SOLVING THE **FERMAT-WEBER PROBLEM**

A NUMERICAL AND GEOMETRIC APPROACH

Beat Trachsler, Martin Guggisberg
GeoGebra Institute of PH FHNW
Switzerland

Abstract:

"Given three points in a plane, find a fourth point such that the sum of its distances to the three given points is as small as possible." This historical problem from the 17th century was put by the French mathematician Fermat to the Italian physicist Torricelli. A modern formulation of this problem could be: "To find a best location for a power plant between three cities in such a way that the sum of the connections between the power plant and the cities is minimal". Torricelli found several mathematical proofs, which can be nicely presented by GeoGebra. A generalization of the original problem leads to a geometric median – the problem of minimizing the sum of weighted distances.

Finding a geometric median is an optimization problem, which has no analytical solution for more than four points. We will present an interactive numerical solution with GeoGebra using the Weiszfeld algorithm and the programming language Python. Our GeoGebra Materials calculate the geometric median in real-time, this allow allows dynamic observations of the solution and the convergence behavior of the Weiszfeld algorithm.

OUTLOOK

- Introduction
- Mathematical exploration with GeoGebra
- Theory around the Fermat-Weber Problem
- Implementation of the Weiszfeld algorithm

FERMAT'S PROBLEM FOR TORRICELLI

Given three points in a plane, find a fourth point such that the sum of its distances to the three given points is as small as possible.

This historical problem was put by the French mathematician Fermat to the Italian physicist Torricelli.

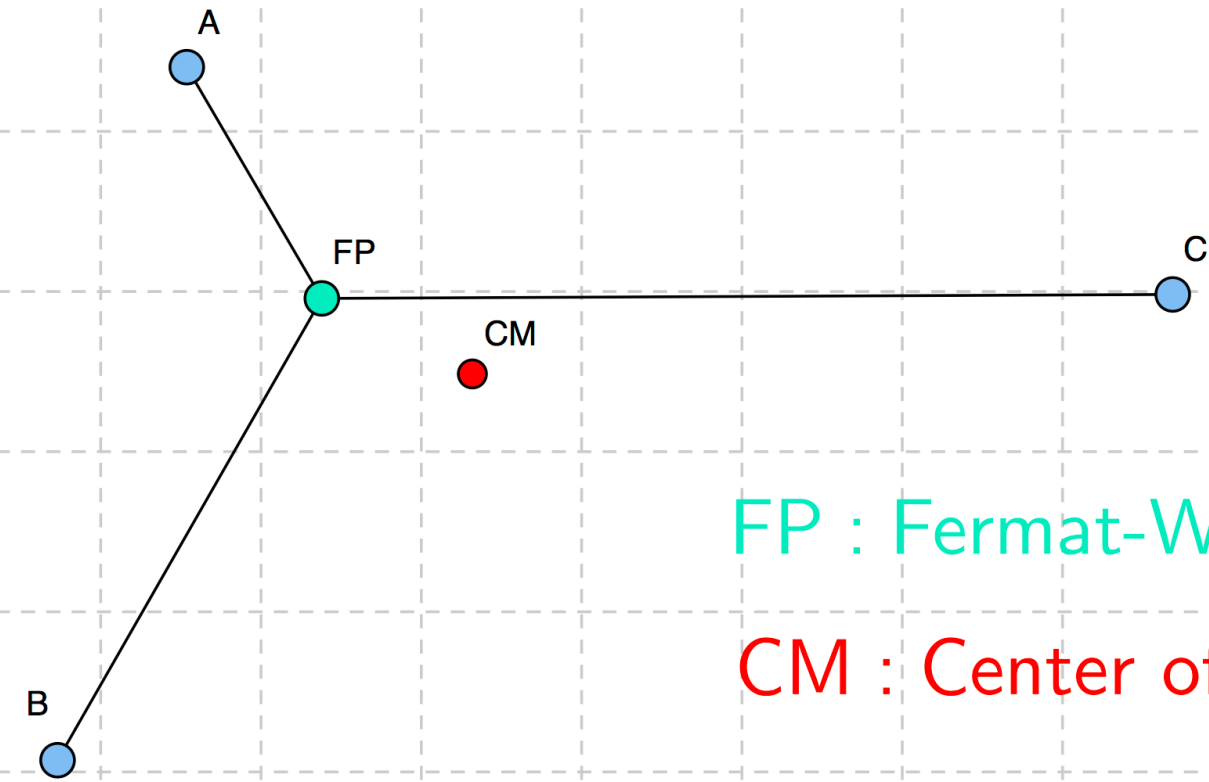
Dorrie, H. (1965). 100 Great problems of elementary mathematics. Dover Publications.

SOME OF MANY NAMES

The long history and the interdisciplinary application have given several names to the problem and its variations: the Fermat problem, the generalized Fermat problem, the Fermat-Torricelli problem, the Steiner problem, the generalized Steiner problem, the Steiner-Weber problem, the Weber problem, the generalized Weber problem, the Fermat-Weber problem, the one median problem, the median center problem, the spatial median problem, the bivariate median problem, the minimum aggregate travel point problem

MINIMIZE THE SUM OF DISTANCES

$$\text{minimize}_{x,y} \quad f(x,y) = \sum_{i=1..n} \sqrt{(x - x_i)^2 + (y - y_i)^2}$$



