Balanced centrifuge problem

MARICHAL Jean-Philippe,
RODRIGUES COSTA André Manuel

Semester 3

1 Introduction

A centrifuge is a laboratory equipment used to separate fluids that have different densities by spinning at high speed. Such a centrifuge has separate slots, equally spaced around the center of the rotor, in which it can contain test tubes. It is very important to place the test tubes in a balanced way, otherwise the machine can be permanently damaged. They should be placed such that the center of gravity of the tubes coincides with the center of gravity of the centrifuge itself. We denote n as the number of available slots and k as the number of test tubes we want to insert in the different slots. The video made by Numberphile on the centrifuge problem gives you a good first understanding of the problem: `https://www.youtube.com/watch?v=7DHE8RnsCQ8`
Understanding when a centrifuge is balanced comes down to knowing for which k may we pick k distinct nth roots of unity whose sum is 0. It has been proven by Gary Sivek in [1], that we can find k distinct nth roots of unity whose sum is 0 if and only if both k and $n-k$ are expressible as linear combinations of prime factors of n with non negative coefficients. When this is the case, the centrifuge can be balanced. We get a set of solutions $S_n \subset \{2, \cdots, n\}$, that contains the different k for which it is possible to balance the centrifuge.
We cannot allow repetition of roots of unity because once a slot in the centrifuge is filled, it is not possible to insert a second tube in that specific slot. So one can find solutions that do not look like a obvious configurations, for example if $n = 24$ and $k = 11$ the balanced centrifuge has this odd configuration, which may not seem balanced.

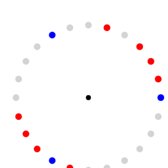


Figure 1: 11 tubes

Solutions like this one are due to the overlap problem. When we are adding configurations into the same centrifuge, we need to make sure that no tubes overlap. Whenever we encounter a slot which is already occupied, we rotate until we can insert the configuration.
In figure 1 we can see that 11 is a solution since $11 = 3 + 4 \times 2$. To 'compute' the solution, first we insert the 3 balanced blue tubes that form an equilateral triangle. Next we insert 4 times the 2 red tubes, which are placed on opposite sides, by rotating to the next slot each time. Until the last 2 tubes are inserted, we encounter no overlap but for the last 2 tubes we see that the slot is free but the opposite slot is occupied, so we have to rotate one time.

In section 3, we will introduce a sequence representing the cardinality of the set of solutions S_n for $n \in \mathbb{N}$. This sequence is the main result and has a list of interesting properties which we will discuss in Theorem 3.7.

2 Sivek Theorem

Theorem 2.1 (Sivek Theorem)**.** *Write $n = p_1^{e_1} \cdots p_r^{e_r}$, with p_i prime and each e_i positive and let $1 \leqslant k < n - 1$. Then n is k-balancing if and only if both k and $n - k$ are in $\mathbb{N}p_1 + \cdots + \mathbb{N}p_r$*

Theorem 2.1 gives us a list of different k, that are solutions for a specific n. This can be implemented on a short python program. It can be written like such:

```python
from sympy.ntheory import primefactors

#Main Theorem
def centrifList(n):
    '''We obtain all the k possible solutions
    for a centrifuge n in a list'''
    sol1 = primefactors(n)
    sol2 = []
    final = []
    while len(sol1) != len(sol2):
        sol2 = sol1
        for i in sol2:
            for j in sol2:
                if i + j < n and i + j not in sol1:
                    sol1.append(i + j)
                    sol1.sort()
        for l in sol1:
            if n - l in sol1:
                final.append(l)
        final += [0,n]
        final.sort()
        return final
```

Example 2.2. If we search all the possible k that work for a specific $n = 14$ we obtain: $[0, 2, 4, 6, 7, 8, 10, 12, 14]$ as a list.

Remark. *Theorem 2.1 helps to understand where solutions come from and why they are a solution. Depending on the relation between n and k, we can determine whether the couple (n,k) forms a solution for a fixed n. For smaller values of n, there are 3 cases that can appear. (1) and (2) have been proved by Gary Sivek in [1] and (3) is a result dating back to [3].*

(1) If gcd(n,k) > 1, then n is k-balancing.

