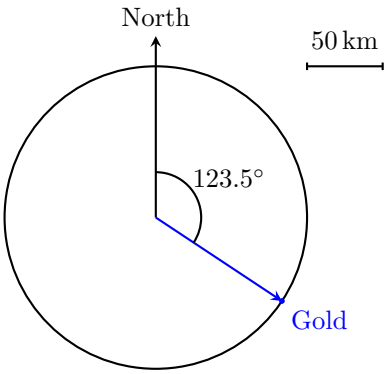
Bearings Challenge - Worked Solutions

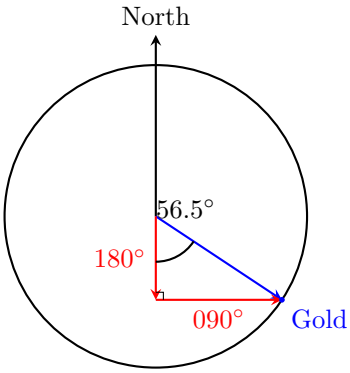
Solutions Guide

1 Introduction

1.



2. The journey describes involves travelling due South, then due east.



We see this sets up a right angled-triangle and so by right angled trigonometry, the distance travelled South is:

$$100 \cdot \cos(56.5°) = 55.19370 \text{ km} \quad \text{(nearest cm)}$$

and the distance travelled East is :

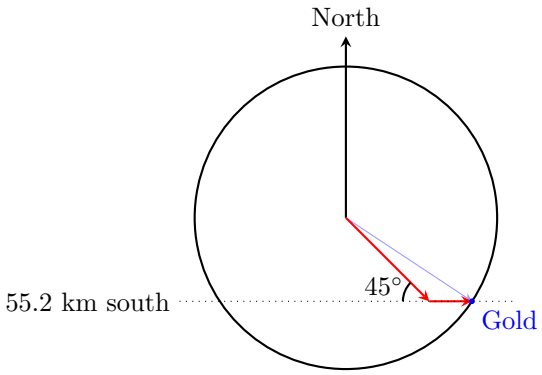
$$100 \cdot \sin(56.5°) = 83.38868 \text{ km} \quad \text{(nearest cm)}$$

3. The total distance Bob travels is:

$$55.19370 + 83.38868 = 138.58 \text{ km}$$

Thus takes $138.58 \div 20 = 6.93$ days (3.s.f)

4. This journey does not form a right angle, Rob will travel Southeast until he has the require 55.2km of Southern displacement, then travel East for the remainder of the journey.



Observe that the end of the final leg of the journey is at a point 55.19370 km South and 55.19370 km East. Using Pythagoras, the total distance for the first leg is:

$$\sqrt{55.19370^2 + 55.19370^2} = 78.0557 \text{ km} \quad \text{(nearest 10cm)}$$

The second leg is the remaining East distance:

$$83.38868 - 55.19370 = 28.2960 \text{ km} \quad \text{(nearest 10cm)}$$

Thus Rob's total distance is $78.0557 + 28.2960 = 106.35$ km (nearest metre)

Which takes $106.35 \div 20 = 5.32$ days (3.s.f)

And thus Rob arrives Roughly 1 day and 15 hours before Bob.

5. It should be clear that travelling as close to the 123.5° bearing will achieve the fastest path.

   One such Journey is as follows:

   - Travel a km an a bearing of 124°
   - Travel a km on a bearing of 123°

   Which forms an isosceles triangle with the base the path of the 123.5° "bearing", with apex angle 179°, and equal sides length a km;

   Thus applying the cosine rule:

$$100^2 = a^2 + a^2 - 2 \cdot a \cdot a \cdot \cos(179°) = (2 - 2 \cdot \cos(179°) \cdot a^2$$

   Thus

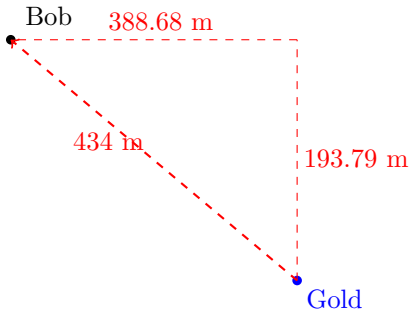
$$a = \sqrt{\frac{100^2}{2 - 2 \cdot \cos(179°)}} = 50.00190 \text{ km} \quad \text{(nearest cm)}$$

   And so the journey takes only a little (less than a minute!) longer than 5 days, and thus even with a 7 hour delay, will arrive before Rob, who needs 5 days 7 hours and 40 minutes.

   Proving the route is the quickest possible is left as an exercise, (hint - consider all triangles with base as in our journey, and apply Pythagoras' theorem. You may also want to consider the graph of $x^2 + (100 - x)^2$).

2 Challenge I: Integer Distances

1. (a) Rounding to the nearest km, Bob travels 55 km South, leaving Bob 193.79m North of the gold; and 83 km East, leaving Bob 388.68m West of the gold. Applying Pythagoras, Bob ends up $\sqrt{193.79^2 + 388.68^2} = 434$ m (3.s.f) from the gold.



