Analysis: Block Holding Analysis

Team: 1402

Project: UMass FMS

Prepared by: Tiernan O'Kane

Date: December 9, 2025

1 Overview and Results

This analysis estimates the force required to separate and conversly press together two interfacing blocks. Each block contains four internal flexible arms which deflect outward when another block's head is inserted. The connection behaves similarly to a press-fit, and for the purpose of hand analysis each arm was idealized as a cantilever beam deflecting under an imposed displacement. The bending stiffness of these arms governs both the holding force and the associated bending stress. The MATLAB script computes the force corresponding to several imposed displacements and checks whether the resulting bending stresses exceed the assumed elastic limit of the material.

Because this geometry operates near the limits of Euler–Bernoulli beam theory—featuring short beams, large local rotations, nonlinear contact, and friction—the hand analysis was reviewed by faculty. It was recommended that ANSYS be used to obtain a more accurate prediction. A nonlinear 4-step frictional simulation was performed, and ANSYS predicted a peak separation force of approximately 19 N. This is higher than the 12.1 N holding force estimated in the hand analysis, which is expected given the simplifying assumptions. The ANSYS force–displacement curve shows an initial rise in force as the arms stick and build elastic energy, followed by a decrease as the arms begin to relax during separation. The resulting stress distribution and predicted factor of safety are included in the figures at the end of this section.

2 MATLAB Results

Displacement Sweep Results

Displacement (m)	Force (N)	Stress (MPa)	Exceeds Elastic Limit	F_{press} (N)
0.000035	15.004	20.63	No	12.125
0.000071	30.308	41.68	Yes	12.125
0.000106	45.483	62.55	Yes	12.125

Maximum Safe Displacement

Quantity	Value
Maximum safe displacement	0.000059 m (0.059 mm)

This displacement corresponds to the limit at which the bending stress reaches the assumed elastic limit of 35 MPa.

3 ANSYS Simulation Results

The ANSYS model used a nonlinear frictional contact simulation with four loading steps to capture arm expansion, static sticking, and sliding separation behavior. The simulation predicted approximately 19 N peak separation force. The following figures show the setup, force–displacement response, stress distribution, and corresponding factor of safety across the block.

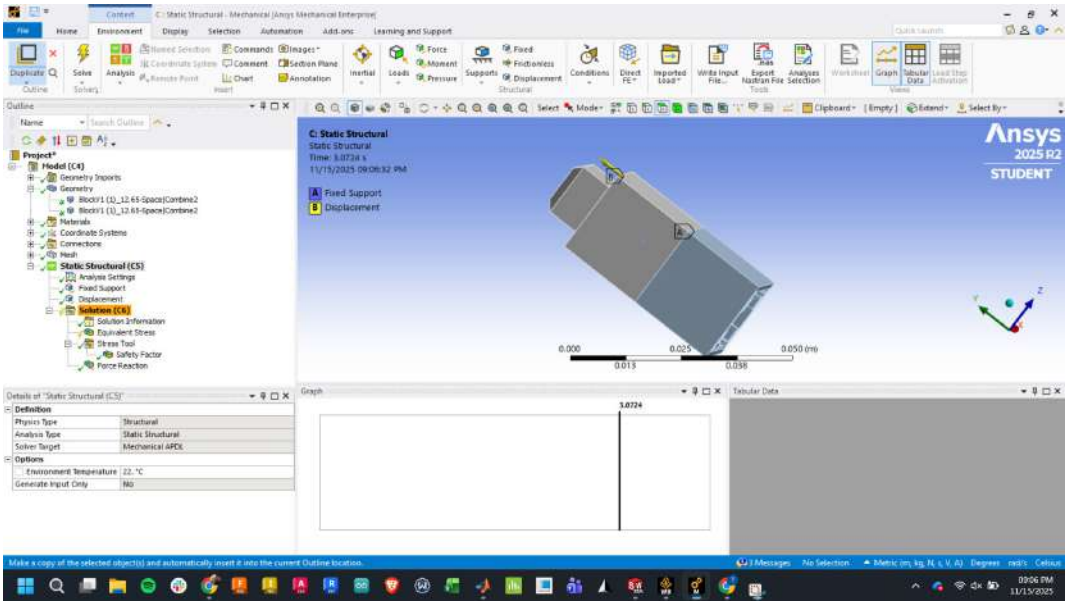


Figure 1: ANSYS simulation setup for block separation.

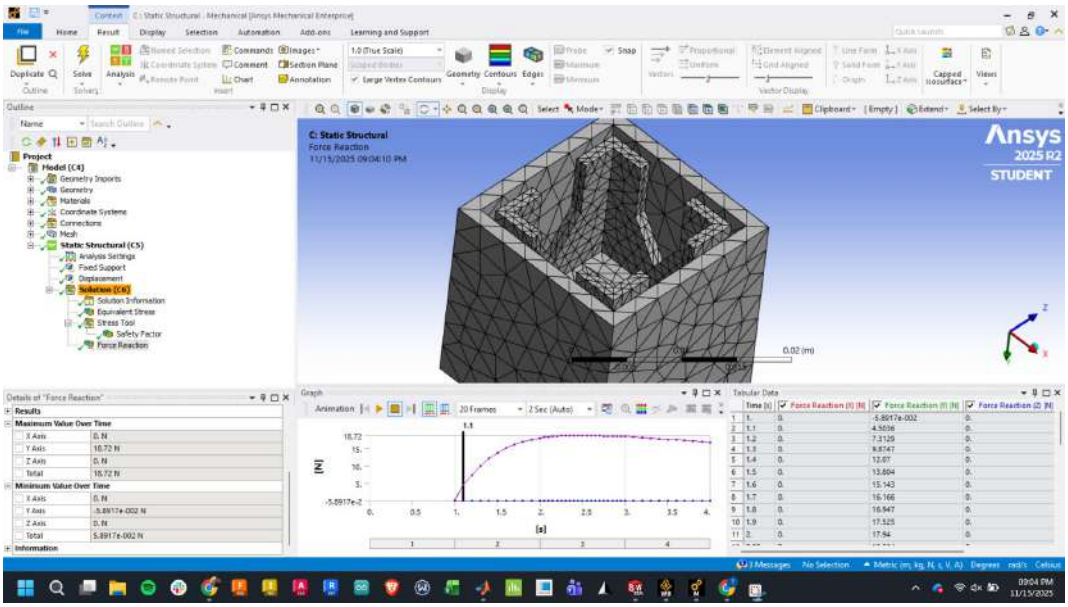


Figure 2: Force vs. displacement curve from nonlinear frictional simulation.

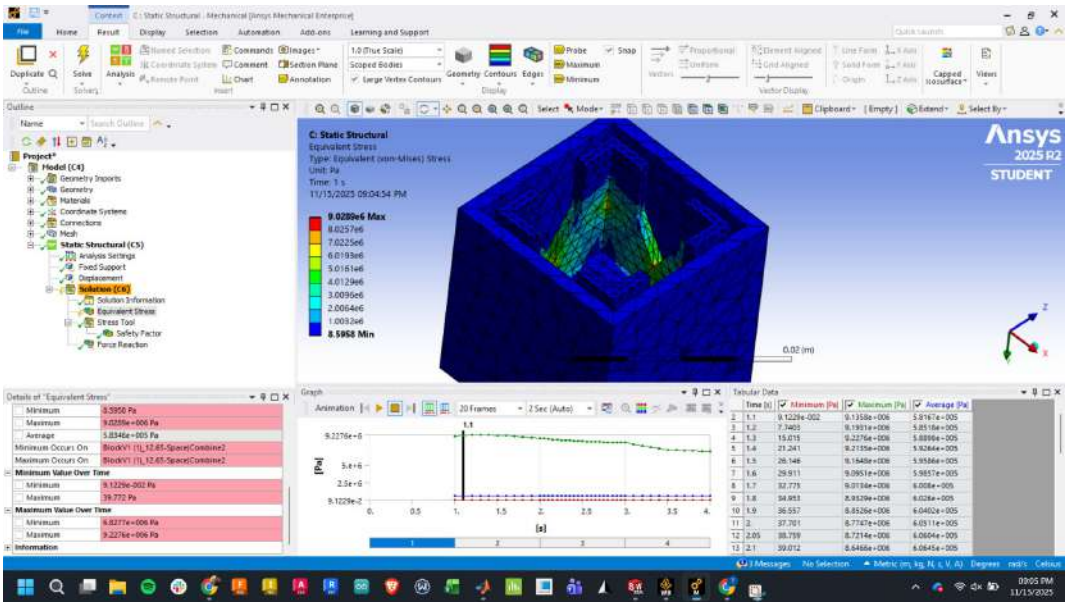


Figure 3: Predicted stress distribution in block arms during separation.

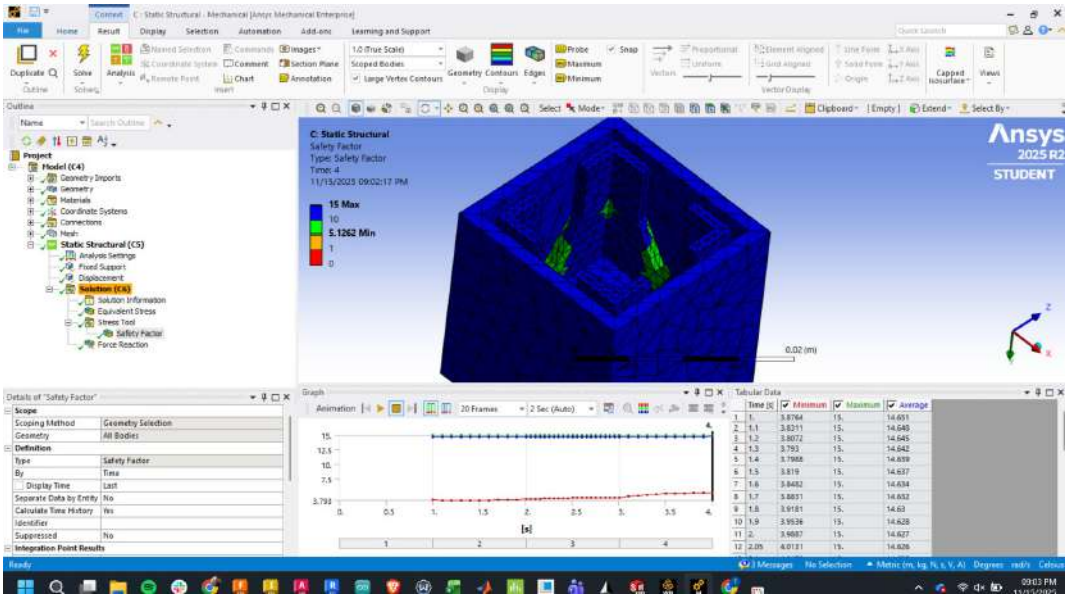


Figure 4: Computed factor of safety across the block geometry.

4 Equations Used

The hand analysis uses Euler–Bernoulli beam bending relations:

$$F = \frac{3EI\,\delta}{L^3}, \qquad \sigma = \frac{FLy}{I},$$

where E is the elastic modulus, I the moment of inertia, L the arm length, y the distance from the neutral axis, δ the imposed displacement, and σ the bending stress.

The normal pressing force used for friction calculations is:

$$F_{\text{normal}} = \frac{3EI\sqrt{2\left(\frac{0.0127-L_{\text{int}}}{2}\right)^2}}{L^3}, \qquad F_{\text{press}} = 4\mu F_{\text{normal}}.$$

The I value of the beams was calcuated using Skyciv Moment of inertia calculator.

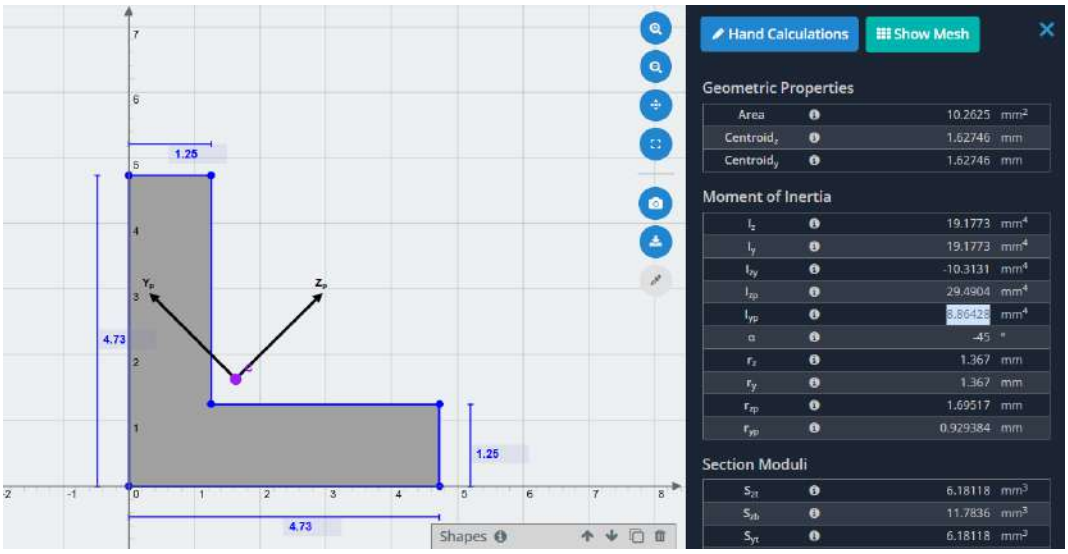


Figure 5: Moment of Inertia Calculation for internal beams

5 MATLAB Code

```
clc;
clear;
close all;

% --- USER PARAMETERS FOR F_press CALC ---
Internal_length = 0.01265;
COF = 0.2;                  % coefficient of friction | change as needed

% Test multiple displacements
displacements = [0.000035, 0.0000707, 0.0001061]; % meters
for i = 1:length(displacements)
    [F, sigma, fail, F_press] = beamStressFromDisplacement(displacements(i),
    ↪  Internal_length, COF);
    fprintf(['Displacement = %.6f m | Force = %.3f N | Stress = %.2f MPa | ', ...
             'Failure = %d | F_press = %.3f N\n'], ...
             displacements(i), F, sigma/1e6, fail, F_press);
end

% Find maximum safe displacement
delta_max = maxSafeDisplacement();
```

```
fprintf('Maximum safe displacement: %.6f m (%.3f mm)\n', delta_max,
↪  delta_max*1000);


% FUNCTION: Stress, Force, and F_press
function [F, sigma, isPastElastic, F_press] = beamStressFromDisplacement(delta,
↪  Internal_length, COF)

    % --- Beam parameters ---
    E = 2400e6;           % Elastic modulus (Pa = N/m^2)
    I = 8.864e-12;        % Moment of inertia (m^4)
    L = 0.0053;           % Beam length (m)
    y = 0.0023;           % Distance from neutral axis (m)
    sigma_max = 35e6;     % Maximum stress (Pa)

    % --- Calculate Force from displacement ---
    F = (3 * E * I * delta) / (L^3);

    % --- Calculate Bending Stress ---
    sigma = (F * L * y) / I;

    % --- Check if stress exceeds elastic limit ---
    isPastElastic = sigma > sigma_max;

    penetration = sqrt(2 * ((0.0127 - Internal_length) / 2)^2);
    F_normal = (3 * E * I * penetration) / (L^3);
    F_press = F_normal * 4 * COF;

end


% FUNCTION: Maximum Safe Displacement
function delta_max = maxSafeDisplacement()

    % Beam parameters
    E = 2400e6;           % Elastic modulus
    L = 0.0053;
    y = 0.0023;
    sigma_max = 35e6;

    % Calculate max allowable displacement
    delta_max = (sigma_max * L^2) / (3 * E * y);
end
```

Analysis: Compressive Loading and Buckling Analysis

Team: 1402

Project: UMass FMS

Prepared by: Tiernan O'Kane

Date: December 2, 2025

1 Overview and Results

This analysis determines the maximum axial compressive load that a single block can withstand before failure. Two failure modes are considered: (1) material yielding under direct compressive stress, and (2) elastic buckling (Euler's critical load). The block is modeled as a short, hollow square column made of ABS material. No Ansys validation was performed for this analysis; all results rely on hand calculations using the Euler column theory.