FP : Fermat-Weber Point

CM : Center of mass

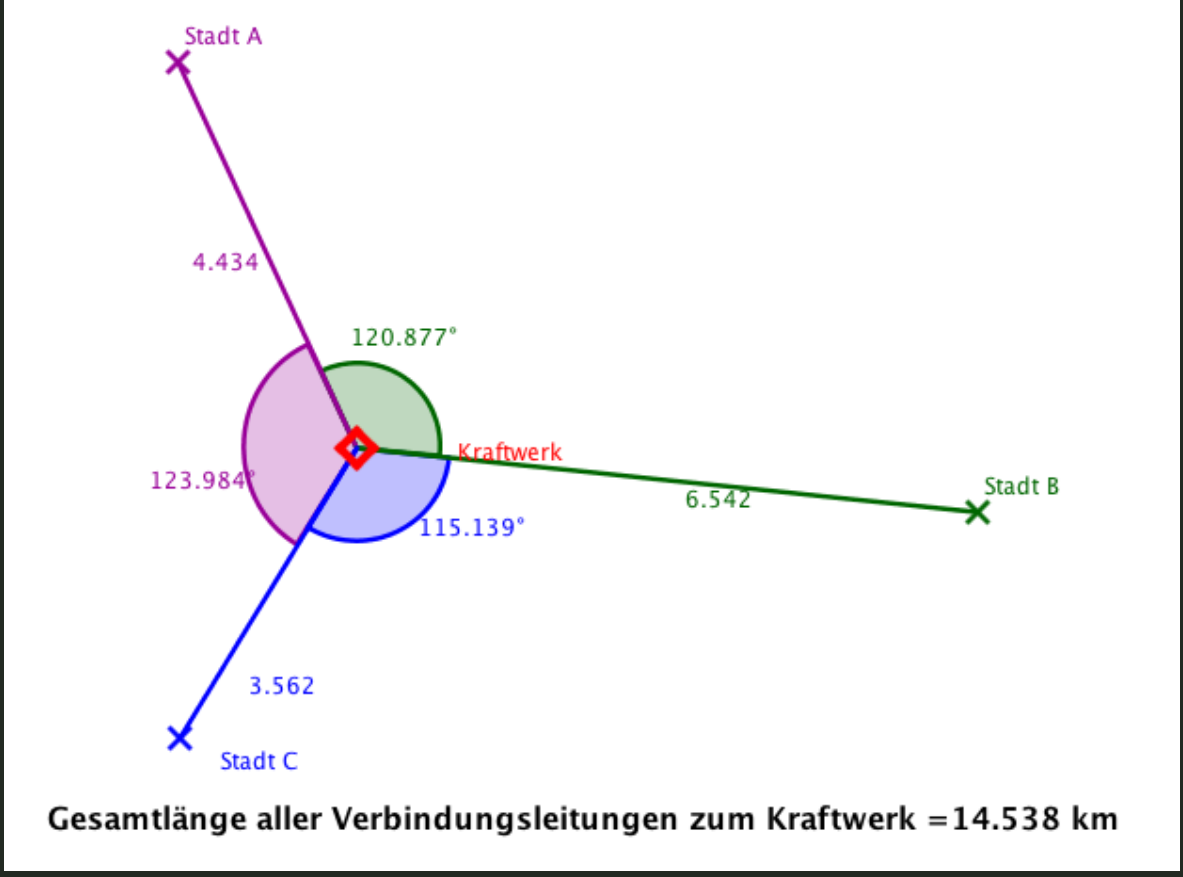
MODERN FORMULATION

The Weber problem finds the point in a plane which minimises the sum of weighted Euclidean distances to a set of fixed points.

This is interpreted as finding the factory location which minimises the total weighted distances from suppliers and customers, where weights represent relative volumes of interactions, e.g. weight of material to be transported from a supplier, or volume of finished products for a customer.

Drezner, Z., & Hamacher, H. W. (Eds.). (2004). Facility location: applications and theory. Springer.

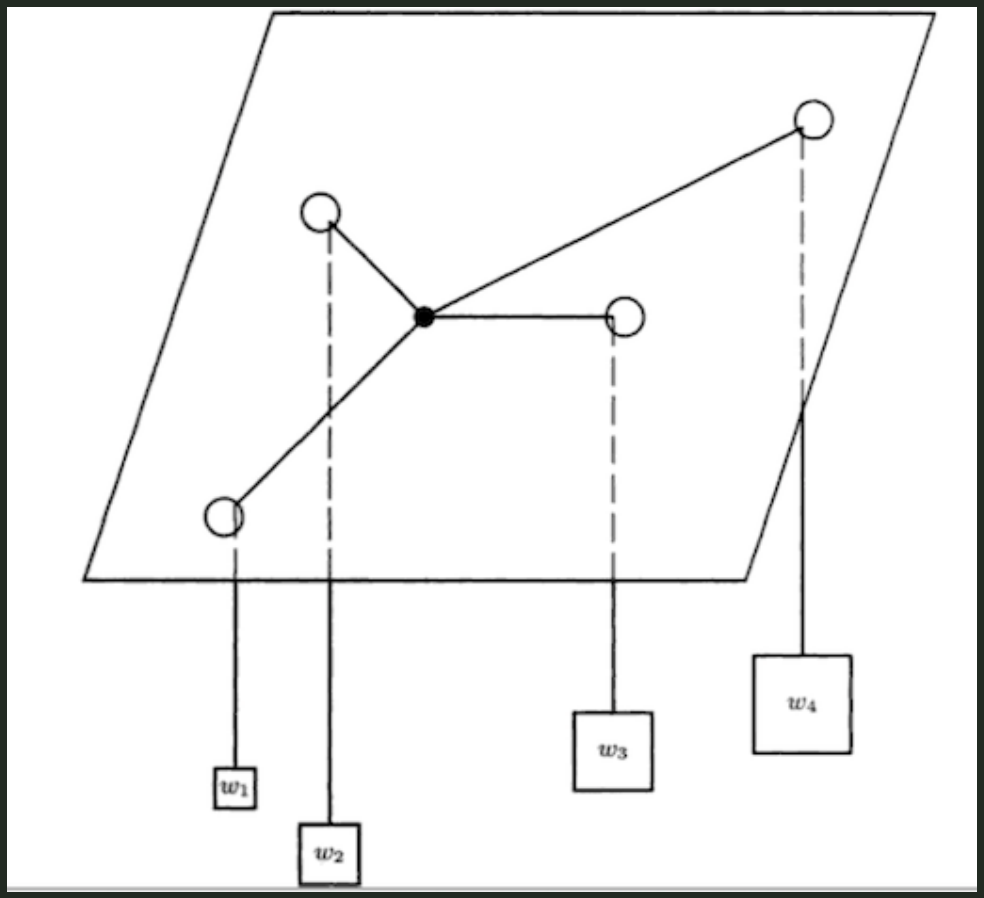
MODERN FORMULATION



Link to GeoGebra Material

Ulrich Steinmetz. (2014)

