Motion Polynomials and Planar Linkages

C. Koutschan
(Johann Radon Institute for Computational and Applied Mathematics, Österreichische Akademie der Wissenschaften)

christoph.koutschan@ricam.oeaw.ac.at

Introduction

We describe an application of computer algebra to the construction of mechanisms with certain prescribed properties. In the CAS Mathematica, we have implemented the package **PlanarLinkages**; it provides commands for constructing and visualizing planar linkages that draw a prescribed algebraic curve. The construction procedure is based on so-called motion polynomials; their basic arithmetic and a factorization algorithm is also provided by the package.

In order to state the problem more precisely, let us introduce some terminology. A *linkage* is a mechanical device consisting of rigid bodies (called *links*) that are connected by *joints*. We restrict our attention to *planar linkages*, i.e., to linkages all of whose links move in parallel planes. Moreover, we consider only *rotational joints*, which means that we don't allow *prismatic joints*. In Figure 1 two examples for this type of linkages can be seen: the first one has four degrees of freedom, while the second one has only a single degree of freedom (we say it has *mobility one*). If we move a linkage of mobility one, the trace of any point located on one of the links yields a bounded curve in the plane.

The problem of constructing a planar linkage that draws a finite segment of a given algebraic curve was first addressed and solved in full generality by Kempe [2]. While his construction is very elegant in theory, it yields quite complicated linkages in practice; see [3] for an implementation. In a recent article [1] the symbolic computation group at RICAM, including the author, designed a novel algorithm for basically the same problem. The advantage of the new algorithm is that it yields much simpler linkages: the number of links and joints is only linear in the degree of the curve. Moreover, it allows for a simple collision detection, which for general linkages is a very hard problem. The drawback of our method is that it is only applicable to bounded rational curves,

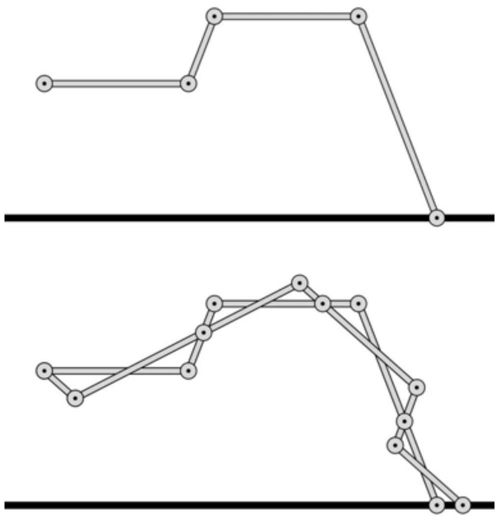


Figure 1: *An open chain linkage and its extension to a linkage of mobility one, both realizing the translational motion given by $P(t)$ in Equation* (4).

i.e., to curves that are parametrizable by rational functions and that are contained in some disk of finite radius.

Theoretical Background

Before we describe our Mathematica package, we sketch the new method for constructing planar linkages and give a bit of theoretical background. The interested reader is referred to the paper [1] where all this is laid out in detail, and from which also the following simple example is taken: we consider the ellipse that is implicitly defined by the polynomial $(x+1)^2+4y^2=1$. The goal is to construct a linkage with rotational joints that draws this ellipse and that admits only one degree of freedom. More precisely, "drawing" means that there is a specific link (to which we attach the pen) that performs a motion along the ellipse while the linkage moves.

