Transactions on Engineering Sciences vol 15, © 1997 WIT Press, www.witpress.com, ISSN 1743-3533

Genericity and set processing in *Geometrica* applied to parallel and paraxial lines in architecture and optics

B.Autin

CERN, PS Division, 1211 Geneva 23, Switzerland

Abstract

The functional programming introduced by *Geometrica* in geometry is illustrated with various aspects of the concept of parallelism. Parallelism is well defined for a continuous curve. Its generalisation to a broken line or a polygon requires some convention at the vertices where the normal and the distance are undefined. The next level of generalisation consists of defining a curve whose distance to a reference curve varies according to a given law. Such a curve is called *paraxial*. Applications to architecture and optics are proposed.

1 Genericity

A property is generic for a set of elements if it applies to all the elements of the set. In *Geometrica*, the set is composed of points, straight lines, circles, arcs of circle, conics and polygonal lines. All the functions of the program are designed to be generic or at least partially generic. The function *Intersections* for instance can be applied to any couple of elements different from points. Transformations such as translation, rotation or homothety are also fully generic. Sometimes however, a concept is missing. Consider the symmetry. The symmetric of any object with respect to a point exists but does the symmetric of a point m with respect to any object exist? The answer is positive for the point and the straight line. For a polygonal line P, the image of m is no longer unique but made of the list of points symmetric of m with respect to the lines collinear to the sides of P. For a circle C, the inversion is usually considered as a generalised symmetry because the mirror circle C remains invariant in the inversion. For conics, the answer is negative in general even though the property for the foci to be images of one another is classical.

2 Set processing

As soon as one has to deal with a real problem of pure or applied geometry, one is faced with the manipulation of a large number of objects. What is called set processing is the treatment of families of objects of the same type such as sets of points, lines, circles, etc. The techniques of list processing in *Mathematica* are then very useful. Among them the quasi systematic application of the attribute *Listable* to the functions of *Geometrica* maintains the syntax identical for single elements or families of elements. In *Mathematica*, the functions which have built-in the attribute *Listable* act on a single argument and the repeated application of the operator *Map* is avoided. In *Geometrica*, the functions have most often at least two arguments, this is inherent to the two dimensions of the plane. Then, *Listable* permits the automatic coupling of the arguments of same position by the operator of the function and its action is related to the application of the operator *Thread*.

3 Parallelism

A curve C' parallel to a curve C is described by the end m' of a segment $m\,m'$ normal to C in m. The segment may have any length and be oriented towards the interior or the exterior of C. Parallelism is commutative: if C' is built parallel to C, then C can be built from C' using a segment of same length but opposite direction. The parallel to a straight line or a circle is also a straight line or a circle but, in the other cases and especially for conics, parallel curves are of different natures. The very definition of a parallel assumes that a normal exists, which is not the case at the vertices of polygonal lines. However, the concept of parallel lines is necessary each time thick objects have to be represented. First, one notes that the definition is valid for a segment. For two consecutive segments, the parallel line is defined in *Geometrica* as the union of the segments t_1, t_2 parallel to s_1 and s_2 and of the extra segments which join the end of t_1 and the beginning of t_2 to the intersection x of the lines collinear to t_1 and t_2.

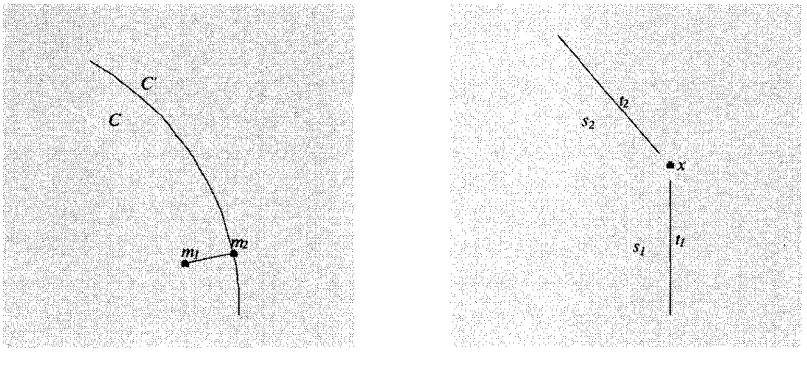


Figure 1: General parallel curves and parallel polygonal lines.