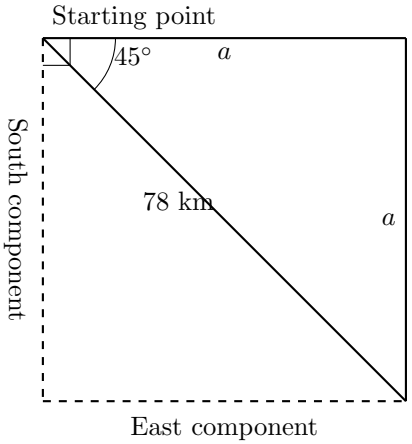
(b) Rounding to the nearest km, Rob travels 78km Southeast, and 28km East. Observe that travelling Southeast constructs an isosceles right angled triangle, as the distance travelled South is equal to the distance travelled East. Denoting this distance a we have that:

$$78^2 = a^2 + a^2$$

And thus $a = \sqrt{\frac{78^2}{2}} = 55.15432$ km (nearest cm)

So Rob travels 55.15432 km South and $55.15432 + 28 = 83.15432$ km East.

This leaves Rob $55.19370 - 55.15432 = 0.0394$ km (3.s.f) North of the gold, and $83.38868 - 83.15432 = 0.234$ km (3.s.f) West of the gold. Applying Pythagoras, he ends up 237 meters away.



After travelling SE

$$\begin{gathered} 78^2 = a^2 + a^2 \\ 6084 = 2a^2 \\ a^2 = 3042 \\ a = \sqrt{3042} \approx 55.154 \text{ km} \end{gathered}$$

(c) The quickest journey, now rounded, becomes:

- Travel 50 km an a bearing of 124°
- Travel 50 km on a bearing of 123°

Applying the cosine rule as per the previous section, this covers:

$$\sqrt{50^2 + 50^2 - 2 \cdot 50 \cdot 50 \cdot \cos(179°)} = 99.99619 \text{ km} \quad \text{(nearest cm)}$$

As this also travels from the origin, at an angle of 123.5° from North, we have that the distance from the gold is:

$$100 - 99.99619 - 0.00381 \text{ km} = 3.81 \text{ m}$$

Thus, neither you, Bob nor Rob end up within 1m of the gold, and it remains buried.

2. We consider the displacement East-West and South-North, setting East and North the positive directions respectively.

We use right-angled trigonometry to resolve each leg into a vector (x, y) indicating the East-West and South-North displacement, and see that (all to the nearest μm):

- 2km on a bearing 179°, $= (2 \cdot \sin(1°), -2 \cdot \cos(1°)) = (0.034904048, -1.999695390)$
- 2km on a bearing of 000°, $= (0.00000000, 2.000000000)$
- 1km on a bearing of 182°, $= (-\sin(2°), -\cos(2°)) = (-0.034899497, 0.999390827)$
- 1km on a bearing of 000°, $= (0.000000000, 1.000000000)$

Which when combined result in a total displacement of:

$$(2 \cdot \sin(1°) - \sin(2°), 3 - 2 \cdot \cos(1°) - \cos(2°)) = (0.000005316, 0.000913783)$$

Which is 0.5316cm East and 91.3783cm North

3. This is the same journey as the previous question, with all directions rotated 90° anticlockwise. And thus the final outcome is: 0.5316cm North and 91.3783cm West.

   We standardise this and write: 0.5316cm North and −91.3783cm East.

4. Let A be the number of repetitions of the 4 leg route from 2), and B the number of repetitions of the 4 leg route from 3), considering the East-West and North-South components respectively, we derive the following simultaneous equations:

$$0.000005316A - 0.000913783B = 100 \cdot \sin(56.5°) \tag{1}$$
$$0.000913783A + 0.000005316B = -100 \cdot \cos(56.5°) \tag{2}$$

Which can be solved via substitution or elimination, to reduce numerical error, we introduce the following constants:

$$\alpha = 2 \cdot \sin(1°) - \sin(2°) \qquad \approx 0.000005316 \tag{3}$$
$$\beta = 3 - 2 \cdot \cos(1°) - \cos(2°) \qquad \approx 0.000913783 \tag{4}$$

Thus we have:

$$\alpha A - \beta B = 100 \cdot \sin(56.5°) \tag{5}$$
$$\beta A + \alpha B = -100 \cdot \cos(56.5°) \tag{6}$$

Which yields $A = \frac{100 \cdot \sin(56.5°)}{\alpha} + \frac{\beta B}{\alpha}$

Thus:

$$\beta \cdot \left(\frac{100 \cdot \sin(56.5°)}{\alpha} + \frac{\beta B}{\alpha} \right) + \alpha B = -100 \cdot \cos(56.5°) \tag{7}$$

$$\left(\frac{\beta^2}{\alpha} + \alpha \right) \cdot B = -100 \cdot \cos(56.5°) - \frac{100\beta \cdot \sin(56.5°)}{\alpha} \tag{8}$$

Thus:

$$B = \frac{-100 \cdot \cos(56.5°) - \frac{100\beta \cdot \sin(56.5°)}{\alpha}}{\frac{\beta^2}{\alpha} + \alpha} \approx -91604.77 \tag{9}$$

Then

$$A = \frac{100 \cdot \sin(56.5°) + \beta B}{\alpha} \approx -59868.41 \tag{10}$$

Observe that to achieve a negative value for A and B we have to reverse the direction of each bearing from 2) and 3).

Since we may only take integer km steps, we thus complete 59868 repeats of reverse of the path from 2), and 91605 repeats of the reversed path from 3).

Using these rounded numbers we get that the East-West displacement is:

$$\alpha \cdot -59868 + \beta \cdot 91605 \approx 83.38879 \text{ km} \quad \text{(nearest cm)}$$

And the North-South displacement is:

$$\beta \cdot -59868 + \alpha \cdot -91605 \approx -55.19333 \text{ km} \quad \text{(nearest cm)}$$

I.e 55.19333 km South.