Key Results

The ultimate compressive capacity of the block is governed by the yield strength of the material, as the buckling load is significantly higher.

- Governing Failure Mode: Material Yielding (crushing).
- Force to Reach Yield Stress (σ_y): 3228 N (725.6 lbf).
- Euler Critical Buckling Load (P_{cr}): 149,773 N (33,670.3 lbf).
- Conclusion: The block will fail by crushing long before it experiences elastic buckling.

2 Rationale for Considering Buckling

Although the block's geometry ($L = 30$ mm, $r_g = 7.6$ mm) clearly places it in the category of a *short column* where material yielding is the expected failure mode, the buckling calculation is still performed for the following critical reasons:

- **Complete Failure Mode Checklist:** Standard engineering practice requires checking all potential failure modes for a loaded component, regardless of expected outcome. This ensures no failure mechanism, even a remote one, is overlooked.
- **Establishing the Failure Regime:** By calculating both F_{yield} and P_{cr}, we confirm that $F_{\text{yield}} \ll P_{cr}$. This rigorously places the component into the short column regime, verifying that the assumption of failure by crushing is mathematically sound.
- **Accounting for Imperfections:** The Euler formula assumes perfect geometry and perfect axial loading. While this is rarely true, calculating P_{cr} provides a theoretical upper bound. If the component were to experience severe eccentricity or manufacturing flaws, a lower, non-axial load could still potentially induce buckling (eccentric buckling).

The calculation confirmed that the block is robust against elastic instability, simplifying the design check to just material stress.

3 Assumptions

- The block is modeled as a perfectly straight, hollow column with a uniform cross-section.
- The material is linear elastic up to the yield stress (σ_y).
- The critical buckling load is calculated using Euler's formula, which is valid for long, slender columns. However, since the calculated critical load is much greater than the yield load, the actual failure is guaranteed to be yielding.
- End conditions are assumed to be pinned-pinned ($K = 1.0$), which is a conservative (worst-case) estimate for the critical load.
- The applied load is perfectly axial (no eccentricity).

4 Numerical Results

Block Geometry and Cross-Sectional Properties

Table 1: Block Geometry and Section Properties

Quantity	Value	Unit
Outer Side (b)	0.0200	m
Wall Thickness (t)	0.00145	m
Height (L)	0.0300	m
Cross-sectional Area (A)	1.0759×10^{-4}	m^2
Second Moment of Area (I)	6.2080×10^{-9}	m^4
Radius of Gyration (r_g)	7.5961×10^{-3}	m

Maximum Compressive Load

Table 2: Calculated Failure Loads

Failure Mode	Load (N)	Load (lbf)
Yielding ($F_{\text{yield}} = \sigma_y \cdot A$)	3227.70	725.6
Elastic Buckling (P_{cr})	149773.13	33670.3

5 Equations Used

This section summarizes the key equations used to determine the cross-sectional properties and the critical failure loads.

5.1 Cross-Sectional Properties

The cross-sectional area (A) for the hollow square section:

$$A = b^2 - (b - 2t)^2$$

The second moment of area (I) for the hollow square section:

$$I = \frac{1}{12}\left(b^4 - (b - 2t)^4\right)$$

5.2 Failure Loads

The force required to cause yielding (crushing) under axial load:

$$F_{\text{yield}} = \sigma_y \cdot A$$

The Euler critical buckling load, assuming pinned-pinned ends ($K = 1$):

$$P_{cr} = \frac{\pi^2 EI}{(KL)^2}$$

6 MATLAB Code (used to generate results)

```
clear; close all; clc;
%% ---------------------- INPUTS ----------------------
b_cm = 2.00;        % outer side (cm)
t_cm = 0.145;       % wall thickness (cm)
L_cm = 3.00;        % height (cm)
% Convert to meters
b = b_cm/100;       % m
t = t_cm/100;       % m
L = L_cm/100;       % m
% Material values
sigma_y = 30e6;     % yield stress, Pa (30 MPa)
E = 2.2e9;          % Young's modulus, Pa (2.2 GPa)
% End condition factor for Euler buckling
K = 1.0;  % pinned-pinned
%% ------------------ CROSS-SECTION PROPERTIES ------------------
% inner side length
inner = b - 2*t;
% cross-sectional area
A = b^2 - inner^2;                % m^2
% second moment of area
I = (b^4 - inner^4)/12;           % m^4
% radius of gyration
r_g = sqrt(I / A);
% --------------------- CALCULATIONS ---------------------
```

```
% Force at yield (axial):
F_yield = sigma_y * A;              % N
% Euler critical buckling load
Pcr = (pi^2 * E * I) / ((K * L)^2);  % N
Fmax_plot = 1.2 * max(F_yield, 0.05*Pcr);
F = linspace(0, Fmax_plot, 400);
% axial stress and axial deflection (short column assumption: axial shortening)
sigma = F ./ A;                     % Pa
delta = (F .* L) ./ (A .* E);       % m
%% --------------------- OUTPUTS ---------------------
fprintf('Geometry (m): b = %.4f, t = %.4f, inner = %.4f, L = %.4f\n', b, t,
    inner, L);
fprintf('Cross-sectional area A = %.4e m^2\n', A);
fprintf('Second moment I = %.4e m^4\n', I);
fprintf('Radius of gyration r_g = %.4e m\n', r_g);
fprintf('Yield stress used sigma_y = %.2e Pa\n', sigma_y);
fprintf('Force to reach yield (F_yield) = %.2f N (%.1f lbf)\n', F_yield,
    F_yield*0.224809);
fprintf('Euler critical load (pinned-pinned) Pcr = %.2f N (%.1f lbf)\n\n', Pcr,
    Pcr*0.224809);
%% --------------------- SUMMARY ---------------------
fprintf('SUMMARY:\n- Yield force (axial) = %.0f N (%.1f lbf)\n- Euler critical
    buckling (pinned-pinned) = %.0f N (%.1f lbf)\n', F_yield, F_yield*0.224809,
    Pcr, Pcr*0.224809);
if Pcr > F_yield * 10
    fprintf('- Buckling is very unlikely: Pcr is >> F_yield.\n');
else
    fprintf('- Check buckling: Pcr is within an order of magnitude of F_yield;
        consider lateral stability and end conditions.\n');
end
```

Analysis: Cyclic Loading and Fatigue Life Analysis

Team: 1402

Project: UMass FMS

Prepared by: Tiernan O'Kane

Date: December 2, 2025

1 Overview and Results

This analysis assesses the fatigue life of the key components (block material, 80/20 beam, and lead screw) under the cyclic loading induced by the assembly process. The factor of safety (FOS) was computed analytically by comparing the nominal stress in each component to the estimated fatigue strength (S_N) for a design life of 10,000 cycles, which is derived from external published data.

Design Life Estimation

The system is designed for a four-year service life before potential obsolescence.

$$N_{\text{cycles}} = (180 \text{ days/year}) \cdot (0.25 \text{ utilization}) \cdot (50 \text{ cycles/use}) \cdot (4 \text{ years}) = 9,000 \text{ cycles}.$$

We round this up to 10,000 cycles for the conservative design requirement.

Key Results ($10,000$ Cycle Fatigue FOS)

All components exceed their required factors of safety based on FOS Jusatification Analysis.

- **Blocks (ABS):** FOS = **3.23**
- **80/20 Beam (Aluminum):** FOS = **57.89**
- **Lead Screw (SUS316):** FOS = **5.07**

This indicates the design can withstand 10,000 cycles of operation without fatigue failure based on the material fatigue strength data used.

2 Assumptions

- The structural stresses used in this analysis (Block: 9.29 MPa, Beam: 1.64 MPa, Screw: 47.30 MPa) are taken from the previous static loading analysis.
- The analysis considers only fully reversed or zero-to-max loading (worst-case), meaning the stress amplitude σ_a is approximately equal to the maximum nominal stress $\sigma_{\max}$.
- Fatigue strength (S_N) at $N = 10,000$ cycles for each material is estimated from published S-N curve data for the respective material grades. The values used are:
  - ABS Blocks: **30** MPa
  - 80/20 Beam (Aluminum): **95** MPa
  - Lead Screw (SUS316): **240** MPa
- Stress concentrations, surface finish effects, and operating temperature effects are not explicitly factored into the fatigue strength.

3 Numerical Results

Fatigue Factor of Safety (N = 10,000 cycles)

Component	Nominal Stress (σ_{max}) (MPa)	Fatigue Strength (S_{10k}) (MPa)	$FOS_{Fatigue}$ (-)
Blocks (ABS)	9.29	30	**3.23**
80/20 Beam (Aluminum)	1.64	95	**57.89**
Lead Screw (SUS316)	47.30	240	**5.07**

4 Equations Used

This section summarizes the primary analytical equations used in the MATLAB model for calculating the nominal stresses and the corresponding Factor of Safety for fatigue.

4.1 Stress Calculation

The nominal stresses (σ_{max}) for the structural components were taken from the previous static analysis (4-block case).

- Block Stress: From Ansys: $\sigma_{block} = 9.289$ MPa.
- Beam Bending Stress:

$$\sigma_{beam} = \frac{M_{max,beam} c_{beam}}{I_{beam}}, \quad \text{where } M_{max,beam} = \frac{F_{each} L_{beam}}{4}.$$

- Lead Screw Bending Stress:

$$\sigma_{screw} = \frac{M_{max,screw} c_{screw}}{I_{screw}}, \quad \text{where } M_{max,screw} = \frac{F_{each} L_{screw}}{4}.$$

4.2 Factor of Safety (Fatigue)

The fatigue factor of safety is computed as the ratio of the estimated fatigue strength at the design life (S_{10k}) to the maximum nominal stress (σ_{max}).

$$FOS_{Fatigue} = \frac{S_{10k}}{\sigma_{max}}.$$

5 MATLAB Code

```
clc; clear; close all;

```

```
% --- Geometry / material ---
E_blocks = 2400e6; I_block = 8.864e-12; L_block = 0.0053; y_block = 0.0023;
delta = 0.000035; % m

% compute block stress
% F_block = (3 * E_blocks * I_block * delta) / (L_block^3); (Hand Calc)
% sigma_block = (F_block * L_block * y_block) / I_block; % Pa  (Hand Calc)
sigma_block = 9.289e6; % Pa (From Ansys)


% piston/beam/screw (4-block case)
COF = 0.2; Internal_length = 0.01265; blocks = 4;
E_beam = 69e9; I_beam = 1.839743e-8; L_beam = 0.254; h_beam = 0.0254;
d_screw = 8e-3; I_screw = pi*d_screw^4/64; E_screw = 193e9;

% F_normal = 3 * E_blocks * I_block * sqrt(2 * ((0.0127 - Internal_length)/2)^2)
↪  / (L_block^3);  (Hand Calc)
% F_press = F_normal * 4 * COF;  (Hand Calc)
F_press = 18.72;
F_total = F_press * blocks;
F_each = F_total / 2;

% stresses
M_beam = (F_each * L_beam) / 4;
sigma_beam = (M_beam * (h_beam/2)) / I_beam;

M_screw = (F_each * L_beam) / 4;
sigma_screw = (M_screw * (d_screw/2)) / I_screw;

% Fatigue strengths at N=1e4
S_al_10k = 95e6; % Pa
% https://asm.matweb.com/search/specificmaterial.asp?bassnum=ma6061t6
S_abs_10k = 30e6;    % Pa
% https://www.sciencedirect.com/science/article/pii/S2452321618302130
%
↪  https://www.researchgate.net/profile/Muhamad-Fitri-2/publication/358110912_The_Fatigue_Str
S_ss_10k  = 240e6;   % Pa
%
↪  https://bssa.org.uk/bssa_articles/fatigue-properties-and-endurance-limits-of-stainless-ste


SF_block = S_abs_10k / sigma_block;
SF_beam  = S_al_10k / sigma_beam;
SF_screw = S_ss_10k / sigma_screw;

fprintf('Block: sigma=%.2f MPa, SF_fatigue(10k) = %.2f\n', sigma_block/1e6,
↪  SF_block);
```

```
46 fprintf('80/20 Beam: sigma=%.3f MPa, SF_fatigue(10k) = %.2f\n', sigma_beam/1e6,
   ↪ SF_beam);
47 fprintf('Lead screw: sigma=%.3f MPa, SF_fatigue(10k) = %.2f\n', sigma_screw/1e6,
   ↪ SF_screw);
```

Analysis: Factor of Safety Justification

Team: 1402

Project: UMass FMS

Prepared by: Tiernan O'Kane

Date: December 2, 2025

1 Overview and Results

This analysis establishes the required factors of safety (FOS) for all subsequent mechanical calculations performed in the project. The approach follows the uncertainty and reliability principles discussed in *Shigley's Mechanical Engineering Design*, where individual sources of modeling error, material variability, manufacturing imprecision, and geometric idealization are represented as multiplicative uncertainty factors. Each assumption in the MATLAB script corresponds directly to a specific uncertainty multiplier, and the product of these multipliers produces a conservative raw FOS value. This value is then rounded up to the nearest 0.5 to maintain consistency and avoid underestimating the safety margin.