3.1 Vertices of the parallel line

The construction of the points x gives the flavour of the techniques of set processing. The reference line has say four vertices but the exact number does not matter and could be arbitrarily large.

```
In[1]:=
m=POINT[{0,1,2,1},{0,0,1,3/2}];
l=LINE@@m
```

```
Out[2]=
LINE[POINT[0, 0], POINT[1, 0],
     POINT[2, 1], POINT[1, 3/2]]
```

The function *Ruler* returns the list of the lines collinear to s_1, s_2, ...

```
In[3]:=
d=Ruler[1]
```

```
Out[3]=
{SIRAIGHT[0, 1, 0], SIRAIGHT[-1, 1, 1],
 SIRAIGHT[-1/2, -1, 2]}
```

The parallels to these lines at the distance .5 are grouped in pairs with an overlap of one element.

```
In[4]:=
d1=Parallel[d,.2];
d2=Partition[d1,2,1]
```

```
Out[5]=
{{SIRAIGHT[0., -1, -0.2],
  SIRAIGHT[0.7071, -0.7071, -0.9071]},
 {SIRAIGHT[0.7071, -0.7071, -0.9071],
  SIRAIGHT[0.4472, 0.8944, -1.9889]}}
```

The intersections are found by replacing the head *List* by the head *Intersections* at the second level of the expression.

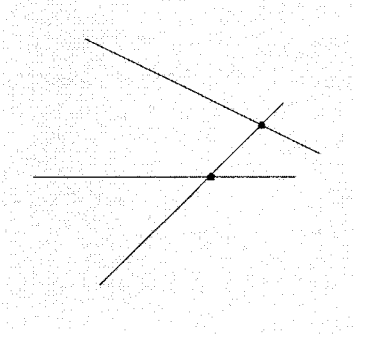
```
In[6]:=
newhead=Intersections@@#&;
x=newhead/@d2
```

```
Out[7]=
{POINT[1.0828, -0.2],
 POINT[2.3376, 1.0548]}
```

The graph of the various elements is produced with *Draw*.

```
In[8]:=
Draw[White,1, Black,d1,x]
```

```
Out[8]=
```



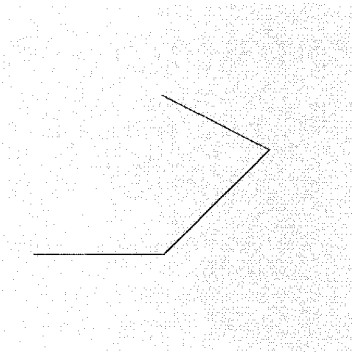
The user does not have to care about the intermediate constructions, *Geometrica* produces the parallel line with the function *Parallel*.

```
In[9]:=
l1=Parallel[l,.2]
```

```
Out[9]=
LINE[POINT[0, 0.2],
     POINT[0.91716, 0.2],
     POINT[1.6624, 0.9452],
     POINT[0.91056, 1.3211]]
```

```
In[10]:=
Draw[White,l, Black,l1]
```

```
Out[10]=
```

