8 Dry Friction

Ref: Hibbeler § 8.2, Bedford & Fowler: Statics § 9.1

To move a heavy crate, such as the one illustrated in the example below, you have to push on it. If you push hard enough to overcome *friction*, the crate will slide along the floor. But there is the danger that the crate will tip, especially if you push too high on the side of the crate. To better understand the calculations involved with this dry friction problem, and to see how you can check for tipping, we will solve the "sliding a crate" problem for several cases, varying the push height between 0.2 and 1 m, and the magnitude of the pushing force between 100 and 120 N.

Example: Pushing a Crate

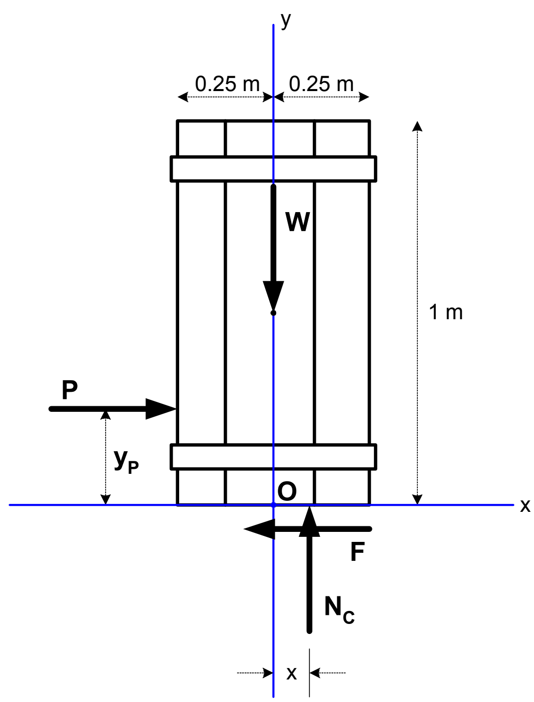
A force, P, is applied to the side of a crate at height y_P from the floor. The crate mass is 35 kg, and is 0.5 m wide and 1.0 m in height. The coefficient of static friction is 0.32. For each of the test cases tabulated below, determine:

a) the magnitude of the friction force, F,

b) the magnitude of the resultant normal force, N_C, and

c) the distance, x, (measured from the origin indicated in the illustration below) at which the resultant normal force acts.

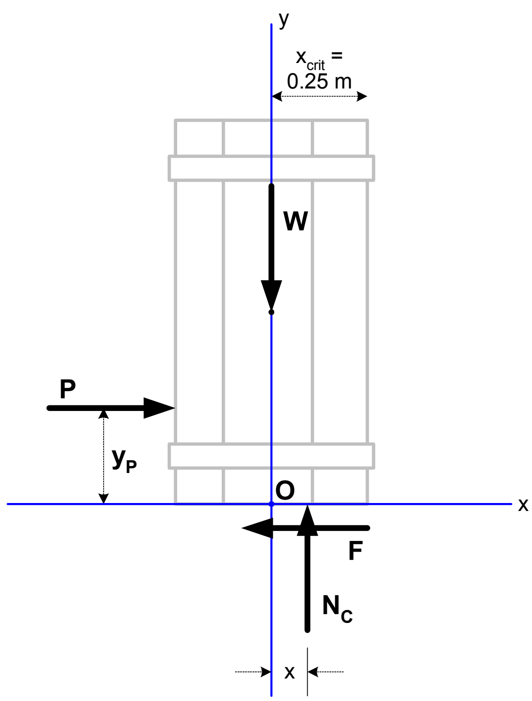
Once these results have been calculated, check for tipping and/or slipping.



Test Cases	P	y_P
1	100 N	0.2 m
2	100 N	0.4 m
3	100 N	0.8 m
4	100 N	1.0 m
5	120 N	0.2 m
6	120 N	0.4 m
7	120 N	0.8 m
8	120 N	1.0 m

Solution, Case 1

First, a free-body diagram is drawn.