Only the assumptions explicitly included in the MATLAB code are used here. These multipliers reflect practical limitations of prototyping (e.g., 3D printing), geometric simplification in beam and interface models, neglected dynamic effects, friction uncertainty, simplifications in load sharing, and similar modeling decisions. Because the design analyses are sequentially dependent, several factors compound into later cases: for example, the rounded Block Holding FOS is carried into the Piston Analysis, and the rounded Piston FOS is subsequently embedded in both the Beam and Lead Screw analyses. This propagation ensures that uncertainty recognized in earlier stages remains accounted for as higher-level structural components are evaluated.

2 Case-by-Case Results

Case 1: Magazine Bending

Assumption	Multiplier
No Dynamic Effects	1.20
3D Printed	1.75
Stress concentrations ignored	1.10
Worst Case Cross-Section	0.85
Assumed fasteners share load equally	1.05
Contact area circular and loaded uniformly	1.05
Raw FOS	2.16
Rounded FOS	2.50

Case 2: Block Holding Power

Assumption	Multiplier
Manufacturing tolerance	1.02
3D Printed	1.75
Simplified beam geometry (fillets omitted)	0.95
No Dynamic Effects	1.20
0.2 Friction Assumption	1.20
Linear-Elastic Behavior Assumed	1.10
Raw FOS	2.69
Rounded FOS	3.00

Case 3: Piston Analysis

Assumption	Multiplier
Uniform pressure	1.05
PSI of 80	1.05
All assumptions from Block FOS	3.00
Raw FOS	3.31
Rounded FOS	3.50

Case 4: Frictional Sliding

Assumption	Multiplier
Friction coefficient uncertainty	1.20
Mass variability	1.15
Geometry simplification (perfect contact assumed)	1.05
Raw FOS	1.45
Rounded FOS	1.50

Case 5: Beam Analysis

Assumption	Multiplier
Total force split evenly	1.20
Neglected weight of assembly	1.10
Dynamic piston impulse neglected	1.20
Simply supported beam with load in middle	0.75
All assumptions from piston FOS	3.50
Raw FOS	4.16
Rounded FOS	4.50

Case 6: Lead Screw Analysis

Assumption	Multiplier
Total force split evenly	1.20
Neglected weight of assembly	1.10
Lead screw approximated as constant circular cross-section	1.20
Dynamic piston impulse neglected	1.20
Simply supported beam with load in middle	0.75
All assumptions from piston FOS	3.50
Raw FOS	4.99
Rounded FOS	5.00

3 Summary of Recommended Factors of Safety

Case	Recommended FOS
Magazine Bending	2.5
Block Holding	3.0
Piston	3.5
Frictional Sliding	1.5
Beam	4.5
Lead Screw	5.0

4 Equations Used

The calculation procedure used is:

$$\text{FOS}_{\text{raw}} = \prod_{i=1}^{n} m_i$$

$$\mathrm{FOS}_{\mathrm{rounded}} = \left\lceil \frac{\mathrm{FOS}_{\mathrm{raw}}}{0.5} \right\rceil \times 0.5$$

where m_i are the multipliers listed in each case.

5 MATLAB Code

```
clc; clear; close all;

%% -------------------- Helper Function --------------------
roundUpHalf = @(x) ceil(x/0.5)*0.5;

%% -------------------- Case 1: Magazine Bending Analysis ----------------
case1_names = { ...
    'No Dynamic Effects', ...
    '3D Printed', ...
    'Stress concentrations ignored', ...
    'Worst Case Cross-Section', ...
    'Assumed fasteners share load equally', ...
    'Contact area circular and loaded uniformly'};

case1_values = [1.2, 1.75, 1.10, 0.85, 1.05, 1.05];

case1_total = prod(case1_values);
case1_total_r = roundUpHalf(case1_total);

fprintf('\n=== Case 1: Magazine Bending Analysis ===\n');
T1 = table(case1_names', case1_values', 'VariableNames',
 ↪  {'Assumption','Multiplier'});
disp(T1);
fprintf('Recommended FOS = %.2f  --> Rounded = %.2f\n', case1_total,
 ↪  case1_total_r);

%% -------------------- Case 2: Block Holding Power ----------------------
case2_names = { ...
    'Manufacturing tolerance', ...
    '3D Printed', ...
    'Simplified beam geometry (fillets omitted)', ...
    'No Dynamic Effects', ...
    '0.2 Friction Assumption', ...
    'Linear-Elastic Behavior Assumed'};

case2_values = [1.02, 1.75, 0.95, 1.2, 1.2, 1.1];

case2_total = prod(case2_values);
case2_total_r = roundUpHalf(case2_total);

```

```
fprintf('\n=== Case 2: Block Holding Power ===\n');
T2 = table(case2_names', case2_values', 'VariableNames',
    {'Assumption','Multiplier'});
disp(T2);
fprintf('Recommended FOS = %.2f  --> Rounded = %.2f\n', case2_total,
    case2_total_r);

%% -------------------- Case 3: Piston Analysis ----------------------
case3_names = { ...
    'Uniform pressure', ...
    'PSI of 80', ...
    'All assumptions from Block FOS'};

case3_values = [1.05, 1.05, case2_total_r];

case3_total = prod(case3_values);
case3_total_r = roundUpHalf(case3_total);

fprintf('\n=== Case 3: Piston Analysis ===\n');
T3 = table(case3_names', case3_values', 'VariableNames',
    {'Assumption','Multiplier'});
disp(T3);
fprintf('Recommended FOS = %.2f  --> Rounded = %.2f\n', case3_total,
    case3_total_r);

%% -------------------- Case 4: Frictional Sliding Analysis ---------------
case4_names = { ...
    'Friction coefficient uncertainty', ...
    'Mass variability', ...
    'Geometry simplification (perfect contact assumed)'};

case4_values = [1.2, 1.15, 1.05];

case4_total = prod(case4_values);
case4_total_r = roundUpHalf(case4_total);

fprintf('\n=== Case 4: Frictional Sliding Analysis ===\n');
T4 = table(case4_names', case4_values', 'VariableNames',
    {'Assumption','Multiplier'});
disp(T4);
fprintf('Recommended FOS = %.2f  --> Rounded = %.2f\n', case4_total,
    case4_total_r);

%% -------------------- Case 5: Beam Analysis ----------------------
case5_names = { ...
    'Total force split evenly', ...
    'Neglected weight of assembly', ...
    'Dynamic piston impulse neglected', ...
```

```
    'Simply supported beam with load in middle', ...
    'All assumptions from piston FOS'};

case5_values = [1.2, 1.1, 1.2, 0.75, case3_total_r];

case5_total = prod(case5_values);
case5_total_r = roundUpHalf(case5_total);

fprintf('\n=== Case 5: Beam Analysis ===\n');
T5 = table(case5_names', case5_values', 'VariableNames',
↪  {'Assumption','Multiplier'});
disp(T5);
fprintf('Recommended FOS = %.2f  --> Rounded = %.2f\n', case5_total,
↪  case5_total_r);

%% -------------------- Case 6: Lead Screw Analysis ----------------------
case6_names = { ...
    'Total force split evenly', ...
    'Neglected weight of assembly', ...
    'Lead screw approximated as constant circular cross section', ...
    'Dynamic piston impulse neglected', ...
    'Simply supported beam with load in middle', ...
    'All assumptions from piston FOS'};

case6_values = [1.2, 1.1, 1.2, 1.2, 0.75, case3_total_r];

case6_total = prod(case6_values);
case6_total_r = roundUpHalf(case6_total);

fprintf('\n=== Case 6: Lead Screw Analysis ===\n');
T6 = table(case6_names', case6_values', 'VariableNames',
↪  {'Assumption','Multiplier'});
disp(T6);
fprintf('Recommended FOS = %.2f  --> Rounded = %.2f\n', case6_total,
↪  case6_total_r);

%% -------------------- Summary Table ----------------------------
summary_names = {'Magazine Bending', 'Block Holding', 'Piston', 'Frictional
↪  Sliding', 'Beam', 'Lead Screw'}';
summary_FOS = [case1_total_r, case2_total_r, case3_total_r, case4_total_r,
↪  case5_total_r, case6_total_r]';

SummaryTable = table(summary_names, summary_FOS, ...
    'VariableNames', {'Case','Recommended_FOS'});

fprintf('\n=== SUMMARY OF RECOMMENDED FACTORS OF SAFETY ===\n');
disp(SummaryTable);
```

Analysis: Magazine Connection and Bending Analysis

Team: 1402

Project: UMass FMS

Prepared by: Tiernan O'Kane

Date: December 2, 2025

1 Overview and Results

This analysis evaluates the structural capacity of three different magazine holder designs. Each magazine holder is modeled as a cantilever beam subjected to its own distributed self-weight and a concentrated load from the contained blocks. The analysis verifies that the maximum bending stress remains below the material's allowable stress and calculates the required fastener size to prevent failure at the connection point. Results were computed using a MATLAB script with a design Factor of Safety of 2.5 and validated using Ansys FEA (included below).

Key Results (Governing Case: Holder 3)

Holder 3 represents the limiting case due to the heavier blocks (85 g) and a smaller cross-sectional height (30 mm), resulting in the highest stress and deflection.

- Maximum Bending Stress: 0.06 MPa (Allowable: 16.00 MPa). All holders are OK.
- Maximum End Deflection: 0.010 mm (in Holder 3). Deflection is negligible.
- Required Bolt Diameter (Tension/Shear): 0.018 in (for Holder 3).
- Required Contact Diameter (Tear-Out): 0.056 in (for Holder 3).

All calculated stresses are significantly lower than the allowable stress, confirming that the current geometry is over-designed for the expected loads. The required bolt and contact areas are minimal, meaning standard fasteners will provide a substantial Factor of Safety.

2 Assumptions

- Each magazine holder is modeled as a fixed-end cantilever beam.
- The total load from the blocks is treated as a single concentrated point load acting at the load centroid (0.12 m from the support).
- The self-weight of the magazine holder is modeled as a uniformly distributed load (w).
- The analysis uses a design Factor of Safety of 2.5 for both material stress and connection shear/tension capacity.
- Tear-out failure is calculated based on the combined resultant force and the allowable material strength for the contact area.

3 Ansys Validation Images

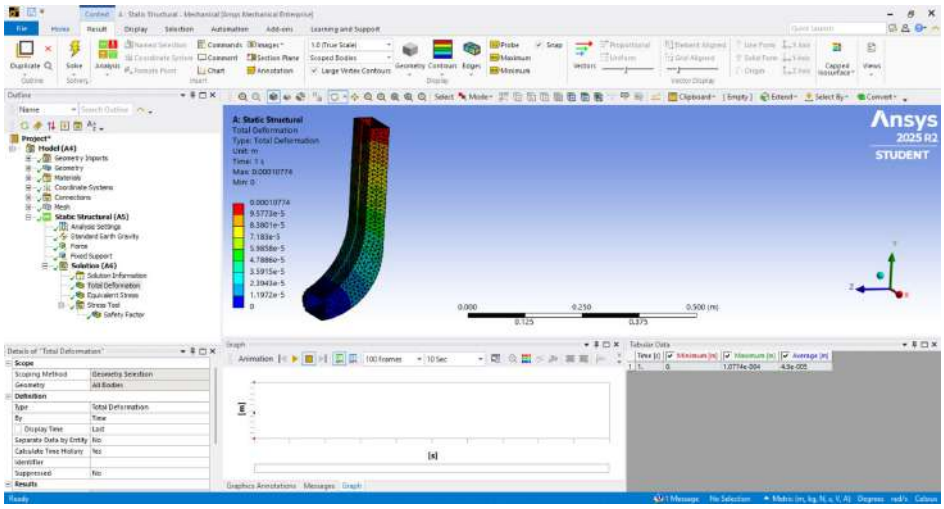


Figure 1: Ansys Total deformation contour under maximum loading.

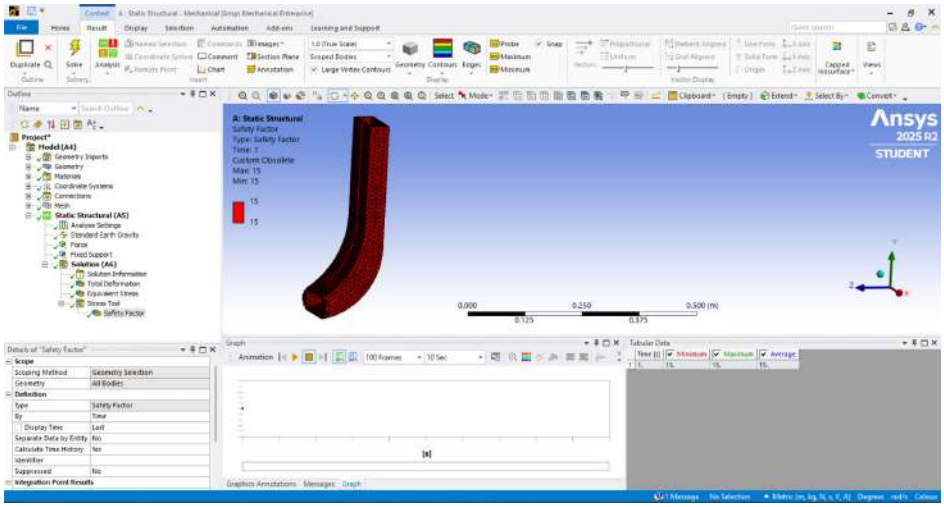


Figure 2: Ansys Factor of Safety distribution (based on Yield Strength).

