The `liftarm` package

Geometric constructions with liftarms using Ti*k*Z and LaTeX3

Matthias Floré

Version 3.0 (2024/05/20)

Abstract

This package is based on the package `tikz` (see [5]) and can be used to draw geometric constructions with liftarms using Ti*k*Z. There are several options for the appearance of the liftarms. It provides an environment to connect multiple liftarms using the Newton-Raphson method and LU decomposition. It also provides an environment to describe a construction and a method to animate a construction with one or more traces.

Contents

1 Usage

The package `liftarm` can be used by putting the following in the preamble.

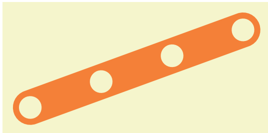
```
\usepackage{liftarm}
```

The package `liftarm` loads the package `xcolor` with the option `dvipsnames`, the package `tikz` and the Ti*k*Z library `calc`. Since `xcolor` is loaded with the option `dvipsnames`, packages such as `pgfplots` and `tcolorbox` must be loaded *after* `liftarm`.

2 Drawing liftarms

`\liftarm[⟨options⟩]{⟨point⟩}{⟨length⟩}{⟨angle⟩}`

This command can be placed inside a `tikzpicture` environment. It draws a liftarm of ⟨*length*⟩ starting at ⟨*point*⟩. The angle between the liftarm and the x-axis can be specified by ⟨*angle*⟩ in degrees. The distance between the holes is 1.



```
\begin{tikzpicture}
\liftarm{1,2}{3}{20}
\end{tikzpicture}
```

Note that the number of holes is ⟨*length*⟩ + 1. The ⟨*options*⟩ can be given with the following keys.

`/liftarm/axle holes={⟨values⟩}` (no default)

This key defines the holes in the liftarm where axle holes will be drawn.

```
\begin{tikzpicture}
\liftarm[axle holes={0,4}]{0,1}{4}{0}
\end{tikzpicture}
```

`/liftarm/brick=⟨boolean⟩` (default `true`, initially `false`)

If true, a brick will be drawn instead of a liftarm.


```
\begin{tikzpicture}
\liftarm[brick]{0,1}{2}{0}
\end{tikzpicture}
```

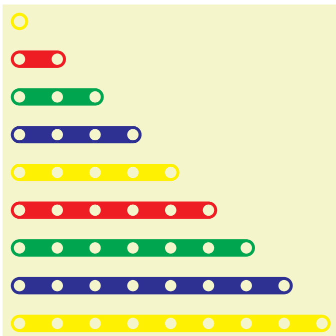
`/liftarm/color={⟨number⟩}{⟨color⟩}` (no default)

This key defines the color of liftarms of length ⟨*number*⟩.

Initially, the colors `Gray`, `darkgray`, `Yellow`, `Orange`, `Red`, `Green`, `Blue` and `Brown` are defined for respectively the lengths `0` till `7`.

`/liftarm/color modulo={⟨number⟩}` (no default, initially `8`)

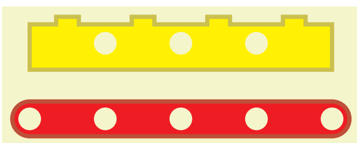
The default colors of the liftarms are determined by computing the length of the liftarm modulo the value of this key and selecting the color defined by the key `color`.



```
\begin{tikzpicture}[scale=0.5]
\pgfkeys{
  /liftarm,
  color={0}{Yellow},
  color={1}{Red},
  color={2}{Green},
  color={3}{Blue},
  color modulo=4
}
\foreach\n in {0,...,8}{
  \liftarm{0,-\n}{\n}{0}
}
\end{tikzpicture}
```

`/liftarm/contour=⟨boolean⟩` (default `true`, initially `false`)

If true, a contour will be drawn around the liftarm.

