MATH 377: INTRODUCTION TO OPERATIONS RESEARCH: FALL 2016
HOMEWORK SOLUTION KEY

STEVEN J. MILLER (SJM1@WILLIAMS.EDU, STEVEN.MILLER.MC.96@AYA.YALE.EDU): MATH 377, FALL 2016

ABSTRACT. A key part of any math course is doing the homework. This ranges from reading the material in the book so that you can do the problems to thinking about the problem statement, how you might go about solving it, and why some approaches work and others don't. Another important part, which is often forgotten, is how the problem fits into math. Is this a cookbook problem with made up numbers and functions to test whether or not you've mastered the basic material, or does it have important applications throughout math and industry? Below I'll try and provide some comments to place the problems and their solutions in context.

CONTENTS

Date: November 14, 2016.

1. HW #2: Due September 19, 2016

1.1. **Problems.** #0: Write a program to generate Pascal's triangle modulo 2. How far can you go? Can you use the symmetries to compute it quickly? You do not need to hand this in. From the textbook: #1: Exercise 1.7.4 (there are many trig tables online: see for example `http://www.sosmath.com/tables/trigtable/trigtable.html`), and read BUT DO NOT DO Exercise 1.7.5. #2: Exercise 1.7.7. #3: Exercise 1.7.18. #4: Exercise 1.7.34. #5: Exercise 1.7.26. #5: Exercise 1.7.36.

1.2. **Solutions.**

#1: Exercise 1.7.4. If we know the values of either $\sin(x)$ or $\cos(x)$ for $0 \le x \le \pi/4$ (or from 0 to 45 degrees if you prefer not to work in radians) then we can find the values of all trig functions using basic relations (such as $\sin(x + \pi/2) = \cos(x)$). Create a look-up table of $\sin(x)$ by finding its values for $x = \frac{k}{45}\frac{\pi}{4}$ with $k \in \{0, 1, \dots, 45\}$. Come up with at least two different ways to interpolate values of $\sin(x)$ for x not in your list, and compare their accuracies. For which values of x are your interpolations most accurate?
Solution: There are many trig tables online (see for example

`http://www.sosmath.com/tables/trigtable/trigtable.html`

for one such table). A very simple way is to use whatever value is closest. Thus if we have values for $\sin x_n$ for $n \in \{0, 1, \dots, N\}$, one option is to approximate $\sin x$ by $\sin x_n$, where n is chosen so that x_n is the closest angle to x.

We can do better. The next idea is to linearly interpolate between the two angles closest. Thus, if $x_n < x < x_{n+1}$ (there is no reason to interpolate if x happens to be one of the angles in our table!), one possibility is to look at a linear combination of $\sin x_n$ and $\sin x_{n+1}$. Thus we can look at $w_n \sin x_n + (1 - w_n) \sin x_{n+1}$, where we have chosen a non-negative weight $w_n \in [0, 1]$. While we have complete freedom in choosing these weights, some choices are better and more natural than others. It seems reasonable that we should weight more the angle *closer* to x. Thus one option is

$$\frac{x_{n+1} - x}{x_{n+1} - x_n} \sin x_n + \frac{x - x_n}{x - x_n} \sin x_{n+1}.$$

Notice the weights add to 1, and the closer x is to x_n, the less we care about $\sin x_{n+1}$ and the more we care about $\sin x_n$ in our approximation (and we have a similar result for x close to x_{n+1}).

There is another option: we can use calculus and use Taylor series. If x_n is the closet angle to x, we have

$$\sin x \;=\; \sin x_n + \frac{\sin' x_n}{1!}(x - x_n) + \frac{\sin'' x_n}{2!}(x - x_n)^2 + \frac{\sin''' x_n}{3!}(x - x_n)^3 + \cdots.$$

However, as the derivative of sine is cosine and the derivative of cosine is negative sine (*we need our angles to be measured in radians for this to hold!*), we get the following:

$$\sin x \;=\; \sin x_n + \frac{\cos x_n}{1!}(x - x_n) - \frac{\sin x_n}{2!}(x - x_n)^2 - \frac{\cos x_n}{3!}(x - x_n)^3 + \cdots.$$

We now have an interesting problem: which is better? Is it better to interpolate with linear weights, or the Taylor series? The more Taylor coefficients we take the more accurate it should be, so eventually the Taylor method should be superior. Numerical computations for one fixed choice suggest that even the one term Taylor series is better.

It's worth noting that the Taylor series expansion of sine only involves sines and cosines; thus given a Taylor series of degree d we can express our answer as a weight of the values of sine and cosine of the closest angle! Of course, the weight used depends on the value of our input, but our output is linear in the values of the lookup table. The difference is that we're not using two adjacent angles for sine, but sine and cosine at the same angle.

We assume we have a lookup table for sin(x) at each degree, from 0 to 360, and discuss various ways to interpolate. As Mathematica evaluates in radians, we need to put in a conversion factor.

The first program, linweightf, calculates the weights needed and uses linear weights. As the separation between entries in the lookup table is 1, the denominator in calculating the weights is fortunately just 1. These weights are easy to find and the calculation is pretty fast.

Taylorf is a bit more involved, but not horribly so. It looks at the Taylor expansion at a point to a given degree. The calculation is longer, but only uses the values in the table. Even doing just one term seems to do better, and of course the more terms we take the better we do.

```
convert = Pi / 180. ;

linweightf[x_] := Module[{},
   w = Ceiling[x] - x;
   new = w Sin[Floor[x] convert] + (1 - w) Sin[Ceiling[x] convert];
   Return[new];
   ];

taylorf[x_, d_] := Module[{},
   step = If[x - Floor[x] < .5, x - Floor[x], Ceiling[x] - x] convert;
   nearest = If[x - Floor[x] < .5, Floor[x], Ceiling[x]];
   new = 0;
   For[k = 0, k <= d, k++,
    new =
     new + If[Mod[k, 2] == 0, Sin[nearest convert],
        Cos[nearest convert]] If[Mod[k, 4] == 2 || Mod[k, 4] == 3, -1,
         1] step^k / k!
    ];
   Return[new];
   ];
```

Below we calculate how good the various approximations do at 13.432 degrees. Even the linear weight is doing a good job. We print out the answer and the two nearest values in the lookup table.

```
In[90]:= angle = 13.432;
Print["Angle = ", angle];
Print[Sin[angle convert], ", ", Sin[Floor[angle] convert], ", ",
  Sin[Ceiling[angle] convert]];
linweightf[angle]
taylorf[angle, 1]
taylorf[angle, 2]
taylorf[angle, 3]
taylorf[angle, 4]

During evaluation of In[90]:= Angle = 13.432

During evaluation of In[90]:= 0.232291, 0.224951, 0.241922

Out[93]= 0.232282

Out[94]= 0.232298

Out[95]= 0.232291

Out[96]= 0.232291

Out[97]= 0.232291
```

In the analysis below, we fix a fractional angle and then look at it plus n, for n from 0 to 359. We calculate the absolute value of the error in each method and sum, and see which method has the lowest aggregate error. The first order Taylor series is only a little better than the linear weights, but it is better. The second order gives us a few more digits of accuracy, and the third and fourth are ridiculously good!

```
decimal = .432;
For[d = 0, d <= 10, d++, error[d] = 0];
For[n = 0, n <= 359, n++,
  {
   angle = n + decimal;
   value = Sin[angle convert];
   error[0] = error[0] + Abs[ value - linweightf[angle]];
   For[d = 1, d <= 4, d++,
    error[d] = error[d] + Abs[value - taylorf[angle, d]]];
   }];
For[d = 0, d <= 4, d++, Print[d, ": ", error[d]]]

0: 0.00856529

1: 0.00651435

2: 0.0000163723

3: 3.0861*10^-8

4: 4.65334*10^-11
```

#2: Exercise 1.7.7. Prove the five log laws. For example, for the first we have $\log_b x_i = y_i$, so $x_i = b^{y_i}$. Thus $x_1 x_2 = b^{y_1} b^{y_2} = b^{y_1+y_2}$. By definition, we now get $\log_b(x_1x_2) = y_1 + y_2$, which finally yields $\log_b(x_1x_2) = \log_b x_1 + \log_b x_2$.
Solution: (2) If $\log_b x = y$ then $x = b^y$ so $x^r = b^{ry}$ and thus $\log_b(x^r) = ry = r\log_b x$. (3) Take the logarithm base b of $b^{\log_b x}$ and use (2). (4) follows from (1) applied to $x_1 x_2^{-1}$ and then using (2) for x_2^{-1}. (5) Is the most interesting. Let $\log_c x = y$ so $x = c^y$. As $b = c^{\log_c b}$ by (3), we find

$$b^{\log_b x} \ = \ c^{\log_b x \log_c b} \ \ \Rightarrow \ \ \log_c b \log_b x \ = \ \log_b x \log_c b \log_c c$$

by taking logs of both sides and using (3); the result follows from division and noting $\log_c c = 1$.

#3: Exercise 1.7.18. Iterate this procedure one or two more times.
Solution: There are many ways to repeat it. One option is to use $\frac{3}{8}m^2 \le 1 + \cdots + m \le \frac{6}{8}m^2$. Let's now use this on the sets S_{11} and S_{12}. For S_{11} we find a lower bound of $\frac{3}{8}(n/2)^2$, while the lower bound for the second is the same plus $n^2/4$; thus the lower bound is

$$\frac{3}{8}\frac{n^2}{4} + \frac{n^2}{4} + \frac{3}{8}\frac{n^2}{4} \ = \ \frac{7}{16}n^2,$$

which is now *very* close to $n^2/2$. Similarly we find for the upper bound

$$\frac{6}{8}\frac{n^2}{4} + \frac{n^2}{4} + \frac{6}{8}\frac{n^2}{4} \ = \ \frac{10}{16}n^2.$$

A possible pattern has emerged: lower bound has constant $(2^\ell - 1)/2^{1+\ell}$, while the upper bound is $(2^\ell + 2)/2^{1+\ell}$.

#4: Exercise 1.7.34. **Solution:** See the referenced paper.

#5: Exercise 1.7.26. Consider four algorithms: the first runs in e^N steps, the second in N^2 steps, the third in $N^{1/2}$ and the fourth in $\log N$ steps. Percentagewise how much of an increase is there in run-time in going from an input of N to $2N$ (i.e., doubling the input)? Express your answer as a function of N, but for definiteness also do for $N \in \{100, 10^{10}, 10^{100}\}$.
Solution: Percentagewise they are (approximately) e^N, $(1+1/N)^2$, $\sqrt{2}$ and $\log 2/\log N$. The table is straightforward.

