# Steam Locomotive Design Optimization

## I. Introduction

Your task is to design an antique steam locomotive drive train using modern optimization techniques. Just go with it. Your goal of this project is to size the main drive wheels and the connecting rod that joins them. Your objective is to minimize the cost of the materials needed to make these parts, while ensuring the parts are adequately sized to withstand the loads to which they are subjected. You do not need to design the main rod, which connects the side rod to the piston.

Power is supplied to the drive wheels by a horizontal piston that pushes one end of the "main rod," whose other end is pinned to one of the wheels, causing the wheel to rotate as the main rod is alternately pushed and pulled. Also pinned to the wheel at this point is the "side rod," whose other end is pinned (at the same radial location) to the second drive wheel. In this way the rods provide power to both wheels and ensure they rotate at the same speed. This arrangement is mirrored on the other side of the locomotive. For pictures and examples, visit the Wikipedia page for "4-4-2 (locomotive)".

All the equations you will need are either given in this document, or are implied by the text below. You may need to do some light research to help you understand how to apply the equations or to clarify the locomotive terminology. Do not ask the TA or the instructor to help you define these terms without first researching them on your own. Use English units when preparing your report and your code. All dimensions should be reported in inches.

## II. Design Requirements

You must determine optimal values for the following variables:

1. The length ($l$), height ($h$), and thickness ($b$) of the side rod. Assume that the side rod's geometry is simplified as a rectangular prism, and you may neglect stress concentrations at the pin connections.

2. The radius of the drive wheels ($R$).

3. The radial distance from the center of the drive wheels at which the side rods are pinned ($r$).

Your goal is to minimize the cost of the materials needed to make the drive wheels and side rods. Use the part volume as a surrogate for cost. For the side rod, the volume is simply the product $lbh$. For the drive wheels, use the formula $volume = (0.4)(4in)\pi R^2$ for each wheel, where $4in$ is the (fixed) thickness of the wheels and 0.4 is a solidity factor.

The locomotive will have a 4-4-2 wheel configuration, meaning that there are 4 drive wheels (2 per side) with 4 leading and 2 trailing smaller, unpowered wheels. The total locomotive weight is expected to be 250,000 lbs, of which 140,000 is carried by the drive wheels. (You do not need to consider this weight as varying with your design). The locomotive must provide a starting tractive force of 32,000 lbs, and a max-speed tractive force of 12,000 lbs. The design maximum speed is 70 mph. There must be 12 inches of free space between the two drive wheels to allow for the brakes and other mechanical equipment. (In other words, the spacing between the drive wheel axles is $2R + 12$.

The maximum allowable piston speed limits the drive wheels to a rotational speed of 300 rpm or less. The piston diameter is 24 inches, and the maximum boiler pressure is 200psi. The combined tractive force of the two pistons may be approximated by:

$$T = 0.85(BoilerPressure)(PistonDiameter)^2(PistonStroke)/(DriveWheelDiameter)$$

Note that the kinematics of the drive mechanism imply that the piston stroke is equal to $2r$.

The side rod will be made of steel with an elastic modulus ($E$) of $29 \times 10^6 psi$, and an endurance limit ($S_e$) of 28,000 psi. The endurance limit is used when calculating the bending stress in order to ensure that the rod will not fail after extended use.

The total tractive force is assumed to be evenly divided among the four drive wheels, so that each wheel sees a torque equal to $RT/4$. This torque is supplied by the side rod, so that the compressive force in the side rod is equal to $RT/4r$. You must ensure that the side rods to not buckle as a result of this force. Buckling should be checked along both axes of the beam, using the following equations for the maximum allowable compressive force

$$f_h = \pi^2 E I_h / l^2$$

$$f_b = \pi^2 E I_b / (2l^2)$$

In this equation, $I_h$ and $I_b$ are the area moments of inertia along the two axes:

$$I_h = \tfrac{1}{12} b h^3$$

$$I_b = \tfrac{1}{12} h b^3$$

The side rod is also subjected to a bending moment due to its rotational motion. The rotational speed of the side rod matches the rotational speed of the wheel, which is equal to $(TrainSpeed/R)$. The centripedal acceleration of the rod is $a_c = r\omega^2$, where $\omega$ is the rotational speed. The maximum bending moment of the rod is approximated as $M = (m)(l)(a_c)/8$, where $m$ is the mass of the side rod. Assume a steel density of $0.28 lb/in^3$. The bending stress is equal to $Mh/(2I_h)$. This value must not exceed the endurance limit given above.

