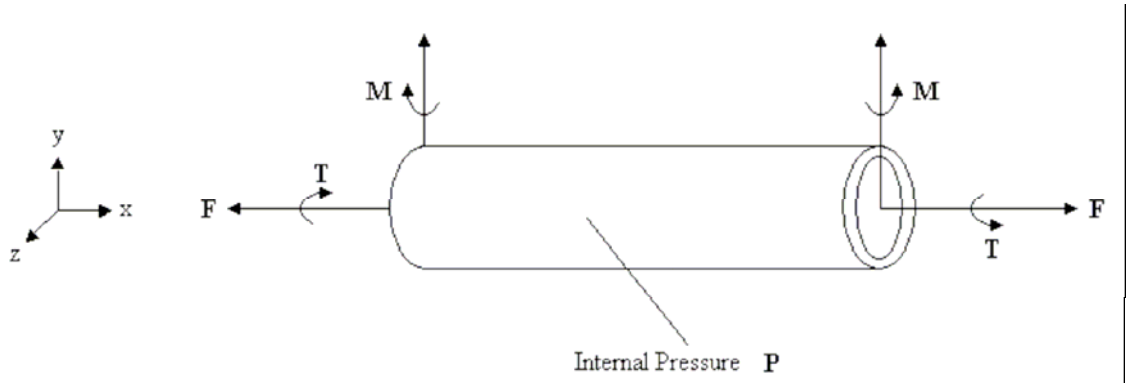
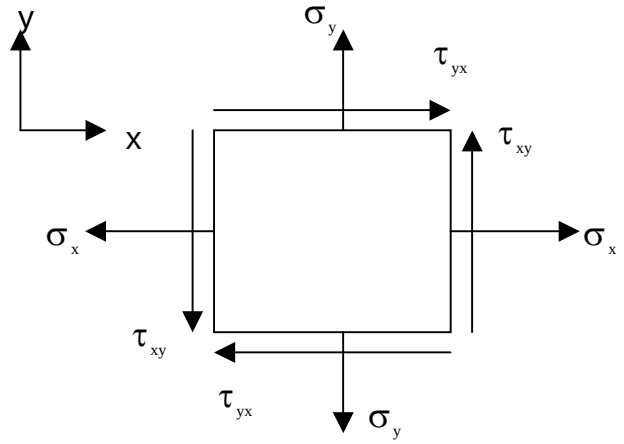


Figure 1: Schematic of pressured vessel with external loads

A pressurized vessel with no end caps is subject to an axial force, a torque, and a pure bending moment, as shown in Figure 1 above. Since there are no end caps there is only hoop stress due to the internal pressure. As a result, we will examine an infinitesimally small element of the vessel, aligned with the x-y plane, shown in Figure 2.



$$\sigma_x = \sigma_{bend} + \sigma_{axial}$$

$$\sigma_y = \sigma_{hoop}$$

$$\tau_{xy} = \tau_{yx} = \tau_{torsion}$$

Figure 2: Infinitesimally small stress element

Stress due to various loads can be calculated by the following:

Bending: $\sigma_{bend} = \frac{M \cdot c}{I}$, where $I \approx \pi r^3 t$

Axial Force: $\sigma_{axial} = \frac{F}{A}$, where $A = 2\pi rt$

Hoop Stress: $\sigma_{hoop} = \frac{P \cdot r}{t}$ (due to internal pressure)

Torsion: $\tau_{tosion} = \frac{T \cdot r}{J}$, where $J = 2I$

Plugging in we obtain expressions for each stress component:

$$\sigma_x = \frac{Mr}{\pi r^3 t} + \frac{F}{2\pi rt} \qquad \sigma_y = \frac{Pr}{t} \qquad \sigma_z = 0$$

$$\tau_{xy} = \frac{Tr}{2\pi r^3 t} \qquad \tau_{yz} = 0 \qquad \tau_{zx} = 0$$

The stress matrix can then be written as:

$$[\sigma] = \begin{bmatrix} \sigma_x & \tau_{xy} & 0 \\ \tau_{yx} & \sigma_y & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

With these values we can set up the Von Mises criteria for the critical stress value of $\sigma_s = 724\text{MPa}$

$$\Phi = (\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6(\tau_{xy}^{\ 2} + \tau_{yz}^{\ 2} + \tau_{zx}^{\ 2}) = 2\sigma_s^{\ 2}$$