#5: Exercise 1.7.36. **Solution:** The matrix is

$$A \ = \ \left(\begin{array}{cccc} 0 & 2 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right).$$

The eigenvalues are

$$(1+\sqrt{5})/2, -1, (1-\sqrt{5})/2, 0$$

with corresponding eigenvectors

```
{{2 + Sqrt[5], 1/2 (3 + Sqrt[5]), 1/2 (1 + Sqrt[5]), 1}, {-1, 1, -1,
  1}, {2 - Sqrt[5], 1/2 (3 - Sqrt[5]), 1/2 (1 - Sqrt[5]), 1}, {0, 0,
  0, 1}}
```

Let’s say the initial condition is that we have a one year old pair and all else zero: $\{0, 1, 0, 0\}$. If M is the matrix of eigenvectors (so column i is the i^{th} eigenvector) then the coefficients c_i in writing our initial condition as a sum of eigenvectors arises from $M^{-1}\{0, 1, 0, 0\}^T$, yielding

```
-((-(3/2) + Sqrt[5]/2)/Sqrt[5]), 1, -((3/2 + Sqrt[5]/2)/Sqrt[5]), 0.
```

HOMEWORK DUE MONDAY SEPTEMBER 26: Choose a project; put that on the google sheet: `https://docs.google.com/spreadsheets/d/178i44ycRXjfVsHQZSpgngrWMZvdJgH9gd1CSyAXpMRg/edit#gid=0`. #1: Investigate the Euclidean algorithm for various choices of x and y. What values cause it to take a long time? A short time? For problems like this you need to figure out what is the right metric to measure success. For example, if $x < y$ and it takes s steps, a good measure might be $s/\log_2(x)$. ă#2: What is the dimension of the Cantor set? #3: Exercise 3.6.38: Find the optimal solution to the diet problem when the cost function is $\text{Cost}(x_1, x_2) = x_1 + x_2$. #4: Exercise 3.6.39. There are three vertices on the boundary of the polygon (of feasible solutions); we have seen two choices of cost functions that lead to two of the three points being optimal solutions; find a linear cost function which has the third vertex as an optimal solution. #5: Exercise 3.6.40. Generalize the diet problem to the case when there are three or four types of food, and each food contains one of three items a person needs daily to live (for example, calcium, iron, and protein). The region of feasible solutions will now be a subset of $\mathbb{R}^3$. Show that an optimal solution is again a point on the boundary. #6: the diet problem with two products and two constraints led us to an infinite region, and then searching for the cheapest diet led us to a vertex point. Modify the diet problem by adding additional constraints so that, in general, we have a region ofăfinite volume, and again show that the optimal point is at a vertex. Your constraints should be reasonable, and you should justify their inclusion.ă

2. HW #3: Due Monday, September 26

#0: Choose a project; put that on the google sheet: https://docs.google.com/spreadsheets/d/178i44ycRXjfVsHQZSp edit#gid=0. #1: Investigate the Euclidean algorithm for various choices of x and y. What values cause it to take a long time? A short time? For problems like this you need to figure out what is the right metric to measure success. For example, if $x < y$ and it takes s steps, a good measure might be $s/\log_2(x)$. ă#2: What is the dimension of the Cantor set? #3: Exercise 3.6.38: Find the optimal solution to the diet problem when the cost function is $\text{Cost}(x_1, x_2) = x_1 + x_2$. #4: Exercise 3.6.39. There are three vertices on the boundary of the polygon (of feasible solutions); we have seen two choices of cost functions that lead to two of the three points being optimal solutions; find a linear cost function which has the third vertex as an optimal solution. #5: Exercise 3.6.40. Generalize the diet problem to the case when there are three or four types of food, and each food contains one of three items a person needs daily to live (for example, calcium, iron, and protein). The region of feasible solutions will now be a subset of $\mathbb{R}^3$. Show that an optimal solution is again a point on the boundary. #6: the diet problem with two products and two constraints led us to an infinite region, and then searching for the cheapest diet led us to a vertex point. Modify the diet problem by adding additional constraints so that, in general, we have a region ofăfinite volume, and again show that the optimal point is at a vertex. Your constraints should be reasonable, and you should justify their inclusion.ă

2.1. Solutions. **#1: Investigate the Euclidean algorithm for various choices of x and y. What values cause it to take a long time? A short time? For problems like this you need to figure out what is the right metric to measure success. For example, if $x < y$ and it takes s steps, a good measure might be $s/\log_2(x)$.**

Solution: The answer are adjacent Fibonacci numbers. Clearly we can make things take more steps by taking the numbers larger, and thus it is important to come up with a reasonable metric. A great choice is to look at the number of steps versus the theoretical maximum. Of course, we don't have to get the theoretical maximum perfectly correct; it suffices to get the correct growth rate with respect to x and we can miss by a multiplicative constant, as that would affect all rations equally. Thus we'll use $\log_2 x$ for our scaling.

Next, we can assume $x \le y < 2x$; if $y \ge 2x$ it won't take more steps than the corresponding y (which has the same remainder when dividing by x) that lives in $[x, 2x)$.

We make the assumption that there is some ratio r between x and y that leads to the worse run-time. The worse possible case is that after each step the two new numbers also have that ratio. Thus, imagine we go from (x, y) to $(y - x, x)$ and both pairs have ratio r:

$$\frac{y}{x} = r = \frac{x}{y-x}.$$

Cross multiplying and simplifying gives

$$y^2 - xy = x^2 \quad \text{therefore} \quad y^2 - xy - x^2 = 0.$$

We solve for y as a function, or equivalently we write $y = rx$ and find r satisfies

$$r^2x^2 - rx^2 - x^2 = 0 \quad \text{or} \quad x^2(r^2 - r - 1) = 0.$$

Notice that the equation for r is the same as the characteristic polynomial, and we find

$$r = \frac{1 \pm \sqrt{5}}{2}.$$

As $x \le y$ we need $r > 0$, and thus the ratio that will give us the most trouble should be $r = (1 + \sqrt{5})/2$; this is the Golden Mean, and leads to the Fibonacci numbers!

The above is not a fully fleshed out proof. It *assumed* there was a worse ratio (clearly the algorithm runs fastest when $y = x$). Let's try to attack this from the other perspective: let's start off with the smallest pair and move upwards: so we go from (1,0) to (1,1) to (2,1) to (3,2) to (5,3), ... At each stage we do what keeps us as small as possible, and makes the next number as small as possible, and thus this is the optimal approach.

#2: What is the dimension of the Cantor set?

Solution: The dimension is $\log_3 2$. We use the formula that the dimension d can be found by $c = r^d$, where when we dilate our set by a factor of r we end up with d copies of it. For the Sierpinski triangle from class, when we doubled the sides we ended up with three copies of our original triangle. Remember the Cantor set is defined as what is left when we remove the middle third of each interval at each stage. Thus we start with $[0, 1]$, then go to $[0, 1/3] \cup [2/3, 1]$, then $[0, 1/9] \cup [2/9, 1/3] \cup [2/3, 7/9] \cup [8/9, 1]$, and so on. If we triple the set (so we map x to $3x$) then we end up with two copies of the Cantor set: the region contained in $[0, 1/3]$ expands to a full Cantor set in $[0, 1]$, as does the region

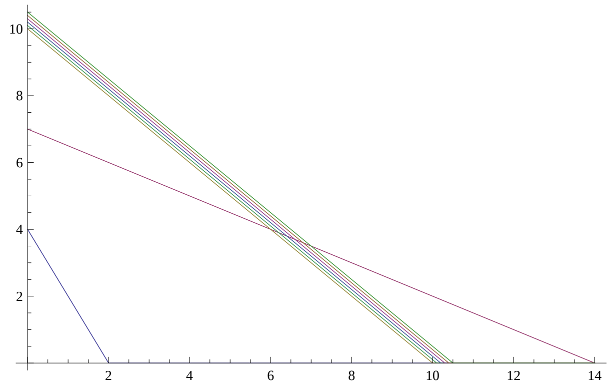

FIGURE 1. Diet Problem 1: Plot of the first diet problem, with several cost lines.

contained in $[2/3, 1]$, but now to a Cantor set in $[2, 3]$. We thus have $c = 2$ when $r = 3$, so $2 = 3^d$ or the dimension d equals $\log_3 2 \approx .63093$. Note this is a number between 0 and 1, which is reasonable as the Cantor set is fatter than a singleton but thinner than a point.

#3: Exercise 3.6.38. Find the optimal solution to the diet problem when the cost function is $\mathrm{Cost}(x_1, x_2) = x_1 + x_2$**.**

Solution: When I first taught this course I made a mistake in describing the Diet Problem in the notes. In the text I had one unit of cereal contributing 30 units of iron and 5 units of protein; however, when I wrote the equations in (1) I transposed things, and had one unit of cereal giving 30 units of iron and 15 units of protein. I'll thus solve the problem both ways.

Using the numbers in the book, we have the following system of equations:

$$\begin{aligned} 30x_1 + 15x_2 &\geq 60 \ \ \text{(iron)} \\ 5x_1 + 10x_2 &\geq 70 \ \ \text{(protein)} \\ x_1, x_2 &\geq 0, \end{aligned} \tag{2.1}$$

and now we want to minimize $\mathrm{Cost}(x_1, x_2) = x_1 + x_2$. It isn't immediately clear what the optimal solution is, as both products have the same cost per unit, but one delivers more iron and the other more protein. We give a plot in Figure 1.

Here is the Mathematica code to generate the plot.

```
line1[x_] := If[-2 x + 4 > 0, -2 x + 4, 0]
Cost[x_, c_] := If[ -x + c > 0, -x + c, 0]
Plot[{line1[x], -.5 x + 7, Cost[x,10.0], Cost[x,10.1],
  Cost[x,10.2], Cost[x,10.3], Cost[x,10.4], Cost[x,10.5]}, {x,0,14}]
```

The cost falls as we shift the cost lines down and to the left. Notice that whenever the protein constraint is satisfied then the iron constraint holds as well, and is thus extraneous. (To see this, note the coefficients from this equation are all larger than those of the one below, and the required amount is less!) The optimal diet will be entirely steak (i.e., only the second product). Thus $x_1 = 0$ and $x_2 = 7$.

We now consider the other diet problem:

$$\begin{aligned} 30x_1 + 5x_2 &\geq 60 \ \ \text{(iron)} \\ 15x_1 + 10x_2 &\geq 70 \ \ \text{(protein)} \\ x_1, x_2 &\geq 0, \end{aligned} \tag{2.2}$$

and now we want to minimize $\mathrm{Cost}(x_1, x_2) = x_1 + x_2$. We give a plot in Figure 2.

The Mathematica code is

```
line2[x_] := If[-6 x + 12 > 0, -6 x + 12, 0]
Cost[x_, c_] := If[ -x + c > 0, -x + c, 0]
Plot[{line2[x], -1.5 x + 7, Cost[x, 10.0], Cost[x, 10.1],
  Cost[x,10.2], Cost[x,10.3], Cost[x,10.4], Cost[x,10.5]}, {x,0,5}]
```

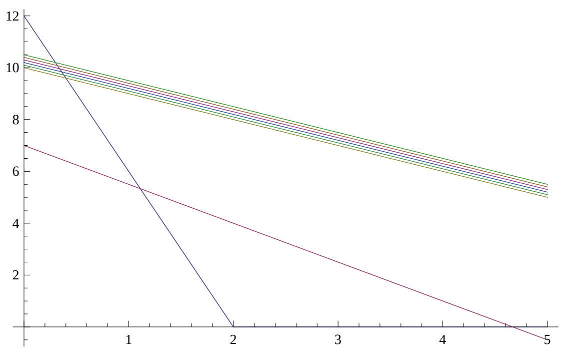

FIGURE 2. Diet Problem 1: Plot of the second diet problem, with several cost lines.

The cost is falling as the cost line moves down and to the left. We flow until we have none of the second product, only buying the first product (thus $x_1 = 4\frac{2}{3}$ and $x_2 = 0$).

#4: Exercise 3.6.39. There are three vertices on the boundary of the polygon (of feasible solutions); we have seen two choices of cost functions that lead to two of the three points being optimal solutions; find a linear cost function which has the third vertex as an optimal solution.
Solution: We have:

$$\begin{aligned} 30x_1 + 5x_2 &\geq 60 \ \text{(iron)} \\ 15x_1 + 10x_2 &\geq 70 \ \text{(protein)} \\ x_1, x_2 &\geq 0. \end{aligned} \tag{2.3}$$

The two lines have slope -6 and -1.5; if we choose our cost function to have a slope between these two values, then the intersection of those two lines will be the unique optimal point. We can do this if we take a slope of -4, or equivalently if the cost function is $\text{Cost}(x_1, x_2) = 4x_1 + x_2$ (though we may replace the 4 with any number strictly between 1.5 and 6).

#5: Exercise 3.6.40. Generalize the diet problem to the case when there are three or four types of food, and each food contains one of three items a person needs daily to live (for example, calcium, iron, and protein). The region of feasible solutions will now be a subset of $\mathbb{R}^3$. Show that an optimal solution is again a point on the boundary.
Solution: If each food can contain exactly one item, then the only way we can have a solution is if each food contains a different item *or* we have more food choices than needed items. If we only have three food items, each food must contain a different nutrient, and then there is only one feasible diet: take the appropriate amount of each food. If instead we have four types of food, we need two of the food types to have the same nutrient, and the other two foods to have the remaining two nutrients. In this case, the only interesting aspect of the problem concerns the nutrient represented by two different foods. We simply take whichever food has a better price per unit of nutrient.

The problem is more interesting if the foods can contain all three items. In this case, if we have x_j units of food j, and food j delivers a_{ij} units of nutrient i then, assuming we need r_i units of nutrient i to stay alive, our constraints are

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &\geq r_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &\geq r_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &\geq r_3 \\ x_1, x_2, x_3 &\geq 0. \end{aligned} \tag{2.4}$$

The cost function is $\text{Cost}(x_1, x_2, x_3) = c_1x_1 + c_2x_2 + c_3x_3$.