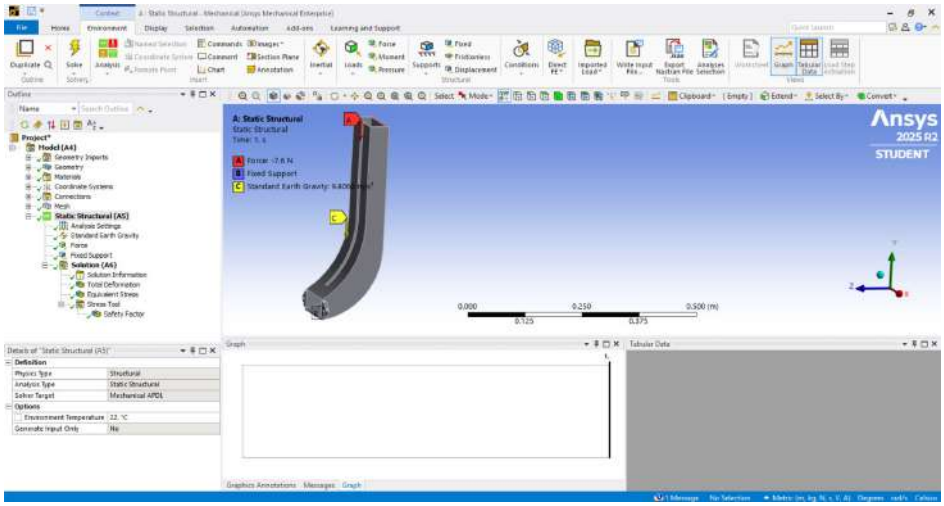


Figure 3: Ansys Model setup showing boundary conditions (fixed support) and loading.

4 Numerical Results

Magazine Structural and Connection Summary

Table 1: Structural Mechanics Results

Table 1: Block Holder Analysis: Load, Stress, and Deflection

Holder	**Total Load** (N)	**Moment** (N·m)	$\sigma_{\max}$ (MPa)	$\sigma_{\mathbf{allow}}$ (MPa)	**Deflection** (mm)
1	5.89	0.500	0.02	16.00	0.002
2	5.40	0.467	0.02	16.00	0.003
3 (Limit)	7.60	0.732	0.06	16.00	0.010

—

Table 2: Connection Mechanics Results

Table 2: Block Holder Analysis: Required Connection Diameters

Holder	**Req. Bolt d** (in)	**Req. Contact d** (in)
1	0.012	0.038
2	0.012	0.037
3 (Limit)	0.018	0.056

5 Equations Used

The analysis uses cantilever beam equations to determine the moment, shear, and deflection at the fixed support, followed by connection analysis.

5.1 Cantilever Loads and Moments

The total moment at the fixed support is the sum of the moment from the blocks (P) and the magazine's self-weight (W).

$$M_{\text{fixed}} = M_{\text{blocks}} + M_{\text{mag}} = (P_{\text{blocks}} \cdot \text{dist}_{\text{centroid}}) + \left(\frac{W_{\text{mag}} L}{2}\right)$$

5.2 Bending Stress and Deflection

The maximum bending stress (σ_{max}) occurs at the fixed support. I is the second moment of area for a hollow rectangle, and y is half the height.

$$\sigma_{\text{max}} = \frac{M_{\text{fixed}} y}{I}$$

The total deflection (δ_{end}) is the superposition of deflections due to the point load (P) and the distributed load (w).

$$\delta_{\text{end}} = \left(\frac{P_{\text{blocks}} a^2 (3L - a)}{6EI}\right) + \left(\frac{wL^4}{8EI}\right)$$

5.3 Required Bolt Area (Tension/Shear)

The moment M_{fixed} is converted into a tensile load (T) on the bolts, acting over a lever arm ($h/2$).

$$T_{\text{per bolt}} = \frac{M_{\text{fixed}}}{\text{lever}_{\text{arm}} \cdot n_{\text{screws}}}$$

The required bolt area (A_{req}) is calculated using the combined shear ($V_{\text{per bolt}}$) and tension ($T_{\text{per bolt}}$) stress equation:

$$A_{\text{req}} = \frac{\sqrt{T_{\text{per bolt}}^2 + 3 \cdot V_{\text{per bolt}}^2}}{\sigma_{\text{allow, screw}}}$$

6 MATLAB Code (used to generate results)

```
clear; clc; close all
%% -------------------- INPUTS --------------------
numBlocks = [5, 5, 5];                  % count of blocks per holder
blockWeight_g = [40, 40, 85];       % grams per block
dist_centroid = [0.12, 0.12, 0.12];     % distance from fixed end to centroid (m)
```

```
holder_length = [0.135, 0.135, 0.135];     % cantilever length (m)
% Magazine self-weight (uniform along holder)
magWeight_g = [400, 350, 350];          % grams total per holder
% ABS material properties
E_abs = 2.0e9 .* ones(size(numBlocks));        % Pa
tensile_strength_abs = [4.0e7, 4.0e7, 4.0e7]; % Pa
FS_common = 2.5;                               % factor of safety
% Hollow rectangular section
b = [0.114, 0.095, 0.114];    % m
h = [0.046, 0.046, 0.030];    % m
t = [0.0052, 0.0052, 0.0052];  % m
% Bolt / fastener properties
tensile_strength_screw = [4.0e8, 4.0e8, 4.0e8]; % Pa
shear_strength_ratio = 0.57;
n_screws = 2;
%% -------------------- UNIT CONVERSIONS --------------------
blockWeight_N = blockWeight_g .* (9.81 / 1000);  % weight per block
magWeight_N   = magWeight_g .* (9.81 / 1000);    % total mag weight per holder
%% -------------------- SECTION PROPERTIES --------------------
N = length(numBlocks);
I = zeros(1,N);
for i = 1:N
    I(i) = I_from_hollow_rect(b(i), h(i), t(i)); % m^4
end
y = h ./ 2; % m
%% -------------------- LOADS, MOMENTS, DEFLECTIONS --------------------
P_blocks = numBlocks .* blockWeight_N;      % N
M_blocks = P_blocks .* dist_centroid;       % N·m
V_blocks = P_blocks;                        % N
W_mag = magWeight_N;                        % N
w = W_mag ./ holder_length;                 % N/m
V_mag = W_mag;
M_mag = W_mag .* holder_length ./ 2;
delta_mag = w .* holder_length.^4 ./ (8 .* E_abs .* I);
P_total = P_blocks + W_mag;
M_fixed = M_blocks + M_mag;
V_fixed = V_blocks + V_mag;
sigma_max = abs(M_fixed .* y ./ I);
sigma_allow_material = tensile_strength_abs ./ FS_common;
material_ok = sigma_max <= sigma_allow_material;
delta_blocks = P_blocks .* dist_centroid.^2 .* (3.*holder_length - dist_centroid)
↪  ./ (6 .* E_abs .* I);
delta_end = delta_blocks + delta_mag;
%% -------------------- BOLT FORCE CALCULATION --------------------
lever_arm = h ./ 2;
T_from_moment_per_bolt = M_fixed ./ (lever_arm .* n_screws);
V_per_bolt = V_fixed ./ n_screws;
sigma_allow_screw = tensile_strength_screw ./ FS_common;
```

```
tau_allow_screw = shear_strength_ratio .* tensile_strength_screw ./ FS_common;
A_req_shear = V_per_bolt ./ tau_allow_screw;
A_req_tension = T_from_moment_per_bolt ./ sigma_allow_screw;
A_req_combined = sqrt(T_from_moment_per_bolt.^2 + 3 .* V_per_bolt.^2) ./ sigma_allow_screw;
A_req_bolt = max([A_req_shear; A_req_tension; A_req_combined], [], 1);
d_req_bolt = sqrt(4 .* A_req_bolt ./ pi);
%% -------------------- TEAR-OUT AREA (CONTACT) --------------------
sigma_allow_mat = sigma_allow_material;
A_contact_req_combined = sqrt(T_from_moment_per_bolt.^2 + 3 .* V_per_bolt.^2) ./ sigma_allow_mat;
A_contact_req_tension = T_from_moment_per_bolt ./ sigma_allow_mat;
d_contact_req_combined = sqrt(4 .* A_contact_req_combined ./ pi);
%% --------------------- OUTPUT ---------------------
m_to_cm = 100; m_to_mm = 1000; m_to_in = 1/0.0254; m2_to_in2 = 1/(0.0254^2);
Holder = (1:N)';
T = table(Holder, ...
    numBlocks(:), blockWeight_g(:), magWeight_g(:), ...
    P_blocks(:), W_mag(:), P_total(:), ...
    M_fixed(:), V_fixed(:), ...
    y(:)*m_to_cm, ...
    sigma_max(:)/1e6, sigma_allow_material(:)/1e6, material_ok(:), ...
    delta_end(:)*m_to_mm, ...
    d_req_bolt(:)*m_to_in, A_req_bolt(:)*m2_to_in2, ...
    d_contact_req_combined(:)*m_to_in, A_contact_req_combined(:)*m2_to_in2, ...
    'VariableNames', {'Holder','NumBlocks','BlockWeight_g','MagWeight_g', ...
    'Total_Block_Load_N','Mag_Load_N','Total_Load_N', ...
    'Moment_total_Nm','Shear_total_N', ...
    'y_cm','Sigma_max_MPa','Sigma_allow_MPa','Material_OK', ...
    'Deflection_total_mm','Req_bolt_diameter_in','Req_bolt_area_in2', ...
    'Req_contact_diameter_in','Req_contact_area_in2'});
disp('----- BLOCK HOLDER ANALYSIS (Weights in grams, outputs in readable units) -----')
disp(T)
%% -------------------- SUMMARY --------------------
for i = 1:N
    fprintf('\nHolder %d summary:\n', i)
    fprintf('  Blocks: %d @ %.0f g each (%.2f N total)\n', numBlocks(i), blockWeight_g(i), P_blocks(i))
    fprintf('  Magazine: %.0f g (%.2f N)\n', magWeight_g(i), W_mag(i))
    fprintf('  Total vertical load: %.2f N\n', P_total(i))
    fprintf('  Total moment at support: %.3f N·m\n', M_fixed(i))
    fprintf('  Max bending stress: %.2f MPa (allow: %.2f MPa) -> %s\n', ...
        sigma_max(i)/1e6, sigma_allow_material(i)/1e6, ternary(material_ok(i),'OK','FAIL'))
    fprintf('  End deflection: %.3f mm\n', delta_end(i)*m_to_mm)
    fprintf('  Required bolt diameter: %.3f in (area %.3f in²)\n', ...
        d_req_bolt(i)*m_to_in, A_req_bolt(i)*m2_to_in2)
```

```
    fprintf('  Required contact diameter: %.3f in (area %.3f in²)\n', ...
        d_contact_req_combined(i)*m_to_in, A_contact_req_combined(i)*m2_to_in2)
end
%% -------------------- HELPER FUNCTIONS --------------------
function I = I_from_hollow_rect(b, h, t)
    b_inner = b - 2*t; h_inner = h - 2*t;
    I = (b.*h.^3 - b_inner.*h_inner.^3) ./ 12;
end
function out = ternary(cond, valTrue, valFalse)
    if cond, out = valTrue; else out = valFalse; end
end
```

Analysis: Magazine Sliding Curvature Analysis

Team: 1402

Project: UMass FMS

Prepared by: Tiernan O'Kane

Date: December 2, 2025

1 Overview and Results

Objective: determine the *maximum* radius of curvature R of a magazine such that a block of length L and mass m will slide under gravity despite friction. In other words, find the largest curvature radius for which sliding still occurs, and compute margins (factors of safety) for a chosen design radius.

Consider a block of length $L = 0.125$ m and mass $m = 0.085$ kg in a curved magazine with coefficient of friction $\mu = 0.20$. The downhill component of weight along the arc is:

$$F_{\text{down}} = mg \sin\theta,$$

and the maximum available friction is

$$F_{\text{fric,max}} = \mu mg \cos\theta,$$

where the geometry gives

$$\theta = \arcsin\left(\frac{L}{2R}\right).$$

Set the sliding threshold by equating $F_{\text{down}} = F_{\text{fric,max}}$. This gives the analytic critical condition $\tan\theta = \mu$. Using $\theta = \arctan(\mu)$ and $\theta = \arcsin(L/(2R))$ yields the critical radius:

$$\boxed{R_{\text{crit}} = \frac{L}{2\sin(\arctan\mu)}.}$$

For the given parameters this evaluates to:

$$R_{\text{crit}} = 0.319 \text{ m}, \qquad \theta_{\text{crit}} = 11.31^\circ.$$

Design check (for $R_{\text{given}} = 0.15$ m):

$$\theta = 24.62^\circ, \quad F_{\text{down}} = 0.347 \text{ N}, \quad F_{\text{fric,max}} = 0.152 \text{ N}.$$

Define sliding margins as:

$$\text{FoS}_{\text{force}} = \frac{F_{\text{down}}}{F_{\text{fric,max}}}, \qquad \text{FoS}_{\text{radius}} = \frac{R_{\text{crit}}}{R_{\text{given}}}.$$

Numerical values for $R_{\text{given}} = 0.15$ m:

- Force-based FoS: $\boxed{2.2917}$
- Radius-based FoS: $\boxed{2.1246}$

2 Assumptions

- Block remains in continuous contact with the magazine walls (no lift or wedging).
- Coulomb friction with constant coefficient $\mu = 0.20$ (ABS to ABS assumption).
- Block and ramp are rigid.
- No dynamic effects (quasi-static equilibrium used).