We compare these to the true displacements of the gold:

East

$$100 \cdot \sin(56.5^\circ) = 83.38868 \text{ km} \quad \text{(nearest cm)}$$

South

$$100 \cdot \cos(56.5^\circ) = 55.19370 \text{ km} \quad \text{(nearest cm)}$$

Thus following this path, we end up roughly 11 cm East, and roughly 37 cm North of the gold's true location.

Applying Pythagoreans, this route ends up $\sqrt{37^2 + 11^2} \approx 39$ cm from the gold, and thus is close enough to find it.

Since each of the reversed routes involves travelling $2 + 2 + 1 + 1 = 6$ km, and we perform $59868 + 91605 = 151473$ total repeats, the total distance required is $151473 \times 6 = 908838$ km, over twice the distance to the moon!

5. The simultaneous equations derived in the previous question can be applied to any (x, y) coordinates, as we have two equations in two variables (exercise - what is the additional requirement to guarantee a solution?)

   Thus we can always find real numbers $\hat{A}, \hat{B} \in \mathbb{R}$ that solve them:

$$\alpha\hat{A} - \beta\hat{B} = x \tag{11}$$

$$\beta\hat{A} + \alpha\hat{B} = y \tag{12}$$

   When rounded to the nearest integer, this adds an error of at most 0.5 of an iteration, for the route from 2), and the route from 3).

   Assuming a worst case scenario, this adds up to:

   - $0.5136 \div 2 = 0.2568$ cm East-West (4.d.p)
   - $91.3783 \div 2 = 45.6892$ cm North-South (4.d.p)
   - another 45.6892 cm East-West
   - another 0.2568 cm North-South

   Which is less than 50cm of error in each dimension, we see that:

$$\sqrt{50^2 + 50^2} = 70.7 \cdots < 100$$

   And thus our final destination, with each step rounded to the nearest km remains within 1m = 100cm of the gold, and thus is sufficient to find it.

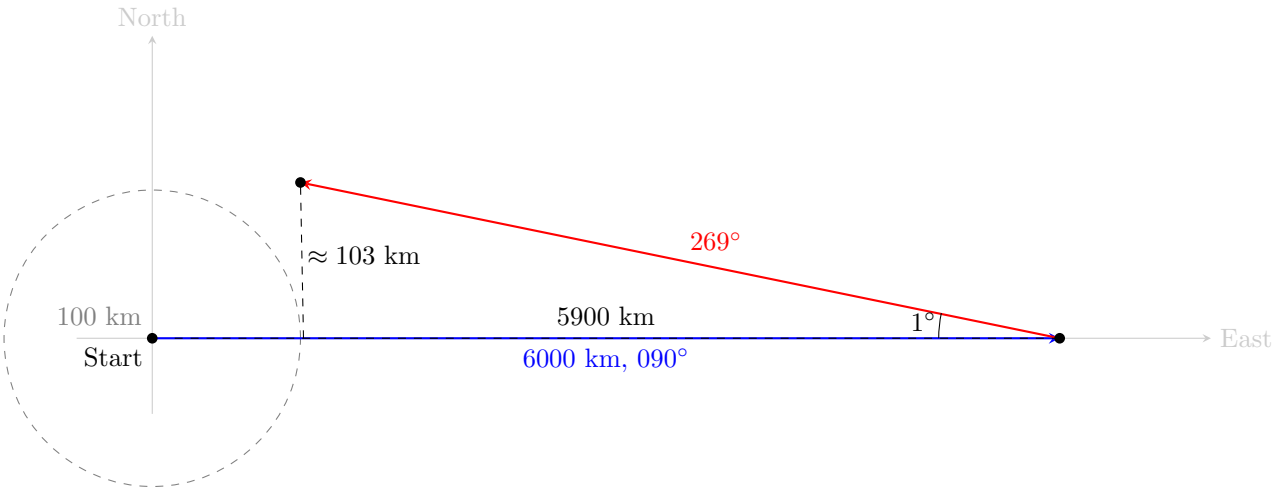
3 Challenge II: Two Legs Only

1. Suppose we head out on a bearing of 090° for 6000 km for the first leg.

   Then consider returning on a bearing of 269°, i.e. with a 1° angle away from our original line.

   Consider a right angled triangle with base 5900 and angle 1°, by right angled trigonometry, this has height:

$$5900 \cdot \sin(1°) \approx 102.97 \text{ km}$$



   Which means that by the time we have returned to within 100km East of the starting point, we are already over 100km North, and will never be able to return to a position inside a circle radius 100km of the origin,

   A similar argument holds for a return bearing of 268°, 267° etc.

   And thus we have only one choice of return bearing: 270°, and then our return positions all lie due East of the origin

   By symmetry this argument holds for all initial bearings $ABC°$, with only $XYZ°$ the exact reverse bearing possible if we want to return within 100 km of the start location

2. Let θ be the non-reflex angle between the first and second leg of the journey, then the final distance from the start D is given by the cosine rule:

$$D^2 = M^2 + N^2 - 2MN\cos(\theta)$$

   We require that $D^2 < 100^2$ to remain within 100km of the start.