The same logic as before shows that an optimal solution must be on a boundary; the difference is now we need to use words like planes rather than lines. Instead of a region in the upper right quadrant we get a region in the positive octant. We now have planes of constant cost; we can decrease the cost by moving towards the origin, and thus if we're at an interior point we can lower the cost by shifting 'down'. Similarly, once we hit the boundary, we can continue to

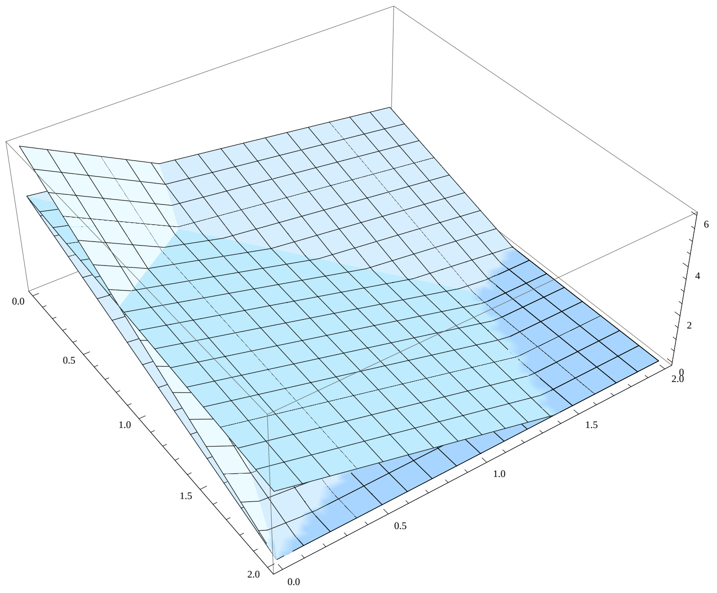

FIGURE 3. Diet Problem 3D: Plot of constraints in a 3-dimensional diet problem.

lower the cost by moving to a vertex (we might not be lowering the cost if the slopes align, but in that case we at least keep the cost constant).

It's a bit harder of course to visualize things in three-dimensions. We give a plot in Figure 3; the constraints are

$$\begin{aligned} 2x + y + z &\geq 4 \\ .6x + 2y + z &\geq 4 \\ 3x + 4y + z &\geq 6 \\ x, y, z &\geq 0. \end{aligned}$$

The Mathematica code is

```
plane1[x_, y_] := If[-2 x - y + 4 >= 0, -2 x - y + 4, 0];
plane2[x_, y_] := If[-.6 x - 2 y + 4 >= 0, -.6 x - 2 y + 4, 0];
plane3[x_, y_] := If[-3 x - 4 y + 6 >= 0, -3 x - 4 y + 6, 0];
Plot3D[{plane1[x,y], plane2[x,y], plane3[x,y]}, {x,0,2}, {y,0,2}]
```

#6: the diet problem with two products and two constraints led us to an infinite region, and then searching for the cheapest diet led us to a vertex point. Modify the diet problem by adding additional constraints so that, in general, we have a region ofăfinite volume, and again show that the optimal point is at a vertex. Your constraints should be reasonable, and you should justify their inclusion.ă

Solution: There are lots of ways to keep things finite. A 'fun' way is to prohibit you from eating too much of any nutrient (in other words, too much of a good thing *can* kill you!). Right now we said we need at least 60 units of iron and at least 70 units of protein; maybe we die if we eat more than 100 units of iron or 140 units of protein. We give a plot in Figure 4.

The Mathematica code is

```
line3[x_, c_] := If[-6 x + c/5 > 0, -6 x + c/5, 0]
line4[x_, c_] := If[-1.5 x + c/10 > 0, -1.5 x + c/10, 0]
Plot[{line3[x, 60], line3[x, 100], line4[x, 70], line4[x, 140]},
{x, 0, 10}]
```

There is a very nice consequence to our restrictions. We now have a closed and bounded subset of the plane. We know from real analysis that any continuous function on a closed and bounded set attains its maximum and its minimum. Thus, there *is* an optimal diet (i.e., a cheapest diet that will keep you alive).

The problem is we don't necessarily know how to find it. When we start studying the simplex method, we'll learn how to flow from a guess to a better guess. This is similar to some items you may have seen. For example, in Lagrange

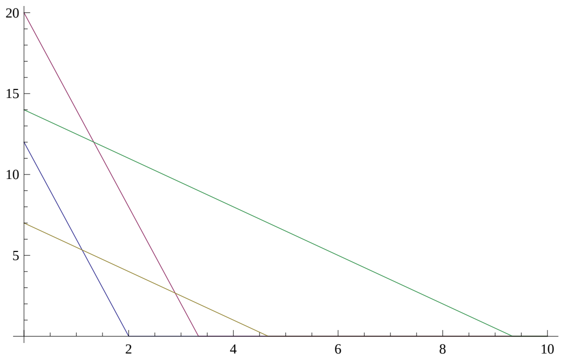

FIGURE 4. Diet Problem 3: Plot of the third diet problem, now with maximum daily allowances.

Multipliers we know candidates for a local extremum of f to the region with constraint function g satisfy $\nabla f = \lambda \nabla g$; if the two gradients are not aligned, we obtain information on which direction to flow. Of course, what's best locally might not be best globally – it might be better to take a small hit in the beginning to get to the global extremum; sadly this issue causes enormous complications in the subject. Another situation where you might have seen this is in contraction mappings, which give an iterative procedure to find fixed points (a nice application of this is in differential equations).

2.2. **HW #4: Due October 3, 2016. Due Friday, Oct 3, 2016:** #0: Make sure you can do, but do not submit, the problems in the "Real Analysis Review" of Chapter 4. #1: Exercise 4.7.7: Prove that any cubic $ax^3 + bx^2 + cx + d = 0$ can be written as $x^3+px+q = 0$ (i.e., we can rewrite so that the coefficient of the x^2 term vanishes and the coefficient of the x^3 term is 1); this is called the depressed cubic associated to the original one. (For fun see the next problem on how to solve the cubic.) #2: Exercise 4.7.12. Can you construct a canonical linear programming problem that has exactly two feasible solutions? Exactly three? Exactly k where k is a fixed integer? #3: Exercise 4.7.14. Find a continuous function defined in the region $(x/2)^2 + (y/3)^2 < 1$ (i.e., the interior of an ellipse) that has neither a maximum nor a minimum but is bounded. #4: Exercise 5.4.3. Imagine we want to place n queens on an $n \times n$ board in such a way as to maximize the number of pawns which can safely be placed. Find the largest number of pawns for $n \leq 5$. #5: Exercise 5.4.4. Write a computer program to expand your result in the previous problem to as large of an n as you can. Does the resulting sequence have any interesting problems? Try inputting it in the OEIS. #6: Consider the problem of placing n queens on an $n \times n$ board with the goal of maximizing the number of pawns which may safely be placed. For each n, let that maximum number be $p(n)$. Find the best upper and lower bounds you can for $p(n)$. For example, trivially one has $0 \leq p(n) \leq n^2$; can you do better? #7: Exercise 5.4.26: Prove $\frac{1}{\sqrt{N}} \sum_{n=-\infty}^{\infty} e^{-\pi n^2/N} = \sum_{n=-\infty}^{\infty} e^{-\pi n^2 N}$. As N tends to infinity, bound the error in replacing the sum on the right hand side with the zeroth term (i.e., taking just $n = 0$). Hint: the Fourier transform of a Gaussian is another Gaussian; if $f(x) = e^{-ax^2}$ then $\widehat{f}(y) = \sqrt{\pi/a} e^{-\pi^2 y^2/a}$.

3. HW #4: Due October 3, 2016

3.1. **Problems. Due Friday, Oct 3, 2016:** #0: Make sure you can do, but do not submit, the problems in the "Real Analysis Review" of Chapter 4. #1: Exercise 4.7.7: Prove that any cubic $ax^3 + bx^2 + cx + d = 0$ can be written as $x^3 + px + q = 0$ (i.e., we can rewrite so that the coefficient of the x^2 term vanishes and the coefficient of the x^3 term is 1); this is called the depressed cubic associated to the original one. (For fun see the next problem on how to solve the cubic.) #2: Exercise 4.7.12. Can you construct a canonical linear programming problem that has exactly two feasible solutions? Exactly three? Exactly k where k is a fixed integer? #3: Exercise 4.7.14. Find a continuous function defined in the region $(x/2)^2 + (y/3)^2 < 1$ (i.e., the interior of an ellipse) that has neither a maximum nor a minimum but is bounded. #4: Exercise 5.4.3. Imagine we want to place n queens on an $n \times n$ board in such a way as to maximize the number of pawns which can safely be placed. Find the largest number of pawns for $n \leq 5$. #5: Exercise 5.4.4. Write a computer program to expand your result in the previous problem to as large of an n as you can. Does the resulting sequence have any interesting problems? Try inputting it in the OEIS. #6: Consider the problem of placing n queens on an $n \times n$ board with the goal of maximizing the number of pawns which may safely be placed. For each n, let that maximum number be $p(n)$. Find the best upper and lower bounds you can for $p(n)$. For example, trivially one has $0 \leq p(n) \leq n^2$; can you do better? #7: Exercise 5.4.26: Prove $\frac{1}{\sqrt{N}} \sum_{n=-\infty}^{\infty} e^{-\pi n^2/N} = \sum_{n=-\infty}^{\infty} e^{-\pi n^2 N}$. As N tends to infinity, bound the error in replacing the sum on the right hand side with the zeroth term (i.e., taking just $n = 0$). Hint: the Fourier transform of a Gaussian is another Gaussian; if $f(x) = e^{-ax^2}$ then $\widehat{f}(y) = \sqrt{\pi/a}e^{-\pi^2 y^2/a}$.

3.2. **Solutions.** #1: Exercise 4.7.7: Prove that any cubic $ax^3 + bx^2 + cx + d = 0$ can be written as $x^3 + px + q = 0$ (i.e., we can rewrite so that the coefficient of the x^2 term vanishes and the coefficient of the x^3 term is 1); this is called the depressed cubic associated to the original one. (For fun see the next problem on how to solve the cubic.)
Solution: Implicit in the above is that $a \neq 0$, as if it did we would not have a cubic but a quadratic; this is the old rectangle-square debate.... If $a \neq 0$ we may divide both sides by a and thus may assume the coefficient of x^3 is 1. We now change variables and let $x = x - b/3$. This sends x^3 to $x^3 - bx^2 + b^2x/3 - b^3/27$, and bx^2 to $bx^2 - 2b^2x/3 + b^3/9$ (the other terms are lower order and don't involve x^2); note the coefficient of the x^2 term is now zero.

#2: Exercise 4.7.12. Can you construct a canonical linear programming problem that has exactly two feasible solutions? Exactly three? Exactly k where k is a fixed integer?
Solution: Assume x_1 and x_2 are two feasible solutions; thus they satisfy $Ax = b$ and have non-negative entries. Then $x_t = tx_1 + (1-t)x_2$ has non-negative entries for all $t \in [0,1]$ and also satisfies the constraints. Thus if there are two solutions there are infinitely many. This idea of a weighted linear combination is extremely important, and occurs frequently in mathematics. It tells us that the solution space is *convex*.

#3: Exercise 4.7.14. Find a continuous function defined in the region $(x/2)^2 + (y/3)^2 < 1$ (i.e., the interior of an ellipse) that has neither a maximum nor a minimum but is bounded.
Solution: Consider $f(x) = x$; it approaches 2 as $(x, y) \to (2, 3)$ and -2 as $(x, y) \to (-2, 3)$ but never hits those values.