(2) If $n = pq$ with p, q two primes, then $k =$ is either a multiple of p or a multiple of q

(3) If gcd(p, q) $= 1$ and $k \geqslant (p-1)(q-1)$ then $k \in \mathbb{N}p + \mathbb{N}q$

These cases are implemented in the python program.

Example 2.3. We can illustrate how the theorem works with an example.

1. Let $n = 24$, the example from before. Here are two pictures of balanced centrifuges.

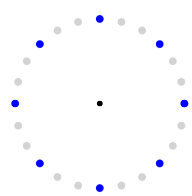


Figure 2
8 tubes

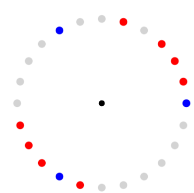


Figure 3
11 tubes

For $k = 8$, just by looking at the figure 2 the centrifuge looks clearly balanced. We have:

$$k = 8 = 4 \times 2 = 2 + 2 + 2 + 2$$

$$n - k = 16 = 2 \times 5 + 2 \times 3 = 5 + 5 + 3 + 3$$

So $k, n-k \in \mathbb{N}p_1 + \cdots + \mathbb{N}p_r$.
For $k = 11$ one might think that the centrifuge is not balanced. In fact, we have:

$$k = 11 = 3 + 4 \times 2 = 3 + 2 + 2 + 2 + 2$$

$$n - k = 13 = 2 \times 5 + 3 = 5 + 5 + 3$$

Again $k, n-k \in \mathbb{N}p_1 + \cdots + \mathbb{N}p_r$ and by Theorem 2.1 both 8 and 11 are balanced configurations.

2. Now let $n = 45$ and $k = 19$.

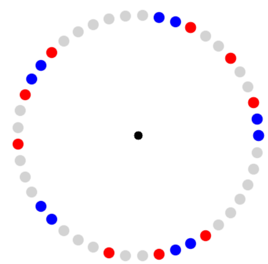


Figure 4: 19 tubes

At first sight the centrifuge may seem to be unbalanced, but in fact:

$$k = 19 = 2 \times 5 + 3 \times 3 = 5 + 5 + 3 + 3 + 3$$

We managed to express 19 as a linear combination of prime factors of n which means that the centrifuge is balanced.