3 Free Body Diagram

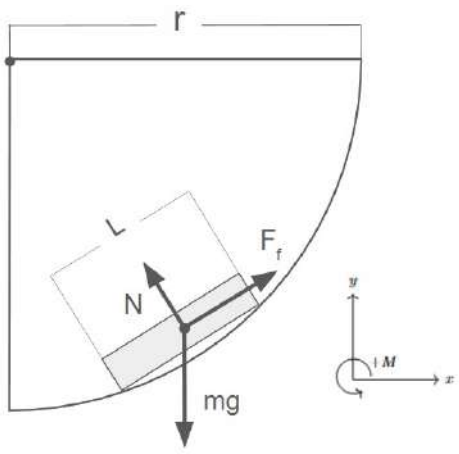


Figure 1: Free-body diagram of the block in the curved magazine.

4 MATLAB Results

Analytic critical radius

Quantity	Value
Critical radius R_{crit}	0.317755 m
Critical angle θ_{crit}	11.3436°

Design check at $R_{\text{given}} = 0.15$ m

Quantity	Value
Angle θ	24.6243°
F_{down}	0.3474 N
$F_{\text{fric,max}}$	0.1516 N
Force-based FoS $= F_{\text{down}}/F_{\text{fric,max}}$	2.2917
Radius-based FoS $= R_{\text{crit}}/R_{\text{given}}$	2.4000

5 MATLAB Plot

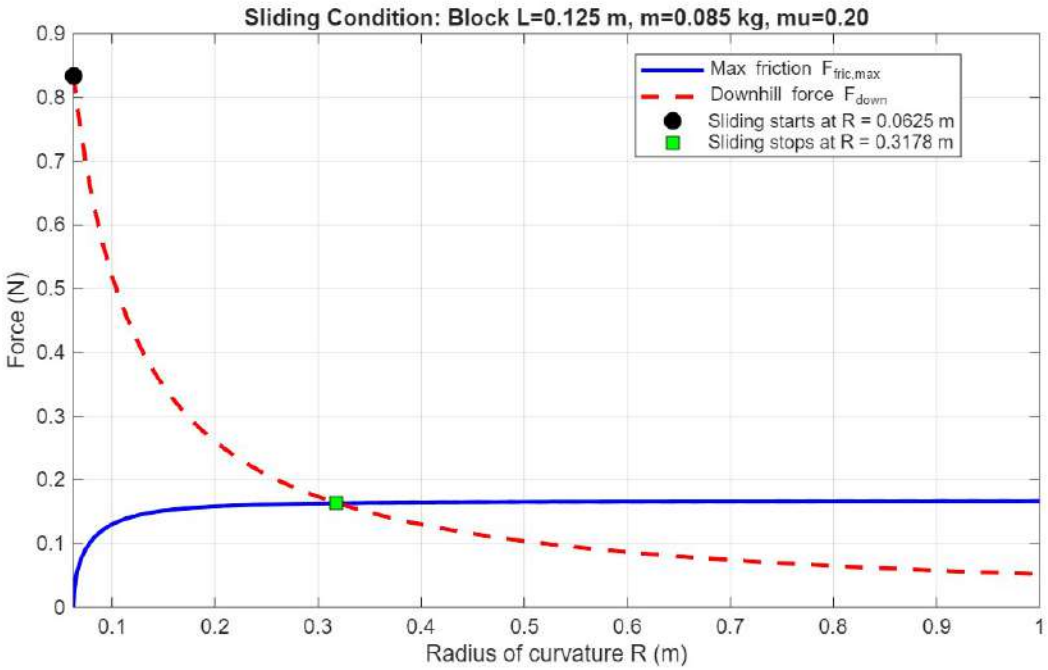


Figure 2: Downhill force F_{down} and maximum friction $F_{\text{fric,max}}$ vs radius R. The crossing point indicates R_{crit}.

6 Equations Used

$$\theta = \arcsin\left(\frac{L}{2R}\right), \qquad F_{\text{down}} = mg\sin\theta, \qquad F_{\text{fric,max}} = \mu mg\cos\theta.$$

Critical condition:

$$F_{\text{down}} = F_{\text{fric,max}} \quad \Rightarrow \quad \tan\theta = \mu \quad \Rightarrow \quad R_{\text{crit}} = \frac{L}{2\sin(\arctan\mu)}.$$

FoS (sliding margin):

$$\text{FoS}_{\text{force}} = \frac{F_{\text{down}}}{F_{\text{fric,max}}}, \qquad \text{FoS}_{\text{radius}} = \frac{R_{\text{crit}}}{R_{\text{given}}}.$$

7 MATLAB Code

```
clc; clear; close all;

%% --- Parameters ---
```

```
m = 0.085;        % kg
L = 0.125;        % m
mu = 0.2;         % friction coefficient
g = 9.81;         % m/s^2

%% --- Radius (R must be >= L/2) ---
Rmin = L/2 + 1e-9; % just above L/2 to avoid errors
Rmax = 1.0;
R = linspace(Rmin, Rmax, 1000);

theta = nan(size(R));
F_down = nan(size(R));
F_fric_max = nan(size(R));

%% --- Compute theta, forces ---
arg = L ./ (2 .* R);              % argument for asin, <= 1
valid = arg <= 1;

theta(valid) = asin(arg(valid));              % theta = asin(L/(2R))  (rad)
F_down(valid) = m .* g .* sin(theta(valid));
F_fric_max(valid) = mu .* m .* g .* cos(theta(valid));

%% --- Determine sliding start and stop ---
will_slide = (F_down > F_fric_max) & valid;

idx_start = find(diff([0 will_slide]) == 1, 1, 'first');
idx_stop  = find(diff([will_slide 0]) == -1, 1, 'first');

R_slide_start = [];
R_slide_stop = [];

if ~isempty(idx_start)
    R_slide_start = R(idx_start);
end
if ~isempty(idx_stop)
    R_slide_stop = R(idx_stop);
end

%% --- Plot ---
figure('Position',[100 100 800 450]);
plot(R(valid), F_fric_max(valid), 'b-', 'LineWidth', 2); hold on;
plot(R(valid), F_down(valid), 'r--', 'LineWidth', 2);

if ~isempty(R_slide_start)
    plot(R_slide_start, F_down(idx_start), 'ko', 'MarkerFaceColor','k',
    ↪  'MarkerSize',8);
end
if ~isempty(R_slide_stop)
```

```
    plot(R_slide_stop, F_down(idx_stop), 'ks', 'MarkerFaceColor','g',
    ↪ 'MarkerSize',8);
end

if ~isempty(R_slide_start) && ~isempty(R_slide_stop)
    legend('Max friction F_{fric,max}', 'Downhill force F_{down}', ...
        sprintf('Sliding starts at R = %.4f m', R_slide_start), ...
        sprintf('Sliding stops at R = %.4f m', R_slide_stop), ...
        'Location','best');
elseif ~isempty(R_slide_start)
    legend('Max friction F_{fric,max}', 'Downhill force F_{down}', ...
        sprintf('Sliding starts at R = %.4f m', R_slide_start), ...
        'Location','best');
else
    legend('Max friction F_{fric,max}', 'Downhill force F_{down}',
    ↪ 'Location','best');
end

xlabel('Radius of curvature R (m)');
ylabel('Force (N)');
title(sprintf('Sliding Condition: Block L=%.3f m, m=%.3f kg, mu=%.2f', L, m,
↪ mu));
grid on;
xlim([Rmin Rmax]);

%% --- Summary ---
if isempty(R_slide_start)
    fprintf('No sliding occurs for R in [%.6f, %.6f] m.\n', Rmin, Rmax);
else
    fprintf('Sliding begins at R = %.6f m\n', R_slide_start);
    fprintf('At that R: theta = %.4f deg, F_down = %.4f N, F_fric_max = %.4f
    ↪ N\n', ...
        rad2deg(theta(idx_start)), F_down(idx_start), F_fric_max(idx_start));
    if ~isempty(R_slide_stop)
        fprintf('Sliding stops at R = %.6f m\n', R_slide_stop);
        fprintf('At that R: theta = %.4f deg, F_down = %.4f N, F_fric_max = %.4f
        ↪ N\n', ...
            rad2deg(theta(idx_stop)), F_down(idx_stop), F_fric_max(idx_stop));
    else
        fprintf('Sliding continues for all larger R (no stop within range).\n');
    end
end

%% ================================================================
%% --- Factor of Safety Calculation for Given Radius R_given ---
%% ================================================================
R_given = 0.15;   % m

```

```
% Compute values at the given radius
theta_given = asin(L / (2*R_given));
F_down_given = m * g * sin(theta_given);
F_fric_given = mu * m * g * cos(theta_given);

% Force-based FoS
FoS_force =  F_down_given / F_fric_given;

% Radius-based FoS (compared to sliding threshold)
if ~isempty(R_slide_start)
    FoS_radius = R_given / R_slide_start;
else
    FoS_radius = Inf; % never slides in range → infinite margin
end

%% --- Display FoS results ---
fprintf("\n=== FACTOR OF SAFETY ANALYSIS ===\n");
fprintf("Given radius R = %.4f m\n", R_given);
fprintf("Theta = %.4f deg\n", rad2deg(theta_given));
fprintf("F_down = %.4f N\n", F_down_given);
fprintf("F_fric_max = %.4f N\n", F_fric_given);
fprintf("FoS (force-based) = %.4f\n", FoS_force);
fprintf("FoS (radius-based) = %.4f\n", FoS_radius);

```

Analysis: Piston Force and Structural Capacity Analysis

Team: 1402

Project: UMass FMS

Prepared by: Tiernan O'Kane

Date: December 2, 2025

1 Overview and Results

This analysis determines the required piston force to compress between four and six blocks during the assembly process and verifies that the supporting structure (an 80/20 aluminum beam and an 8 mm SUS316 lead screw) can carry the resulting load without yielding or excessive deflection. Results below are taken from the MATLAB model and were validated with Ansys.

Key Results

- **4-block case (Chosen Case):**
  - Required piston force: **74.88** N
  - Required piston area (at 80 psi): **0.210** in^2
  - 80/20 beam: stress = **1.64** MPa, deflection = **0.010** mm, FOS = **105.05**
  - Lead screw: stress = **56.79** MPa, deflection = **0.570** mm, FOS = **4.44**
- **6-block case:**
  - Required piston force: **112.32** N
  - Required piston area (at 80 psi): **0.316** in^2
  - 80/20 beam: stress = **2.46** MPa, deflection = **0.015** mm, FOS = **70.03**
  - Lead screw: stress = **85.19** MPa, deflection = **0.855** mm, FOS = **2.96**

All computed stresses lie below material yield strengths and deflections are small; therefore the selected piston force for the 6-block case is acceptable for the assembly given the assumed load-sharing and boundary conditions. Ansys validation images are included in the following section.

2 Assumptions

- Per-block pressing force used: $F_{\text{press}} = 18.72$ N.
- Piston operates at up to 80 psi; piston area is sized from required force at that pressure.
- Load from the piston is split equally between the 80/20 beam and the lead screw. (This assumption is the weakest and explains most of the difference between the hand calculations and Ansys results).
- Beam modeled as simply supported with a center point load; lead screw modeled as simply supported circular beam under bending.
- Materials are linear elastic; fatigue and buckling are not considered in this analysis.

3 Piston Force Graph

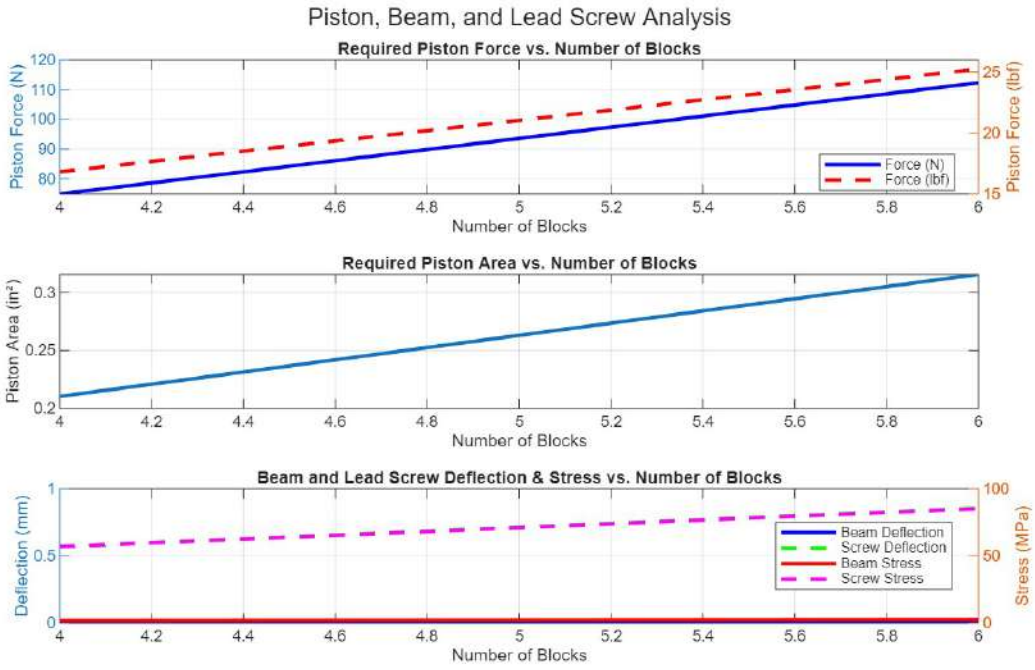


Figure 1: MATLAB-generated plot: required piston force / piston area and structural responses vs. number of blocks.

4 Ansys Validation Images

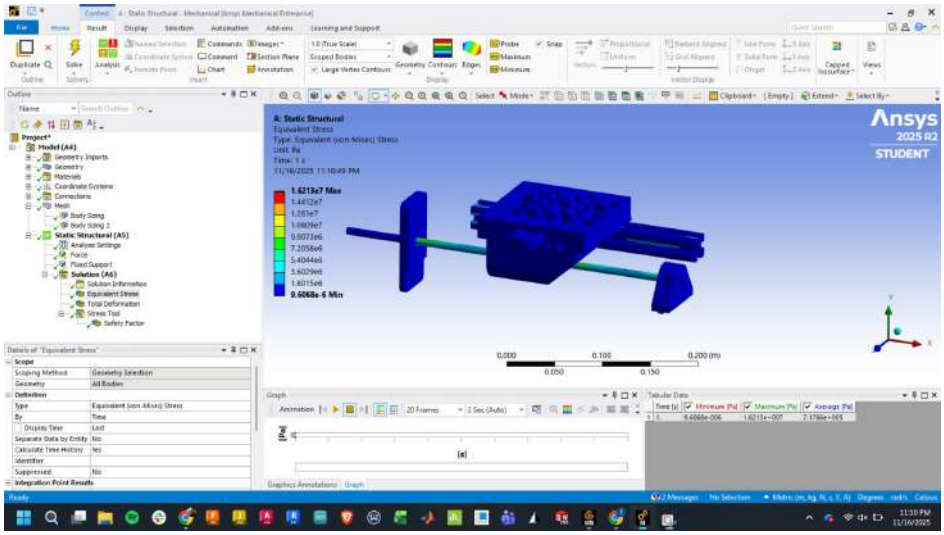


Figure 2: Ansys Stress contour under piston loading.