#4: Exercise 5.4.3. Imagine we want to place n queens on an $n \times n$ board in such a way as to maximize the number of pawns which can safely be placed. Find the largest number of pawns for $n \leq 5$.
Solution: For $n \in \{1, 2\}$ the answer is zero; a little work shows also zero for $n = 3$, then gets harder. The code below is good enough to brute force $n \leq 5$ in seconds, will be very bad for $n = 6$ (horrible memory management). We get 1 pawn for $n = 4$ and 3 for $n = 5$.

```
chess[n_] := Module[{},
   maxpawns = 0; (*
   max number pawns observe on  nxn board that are safe with n queens \
present *)
   board = {}; (* create the board *)
   For[i = 1, i <= n, i++,
    For[j = 1, j <= n, j++,
     {
      board = AppendTo[board, {i, j}];
      }]];(* end of i, j loops *)
   listboard = Subsets[board, {n}]; (*
   all subsets of EXACTLY n squares *)
   Print[Length[listboard]]; (*
   prints how many cases have to study *)
   (* crashes if don't  subtract 1 below -- strange *)
   For[p = 1, p  <= Length[listboard] - 1, p++
     {
      pawns = 0; (* initialize pawns safe to zero *)
      list = listboard[[p]]; (* these  are the squares chosen *)
      (* mark off what rows,
      columns  and diagonals are not allowed for  pawns *)
      horiz = {}; vert = {}; diag1 = {}; diag2 = {};
      For[k = 1, k <= n, k++,
       {
        horiz = AppendTo[horiz, list[[k, 1]]];
        vert = AppendTo[vert, list[[k, 2]]];
        diag1 = AppendTo[diag1, list[[k, 1]] - list[[k, 2]]];
        diag2 = AppendTo[diag2, list[[k, 2]] + list[[k, 1]]];
        }];
      (* for each square see if can place a pawn;
      if can increase pawn count by 1 *)
      For[i = 1, i <= n, i++,
       For[j = 1, j <= n, j++,
        {

         If[
           MemberQ[horiz, i] == False && MemberQ[vert, j] == False &&
            MemberQ[diag1, i - j] == False &&
            MemberQ[diag2, j + i] == False,  pawns = pawns + 1];
         }]]; (* end of i, j loops *)
      (* see if have more pawns safe than previous best *)
      If[pawns > maxpawns, maxpawns = pawns];
      }]; (* end of p loop *)
   Print["For n = ", n, " max pawns safe is ", maxpawns]; (*
   print best *)
   ];
```

#5: Exercise 5.4.4. Write a computer program to expand your result in the previous problem to as large of an n as you can. Does the resulting sequence have any interesting problems? Try inputting it in the OEIS.
Solution: Need to do better than what I wrote; saving all options into a list is expensive; if we wrote to file and then called would be better. Would make a huge difference if we code symmetries. Actually, didn't realize I left the program running and it got $n = 6$ fairly quickly, giving 5 pawns. The OEIS lists 39 options; one more data point would be nice! Kirby Gordon sent me the following: 0, 0, 0, 1, 3, 5, 7, 11, 18, 22, 30, which is enough to move us to just one option.

#6: Consider the problem of placing n queens on an $n \times n$ board with the goal of maximizing the number of pawns which may safely be placed. For each n, let that maximum number be $p(n)$. Find the best upper and lower bounds you can for $p(n)$. For example, trivially one has $0 \le p(n) \le n^2$; can you do better?
Solution: Clearly $p(n) \le n^2 - n$; it can unfortunately be zero at times, so must be careful. Let's look at some large values of n, for example let's assume $n = m^2$. Then we have an $n \times n$ board, which is $m^2 \times m^2$, and we need to place $n = m^2$ queens. One option is to place these in an $m \times m$ square in the bottom right. Let's see how many squares that kills. It kills m^3 vertically and m^3 horizontally, but this double counts the m^2 in the bottom, so a total of $2m^3 - m^2$ spaces are killed. We now must look at the diagonals killed. We could compute it exactly, but a rough calculation suffices. We have $2m - 1$ diagonals and each diagonal kills at most m^2 for at most $2m^3 - m^2$. There's of course some double counting of this in the bottom right square and the vertical / horizonal rows, but that is going to be lower order. Thus combining everything we see that on the order of $4m^3$ (up to order m^2) of the m^4 are killed. Thus it looks like *in the limit* we can have almost all squares safe! In general, if you have an $n \times n$ consider the sub-block that's $x \times x$ where $x = \lceil \sqrt{n} \rceil$, the smallest integer at least $\sqrt{n}$. Thus the number of pawns that can be safely placed is at least (approximately) n^2 minus something of order $n^{3/2}$ (if we wanted we could figure out the constant multiple of $n^{3/2}$, we know it's around 4).

It's easier to get a good lower bound than an upper bound here; for a lower bound we just need to find the number of pawns that work for one special configuration, while for an upper bound we must get something that holds for all configurations. We give an argument that gets an outstanding upper bound without too much work, and allows us to see the correct order of magnitude. Remember the more rows or columns we have queens in, the more squares are killed; thus it is reasonable to try and minimize the number of rows or columns with queens, being careful always not to accidentally do what is locally best as that may not lead to a globally best placement.

Consider *any* configuration. Let $f(n)$ denote the number of rows that have queens. This kills at least $f(n)n$ squares, n squares in each row with a queen (we can do a little better as it has to kill some squares in columns; I'll leave it to you to try and do that). How many columns have queens? Note the more columns with queens, the more squares killed; thus if we are going for an upper bound we want to consider the fewest possible number of columns with queens. As we can only have queens in $f(n)$ rows, no column can have more than $f(n)$ queens. There are n queens to place, and thus by the pigeonhole principle at least one column must receive $n/f(n)$ queens (which is the average number of queens per column). Note that each queen in this special column kills at least n squares, and thus at least $n^2/f(n)$ squares are killed from the column locations. It's important to note that these squares are *not* necessarily distinct from the squares killed by looking at row placements.

We have shown that the number of squares killed is at least $f(n)n$ and also at least $n^2/f(n)$; thus the number of squares killed is at least $\max(f(n)n, n^2/f(n))$. To get an upper bound for *all* configurations, we choose $f(n)$ to minimize the maximum of the two. We find this by setting the two expressions equal and solving for $f(n)$, as if we take $f(n)$ so that they are equal then increasing it will increase the first factor, while decreasing it would increase the second. Thus

$$f(n)n \;=\; \frac{n^2}{f(n)} \quad \text{thus} \quad f(n) \;=\; \sqrt{n},$$

implying that at least $n^{3/2}$ of the n^2 squares are killed (we should really use floor or ceiling functions for n in general, but this gives the right order of magnitude and I'll just leave it like this).

Thus, combining all our work, we find there exist positive constants $c_1 > c_2$ such that

$$n^2 - c_1 n^{3/2} \;\le\; p(n) \;\le\; n^2 - c_2 n^{3/2};$$

in other words, the order of magnitude of the number of spaces on an $n \times n$ board that are killed is of size $n^{3/2}$!

#7: Exercise 5.4.26: Prove $\frac{1}{\sqrt{N}} \sum_{n=-\infty}^{\infty} e^{-\pi n^2/N} \;=\; \sum_{n=-\infty}^{\infty} e^{-\pi n^2 N}$. As N tends to infinity, bound the error in replacing the sum on the right hand side with the zeroth term (i.e., taking just $n = 0$). Hint: the Fourier transform of a Gaussian is another Gaussian; if $f(x) = e^{-ax^2}$ then $\widehat{f}(y) = \sqrt{\pi/a}e^{-\pi^2 y^2/a}$.
Solution: Step one is to compute the Fourier transform:

$$\widehat{f}(y) \quad = \quad \int_{-\infty}^{\infty} f(x) e^{-2\pi i x y} dx;$$

we are fortunate in that we're given the answer, but that can be found by completing the square. We are able to use the Poisson Summation formula, which states

$$\sum_{n=-\infty}^{\infty} f(n) \;=\; \sum_{n=-\infty}^{\infty} \widehat{f}(n)$$

for nice functions (such as the Gaussian). We can use this formula with $a = \pi/N$, and find

$$\frac{1}{\sqrt{N}} \sum_{n=-\infty}^{\infty} e^{-\pi n^2/N} \;=\; \sum_{n=-\infty}^{\infty} e^{-\pi n^2 N}.$$

Note this problem is *not* hard because of what we're given. What makes it hard is *proving* those givens. Prove the Fourier transform is as claimed. Prove the Poisson Summation Formula.

Also, it's worth seeing why we *care* about this. What happens as $N \to \infty$? On the left hand side, we have to deal with all terms with $|n|$ up to about $\sqrt{N}$, whereas on the right hand side only the $n = 0$ term contributes significantly (as $|n| \ge 1$ already has the argument of the exponential at essentially negative N). This is the power of Poisson Summation; we pass from a long slowly decaying sum to a short rapidly decaying sum. If we keep just the zeroth term on the right, the error is mostly given from the $n = \pm 1$ terms. We can get an upper bound on the error by noting that for $a > 0$,

$$sum_{|n|\geq 1} e^{-an^2} \;=\; 2\sum_{n=1}^{\infty} e^{-an} \;=\; \frac{e^{-a}}{1-e^{-a}}.$$

We can do a bit better by splitting off the $n = 1$ term:

$$\sum_{|n|\geq 1} e^{-an^2} \;=\; 2e^{-a} + 2\sum_{n=2}^{\infty} e^{-an^2} \;\leq\; 2e^{-a} + \frac{2e^{-4}}{1-e^{-4}}.$$

3.3. **New Homework. Due Monday Oct 10, 2016. #1. Prove that if A' has M rows and k columns, with $M \geq k$, then $A'^T A'$ is invertible. Note this is the A' from the text, and thus the k columns of A' are linearly independent.#2. For fixed M, find some lower bounds for the size of $\sum_{k=1}^{M} \binom{N}{k}$. If $M = N = 1000$ (which can easily happen for real world problems), how many basic feasible solutions could there be? There are less than 10^{90} sub-atomic objects in the universal (quarks, photons, et cetera). Assume each such object is a supercomputer capable of checking 10^{20} basic solutions a second (this is much faster than current technology!). How many years would be required to check all the basic solutions? #3: Imagine you want to transmit the shape of the plot $f(x) = \sin(x^3)$ on the interval [-3,3]. You have the ability to sample the value of this function for 360 different choices of x. Plot it if you sample uniformly. Is this the best way to sample? How should you sample / choose where to sample? #4: We say a x is an ordered feasible solution if its non-negative entries are ordered from smallest to largest; thus (1,0,0,4,3,0,0,5,8) is not ordered (as 4 is less than 3) but (1,0,0,3,3,0,0,5,8) is. Prove or disprove: if a canonical linear programming problem has a feasible solution then it has an ordered feasible solution. #5: Give an example of a 4×4 matrix such that each entry is positive and all four columns are linearly independent; if you cannot find such a matrix prove that one exists. #6: Redo the previous problem but for an arbitrary N (thus find an $N \times N$ matrix where all entries are positive and the N columns are linearly independent). Extra Credit: For each positive integer N find a matrix with N rows and infinitely many columns so that all entries are positive and any set of N columns is linearly independent.**

4. HW #5: Due October 10, 2016

4.1. **Problems. Due Friday, Oct 10, 2016:** #1. Prove that if A' has M rows and k columns, with $M \geq k$, then $A'^T A'$ is invertible. Note this is the A' from the text, and thus the k columns of A' are linearly independent.#2. For fixed M, find some lower bounds for the size of $\sum_{k=1}^{M} \binom{N}{k}$. If $M = N = 1000$ (which can easily happen for real world problems), how many basic feasible solutions could there be? There are less than 10^{90} sub-atomic objects in the universal (quarks, photons, et cetera). Assume each such object is a supercomputer capable of checking 10^{20} basic solutions a second (this is much faster than current technology!). How many years would be required to check all the basic solutions? #3: Imagine you want to transmit the shape of the plot $f(x) = \sin(x^3)$ on the interval [-3,3]. You have the ability to sample the value of this function for 360 different choices of x. Plot it if you sample uniformly. Is this the best way to sample? How should you sample / choose where to sample? #4: We say a x is an ordered feasible solution if its non-negative entries are ordered from smallest to largest; thus (1,0,0,4,3,0,0,5,8) is not ordered (as 4 is less than 3) but (1,0,0,3,3,0,0,5,8) is. Prove or disprove: if a canonical linear programming problem has a feasible solution then it has an ordered feasible solution. #5: Give an example of a 4×4 matrix such that each entry is positive and all four columns are linearly independent; if you cannot find such a matrix prove that one exists. #6: Redo the previous problem but for an arbitrary N (thus find an $N \times N$ matrix where all entries are positive and the N columns are linearly independent). Extra Credit: For each positive integer N find a matrix with N rows and infinitely many columns so that all entries are positive and any set of N columns is linearly independent.

4.2. **Solutions. #1. Prove that if A' has M rows and k columns, with $M \geq k$, then $A'^T A'$ is invertible. Note this is the A' from the text, and thus the k columns of A' are linearly independent.**
Solution: If z is any vector with k components, then $z^T A'^T A' z = ||A'z||^2$, where $||v||$ denotes the length of a vector v. Imagine $A'^T A'$ is not invertible. Then the columns of this matrix are dependent, and there is some non-zero vector v such that $A'^T A' v$ is the zero vector. Thus $v^T A'^T A' v = 0$, or $||A'v||^2 = 0$. The only way the length of the vector $A'v$ can be zero is if $A'v$ is zero. What does it mean for $A'v$ to be zero? If v is not the zero vector, it means the columns of A' are linearly dependent. As we know these columns are linearly dependent, we must have v the zero vector. This contradicts our assumption that v is not the zero vector, completing the proof.