Mathematically speaking, a *motion* is a one-dimensional family of direct isometries (i.e., translations and rotations). We denote by SE_2 the special Euclidean group, which is the set of direct isometries in the plane with composition as the group operation. For a convenient treatment in a computer algebra system, we encode direct isometries as elements of the noncommutative $\mathbb{R}$-algebra $\mathbb{K}$ of *dual complex numbers*:

$$\mathbb{K} = \mathbb{C}[\eta] \,/\, (\eta^2, i\eta + \eta i).$$

Its elements are of the form $z + \eta w$ with complex numbers $z, w \in \mathbb{C}$, and according to the defining relations, which can be seen as rewriting rules, they are multiplied as follows:

$$(z_1 + \eta w_1) \cdot (z_2 + \eta w_2) = z_1 z_2 + \eta\,(\overline{z_1} w_2 + z_2 w_1). \quad (1)$$

By defining on $\mathbb{K}$ the equivalence relation

$$k_1 \sim k_2 \;:\Longleftrightarrow\; k_1 = \alpha k_2 \text{ for some } \alpha \in \mathbb{R} \setminus \{0\}, \quad (2)$$

we can show that the multiplicative group

$$\{z + \eta w \in \mathbb{K} \mid z \neq 0\} \,/\!\sim$$

is isomorphic to SE_2; in Out[8] below the isomorphism is given explicitly. A univariate polynomial in $\mathbb{K}[t]$ gives rise to a one-dimensional family of direct isometries and is therefore called a *motion polynomial*. Motions that can be represented in this way are called *rational motions*. The algorithm we are going to describe takes as input a motion polynomial and outputs a planar linkage of mobility one realizing the corresponding rational motion. This task is slightly more general than drawing a rational curve, since also the orientation of the end effector is taken into account.

The ellipse $(x + 1)^2 + 4y^2 = 1$ admits the rational parametrization

$$\varphi(t) = \Big(-\frac{2}{t^2+1}, \frac{t}{t^2+1}\Big), \quad t \in \mathbb{R} \cup \{\infty\}, \quad (3)$$

from which one can read off that a translational motion along this ellipse is represented by the motion polynomial

$$P(t) = (t^2 + 1) + \eta\,(it - 2) \quad (4)$$

("translational" means that the orbit of *any* point under this motion is a translate of the ellipse). A motion polyomial $Z + \eta W \in \mathbb{K}[t]$ is called *bounded* if the complex polynomial $Z \in \mathbb{C}[t]$ does not have any real roots; the connection to the boundedness of the corresponding curve (the orbit of the origin) is established by the fact that Z appears as the denominator of the parametrization.

In order to construct a linkage that realizes the motion $P(t)$, we want to decompose it into simpler motions, namely into revolutions; these correspond exactly to motions that can be realized by a single (rotational) joint. We find [1, Lemma 4.3] that each linear motion polynomial, whose orbits are bounded, represents a revolute motion. Therefore, the desired decomposition is obtained by a factorization of P into linear polynomials.

In our example, however, one can easily check (e.g., by an ansatz with undetermined coefficients) that such a factorization does not exist. But this doesn't mean that we have to give up on drawing the ellipse! Recall that by the definition (2) of the equivalence relation $\sim$, the motion polynomial $RP \in \mathbb{K}[t]$ describes the same motion as P for any real polynomial $R \in \mathbb{R}[t]$. In this case we can take $R = t^2 + 1$ and observe that

$$R(t) \cdot P(t) = \big(t^4 + 2t^2 + 1\big) + \eta\,\big(it^3 - 2t^2 + it - 2\big)$$

indeed admits a factorization into linear polynomials:

$$\big(t + i - \eta\, i\big) \cdot \big(t - i + \tfrac{1}{2}\eta\, i\big) \cdot \big(t - i + \tfrac{3}{2}\eta\, i\big) \cdot (t + i). \quad (5)$$

In [1, Theorem 5.15] it is shown that for any bounded motion polynomial P such a real polynomial R exists, and an algorithm to compute R and the complete factorization of RP is described.

The factorization (5) allows us to construct a linkage, in the form of an open chain (upper part of Figure 1), whose links can move according to the revolutions represented by the linear factors. Since such a linkage has many degrees of freedom, we need to constrain its mobility. This is done by adding more links and joints (lower part of Figure 1), which is achieved by an iteration of the so-called flip procedure [1, Sections 6–7].