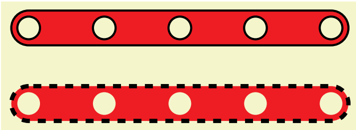


```
\begin{tikzpicture}
\liftarm[contour]{0,1}{4}{0}
\liftarm[brick,contour]{1,2}{2}{0}
\end{tikzpicture}
```

`/liftarm/contour style={⟨options⟩}` (style, no default, initially empty)

The style of the contour is determined as follows. First, the color is defined as ⟨*initial color of the liftarm*⟩`!75!black`. Then the option `ultra thick` is added. Thereafter, the style of the key `contour style` is added.

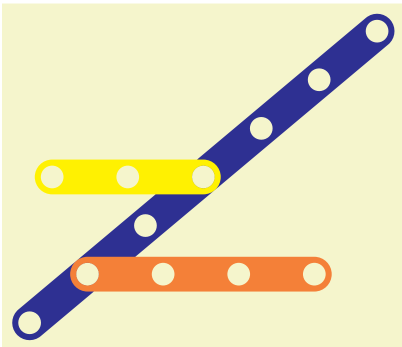
The style `contour style` only applies to the border of the liftarm. The style `liftarm style` also applies to the holes of the liftarm.



```
\begin{tikzpicture}
\liftarm[
  contour,
  contour style={dashed,black}
]{0,1}{4}{0}
\liftarm[
  liftarm style={draw=black,thick}
]{0,2}{4}{0}
\end{tikzpicture}
```

`/liftarm/coordinate={⟨number 1/name 1,...⟩}` (no default)

This key defines coordinates with name ⟨*name i*⟩ at hole ⟨*number i*⟩ of the liftarm.



```
\begin{tikzpicture}
\liftarm[
  coordinate={1/A,3/B}
]{0,1}{6}{40}
\liftarm{A}{3}{0}
\liftarm{B}{2}{180}
\end{tikzpicture}
```

`/liftarm/hole radius={⟨value⟩}` (no default, initially `0.3`)

The ⟨*value*⟩ of this key, multiplied with the ⟨*value*⟩ of the key `scalefactor` defines the radius of the holes.

```
\begin{tikzpicture}
\liftarm[hole radius=0.1]{0,0}{5}{0}
\end{tikzpicture}
```

`/liftarm/liftarm style={⟨options⟩}` (style, no default, initially empty)

The style of the liftarm is determined as follows. First, the color is defined by the keys `color` and `color modulo`. Thereafter, the style of the key `liftarm style` is added.

`/liftarm/liftarm thickness={⟨value⟩}` (no default, initially `0.92`)

The ⟨*value*⟩ of this key, multiplied with the ⟨*value*⟩ of the key `scalefactor` defines the thickness of the liftarm.

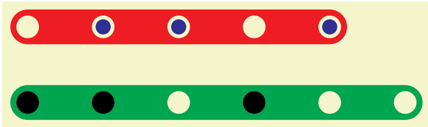

```
\begin{tikzpicture}
\liftarm[
  hole radius=0.1,
  liftarm thickness=0.3
]{0,0}{5}{0}
\end{tikzpicture}
```

`/liftarm/mark holes={⟨values⟩}` (no default)

`/liftarm/mark radius={⟨factor⟩}` (no default, initially `1`)

`/liftarm/mark style={⟨options⟩}` (style, no default, initially empty)

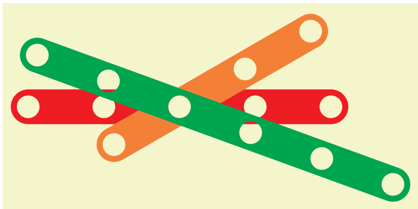
The key `mark holes` defines the holes in the liftarm which will be marked. The radius is the product of the ⟨*factor*⟩ given to the key `mark radius` and the value of the key `hole radius`. The style of these marks is determined as follows. First, the color is set to `black`. Thereafter, the style of the key `mark style` is added.



```
\begin{tikzpicture}
\liftarm[
  mark holes={0,1,3}
]{0,0}{5}{0}
\liftarm[
  mark holes={1,2,4},
  mark radius=2/3,
  mark style=Blue
]{0,1}{4}{0}
\end{tikzpicture}
```

