

# EN4 Dynamics and Vibrations

## Design Project

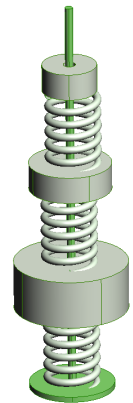
### Optimizing a dynamically tuned projectile launcher

#### Synopsis

In this project you will use MATLAB simulations to design a simple system to launch a mass to the maximum possible height. You will assemble and test the device, and measure the velocity of the projectile using a high-speed camera and some MATLAB image processing.

#### 1. Organization

- This is a group project. You can form your own groups, or ask to be assigned to a random group.
- The project deliverables are:
  - (i) A MATLAB code that will predict the optimal combination of masses and springs;
  - (ii) A MATLAB code that will read a 'csv' file of the position of each mass in the system as a function of time (measured using a high-speed camera), determine and plot the velocity of each mass as a function of time, and compare the measured and predicted velocities;
  - (iii) A 3-4 page report summarizing your design calculations and the final design.
- You will assemble and test the design. Test procedures will be announced once COVID restrictions are known. During the test, you will
  - (i) Find the correct springs and masses for your design and assemble it;
  - (ii) Test the design, and record a high-speed video of the launch.
  - (iii) Use a MATLAB code provided on the course website to extract a csv file of the position of each mass as a function of time from the movie. *At least one member of your team should practice using this code before the day of the test, so you can run the code quickly on test day.*
  - (iv) Read the csv file into your own MATLAB code, and calculate and plot the velocities of the masses. You should compare the predicted and measured velocity of the projectile, and you can make your code calculate and plot any additional quantities that you think might be interesting.
  - (v) Give a prepared 5-10 min oral presentation describing your design; your design process; the performance of the design, and anything you learned from the experimental measurement.



#### 2. Overview of design requirements

The figure shows a simple design for launching a mass into the air. It consists of a set of springs and masses that are mounted on a steel rod. The springs and masses are free to slide up and down the rod. To launch the projectile, masses are dropped from an initial height  $h$  (in actual testing the bottom spring starts on the ground, and the lowest mass is dropped onto it). The springs make the masses rebound. If the masses and springs are selected carefully, a significant fraction of the energy in the system can be transferred to the small, topmost mass during the rebound. This will make the topmost mass fly off the rod at high speed.

Your goal in this project is to use MATLAB to simulate the motion of the spring-mass system, for the period of time between the instant that the lowest mass hits the lowest spring, and instant where the topmost mass loses contact with the spring beneath it. You can then use the simulation to predict the

combination of springs and masses that will maximize the velocity of the topmost mass (the projectile). You will then assemble and test the design, and see what velocity you actually get!

Your design must meet the following constraints

1. The total mass used must not exceed 8lb
2. The launched mass must exceed 0.1 lb
3. The height of the topmost point on the stack of springs/masses cannot exceed 38 inches when the assembly is dropped
4. Springs must be selected from the list below (you can find detailed specifications for each spring on <http://www.mcmaster.com/> - you can type in the part number to get an engineering drawing for each spring)
5. Masses must be selected from the list below. If you wish, you can stack masses on top of one another.

<b>LIST OF AVAILABLE SPRINGS (helical compression springs)</b>							
OD (in)	ID (in)	Length (in)	Max Load (lb)	Max Deflection (in)	Rate (lb/in)	Part #	Color Code (For test day)
0.75	0.566	5	39	2.64	15	9657K215	None
1.938	1.642	4	67	2.88	23.7	96485K125	Green/Blue
2.188	1.774	4	176	2.55	68	96485K154	Brown
2.906	2.344	5	260	2.89	90	96485K225	Red
2.188	1.688	4	274	2.06	134.4	96485K161	Yellow/Blue
2.688	2.064	5	445	2.57	175.3	96485K429	Black
2.438	1.814	4	470	1.74	270	96485K178	Yellow
2.906	2.156	5	675	2.14	312	96485K233	Blue
1.938	1.314	4	597	1.27	470	96485K384	Silver/Blue
2.438	1.688	4	786	1.28	617	96485K187	Green
2.188	1.438	4	907	1.09	831	96485K171	Yellow/Black
3.156	2.156	5	1438	1.5	959	96485K262	Silver
1.938	1.188	4	1162	1.27	1217.0	96485K147	Yellow/Silver
2.438	1.564	4	1232	0.94	1310	96485K194	Silver/Black

