SOLVING THE
FERMAT-WEBER PROBLEM

A NUMERICAL AND GEOMETRIC APPROACH

Beat Trachsler, Martin Guggisberg
GeoGebra Institute of PH FHNW
Switzerland

23

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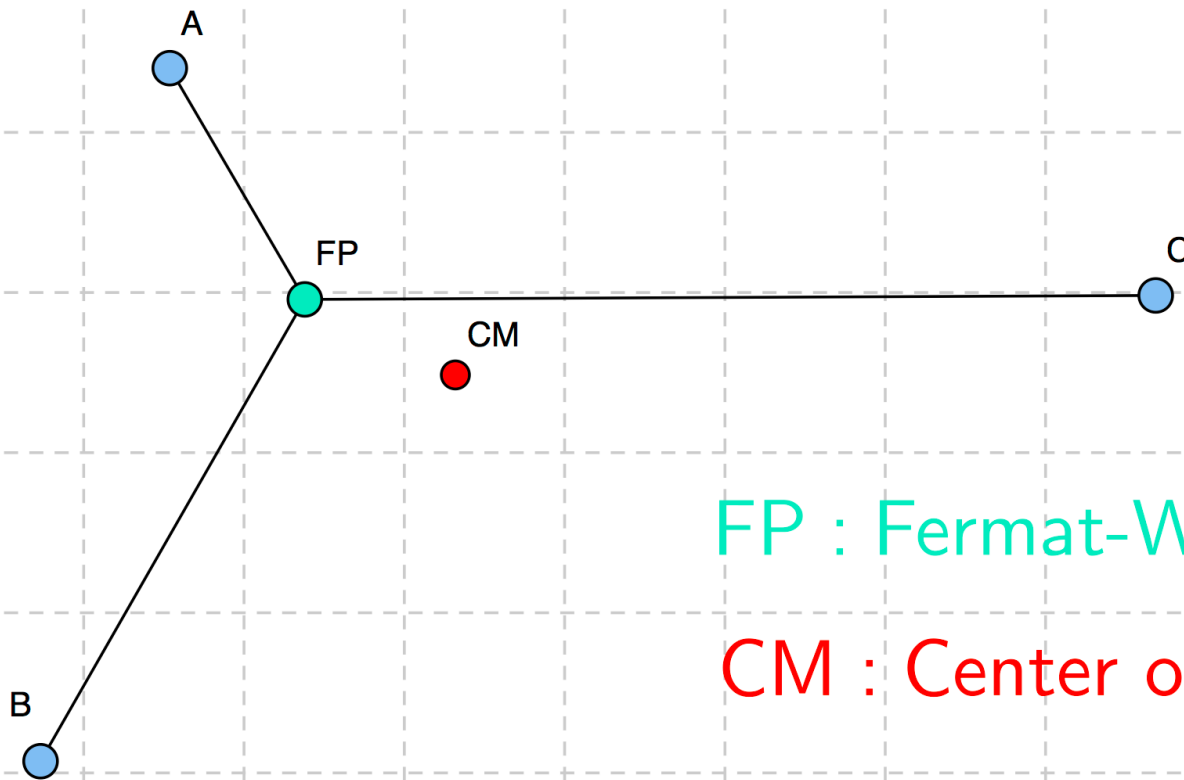
Dorrie, H. (1965). 100 Great problems of elementary mathematics. Dover Publications.

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

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

A

FP

CM

C

B

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.

I was sixteen when I became intrigued with the following problem involving N points:

Cosider N points and one more point, X. Measure the distances between X and the given points, then add the distances. Find point X so that this sum is the smallest possible.
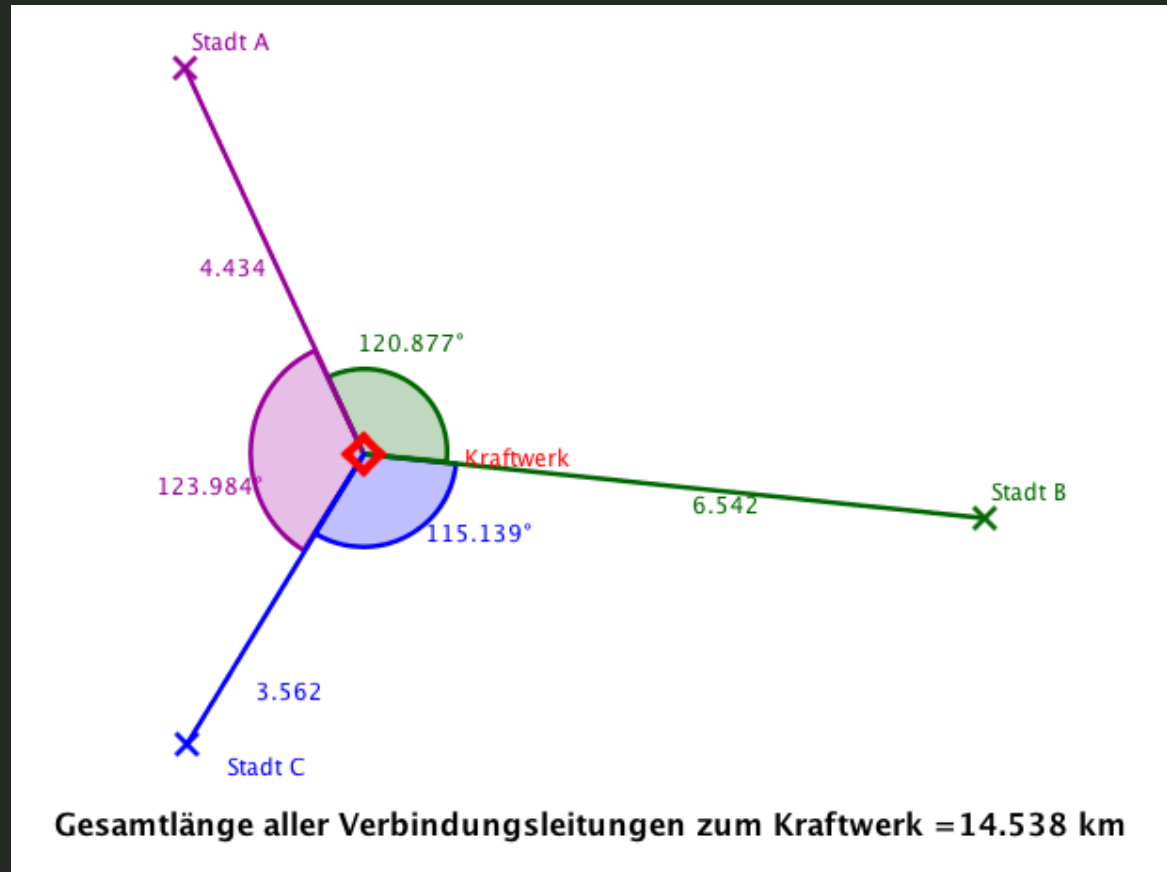
Andrew Vázsonyi