`/liftarm/origin={`⟨*number*⟩`}` (no default, initially `0`)

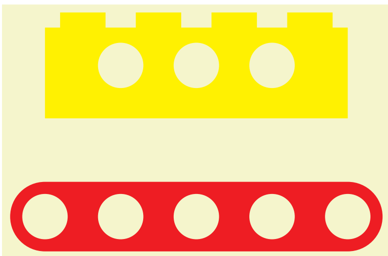
This key defines the number of the hole which will be placed at the coordinate given as argument to the liftarm.



```
\begin{tikzpicture}
\liftarm{-2,0}{4}{0}
\liftarm[origin=1]{0,0}{3}{30}
\liftarm[origin=2]{0,0}{5}{-20}
\end{tikzpicture}
```

`/liftarm/scalefactor={`⟨*value*⟩`}` (no default, initially `0.5`)

The ⟨*value*⟩ of this key defines the factor which scales the thickness of the liftarm and the radius of the holes.



```
\begin{tikzpicture}
\liftarm[scalefactor=1]{0,0}{4}{0}
\liftarm[brick,scalefactor=1]{1,2}{2}{0}
\end{tikzpicture}
```

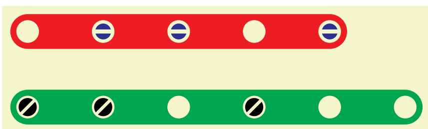
`/liftarm/screw angle={`⟨*angle*⟩`}` (no default, initially `10`)

`/liftarm/screw holes={`⟨*values*⟩`}` (no default)

`/liftarm/screw radius={`⟨*factor*⟩`}` (no default, initially `0.8`)

`/liftarm/screw style={`⟨*options*⟩`}` (style, no default, initially empty)

The key `screw holes` defines the holes in the liftarm where a screw will be drawn. The angle of these screws is determined by the key `screw angle` which is an angle in degrees. The radius is the product of the ⟨*factor*⟩ given to the key `screw radius` and the value of the key `hole radius`. The style of these screws is determined as follows. First, the color is set to `black`. Then the option `rotate=45` is added. Thereafter, the style of the key `screw style` is added.



```
\begin{tikzpicture}
\liftarm[
  screw holes={0,1,3}
]{0,0}{5}{0}
\liftarm[
  screw angle=15,
  screw holes={1,2,4},
  screw radius=0.7,
  screw style={Blue,rotate=-45}
]{0,1}{4}{0}
\end{tikzpicture}
```

`/liftarm/type=liftarm|line segment` (no default, initially `liftarm`)

`liftarm` In this case, the command `\liftarm` draws a liftarm.

`line segment` In this case, the command `\liftarm` draws a line segment.

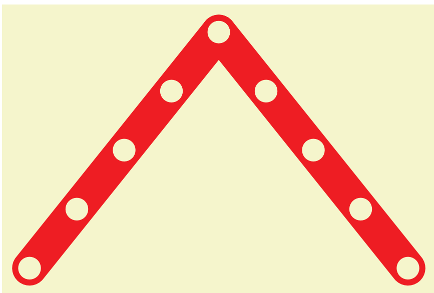
3 Connecting liftarms

```
\begin{liftarmconnect}[⟨options⟩]
  ⟨environment contents⟩
\end{liftarmconnect}
```

This environment can be placed inside a `tikzpicture` environment. It can be used to connect liftarms where the angles are computed automatically. The ⟨*options*⟩ can be a list of keys from the liftarm key family.

The contents should consist only of commands `\liftarm` and spaces.