<b>LIST OF AVAILABLE MASSES (all tubular cylinders)</b>					
OD (in)	ID (in)	Length (in)	Weight (lb)	Part #	Cost (US\$)
1	0.48	0.7	0.117	90075K231	8.90 (for 12" bar)
2	0.75	0.125	0.110	Washer (not a McMaster part)	
2	0.5	0.41	0.117	8974K71	15.21 (for 6" bar)
2.5	1	0.125	0.168	Washer (not a McMaster part)	
1	0.48	1	0.182	90075K231	8.90 (for 12" bar)
2	0.48	1/2	0.417	7786T12	3.44
2	0.48	1	0.829	7786T14	5.17
3	0.48	1/2	1.00	8960K51	12.97
3	0.48	1	2.18	8960K52	16.74
3	0.48	1 1/2	3.04	8960K53	20.75
3	0.48	2	4.12	8960K54	24.97
3	0.48	3	5.84	4470T15	32.68

### 3. Design calculations

You will be designing a complicated dynamical system that can't easily be analyzed by hand. Computer-aided design can help you to solve this sort of problem. In this problem, you can use MATLAB to predict the launch velocity of the projectile for a given design. MATLAB can even search the design space for you and identify a candidate design.

You will need to write your MATLAB code in two steps (see Sect 3.4 for a code template)

1. You will need to write a function that takes as input variables values of the spring stiffnesses and masses, and predicts the launch velocity. To be compatible with the MATLAB optimizer, this function should have the form

```
function predicted_launch_vel = findlaunchvel(Design_Variables)
    Enter code to calculate the launch velocity here
end
```

Here, 'Design\_Variables' is a MATLAB vector that stores the values of mass and spring stiffness in your design – for example Design\_Variables could contain  $[m_1, m_2, m_3, k_1, k_2, k_3]$ , where  $m_i$  is the value of the  $i$ th mass; and  $k_i$  is the stiffness of the  $i$ th spring.

2. Once you can predict the launch velocity, you will need to write some more code to search for the optimal design. You can use the MATLAB optimizer to do this, or you can invent your own way to do the search (eg looping over all feasible designs – but you have to be smart about this otherwise you will graduate before the calculation finishes).

#### 3.1 Predicting the launch velocity

You can use the methods discussed in class to predict the launch velocity: (i) use Newton's law and the spring force law to derive differential equations that govern the position of each mass; and (ii) use the MATLAB ode45 function to solve the equations, and hence predict the launch velocity.

You might find it useful to follow the 2-storey building vibration example solved in class. Here are some rough guidelines:

1. The launch has 3 phases of motion (i) The period of free-fall before the masses land on the ground; (ii) the rebound, while some or all the springs are compressed; and (iii) the period after the topmost mass has been launched up to the point where it (hopefully) hits the ceiling (if we test indoors!). You can analyze (i) and (iii) by hand. Your MATLAB simulation should consider the motion of the masses during phase (ii), starting from the point where the masses land on the ground and just start to compress the springs. At this instant, all the masses are moving downwards with the same initial speed  $v_0$ , and the springs are free of force.
2. You might find it convenient to describe the motion of the masses using the displacement (i.e. the distance) of each mass from its position at the instant in step 1 (other choices are fine too but you might end up with slightly more complicated equations of motion).
3. Derive the equations of motion using the usual approach: draw a free body diagram showing forces acting on each mass; write down the forces using the spring force law, and use Newton's law. You will get one second order differential equation for each mass in your system. You will need to split each of these into two first order equations so MATLAB can solve them.
4. Write a MATLAB code that will solve the equations of motion you derived in step (3) using the ode45 solver, and hence determine the launch velocity  $v_1$ . Your code will look a bit like the solution to the 2-storey building problem solved in class.
5. Your code will need numbers – be very careful with units. The optimizer will work best if you specify the spring stiffness in lb/in and the masses in pounds, but you will need to convert the