   See that $|M - N|^2 = (M - N)^2 = M^2 - 2MN + N^2$

   Thus:

$$D^2 = |M - N|^2 + 2MN - 2MN\cos(\theta) = |M - N|^2 + 2MN(1 - \cos(\theta))$$

   Since $\cos(\theta) \leq 1 \quad \forall\theta$, we have that, $2MN(1 - \cos(\theta))$ is always positive (or zero). Thus:

$$D^2 \geq |M - N|^2$$

   Thus to ensure $D^2 < 100^2$ it must be the case that $|M - N|^2 < 100^2$, i.e.

$$|M - N| \leq 100$$

   So N must be within the range:

$$[M - 100, M + 100]$$

   as a necessary condition to return within 100 km of the start. Not all such N and $XYZ°$ will do so, but those that do will definitely meet this criterion.

3. From now on we omit journeys such as discussed in 1) since they can be replicated without a return leg, e.g. 6000 km East and then 5950km West ends up in the same location as travelling 50km East. All journeys with $M > 6000$ are not counted, as they replicate existing possibilities.

   Thus we have:

   - At most 360 options for $ABC°$
   - At most 6000 options for M
   - At most 360 options for $XYZ°$
   - At most 203 options for N (by the previous question, +2 so the radius is 101km which is a necessary condition to be within 1m of the gold.)

   Not all of these combinations will end up within 101km of the start, but all those unique journeys that do are included here.

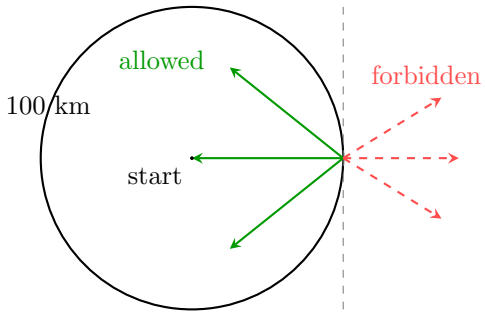
   Thus there are at most $360 \times 6000 \times 360 \times 203 \approx 1.58 \times 10^{11}$ unique final destinations within 101km of the start.

4. Not necessarily, but possibly, since we would have to check each of the now finite possibilities and whether they end up within 1 meter of the gold.

5. (a) Consider the first row of the table, $M \leq 100$, in this case we opt to allow all possibilities:

	$ABC°$ Option count	M in range	$XYZ°$ Option count per M value (upper bound)	N Option count per M value (upper bound)	Total journey option count for this M range
$M \leq 100$	360	101	360	203	2.66×10^9
$100 < M \leq 500$					
$500 < M \leq 2000$					
$2000 < M \leq 6000$					

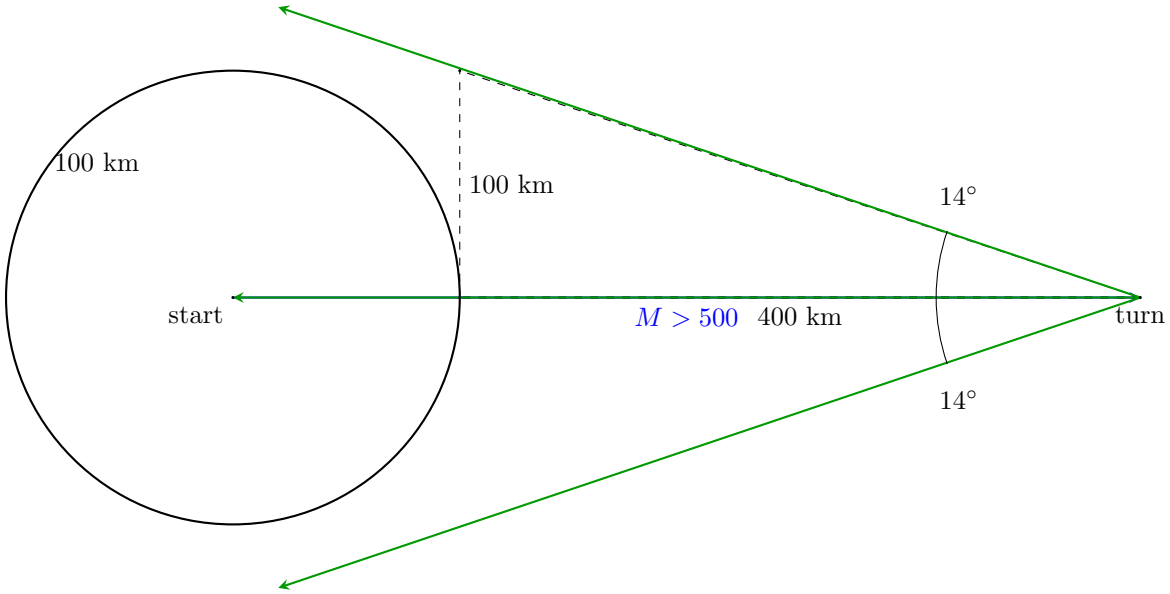
   (b) when $M > 100$, we have left the circle in our first leg, so only have at most 180 possible return bearings, since otherwise we would end up further away



   So we set the $XYZ°$ option count to 180:

	$ABC°$ Option count	M in range	$XYZ°$ Option count per M value (upper bound)	N Option count per M value (upper bound)	Total journey option count for this M range
$M \leq 100$	360	101	360	203	2.66×10^9
$100 < M \leq 500$	360	400	180	203	5.26×10^9
$500 < M \leq 2000$					
$2000 < M \leq 6000$					

   (c) For $M > 500$, we consider a right angled triangle with base 400 and height 100, which then must have an angle of $\tan^{-1}(100/400) = 14.03°$ as the return angle.