The conditions to connect the liftarms are specified by the key `coordinate`. The resulting equations are determined automatically by the environment `liftarmconnect`. The number of liftarms needs to be equal to the number of equations. In the example below, there are 2 liftarms and 1 condition specified with the coordinate `A` resulting in 2 equations.



```
\begin{tikzpicture}
\coordinate (X) at (5,0);
\begin{liftarmconnect}
  \liftarm[coordinate=4/A]{0,0}{4}{60}
  \liftarm[coordinate=4/A]{X}{4}{120}
\end{liftarmconnect}
\end{tikzpicture}
```

The similar code below does not work because the coordinate `A` is used as the starting point of the second liftarm but is unknown since it is used in a condition for the first liftarm and furthermore, there is no liftarm to complement the condition involving `A` in the first liftarm.

```
\begin{tikzpicture}
\coordinate (X) at (5,0);
\begin{liftarmconnect}
  \liftarm[coordinate=4/A]{0,0}{4}{60}
  \liftarm[coordinate=4/X]{A}{4}{-60}
\end{liftarmconnect}
\end{tikzpicture}
```

If the environment `liftarmconnect` consists of 2 liftarms then the law of cosines is used to compute the angles.

If there are more than 2 liftarms then the set of equations is solved with the Newton-Raphson method. The initial values for the angles are given by the last arguments of the commands `\liftarm`. The Jacobian matrix is defined by the environment `liftarmconnect`. The resulting set of linear equations is solved with LU decomposition. The iteration stops if the condition determined by the key `connect stop` is satisfied.

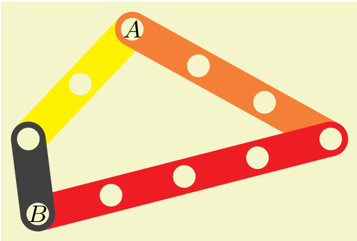
Since the *let operation* from the TikZ library `calc` is used, it is not possible to use the variable names `\n`, `\p`, `\x` and `\y` inside the starting point of a command `\liftarm` which is used in the environment `liftarmconnect`.

`/liftarm/connect stop=1-norm|2-norm|iterations` (no default, initially `1-norm`)

`1-norm` In this case, the iteration stops if the 1-norm is smaller than the value given to this key. Its default value is 0.001.

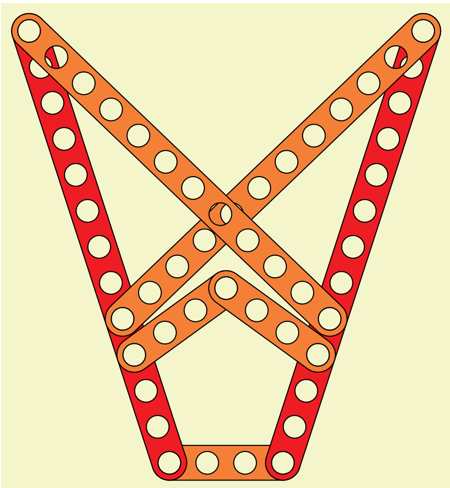
`2-norm` In this case, the iteration stops if the 2-norm is smaller than the value given to this key. Its default value is 0.001.

`iterations` In this case, a number of iterations is executed where the number is the one given to this key. Its default value is 10.