- mass values to blobs (or slugs) when you substitute into the equations of motion, because lb is a unit of weight, not mass.
6. You will need to add an 'event' function to your MATLAB code to identify the point when the topmost mass leaves the launcher.
  7. Note that the ratio of launch speed to initial speed  $v_1 / v_0$  is independent of  $v_0$ , and also does not depend on the un-stretched lengths of the springs or the size of the masses. This means that if you maximize the ratio  $v_1 / v_0$  with the MATLAB optimizer, you can use any arbitrary value for  $v_0$  and the spring and mass geometry.
  8. The actual value of the initial velocity of the masses  $v_0$  can be determined from the distance that the masses drop before the launcher impacts the ground (either using straight-line motion, or energy methods). The drop distance will depend on the spring lengths and the size of the masses (the longer your device, the smaller the drop distance).

### 3.2 Using the MATLAB optimizer

If you like, you can use the MATLAB 'fmincon' function to predict the optimal design.

The MATLAB 'fmincon' function will search for the *minimum* value of a function (there is a simple way to fool the minimizer to make it maximize your launch velocity). It won't actually pick the best design for you, because it assumes that you can use any value of mass or spring stiffness, instead of the finite set you can actually choose from, but you can use it to suggest a good design and then select masses and springs close to the suggested values (as a check, you can then re-calculate the launch velocity for your actual design, and see if it is close to the theoretical optimal design).

You can find instructions for using the MATLAB optimizer in the MATLAB tutorial, Section 12, as well as the video online.

Note that the solution to this design problem is not unique – the optimizer may predict many different possible optimal solutions depending on the initial guess you give it (this means that not everyone will end up with the same design). They will all perform pretty well, but some are better than others. If you have time you could try to come up with more than one possible design.

You are welcome to use other methods of your own to optimize the design as well, if you prefer.

### 3.3 Final calculations.

Once you have selected a final design, you should

1. **Check that you can actually assemble the design and it will work, from the geometry information given in the two tables on page 2.** The OD in the first column of both tables stands for 'outer diameter', and 'ID' stands for 'Inner diameter' (i.e. the diameter of the hole in the middle of a mass, or the empty space inside a spring). If the 'OD' of a mass is less than the 'ID' of a spring, it will fall through the spring (You can fix this by putting one of the flat washers on top of the spring, and then putting your mass on top of the washer. Assume that the washer and the mass on top of it behave like a single mass. Similarly, if the ID of a mass is bigger than the OD of a spring, the spring will poke through the mass
2. Check to make sure that the compression of the springs does not exceed the maximum allowable value (if it does, you can always reduce the drop height). Note that the maximum allowable compression of the spring refers to the change in length of the spring, not the positions of the masses.

3. Predict the actual launch velocity you expect (you will compare the prediction with what actually happens on test day).
4. Calculate the expected 'dynamic efficiency' of the device (the ratio of the kinetic energy of the projectile to the initial potential energy of the system)
5. Plot a graph of the velocity and/or position of the masses as a function of time (you can compare this with experimental measurements as well)
6. Make any other predictions that you would be interested to compare with measured behavior...