And thus we have at most $14 + 14 + 1 = 29$ possible choices for our return bearing $XYZ°$:

	$ABC°$ Option count	M in range	$XYZ°$ Option count per M value (upper bound)	N Option count per M value (upper bound)	Total journey option count for this M range
$M \leq 100$	360	101	360	203	2.66×10^9
$100 < M \leq 500$	360	400	180	203	5.26×10^9
$500 < M \leq 2000$	360	1500	29	203	3.18×10^9
$2000 < M \leq 6000$					

(d) The argument for the case $M > 2000$ is similar to the $M > 500$, and we evaluate:

$$\tan^{-1}(100/1900) \approx 3.01$$

And thus we have at most $3 + 3 + 1 = 7$ options for the return bearing, and our complete table becomes:

	$ABC°$ Option count	M in range	$XYZ°$ Option count per M value (upper bound)	N Option count per M value (upper bound)	Total journey option count for this M range
$M \leq 100$	360	101	360	203	2.66×10^9
$100 < M \leq 500$	360	400	180	203	5.26×10^9
$500 < M \leq 2000$	360	1500	29	203	3.18×10^9
$2000 < M \leq 6000$	360	4000	7	203	2.05×10^9

By summing the last column we see that there are at most:

$$2.66 \times 10^9 + 5.26 \times 10^9 + 3.18 \times 10^9 + 2.05 \times 10^9 = 1.32 \times 10^{10}$$

Possible endpoints for a two leg journey that finish within 101km of the start.

6. The journey described is the same as the generic journey:

- Travel Mkm on a bearing of $ABC°$
- Travel Nkm on a bearing of $XYZ°$

Except with the order of the legs swapped. We see that this ends up in the same end point.

Observe that if a journey M, ABC°, N, XYZ° does finish within 101km of the start, then so will N, XYZ°, M, ABC°, and thus we can halve the upper bound from the previous question.

(There is some subtlety to this argument, since we first discard the journeys that end outside the radius to consider a true count with duplicates: U', with $U' \leq U = 1.32 \times 10^{10}$, then see that omitting the duplicates yields $\frac{U'}{2} \leq \frac{U}{2}$)

Thus we establish an upper bound on the number of two leg journeys that finish within 101km of the starting point

$$U = 6.6 \times 10^9 < 10^{10}$$

7. 100km is 10^5m and thus the area is $10^{10}\pi$ square meters.

   Consider the destination of each of the 6.6×10^9 potential journeys, if they do finish inside the 100km radius circle, they make accessible a circle of points radius 1m from the endpoint, that is they cover an area of π square meters.

   So all the possibilities cover at most $6.6 \times 10^9\pi$ square meters, but this still leaves over $3 \times 10^9\pi$ square meters, all the points inside of which are not accessible.

   Thus some locations exist that cannot be reached within 1m, following a two leg journey as described.

8. Naive attempt:

```python
import math
from tqdm import tqdm

# ------------------------------
# PARAMETERS
# ------------------------------

# Gold location (km) | CHANGE THIS
GOLD_X = 83.38868
GOLD_Y = -55.19370

# Tolerance: 100 m = 0.1 km
TOL = 0.1

# Search limits
MAX_M = 6000
MAX_N_OFFSET = 101

# ------------------------------
# HELPER FUNCTIONS
# ------------------------------

def step(x, y, distance, bearing_deg):
    """
    Move distance (km) from (x, y) on a bearing in degrees.
    Bearings measured clockwise from North.
    """
    theta = math.radians(bearing_deg)
    dx = distance * math.sin(theta)
    dy = distance * math.cos(theta)
    return x + dx, y + dy


def in_gold_box(x, y):
    return (
        abs(x - GOLD_X) <= TOL and
        abs(y - GOLD_Y) <= TOL
    )

# ------------------------------
# BRUTE FORCE SEARCH
# ------------------------------

hits = []

for ABC in tqdm(range(360)):
    for M in tqdm(range(1, MAX_M + 1)):

        # First leg
        x1, y1 = step(0.0, 0.0, M, ABC)

        for XYZ in range(360):

            # Restrict N range
            for N in range(max(1, M - MAX_N_OFFSET), M + MAX_N_OFFSET + 1):

                # Second leg
                x2, y2 = step(x1, y1, N, XYZ)
```

```
                # Check if inside 100m box around gold
                if in_gold_box(x2, y2):
                    hits.append({
                        "ABC": ABC,
                        "M": M,
                        "XYZ": XYZ,
                        "N": N,
                        "x": x2,
                        "y": y2
                    })

# ------------------------------
# OUTPUT
# ------------------------------

print(f"Total solutions found: {len(hits)}")

for h in hits:
    print(
        f"ABC={h['ABC']:3d}, M={h['M']:4d}, "
        f"XYZ={h['XYZ']:3d}, N={h['N']:4d}, "
        f"x={h['x']:.4f}, y={h['y']:.4f}"
    )
```