Inserting our values for the stress components, we obtain

$$\Phi = \frac{4M^2 + F^2r^2 + 4\pi^2P^2r^6 + 4FMr - 2\pi F\,Pr^4 - 4\pi M\,Pr^3 + 3T^2}{2\pi^2 r^4 t^2}$$

We can now write down the normalized cost function

$$\Pi = \left(\frac{\Phi - 2\sigma_s^{\ 2}}{2\sigma_s^{\ 2}} \right)^2 = \left(\frac{\Phi}{2\sigma_s^{\ 2}} - 1 \right)^2$$

Let $\Phi = 2\phi$, so $\Pi = \left(\frac{\phi}{\sigma_s^{\,2}} - 1 \right)^2$

With this calculation, we can now program a Newton's Method minimizing scheme to find the values of F, M, T, P, r that satisfy the Von Mises criteria. From class, we know that Newton's Method takes on the following form:

$$\{\Lambda^{i+1}\} = \{\Lambda^{i}\} - [\nabla^2 \Pi(\Lambda^{i})]^{-1} \{\nabla \Pi(\Lambda^{i})\}$$

where $\Lambda = \{F, M, T, P, r\}$. For simplicity we will assume that the thickness is fixed at t = 0.01m. The gradient, $\{\nabla \Pi\}$, can be calculated symbolically by MATLAB. However, it is also done by hand in Appendix B, for reference. In order to calculate the Hessian we will use a numerical method to find the derivative of the gradient and form the Hessian one column at a time. This method takes the following form:

$$[\nabla^2 \Pi] = \left[\frac{\partial}{\partial F}\{\nabla \Pi\} \quad \frac{\partial}{\partial M}\{\nabla \Pi\} \quad \frac{\partial}{\partial T}\{\nabla \Pi\} \quad \frac{\partial}{\partial P}\{\nabla \Pi\} \quad \frac{\partial}{\partial r}\{\nabla \Pi\} \right]$$

$$\text{where, } \frac{\partial}{\partial F}\{\nabla \Pi\} = \frac{[\nabla \Pi]_{F+\delta,M,T,P,r} - [\nabla \Pi]_{F,M,T,P,r}}{\delta \cdot F}$$

$$\frac{\partial}{\partial M}\{\nabla \Pi\} = \frac{[\nabla \Pi]_{F,M+\delta,T,P,r} - [\nabla \Pi]_{F,M,T,P,r}}{\delta \cdot M}$$

$$\vdots$$

With this, a simple MATLAB script file can be written to calculate the maximum values for F, M, T, P, r that satisfy the Von Mises criteria. It is found in Appendix A. The following initial guesses for Λ are used with a given tolerance of 1×10^{-8}. The values are in terms of MN, MN-m, and m.

Case 1:	{0.2	0.2	1	1	0.2}
Case 2:	{0.001	0.004	1.3	19	0.3}
Case 3:	{0.0005	0.03	3.52	21.4	0.4}
Case 4:	{0.003	0.01	8	24.94	0.5}

MATLAB returns the following results:

Case	1	2	3	4
F (MN)	1.8263	0.0239	0.0069	0.0127
M (MN-m)	0.0420	0.0007	0.0194	0.0019
T (MN-m)	1.0214	1.3548	2.4347	3.7929
P (MN/m^2)	1.0212	19.8017	14.8039	11.8262
r (m)	0.2000	0.3000	0.4000	0.5000
t (m)	0.0100	0.0100	0.0100	0.100

It is interesting to note that the values for r did not deviate at all from the initial guesses. This seems to suggest that the radius of the pressured vessel does not play a significant part in reaching the critical value that satisfies the Von Mises criteria. Each initial guess for $\{\Lambda\}$ also arrives at different final values. This implies that our cost function has many local critical points. Yet it would probably be most useful to find the absolute minimum instead of local minima. Although it was suggested that one set of initial conditions would not converge, all four cases appeared to converge nicely, not taking more than six iterations.

A plot of the 2-norm for final values of $\{\nabla\Lambda\}$ versus the number of iterations can be produced in order to get an idea of how long it takes to converge. The plots for each case are shown in Figure 3.

