Desert Crossing

Callum Teape, Jet Hughes, Jake Norton, Cody Airey

Question 1.

Using the vehicle without refuelling, how far into the desert can you safely go?

Without refuelling, the vehicle has a capacity of 60L + 4x20L, which is 140L. The vehicle can travel 12km per 1L of fuel. This means without refuelling the vehicle can travel 12km x 140L = 1680km.

Question 2.

Describe a procedure whereby you could cross the desert in the vehicle.

To cross the desert, you could make several trips out into the desert, to drop fuel at "checkpoints" along the route. With enough fuel ready at points spaced close enough, you could make a trip across the desert by refuelling at the checkpoints along the way.

Question 3.

Describe a procedure whereby you could cross the desert and return in the vehicle.

To cross the desert and return, you could virtually the same procedure as in question 2. In order to facilitate a return trip, you would need to drop enough extra fuel at the checkpoints to be able to make a return trip while refuelling along the way.

Question 4.

Describe a procedure whereby you could cross the desert in the vehicle using the minimum amount of fuel.

Firstly, we note that the further into the desert you drop fuel, the amount of fuel you use getting there and back increases. So there is clearly some sort of cost associated with dropping fuel further out into the desert. With this in mind, we seek to maximise the distance between the fuel depot locations and the end of the desert. This leads us to a few critical assumptions:

1. We assume that the optimal solution will have the final fuel depot one full tank away from the end, as this is the maximum possible distance away from the end of the desert.
2. We assume that by the time we're done, there should be no fuel left in the desert. If there is then that means we've wasted fuel, and so our solution is not minimal.
3. We assume that every time we leave a depot driving out into the desert, we have full tank. This seems logical, as if we are going to be dropping fuel at the next depot, then we have to bear the cost of driving there, and we want to maximise the amount of fuel we drop per unit of cost to get the most out of every mile we drive. It doesn't make any sense to have only half a tank and then only be able to drop 20 litres, when we could have had a full tank and dropped 80 litres for the same driving cost.
4. We drop fuel at only one depot per trip from base. This assumption is more or less a consequence of 3), as it doesn't make sense to drop fuel at a location and then fuel up to a full tank to drive to another. This is because there is more driving cost associated with this move then simply leaving more fuel there in a previous trip (where we didn't drive as far from base).

5. As a consequence of 4), we also assume that we drop fuel at each depot only once. At a distance of this range, this assumption is ok. If the desert was much longer it wouldn't be possible to do this without making the distance between each depot minute (which would lead to a less optimal solution then just dropping fuel in the same place twice). This assumption seems logical, because we only drop fuel at one depot per trip, so two drop fuel at a depot twice means taking two whole separate trips to do it, which by 3) involves two refuels at base.

Using these assumptions, we try to find a solution.

Let $(a_n) = (a_1, a_2, ...a_n)$ be the locations of the fuel depots in litres from the start of the desert. Let c be the fuel capacity of the vehicle (tank and barrels). Let d be the length of the desert.

By the first assumption, we fix a_n at $\frac{d-mc}{m}$

By assumption 3, we need to have a full tank when we leave a_{n-1}, which means that by the time we get to a_n we have used $d(a_n, a_{n-1})$ litres of fuel. This means that we need to leave $d(a_n, a_{n-1})$ litres of fuel at the last depot a_n for the final journey. The question is how to choose $d(a_n, a_{n-1})$.

As we stated in the beginning, there is a cost associated with dropping fuel further out into the desert. We know that we're going to have to drop fuel at depot a_{n-1}, so to minimise this cost, we want to make it as far away from the end of the desert as it can be with the given fuel capacity c. We know that we have to drop $d(a_n, a_{n-1})$ litres at depot a_n, and obviously we have to drive there and back, which uses $2 * d(a_n, a_{n-1})$, so we have $3 * d(a_n, a_{n-1}) = c.$ so $d(a_n, a_{n-1}) = \frac{c}{3}$.

For $d = 2413, c = 140, m = 12$, this gives us $a_n = 61.083$ and $a_{n-1} = 14.417$. The question now is if we should add in more depots between the start of the desert and a_{n-1}. Because of assumptions 4 and 5, including extra depots would mean that we have to make more trips from base, and each trip obviously adds to the total cost. So if we want to minimise cost, we want to minimise the number of depots. Therefore we take $a = (0, 14.417, 61.083)$.