#2. For fixed M, find some lower bounds for the size of $\sum_{k=1}^{M} \binom{N}{k}$. If $M = N = 1000$ (which can easily happen for real world problems), how many basic feasible solutions could there be? There are less than 10^{90} sub-atomic objects in the universal (quarks, photons, et cetera). Assume each such object is a supercomputer capable of checking 10^{20} basic solutions a second (this is much faster than current technology!). How many years would be required to check all the basic solutions?
Solution: The binomial coefficients are increasing to the middle, then decreasing. If $M \leq N/2$ a decent bound for the sum is $\binom{N}{M}$; if $N/2 \leq M \leq N$ a reasonable bound is $\binom{N}{N/2}$, though even better would be $\frac{1}{2}(1+1)^N$.

A basic feasible solution is a feasible solution where the columns corresponding to the non-zero entries are linearly independent. If we let c be the number of such columns, we find $1 \leq c \leq 1000$, and for each c the largest number of basic feasible solutions would be $\binom{1000}{c}$. We thus have $\sum_{c=1}^{1000} \binom{1000}{c}$. By the Binomial Theorem, this is $2^{1000} - 1$ (we subtract 1 as we don't have $c = 0$), which is approximately $1.07151 \cdot 10^{301}$. Under our assumptions, we can check 10^{110} possibilities a second, which means we need about $1.07151 \cdot 10^{191}$ seconds. As there are about $1.32016 \cdot 10^8$ seconds in a year, we would need approximately $8.11651 \cdot 10^{182}$ years, far longer than the 15 billion or so years we believe the universe has existed.

#3: Imagine you want to transmit the shape of the plot $f(x) = \sin(x^3)$ on the interval [-3,3]. You have the ability to sample the value of this function for 360 different choices of x. Plot it if you sample uniformly. Is this the best way to sample? How should you sample / choose where to sample?
Solution: If we want to transmit a function such as $f(x) = 3x^5 - 2x^3 + 4x^2 + 8x + 1$ for $-3 \leq x \leq 3$, we could just send the coefficients 1, 8, 4, -2, 0, 3 and then the receiver could easily reconstruct the function. In streaming information we can represent the intensities of the pixel colors (red, green and blue) as functions; unfortunately we typically do not have such simple expressions. While we cannot send a few bits of information and uniquely identify the function, what we can do is send its values at a representative set of points, which allows the receiver to approximately recover it.

In general one would do a Fourier analysis of the signal and expand in terms of sines and cosines (or perhaps better use wavelets). Here we'll confine ourselves to discussing how to choose points to sample plotting a function. First we give the code and then a plot (Figure 5) of the function.

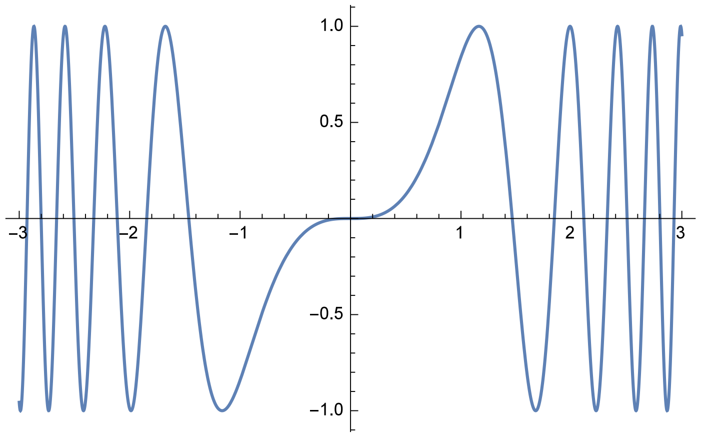

FIGURE 5. Plot of $\sin(x^3)$.

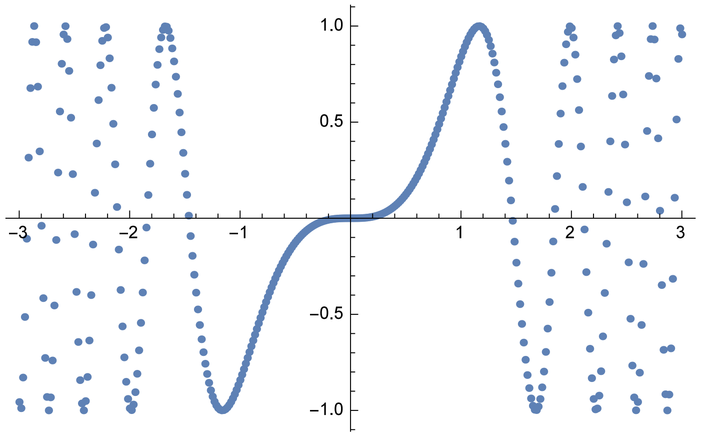

FIGURE 6. Plot of $\sin(x^3)$.

```
 Plot[Sin[x^3], {x, -3, 3}]
```

If we sample uniformly from -3 to 3 say 361 times we get:

```
uniformlist = {};
For[n = -180, n <= 180, n++,
  uniformlist = AppendTo[uniformlist, {n/60, Sin[(n/60)^3]}]];
ListPlot[uniformlist]
```

Notice this fails to capture all of the shape (see Figure 6). The reason is that we're being wasteful. We only have so many observations to make, and we're wasting a lot of them in a region where the function doesn't change much. Instead we should sample more towards the endpoints and less in the center.

```
biaslist = {};
For[n = -180, n <= 180, n++,
  {
   (*x = Sign[n] (3Abs[n]/30)^(1/2) ;*)
   If[Abs[n] > 10, x = Sign[n] (1 + 2 Sqrt[(Abs[n] - 10)/170]),
    x = n/10];
   (*x = Sign[n] (n^2/50^2) / 3 ;*)
   biaslist = AppendTo[biaslist, {x, Sin[x^3]}];
   }];
ListPlot[biaslist]
```

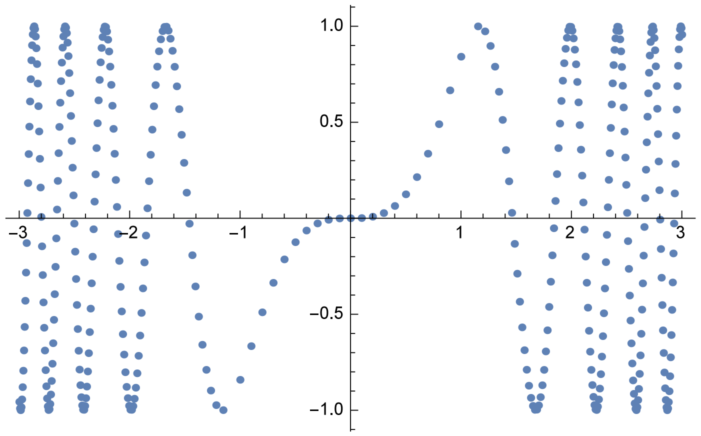

FIGURE 7. Plot of $\sin(x^3)$.

The dynamic sampling (Figure 7 does a much better job. What we did above is not the best, but was easily programmed. We sampled uniformly from -1 to 1, and took 21 values there. In the other regions, we let n run from -180 to -10 and then 10 to 180 and spread the points out a bit within that region. For example, for n positive we chose x_n to be $1 + 2\sqrt{(n-10)/170}$; note this gives us values running from 1 to 3 but the x-coordinates are not spaced equally. This is called dynamic sampling, and is extremely important. When it is expensive to gather data, you want to gather the right data. If the function is approximately constant, as our function is from -1 to 1 (or at least from -1/2 to 1/2), it is wasteful to be frequently sampling there. As we can only sample a fixed number of times, it is better to sample where the function is wildly fluctuating.

Some students noticed that if the second derivative is close to zero we don't need to sample as much, as that requires the first derivative is approximately constant and thus we have a linear growth. Depending on how much one is willing to consider, one can choose points better and better.

#4: We say a x is an ordered feasible solution if its non-negative entries are ordered from smallest to largest; thus (1,0,0,4,3,0,0,5,8) is not ordered (as 4 is less than 3) but (1,0,0,3,3,0,0,5,8) is. Prove or disprove: if a canonical linear programming problem has a feasible solution then it has an ordered feasible solution.
Solution: Unfortunately, the concept of an ordered feasible solution (which I made up for this homework) is not useful. Image $A = I$, the identity matrix. Let b be a vector whose entries are positive an in decreasing order. Then there are no ordered feasible solutions, even though we always have a feasible solution (just take $x = b$). For definiteness, consider $A = I_3$, the 3×3 identity matrix, and let $b = (3, 2, 1)^T$ (I'm using the transpose symbol so as to write b as a row and not a column vector). So we can have feasible solutions and basic feasible solutions here, but we won't have an ordered feasible solution.

#5: Give an example of a 4×4 matrix such that each entry is positive and all four columns are linearly independent; if you cannot find such a matrix prove that one exists.
Solution: If we start off with the identity matrix the columns are linearly independent, but the entries are not positive. The idea is if we add a very small ϵ we'll be fine. I'll leave that as an exercise for you to play with. Instead, let's look at a new approach. If the matrix has a non-zero determinant it is invertible, so consider

$$A = \begin{pmatrix} M & 1 & 1 & 1 \\ 1 & M & 1 & 1 \\ 1 & 1 & M & 1 \\ 1 & 1 & 1 & M \end{pmatrix}.$$

A brute force calculation shows the determinant is $-3 + 8M - 6M^2 + M^4$, and so if we take M large we'll be fine.

Of course, we don't need to calculate the determinant exactly. Imagine an $N \times N$ matrix where all entries are 1, save for the main diagonal where all entries are M. When we expand the determinant we have $N!$ terms. Though this can't happen, the worse possible case (to make the determinant as small as possible) is one only one term is positive (the product of the N values on the main diagonal, and thus contributing M^N), and all other cases come in negative (which can't happen as only half of the terms are negative in the determinant expansion!) and each has $N-1$ factors of M (again, can't happen).

Thus the determinant *is at least*

$$M^N - (N!-1)M^{N-1} \ \geq\ M^N(M - N! + 1);$$

thus if $M \geq N!$ the determinant is positive, and we've found a matrix with all non-negative entries with columns linearly independent.

#6: Redo the previous problem but for an arbitrary N (thus find an $N \times N$ matrix where all entries are positive and the N columns are linearly independent). Extra Credit: For each positive integer N find a matrix with N rows and infinitely many columns so that all entries are positive and any set of N columns is linearly independent.
Solution: See the previous problem for a solution.

4.3. **Next HW: HW #6: Due Oct 17, 2016. #1: Formulate Sudoku as a linear programming problem (you can do either 4×4 or 9×9 Sudoku). #2: Consider the 3×3 constraint matrix A where the first row is 1, 2, 3, the second row is 4, 5, 6 and the third row 7, 8, 9 (thus it's the numbers 1 through 3^2). Let the vector b equal $(1,1,1)^T$. Find all basic feasible solutions to $Ax = b$ with $x \geq 0$. #3: Let's revisit the chess problem from class. Consider an $n \times n$ chess board. We want to put down n queens and maximize the number of pawns that can be safely placed on the board. Set this up as a linear programming problem. #4: Do Exercise 6.6.30. #5: Hand in a short write-up saying who is in your group and what you will be studying / doing. Give a brief outline of what you think you'll need to learn, what data you think you'll need to gather, what you've done so far .... Describe why you feel your group has the necessary skill sets to complete the task, or if not what your plan is to remedy that.**

5. HW #6: Due Friday, October 17, 2016

#1: Formulate Sudoku as a linear programming problem (you can do either 4×4 or 9×9 Sudoku). #2: Consider the 3×3 constraint matrix A where the first row is 1, 2, 3, the second row is 4, 5, 6 and the third row 7, 8, 9 (thus it's the numbers 1 through 3^2). Let the vector b equal $(1, 1, 1)^T$. Find all basic feasible solutions to $Ax = b$ with $x \geq 0$. #3: Let's revisit the chess problem from class. Consider an $n \times n$ chess board. We want to put down n queens and maximize the number of pawns that can be safely placed on the board. Set this up as a linear programming problem. #4: Do Exercise 6.6.30. #5: Hand in a short write-up saying who is in your group and what you will be studying / doing. Give a brief outline of what you think you'll need to learn, what data you think you'll need to gather, what you've done so far .... Describe why you feel your group has the necessary skill sets to complete the task, or if not what your plan is to remedy that.

