Meccano pentagons

https://github.com/heptagons/meccano/penta

1 Meccano pentagons

To identify a pentagon we use two angles A and B. Some identities are solved for $a + b\sqrt{5}$ values to be used later.

$$
\begin{aligned}
5A &= 2\pi \\
5B &= \pi \\
4\cos(A) &= -1 + \sqrt{5} \\
4\cos(B) &= 1 + \sqrt{5} \\
8\cos^2(A) &= 3 - \sqrt{5} \\
8\cos^2(B) &= 3 + \sqrt{5} \\
4\cos(A)\cos(B) &= 1 \\
8\sin^2(A) &= 5 + \sqrt{5} \\
8\sin^2(B) &= 5 - \sqrt{5} \\
4\sin(A)\sin(B) &= \sqrt{5}
\end{aligned}
$$

1.1 Pentagons of type 1

A pentagon of type 1 shown in Figure **??**. We note three rods (or sections of rods) a, b, and c at fixed angles and with integer sizes as for any meccano figure. We want to find the fourth rod d which also needs to be of integer size to make the pentagon.

We start by looking the rods' related formulas:

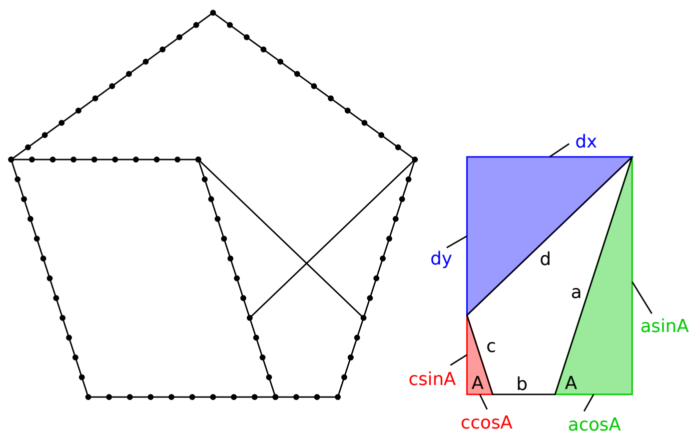

Figure 1: From integer rods a, b and c we expect to find the rod d also integer to make a pentagon of type 1. Actually, the pentagon shown is the unique solved so far for small values of rods, $a = 12$

$$
\begin{aligned}
d_x^2 &= ((a+c)\cos(A)+b)^2 \\
&= (a+c)^2\cos^2(A) + 2(a+c)b\cos(A) + b^2 \\
d_y^2 &= ((a-c)\sin(A))^2 \\
&= (a-c)^2\sin^2(A) \\
d^2 &= d_x^2 + d_y^2 \\
&= (a+c)^2\cos^2(A) + (a-c)^2\sin^2(A) + 2(a+c)b\cos(A) + b^2 \\
&= (a+c)^2(3-\sqrt{5})/8 \\
&\qquad + (a-c)(5+\sqrt{5})/8 \\
&\qquad + 2(a+c)b(-1+\sqrt{5})/4 \\
&\qquad + b^2 \\
&= m\sqrt{5} + n
\end{aligned}
$$

We define two variables m and n. m is the sum of all the terms multiplied by $\sqrt{5}$ while n is the sum of all the terms not multipled by $\sqrt{5}$.

$$
\begin{aligned}
8m &= -(a+c)^2 + (a-c)^2 + 4(a+c)b \\
&= 4(a+c)b - 4ac \\
8n &= 3(a+c)^2 + 5(a-c)^2 - 4(a+c)b + 8b^2
\end{aligned}
$$

Simplifying we get a value for the rod d^2 in function of the rest of rods.

$$
\begin{aligned}
m &= \frac{ab - ac + bc}{2} \\
n &= a^2 + b^2 + c^2 - \frac{ab + ac + bc}{2} \\
&= a^2 + b^2 + c^2 - ac - m \\
d^2 &= m\sqrt{5} + a^2 + b^2 + c^2 - ac - m
\end{aligned}
$$

Now, we want rod d to be as simple as possible so is good idea to make $m = 0$ wich requires $ac = (a+c)b$. This way the rod d is a simpler function of a, b and c.

$$
\begin{aligned}
ac &= (a+c)b \\
d &= \sqrt{a^2 + b^2 + c^2 - ac}
\end{aligned}
$$

1.1.1 Pentagon type 1 search

With last equations, a program can iterate over the integer values of the rods a, b and c to discover the rod d to be integer too. Next javascript program was run and found a single solution $a = 12, b = 3, c = 4, d = 11$ after 5000 iterations. Scaled solutions are discarded as repetitions.

```
function meccano_pentagons_1(sols)
{
  this.find = (max)=> {
    for (let a=1; a < max; a++)
      for (let b=1; b <= max; b++)
        for (let c=0; c <= a; c++)
          if (a*c == (a + c)*b)
            mZero(a, b, c)
  }
```