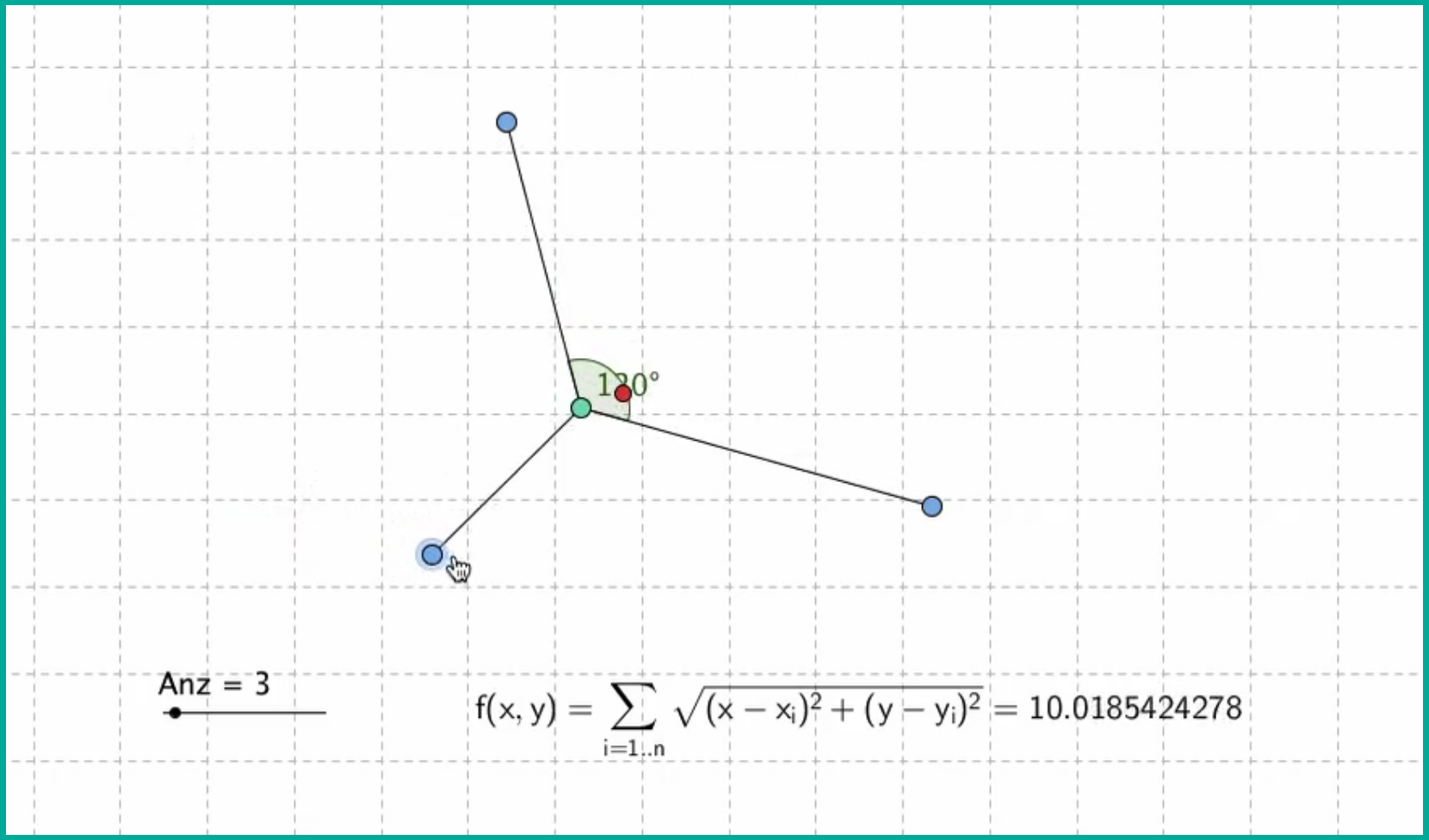
VARIGNON FRAME



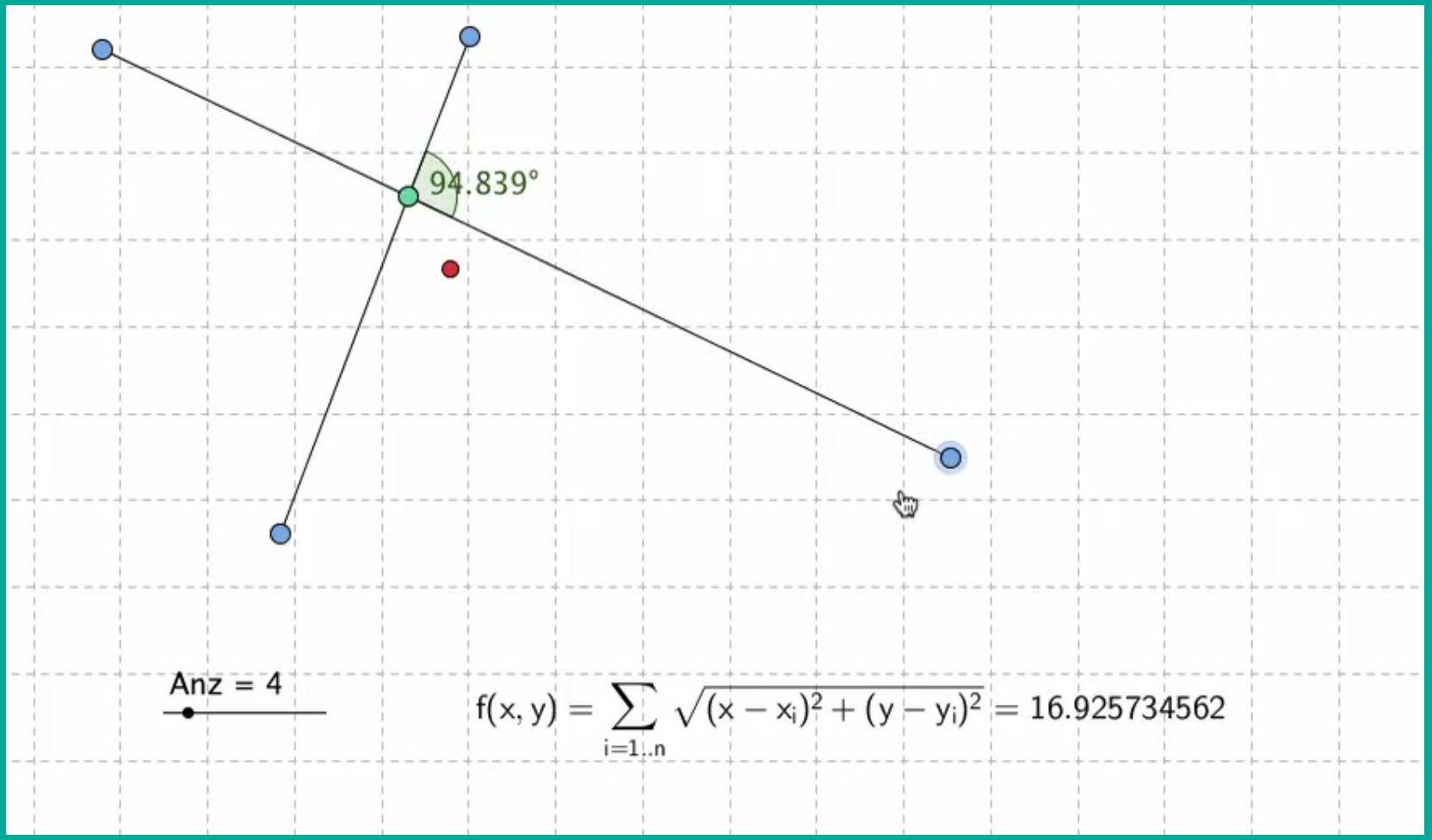
Varignon proposed a mechanical analogue device which has actually been used in practice