I found the point X by using an infinite, recursive algorithm, a most unusual solution for a problem in geometry. You start with a point X0, anywhere, and search for a better solution.

Andrew Vázsonyi, 1937

30 YEARS LATER
THE SAME PROBLEM IS CALLED
LOCATION THEORY

Suppose we have a bunch of factories and want to build a warehouse in a central location so the transportation costs are minimized.
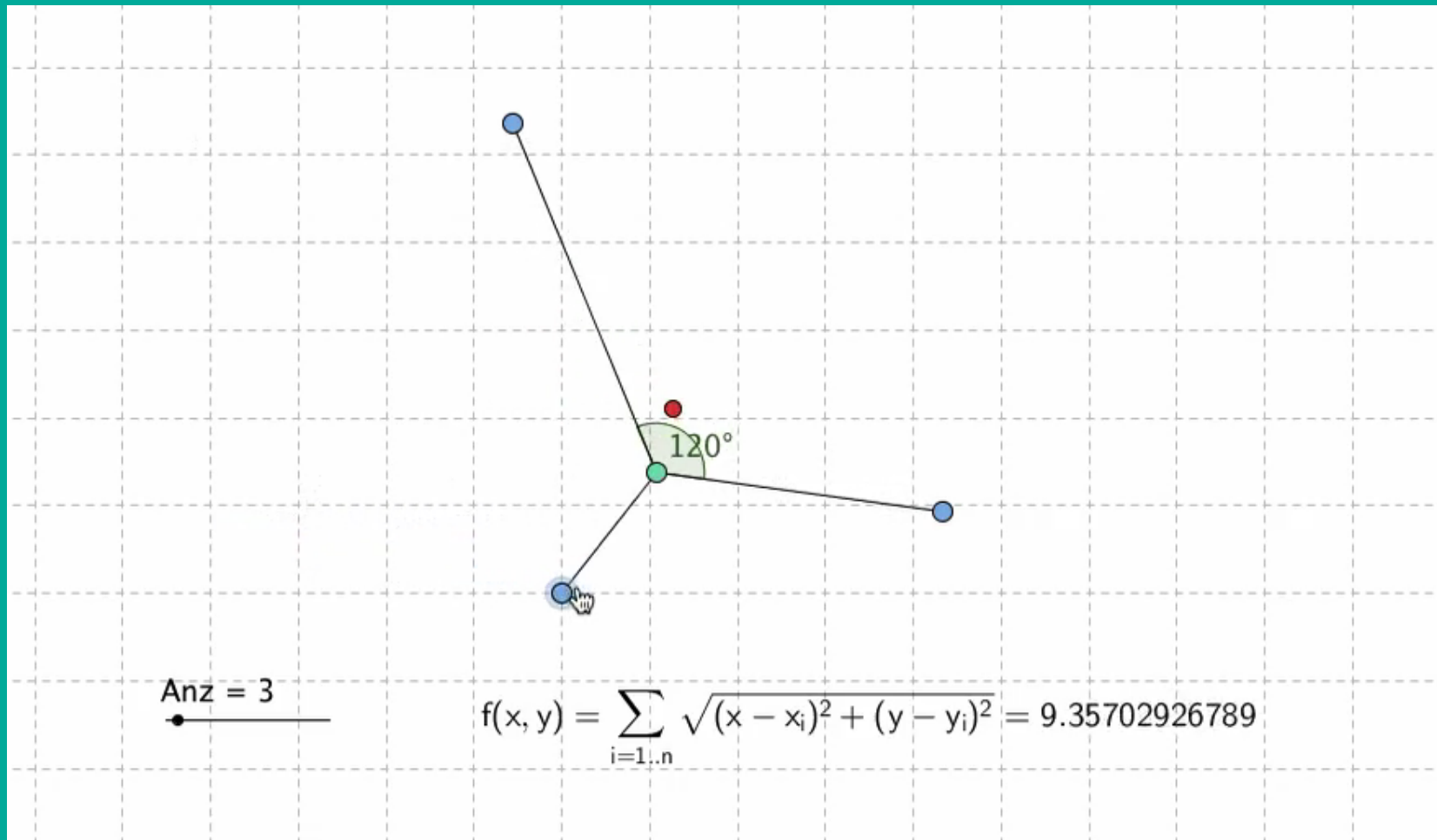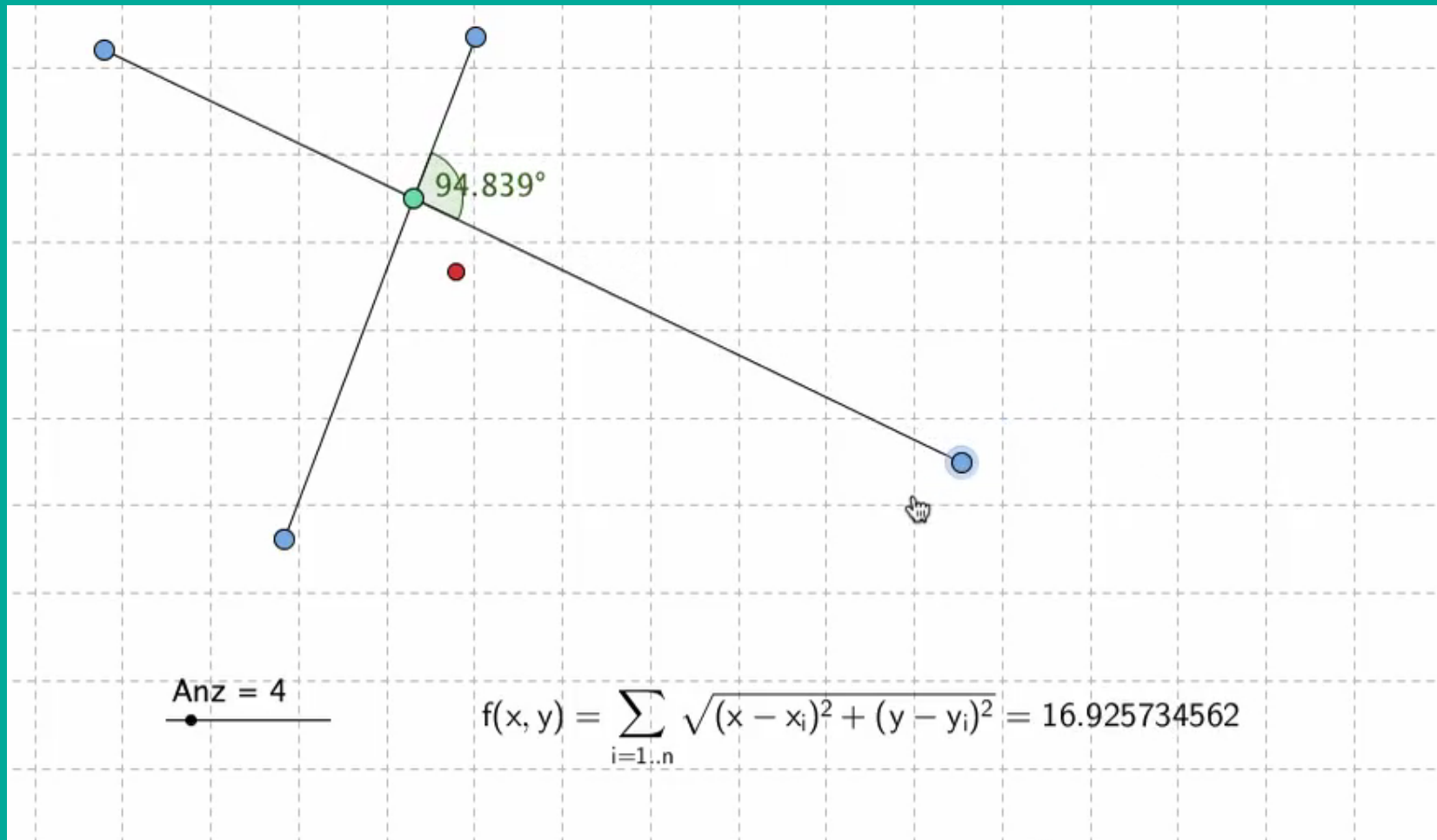
MODERN FORMULATION