### 3.4 Code template

A rough outline of a possible code that makes use of the MATLAB optimizer is shown below

```
function masslauncher_project
% This is the function that MATLAB will execute
1. Define values for variables
2. Set up the constraints for the optimization for the MATLAB optimizer (see the MATLAB tutorial)
3. Call the MATLAB optimizer (which will substitute different values for 'Design Variables' into the function below, and find the values that give the lowest value for launchvel). The output of the optimizer is a vector storing the values of mass and stiffness that minimize the launch velocity
4. Post-process results – simulate the behavior of the optimal design, plot graphs, calculate the efficiency, etc

end
function launchvel = findlaunchvel(Design_Variables)
% Function to calculate the launch velocity, given values
% for the masses and spring stiffnesses in the mass launcher
% Design_Variables is a vector that contains a list of the
% values of masses and stiffnesses, eg [m1,m2,m3...,k1,k2,k3...]
% It is best to work with m in lb and k in lb/in. NB – lb is not a
% of mass – you have to convert to slugs or blobs to do the calculation.
%
% This function must calculate a value for variable launchvel –
% this is the velocity of the topmost mass at the instant that it leaves the
% mass launcher. You will have to use a trick to make sure the optimizer
% maximizes, rather than minimizes, the launch velocity
1. Define values for masses and spring stiffnesses, eg m1 = Design_Variables(1), m2 = Design_Variables(2) etc
2. Do any necessary unit conversions (eg convert mass from lb to slugs or blobs); specify initial conditions; the time interval for the simulation, etc.
2. Call ode45 here. Don't forget to use the 'options' variable so matlab can detect the separation of the missile.
3. The ODE solver will give you a vector of time values, and a matrix of solution values, as usual. You will need to determine the launch velocity from the solution (it will be stored somewhere in the solution matrix).
4. Before running the optimization, it's a good idea to test this function by itself with some sensible guess for the design variables. You can plot the solution and make sure it looks sensible. If you use computer simulations to help with design, you always need to be very careful – a tiny error in the code can cause big problems!
end
function dwdt = diffeq(t,w, plus any variables eg m1,m2,m3,k1,k2,k3)
% Function to calculate the time derivative of the variables
% describing the position and velocity of the masses.
Enter code for your equations of motion here
end
function [eventvalue,stopthecalc,eventdirection] = event(t,w)
% Function to identify the point where the top mass is launched.
% You will have to use some physics to find a criterion to detect this point

Enter code to detect the separation here – make 'eventvalue' = a formula that goes to zero at separation.
end
```

#### 4. Writing MATLAB code to determine the velocity from experimental measurements

On test day, you will be provided with a high-speed camera that will record a movie of the launch. We have provided two simple MATLAB scripts that will extract frames from the movie, and then measure the position of a mass in the images using the image processing toolbox in matlab (if you would like a coding challenge, you could try to write your own version of this code, instead of using ours). The script will print a csv file that stores (in columns) a value for time, and the height of each mass, in successive frames.

You should write a MATLAB script that reads this file, and calculates and plots the velocity of each mass as a function of time (you can calculate and plot additional quantities as well if you like).

You can follow the procedure in [Section 2.7 of the lecture recordings to do this](#).

#### 5. Assembly and testing

Procedures for testing designs will be announced once COVID restrictions for Summer 2021 are known.

On test day:

- You will be provided with a set of springs and masses. The springs are marked with a color code so you can select the ones you need.
- You will be provided with the rod that holds the springs and masses – all you need to do is to assemble your design and test it.
- Run several tests and record movies of each one (to make sure you get data from at least one successful test)
- The movies will be stored on the memory card in the camera. You should be able to plug the memory card into your laptop and copy the movie files to your own computer.
- Run the image processing codes provided on the course website to generate a csv file of position-v-time. Be sure to practice using the codes before test day so you can run the test efficiently.
- Run your own code to determine the launch velocity, and plot any graphs that you would like to show during your presentation
- Give the presentation.

You should plan roughly on 10 mins for tests; 10 mins for processing the data; and 10 mins for the presentation, if all goes smoothly. But it's best not to schedule your test time right before another important meeting across campus in case something goes wrong to delay your test.

## 6. Presentation, Report and Grading Rubric

Before the test day, one member of your group should upload to CANVAS (**we only need one submission per group**):

1. A commented MATLAB .m file that you used to do the optimization calculations
2. A commented MATLAB .m file that processes the csv file with experimental measurements
3. A 3-4 page description of your design, which should include (a) A table of the predicted spring stiffnesses and masses (ii) A prediction of the launch velocity; (iii) An estimate of the *energy efficiency* of the device, i.e. the ratio of the kinetic energy of the projectile at the instant of launch to the initial potential energy of the system before the assembly is dropped; and (iv) an outline of the method you used to determine the optimal values of mass and spring stiffness.