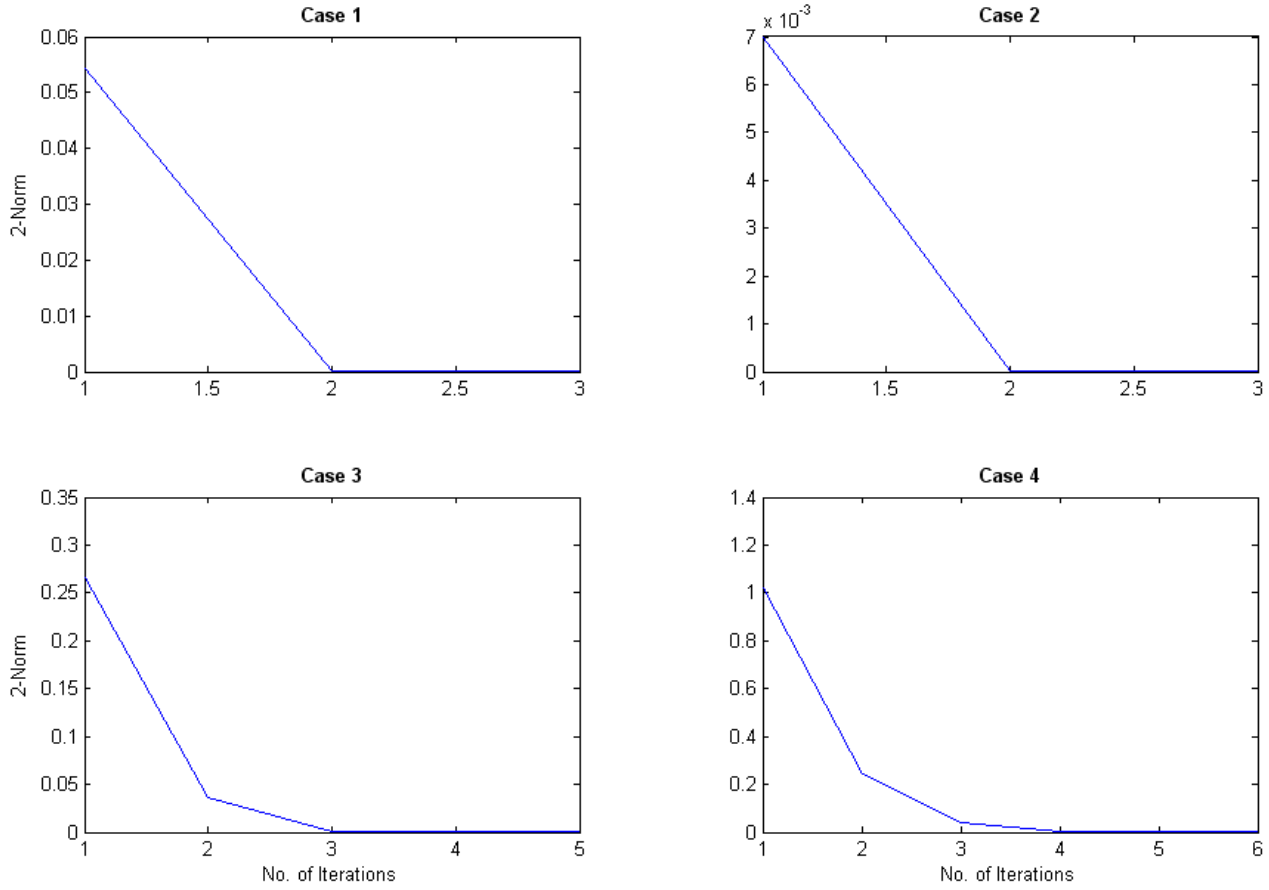


Figure 3: 2-Norm versus No. of Iterations for four different cases

Figure 3 above shows that the first two cases converged very quickly, while Cases 3 and 4 need a little bit more time. Nonetheless, Newton's method appears to be a very quick way to converge to a minimum. However, there is no guarantee that these local minima are the absolute minima. In this case, you would have to make a very lucky initial guess in order to achieve an absolute critical point. Thus, we can conclude that Newton's method is very quick and effective when you can easily take derivatives of the objective function and you have a good idea of where the absolute minimum is. If these two facts were not true, then a more robust method would be necessary.

```
% Project #1
% newton.m

% ------------------- %
% Scott Moura         %
% SID 15905638        %
% ME C124             %
% Prof. Zohdi         %
% Due: April 21, 2005 %
% ------------------- %

clear all

% Create symbolic variables
syms F M T P r t sigma_s
variables = {F,M,T,P,r,t,sigma_s};

% Calculate stress components
sigma_x = (M * r) / (pi * r^3 * t) + F / (2 * pi * r * t);
sigma_y = P * r / t;
sigma_z = 0;
tau_xy = (T * r) / (2 * pi * r^3 * t);
tau_yz = 0;
tau_zx = 0;

% Set up the von Mises stress criteria for failure, f = 2*sigma_s^2
f = (sigma_x - sigma_y)^2 + (sigma_y - sigma_z)^2 + (sigma_z - sigma_x)^2 + 6 * (tau_xy^2 ↙
 + tau_yz^2 + tau_zx^2);
pretty(simplify(collect(expand(f))))

% Let f = 2 * phi
phi = 0.5 * f;

% Define the Cost Function
Pi = (phi / sigma_s^2 - 1)^2;

% Calculate the gradient vector
Pi_F = diff(Pi,F);
Pi_M = diff(Pi,M);
Pi_T = diff(Pi,T);
Pi_P = diff(Pi,P);
Pi_r = diff(Pi,r);

Pi_grad_sym = [Pi_F; Pi_M; Pi_T; Pi_P; Pi_r];

% Prompt User for initial guesses of Lambda
Lambda_0(1) = input('What is the initial value of F, in Mega-Newtons? ');
Lambda_0(2) = input('What is the initial value of M, in Mega-Newton-meters? ');