Stadt A

4.434

120.877°

123.984°

Kraftwerk

Stadt B

6.542

115.139°

3.562

Stadt C

Gesamtlänge aller Verbindungsleitungen zum Kraftwerk =14.538 km

Link to GeoGebra Material

MATHEMATICAL EXPLORATION

FOR N=3 THE ANGLE 120° CAN BE OBSERVED

120°

Anz = 3

$$f(x, y) = \sum_{i=1..n} \sqrt{(x - x_i)^2 + (y - y_i)^2} = 9.35702926789$$

NO ANGLE DEPENDENCY FOR MORE THAN 3 POINTS

$94.839°$

Anz = 4

$$f(x, y) = \sum_{i=1..n} \sqrt{(x - x_i)^2 + (y - y_i)^2} = 16.925734562$$

FERMAT POINT SNAP TO INTERNAL POINT

Anz = 8

$$f(x, y) = \sum_{i=1..n} \sqrt{(x - x_i)^2 + (y - y_i)^2} = 41.3216296106$$

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

Adventures of a Real-Life Mathematician

Which Door has the Cadillac

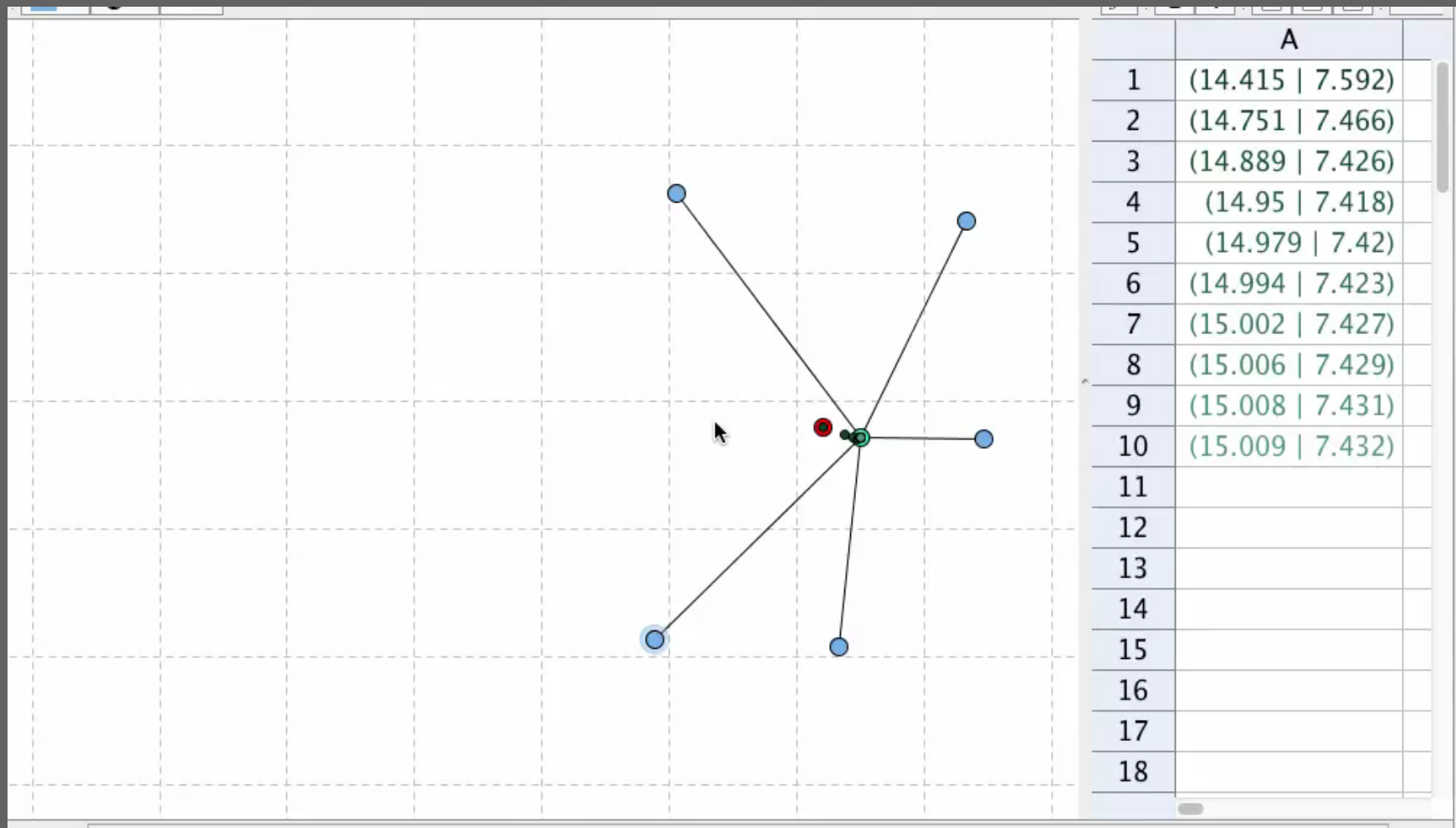DOOR #1 DOOR#2 DOOR #3

C=A+B

Andrew Vazsonyi

Computers have the potential to turn the biggest math-phobe into a math user, if not a lover.

Andrew Vázsonyi in his book:

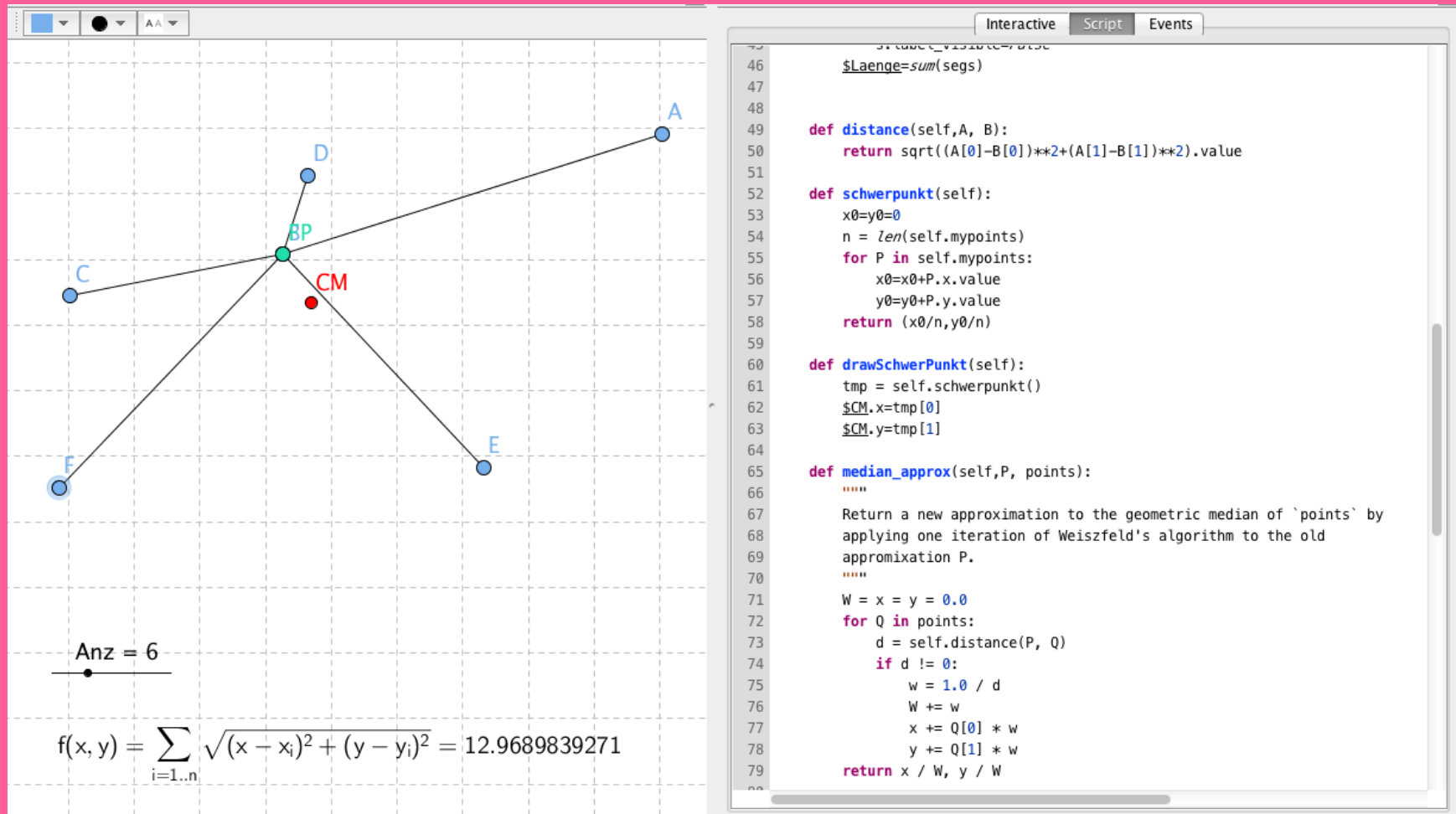Which Door has the Cadillac Adventures of a Real-Life Mathematician

CONVERGENCE OF THE WEISZFELD ALGORITHM

ALGORITHM

IMPLEMENTATION WITH THE PROGRAMMING LANGUAGE PYTHON

PROGRAMMING WITH PYTHON IN GEOGEOBRA (BETA)

$$f(x, y) = \sum_{i=1..n} \sqrt{(x - x_i)^2 + (y - y_i)^2} = 12.9689839271$$

Anz = 6

```
46      $Laenge=sum(segs)
47
48
49      def distance(self,A, B):
50          return sqrt((A[0]-B[0])**2+(A[1]-B[1])**2).value
51
52      def schwerpunkt(self):
53          x0=y0=0
54          n = len(self.mypoints)
55          for P in self.mypoints:
56              x0=x0+P.x.value
57              y0=y0+P.y.value
58          return (x0/n,y0/n)
59
60      def drawSchwerPunkt(self):
61          tmp = self.schwerpunkt()
62          $CM.x=tmp[0]
63          $CM.y=tmp[1]
64
65      def median_approx(self,P, points):
66          """
67          Return a new approximation to the geometric median of `points` by
68          applying one iteration of Weiszfeld's algorithm to the old
69          appromixation P.
70          """
71          W = x = y = 0.0
72          for Q in points:
73              d = self.distance(P, Q)
74              if d != 0:
75                  w = 1.0 / d
76                  W += w
77                  x += Q[0] * w
78                  y += Q[1] * w
79          return x / W, y / W
```

Interactive   Script   Events

WEISZFELD CLASS

```python
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
DEFINE EVENTLISTENERS

```python
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
        ...
...
```