For both the buckling and bending moment constraints, apply a factor of safety of 2. In other words, ensure that the constraints would still be satisfied even if the calculated forces/moments were doubled.

## III. Setup and Preparation

1. Download, install, and activate MATLAB 2014b from Georgia Tech's OIT Software Distribution site:
   `https://software.oit.gatech.edu/`
   Note: You will need to use this version of MATLAB for the project.

2. Download, install, and activate a trial of the Global Optimization Toolbox v3.3 from the Mathworks website:
   `http://www.mathworks.com/products/global-optimization/`
   Note: you will need to use the Mathworks account you created when installing MATLAB 2014b. Also, this trial will only work on one computer. You can check your toolbox version by using the `ver` command. If you cannot complete this step, please notify the TA immediately.

3. For grading consistency, add the line `rng(12345)` at the beginning of your main function/script. This will seed your random number generator with 12345 every time you run your code.

4. Use the following code to define your baseline configuration of MATLAB's built-in genetic algorithm, `ga()`:

```
baseoptions = gaoptimset(@ga);
baseoptions = gaoptimset(baseoptions,'PlotFcns',{@gaplotdistance, @gaplotrange},...
'PlotInterval',1,...
'NonlinConAlgorithm','penalty',...
'EliteCount',2,...
'SelectionFcn',@selectionroulette,...
'CrossoverFcn',@crossoverscattered,...
'CrossoverFraction',0.75,...
'MutationFcn',{@mutationuniform, 0.01},...
'PopulationSize',50,...
'Generations',50,...
'OutputFcns',@gaoutputfun);
```

## IV. Tasks

1. Formulate the problem described above as a formal optimization statement of the form:

   minimize *objective function*
   by changing *design variables*
   subject to *constraints*

   Express constraints in terms of high level variables (i.e. you do not need to substitute in equations to express constraints in only design variables), and, aside from side constraints, express all constraints in standard form. Any side constraints should be expressed as upper and/or lower bounds. You may add additional constraints that you feel are necessary and not explicitly mentioned above. Additional constraints must be accompanied by a statement justifying their necessity. Use the following bounds: $[1, 1, 20, 1] \leq [h, b, R, r] \leq [10, 10, 60, 60]$.

2. Solve the optimization problem using *one* run of the baseline configuration of MATLAB's `ga()` function. Report your optimum design, the objective function value at the optimum, and include the corresponding plots, `@gaplotdistance` and `@gaplotrange`. In addition, include the code from your nonlinear constraints function here.

3. Explore the effects of individually changing the following parameters to the given levels:

   EliteCount = [0, 2, 10]

   CrossoverFraction = [0.4, 0.75, 0.99]

   Mutation rate = [0, 0.01, 0.1]

   PopulationSize = [20, 50, 100]

   Create plots similar to `@gaplotdistance` and `@gaplotrange`, comparing the optimizer's behavior for different values of each parameter (8 plots total). Describe and explain your observations of changes in optimizer behavior for each parameter (4 paragraphs total). You may need to read into MATLAB's documentation to understand what each parameter controls.

   Since genetic algorithms are stochastic, a single run may not be representative of changes in the optimizer's behavior. Instead, perform 10 consecutive runs (i.e. 10 looped calls of `ga()`, as opposed to clicking the green arrow 10 times) for each parameter level and average the data into a single curve. The attached `gaoutputfun` function will assist you in creating these plots.

4. Code your own genetic algorithm using any methods you know or have learned in class and solve the optimization problem. Report your optimum design and the objective function value at the optimum. In a table, clearly list out your selection, crossover, and mutation methods, and the settings of any and all parameters. Then, give a description of how constraints are handled. Compare the behavior of your genetic algorithm to the baseline MATLAB `ga()` by creating plots similar to `@gaplotdistance` and `@gaplotrange` for the averages over 10 runs (2 plots).

   Discuss and explain observed differences in behavior and performance.

## V. Grading

Your grade will be heavily weighted toward your Task 3 and 4 discussions and plots. Discussions should be clear, comprehensive, and technical. For a high quality discussion, it is suggested that you do some background research into genetic algorithms to gain an understanding of keywords and common discussion points. Include any citations you found useful.

Plots must be clear and easy to understand. Consider aspects such as size, labels, legends, line type, color, scale, etc. An example of an acceptable plot is attached.