So we cross the desert as follows:

1. Start with a full tank (+140L) go 14.137 litres outs, drop $3 * 14.137$ litres and return. Arrive back at base with 69.315 litres.
2. Refuel (+70.685) go 14.137 litres, fuel up 14.137 litres, drive a further 46.666 litres out to 61.083 litres, drop 46.666 litres, drive back to 14.137 litres, we're empty at this point, so pick up 14.137 litres and drive back to base.
3. Refuel (+140), drive 14.137 litres out, pick up all the remaining 14.137 litres so we have a full tank again. Drive to 61.083 litres, pick up the 46.666 litres there so we have a full tank again. Drive all the way to the end of the desert.

The total expenditure for this trip is 140 + 70.685 + 140 = 350.685 litres

Question 5.

Describe a procedure whereby you could cross the desert and return in the vehicle using the minimum amount of fuel.

Let the total length of the desert be d, let the mileage of the vehicle be m in km per litre, let the total fuel capacity of the vehicle be c.

We start by seeking some sort of standard procedure to cross the desert. With this we can find some sort of general formula for expenditure, and therefore can evaluate solutions programmatically rather than by hand. So we start with a few assumptions, which we came by logic and trial and error:

1. As before, we assume that the best solution has it's final fuel depot the maximum possible distance from the end of the desert, which in this case is half a tank. Which gives us that we need to have a full tank when we depart this depot for the end of the desert.
2. 2 from Q4)
3. 3 from Q4)
4. 4 from Q4)
5. On the final return journey, we assume that the optimum solution will have us run out of fuel exactly as we come to each fuel depot, which in turn gives us that to make it back, we need to leave at each fuel depot the exact amount of fuel that it takes to drive to the next. This seems logical, as it means we arrive back at base empty, implying we used the minimal fuel.

With these assumptions, we can work backwards from the finish line to construct a general procedure, and therefore a general expenditure. To simplify the explanation, let $(a_n) = (a_1, a_2, ...a_n)$ be the locations of the fuel depots in litres from the start of the desert, let c, m, d be as defined before.

By 1), we need a full tank when we leave the final depot for the end of the desert. By 4), we know that we have a full tank every time we leave the depot before the final depot so to have a full tank when we leave for the end, we will need to have $d(a_n, a_{n-1})$ litres there to refuel what we lost by driving there. By 2), we know that we will need to leave another $d(a_n, a_{n-1})$ litres at the final depot for the return journey. So we need $2 * d(a_n, a_{n-1})$ at the final depot.

By 4), we have a full tank every time we leave depot a_{n-1}, so we can drop $min(80, c - 2 * d(a_n, a_{n-1})$, litres at the final depot with every trip we make from the depot before. Therefore we need to make $\frac{2*d(a_n,a_{n-1})}{min(80,c-2*d(a_n,a_{n-1})}$ trips from depot a_{n-1} in order to drop enough fuel at the last depot.

By 4) we have a full tank every time we leave depot a_{n-2}, so to ensure that 4) holds for a_{n-1}, we need to leave enough there to refuel the amount lost by driving from the depot before however many times we will pass it. We know that we have to make $\frac{2*d(a_n,a_{n-1})}{min(80,c-2*d(a_n,a_{n-1})}$ trips beyond depot a_{n-1} to ensure there is enough fuel at the last depot for the final journey, and we also know that will pass depot a_{n-1} twice more (there and back) for that final journey. So the amount of fuel needed at depot a_{n-1} is $(2*\frac{2*d(a_n,a_{n-1})}{min(80,c-2*d(a_n,a_{n-1})}+2)*d(a_{n-1}, a_{n-2})$.