```python
def refresh(self):
    alg.drawSchwerPunkt()
    alg.drawFermatPunkt()
```
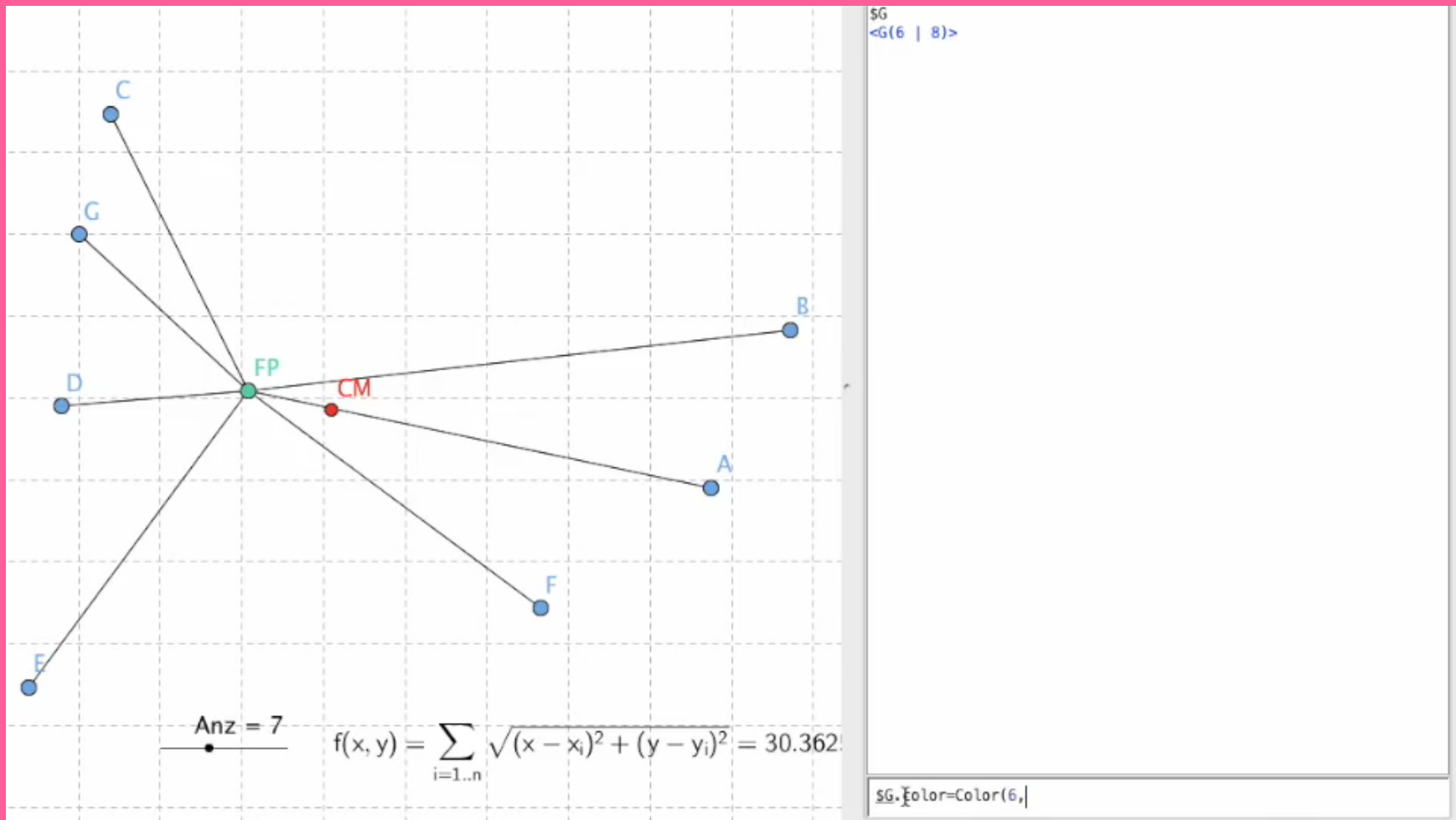
CALCULATE NEXT INTERATION POSITION OF FP

```python
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
```

MAIN LOOP

```python
def geometric_median(self,points, epsilon):
    """

    Return an approximation to the geometric median for
    `points`. Start with the centroid and apply Weiszfeld's
    algorithm until the distance between steps is less
    than `epsilon`.
    """