**Make sure you bring a laptop to the project test with the image processing code from the course website preloaded on it, as well as your own code that will read the csv datafile and plot velocity-v-time for each mass in your launcher. Please have at least one of your team practice using these codes before the day of the test with the sample files on the course website.**

You will also give a short presentation to faculty or TAs presenting the merits of your design, describing how you did the optimization, and describing the results of your experimental measurements. You should plan the presentation ahead of time, so you know what each group member's role is, avoid repeating information, and keep the presentation on time. Power-point can be helpful.

### **Grading**

1. Matlab program to calculate the launch velocity; correctly programmed and clearly commented so the procedure can be understood: 10 points
2. Code to calculate velocity from experimental measurements: 5 points
3. Design report: Design specification reported clearly; reported choice clearly; includes all relevant information. 7 points
4. Oral presentation: able to explain clearly the calculations and optimization process, discuss performance of design and experimental results. 6 points
5. Assembly and testing of design; successful image processing and calculating actual launch velocity 6 points
6. Design performance – 1 point if you can hit the ceiling in BDW (if we test outdoors we will calculate the expected height based on the measured launch velocity and compare to the ceiling height)
7. Peer evaluation (for projects done in groups – students doing the project individually automatically receive full credit) 5 points

**TOTAL 40 POINTS**

The design report and oral presentation will be graded based not just on getting the correct answers to the calculations, but also on the quality of the report/presentation:

- Organize your report and presentation to communicate information about your design in the most effective manner possible – in this project, your design and its expected performance are the key deliverables, so make these the focus of your report or presentation. Use sections and headings to organize the information. Start the report with a short executive summary; give enough background that a reader unfamiliar with the project would understand what you did; report your findings, and include enough detail about your calculations that a reader will believe your conclusions.
- Please avoid hand-drawn sketches unless you are a visual arts expert and can produce something really professional – aim to at least match the quality of graphics we give you in homework problems and project descriptions; use an equation editor for equations, and be careful about details like labeling graphs, units for numbers, and defining variables in equations.

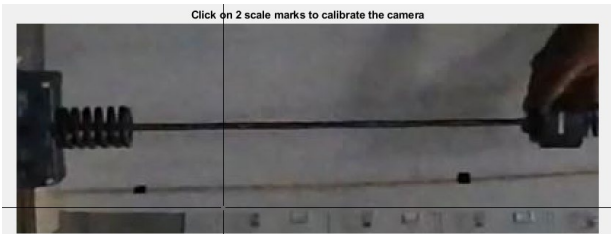
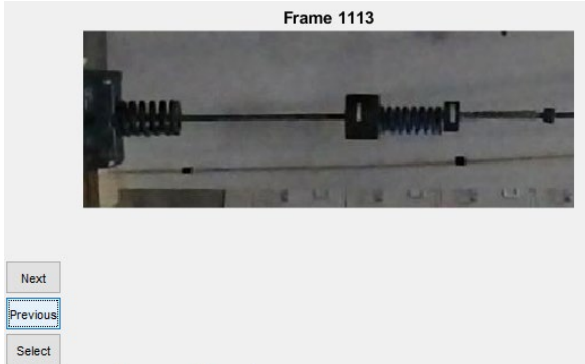
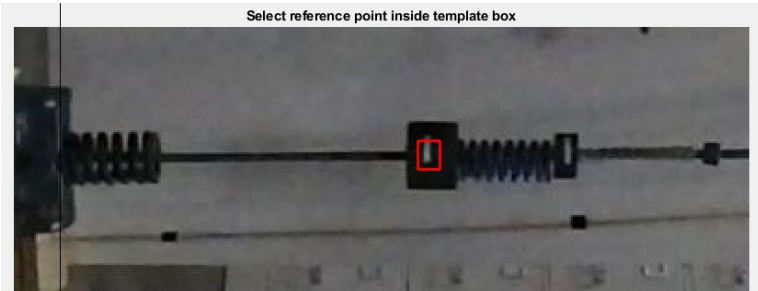


## Appendix 2: Using the image processing code (see also the Project 1 Video Presentation)

### 1. Convert the .mov file in to image files

- Download the MATLAB file called convertmovietoimages.m from the Projects page of the course website and store it in the directory where you will be doing your work
- Open the file and edit the directory/file names as needed
- Download one of the sample high-speed videos from the Projects page of the course website and store it with a .mov extension in the same directory as your convertmovietoimages code
- Run the convertmovietoimages script. It will take some time to complete.