#1: Formulate Sudoku as a linear programming problem (you can do either 4x4 or 9x9 Sudoku).
Solution: Let x_{ijd} be the binary variable which is 1 if the cell in row i and column j is d, and zero otherwise. Let n be either 4 or 9. Then the constraints are

- For all $j \in \{1, \dots, n\}$ and for all $d \in \{1, \dots, n\}$: $\sum_{i=1}^{n} x_{ijd} = 1$. This means each column has each digit exactly once.
- For all $i \in \{1, \dots, n\}$ and for all $d \in \{1, \dots, n\}$: $\sum_{j=1}^{n} x_{ijd} = 1$. This means each row has each digit exactly once.
- Let $\mathcal{F} = \{(1,1), (1,2), \dots, (\sqrt{n}, \sqrt{n})\}$, and let $(a, b) + \mathcal{F}$ be the set of all pairs of the form $(a + x, b + y)$ for some $(x, y) \in \mathcal{F}$. Then For all $a, b \in \{0, 1, \dots, \sqrt{n} - 1\}$ and all $d \in \{1, \dots, n\}$ we have $\sum_{(i,j) \in (a,b)+\mathcal{F}} x_{ijd} = 1$. This means that in each $\sqrt{n} \times \sqrt{n}$ box we have each digit.

We need an objective function. As all we care is for a feasible solution, we can take as our objective function $\sum_i \sum_j \sum_d x_{ijd}$.

Finally, often Sudokus have certain cells given to us; in that case, we simply add these as constraints: if $\mathcal{S}$ is the set of indices where we are given values, and v_{ij} is the given value, then for all $(i, j) \in \mathcal{S}$ we have $x_{ijd} = 1$ if $d - v_{ij}$ and 0 otherwise.

There are other ways to try and solve this. We could instead let $x_{ij} \in \{1, 2, 3, 4\}$ and try to make that work. I know one group tried the constraint that each column, each row and each of the four blocks of four had to sum to 10, trying to use the only way to get 10 from these numbers is $1 + 2 + 3 + 4$. Unfortunately, $2 + 3 + 2 + 3$ also works, but leads to an invalid Sudoku:

$$\begin{pmatrix} 2 & 3 & 2 & 3 \\ 3 & 2 & 3 & 2 \\ 2 & 3 & 2 & 3 \\ 3 & 2 & 3 & 2 \end{pmatrix}.$$

#2: Consider the 3×3 constraint matrix A where the first row is 1, 2, 3, the second row is 4, 5, 6 and the third row 7, 8, 9 (thus it's the numbers 1 through 3^2). Let the vector b equal $(1, 1, 1)^T$. Find all basic feasible solutions to $Ax = b$ with $x \geq 0$.
Solution: We give a one-line solution at the end; as a large part of homework is to learn the methods and techniques, it is good to see the straightforward approach.

The matrix A is not invertible (the $n \times n$ matrix with entries going from 1 to n^2 is invertible only when $n \leq 2$); one way to see this is to note that the first plus third columns are twice the second. Note that any pair of columns are linearly independent, and any column is linearly independent. Thus there are 6 sub-matrices that generate basic feasible solutions, and each generates a unique candidate for a basic feasible solution. If A' is the reduced matrix, then the candidate for the basic feasible solution is found by solving $A'x' = b$. We multiply by A'^T on the left since $A'^T A'$ is invertible. This gives $A'^T A'x' = A'^T b$, or $x' = (A'^T A')^{-1} A^T b$. This gives us the non-zero entries of the candidate for the basic feasible solution; we finish by adding the zero entries.

- Using the first column, $(1, 4, 7)$, we get a non-zero element of $2/11$ and thus the candidate for the basic feasible solution is $(2/11, 0, 0)$.
- Using the second column, $(2, 5, 8)$, we get a non-zero element of $5/31$ and thus the candidate for the basic feasible solution is $(0, 5/31, 0)$.

- Using the third column, $(3,6,9)$, we get a non-zero element of $1/7$ and thus the candidate for the basic feasible solution is $(0,0,1/7)$.
- Using the first two columns we get non-zero elements $(-1,1)$, and thus the candidate for the basic feasible solution is $(-1,1,0)$.
- Using the first and third columns we get non-zero elements $(-1/2,1/2)$, and thus the candidate for the basic feasible solution is $(-1/2,0,1/2)$.
- Using the second and third columns we get non-zero elements $(-1,1)$, and thus the candidate for the basic feasible solution is $(0,-1,1)$.

Note we can check to make sure these are feasible solutions. When we check, however, the first three all ***fail*** to satisfy $Ax=b$, though the last three do. What went wrong? The problem is that b is not a linear combination of fewer than 2 columns of A, and when we try to take just one column it breaks down. This shouldn't be surprising. In that case $A'^T A$ is a 1×1 matrix and b is not in the column space of A'. While the last three solve the constraints, they are not basic feasible solutions as each has a negative entry. Thus, there are ***no*** basic feasible solutions.

One can do these calculations in a system such as Mathematica, though you have to be careful with the syntax. Here's the code for it.

```
A = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
Transpose[A];
b = Transpose[{{1, 1, 1}}];
A.b
Ap = Transpose[{{1, 4, 7}, {2, 5, 8}}];
Transpose[Ap] .Ap
Inverse[Transpose[Ap] .Ap]
Inverse[Transpose[Ap]. Ap]. (Transpose[Ap] . b)
```

Now, for the promised one-line solution. Imagine there is a basic feasible solution. Then we have $Ax=b$ with the entries of x non-negative and each entry of b is 1. Notice that the second row of A dominates the first row (each matrix element in the second row is larger than the corresponding entry in the first row), yet the constraints want the resulting dot products to be equal. In other words, $x_1+2x_2+3x_3=1$ and $4x_1+5x_2+6x_3=1$. This is impossible, as the second constraint can be written as

$$(x_1+2x_2+3x_3)+3(x_1+x_2+x_3) \;=\; 1;$$

as $x_1+2x_2+3x_3=1$ this implies $3(x_1+x_2+x_3)=0$, which implies each $x_i=0$ (as they must be non-negative for a feasible solution), clearly violating the weighted sum equalling 1.

#3: Let's revisit the chess problem from class. Consider an $n\times n$ chess board. We want to put down n queens and maximize the number of pawns that can be safely placed on the board. Set this up as a linear programming problem.

Solution: Let $x_{ij}=1$ if we have a queen on the board in row i and column j, and zero otherwise. Our first constraint is

$$\sum_{i=1}^n\sum_{j=1}^n x_{ij} \;=\; n.$$

This constraint says we place exactly n queens on the board. In fact, this is the only 'real' constraint; the other constraints come from helping to write the objective function.

For each point (i,j) on the chessboard, let $\mathcal{A}_{i,j}$ denote the squares that a queen placed at (i,j) can attack (plus the square (i,j)). For example, if $(i,j)=(1,1)$ then $\mathcal{A}_{1,1}$ is the first row, the first column, and the diagonal of all pairs (d,d). We're going to introduce some new binary variables y_{ij}. We should think of these as being 1 if we can place a pawn safely at (i,j) and zero otherwise. Consider the constraints for all pairs (i,j) ***such that there is a queen at*** (i,j) we have

$$\sum_{(i,j)\in\mathcal{A}_{ij}} y_{ij} \;=\; 0.$$

This means we cannot place a pawn in the kill zone caused by a queen at (i,j); the difficulty, though, is we don't know where the queens are. One solution is to multiply this constraint by x_{ij} on the left, so it only comes into play if there is

	X	X	X
X	X	X	X
	X	X	X
X		X	

TABLE 1. The 12 squares that attack (2,3) on a 4×4 board.

a queen at (i, j). In other words, consider

$$x_{ij} \sum_{(i,j)\in\mathcal{A}_{ij}} y_{ij} \ = \ 0;$$

if $x_{ij} = 0$ (so no queen at (i, j)) then the y_{ij}'s are free; if there is a queen there then each $y_{ij} = 0$ (i.e., cannot place a pawn there). Unfortunately, this is not linear. If it were, we'd be done, and we'd try to maximize the sum of the y_{ij}'s, as that would give us the most pawns placeable; technically, we need a minimization problem, so we minimize

$$-\sum_{i,j=1}^{n} y_{ij}.$$

This would give us a ***quadratic*** programming problem; the constraints are quadratic in places, although the objective function is still linear. It is possible to do this problem, however, with linear constraints.

Let $\mathcal{Q}_{ij}$ be the set of all pairs on the $n \times n$ chessboard that can attack square (i, j) **and** the square (i, j) as well. We're using a script Q to emphasize that these are the places to put a queen to eliminate the possibility of a pawn being safely placed at (i, j). For example, if $n = 4$ then

$$\mathcal{Q}_{2,3} \ = \ \{(2,1),(2,2),(2,3),(2,4),(1,3),(3,3),(4,3),(1,2),(3,4),(1,4),(3,2),(4,1)\}$$

(see Table 1 for a visualization).

Our objective function is the same as before:

$$-\sum_{i,j=1}^{n} y_{ij}.$$

We want this to be as small as possible, which means we want the sum of the y_{ij}'s to be as large as possible. In other words, we want to have as many squares as possible not under attack by queens.

As we are placing n queens on the board, at most n queens can make the square (i, j) unsafe for a pawn. Consider the constraint: for all $(i, j) \in \{1, \dots, n\}^2$ we have

$$2n(1 - y_{ij}) \ \geq \sum_{(i',j')\in\mathcal{Q}_{ij}} x_{i'j'}.$$

What does this do?

- If there are no queens on the board attacking the square (i, j) then the right hand side is zero and there is no effect on y_{ij}, as the left hand side is always non-negative. We thus have complete freedom in choosing y_{ij} in this case. As we are trying to minimize the negative of the sum of the y_{ij}'s (or, equivalently, maximize the sum of the y_{ij}'s), we the program will take $y_{ij} = 1$ and place a pawn safely there.
- What if there is at least one queen attacking the square (i, j)? Then the sum on the right hand side is positive. Further, **it is at most n as there are only n queens**. If $y_{ij} = 1$ then the left hand side is 0, which is smaller than n and contradicts the inequality! Thus we cannot take $y_{ij} = 1$, and this case forces y_{ij} to be zero. This is exactly what we want, as it now tells us we cannot have a pawn safely placed at (i, j).

As we took a long path to the answer, it's worth writing down the constraints cleanly:

- Parameters: $\mathcal{Q}_{ij}$: all the pairs (i, j) on an $n \times n$ chessboard that can attacked a pawn located at (i, j), including (i, j); equivalently, these are all the squares where a queen placed there would attack a pawn at (i, j).
- Variables: $x_{ij} = 1$ if a queen is at (i, j) and 0 otherwise; $y_{ij} \in \{0, 1\}$ (constraints chosen later will force y_{ij} to be 0 if the location of the queens prevents a pawn from being placed safely at (i, j)).
- Constraint: Location of Queens: $\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} = n$. This forces exactly n queens to be placed on the $n \times n$ board.

- Constraint: Location of Pawns: $2n(1-y_{ij}) \geq \sum_{(i',j')\in\mathcal{Q}_{ij}} x_{i'j'}$. We may rewrite this in more standard form as
$$2ny_{ij} + \sum_{(i',j')\in\mathcal{Q}_{ij}} x_{i'j'} \ \leq\ 2n.$$
If a queen is placed and attacks (i,j) then y_{ij} must be zero (as otherwise the left hand side exceeds the right hand side). If no queen is placed that attacks square (i,j) then y_{ij} is free.
- Objective function: Minimize $-\sum_{i=1}^n \sum_{j=1}^n y_{ij}$. This is the negative of the number of pawns that may safely be placed on the board.

Note that our choice of objective function will make us set y_{ij} to 1 whenever possible. If we wanted to truly make y_{ij} indicate whether or not a pawn ***is*** safely placed at (i,j), all we need to do is ***force*** ourselves to place a pawn at (i,j) if possible. We can do this by adding the constraint: for all (i,j):
$$-ny_{ij} + \sum_{(i,j)\in\mathcal{Q}_{ij}} x_{ij}) \ \leq\ 1/2.$$
Why does this work? If there are no queens placed that attack (i,j) then y_{ij} is free. If, however, at least one queen is there then we must have $y_{ij} = 1$ as otherwise the inequality fails (note the sum is at most n, so taking $y_{ij} = 1$ will ensure it is satisfied).