MATHEMATICAL EXPLORATION

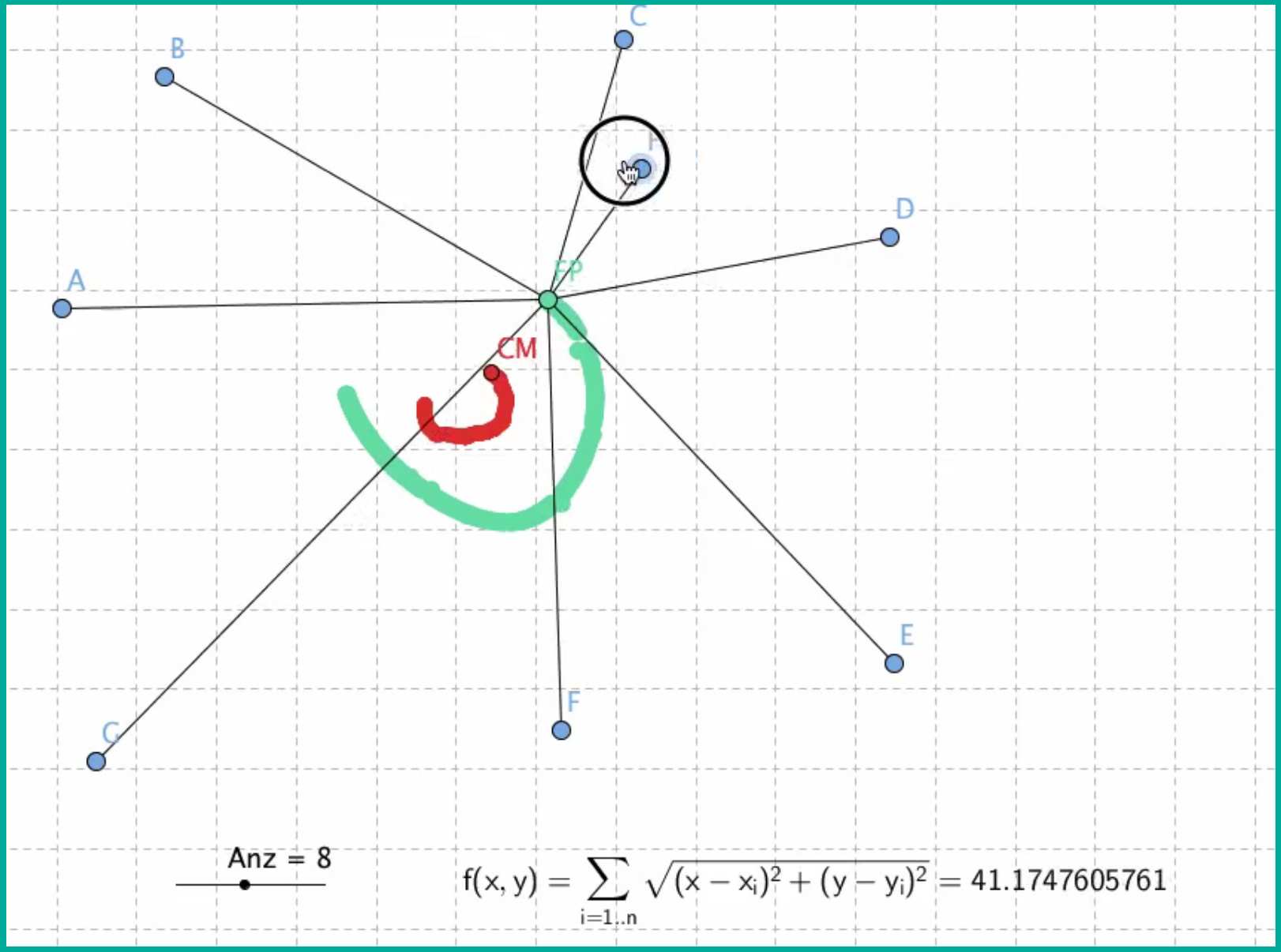
FOR N=3 THE ANGLE 120° CAN BE OBSERVED



NO ANGLE DEPENDENCY FOR MORE THAN 3 POINTS



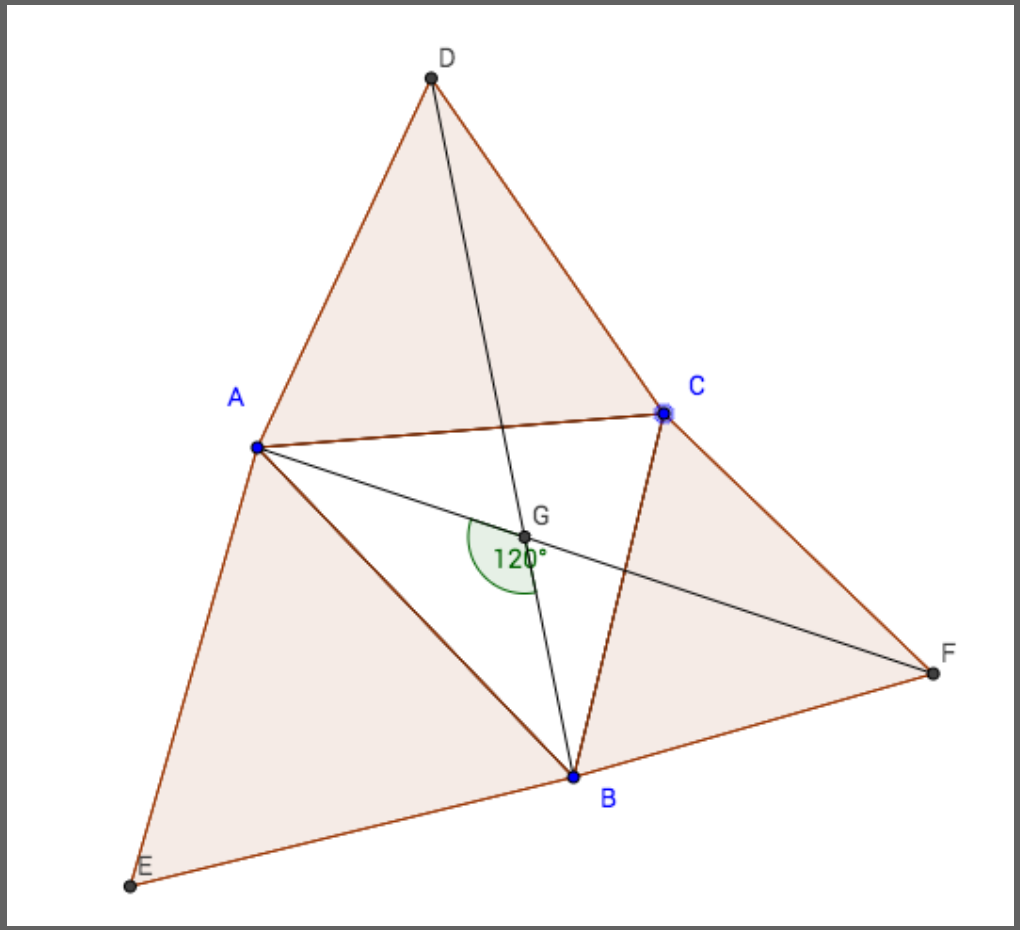
FERMAT POINT SNAP TO INTERNAL POINT



THEORY

AROUND THE FERMAT-WEBER PROBLEM

GEOMETRIC SOLUTION FOR 3 LOCATION

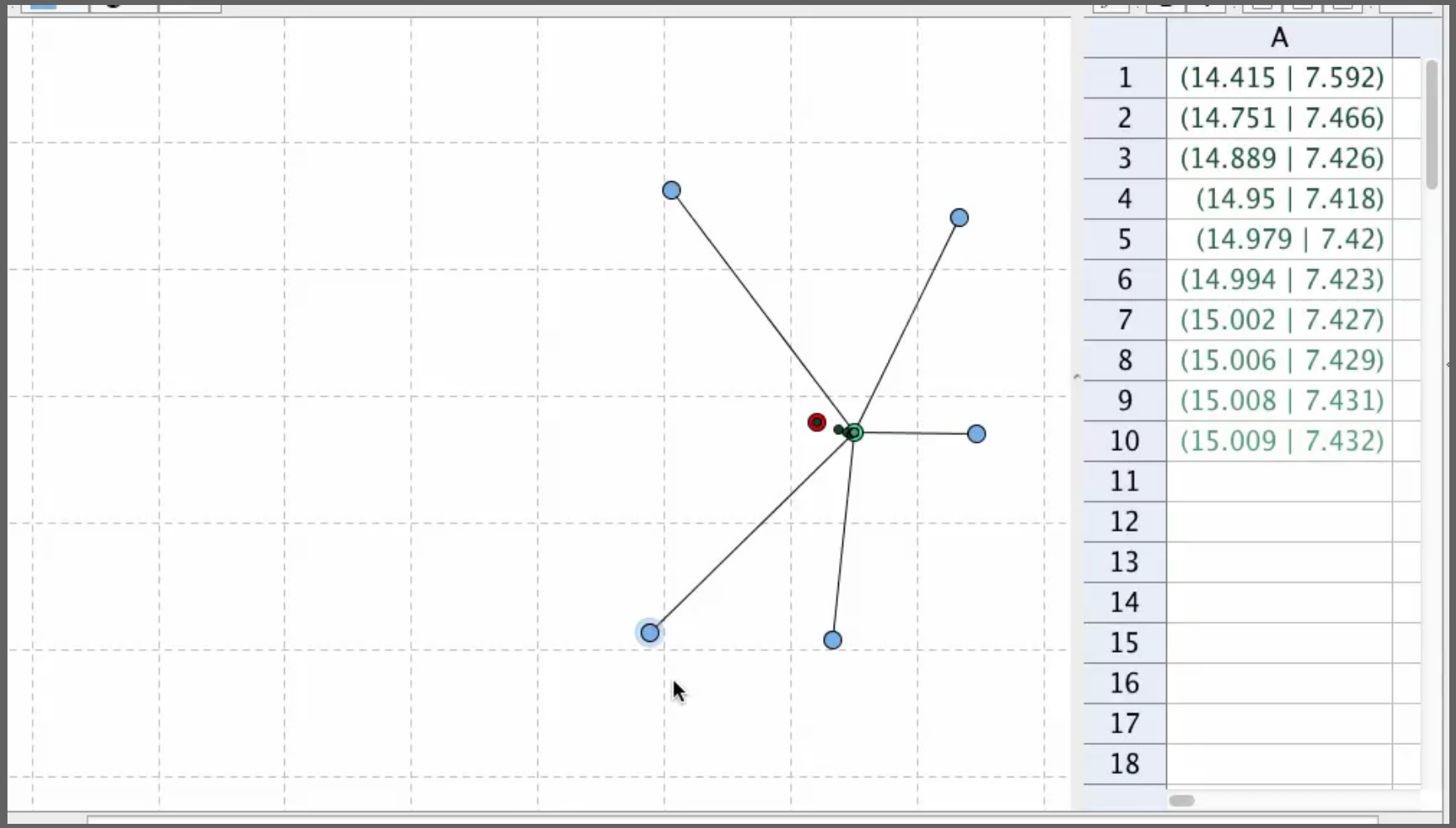


Link to Geogebra Material

WEISZFELD ALGORITHM

by Endre Weiszfeld, alias Andrew Vázsonyi (1916–2003), born in Budapest

CONVERGENCE OF THE WEISZFELD ALGORITHM

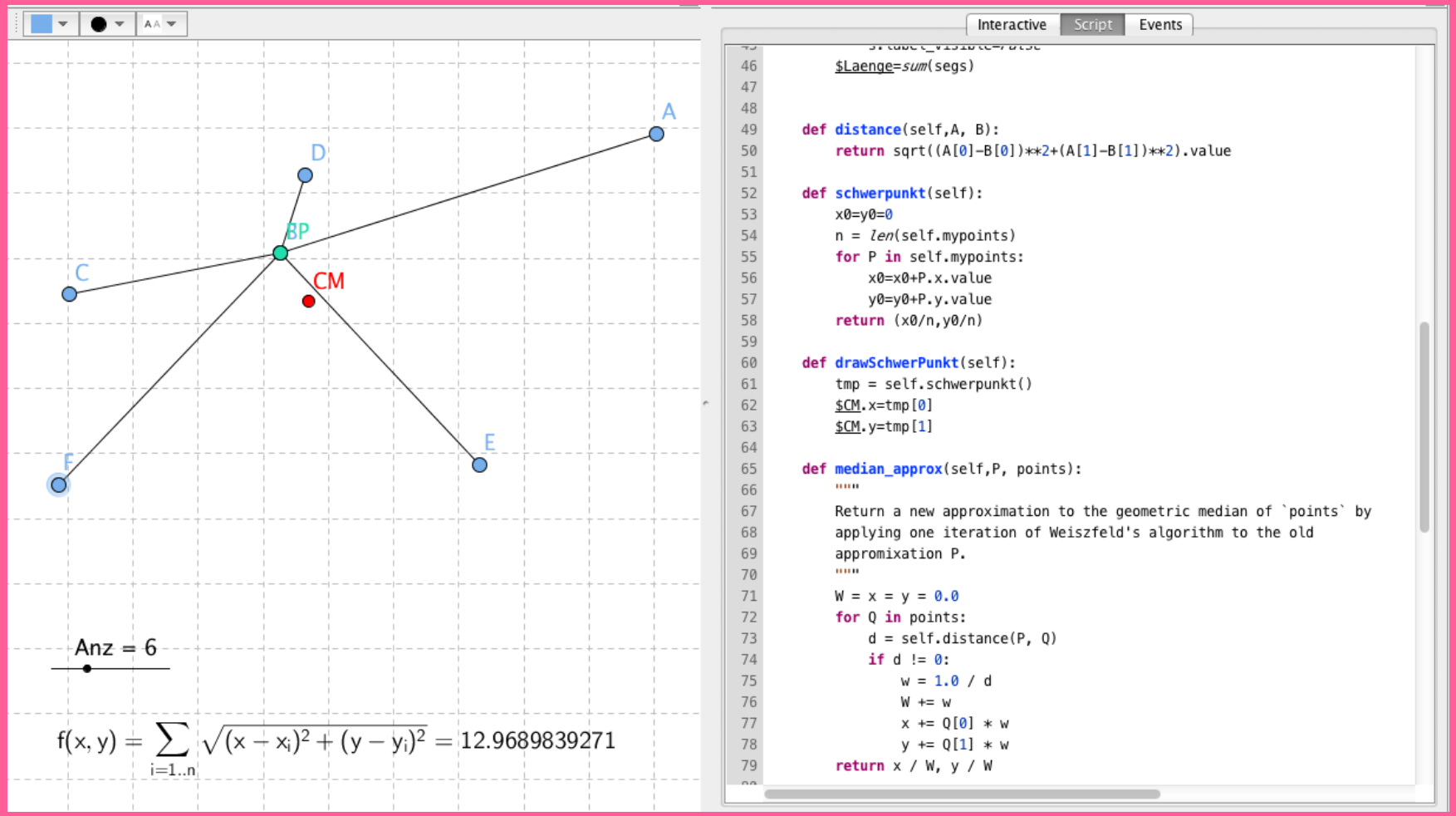


	A
1	(14.415 | 7.592)
2	(14.751 | 7.466)
3	(14.889 | 7.426)
4	(14.95 | 7.418)
5	(14.979 | 7.42)
6	(14.994 | 7.423)