To start the solution in MATLAB we begin by assigning the values given in the problem statement to variables.

```
» %Data from the problem statement...
» %      for specific case:
» Case = '1';
» P = 100;                              %Newtons
» y_p = 0.2;                            %meters
»
» %      for all cases:
» M = 35;                               %kg
» x_crit = 0.25;                        %meters
» mu_s = 0.32;
» g = 9.807;                            %meters/s^2
```

Then we calculate the weight of the crate.

```
» W = -M * g                            %Newtons
W =
 -343.2450
```

Next, we use the equilibrium condition that the sum of the forces in the x direction must be zero. (This condition holds if the crate is not sliding or tipping.)

```
» % P + F = 0     so...
» F= -P                                         %Newtons
F =
  -100
```

Then, we use the equilibrium relationship for the y-components of force to determine the resultant normal force, N_C.

```
» % W + N_c = 0     so...
» N_c = -W                                      %Newtons
N_c =
  343.2450
```

The last equilibrium relationship, that the sum of the moments about O must be zero, can be used to determine the location at which N_C acts, called x in the illustration above.

```
» % -P * y_p + N_c * x = 0     so...
» x = P * y_p / N_c                             %meters
x =
    0.0583
```

At this point the equilibrium calculations are complete. We can solve for F_{max}, the maximum force that can be applied before overcoming friction. If the frictional force, F, exceeds F_{max}, then the crate would slip. (As it begins to slip, the equilibrium relationships are no longer valid.)

```
» if (abs(F) > F_max)
   fprintf(['Case ', Case, ' is slipping\n'])
else
   fprintf(['Case ', Case, ' is NOT slipping\n'])
end


Case 1 is NOT slipping
```

Note: The absolute value function was used on F since both the magnitude and direction of F (the minus sign) were determined using the equilibrium relationship. Only the magnitude is used to test for slippage.

Finally, we can check for tipping by testing to see if the calculated x is beyond the edge of the crate (x_{crit}).

```
» if (x > x_crit)
   fprintf(['Case ', Case, ' is tipping\n'])
else
   fprintf(['Case ', Case, ' is NOT tipping\n'])
end


Case 1 is NOT tipping
```

Note: The if statements are certainly not required here – the test could be performed by inspection – but it is convenient to let MATLAB do the checking for each of the eight cases.

From these tests we can see that a push of 100 N applied at a height of 0.2 m will not cause the crate to tip, but it will not cause the crate to move (slip), either.

Solution, Case 2

The only change between Case 1 and Case 2 is the height at which the push is applied. In Case 2 y_P = 0.4 m. This value is change at the beginning of the MATLAB script, and the script is simply re-executed to calculate the values for Case 2. The complete, annotated script solution for Case 2 is shown below.

Annotated MATLAB Script Solution – Case 2

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Pushing a Crate Problem - Case 2                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Data from the problem statement...
%  for specific case:
Case = '1';
P = 100;                                       %Newtons
y_p = 0.4;                                     %meters

%  for all cases:
M = 35;                                        %kg
x_crit = 0.25;                                 %meters
mu_s = 0.32;
g = 9.807;                                     %meters/s^2

%Calculate the weight of the crate.
W = -M .* g;                                   %Newtons
fprintf('\nW = %1.1f N\t\t', W)

%EQUILIBRUM: The sum of the x components of force is zero.
%P + F = 0 so...
F = -P;                                        %Newtons
fprintf('F     = %1.1f N\t\t', F)

%EQUILIBRUM: The sum of the y components of force is zero.
%W + N_c = 0 so...
N_c = -W;                                      %Newtons
fprintf('N_c = %1.1f N\n', N_c)

%EQUILIBRUM: The sum of the moments about 0 is zero.
%-P * y_p + N_c * x = 0 so...
x = P .* y_p ./ N_c;                           %meters
fprintf('x = %1.3f m\t\t', x)