Unfortunately this would take over 42 hours to run, due to the slowness of python, to circumvent this, use the Numpy library, with vectorisation, the following code takes roughly 40 minutes to complete.

```python
import numpy as np
from tqdm import tqdm

# ------------------------------
# PARAMETERS
# ------------------------------

# Gold location (km) | CHANGE THIS
GOLD_X = 83.38868
GOLD_Y = -55.19370

# Tolerance: 15 m = 0.015 km
TOL = 0.015

# Search limits
MAX_M = 6000
MAX_N_OFFSET = 101

# ------------------------------
# MORE MEMORY-EFFICIENT VERSION (still vectorized over M)
# ------------------------------

def find_solutions_vectorized_M():
    """
    Vectorized over M with chunked XYZ processing to manage memory.
    """
    all_abc = np.arange(360)
    all_xyz = np.arange(360)

    # Precompute trig
    sin_abc = np.sin(np.deg2rad(all_abc))
    cos_abc = np.cos(np.deg2rad(all_abc))
    sin_xyz = np.sin(np.deg2rad(all_xyz))
    cos_xyz = np.cos(np.deg2rad(all_xyz))

    all_M = np.arange(MAX_M + 1)
    max_N_len = 2 * MAX_N_OFFSET + 1

    hits = []

    # Process each ABC
    for abc_idx in tqdm(range(360), desc="Processing ABC"):

        abc = all_abc[abc_idx]
        sin_abc_val = sin_abc[abc_idx]
        cos_abc_val = cos_abc[abc_idx]

        # x1 and y1 for all M
        x1 = all_M * sin_abc_val
        y1 = all_M * cos_abc_val

        # N ranges for all M
        N_min = np.maximum(0, all_M - MAX_N_OFFSET)
        N_max = np.minimum(all_M + MAX_N_OFFSET, MAX_M + MAX_N_OFFSET)

        # Create array of N values for each M
        N_arrays = []
        valid_lengths = []
```

```
        for i, M in enumerate(all_M):
            if N_min[i] <= N_max[i]:
                N_range = np.arange(N_min[i], N_max[i] + 1)
                N_arrays.append(N_range)
                valid_lengths.append(len(N_range))
            else:
                N_arrays.append(np.array([], dtype=int))
                valid_lengths.append(0)

        # Process XYZ in chunks to manage memory
        xyz_chunk_size = 90 # Process 180 XYZ at a time
        for xyz_start in range(0, 360, xyz_chunk_size):
            xyz_end = min(360, xyz_start + xyz_chunk_size)
            xyz_chunk = all_xyz[xyz_start:xyz_end]

            sin_xyz_chunk = sin_xyz[xyz_start:xyz_end]
            cos_xyz_chunk = cos_xyz[xyz_start:xyz_end]

            # Process each M and its N range
            for m_idx, M in enumerate(all_M):
                if valid_lengths[m_idx] == 0:
                    continue

                N_range = N_arrays[m_idx]
                L = len(N_range)

                # Vectorized over N and XYZ chunk
                # x2 shape: (L, xyz_chunk_size)
                x2 = x1[m_idx] + np.outer(N_range, sin_xyz_chunk)
                y2 = y1[m_idx] + np.outer(N_range, cos_xyz_chunk)

                # Check tolerance
                x_diff = np.abs(x2 - GOLD_X)
                y_diff = np.abs(y2 - GOLD_Y)
                mask = (x_diff <= TOL) & (y_diff <= TOL)

                # Get indices
                n_indices, xyz_chunk_indices = np.where(mask)

                for i in range(len(n_indices)):
                    N = N_range[n_indices[i]]
                    xyz = xyz_chunk[xyz_chunk_indices[i]]

                    hits.append({
                        "ABC": abc,
                        "M": M,
                        "XYZ": xyz,
                        "N": N,
                        "x": x2[n_indices[i], xyz_chunk_indices[i]],
                        "y": y2[n_indices[i], xyz_chunk_indices[i]]
                    })

    return hits

# ----------------------------
# MAIN EXECUTION
# ----------------------------
```

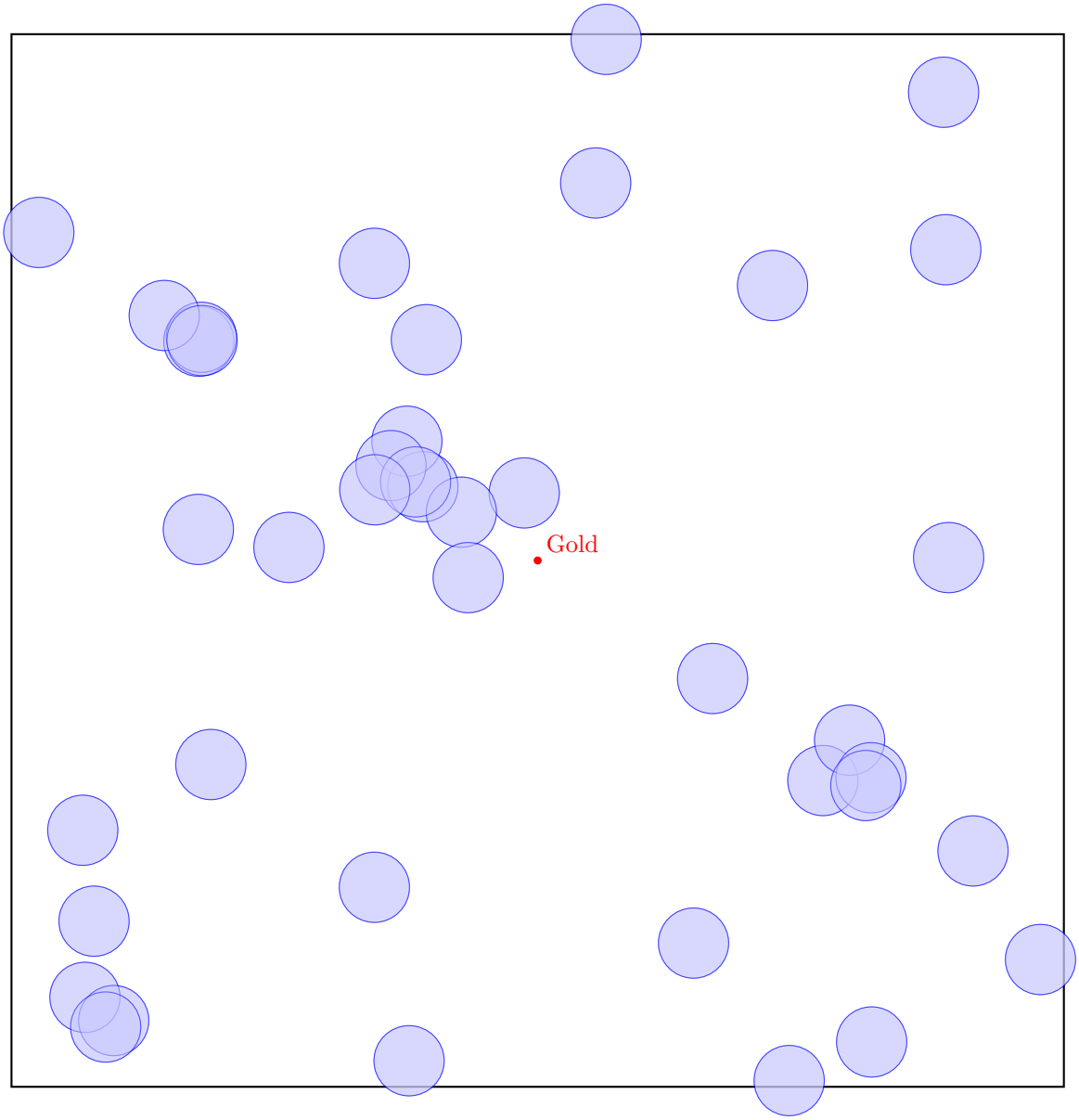
```
if __name__ == "__main__":


    print("Running vectorized version (M dimension fully vectorized)...")
    hits = find_solutions_vectorized_M()

    # Output
    print(f"Total solutions found: {len(hits)}")

    for h in hits:  # Print solutions
        print(
            f"ABC={h['ABC']:3d}, M={h['M']:4d}, "
            f"XYZ={h['XYZ']:3d}, N={h['N']:4d}, "
            f"x={h['x']:.9f}, y={h['y']:.9f}"
        )
```