We can then write a code that draws a specific solution for a given n and k as such:

```
import matplotlib.pyplot as plt
import numpy as np
import math
import itertools
from sympy.ntheory import primefactors
import sympy

def valueB(k, p1, p2):
    '''Finds the value of b for k = ap + bq'''
    count = 0
    while k % p1 != 0 :
        k = k - p2
        count += 1
    return count

def valueA(k, p1, p2, b):
    '''Finds the value of a for k = ap + bq'''
    return int((k - p2 * b) / p1)

def centrifListCoef(n):
    '''Gives a list of tuples (k, coef) with k beeing a solution for the
    Theorem 2.1 and coef a list of primes coefficient for k'''
    prim = primefactors(n)
    sol1 = []
    sol2 = []
    final = []
```

```
    lim2PSum = (prim[0] - 1)*(prim[1] - 1)
    for i in range(0, len(prim)):
        coef = [0]*len(prim)
        coef[i] += 1
        sol1 += [(prim[i], coef)]
    while len(sol1) != len(sol2):
        sol2 = list(sol1)
        for i in sol2:
            for j in range(0, len(prim)):
                ksol = []
                for g in sol1:
                    ksol.append(g[0])
                sumij = int(i[0] + prim[j])
                if sumij < n and sumij not in ksol:
                    if sumij < lim2PSum:
                        coef = list(i[1])
                        coef[j] += 1
                    else:
                        b = valueB(sumij, prim[0], prim[1])
                        a = valueA(sumij, prim[0], prim[1], b)
                        coef = [a, b]+[0]*(len(prim)-2)
                    sol1.append((sumij, coef))
                    sol1.sort()
    for l in sol1:
        if n - int(l[0]) in ksol:
            final.append(l)
    final += [(0,[0]*len(prim)),(n,[int(n/prim[0])]+[0]*(len(prim)
    -1))]
    final.sort()
    return final

def draw(m, color, theta, ax):
    '''Draws roots of unity m starting at angle theta'''
    theta = theta + np.linspace(0, 2 * np.pi, m + 1)
    r = 2
    a = r * np.cos(theta)
    b = r * np.sin(theta)
    ax.scatter(a, b, s=100, c=color)
    ax.scatter(0, 0, s=50, c='black')

def moduloList(num, add, ran):
    '''Gives a list of numbers that represents the space the
    solution occupies'''
    modulo = []
    for i in range(0, ran - 1):
        if ((i - add) % num) == 0:
            modulo.append(i)
    return modulo

def compModulo(li1, li2):
    '''Checks if the two solutions doesn't overlap'''
    count = 0
    for i in li1:
        for j in li2:
            if i == j:
                count += 1
    return (count != 0)
```

```python

def drawP(n, prim, coef, angle, color, ax):
    '''Draws roots of unity for given primes and given amount'''
    spotsTaken = []
    coef.reverse()
    prim.reverse()
    for i in range(0, len(coef)):
        jump = 0
        pickColor = i % 4
        for j in range(0, coef[i]):
            spotsWants = moduloList(int(n/prim[i]), j + jump, n)
            while compModulo(spotsTaken, spotsWants):
                jump += 1
                spotsWants = moduloList(int(n/prim[i]), j + jump, n)
            spotsTaken = spotsTaken + spotsWants
            draw(prim[i], color[pickColor], (j + jump) * angle, ax)

def findPrime2(li):
    '''finds the second smallest prime dividing n'''
    li.pop(1)
    li.pop(0)
    for i in li:
        if sympy.isprime(i):
            p = i
            break
    return p

def drawSolution(n, k, ax):
    '''Draws the solution for a balanced centrifuge if it exsists'''
    gcd = math.gcd(n, k)
    angle = (2 * np.pi) / n
    color = ['b', 'r', 'g', 'y']
    if gcd > 1:
        steps = int(k / gcd)
        for i in range(0, steps):
            pickColor = i % 4
            draw(gcd, color[pickColor], i * angle, ax)
    else:
        solC = centrifListCoef(n)
        solK = []
        for g in solC:
            solK.append(g[0])
        prim = primefactors(n)
        if k in solK:
            if k > n / 2:
                k = n - k
                draw(n, 'red', 0)
                color = ['lightgray','lightgray', 'lightgray','lightgray']
            else:
                color = color
            p1 = prim[0]
            p2 = prim[1]
            pos = [x for x, y in enumerate(solC) if y[0] == k]
            pos = pos[0]
```

```python
            coef = solC[pos][1]
            drawP(n, prim, coef, angle, color, ax)
        else:
            print('There is no solution')

def menu():
    n = int(input("Enter the number of buckets in the centrifuge:")
    )
    k = int(input("Number of tubes:"))
    fig, ax = plt.subplots()
    draw(n, 'lightgray', 0, ax)
    drawSolution(n, k, ax)
    ax.set_aspect('equal', 'box')
    plt.axis('off')
    plt.show()

menu()
```

3 Sequence of solutions

For every n, we have a set of solutions S_n with cardinality $< n$. Then the $\#S_n$ is the number of distinct k that we can choose to balance a n-centrifuge. Note that 0 and n are also a solution although we will not consider them.