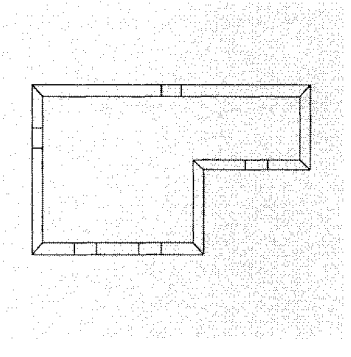


3.2 Ribbons

The space between the parallel lines can be covered with adjacent quadrangles whose vertices are those of two parallel segments, in the extended sense defined above. This is the purpose of the option *Ribbon*. Here too, the generation of the polygons is an example of set processing but it will not be described in detail, the result is only given. The example is chosen in the field of architecture. The ground floor of a house is supposed to be made of a wall of uniform thickness with several openings for doors and windows.

```
In[1]:=
m= POINT[{0 , 2 , 3 , 5, 6, 8, 8, 10,
          11, 13, 13, 7, 6, 0, 0, 0},
         {0, 0, 0, 0, 0, 0, 4, 4,
          4, 4, 8, 8, 8, 8, 6, 5}];
p= POLYGON@@m;
p1= Parallel[p, .5, Ribbon-> True]
Draw[p1];
```

```
Out[4]=
```



The openings can be distinguished from the rest of the wall by noting that they all have the same area .5.

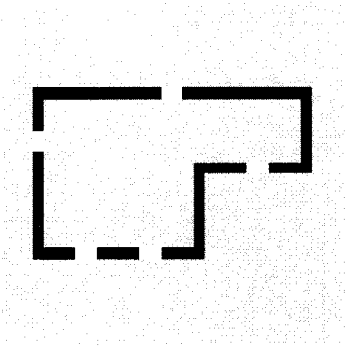
```
In[5]:=
a=Area[p1];
pos=Position[a,.5]
```

```
Out[6]=
{{2},{4},{8},{12},{15}}
```

The polygons are then painted in black at the exception of the openings which are painted in white. A list (*col*) is first created with identical elements (*Black*), then updated with *ReplacePart* to include the *White* elements at the right position.

```
In[7]:=
n=Length[p1];
col=Table[Black,{i,n}];
col=ReplacePart[col,White,pos];
Draw[Paint[p1,col]];
```

```
Out[10]=
```



4 Paraxial curves

The parallel lines are generated with a segment of constant length. It is conceivable to make this length vary according to a prescribed law, the line generated this way is called *paraxial*. This concept is borrowed from optics but it will first be studied from a purely geometric view point.

4.1 The function *Paraxial*

The syntax is the same for the functions *Paraxial* and *Parallel* but the distance *d* is either an operator or a pure function *f*. Moreover an origin corresponding to *f[0]* is needed for the geometric object. The first example concerns a segment *l*. The pure function is associated with the function $y = x$ and acts on the parameter of a point of the segment. The parameter takes the values 0 and 1 at the origin and at the end. The result is a pure point of the paraxial segment.

```
In[1]:=
{m1,m2}=POINT[0,{0,2}];
l=LINE[m1,m2];
mp1=Paraxial[l,#&]
```

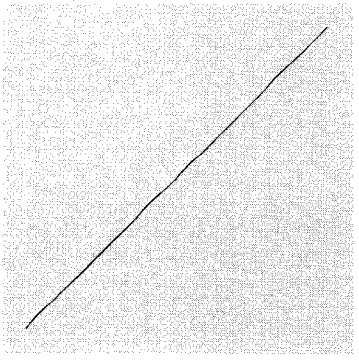
```
Out[2]=
POINT[#1, 2 #1]&
```

If the pure function had to act on the distance from the point of *l* to the origin of *l*, then the pure function should be redefined.

```
In[4]:=
d=Distance[l];
mp2=Paraxial[l,d*#&]
In[6]:=
Draw[mp2,White,l];
```

```
Out[5]=
POINT[2 #1, 2 #1]&

Out[7]=
```



In the second example, a sine curve is drawn around a circle for three different amplitudes to show how the shape of the paraxial line may vary for the same reference curve.

```
In[1]:=
{m1,m2}=POINT[{0,1},0];
c=ARC[m1,m2,2π];
mp=Paraxial[c,
            {0.1*Sin[#*8π]&,
                 Sin[#*8π]&,
               2*Sin[#*8π]&];
g=Draw[#,White,c,
       PlotRange->All]&;
g/@mp
```

```
Out[5]:=

see Figure 2
```