Now, by the same logic, we will need to make

$$\frac{(2*\frac{2*d(a_n,a_{n-1})}{min(80,c-2*d(a_n,a_{n-1})}+2)*d(a_{n-1},a_{n-2})}{min(80,c-2*d(a_{n-1},a_{n-2}))}$$

trips from depot a_{n-2} in order to drop of enough fuel, but we also will pass it $2 * \frac{2*d(a_n,a_{n-1})}{min(80,c-2*d(a_n,a_{n-1})}$ more when we make journeys from depot a_{n-1} to drop fuel at the final depot, and we also need to leave $2 * d(a_{n-2}, a_{n-3})$ for the final journey. So we require

$$(2 * \frac{(2*\frac{2*(d(a_n,a_{n-1})}{min(80,c-2*d(a_n,a_{n-1})}+2)*d(a_{n-1},a_{n-2})}{min(80,c-2*d(a_{n-1},a_{n-2}))} + 2 * \frac{2*d(a_n,a_{n-1})}{min(80,c-2*d(a_n,a_{n-1})} + 2) * d(a_{n-2}, a_{n-3})$$

litres of fuel at depot a_{n-2}.

We can express the logic above more concisely by iteratively defining a sequences $(r_{n\in\mathbb{N}})$, for the required fuel at each depot, and $(t_{n\in\mathbb{N}})$ for the number of trips from each depot. Then with initial conditions $r_0 = 2 * d(a_n, a_{n-1})$ and $t_0 = \frac{2*d(a_n,a_{n-1})}{min(80,c-2*d(a_n,a_{n-1})}$.

$$\text{i) } r_i = d(a_{n-(i+1)}, a_{n-i})(2(\textstyle\sum_{k=0}^{i-1} t_k) + 2)$$

$$\text{ii) } t_i = \frac{r_i}{min(80,c-2*d(a_{n-i},a_{n-(i+1)})} \text{ rounded strictly up to nearest whole number}$$

Note that the length of the (r) and (t) will end up being one less than the length of (a). We obtain a general formula for expenditure where n is the length of the array (a):

$$\text{iii) } E = c + \textstyle\sum_{i=0}^{n-1}[t_i * 2 * d(a_{n-i}, a_{n-(i+1)}) + r_i]$$

Finally, we can codify this in the following method:

```python
def calcExpenditure5(a, c):
    """
    calculating the expenditure given an array of fuel drop locations for round trip.

    :params a: integer array
        the array of fuel drop positions in litres from the start of the desert
    """
    numTrips = 0

    tripcost = []  #array which holds the cost of each leg of the journey

    n = len(a)

    for k in range(1,n):
        j = n-k
        distance = a[j] - a[j-1]

        #if the distance between the points is greater than half of the fuel tank, we can’t make
            it, so return FAIL
        if(distance*2 > c):
            return "FAIL"

        if j == n-1: #first case for the last drop
            required = 2*distance
        else:
            required = (numTrips*2+2)*distance #this will be the numTrips from the prior iteration.


        thisTrip = math.ceil(required/min(80, (c-2*distance))) #how many trips does it take to drop
            the required amount of fuel at the spot, we can drop max of 80L, hence the min function?
        numTrips += thisTrip  #we need enough fuel not only to drop the fuel at the next spot, but
            also for any trips beyond that from drop sites further up the chain, so need numtrips
            to be accumulative

        #building array to show cost at each step of the journey.
        tripcost.append(thisTrip*2*distance + required)


    return (sum(tripcost) + c)
```

Now that we have this method, we don't have to evaluate solutions by hand anymore. This will make it much easier to find an optimal solution.

At this point, we have to figure out how to space the fuel depots so that we use the minimum fuel. To achieve this, it seemed like a good idea to just test as many different spacings as possible. We hoped that this brute force approach would reveal general trends that make some solutions good and others bad, and might lead us to the principles that would shape our final optimisation. We could have drafted up combinations by hand, but this would have been time consuming and laborious. We thought it might be better to find an algorithmic way to generate combinations en masse. What we wanted was an algorithm where we could specify the ratio of the distances between each fuel depot, and it would distribute the depots accordingly. Then we could just give the computer as many different ratios as we could think of, i.e. exponential, logarithmic, linear, harmonic etc, and it would do all the heavy lifting.

Let $(a_n) = (a_0, a_1, a_2, ...a_n)$ be the locations of the fuel depots in litres from the start of the desert as in the above.