Definition 3.1. *Let $(s_n)_{n\in\mathbb{N}}$ be a sequence with values in the natural numbers, $n \geqslant 1$. $S_n := \{0 < k < n : n \text{ is } k\text{-balancing}\}$ is the set of all balanced solutions. The sequence is defined by:*

$$s_n = \#S_n$$

By definition $s_1 = 0$

Example 3.2. The first 24 elements in the sequence are:

$$0, 0, 0, 1, 0, 3, 0, 3, 2, 5, 0, 9, 0, 7, 6, 7, 0, 15, 0, 15, 8, 11, 0, 21, ...$$

Lemma 3.3. *Let $n \geqslant 5$ such that n is not prime. Then $s_n \geqslant 2$*

Proof. Let $n \geqslant 5$ and p be a prime such that $p|n$. In the case where n is even choose $p = 2$. We know $n-p \neq p$ since $n \geqslant 5$ so $p, n-p \in S_n \Rightarrow s_n \geqslant 2$. Suppose n is not even, if $n-p = p \Rightarrow n = 2p$ but n is not even so $n-p \neq p \Rightarrow s_n \geqslant 2$ □

Lemma 3.4. *Let $x = ab$ if $a \leqslant b \Rightarrow b \geqslant \sqrt{x}$ for $a, b \in \mathbb{R}_{\geqslant 0}$.*

Proof. Suppose $b \geqslant a \Leftrightarrow \sqrt{b} \geqslant \sqrt{a} \Leftrightarrow b \geqslant \sqrt{a}\sqrt{b} = \sqrt{x}$ □

Proposition 3.5. *Let $(s_n)_{n\in\mathbb{N}}$ be defined as above.*

(a) $s_n = 0 \iff n$ *is a prime number*

(b) $s_n = 1 \iff n = 4$

Proof. (a) Let n be a prime number. If n is k-balancing then $k \in n\mathbb{N} \Rightarrow k \geqslant n$. So $S_n = \varnothing$ thus $s_n = 0$
If n is not prime, then $n = pa$ where $a = \frac{n}{p}$ and p is a prime divisor of n and if we take $k = p$ we have that $\gcd(n, k) > 1$ so by Sivek n is k-balancing and we have $k \in S_n \Rightarrow s_n \neq 0$
(b) Suppose that $s_n = 1$. If $n = 2$ or $n = 3$ then by (a) $s_n = 0$. Since $s_n < 2$ by Lemma 3.3 $n < 5$ so the only possible value is $n = 4$.
If $n = 4 \Rightarrow S_n = \{2\} \Rightarrow s_n = 1$

□

Example 3.6. We can observe in figure 5 that the graph of the sequence follows specific patterns if n is not a prime number and also that the sequence $(s_n)_{n\in\mathbb{N}}$ is bounded and has several properties that we will discuss in Theorem 3.7.

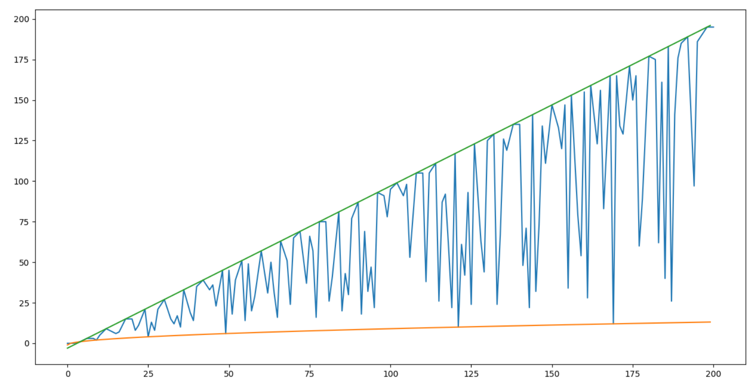


Figure 5: n = 200

Remark. *The symmetry of Sivek theorem and Euler's totient function in [4] will be useful to us in Theorem 3.7.*

(1) We know that the sum of all the nth roots of unity is 0 and since we do not allow repetition of roots, if a subset of the nth roots of unity has vanishing sum so does its complement. Thus if n *is* k*-balancing and* $k \in \mathbb{N}p_1 + \cdots + \mathbb{N}p_r$ *then by symmetry so is* $n - k$.