However, if we just want to draw the ellipse, we need not realize exactly the translational motion $P(t)$: it is enough to find a motion for which the orbit of one point is the ellipse. One can check that multiplying with a polynomial $C \in \mathbb{C}[t]$ from the left does not change the orbit of the origin. In our case, we find that the polynomial CP with $C(t) = t - i$ factors completely:

$$C(t) \cdot P(t) = \big(t - i - \tfrac{1}{2}\eta\, i\big) \cdot \big(t - i + \tfrac{1}{2}\eta\, i\big) \cdot \big(t + i + \eta\, i\big).$$

This factorization gives rise to a slightly simpler construction, which is depicted in Figure 2.

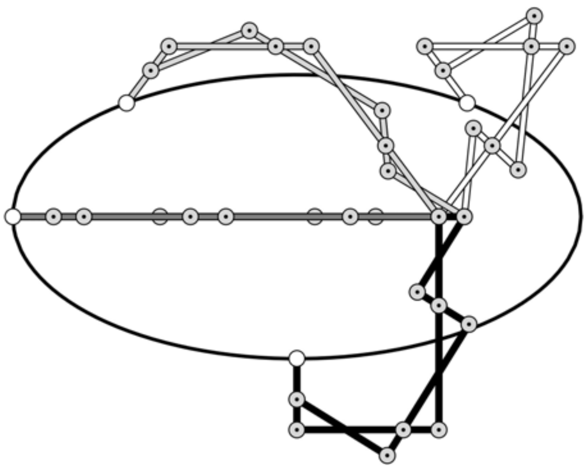


Figure 2: *A linkage drawing the ellipse* (3)*; it is shown in different positions: $t = 2$ (white), $t = \frac{1}{2}$ (light gray), $t = 0$ (dark gray), and $t = -1$ (black).*

The Mathematica Package

We now give a brief demonstration of our Mathematica package **PlanarLinkages**. The package, its source code, and a Mathematica notebook with some sample computations are freely available [4].

In[1]:= $<<$ **PlanarLinkages.m**

Motion polynomials are entered using the special symbol **eta** and Mathematica's **NonCommutativeMultiply** (written as ******). As output we obtain a pretty-printed version of the motion polynomial, which internally is represented as a Mathematica expression with head **MP**.

In[2]:= $\boldsymbol{P = t + \mathrm{I} + \mathrm{eta} ** (2 - \mathrm{I})}$

Out[2]= $(\mathbb{i} + t) + \eta \cdot (2 - \mathbb{i})$

In[3]:= **FullForm**$[\boldsymbol{P}]$

Out[3]= MP[Plus[Complex[0, 1], t], Complex[2, -1]]

Arithmetic can be done in the usual way, by taking into account the noncommutative multiplication.

In[4]:= $\boldsymbol{P + 1 + \mathrm{eta} ** \mathrm{I}}$

Out[4]= $((1 + \mathbb{i}) + t) + \eta \cdot 2$

In[5]:= $\boldsymbol{P ** (1 - \mathrm{eta}) ** P}$

Out[5]= $\left(-1 + 2\,\mathbb{i}\,t + t^2\right) + \eta \cdot \left(-1 + (4 - 2\,\mathbb{i})\,t - t^2\right)$

When executing the multiplication symbolically, we recover Equation (1):

In[6]:= $\mathbf{MP}[z_1, w_1] ** \mathbf{MP}[z_2, w_2]$

Out[6]= $z_1\,z_2 + \eta \cdot (\mathrm{Conjugate}[z_1]\,w_2 + w_1\,z_2)$

The command **ActR2** performs the action of an element $(x_1 + ix_2) + \eta(y_1 + iy_2) \in \mathbb{K}$, which itself represents a direct isometry in SE_2, on a point $(a, b) \in \mathbb{R}^2$, see [1, (4.2)].