    n = float(len(points))
        P = tuple(sum(P[i] for P in points) / n for i in range(2))
        while True:
            Q = self.median_approx(P, points)
            if self.distance(P, Q) < epsilon:
                return Q
            P = Q
```
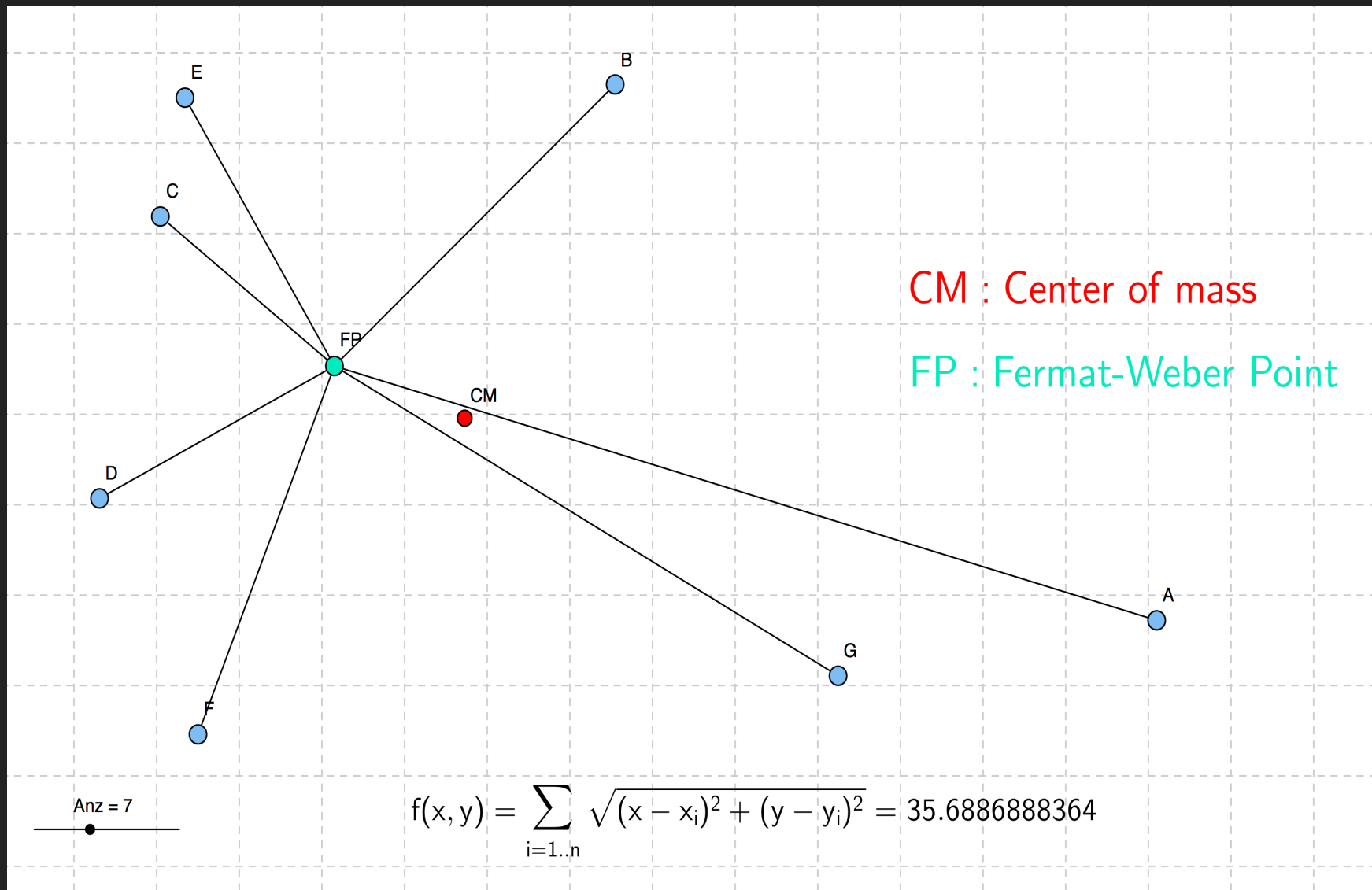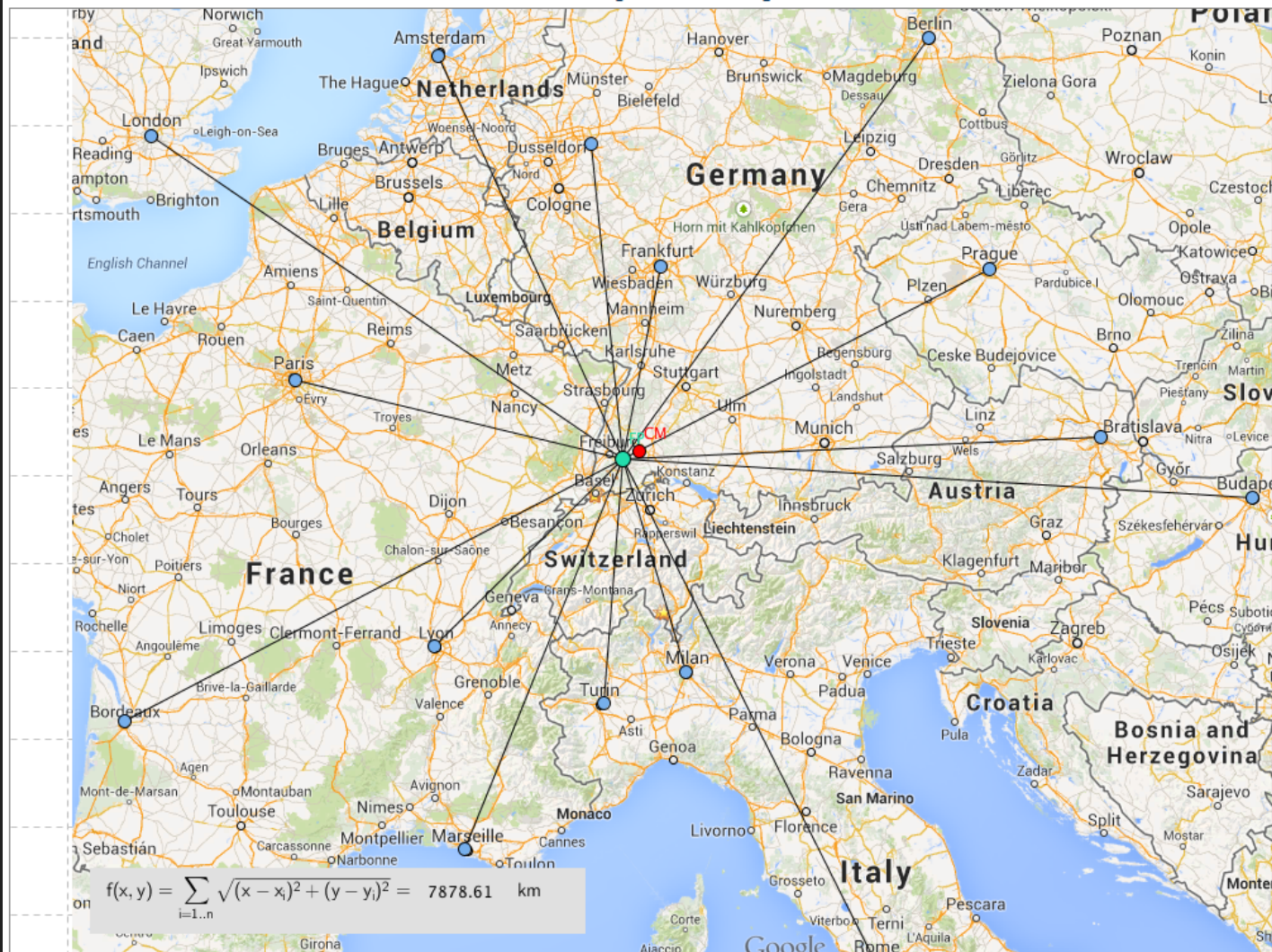
INTERACTIVE PYTHON

CONCLUSION

INTERACTIVE DEMO (JAVASCRIPT)

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