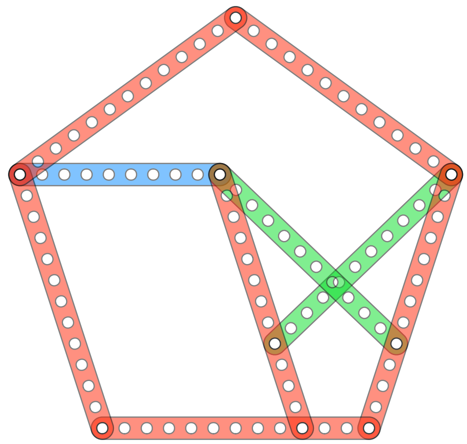

Figure 2: The smallest and maybe unique of pentagons of type 1.

```
  const mZero = (a, b, c)=> {
    const d = Math.sqrt(a*a + b*b + c*c - a*c)
    if (d > 0 && d % 1 === 0)
      dInteger(a, b, c, d)
  }
  const dInteger = (a, b, c, d) => {
    for (let i=0; i < sols.length; i++) {
      const s = sols[i]
      if (a % s.a == 0) {
        const f = a / s.a
        const bS = (b % s.b == 0) && b / s.b == f
        const cS = (c % s.c == 0) && c / s.c == f
        const dS = (d % s.d == 0) && d / s.d == f
        if (bS && cS && dS)
          return // scaled solution already
      }
    }
    sols.push({ a:a, b:b, c:c, d:d }) // solution!
  }
}
```

1.2 Pentagons of type 2

A pentagon of type 2 is shown in figure **??**. We identify in this type of pentagon four rods a, b, c and d at fixed angles. We want to find a fifth rod e with integer length to make the pentagon. Actually, the example pentagon shown is the smallest found $a = 12, b = 2, c = 9, d = 6, e = 11$. For each solution there are two versions whether the green rods are used or the green ones.

We start with the rods relation formulas

$$
\begin{aligned}
e_x &= b\cos(A) + a + c\cos(A) - d\cos(B)\\
&= a + (b+c)\cos(A) - d\cos(B)\\
e_y &= c\sin(A) - b\sin(A) - d\sin(B)\\
&= (c-b)\sin(A) - d\sin(B)\\
e^2 &= e_x^2 + e_y^2\\
&= a^2 + (b+c)^2\cos^2(A) + d^2\cos^2(B)\\
&\qquad + 2a(b+c)\cos(A) - 2ad\cos(B)\\
&\qquad - 2(b+c)d\cos(A)cos(B)\\
&\qquad + (c-b)^2\sin^2(A) + d^2\sin^2(B)\\
&\qquad - 2(c-b)d\sin(A)\sin(B)\\
&= a^2 - 2(b+c)d/4\\
&\qquad + (b+c)^2(3-\sqrt{5})/8\\
&\qquad + d^2(3+\sqrt{5})/8\\
&\qquad + 2a(b+c)(-1+\sqrt{5})/4\\
&\qquad - 2ad(1+\sqrt{5})/4\\
&\qquad + (c-b)^2(5+\sqrt{5})/8\\
&\qquad + d^2(5-\sqrt{5})/8\\
&\qquad - 2(c-b)d(\sqrt{5})/4\\
&= m\sqrt{5} + n
\end{aligned}
$$

As we did with the pentagon type 1, we define variables m and n:

$$8m = -(b+c)^2 + d^2 + 4a(b+c) - 4ad + (c-b)^2 - d^2 - 4(c-b)d$$
$$8n = 8a^2 + 3(b+c)^2 + 3d^2 - 4a(b+c) - 4ad - 4(b+c)d + 5(c-b)^2 + 5d^2$$

Simplifying, we get a value for rod e in function of the rest of rods:

$$\begin{aligned} m &= \frac{(a-b)(c-d) + ab - cd}{2} \\ n &= a^2 + b^2 + c^2 + d^2 - \frac{(a+b)(c+d) + ab + cd}{2} \\ &= a^2 + b^2 + c^2 + d^2 - ad - bc - cd - m \\ e^2 &= m\sqrt{5} + a^2 + b^2 + c^2 + d^2 - ad - bc - cd - m \end{aligned}$$

Again we decide to make $m = 0$ which now requires $cd = (a-b)(c-d) + ab$. This way the rod e is a simple function of rods a, b, c and d:

$$\begin{aligned} cd &= (a-b)(c-d) + ab \\ e &= \sqrt{a^2 + b^2 + c^2 + d^2 - ad - bc - cd} \end{aligned}$$

1.2.1 Pentagon type 2 search

With last equations, another program, for the pentagon type 2, can iterate over the integer values of rods a, b, c and d to discover a rod e with integer length too. Next javascript program was run and found 40 different pentagons with rods length <= 183.

```
function meccano_pentagons_2(sols)
{
  this.find = (max) => {
    for (let a=1; a < max; a++) {
      for (let b=1; b < a; b++)
        for (let c=1; c < a; c++)
          for (let d=1; d < a; d++)
            if ((a - b)*(c - d) + a*b == c*d)
              mZero(a, b, c, d)
    }
  }
```

```
  const mZero = (a, b, c, d)=> {
    const e = Math.sqrt(a*a + b*b + c*c + d*d - a*d - b*c - c*d)
    if (e > 0 && e % 1 === 0)
      eInteger(a, b, c, d, e)
  }
  const eInteger = (a, b, c, d, e)=> {
    for (let i=0; i < sols.length; i++) {
      const s = sols[i]
      if (a % s.a == 0) {
        const f = a / s.a
        const bS = (b % s.b == 0) && b / s.b == f
        const cS = (c % s.c == 0) && c / s.c == f
        const dS = (d % s.d == 0) && d / s.d == f
        const eS = (e % s.e == 0) && e / s.e == f
        if (bS && cS && dS && eS)
          return // scaled solution already
      }
    }
    sols.push( { a:a, b:b, c:c, d:d, e:e }) // solution
  }
}
```