(2) Euler's totient function is defined by:

$$\phi(n) = n \prod_{p_i|n} (1 - \frac{1}{p_i})$$

Where p_i *are prime factors of* n *and* $1 \leqslant i \leqslant r$*. This function counts the positive integers that are relatively prime to* n.
In fact: $\phi(n) = \#\{1 \leqslant k \leqslant n \mid gcd(n,\, k) = 1\}$.

Theorem 3.7. *Let n be non-prime. The sequence $(s_n)_{n\in\mathbb{N}}$ has the following properties:*

(i) *(a)* $\forall n \geqslant 2$ *we have* $s_n \leqslant n-3$
(b) $s_n = n-3 \Leftrightarrow 6|n$

(ii) $\forall n \geqslant 3$ *we have* $s_{2n} \geqslant n-1$

(iii) $\forall p$ *with p prime,* $\forall r \geqslant 1$ *then* $s_{p^r} = p^{r-1}-1$

(iv) $\forall n \geqslant 1$ *we have* $s_n \geqslant \sqrt{n}-1$

(v) *If* $n = p^2$ *for p prime* $\Rightarrow s_n = \sqrt{n}-1$

(vi) $\lim_{n\to\infty} s_n = +\infty$

Proof. (i) (a) Let $n = p_1^{e_1}\cdots p_r^{e_r}$, we have that $1 \leqslant s_n \leqslant n-1$. For n to be 1-balancing, 1 and $n-1$ have to be in $\mathbb{N}p_1 + \cdots + \mathbb{N}p_r$. We can observe that $1 \neq p_1a_1+\cdots+p_ra_r$ for $a_1,\cdots,a_r \in \mathbb{N}$ since $1 < p_1 < p_2 < \cdots < p_r$ and at least one $a_i \neq 0$. Because $n-1 \in \mathbb{N}p_1+\cdots+\mathbb{N}p_r$ is a necessary condition for Sivek theorem to hold, we can conclude by symmetry that $1 \notin S_n$ and $n-1 \notin S_n$. This means that $\forall n \geqslant 2$ we get $s_n \leqslant (n-1)-2 = n-3$.
(b) $\Rightarrow$ Let us suppose that $n \geqslant 5$ then by Lemma 3.3 we know that $s_n \geqslant 2$. If $s_n = n-3$ this means that $k \in S_n$ for $2 \leqslant k \leqslant n-2$ so in particular $2 \in S_n$ and $3 \in S_n$. By Sivek we know that 2 and 3 are in $\mathbb{N}p_1+\cdots+\mathbb{N}p_r$ as well as $n-2$ and $n-3$. Since 2 is the smallest prime, we cannot write 2 as a sum of prime divisors of n that are different from 2. Thus 2 must be one of the prime divisors of n.
Similar, we have that $3 = 2+1$ where 2 is prime but 1 is not prime, again we cannot write 3 as a sum of prime divisors of n that are different from 3. Thus 3 is also a prime divisor of n. Since $2|n$ and $3|n \Rightarrow 6|n$.
$\Leftarrow$ Let $2 \leqslant k \leqslant n-2$, if k is even then $k \in 2\mathbb{N}$. Since 3 is the smallest odd number, if k is odd then $k \in 3+2\mathbb{N}$. Thus $k \in 2\mathbb{N}+3\mathbb{N}$ and by symetry of Sivek theorem, for n to be k-balancing, we have that $n-k \in 2\mathbb{N}+3\mathbb{N}$.
(ii) If $2 \leqslant k \leqslant 2n-2$ is even then for $k' \in \mathbb{N}, 2n-k = 2(n-k')$ so $k, 2n-k \in \mathbb{N}2$ and by Sivek $2n$ is k-balancing. Thus all the even k between 2 and $2n-2$ are in S_{2n} and we must have $s_{2n} \geqslant n-1$ since there are $\frac{2n-2}{2}$ possible even values for k.
(iii) Let p be prime, $r \geqslant 1$ and $1 \leqslant k \leqslant p^r-1$.
If p^r is k-balancing then $k, p^r-k \in p\mathbb{N}$ so in particular $s_n \geqslant \frac{p^r}{p}-1 = p^{r-1}-1$. We need to subtract 1 since $n = p^r \notin S_n$. Furthermore, since $k \leqslant p^r-1$ then S_n contains exactly the number of multiples of p for $1 \leqslant k \leqslant p^r-1$. Using Euler's totient function we can compute s_n the following way:

$$\phi(p^r) = p^r(1-\frac{1}{p}) = p^r - p^{r-1}$$

$\phi(p^r) = \#\{1 \leqslant k \leqslant p^r |\ \gcd(p^r, k) = 1\}$ and since we want only the multiples of p we have:

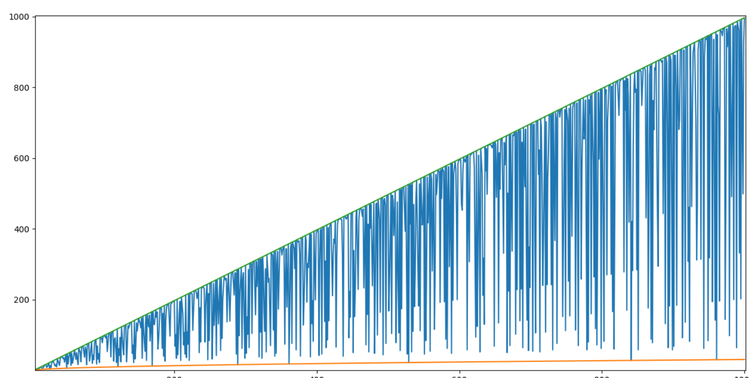
$$s_n = p^r - \phi(p^r) - 1 = p^r - p^r + p^{r-1} - 1 = p^{r-1} - 1$$

(iv) Let $\pi(n)$ be the smallest prime dividing n.
$c \times \pi(n) \in S_n$ for $1 \leqslant c \leqslant \frac{n}{\pi(n)} - 1$. Thus $s_n \geqslant \frac{n}{\pi(n)} - 1$
We have that $n = ab$ for $a = \pi(n)$ and $b = \frac{n}{\pi(n)}$ and since $a \leqslant b$ by Lemma 3.3.
$b \geqslant \sqrt{n}$ thus $s_n \geqslant \sqrt{n} - 1$
(v) This is a special case of (iii)
(vi) By point (iv) $s_n \xrightarrow[n\to\infty]{} \infty$ $\square$

On a short program using the Main Theorem, we can compute the sequence:

```
# s_n = #{0<k<n : the n-centrifuge can balance k tubes}
def sequenceCentrif(n):
    seq = [0,0]
    for i in range(2, n+1):
        seq.append(len(centrifList(i))- 2)
    return(seq)
```

Example 3.8. Without the prime numbers, we can plot the graph for $n = 1000$ as such in blue with upper bound $n-3$ (green) and lower bound $\sqrt{n}-1$ (orange):



4 Multiple centrifuges

In this section we want to study if it is possible to find a balanced configuration for multiple unbalanced centrifuges that are placed on top of another. Although we will not prove anything rigorously, we want to give an idea of what the problem looks like. A centrifuge is not balanced if n is not k-balancing, in other

words if the center of gravity of the tubes does not coincide with the center of gravity of the centrifuge. In that case the tubes are not regularly arranged in the centrifuge.
The question is the following: given m centrifuges C_i for $1 \leqslant i \leqslant m$ with different radius $r_i \in \mathbb{R}$, such that all C_i are centered at 0. Can we find k_i where k_i is the number of tubes inserted in centrifuge C_i such that C_i is not k_i-balancing but $\bigcup C_i$ is balanced.
In order for this to work, all the centrifuges need to have the same number of slots n and $1 \leqslant k_i \leqslant n$. We want to say that one or more solutions exist.