Using the knowledge that the cost of dropping fuel further out increases, we decide that having the distance between fuel depots grow by some exponential function might yield a good solution. So we have $d(a_i - a_{i-1}) = g_i(e)d(a_{i-1} - a_{i-2})$. For the sake of simplicity, we assume that g_i only acts on e as an exponent, so we say $g_i e = e^{f(i)}$. We also force assumption 1 that the last fuel depot should be as far away from the end of the desert as possible, giving $a_n = \frac{d - \frac{mc}{2}}{m}$. Putting all of these together gives:

$$a_0 = 0,$$

$$a_1 = \frac{d - cm}{m(1 + \sum_{k=1}^{n} e^{\sum_{j=1}^{k} f(j)})},$$

$$a_i = a_1(1 + \sum_{k=1}^{i} e^{\sum_{j=1}^{k} f(j)}) \text{ for } i \in [1, n] \subset \mathbb{N}$$

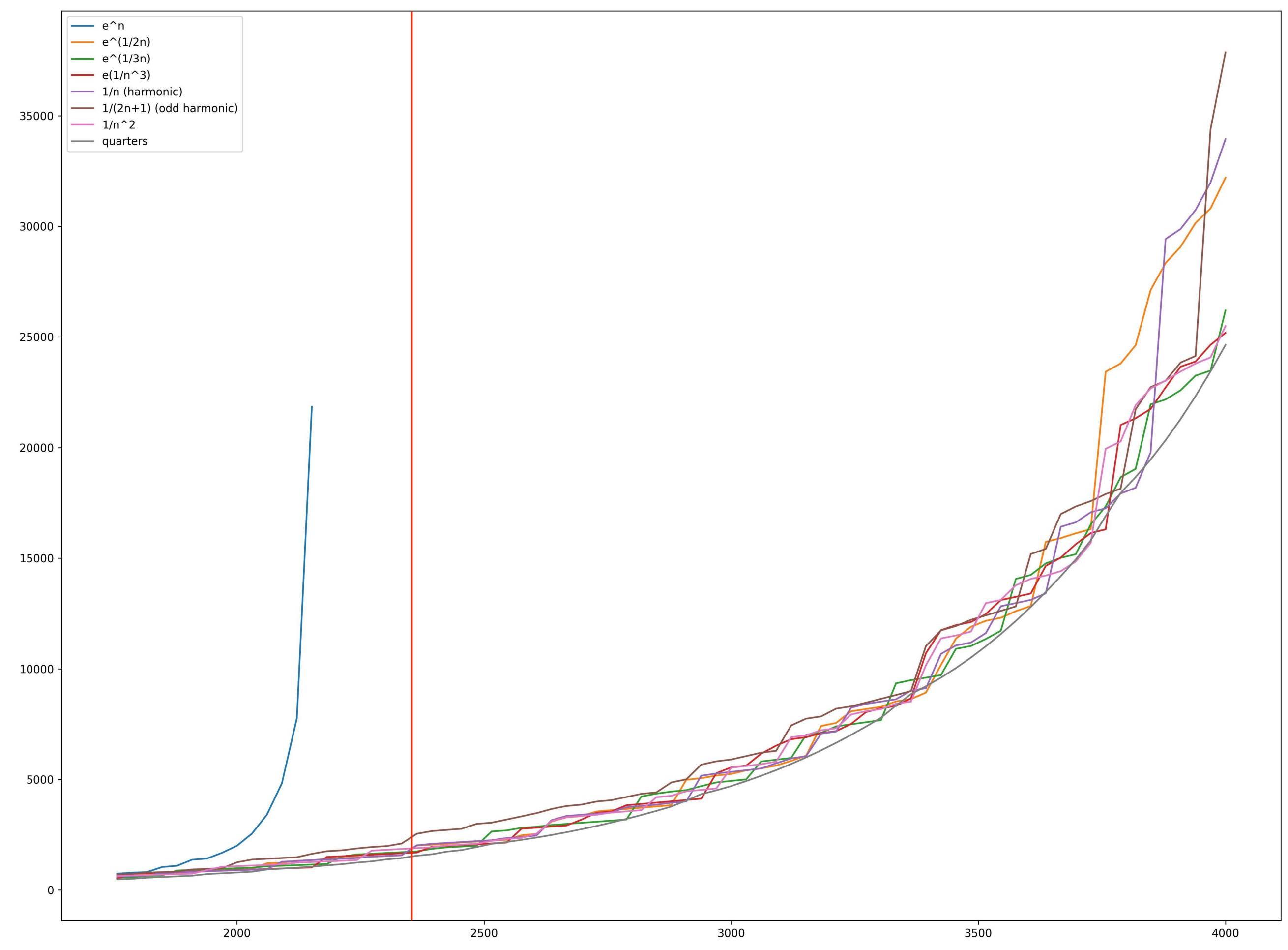
So now we essentially have an iterative function which distributes n fuel depots based on some exponential ratio for an arbitrary distance, capacity, and mileage. Note that we can test combinations that aren't exponential by taking the natural log of the e terms.