7	(15.002 | 7.427)
8	(15.006 | 7.429)
9	(15.008 | 7.431)
10	(15.009 | 7.432)
11	
12	
13	
14	
15	
16	
17	
18	

ALGORITHM

IMPLEMENTATION WITH THE PROGRAMMING LANGUAGE PYTHON

PROGRAMMING WITH PYTHON IN GEOGEOBRA (BETA)



[Download Material (Geogebra 5 Beta)]

WEISZFELD CLASS

```
class Weiszfeld(object):
    mypoints = None
    text1 = None
    def __init__(self):
        ...
    def drawPoints(self):
        ...
    def distance(self,A, B):
        ...
    def schwerpunkt(self):
        ...
    def median_approx(self,P, points):
        ...
    def geometric_median(self,points, epsilon):
        ...
    def update(self):
        ...
```

DRAW RANDOM POINTS
SET EVENTLISTENER

```
class Weiszfeld(object):
    def drawPoints(self):
        N = int($Anz.value)
        xcoords = [random.uniform(3, 17) for i in range(N)]
        ycoords = [random.uniform(3, 10) for i in range(N)]
        self.mypoints = [Point(x, y,point_size=6, \
            color=Color(118,175,236), label_visible=True) \
            for x, y in zip(xcoords, ycoords)]
        ...
        for p in Point.all:
            p.onupdate=refresh
        ...
...
```

```
def init(self):
    alg.drawPoints()
    alg.drawSchwerPunkt()
    alg.drawFermatPunkt()
```

GLOBAL FUNCTION

```
# UPDATE CM AND FP
def refresh(self):
    alg.drawSchwerPunkt()
    alg.drawFermatPunkt()
```