#4: Exercise 6.6.30: Consider the following Linear Programming problem: $x_j \geq 0$,
$$\begin{pmatrix} 1 & 4 & 5 & 8 & 1 \\ 2 & 2 & 3 & 8 & 0 \\ 3 & 2 & 1 & 6 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 311 \\ 389 \\ 989 \end{pmatrix}, \tag{5.1}$$
and we want to minimize
$$5x_1 + 8x_2 + 9x_3 + 2x_4 + 11x_5. \tag{5.2}$$
Find (or prove one does not exist) an optimal solution.

Solution: There are several ways to go. We give a one-line solution from the TA at the end; as a large part of homework is to learn the methods and techniques, it is good to see the straightforward approach.

We have 5 columns, and a basic optimal solution (if it exists) must come from a basic feasible solution. There are $\binom{5}{3} = 10$ ways to choose 3 columns from 5 to find a basic feasible solution, and the basic feasible solution must have exactly 3 non-zero entries. We could look at all of these candidates and see which, if any, is the optimal solution. We know that our objective function will achieve a maximum and minimum on any compact subset of $\{(x_i)_{i=1}^5 \mid 0 \leq x_i \leq 989\}$ using standard results from analysis (a continuous function on a compact set attains its maximum and minimum). But are any solutions in such subsets feasible?

We need to find a basic feasible solution. If we try the first three columns of A, we get $A'x = b$. As A is a 3×3 matrix with linearly independent columns it is invertible, and we get $x = A'^{-1}b$. Unfortunately $A'^{-1}b$ has a negative entry, and thus cannot be a basic feasible solution. Remember our method only generates ***candidates*** for basic feasible solutions; it cannot ensure that they ***are*** basic feasible.

Undaunted, we continue. We find that there are no basic feasible solutions – all of the candidates have a negative entry, and thus there are no solutions. Here is code to generate the matrices:

```
B = {{1, 4, 5}, {2, 2, 3}, {3, 2, 1}};
B = {{1, 4, 8}, {2, 2, 8}, {3, 2, 6}};
B = {{1, 4, 1}, {2, 2, 0}, {3, 2, 0}};
B = {{1, 5, 8}, {2, 3, 8}, {3, 1, 6}};
B = {{1, 5, 1}, {1, 3, 0}, {3, 1, 0}};
B = {{1, 8, 1}, {2, 8, 0}, {3, 6, 0}};
B = {{4, 5, 8}, {2, 3, 8}, {2, 1, 7}};
B = {{4, 5, 1}, {2, 3, 0}, {2, 1, 0}};
B = {{4, 8, 1}, {2, 8, 0}, {2, 6, 0}};
B = {{5, 8, 1}, {3, 8, 0}, {1, 6, 0}};
```

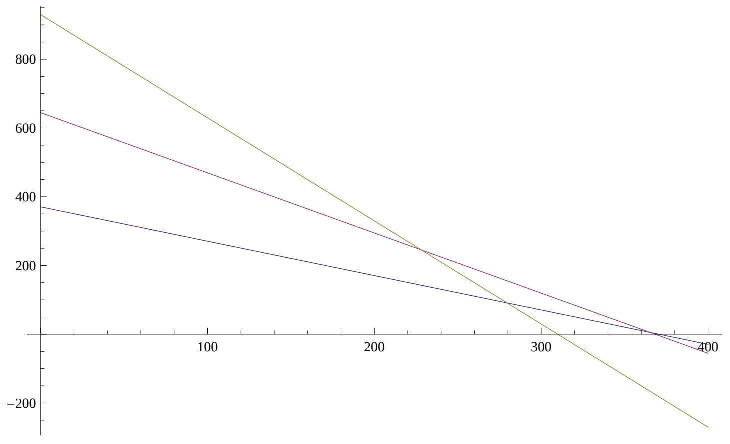

FIGURE 8. Plot of the three inequalities. The valid points are *below* the first and third lines (gold and blue) and *above* the middle (purple) line.

Here is code to check one of the cases:

```
B = {{4, 8, 1}, {2, 8, 0}, {2, 6, 0}};
Print["Our pruned matrix is ", MatrixForm[B]];
b = Transpose[{{311, 389, 989}}];
MatrixForm[b];
basicsoln = Inverse[B].b;
Print["Candidate for basic feasible is ", MatrixForm[basicsoln]];
```

We can try and solve this directly:

```
Clear[x1]; Clear[x2]; Clear[x3]; Clear[x4]; Clear[x5];
Solve[x1 + 4 x2 + 5 x3 + 8 x4 + x5 == 311 &&
  2 x1 + 2 x2 + 3 x3 + 8 x4 == 389
  && 3 x1 + 2 x2 + x3 + 6 x4 == 989, {x1, x2, x3, x4, x5}]
```

The output is $x1, x2$ free and

```
{{x3 -> -(2789/5) + (6 x1)/5 + (2 x2)/5,
  x4 -> 1289/5 - (7 x1)/10 - (2 x2)/5,
  x5 -> 5188/5 - (7 x1)/5 - (14 x2)/5}}
```

If we plot the three lines that arise from forcing x_3, x_4 and x_5 to be non-negative, we see that there is no solution to these inequalities that has all five variables positive. The Mathematica code is

```
Plot[{-x1 + 5188/14, (-7/4) x1 + 1289/2, -3 x1 + 2789/3}, {x1,0,400}]
```

and we give the plot in Figure 8.

Now, the one-line solution. These numbers were not randomly chosen (though I forgot when initially looking at this problem). I wanted something without any feasible solutions. If $(x_1, x_2, x_3, x_4, x_5) \geq (0, 0, 0, 0, 0)$ then there cannot be a solution to $Ax = b$. To see this, note that the sum of the entries in the j^{th} column in the first and second rows exceeds the value in the j^{th} column in the third row, ***but*** the sum of the first two entries of b is less than the third. There cannot be a solution. More mathematically, adding the first two constraints gives

$$3x_1 + 6x_2 + 8x_3 + 16x_4 + x_5 \ = \ 700,$$

while the third row is

$$3x_1 + 2x_2 + x_3 + 6x_4 \ = \ 989.$$

Subtracting yields

$$4x_2 + 7x_3 + 10x_4 + x_5 \ = \ -289,$$

which is impossible as all the x_i are supposed to be non-negative.

Remark: This (and the earlier problem with the 3×3 matrix) indicate the value of really looking at a problem and its algebra first before ploughing away. Often we can make our lives much easier by studying the problem, looking at symmetries, finding something to exploit. We ***can*** plug away, but we can save time. I consider Henry David Thoreau the patron saint of mathematics for his sage advice of `Simplify, simplify`. (Of course, this should be simplified to **Simplify**, but I'll grant him this as he has a point to make.) Look for savings first before doing calculations; this is in line with the spirit of duality and the savings available there.

5.1. **Next Assignment: HW #7: Due Monday, October 31, 2016.** Due Monday October 31: Problem #1: Exercise 8.7.18. Frequently in problems we desire two distinct tuples, say points $(a_1, \dots, a_k) \neq (\alpha_1, \dots, \alpha_k)$. Find a way to incorporate such a condition within the confines of integer linear programming. Problem #2: Medical Residencies: Imagine there are P people who have just graduated from medical school and H hospitals. We are trying to match medical students with hospitals. Each student ranks the hospitals and each hospital ranks the students. Formulate this assignment problem as a linear programming problem; you may need to make some assumptions to finish the modeling. There are a lot of ways to do this; what do you want to maximize? Does a feasible solution always exist, and if so when? Does the existence of a feasible solution depend on the function you want to optimize? Problem #3: Exercise 9.3.5. Modify the decomposition problem so that we write S as a sum of non-negative summands, but now we want to maximize the product of the squares of the summands; what is the answer? #4: Exercise 9.3.13. Generalize the knapsack problem so that in addition to needing the total weight to be below a critical threshold, there is also a volume constraint. Set this up as a linear programming problem. #5: Write down linear constraints for the event A or B or C must happen. #6: Consider an $n \times n \times n$ chesscube. Write down a linear programming problem to figure out how many hyperpawns can safely be placed given that n hyperqueens are placed in the chesscube. Note the hyperqueens can attack diagonally, horizontally, vertically, and forward-backly.

6. HW #7: DUE MONDAY, OCTOBER 31, 2016

6.1. **Assignment.** Due Monday October 31: Problem #1: Exercise 8.7.18. Frequently in problems we desire two distinct tuples, say points $(a_1, \dots, a_k) \neq (\alpha_1, \dots, \alpha_k)$. Find a way to incorporate such a condition within the confines of integer linear programming. Problem #2: Medical Residencies: Imagine there are P people who have just graduated from medical school and H hospitals. We are trying to match medical students with hospitals. Each student ranks the hospitals and each hospital ranks the students. Formulate this assignment problem as a linear programming problem; you may need to make some assumptions to finish the modeling. There are a lot of ways to do this; what do you want to maximize? Does a feasible solution always exist, and if so when? Does the existence of a feasible solution depend on the function you want to optimize? Problem #3: Exercise 9.3.5. Modify the decomposition problem so that we write S as a sum of non-negative summands, but now we want to maximize the product of the squares of the summands; what is the answer? #4: Exercise 9.3.13. Generalize the knapsack problem so that in addition to needing the total weight to be below a critical threshold, there is also a volume constraint. Set this up as a linear programming problem. #5: Write down linear constraints for the event A or B or C must happen. #6: Consider an $n \times n \times n$ chesscube. Write down a linear programming problem to figure out how many hyperpawns can safely be placed given that n hyperqueens are placed in the chesscube. Note the hyperqueens can attack diagonally, horizontally, vertically, and forward-backly.

6.2. **Solutions.** #1: Problem #1: Exercise 8.7.18. Frequently in problems we desire two distinct tuples, say points $(a_1, \dots, a_k) \neq (\alpha_1, \dots, \alpha_k)$. Find a way to incorporate such a condition within the confines of integer linear programming.
Solution: There are many ways; here's an easy way but a slow one. For each i we can form random variables z_i and y_i such that z_i is 1 if $a_1 - \alpha_i \geq 0$ and 0 otherwise, while y_i is 1 if $\alpha_i - a_i \geq 0$ and 0 otherwise. We then let $x_i = 1$ if z_i and y_i are both 1, and zero otherwise. Then look at $(n-1) - x_1 + \cdots + x_n$; if the two points are equal the sum is -1, otherwise it is non-negative. Thus if we let $w = 1$ if this sum is non-negative and 0 otherwise, w will encode whether or not the two points are distinct.

#2: Medical Residencies: Imagine there are P people who have just graduated from medical school and H hospitals. We are trying to match medical students with hospitals. Each student ranks the hospitals and each hospital ranks the students. Formulate this assignment problem as a linear programming problem; you may need to make some assumptions to finish the modeling. There are a lot of ways to do this; what do you want to maximize? Does a feasible solution always exist, and if so when? Does the existence of a feasible solution depend on the function you want to optimize?
Solution: First, the existence of a feasible solution is independent on whether or not an optimal solution exists. Let x_{ph} equal 1 if we assign student p to hospital h, and 0 otherwise. What are the constraints?

- No student can be assigned to more than one hospital: for all $p \in \{1, \dots, P\}$ we have $\sum_{h=1}^{H} x_{ph} \leq 1$. We write less than or equal to and not equal to as perhaps some students *will not be assigned to hospitals!*.
- Perhaps each hospital has a certain number of students needed, say d_i. Then for all $h \in \{1, \dots, H\}$ we have $\sum_{p=1}^{P} x_{ph} \geq d_i$. We might want equality here (no need to hire people you don't need, unless you want to keep them in the labor pool and have them gain experience for later).