In[7]:= $\mathbf{ActR2}[(x_1 + \mathrm{I}\,x_2) + \mathbf{eta} ** (y_1 + \mathrm{I}\,y_2),\ \{a, b\}];$

In[8]:= $\mathbf{Simplify}[\%,\ \mathbf{Element}[\{x_1, x_2, y_1, y_2\},\ \mathbf{Reals}]]$

Out[8]= $$\left\{\frac{a\,x_1^2 - a\,x_2^2 - 2\,b\,x_2\,x_1 + x_1\,y_1 - x_2\,y_2}{x_1^2 + x_2^2},\ \frac{2\,a\,x_2\,x_1 + b\,x_1^2 - b\,x_2^2 + x_1\,y_2 + x_2\,y_1}{x_1^2 + x_2^2}\right\}$$

The command **AnimateMP** visualizes the action of a motion polynomial; as a result we obtain an animation (not printed here!) showing a small triangle that moves according to the given motion. Since a linear bounded motion polynomial corresponds to a revolute motion, we can compute its fixed point. From the output, it becomes clear that the fixed point can only be given if the input polynomial is bounded, i.e., if the polynomial $t + z \in \mathbb{C}[t]$ has no real roots. In contrast, the motion polynomial $t + 1 + \eta$ corresponds to an unbounded motion, in this case a horizontal translation, and the attempt to compute its fixed point results in an error message.

In[9]:= $\mathbf{FixPoint}[t + z + \mathbf{eta} ** (2w)]$

Out[9]= $\left\{-\frac{\mathrm{Im}(w)}{\mathrm{Im}(z)}, \frac{\mathrm{Re}(w)}{\mathrm{Im}(z)}\right\}$

In[10]:= $\mathbf{Catch}[\mathbf{FixPoint}[t + 1 + \mathbf{eta}]]$

Out[10]= FixPoint: Input is not a normed bounded motion polynomial of degree 1.

Next, we provide a command to compute a factorization of a motion polynomial into linear factors; as a consistency check, we expand the result and obtain the original polynomial back.

In[11]:= $\mathbf{FactorMP}[(t + \mathrm{I})\hat{\ }5 + \mathbf{eta} ** t]$

Out[11]= $$\left((\mathbb{i} + t) + \eta \cdot \frac{\mathbb{i}}{16}\right) \cdot \left((\mathbb{i} + t) - \eta \cdot \frac{\mathbb{i}}{8}\right) \cdot ((\mathbb{i} + t) + \eta \cdot 0) \cdot \left((\mathbb{i} + t) + \eta \cdot \frac{\mathbb{i}}{8}\right) \cdot \left((\mathbb{i} + t) - \eta \cdot \frac{\mathbb{i}}{16}\right)$$

In[12]:= $\mathbf{Expand}[\%]$

Out[12]= $\left(t^5 + 5\,\mathbb{i}\,t^4 - 10\,t^3 - 10\,\mathbb{i}\,t^2 + 5\,t + \mathbb{i}\right) + \eta \cdot t$

If the polynomial itself cannot be factored, then the command automatically determines a minimal-degree real polynomial such that the product of the two polynomials factors completely.

In[13]:= $\mathbf{fact} = \mathbf{FactorMP}[t\hat{\ }2 + 1 + \mathbf{eta} ** (\mathrm{I}\,t - 2)]$

FactorMP::R : Multiply the input with $R = 1 + t^2$

Out[13]= $$\left((\mathbb{i} + t) + \eta \cdot \left(\mathrm{C}[2] - \frac{\mathbb{i}}{2}\right)\right) \cdot ((-\mathbb{i} + t) - \eta \cdot \mathrm{C}[2]) \cdot \left((-\mathbb{i} + t) + \eta \cdot \left(\mathrm{C}[1] + \frac{3\mathbb{i}}{2}\right)\right) \cdot ((\mathbb{i} + t) - \eta \cdot \mathrm{C}[1])$$