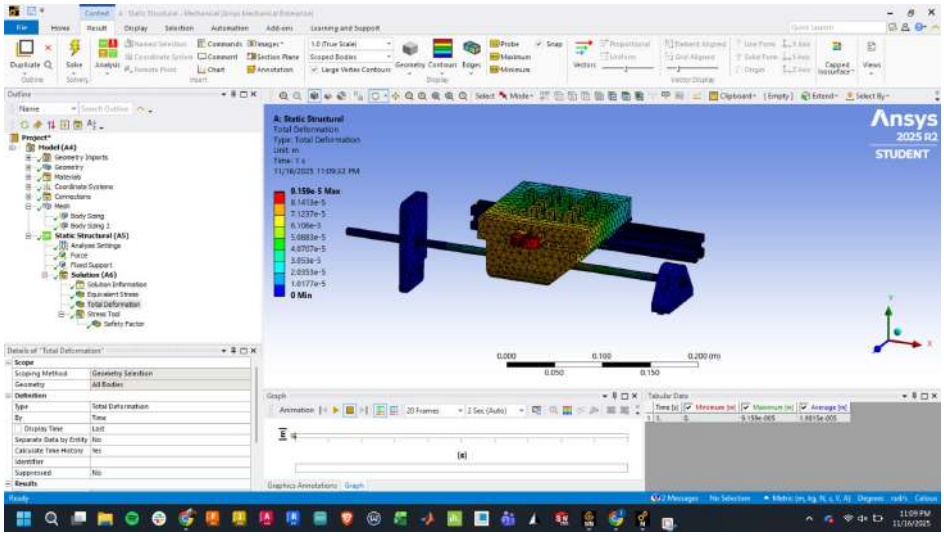


Figure 3: Ansys Total deformation under piston loading.

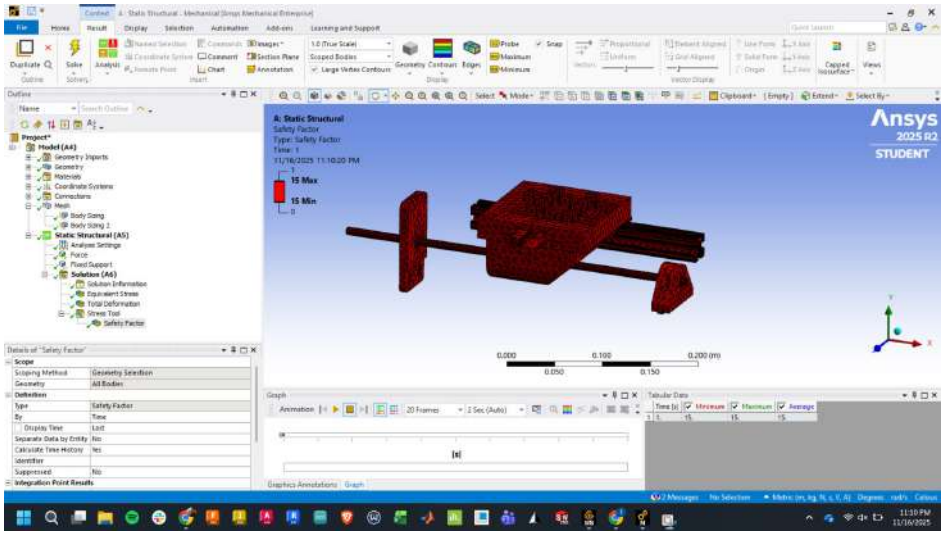


Figure 4: Ansys Factor of Safety distribution.

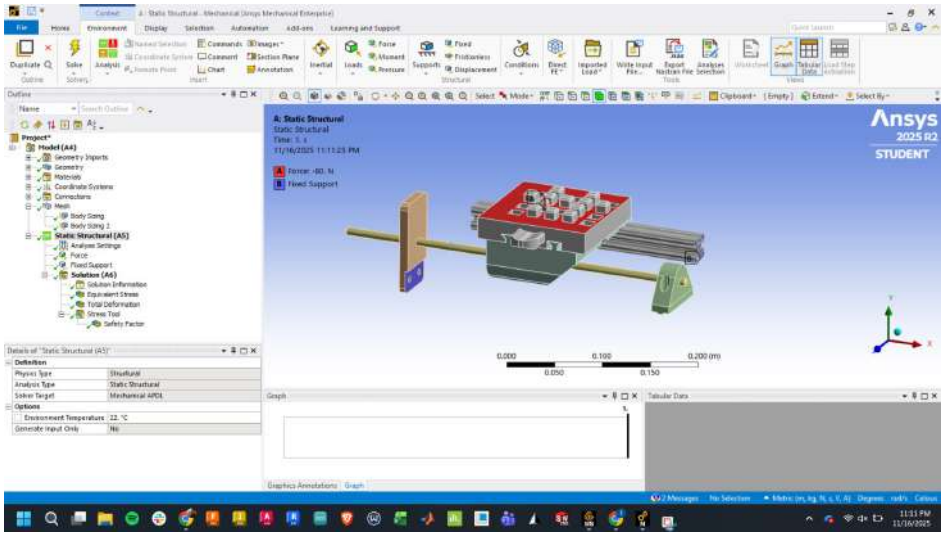


Figure 5: Ansys Model setup showing boundary conditions and loading.

5 Numerical Results

4-Block Case

Quantity	Value	Unit
Required piston force	74.88	N
Required piston area	0.210	in^2
80/20 beam stress	1.64	MPa
80/20 beam deflection	0.010	mm
80/20 beam FOS	105.05	-
Lead screw stress	56.79	MPa
Lead screw deflection	0.570	mm
Lead screw FOS	4.44	-

6-Block Case (Limiting)

Quantity	Value	Unit
Required piston force	112.32	N
Required piston area	0.316	in^2
80/20 beam stress	2.46	MPa
80/20 beam deflection	0.015	mm
80/20 beam FOS	70.03	-
Lead screw stress	85.19	MPa
Lead screw deflection	0.855	mm
Lead screw FOS	2.96	-

6 Equations Used

This section summarizes the analytical equations used in the MATLAB model to compute piston force, piston area, beam stresses, and deflections for both the 80/20 beam and the lead screw. All expressions assume linear elasticity and small deflection theory.

6.1 Piston Force and Area

The total required pressing force is computed from the per-block force multiplied by the number of blocks:

$$F_{\text{total}} = F_{\text{press}} \cdot N_{\text{blocks}}.$$

Using the maximum available piston pressure, the required piston area is:

$$A_{\text{piston}} = \frac{F_{\text{total}}}{P_{\text{max}}}.$$

6.2 Load Sharing

Assumption: The total piston load is assumed to split evenly between the 80/20 beam and the lead screw:

$$F_{\text{each}} = \frac{F_{\text{total}}}{2}.$$

6.3 Beam Bending (80/20 Aluminum Beam)

The beam is modeled as a simply supported beam with a center point load.

Deflection:

$$\delta_{\text{beam}} = \frac{F_{\text{each}} L_{\text{beam}}^3}{48 E_{\text{beam}} I_{\text{beam}}}.$$

Maximum Bending Moment:

$$M_{\text{max,beam}} = \frac{F_{\text{each}} L_{\text{beam}}}{4}.$$

Bending Stress:

$$\sigma_{\text{beam}} = \frac{M_{\text{max,beam}} c_{\text{beam}}}{I_{\text{beam}}},$$

where $c_{\text{beam}} = \frac{h_{\text{beam}}}{2}$.

Factor of Safety:

$$\text{FOS}_{\text{beam}} = \frac{\sigma_{\text{yield, beam}}}{\sigma_{\text{beam}}}.$$

6.4 Lead Screw Bending (8 mm SUS316 Shaft)

The lead screw is modeled as a simply supported circular beam under a center load.

Second Moment of Area:

$$I_{\text{screw}} = \frac{\pi d^4}{64}.$$

Deflection:

$$\delta_{\text{screw}} = \frac{F_{\text{each}} L_{\text{screw}}^3}{48 E_{\text{screw}} I_{\text{screw}}}.$$

Maximum Bending Moment:

$$M_{\text{max,screw}} = \frac{F_{\text{each}} L_{\text{screw}}}{4}.$$

Bending Stress:

$$\sigma_{\text{screw}} = \frac{M_{\text{max,screw}} c_{\text{screw}}}{I_{\text{screw}}},$$

where $c_{\text{screw}} = \frac{d}{2}$.

Factor of Safety:

$$\text{FOS}_{\text{screw}} = \frac{\sigma_{\text{yield, screw}}}{\sigma_{\text{screw}}}.$$

6.5 Unit Conversions

$$1 \text{ psi} = 6894.76 \text{ Pa}, \qquad 1 \text{ in}^2 = 1550.003 \text{ mm}^2.$$

7 MATLAB Code (used to generate results)

```
clc;
clear;
close all;

%% --- Constants ---
E_blocks = 2400e6;              % Elastic modulus of block material (Pa)
I_block = 8.864e-12;            % Moment of inertia of block interface (m^4)
L_block = 0.0053;               % Beam length (m)
COF = 0.2;                      % Coefficient of friction between blocks
Internal_length = 0.01265;      % Single internal length (m)
blocks_range = 4:1:6;           % Number of blocks varied
P_max_psi = 80;                 % Max piston pressure (psi)
P_max_Pa = P_max_psi * 6894.76; % Convert psi to Pa

```

```
%% --- 80/20 Beam Properties ---
I_beam_cm4 = 1.839743;           % Given in cm^4
I_beam = I_beam_cm4 * 1e-8;      % Convert to m^4
L_beam = .254;                   % Beam length (m)
h_beam_in = 1;                   % Cross-section height (inch)
h_beam = h_beam_in * 0.0254;     % Convert to m
E_beam = 69e9;                   % Approx. Aluminum modulus (Pa)
yield_beam = 172.4e6;            % Yield strength (Pa)

%% --- Lead Screw Properties ---
d_screw = 8e-3;                  % Diameter (m)
I_screw = (pi * d_screw^4) / 64; % Second moment of area (m^4)
E_screw = 193e9;                 % SUS316 modulus (Pa)
yield_screw = 252e6;             % Yield strength (Pa)
L_screw = .305;                  % Same span (m)

%% --- Preallocate Arrays ---
F_cylinder = zeros(size(blocks_range));
A_piston_m2 = zeros(size(blocks_range));
A_piston_in2 = zeros(size(blocks_range));
defl_beam_mm = zeros(size(blocks_range));
stress_beam_MPa = zeros(size(blocks_range));
defl_screw_mm = zeros(size(blocks_range));
stress_screw_MPa = zeros(size(blocks_range));

%% --- Calculations ---
for i = 1:length(blocks_range)
    blocks = blocks_range(i);

    % --- Force from block press (assumed)
    F_press = 18.72;

    % --- Total required piston force ---
    F_total = F_press * blocks;
    F_cylinder(i) = F_total;

    % --- Required piston area (for 80 psi) ---
    A_piston_m2(i) = F_total / P_max_Pa;
    A_piston_in2(i) = A_piston_m2(i) * 1550.003; % Convert to in²

    % --- Split total load equally between beam and lead screw ---
    F_each = F_total / 2;

    %% --- Beam Bending (simply supported, center load) ---
    defl_beam = (F_each * L_beam^3) / (48 * E_beam * I_beam); % m
    M_max_beam = (F_each * L_beam) / 4;                       % Nm
    stress_beam = (M_max_beam * (h_beam / 2)) / I_beam;       % Pa