CM : Center of mass

FP : Fermat-Weber Point

Anz = 7

$$f(x, y) = \sum_{i=1..n} \sqrt{(x - x_i)^2 + (y - y_i)^2} = 35.6886888364$$

DEMO 2

Fermat Weber Point of European Cyties

$$f(x, y) = \sum_{i=1..n} \sqrt{(x - x_i)^2 + (y - y_i)^2} = \quad 7878.61 \quad \text{km}$$
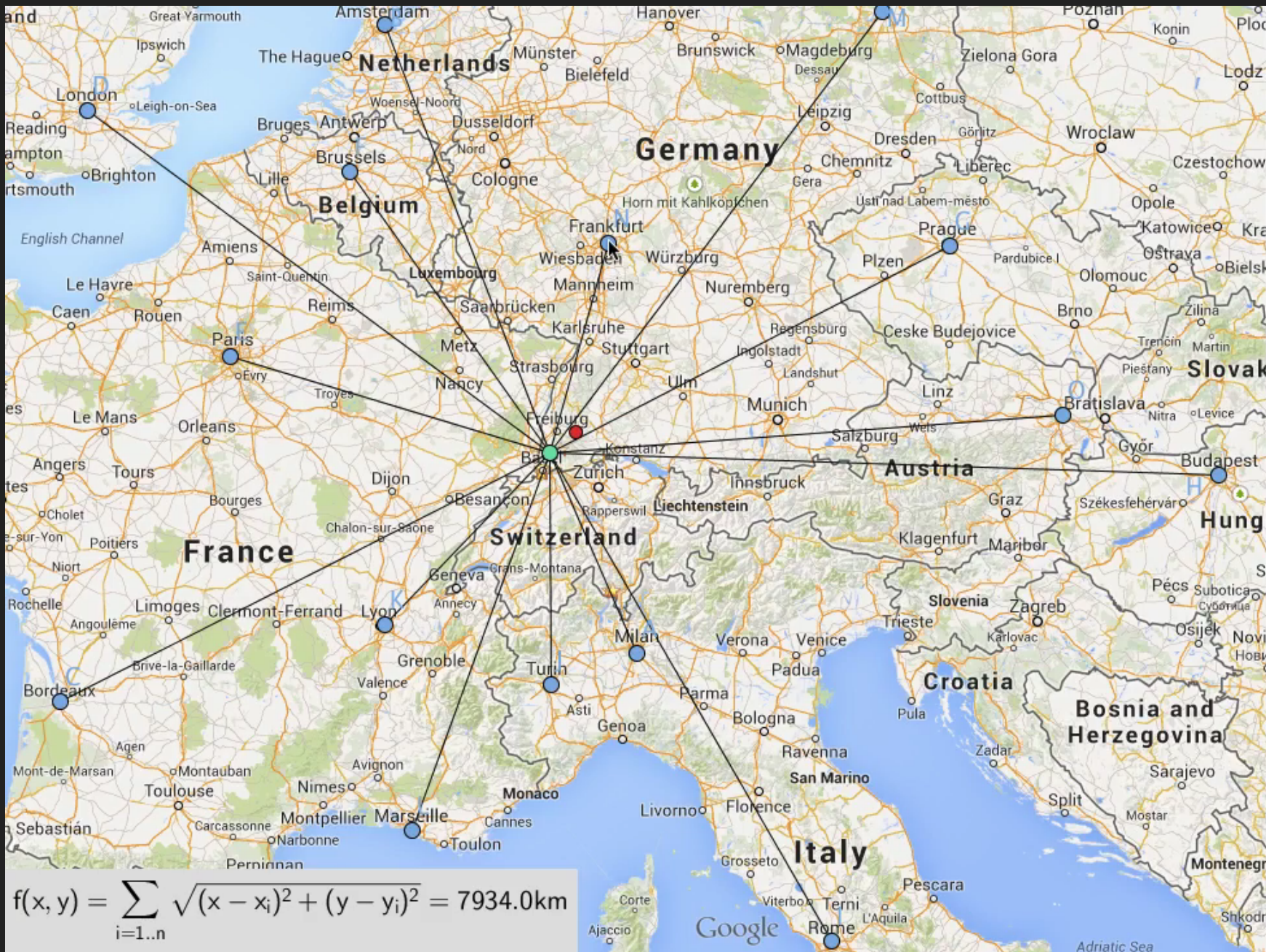
SUMMARY

- 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/

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