This factorization can now be used to construct a linkage, by calling the command **ConstructLinkage**. For this purpose we instantiate the free parameters C[1] and C[2], and give a "random" polynomial **rand** as second argument according to [1, Lemma 7.5]. While almost any choice of C[1], C[2], and **rand** yield a valid linkage, we can play with these parameters to influence the shape of the resulting linkage. For example, we can omit the second argument, in which case the program chooses a random polynomial, but the linkage then will usually look very "ugly", in the sense that some links are much longer than others. The output has to be understood as follows: each triple $\{i, j, p\}$ stands for "link i is connected to link j by a joint and their relative motion is given by the motion polynomial p", where the links are labeled with integers from 1 to 10.

In[14]:= $\mathbf{fact} = \mathbf{fact}\ /.\ \{\mathrm{C}[1] \to 0, \mathrm{C}[2] \to -\mathrm{I}/2\};$

In[15]:= $\mathbf{rand} = t + (9/5)\,\mathrm{I} + \mathbf{eta} ** 0;$

In[16]:= $\mathbf{L} = \mathbf{ConstructLinkage}[\mathbf{fact},\ \mathbf{rand}]$

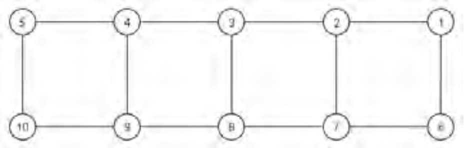
Out[16]= $$\left\{\left\{1, 2, (\mathbb{i} + t) + \eta \cdot 0\right\}, \left\{2, 3, (t - \mathbb{i}) + \eta \cdot \tfrac{3\mathbb{i}}{2}\right\}, \left\{3, 4, (t - \mathbb{i}) + \eta \cdot \tfrac{\mathbb{i}}{2}\right\}, \left\{4, 5, (\mathbb{i} + t) - \eta \cdot \mathbb{i}\right\},\right.$$

$$\{6,7,(\mathbb{i}+t)-\eta\cdot\tfrac{45\mathbb{i}}{56}\},\{7,8,(t-\mathbb{i})+\eta\cdot\tfrac{3\mathbb{i}}{8}\},$$
$$\{8,9,(t-\mathbb{i})+\eta\cdot\tfrac{41\mathbb{i}}{28}\},\{9,10,(\mathbb{i}+t)+\eta\cdot\tfrac{2\mathbb{i}}{7}\},$$
$$\{1,6,(t+\tfrac{9\mathbb{i}}{5})-\eta\cdot\tfrac{9\mathbb{i}}{28}\},\{2,7,(t+\tfrac{9\mathbb{i}}{5})-\eta\cdot\tfrac{9\mathbb{i}}{8}\},$$
$$\{3,8,(t+\tfrac{9\mathbb{i}}{5})-\eta\cdot\tfrac{9\mathbb{i}}{4}\},\{4,9,(t+\tfrac{9\mathbb{i}}{5})-\eta\cdot\tfrac{9\mathbb{i}}{7}\},$$
$$\{5,10,(t+\tfrac{9\mathbb{i}}{5})+\eta\cdot 0\}\}$$

Still, this mathematical description of a linkage is not very intuitive. To get an idea of how this linkage looks like and how it draws the ellipse, our package provides the command **ShowLinkage**, which offers a large variety of ways to visualize and animate linkages. For example, we can draw the *link graph*, which is a graph whose vertices correspond to the links and whose edges correspond to the joints of the linkage.

```
In[17]:= ShowLinkage[L, Return → "graph"]
```