```

```
    defl_beam_mm(i) = defl_beam * 1000;                          % mm
    stress_beam_MPa(i) = stress_beam / 1e6;                      % MPa

    %% --- Lead Screw Bending (same length, circular cross section) ---
    defl_screw = (F_each * L_screw^3) / (48 * E_screw * I_screw); % m
    M_max_screw = (F_each * L_screw) / 4;                        % Nm
    stress_screw = (M_max_screw * (d_screw / 2)) / I_screw;      % Pa

    defl_screw_mm(i) = defl_screw * 1000;                        % mm
    stress_screw_MPa(i) = stress_screw / 1e6;                    % MPa
end

%% --- Display Results for 4-Block Case ---
fprintf('--- Choosen Case (4 Blocks) ---\n');
fprintf('Required Piston Force: %.2f N\n', F_cylinder(1));
fprintf('Required Piston Area: %.3f in²\n', A_piston_in2(1));
fprintf('\n80/20 Beam:\n');
fprintf('  Stress: %.2f MPa | Deflection: %.3f mm | FOS: %.2f\n', ...
    stress_beam_MPa(1), defl_beam_mm(1), yield_beam/(stress_beam_MPa(1)*1e6));
fprintf('\nLead Screw:\n');
fprintf('  Stress: %.2f MPa | Deflection: %.3f mm | FOS: %.2f\n\n', ...
    stress_screw_MPa(1), defl_screw_mm(1),
    ↪ yield_screw/(stress_screw_MPa(1)*1e6));

%% --- Display Results for Limiting Case (6 Blocks) ---
fprintf('--- Limiting Case (6 Blocks) ---\n');
fprintf('Required Piston Force: %.2f N\n', F_cylinder(end));
fprintf('Required Piston Area: %.3f in²\n', A_piston_in2(end));
fprintf('\n80/20 Beam:\n');
fprintf('  Stress: %.2f MPa | Deflection: %.3f mm | FOS: %.2f\n', ...
    stress_beam_MPa(end), defl_beam_mm(end),
    ↪ yield_beam/(stress_beam_MPa(end)*1e6));
fprintf('\nLead Screw:\n');
fprintf('  Stress: %.2f MPa | Deflection: %.3f mm | FOS: %.2f\n', ...
    stress_screw_MPa(end), defl_screw_mm(end),
    ↪ yield_screw/(stress_screw_MPa(end)*1e6));

%% --- Plots ---
figure;
tiledlayout(3,1);

% (1) Piston Force (with lbf secondary axis)
nexttile;
yyaxis left
plot(blocks_range, F_cylinder, 'b', 'LineWidth', 2);
ylabel('Piston Force (N)');
yyaxis right
plot(blocks_range, F_cylinder / 4.44822, 'r--', 'LineWidth', 2);
```

```
ylabel('Piston Force (lbf)');
xlabel('Number of Blocks');
title('Required Piston Force vs. Number of Blocks');
legend('Force (N)', 'Force (lbf)', 'Location','best');
grid on;

% (2) Piston Area
nexttile;
plot(blocks_range, A_piston_in2, 'LineWidth', 2);
ylabel('Piston Area (in²)');
xlabel('Number of Blocks');
title('Required Piston Area vs. Number of Blocks');
grid on;

% (3) Beam and Screw Deflection/Stress
nexttile;
yyaxis left
plot(blocks_range, defl_beam_mm, 'b', 'LineWidth', 2); hold on;
plot(blocks_range, defl_screw_mm, 'g--', 'LineWidth', 2);
ylabel('Deflection (mm)');

yyaxis right
plot(blocks_range, stress_beam_MPa, 'r', 'LineWidth', 2);
plot(blocks_range, stress_screw_MPa, 'm--', 'LineWidth', 2);
ylabel('Stress (MPa)');
xlabel('Number of Blocks');
title('Beam and Lead Screw Deflection & Stress vs. Number of Blocks');
legend('Beam Deflection','Screw Deflection','Beam Stress','Screw
↪  Stress','Location','best');
grid on;

sgtitle('Piston, Beam, and Lead Screw Analysis');
```

Analysis: Flexibend Design Analysis

Team: 1402

Project: UMass FMS

Prepared by: Tiernan O'Kane

Date: May 8, 2026

1 Overview

This analysis evaluates a cantilever beam used as a compliant pushthrough mechanism. The beam is modeled using the pseudo-rigid-body (PRB) formulation from the *Handbook of Compliant Mechanisms*.

A full design sweep is performed over beam thickness and load position to determine combinations that satisfy:

- Maximum allowable pushthrough force
- Minimum support force
- Maximum allowable rotation
- Yield stress constraint

2 Free Body Diagram

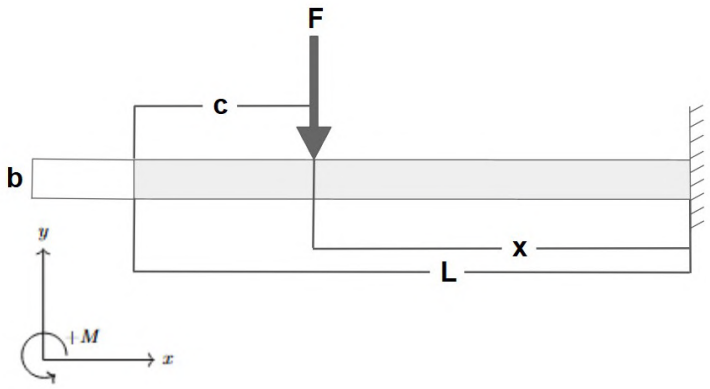


Figure 1: Free body diagram of cantilever beam showing applied force at distance d from the fixed wall.

3 Pseudo-Rigid-Body Model

The beam is modeled using the PRB approximation:

$$K = 2.258\frac{EI}{L}$$

where the second moment of area for a rectangular beam is:

$$I = \frac{bh^3}{12}$$

The geometric relations for tip position are:

$$a = L\left[1 - 0.85(1 - \cos\theta)\right]$$

$$b = 0.85L\sin\theta$$

For accurate force prediction:

$$\theta < 58.5^\circ$$

Pushthrough force is computed from:

$$F_{pt} = \frac{K\theta}{0.85L\cos\theta}$$

Maximum bending stress at the fixed end:

$$\sigma_{max} = \frac{Pdc}{I}$$

where $c = h/2$ for a rectangular beam.

4 Design Variables

The MATLAB script evaluates a 2D design space:

- Beam thickness: $0.3mm - 3.0mm$
- Load position ratio: $d/L = 0.2$–0.8

5 Design Constraints

A design is feasible only if:

- $F_{pt} \leq 80$ N
- $F_{support} \geq 1$ N
- $\theta_{pt} \leq 58^\circ$
- $\sigma_{max} \leq 10$ MPa

A feasibility mask is generated to identify valid designs.

6 Results

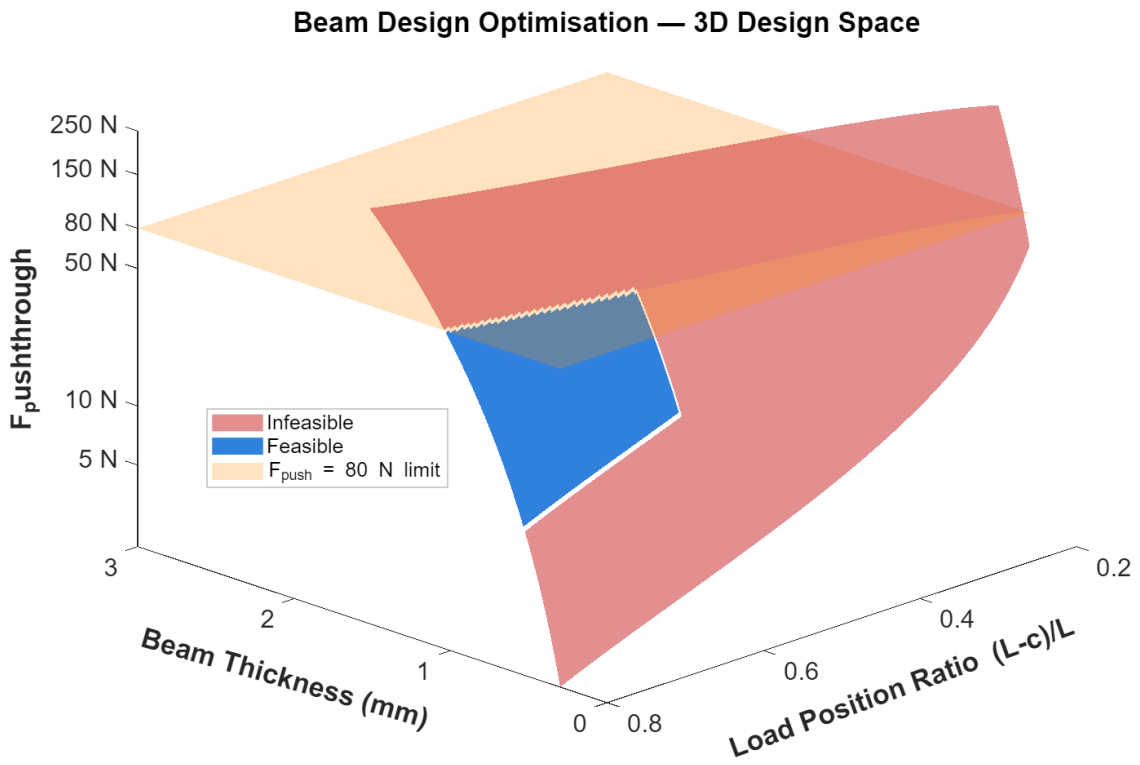


Figure 2: 3D design Curve of Flexibend Wing designs with feasible designs highlighted in blue and a yellow maximum piston force cap is shown.

7 ANSYS Finite Element Verification

To validate the analytical pseudo-rigid-body model predictions, a nonlinear finite element analysis was performed in ANSYS Mechanical.

7.1 Simulation Setup

- **Analysis Type:** Nonlinear static structural analysis with large deflection enabled.
- **Material Model:** Mooney-Rivlin hyperelastic model for TPU (thermoplastic polyurethane), selected to capture the nonlinear stress-strain behavior of the elastomeric wing material under large deformations.
- **Solver:** Nonlinear solver with incremental load stepping to ensure convergence through the full range of deformation.
- **Boundary Conditions:** A displacement boundary condition was enforced on the rigid block, driving it through the TPU wing geometry. The wing base was fixed (fully constrained). The block displacement was ramped incrementally, causing progressive deflection of the compliant wing.

- **Contact:** Frictional or frictionless contact was defined between the rigid block and the deformable TPU wing surface.

7.2 Loading Methodology

Rather than applying a direct force, the simulation enforced a prescribed displacement on the block. As the block pushed through the wing, the resulting reaction forces at the contact interface were recorded. This displacement-controlled approach provides stable convergence for highly nonlinear, large-deformation problems and directly mimics the physical pushthrough scenario.

7.3 Results

The reaction force history is shown in Figure 3. The graph plots the reaction forces in the x-direction, y-direction, and the resultant force magnitude as a function of simulation time (load step progression). Key observations:

- Prior to $t = 1.0$ s (the contact initiation point), reaction forces remain at zero as the block has not yet engaged the wing.
- After $t = 1.0$ s, the enforced displacement drives the block into the wing, generating increasing reaction forces.
- The x- and y-components of the reaction force evolve nonlinearly as the wing undergoes large deflection.
- The resultant combined reaction force reaches a magnitude of approximately **1.76 N** (total across the simulation).

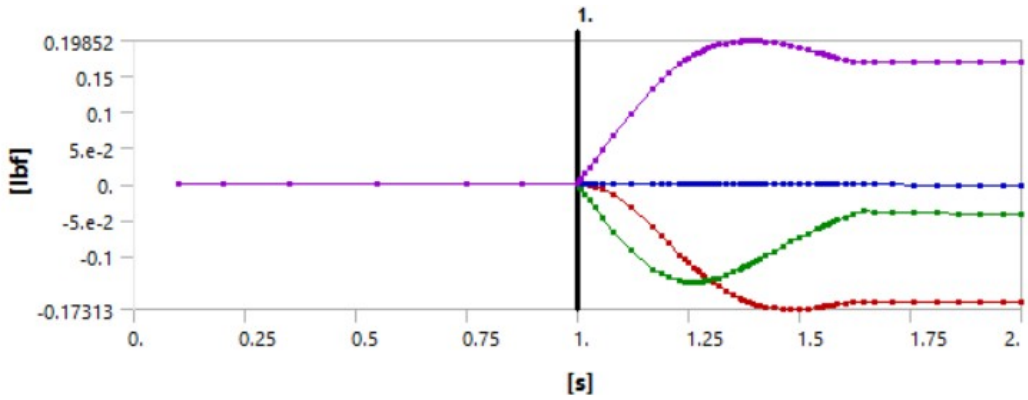


Figure 3: ANSYS force reaction results: reaction forces in the x-direction (red), y-direction (green), and resultant magnitude (purple) versus simulation time. The vertical line at $t = 1.0$ s indicates the onset of block-wing contact and displacement enforcement.

7.4 Validation

The ANSYS simulation validated that the flexibend wing provides a total resistance force of approximately:

$$F_{total} \approx 1.76 \text{ N}$$

This total force is split across two symmetric wings in the assembly, yielding a per-wing resistance of:

$$F_{wing} = \frac{F_{total}}{2} \approx 0.88 \text{ N per wing}$$

The nonlinear FEA results using the Mooney-Rivlin hyperelastic model confirm that the analytical PRB model provides a reasonable approximation of the wing behavior. The agreement between the simplified analytical approach and the full nonlinear FEA lends confidence to the design space exploration performed in the MATLAB parametric study.

7.5 Mooney-Rivlin Material Model

The Mooney-Rivlin strain energy function used in the simulation is defined as:

$$W = C_{10}(I_1 - 3) + C_{01}(I_2 - 3) + \frac{1}{d}(J - 1)^2$$

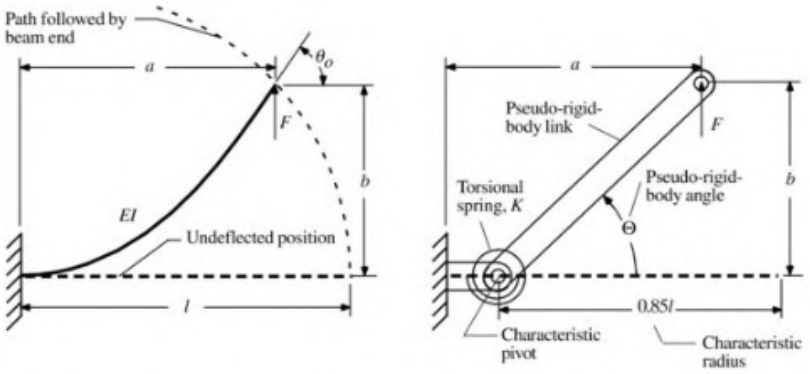
where C_{10} and C_{01} are material constants determined from TPU material characterization data, I_1 and I_2 are the first and second invariants of the deviatoric left Cauchy-Green deformation tensor, J is the determinant of the deformation gradient, and d is the material incompressibility parameter. This model is well-suited for capturing the moderate-strain hyperelastic behavior exhibited by TPU during the wing deflection.

8 Assumptions

- Linear elastic material behavior (analytical model).
- Pseudo-rigid-body approximation is valid for $\theta < 58°$.
- Rectangular cross-section remains constant.
- Stress evaluated at the fixed end.
- No plastic deformation or fatigue effects included.
- ANSYS verification uses hyperelastic Mooney-Rivlin model to capture nonlinear TPU behavior beyond the linear elastic assumption.