Lambda_0(3) = input('What is the initial value of T, in Mega-Newton-meters? ');
Lambda_0(4) = input('What is the initial value of P, in Mega-Newton/meters? ');
Lambda_0(5) = input('What is the initial value of r, in meters? ');

% Set the Initial guess to be Lambda
Lambda = Lambda_0';

% Set constants for Newton's Method
tol = 1e-8;     % Set tolerance value
```

```
delta = 1e-4;   % delta value used to compute the numerical derivatives
error = 1;      % Initialize error
iterations = 0; % Set the number of iterations

% Loop Newton's method until tolerance is met
while error >= tol

    % Substitute in Lambda_i values into Pi_grad
    Pi_grad = calc(Pi_grad_sym, variables, Lambda);
    Pi_hessian = zeros(5,5);

    % Numerically calculate the Hessian
    for i = 1:5

        %Initialize and set delta vector
        delta_vec = zeros(5,1);
        delta_vec(i) = delta;
        Lambda_delta = Lambda .* (1 + delta_vec);

        % Calculate the Hessian
        Pi_hessian(:,i) = (calc(Pi_grad_sym, variables, Lambda_delta) - calc(Pi_grad_sym, ↙
 variables, Lambda)) / (delta*Lambda(i));
    end

    % Solve for new Lambda using Newton's Method
    Lambda_new = Lambda - inv(Pi_hessian)*Pi_grad;

    % Calculate new Pi_grad and find error
    Pi_grad_new = calc(Pi_grad_sym, variables, Lambda_new);
    error = norm(Pi_grad_new);

    % Find number of iterations to plot against 2-norm
    iterations = iterations + 1;
    norm_plot(iterations) = error;
    iter_plot(iterations) = iterations;

    % Reassign new Lambda to Lambda
    Lambda = Lambda_new;
end

Lambda_Final = Lambda

% Plot the 2-norm against the number of iterations
figure(1)
plot(iter_plot, norm_plot)
title('\bf2-Norm versus No. of Iterations')
xlabel('No. of Iterations')
ylabel('2-Norm')
```

```
function [output] = calc(Pi_grad_sym, variables, Lambda_x)
% Evaluates the gradient of Pi at Lambda_x
% calc.m

[output] = subs(Pi_grad_sym, variables, {Lambda_x(1), Lambda_x(2), Lambda_x(3), Lambda_x( ↙
4), Lambda_x(5), 0.01, 724});
```