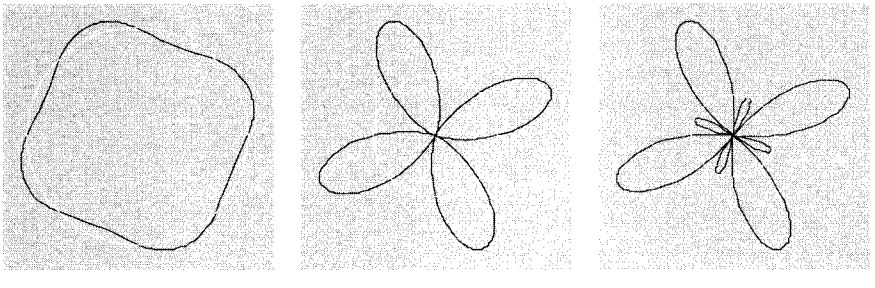


Figure 2: Sinusoidal paraxial curves of a circle for three different amplitudes.

4.2 Application to optics

The shape of a beam is determined by the trajectory T of the centre of gravity and by the motion of particles which slightly deviate in position, angle and energy (or wavelength) about the centre of gravity.

4.2.1 Reference trajectory In light optics, a ray is a broken line whose vertices occur each time there is a change of the refraction index. In charged particle optics, the rays are straight in drift spaces only, arcs of circle inside constant field magnets and arcs of sinusoids or catenaries in focusing or defocusing elements.

4.2.2 Beam envelope The geometric dispersion is characterised by the beam *emittance* which is the area of an ellipse on which the positions x and the angle x' of the extreme rays are located. The set of all the values of x and x' is the *phase space*. At a given curvilinear distance on T, all the rays which intersect the plane P normal to T are represented by a point in the phase-space attached to P, this local phase space is a Poincaré section (Figure 3).

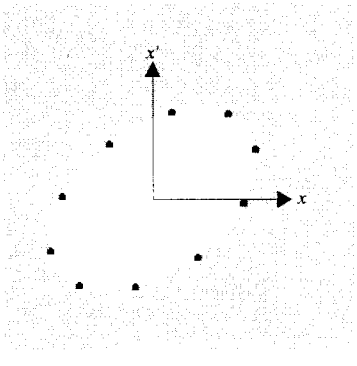


Figure 3: Poincaré section of a particle beam.

The shape of the beam indeed varies along T and so do the phase-space ellipses but their area remains constant. In other words, the emittance is an invariant of the motion. The theory [1] shows that the beam envelope is of the same nature as the trajectories in focusing elements and parabolic in drift spaces.

4.2.3 Beam shape A complete optical channel is thus defined by two lists; one contains the segments and arcs of circle of the reference trajectory T, the other contains the envelope functions. The beam shape is obtained by coupling the two lists with the operator *Paraxial*, the coupling being automatic when the symbol *Paraxial* has the attribute *Listable*. An example of realistic beam representation in a particle accelerator is shown in Figure 4. This information is of primary importance for the design of the vacuum vessel inside which the particles circulate.

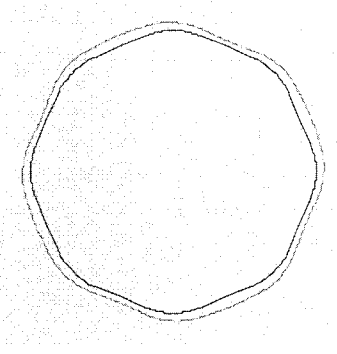


Figure 4: Typical beam shape in a particle accelerator.

5 Conclusion

It has been shown how a geometry program based on a symbolic code like *Mathematica* can deal with problems which are outside the realm of a mouse driven geometry package or of a numerical code. The symbolic aspect appears at the level of the theoretical analysis of a problem and in the use of pure functions. Another feature related to the programming language of *Mathematica* lies in the set processing with which as heavy information as the one needed to describe the shape of a beam in a high energy physics machine can be treated in a surprisingly concise style.

References

1. E.D. Courant and H.S. Snyder, Theory of the alternating gradient synchrotron, *Annals of Physics*, 1958, **3**, 1-48.