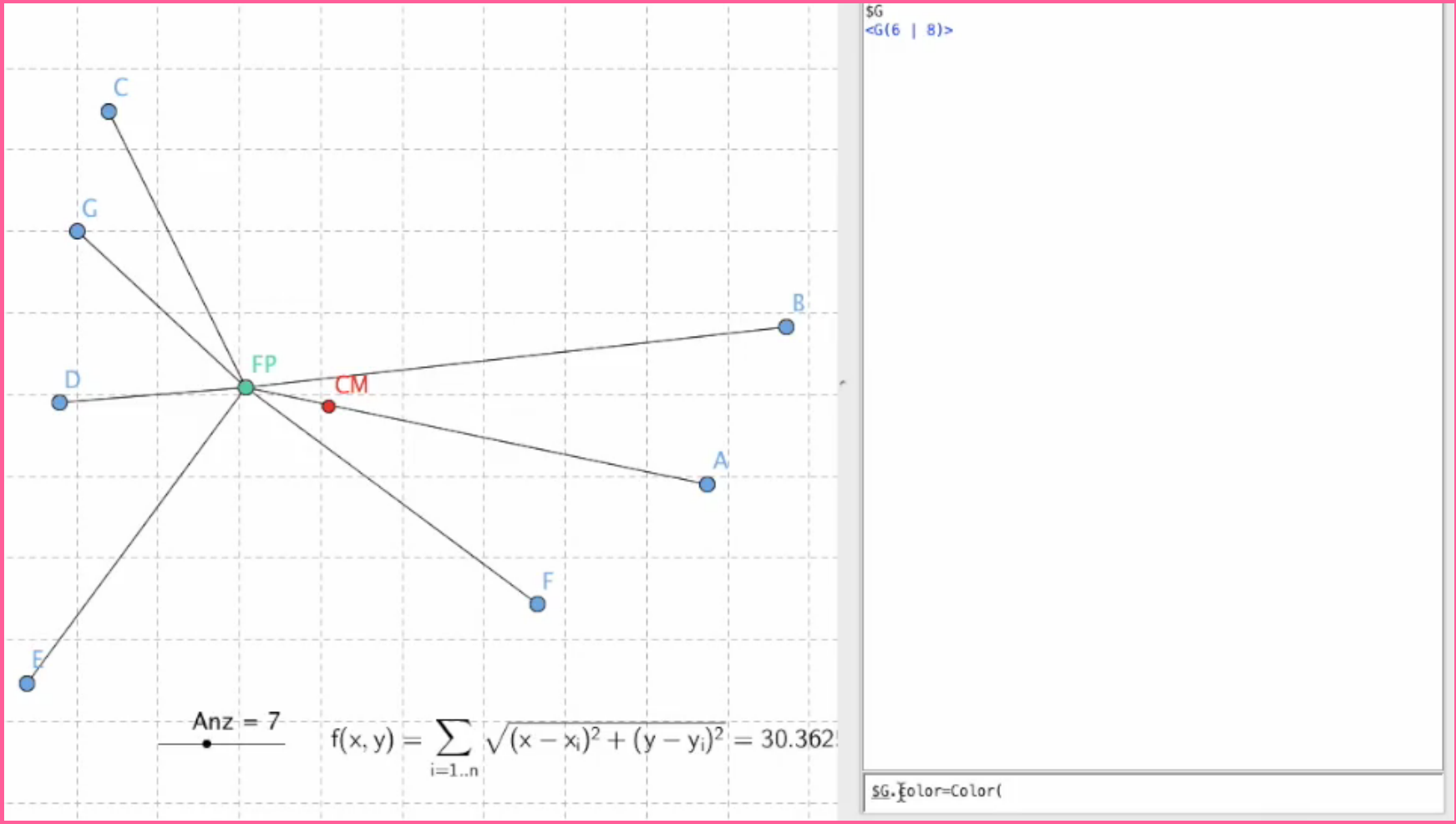
GLOBAL INITIALIZATION

```
# GLOBAL VAR
global alg
alg = Weiszfeld()
alg.drawPoints()
alg.update()
$Anz.onupdate=init
```

CALCULATE NEXT INTERATION POSITION OF FP

```
    def median_approx(self,P, points):
        """
        Return a new approximation to the geometric median
        of `points` by applying one iteration of Weiszfeld's
        algorithm to the old appromixation P.
        """
        W = x = y = 0.0
        for Q in points:
            d = self.distance(P, Q)
            if d != 0:
                w = 1.0 / d
                W += w
                x += Q[0] * w
                y += Q[1] * w
        return x / W, y / W
```