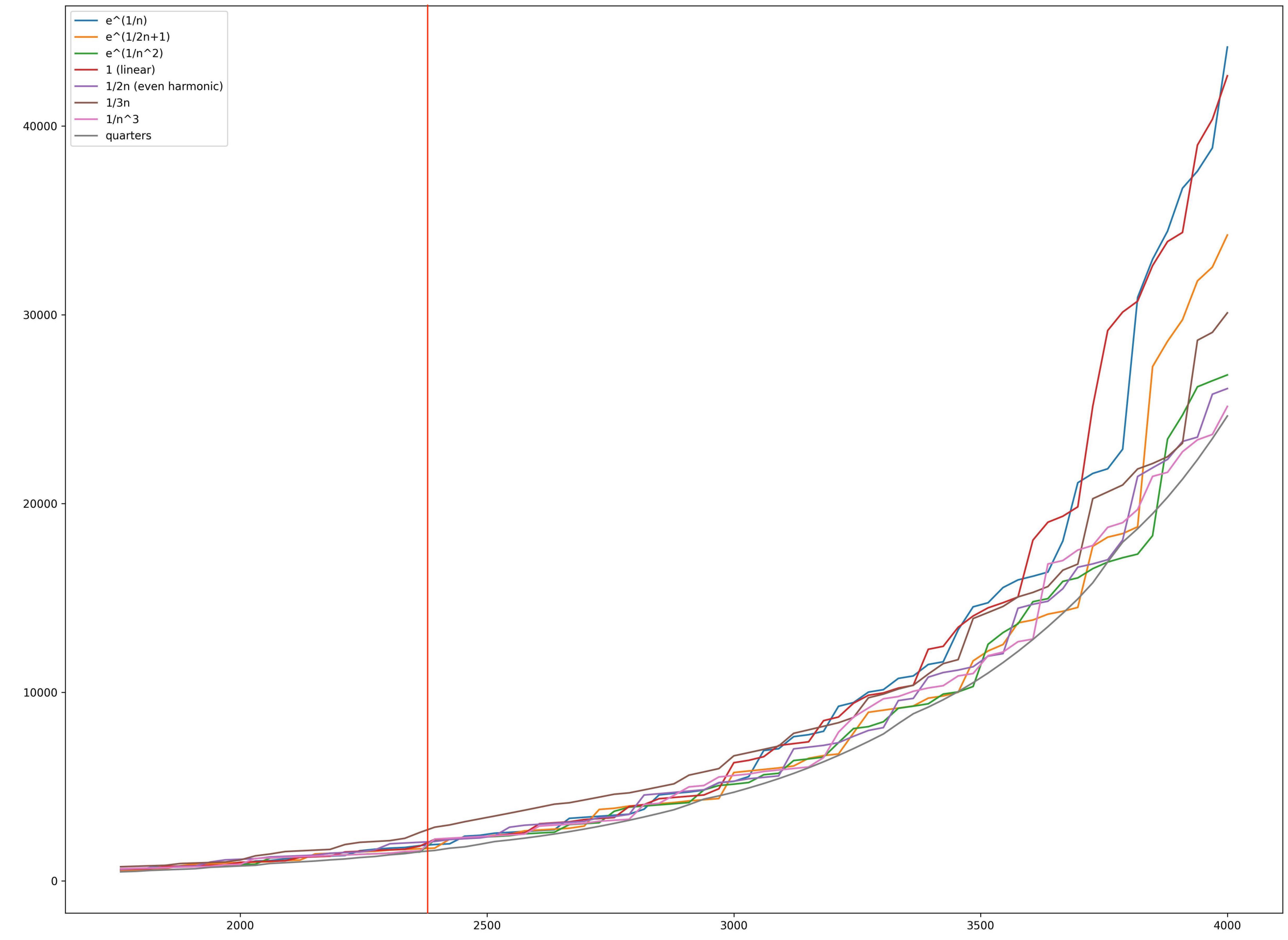
The next question is how to choose n. Obviously we have to choose it such that the maximum distance $|a_n - a_{n-1}| \leq \frac{cm}{2}$, because otherwise we would run out of fuel. From there on out it becomes a question of how much fuel is it optimal to drop on each run. For instance, it seems very wasteful to drive 65 litres out and only be able to drop 10 litres in order to make it back. In practice, we will just compute the expenditure $n = 1$ through 10 and take the minimum.