### 2. Processing the image files

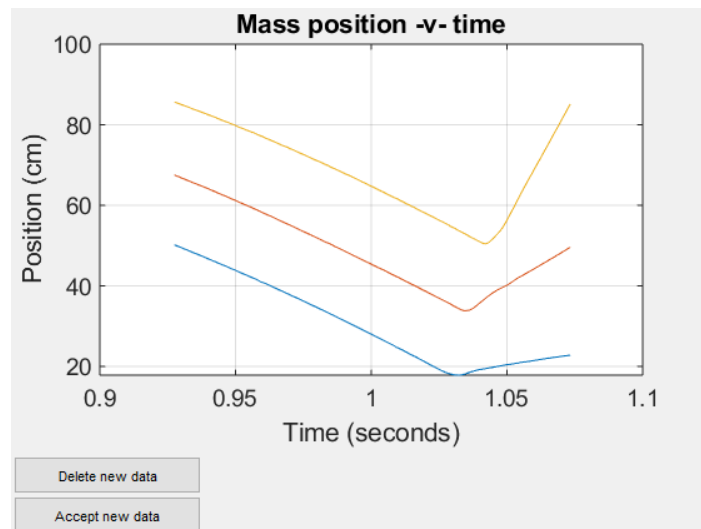
- Download the MATLAB file called track\_masses.m from the Projects page of the course website
- Open the file and (if necessary) change the name of the directory on line 15 to specify where the image files are stored, as well as the name of the csv data file that will be printed at the end
- Run the script.
- A window will open and show the first frame in the movie. You will see a bar next to the mass launcher with two black marks on it. These are spaced 50cm apart. Click on the left edge of each black bar (in any order) with the cross-hairs in the image window. The MATLAB code will count the number of pixels between the marks and hence calculate the conversion factor from image pixels to cm.
- The movie will start to play back. We only need to process a small part of the movie (the period of the impact). When you see all 3 masses appear in the image, press the button on the bottom of the window to stop the movie. You can use the buttons to scroll back or forward through the images until you find one that you would like to use as the start image. Once you are done press 'select'
- Repeat the process to select the last frame to be processed. Make sure the bounce has finished, but all 3 masses are still visible in the image (and preferably are still on the rod).
- A new window will open asking you to select a 'correlation template box.' This is a rectangular region that crops out a piece of the image on one of the masses that will look similar in all the frames of the movie (the larger masses have white strips on them that make convenient tracking markers; for the small mass might need to use the whole mass, or the upper half of the mass, together with a bit of the white background). MATLAB will search for similar



regions of the image in subsequent image frames, and hence track how the mass is moving. Use the cross-hairs to mark the top left, then bottom right corners of a rectangle surrounding a readily identifiable portion of a mass, as shown. In the example, the white strip on the big mass was selected, because we expect this to look the same in all frames of the movie. The sharp edges will help identify coordinates of the strip in each image. You can practice this step with different correlation windows until you get the best quality data.



- A message will appear on the top of the window asking you to select a reference point on the mass. You can pick any point inside the red rectangle you like – the code will plot and save the position of the point you have selected as a function of time (the origin for the coordinate system is at the top left of the image, with  $x$  horizontally, and  $y$  vertically downwards).
- The script will start processing images to track the corner you selected – you should see a green x marking the point as it moves.
- Once the processing is done, a figure will open displaying the position-v-time graph for the mass you just tracked. You can accept the data if you like it, or reject it if you don't (it will then be deleted – you can select a new correlation template if you like and try again).
- The start frame will open again and ask you if you would like to track another mass, or save your data and quit.



If you press 'save data and quit', a .csv file will be saved with two or more columns of data (time, and the positions of the masses in the order you selected them in the images).