%Checking the Results...
F_max = mu_s .* N_c;                           %Newtons
fprintf('F_max = %1.1f N\n\n', F_max)

if (abs(F) > F_max)
   fprintf(['Case ', Case, ' is slipping\n'])
```

```
else
   fprintf(['Case ', Case, ' is NOT slipping\n'])
end

if (x > x_crit)
   fprintf(['Case ', Case, ' is tipping\n'])
else
   fprintf(['Case ', Case, ' is NOT tipping\n'])
end
```

Summary of Results for All Eight Cases

By repeatedly changing the assigned values for P and y_P in the MATLAB script, the results for all cases can be determined. Those results are summarized here. The value N_C = 343.2 N is not case dependent in this problem.

Test Cases	P	y_P	F	x	Slip?	Tip?
1	100 N	0.2 m	-100 N	0.058 m	No	No
2	100 N	0.4 m	-100 N	0.117 m	No	No
3	100 N	0.8 m	-100 N	0.233 m	No	No
4	100 N	1.0 m	-100 N	0.291 m*	No	Yes
5	120 N	0.2 m	-120 N	0.070 m	Yes	No
6	120 N	0.4 m	-120 N	0.140 m	Yes	No
7	120 N	0.8 m	-120 N	0.280 m*	Yes	Yes
8	120 N	1.0 m	-120 N	0.350 m*	Yes	Yes

* The calculated result for x has no meaning except to indicate that the crate is tipping if x > x_{crit}.

A faster approach to solving the problem is to define P and y_P as vectors and having MATLAB perform all the calculations simultaneously. By default MATLAB performs matrix calculations when using the *, /, and ^ operators. To force MATLAB to perform element-by-element operations place a period in front of the operator. To display the results we can then take advantage of the MATLAB for() statement to loop through the solution vectors and display the results by Case.

Annotated MATLAB Script Solution – All Cases

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                Pushing a Crate Problem  - All Cases                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Data from the problem statement...
%  for specific case:
Case = [ 1 2 3 4 5 6 7 8 ];
P   = [ 100 100 100 100 120 120 120 120]';   %Newtons
y_p = [ 0.2 0.4 0.8 1.0 0.2 0.4 0.8 1.0]';   %meters

%  for all cases:
M = 35;                                      %kg
x_crit = 0.25;                               %meters
mu_s = 0.32;
g = 9.807;                                   %meters/s^2

%Calculate the weight of the crate.
W = -M .* g;                                 %Newtons
```

```
%EQUILIBRUM: The sum of the x components of force is zero.
%P + F = 0 so...
F= -P;                                               %Newtons

%EQUILIBRUM: The sum of the y components of force is zero.
%W + N_c = 0 so...
N_c = -W;                                            %Newtons

%EQUILIBRUM: The sum of the moments about 0 is zero.
%-P * y_p + N_c * x = 0 so...
x = P .* y_p ./ N_c;                                 %meters

%Display and Check the Results...
F_max = mu_s .* N_c;                                 %Newtons
fprintf('F_max = %1.1f N\t', F_max)
fprintf('\nW = %1.1f N\t', W)
fprintf('N_c = %1.1f N\n\n', N_c)

%Display results for each case.
for i = 1 : length(Case)
   fprintf('\nCASE %1.0f:\n',Case(i))
   fprintf('P =  %1.1f N\t\t', P(i))
   fprintf('y_p = %1.2f m\n', y_p(i))
   fprintf('F = %1.1f N\t\t', F(i))
   fprintf('x   = %1.3f m\n', x(i))

   if (abs(F(i)) > F_max)
      fprintf(['Case ', num2str(Case(i)), ' is slipping\n'])
   else
      fprintf(['Case ', num2str(Case), ' is NOT slipping\n'])
   end

   if (x > x_crit)
      fprintf(['Case ', num2str(Case(i)), ' is tipping\n'])
   else
      fprintf(['Case ', num2str(Case(i)), ' is NOT tipping\n'])
   end
end
```