Out[17]=



While the link graph gives information about the topological structure of the linkage, it completely hides its geometry. To see how a physical model of the linkage would look like, one can use the same command with different values of the **Return** option. This way one can get a two-dimensional picture (similar to the one in Figure 2) or the corresponding animation. By specifying **Return → "picture3D"** one obtains a three-dimensional drawing of the linkage as in Figure 3. Such a 3D graphics can be animated as well; we refer to our website [4] for some sample movies. Figure 3 also shows that the depicted linkage cannot move without self-collisions. Thanks to the special structure of the linkages constructed by our algorithm — the link graph will always be ladder-shaped as in Out[17] — these collisions can be detected by solving a relatively simple system of polynomial equations. The following computation shows that when we choose a certain spatial arrangement of the links, then there are only few collisions and all happen at the same position $t = \infty$. This is the position when the pen passes through the origin (the right-most point of the ellipse) and when all joints are located on the horizontal axis (compare Figure 2).

```
In[18]:= ord = {2, 1, 6, 7, 3, 8, 4, 5, 10, 9};
In[19]:= ShowLinkage[L, Links → ord,
            Return → "collisions"]
```

Out[19]= $\{\{\{2,6,7\},\tfrac{2}{7},\infty\},\{\{3,8,4\},\tfrac{54}{61},\infty\},$
$\{\{3,8,4\},\tfrac{6}{7},\infty\},\{\{4,5,9\},\tfrac{5}{7},\infty\}\}$

We conclude with an example that is motivated by a popular formulation of Kempe's theorem, stating that *"There is a linkage that signs your name"*, which is attributed to William Thurston. However, as remarked by O'Rourke [5], it is not very plausible that a concrete "signing linkage" has ever been constructed by Kempe's procedure due to its complexity, in terms of links and joints.

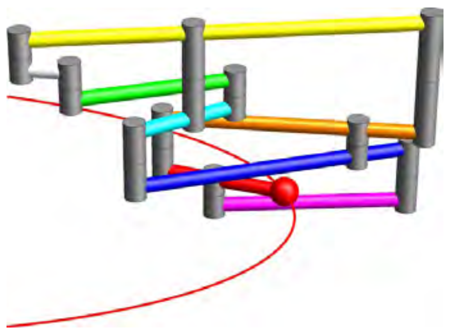


Figure 3: *Three-dimensional view of a linkage drawing the ellipse given by the parametrization* (3).

As an example to support his claim, O'Rourke points out that already constructing a linkage drawing the first letter "J" of John Hancock's famous signature on the United States Declaration of Independence would be a challenging task. The solution to this problem was found by our program and is depicted in Figure 4.

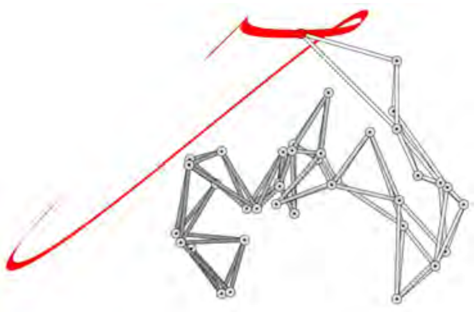


Figure 4: *A rational curve approximating the "J" in John Hancock's signature and a linkage drawing it using a quill pen whose shape is a line segment in direction* (5, 6).

References

[1] Matteo Gallet, Christoph Koutschan, Zijia Li, Georg Regensburger, Josef Schicho, and Nelly Villamizar. Planar linkages following a prescribed motion. *Mathematics of Computation*, 2016. To appear (preprint on arXiv:1502.05623), DOI: 10.1090/mcom/3120.

[2] Alfred B. Kempe. On a general method of describing plane curves of the n^{th} degree by linkwork. *Proceedings of the London Mathematical Society*, s1-7(1):213–216, 1876.

[3] Alexander Kobel. Automated generation of Kempe linkages for algebraic curves in a dynamic geometry system. Bachelor's thesis, University of Saarbrücken, 2008.

[4] Christoph Koutschan. Mathematica package PlanarLinkages and electronic supplementary material for the paper "Planar linkages following a prescribed motion", 2015. Available at `http://www.koutschan.de/data/link/`.

[5] Joseph O'Rourke. How to fold it. Cambridge University Press, Cambridge, 2011.