Triangles in the plane

Alexandra Popova and Mark Woolley

June 2022

1 Description of the Code

There are several ways one can generate a triangle, for example through defining three points or through the side lengths combined with the angles. Our goal was to create a code which can generate a triangle for any kind of input that is enough to define a triangle. So one can input a combination of the following measurements; vertices, lengths, angles or inclination, the the code will check if the input does define a triangle and otherwise will give a warning. For even more flexibility we included the option to specify the input which can be done with "arrangement".

Before actually generating the triangle the code will check if the inputs are given in the correct format and send a warning if they are not. In the case that the code gets too many input it will generate a triangle using the first few inputs and ignoring the rest and outputting a warning.

Assuming the input is valid the code would compute the following:

1. create a triangle with given set of angles/length of edges/some combinations of length and angles
2. compute the area, circumference and center of mass
3. check whether the triangle is equilateral, isosceles or a right triangle
4. check whether two triangles are congruent or similar
5. compute angle bisector and perpendicular bisector
6. compute the incircle and circumcircle
7. compute the Euler line
8. plot the triangle, center of mass, angle bisector, perpendicular bisector, incircle, circumcircle and the Euler line

How exactly the functions work is described within the commentaries in the code itself so we will not further discuss them.

2 Difficulties

There were some difficulties with finding the intersections of lines and circles. Therefore anything that relied on the existence and correctness of the intersection points wasn't working properly.

3 Solutions for the Difficulties

The problem described above occurred because the functions were first defined using the implemented function "Curves" over the real plane. So the code wasn't always able to find the solutions due to rounding mistakes.
The chosen solution was to define the geometric objects as polynomials and then solve the linear systems. This works because it enabled the code to use symbolic computation which made it possible to always find a solution.