Running this code finds all the points within 15m of the gold:

```
Total solutions found: 82
ABC=  2, M=4832, XYZ=181, N=4885, x=83.379362620, y=-55.199514683
ABC=  2, M=4833, XYZ=181, N=4886, x=83.396809711, y=-55.199971551
ABC=  4, M=4937, XYZ=183, N=4987, x=83.388297091, y=-55.191773688
ABC= 18, M= 469, XYZ=187, N= 505, x=83.384951942, y=-55.190300436
ABC= 25, M= 230, XYZ=183, N= 264, x=83.385507752, y=-55.187406157
ABC= 34, M=5729, XYZ=213, N=5729, x=83.379112383, y=-55.187430591
ABC= 34, M=5730, XYZ=213, N=5730, x=83.393666251, y=-55.197063587
ABC= 35, M=1910, XYZ=212, N=1910, x=83.385198745, y=-55.191459067
ABC= 37, M= 819, XYZ=210, N= 819, x=83.386503962, y=-55.192322971
ABC= 40, M=1147, XYZ=215, N=1140, x=83.400250870, y=-55.180354232
ABC= 41, M= 383, XYZ=206, N= 383, x=83.374458883, y=-55.184350507
ABC= 46, M= 231, XYZ=201, N= 231, x=83.384497533, y=-55.190994945
ABC= 60, M= 505, XYZ=229, N= 469, x=83.384035787, y=-55.191684597
ABC= 61, M= 106, XYZ=185, N= 107, x=83.384024483, y=-55.203012950
ABC= 62, M= 107, XYZ=186, N= 106, x=83.395375330, y=-55.185863691
ABC= 64, M= 264, XYZ=222, N= 230, x=83.381588760, y=-55.193327107
ABC= 64, M=4987, XYZ=243, N=4937, x=83.386698976, y=-55.194188177
ABC= 66, M=4885, XYZ=245, N=4832, x=83.390333623, y=-55.182939306
ABC= 66, M=4886, XYZ=245, N=4833, x=83.397571294, y=-55.198820924
ABC= 73, M= 178, XYZ=219, N= 138, x=83.376032597, y=-55.203979240
ABC= 74, M=  77, XYZ=173, N=  77, x=83.401090029, y=-55.201977278
ABC= 76, M=  74, XYZ=171, N=  74, x=83.378034157, y=-55.186716930
ABC= 90, M= 533, XYZ=263, N= 453, x=83.376593306, y=-55.206812563
ABC= 94, M=  24, XYZ=132, N=  80, x=83.393123244, y=-55.204603879
ABC= 98, M= 500, XYZ=272, N= 412, x=83.385013639, y=-55.207957839
ABC=100, M=1234, XYZ=278, N=1143, x=83.376364645, y=-55.206996844
ABC=105, M= 700, XYZ=282, N= 606, x=83.390632358, y=-55.178846936
ABC=108, M= 110, XYZ=225, N=  30, x=83.403013357, y=-55.205072817
ABC=113, M= 620, XYZ=291, N= 522, x=83.384026509, y=-55.185230001
ABC=115, M=  80, XYZ=153, N=  24, x=83.400394957, y=-55.193617520
ABC=122, M=  25, XYZ=124, N=  75, x=83.379020346, y=-55.187449366
ABC=122, M= 103, XYZ=261, N=   4, x=83.398200542, y=-55.207422076
ABC=123, M=  49, XYZ=124, N=  51, x=83.375774030, y=-55.206150793
ABC=123, M=  50, XYZ=124, N=  50, x=83.385407025, y=-55.191596924
ABC=123, M=  75, XYZ=125, N=  25, x=83.379093703, y=-55.187338535
ABC=123, M= 150, XYZ=302, N=  50, x=83.398180384, y=-55.199892041
ABC=124, M=  50, XYZ=123, N=  50, x=83.385407025, y=-55.191596924
ABC=124, M=  51, XYZ=123, N=  49, x=83.375774030, y=-55.206150793
ABC=124, M=  75, XYZ=122, N=  25, x=83.379020346, y=-55.187449366
ABC=124, M= 150, XYZ=305, N=  50, x=83.398033669, y=-55.200113703
```

```
ABC=125, M=  25, XYZ=123, N=  75, x=83.379093703, y=-55.187338535
ABC=132, M=  80, XYZ= 94, N=  24, x=83.393123244, y=-55.204603879
ABC=134, M= 620, XYZ=316, N= 522, x=83.379006830, y=-55.192813908