That's it! This is all we need to determine whether or not we can assign students to hospitals! The difficulty is in choosing an objective function. What do we want to minimize? A simple possibility is to have each student rank the H hospitals and each hospital rank each student, giving a 1 for first choice, 2 for second and so on. We then want to minimize the total score. Letting r_{ph} be the rank person p attaches to working at hospital H, and ρ_{ph} the rank hospital h attaches to having person p, we need to minimize $\sum_p \sum_h (r_{ph} + \rho_{ph}) x_{ph}$. Notice that this assumes the students and the hospitals are equally important; if not we can introduce weights (non-negative and summing to 1). In fact, we can even go further and say some hospitals are more important than others, and perhaps some students are too (those coming from a 'good' school). We reach

$$\sum_p \sum_h (w_p r_{ph} + \omega_h \rho_{ph}) x_{ph};$$

while we need the w_p and ω_p to be non-negative, it's fine if they don't sum to 1 (all that does is rescale the objective function).

There are other rankings we can use. Perhaps each person gets 100 points and must assign them among the H hospitals. Or perhaps each person writes down how happy they would be working at each hospital, with 100 high and

0 low. There are lots of tweaks like this that we can do that will keep the objective function linear. Note something similar to this *is* used in assigning doctors to residency programs.

Good podcast found by classmate:
http://freakonomics.com/podcast/make-me-a-match-a-new-freakonomics-radio-episode/.

#3: Problem #3: Exercise 9.3.5. Modify the decomposition problem so that we write S as a sum of non-negative summands, but now we want to maximize the product of the squares of the summands; what is the answer?
Solution: Maximizing a square is the same as maximizing the original value (if it is non-negative, as is the case here). Thus it is the same answer as before! To see this more formally, there are no new critical points as our expression is never zero, and the answer is not on the boundary.

#4: Exercise 9.3.13. Generalize the knapsack problem so that in addition to needing the total weight to be below a critical threshold, there is also a volume constraint. Set this up as a linear programming problem.
Solution: There are a lot of ways to do this. The simplest is to assume that all that matters is the total volume; we cannot shrink the volume of items as we place them, but we can squish and twist the pieces. If this is the case, all we need is a constraint saying the sum of the volumes in the bag is at most the total volume of the bag: $v_1x_1 + \cdots + v_nx_n \le v$. If we want a more complicated one, we would need to have variables for the location of each object and the orientation as we place it, and make sure nothing overlaps.

#5: Write down linear constraints for the event A or B or C must happen.
Solution: We start with decision variables x_A, x_B, x_C where $x_E = 1$ if event E happens and 0 if event E does not occur. We have the inclusive or; thus our constraint is simply $x_A + x_B + x_C \ge 1$. The only way this constraint fails is if $x_A = x_B = x_C = 0$, in other words, if none of the events happen.

#6: Consider an $n \times n \times n$ chesscube. Write down a linear programming problem to figure out how many hyperpawns can safely be placed given that n hyperqueens are placed in the chesscube. Note the hyperqueens can attack diagonally, horizontally, vertically, and forward-backly.
Solution: We need to slightly generalize our arguments from the last assignment. Let $x_{ijk} = 1$ if we place a queen at (i, j, k) and 0 otherwise, and let $y_{ijk} = 1$ if there is a pawn at (i, j, k) and 0 otherwise. Let $\mathcal{Q}_{ijk}$ be the set of all locations that can attack (i, j, k).

Our first constraint is
$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} \ = \ 1;$$
this ensures we place exactly n queens on the board.

The second constraint is for the location of the pawns: for $1 \le i, j, k \le n$:
$$2n(1 - y_{ijk}) \ \ge \sum_{(i',j',k') \in \mathcal{Q}_{ijk}} x_{i'j'k'}.$$
If no queens attack (i, j, k) then the sum on the right is zero and there is no effect on y_{ijk}. If however there is at least one queen attacking the location (i, j, k) then the only way the inequality is satisfied is to have $y_{ijk} = 0$ (note in this case the sum on the right is non-zero, and is at most n as there are only n queens on the board).

The objective function to minimize is $-\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} y_{ij}$. This is the negative of the number of pawns that may safely be placed on the board. Note now that if we *can* place a pawn at (i, j, k) we will.

Homework for next week: Work on Project, work on Chapter, start letter of Rec.

7. HW #8: Due Monday, November 14, 2016

7.1. **Assignment.** #1. In the plane we have $\overrightarrow{v}^{(0)} = (3,-2)$, $\overrightarrow{v}^{(1)} = (1,5)$ and $\overrightarrow{v}^{(2)} = (-7,1)$. Let $\overrightarrow{x}$ have cartesian coordinates (c_1, c_2) and barycentric coordinates (x_0, x_1, x_2). Write the cartesian coordinates in terms of the barycentric coordinates, and vice versa. #2: Consider the map from the unit circle (all points $(x,y) : x^2+y^2 \le 1$ to itself given by $f(x,y) = ((y-1/2)^2/8, (x-1/2)^2/8)$. Does this map have any fixed points? Why or why not. If yes find or approximate it. #3: Consider $x_n = \cos(n)$ (measured in radians). The Bolzano-Weierstrass Theorem asserts it has a subsequence which converges to a point in $[-1,1]$. Explicitly find such a subsequence. Is it easier, harder or the same to prove the analogous statement for $y_n = \sin(n)$? Do so. Hint: you may use properties of π, such as its decimal or continued fraction expansion. Also, if you have a homework exemption could be a good time to use it....

7.2. **Solutions.** **Solution:** #1: First remember the x_i's are in $[0,1]$ and sum to 1, so $x_2 = 1 - x_1 - x_0$. We have a system of equations and find

$$\begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 3x_0 + 1x_1 - 7x_2 \\ -2x_0 + 5x_1 + 1x_2 \end{pmatrix} = \begin{pmatrix} 10x_0 + 8x_1 - 7 \\ -3x_0 + 4x_1 + 1 \end{pmatrix}.$$

Thus we have two equations with two unknowns. As written, it is very easy to get the cartesian from the barycentric:

$$c_1 \ = \ 10x_0 + 8x_1 - 7, \quad c_2 \ = \ -3x_0 + 4x_1 + 1.$$

For the other direction, after some algebra we find

$$\begin{pmatrix} 10 & 8 \\ -3 & 4 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \ = \ \begin{pmatrix} c_1 + 8 \\ c_2 - 1 \end{pmatrix}.$$

The matrix is invertible, and thus

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \ = \ \begin{pmatrix} 1/16 & -1/8 \\ 3/64 & 5/32 \end{pmatrix} \begin{pmatrix} c_1 + 8 \\ c_2 - 1 \end{pmatrix} \ = \ \begin{pmatrix} (8+c_1)/16 + (1-c_2)/8 \\ 3(8+c_1)/64 + 5(-1+c_2)/32 \end{pmatrix}.$$

Solution: #2: Yes, it has a fixed point. To see this note that it is a continuous map from the unit circle to itself (the largest either component can be is $(-1-1/2)^2/8 = 9/32$, and $(9/32)^2 + (9/32)^2 < 1$). Thus the Brouwer fixed point theorem applies, and a fixed point exists. To find the fixed point, we must solve

$$x \ = \ (y-1/2)^2/8, \quad y \ = \ (x-1/2)^2/8.$$

Using Mathematica we find $x = y = \frac{9}{2} - 2\sqrt{5} \approx 0.027864$. The code is

```
Simplify[Solve[{x == (y - 1/2)^2/8,  y ==  (x - 1/2)^2/8}, {x, y}]]
```

We could also find this directly. We have $8x = (y-1/2)^2$ and $8y = (x-1/2)^2$. While we could square both sides or directly replace one variable with another, we can directly try looking for a solution with $x = y$. That gives us

$$8x \ = \ (x-1/2)^2 \quad \text{or} \quad x^2 - 9x + \frac{1}{4} \ = \ 0;$$

this is a simple quadratic equation, and the root is the claimed $\frac{9}{2} - 2\sqrt{5}$. We could show this is the unique fixed point by showing that the above is a contraction map – doing so would possibly require some multivariable calculus and looking at the gradient, or just directly showing that two distinct points are moved closer.

Solution: #3: The two problems are equally hard. A beautiful theorem of Dirichlet (usually proved using the pigeon-hole principle) states that if α is irrational then there are infinitely many relatively prime p_n, q_n such that $|\alpha - p_n/q_n| \le C/q_n^2$ for a fixed $C > 0$ and $q_n < q_{n+1}$. (It is a nice exercise to prove this. Interestingly, the number which requires the largest C is the golden mean!) As π is irrational, we find a sequence $\{p_n, q_n\}$ as above. Thus $|q_n\pi - p_n| \le C/q_n$, so for each n we may find ϵ_n at most 1 in absolute value such that $p_n = q_n\pi + \epsilon_n C/q_n$. Thus

$$\sin(p_n) \ = \ \sin(q_n\pi + \epsilon_n C/q_n) \ = \ \sin(\epsilon_n C/q_n);$$

as $\sin(x) = x - x^3/3! + \cdots$, we see the above sine tends to zero. If we wanted to work with cosine we could approximate $\pi/2$.

Note this problem requires a lot of input. We need Dirichlet's theorem, which can be proved elementarily, but we also need π is irrational. For fun, let's prove a bit more. Fix a large n (how large n must be will be determined later).

Let $f(x) = \frac{x^n(1-x)^n}{n!}$. Show f attains its maximum at $x = \frac{1}{2}$, for $x \in (0,1)$, $0 < f(x) < \frac{1}{n!}$, and all the derivatives of f evaluated at 0 or 1 are integers. Assume π^2 is rational; thus we may write $\pi^2 = \frac{a}{b}$ for integers a, b. Consider

$$G(x) \ = \ b^n \sum_{k=0}^{n} (-1)^k f^{(2k)}(x) \pi^{2n-2k}. \tag{7.1}$$

Show $G(0)$ and $G(1)$ are integers and

$$\frac{d}{dx}\left[G'(x)\sin(\pi x) - \pi G(x)\cos(\pi x)\right] \quad = \quad \pi^2 a^n f(x)\sin(\pi x). \tag{7.2}$$

Deduce a contradiction (to the rationality of π^2) by showing that

$$\pi \int_0^1 a^n f(x)\sin(\pi x)dx \ = \ G(0) + G(1), \tag{7.3}$$

which cannot hold for n sufficiently large. The contradiction is the usual one, namely the integral on the left is in $(0,1)$ and the right hand side is an integer. Thus π^2 is irrational (and hence so too is π!).

```
f[n_] := Numerator[FromContinuedFraction[ContinuedFraction[Pi, n]]]
g[n_] := 1.0 Sin[f[n]]
For[n = 1, n <= 10, n++,
  Print["n = ", n, ", x_n = ", f[n], " and sin(x_n) = ", g[n]]];

n = 1,  x_n =       3 and sin(x_n) =  0.14112
n = 2,  x_n =      22 and sin(x_n) = -0.00885131
n = 3,  x_n =     333 and sin(x_n) = -0.00882117
n = 4,  x_n =     355 and sin(x_n) = -0.0000301444
n = 5,  x_n =  103993 and sin(x_n) = -0.0000191293
n = 6,  x_n =  104348 and sin(x_n) = -0.000011015
n = 7,  x_n =  208341 and sin(x_n) =  8.11432*10^-6
n = 8,  x_n =  312689 and sin(x_n) =  2.9007*10^-6
n = 9,  x_n =  833719 and sin(x_n) =  2.31292*10^-6
n = 10, x_n = 1146408 and sin(x_n) = -5.8778*10^-7
```

Why do we care about the irrationality of π^2? Another beautiful result is Euler's solution to the Basel problem:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} \ = \ \frac{\pi^2}{6}.$$

However, by the Fundamental Theorem of Arithmetic we have the Euler product representation of the zeta function: if the real part of s is greater than 1, then

$$\zeta(s) \ = \ \sum_{n=1}^{\infty} \frac{1}{n^s} \ = \ \prod_{p \text{ prime}} \left(1 - \frac{1}{p^s}\right)^{-1}.$$

Taking $s = 2$ yields

$$\frac{\pi^2}{6} \ = \ \prod_{p \text{ prime}} \left(1 - \frac{1}{p^2}\right)^{-1} \ = \ \prod_{p \text{ prime}} \frac{p^2}{p^2 - 1}.$$

If there were only finitely many primes than the product is rational, but we just showed π^2 is irrational; thus we have just shown the irrationality of π^2 implies the infinitude of primes! (Yes, this is the danger, or benefit, of having a number theorist teach operations research!)

7.3. **Homework Due Monday November 21, 2016.**

Read: Chapter 16, http://www.maa.org/sites/default/files/pdf/upload_library/22/Hasse/00029890.di011943.01p0581t.pdf

Homework: Hand in first draft of letter of recommendation, hand in outline of project, hand in outline of chapter work.