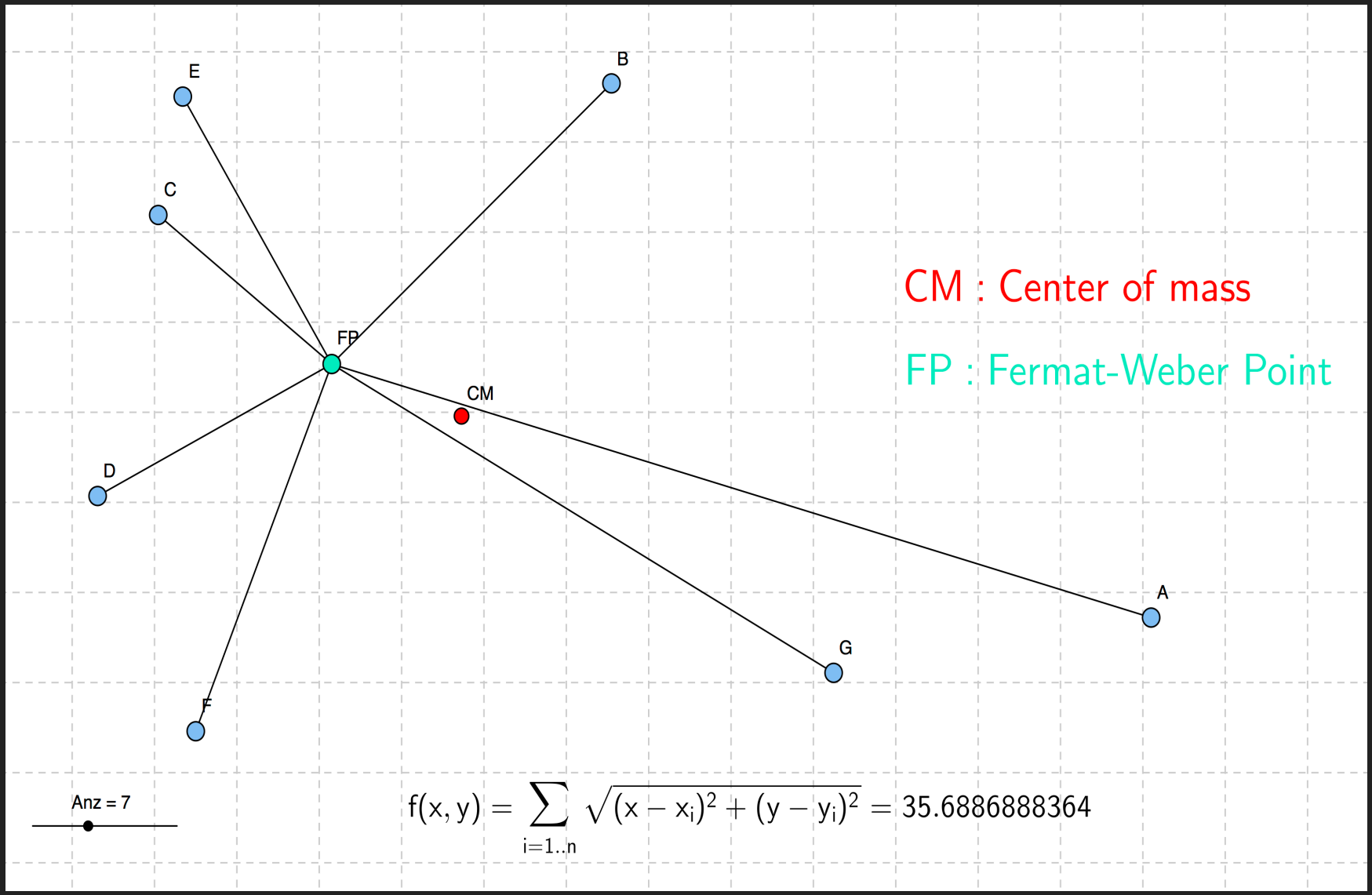
INTERACTIVE PYTHON



CONCLUSION

INTERACTIVE DEMO (JAVASCRIPT)

GeoGebra Materials : http://www.geogebratube.org/material/show/id/66049



DEMO 2

GeoGebra Materials : http://www.geogebratube.org/student/m66688

SUMMARY

- We presented the historical fermat-weber problem
- The use of GeoGebra to solve this problem
- A python implementation of the Weiszfeld algorithm

This presentation and all interactive material are published on github.

http://mgje.github.io/presentations/

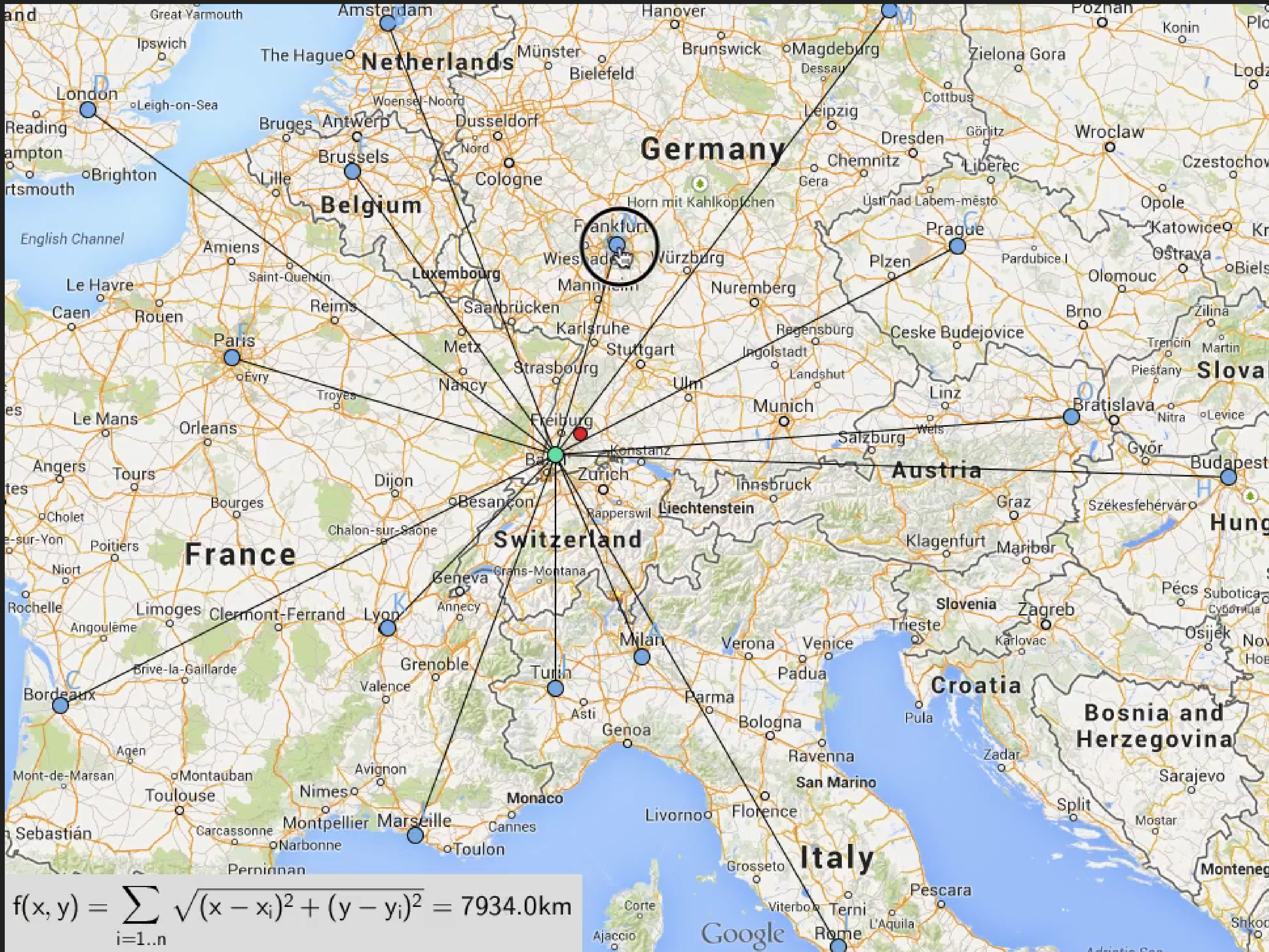
USING JAVASCRIPT

Demo Runs on the web an standard Geogebra4.4

THANKS FOR YOUR ATTENTION

FINDING THE FERMAT-WEBER POINT FOR

AMSTERDAM, BERLIN, BORDEAUX, BRUSSELS, BUDAPEST, FRANKFURT, LONDON, LYON, MARSEILLE, MILAN, PARIS, PRAGUE, ROME, TURIN AND VIENNA



$$f(x, y) = \sum_{i=1..n} \sqrt{(x - x_i)^2 + (y - y_i)^2} = 7934.0\text{km}$$

THE END

QUESTIONS ?

THANKS TO HAKIM EL HATTAB / HAKIM.SE
FOR REVEAL.JS