Approach

In the case where a centrifuge is not balanced, the center of gravity of the tubes does not coincide with the one of the centrifuge. If we insert a unbalanced configuration of tubes, the sum of the roots of unity corresponding to the occupied spots is not 0. We can look at this problem in two different ways:

- By Sivek we know how to find all k which can balance a centrifuge. So in particular we are able to find all k which balance a specific centrifuge C_i and adding balanced configurations on top of another would still be balanced. If we consider unbalanced configurations we cannot say much.
- Another approach consists in considering the center of gravity of the tubes k_i for $1 \leqslant i \leqslant m$, denoted g_i respectively and check if:

$$\sum_{i=1}^{m} g_i = 0$$

  If it is the case then $\bigcup C_i$ is balanced and if not, one might have to rotate one or more of the C_i in order find the right configuration. Knowing which rotation should be applied would be based on finding the centers of gravity that cancel each other.

The second approach is better suited in this case because we cannot use Sivek. The center of gravity changes depending on how the tubes are inserted in the different centrifuges. It may be sufficient to rotate the right configurations to get a balanced configuration with the tubes which are already inserted. So the goal is to find different configurations for the centrifuges which are not balanced so that the center of gravity $g_i \neq 0$. For all the C_i to be balanced we need to find centers of gravity which balance each other out.

Remark. *Let R be the set of all the nth roots of unity. We define $\zeta_n = e^{2\pi i/n}$ and $(\zeta_n)^{r_i} = e^{r_i 2\pi i/n} \in R$ with s_i being the spots where the k_i tubes are inserted for $1 \leqslant r_i \leqslant n$.*
The center of gravity g_i of k_i inserted tubes is :

$$g_i = \frac{\sum_{i=1}^{k_i} \zeta_n^{s_i}}{k}$$

2 unbalanced centrifuges

In order to generalize this result for multiple unbalanced centrifuges, we want to look at the case with only 2 unbalanced centrifuges and try to understand how to find a solution.
Here are two pictures of unbalanced centrifuges, for $n = 18$ in figure 6 and for $n = 35$ in figure 7.

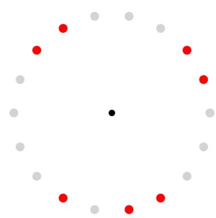


Figure 6
7 tubes

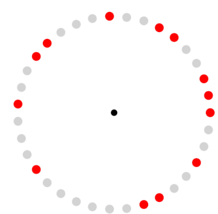


Figure 7
13 tubes

Example

Let C_1 and C_2 be two centrifuges, both with n slots. Centrifuge C_1 has a radius of 1 and centrifuge C_2 has radius 2. We want to find k_1 tubes inserted in C_1 and k_2 tubes inserted in C_2 and let g_1 and g_2 be their respective center of gravity, it is possible to find balanced configurations for $C_1 \cup C_2$ if the tubes are placed such that:

$$g_1 + g_2 = 0 \text{ and } g_1 \neq g_2 \neq 0$$

We might have to rotate the k_1 or k_2 tubes for the sum of their respective center of gravity to vanish.

m unbalanced centrifuges

Let $m \in \mathbb{N}$ be the number of centrifuges and k_i the number of tubes inserted in the centrifuge C_i with respective center of gravity g_i for $1 \leqslant i \leqslant m$. It is possible to balance the m centrifuges if we can find $\sum_{i=1}^{m} g_i = 0$.

References

[1] Gary Sivec, On vanishing sums of distinct roots of unity, Integer **10** (2010), 365-368

[2] Lam, T. Y. and Leung, K. H., On vanishing sums of roots of unity, J. Algebra **224** (2000), 91–109

[3] Sylvester, J. J., Question 7382, Mathematical Questions from the Educational Times **41** (1884)

[4] Euler's totient function, *LaTeX: Computing Euler's totient function*, Wikipedia `https://en.wikipedia.org/wiki/Euler%27s_totient_function#Computing_Euler's_totient_function`

[5] Holly Krieger, The Centrifuge Problem - Numberphile, YouTube (2018) `https://www.youtube.com/watch?v=7DHE8RnsCQ8`