9 Reference Figure (PRB Model)

Figure A.5.2 Pseudo-rigid-body model of a vertical force at the free end of a cantilever beam



(A.4) $a = l[1 - 0.85(1 - \cos\Theta)] \qquad b = 0.85l \sin\Theta$

(A.5) $\Theta < 64.3\,\text{deg}$ for accurate position prediction

(A.6) $\theta_o = 1.24\Theta \qquad K = 2.258\dfrac{EI}{l}$

(A.7) $\Theta < 58.5\,\text{deg}$ for accurate force prediction

(A.8) $$F = \frac{K\Theta}{\gamma l \cos\Theta}$$

(A.9) $\sigma_{max} = \dfrac{Pac}{I}$ at the fixed end

where c is the distance from the neutral axis to the outer surface of the beam (i.e. half the beam height for rectangular beams, the radius of circular cross section beams, etc.)

Figure 4: Pseudo-rigid-body model equations from the Handbook of Compliant Mechanisms.

10 MATLAB Code

```
clear; clc; close all;

% Inputs
beam_length = 0.01;          % meters (10 mm)
beam_width  = 0.64;           % meters (.64 mm)
E           = 0.055e9;       % Pa (55 MPa)

F_push_allowed  = 80;            % N   - pushthrough force
F_hold_needed   = 1;             % N   - support force needed
```

```
theta_pt_max    = 58 * (pi/180); % rad - allowable pushthrough angle (58 deg)
sigma_yield     = 10e6;          % Pa  - yield stress (10 MPa)

theta_support = 5 * (pi/180);   % rad - support angle (5 deg)

% Design Space Ranges
% beam_thickness: 0.3 mm to 3 mm
thickness_vec = linspace(0.3e-3, 3e-3, 300);

% beam_length_ratio: 0.2 to 0.8 (load between wall and tip)
ratio_vec = linspace(0.2, 0.8, 300);

% Output Grids
[TH, RA] = meshgrid(thickness_vec, ratio_vec);

F_pushthrough_grid = NaN(size(TH));
F_support_grid     = NaN(size(TH));
theta_pt_grid      = NaN(size(TH));
sigma_grid         = NaN(size(TH));

% Design Space
for i = 1:numel(TH)
    h = TH(i);                    % beam thickness (m)
    b = beam_width;               % beam width (m)
    L = beam_length;              % beam length (m)
    beam_length_ratio = RA(i);

    d = L * beam_length_ratio;    % force distance from wall

    beam_I = (b * h^3) / 12; % Moment of inertia

    K = 2.258 * (E * beam_I) / L; % Stiffness constant

    arg_push = 1 - (1 - (d/L)) / 0.85; % Pushthrough condition
    if arg_push < -1 || arg_push > 1
        continue;   % outside valid acos domain, skip
    end
    theta_pt = acos(arg_push);

    cos_pt = cos(theta_pt);
    if abs(cos_pt) < 1e-10
        continue;   % avoid division by zero
    end
    F_pt = (K * theta_pt) / (0.85 * L * cos_pt);

    % Bending stress at pushthrough
    sigma = (F_pt * d * (h/2)) / beam_I;
```

```
    cos_sp = cos(theta_support); % Support condition
    F_sp = (K * theta_support) / (0.85 * L * cos_sp);

    F_pushthrough_grid(i) = F_pt;
    F_support_grid(i)     = F_sp;
    theta_pt_grid(i)      = theta_pt;
    sigma_grid(i)         = sigma;
end

% Feasibility Mask
% All four criteria must be satisfied:
%   1. F_pushthrough <= 80 N
%   2. F_support     >= 1 N
%   3. theta_pushthrough <= 58 deg
%   4. sigma <= sigma_yield (10 MPa)

feasible = (F_pushthrough_grid <= F_push_allowed)  & ...
           (F_support_grid     >= F_hold_needed)    & ...
           (theta_pt_grid      <= theta_pt_max)     & ...
           (sigma_grid         <= sigma_yield);

% Diagnostics
fprintf('--- Force & Stress Diagnostics ---\n');
fprintf('F_pushthrough range:  %.4f to %.4f N\n',
↪  min(F_pushthrough_grid(:),[],'omitnan'),
↪  max(F_pushthrough_grid(:),[],'omitnan'));
fprintf('F_support range:      %.4f to %.4f N\n',
↪  min(F_support_grid(:),[],'omitnan'),
↪  max(F_support_grid(:),[],'omitnan'));
fprintf('theta_pt range:       %.2f to %.2f deg\n',
↪  min(theta_pt_grid(:),[],'omitnan')*180/pi,
↪  max(theta_pt_grid(:),[],'omitnan')*180/pi);
fprintf('Sigma range:          %.4f to %.4f MPa\n',
↪  min(sigma_grid(:),[],'omitnan')/1e6,
↪  max(sigma_grid(:),[],'omitnan')/1e6);
fprintf('\nConstraint limits:\n');
fprintf('  F_push_allowed:  %.1f N\n',  F_push_allowed);
fprintf('  F_hold_needed:   %.1f N\n',  F_hold_needed);
fprintf('  theta_pt_max:    %.1f deg\n',theta_pt_max*180/pi);
fprintf('  sigma_yield:     %.1f MPa\n',sigma_yield/1e6);
fprintf('\nPoints passing each constraint:\n');
fprintf('  F_push <= 80N:       %d / %d\n', sum(F_pushthrough_grid(:) <=
↪  F_push_allowed, 'omitnan'), sum(~isnan(F_pushthrough_grid(:))));
fprintf('  F_support >= 1N:     %d / %d\n', sum(F_support_grid(:)     >=
↪  F_hold_needed,  'omitnan'), sum(~isnan(F_support_grid(:))));
fprintf('  theta_pt <= 58deg:   %d / %d\n', sum(theta_pt_grid(:)      <=
↪  theta_pt_max,   'omitnan'), sum(~isnan(theta_pt_grid(:))));
```

```
fprintf('  sigma <= 10MPa:      %d / %d\n', sum(sigma_grid(:)         <=
↪  sigma_yield,    'omitnan'), sum(~isnan(sigma_grid(:))));
fprintf('  Feasible (all):      %d / %d\n', sum(feasible(:)), numel(feasible));

% Feasible Design Range Output
fprintf('\n--- Feasible Design Ranges ---\n');
if sum(feasible(:)) == 0
    fprintf('No feasible designs found in the current design space.\n');
else
    feasible_h = TH(feasible) * 1000;   % convert to mm
    feasible_r = RA(feasible);

    fprintf('Beam Thickness (h):\n');
    fprintf('  Min: %.3f mm\n', min(feasible_h));
    fprintf('  Max: %.3f mm\n', max(feasible_h));

    fprintf('Load Position Ratio (d/L):\n');
    fprintf('  Min: %.3f\n', min(feasible_r));
    fprintf('  Max: %.3f\n', max(feasible_r));

    fprintf('F_pushthrough in feasible region:\n');
    fprintf('  Min: %.3f N\n', min(F_pushthrough_grid(feasible)));
    fprintf('  Max: %.3f N\n', max(F_pushthrough_grid(feasible)));

    fprintf('F_support in feasible region:\n');
    fprintf('  Min: %.3f N\n', min(F_support_grid(feasible)));
    fprintf('  Max: %.3f N\n', max(F_support_grid(feasible)));

    fprintf('theta_pushthrough in feasible region:\n');
    fprintf('  Min: %.2f deg\n', min(theta_pt_grid(feasible))*180/pi);
    fprintf('  Max: %.2f deg\n', max(theta_pt_grid(feasible))*180/pi);

    fprintf('Bending stress in feasible region:\n');
    fprintf('  Min: %.4f MPa\n', min(sigma_grid(feasible))/1e6);
    fprintf('  Max: %.4f MPa\n', max(sigma_grid(feasible))/1e6);
end

% Split grids for plotting
F_push_feasible   = F_pushthrough_grid;
F_push_infeasible = F_pushthrough_grid;
F_push_feasible(~feasible)  = NaN;
F_push_infeasible(feasible) = NaN;

% Convert forces to dB scale (ref = 1 N)
F_push_dB            = 20*log10(F_pushthrough_grid);
F_push_feasible_dB   = 20*log10(F_push_feasible);
F_push_infeasible_dB = 20*log10(F_push_infeasible);
F_push_allowed_dB    = 20*log10(F_push_allowed);
```

```
F_cap_dB             = 20*log10(250);   % display cap

zlims = [min(F_push_dB(:), [], 'omitnan'), F_cap_dB];

% 3D Design Space Plot
figure('Color','k','Position',[100 100 950 700]);

colormap(gca, parula);
hold on;

% Feasible region - coloured by F_pushthrough magnitude
h_surf = surf(TH*1000, RA, F_push_feasible_dB, ...
    'FaceColor',[0.2 0.6 1.0], 'EdgeColor','none', 'FaceAlpha', 0.9);

% Infeasible region - flat red
surf(TH*1000, RA, F_push_infeasible_dB, ...
    'FaceColor', [0.75 0.15 0.15], 'EdgeColor','none', 'FaceAlpha', 0.55);

% F_pushthrough = 80 N ceiling plane
[Xp, Yp] = meshgrid(linspace(min(thickness_vec)*1000, max(thickness_vec)*1000,
↪  2), ...
                    linspace(min(ratio_vec), max(ratio_vec), 2));
surf(Xp, Yp, ones(size(Xp))*F_push_allowed_dB, ...
    'FaceColor',[1 0.6 0], 'EdgeColor','none', 'FaceAlpha', 0.25);


xlabel('Beam Thickness (mm)', 'Color','w', 'FontSize',12, 'FontWeight','bold');
ylabel('Load Position Ratio  d/L', 'Color','w', 'FontSize',12,
↪  'FontWeight','bold');
zlabel('F_{pushthrough} (dB re 1N)', 'Color','w', 'FontSize',13,
↪  'FontWeight','bold');
title({'Beam Design Optimisation | 3D Design Space', ...
       'Feasibility: F_{push} \leq 80N | F_{support} \geq 1N | \theta_{pt} \leq
       ↪  58\circ | \sigma \leq 10 MPa'}, ...
      'Color','w','FontSize',12);

ax = gca;
ax.Color     = [0.08 0.08 0.12];
ax.XColor    = 'w';
ax.YColor    = 'w';
ax.ZColor    = 'w';
ax.GridColor = [0.4 0.4 0.4];
ax.GridAlpha = 0.4;
ax.FontSize  = 10;
ax.View      = [225, 28];
zlim(zlims);

% Readable N-value tick labels on dB z-axis
```

```
ref_N  = [0.1, 0.5, 1, 5, 10, 50, 80, 150, 250];
ref_dB = 20*log10(ref_N);
ref_dB = ref_dB(ref_dB >= zlims(1) & ref_dB <= zlims(2));
ref_N  = 10.^(ref_dB/20);
ax.ZTick      = ref_dB;
ax.ZTickLabel = arrayfun(@(n) sprintf('%.0f N', n), ref_N, 'UniformOutput',
↪  false);

% Legend proxies
p1 = patch(NaN,NaN,[0.75 0.15 0.15],'FaceAlpha',0.55,'EdgeColor','none');
p2 = patch(NaN,NaN,[0.2  0.6  1.0 ],'FaceAlpha',0.9, 'EdgeColor','none');
p3 = patch(NaN,NaN,[1    0.6  0   ],'FaceAlpha',0.25,'EdgeColor','none');
legend([p1 p2 p3], ...
    'Infeasible', ...
    'Feasible design space', ...
    'F_{push} = 80 N limit', ...
    'TextColor','w','Color',[0.12 0.12 0.18],'EdgeColor',[0.4 0.4 0.4], ...
    'Location','northeast','FontSize',9);

hold off;
```

Analysis: Spring Force Requirement

Team: 1402

Project: UMass FMS

Prepared by: Tiernan O'Kane

Date: March 4, 2026

1 Overview

This analysis determines the minimum force required per spring to prevent sliding of stacked blocks within the system. The required spring force must overcome friction generated by the total normal force from the stacked blocks and plunger weight. A conservative factor of safety (FOS) is applied to account for uncertainty in friction, spring behavior, dynamic effects, and tolerances.

2 Methodology

The total normal force is calculated from the weight of all blocks plus the plunger:

$$N = n_{\text{blocks}} W_{\text{block}} + W_{\text{plunger}}$$

The friction force resisting motion is:

$$F_{\text{friction}} = \mu N$$

A combined factor of safety is applied:

$$FOS = \lceil k_{\text{friction}} \cdot k_{\text{spring}} \cdot k_{\text{dynamic}} \cdot k_{\text{tol}} \rceil$$

The total required spring force becomes:

$$F_{\text{total}} = \mu N \cdot FOS$$

Since two springs are used, the required force per spring is:

$$F_{\text{spring}} = \frac{F_{\text{total}}}{n_{\text{springs}}}$$

3 Inputs

Parameter	Value
Block Weight	80 g
Plunger Weight	70 g
Number of Blocks	5
Number of Springs	2
Coefficient of Friction (μ)	0.20
Combined Factor of Safety	3

(The calculated FOS rounds up to 3.)

4 Results

The MATLAB script computes:

$$F_{\text{spring}} \approx 2.83 \text{ N}$$

$$F_{\text{spring}} \approx 0.64 \text{ lbf}$$

This represents the minimum force each spring must provide to reliably prevent block sliding under worst-case conditions.

5 Assumptions

- Static friction governs the system.
- All blocks are fully loaded simultaneously.
- The load is evenly distributed between both springs.
- Gravity is constant at 9.81 m/s^2.
- Factors of safety adequately capture uncertainty in dynamics and manufacturing tolerances.

6 MATLAB Code

```
clear; clc; close all;

% Inputs
W_block_g   = 80;
W_plunger_g = 70;
n_blocks    = 5;
n_springs   = 2;
mu          = 0.20;

k_friction  = 1.30;
k_spring    = 1.20;
k_dynamic   = 1.20;
k_tol       = 1.05;

FOS = ceil(k_friction * k_spring * k_dynamic * k_tol);

g = 9.81;

W_block   = (W_block_g   / 1000) * g;
W_plunger = (W_plunger_g / 1000) * g;
```

```
Normal_force = n_blocks * W_block + W_plunger;

F_total_required = mu * Normal_force * FOS;

F_per_spring = F_total_required / n_springs;

fprintf('Minimum required force per spring:\n');
fprintf('%.4f N\n', F_per_spring);
fprintf('%.4f lbf\n', F_per_spring * 0.224809);
```