This distribution is automatically calculated using the functions below, with f(j) being the given function.

```
(in main...)
    for n in (range(1,10)):
        lastDrop = (d-(1/2)*m*c)/m #half of full tank for the return journey, a full tank for the
            one way

        a1 = lastDrop/(eTerm(n))
        a = [0, a1]
        for i in range(2, n+2):
            a.append(a[i-1] + a1*eNum(i-1, fj))


def eTerm(n):
    result = 1
    for k in range(1,n+1):
        result += eNum(k)

    return result


def eNum(k):
    exponent = 0
    for j in range(1,k+1):
        exponent += f(j)

    return math.pow(math.e, exponent) #return ln(e) to get rid of the e and just return the
        exponent (for testing out non exponential trends)

def f(j):
    #some function
```

Now, we calculate expenditure using the same code as before, and we observe the data for many different functions f(j). This data is plotted below, with distance on the x axis and expenditure on the y.


e^n
e^(1/2n)
e^(1/3n)
e(1/n^3)
1/n (harmonic)
1/(2n+1) (odd harmonic)
1/n^2
quarters
35000
30000
25000
20000
15000
10000
5000
0
2000
2500
3000
3500
4000



We can see that generally, and certainly on the red line which marks $d = 2413$, the solution that uses the minimum fuel is the grey solution. This distribution, 'quarters', is the distribution where the depots are placed an even quarter of the fuel capacity c away from each other. For the input parameters in the question $(d = 2413, c = 140, m = 12)$, this distribution gives $a = [0, 26.083, 61.083, 96.083, 131.083]$. The code earlier tells us that this distribution uses 1850.033 litres of fuel.

The next step was to find out why this solution performed so well. To do this, we analysed the required array, $(r_n)_{n\in\mathbb{N}}$, and the trips array, $(t_n)_{n\in\mathbb{N}}$, using the same notation as earlier ($(r_n)_{n\in\mathbb{N}}$, is the required fuel at each depot, and $(t_n)_{n\in\mathbb{N}}$ for the number of trips to each depot). When we did this we found $r = (70.0, 140.0, 280.0, 417.3)$ and $t = (1, 2, 4, 6)$.

We noticed that the amount of fuel required at the first three depots were even multiples of 70, and that these multiples were the first three values in the trips array. This lead us to the key idea that would shape our solution: the best solutions require at each fuel depot an even multiple of the amount that can be dropped there. Looking at the formula for expenditure (iii), this makes sense as we want to minimise the amount that we have to round each t_i up by. Every bit that we round it up by is essentially wasted fuel, as we had to bear the cost of driving there.

It is also clear that more trips between the later depot compounds into far more trips near the beginning. This is because additional trips between a_{n-1} and a_n require more fuel at depot a_{n-1}, which requires more trips to drop that fuel between a_{n-2} and a_{n-1}, which requires more trips and so on. So we expect that optimal solution to minimise the number of trips required between last few depots.

At first, we tried the solution where t is constant at 1, meaning that we only drop fuel once at every depot. Doing this forces r to be constantly 80. By the equations (i, ii) from earlier, this allows us to solve $d(a_i, a_{i-1})$ for all $i \in [1, n] \subset \mathbb{N}$. This gives $a = (0, 1.5, 3.75, 6.4, 9.2, 12.25, 15.55, 19.15, 23.1, 27.5, 32.5, 38.15, 44.8, 52.8, 62.77, 76.1, 96.1, 131.1$ As we can see, the sheer number of depots required to keep this condition throughout is staggering, and the distances between them becomes minute. The code tells us that this arrangement expends 1707.640 litres. This is the best result we have so far, but we think that we can shave off some litres by using less depots, and allowing some depots to require two trips between them.