ABC=142, M= 700, XYZ=325, N= 606, x=83.375712299, y=-55.201388686
ABC=149, M= 500, XYZ=335, N= 412, x=83.400313618, y=-55.184842092
ABC=153, M=  24, XYZ=115, N=  80, x=83.400394957, y=-55.193617520
ABC=171, M=  74, XYZ= 76, N=  74, x=83.378034157, y=-55.186716930
ABC=171, M=1122, XYZ=355, N=1057, x=83.395849691, y=-55.208522265
ABC=173, M=  77, XYZ= 74, N=  77, x=83.401090029, y=-55.201977278
ABC=181, M=4885, XYZ=  2, N=4832, x=83.379362620, y=-55.199514683
ABC=181, M=4886, XYZ=  2, N=4833, x=83.396809711, y=-55.199971551
ABC=183, M= 264, XYZ= 25, N= 230, x=83.385507752, y=-55.187406157
ABC=183, M=4987, XYZ=  4, N=4937, x=83.388297091, y=-55.191773688
ABC=185, M= 107, XYZ= 61, N= 106, x=83.384024483, y=-55.203012950
ABC=186, M= 106, XYZ= 62, N= 107, x=83.395375330, y=-55.185863691
ABC=187, M= 505, XYZ= 18, N= 469, x=83.384951942, y=-55.190300436
ABC=201, M= 231, XYZ= 46, N= 231, x=83.384497533, y=-55.190994945
ABC=206, M= 383, XYZ= 41, N= 383, x=83.374458883, y=-55.184350507
ABC=210, M= 819, XYZ= 37, N= 819, x=83.386503962, y=-55.192322971
ABC=212, M=1910, XYZ= 35, N=1910, x=83.385198745, y=-55.191459067
ABC=213, M=5729, XYZ= 34, N=5729, x=83.379112383, y=-55.187430591
ABC=213, M=5730, XYZ= 34, N=5730, x=83.393666251, y=-55.197063587
ABC=215, M=1140, XYZ= 40, N=1147, x=83.400250870, y=-55.180354232
ABC=219, M= 138, XYZ= 73, N= 178, x=83.376032597, y=-55.203979240
ABC=222, M= 230, XYZ= 64, N= 264, x=83.381588760, y=-55.193327107
ABC=225, M=  30, XYZ=108, N= 110, x=83.403013357, y=-55.205072817
ABC=229, M= 469, XYZ= 60, N= 505, x=83.384035787, y=-55.191684597
ABC=243, M=4937, XYZ= 64, N=4987, x=83.386698976, y=-55.194188177
ABC=245, M=4832, XYZ= 66, N=4885, x=83.390333623, y=-55.182939306
ABC=245, M=4833, XYZ= 66, N=4886, x=83.397571294, y=-55.198820924
ABC=261, M=   4, XYZ=122, N= 103, x=83.398200542, y=-55.207422076
ABC=263, M= 453, XYZ= 90, N= 533, x=83.376593306, y=-55.206812563
ABC=272, M= 412, XYZ= 98, N= 500, x=83.385013639, y=-55.207957839
ABC=278, M=1143, XYZ=100, N=1234, x=83.376364645, y=-55.206996844
ABC=282, M= 606, XYZ=105, N= 700, x=83.390632358, y=-55.178846936
ABC=291, M= 522, XYZ=113, N= 620, x=83.384026509, y=-55.185230001
ABC=302, M=  50, XYZ=123, N= 150, x=83.398180384, y=-55.199892041
ABC=305, M=  50, XYZ=124, N= 150, x=83.398033669, y=-55.200113703
ABC=316, M= 522, XYZ=134, N= 620, x=83.379006830, y=-55.192813908
ABC=325, M= 606, XYZ=142, N= 700, x=83.375712299, y=-55.201388686
ABC=335, M= 412, XYZ=149, N= 500, x=83.400313618, y=-55.184842092
ABC=355, M=1057, XYZ=171, N=1122, x=83.395849691, y=-55.208522265
```

We then can plot them to see if the gold will be found:



And thus see that the gold cannot be found via a two step journey

9. One journeys end up within 2m of the gold

   - 4937km on a bearing of 004°, then 4987 on a bearing of 183°

   Which finishes in the following coordinates:

$$(4937 \cdot \sin(4°) - 4987 \cdot \sin(3°), 4937 \cdot \cos(4°) - 4987 \cdot \cos(3°)) = (83.38830, -55.19177)$$