Lets start by fixing the last two positions in t at 1, meaning that the last two depots are only allowed one fuel drop. Now we try switching to 2 drops per depot. To retain even multiples of 80, this forces the remaining positions in r to be 160. This allows us to calculate a by the equations (i, ii) from earlier, giving $a = (0, 5.95, 13.2, 22.075, 33.5, 49.5, 76.1, 96.1, 131.1)$. The code tells us that this arrangement expends 1657.9 litres.

From here on out we experimented with many different arrangements. We tried switching to three drops somewhere down the line, but we couldn't get a number lower than 1658. We therefore conclude that this is the optimal solution. For the sake of clarity, we outline exactly what this solution looks like in practice below.

1. Start with 88.9 litres, drive 5.95 litres out, drop 77 litres, come back to base empty
2. Repeat 1)
3. Fuel up to 94.5 litres, drive 5.95 litres out, fuel up 5.95 litres back to 94.5. Drive a further 7.25 litres out to 13.2 litres, drop 80 litres. Arrive back at 5.95 litres empty, fuel up 5.9 litres to make it back to base empty.
4. Repeat 3)
5. Fuel up to 97.75 litres, drive 5.95 litres out, fuel up 5.95 litres back to 97.75 litres. Drive a further 7.25 litres out to 13.2 litres, fuel up 7.25 litres back to 97.75 litres. Drive a further 8.875 litres out to 22.075 litres from base, drop 80 litres. Arrive back at 13.2 litres empty, fuel up 7.25 litres to make it back to 5.95 litres empty, fuel up 5.95 to make it back to base empty.
6. Repeat 5)
7. Fuel up 102.85 litres. Drive 22.075 litres out using the procedure described above. fuel up 8.875 back to 102.85 litres, drive a further 11.425 litres out to 33.5 litres, drop 80, come back using the procedure described above.
8. Repeat 7)
9. Fuel up to 112. Drive 33.5 litres out by the procedure above, fuel up 11.425 litres back to 112 litres. Drive a further 16 litres out to 49.5 litres, drop 80. Drive back using the procedure above.
10. Repeat 9)
11. Fuel up to 133.2. Drive 49.5 litres out by the procedure above, fuel up 16 litres back to 133.2. Drive a further 26.6 litres out to 76.1 litres. Drop 80. Come back using the procedure above.
12. Repeat 11)

13. Fuel up to 120 litres. Drive 76.1 litres out by the procedure above, fuel up 26.6 litres back to 120 litres. Drive a further 20 litres to 96.1 litres. Drop 80. Come back using the procedure above.
14. Fuel up to 140 litres. Drive 96.1 litres out using the procedure above, fuel up 20 litres back to 140 litres. Drive a further 35 litres out to 131.1 litres. Drop 70 litres. Drive back using the procedure above.
15. Fuel up to 140 litres, drive all the way there and back.

Statement of Contributions:
Jet Hughes

- Write up of questions 1,2,3, and 4.
- Logic and assumptions for questions 4 and 5
- Testing and trouble shooting all code throughout.
- Analysis of well-performing distributions
- Finding final answer for q4
- Finding final answer for q5

Cody Airey

- Verifying output of method "calcExpenditure5" by hand.
- Testing and trouble shooting all code throughout.
- Logic and assumptions for questions 4 and 5
- Analysis of well-performing distributions
- Finding final answer for q4
- Finding final answer for q5

Jake Norton

- Verifying output of method "calcExpenditure5" by hand.
- Logic and assumptions for questions 4 and 5
- Testing and trouble shooting all code throughout.
- Analysis of well-performing distributions
- Finding final answer for q4
- Finding final answer for q5

Callum Teape

- Write-up of question 5.
- Logic and assumptions of question 4 and 5.
- Creating graphs with python script.
- Writing method "calcExpenditure5"
- Analysis of well-performing distributions
- Finding final answer for q4
- Finding final answer for q5