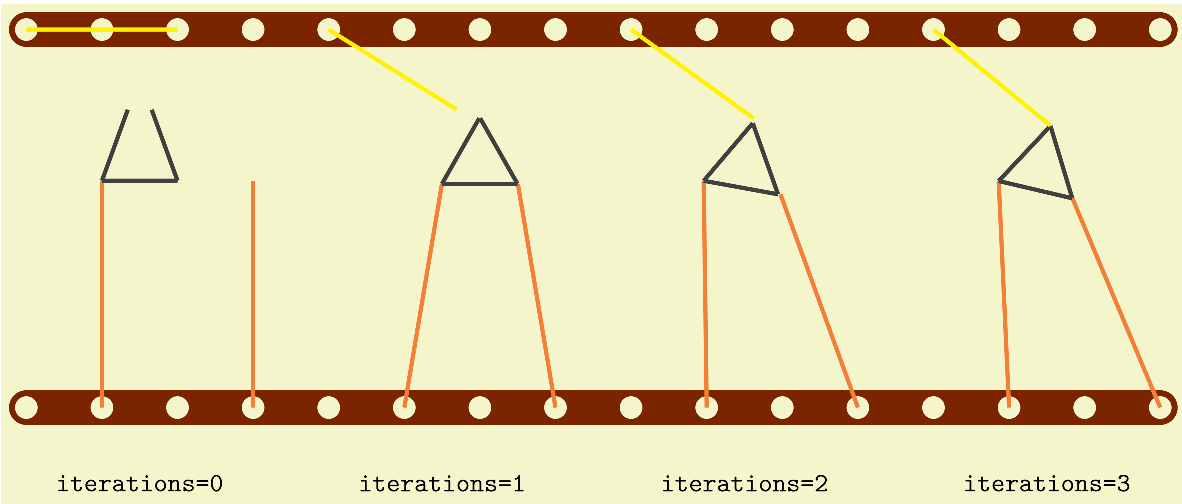

```
\begin{tikzpicture}
\begin{liftarmconnect}
  \liftarm[coordinate=2/A]{0,0}{2}{70}
  \liftarm[coordinate=3/A]{4,0}{3}{120}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=4/B]{4,0}{4}{200}
  \liftarm[coordinate=1/B]{0,0}{1}{-90}
\end{liftarmconnect}
\node at (A) {\small $A$};
\node at (B) {\small $B$};
\end{tikzpicture}
```

The example below shows the regular pentagon from [1]. In the first environment `liftarmconnect` there are 4 liftarms and 2 conditions resulting in 4 equations. Hence the Jacobian matrix has size 4×4.



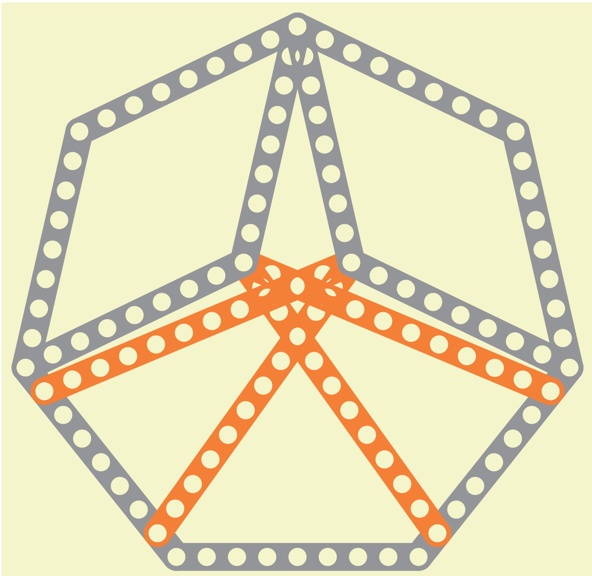
```
\begin{tikzpicture}[scale=0.5]
\pgfkeys{/liftarm,liftarm style={draw=black},scalefactor=1}
\liftarm{0,0}{3}{0}
\begin{liftarmconnect}
  \liftarm[coordinate={3/A,4/B,12/C}]{0,0}{12}{100}
  \liftarm[coordinate={3/D,4/E,12/F}]{3,0}{12}{80}
  \liftarm[coordinate=11/F]{B}{11}{60}
  \liftarm[coordinate=11/C]{E}{11}{120}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=3/G]{A}{3}{30}
  \liftarm[coordinate=3/G]{D}{3}{150}
\end{liftarmconnect}
\end{tikzpicture}
```

The example below shows iterations 0 till 3 of a construction with 6 liftarms and 3 conditions resulting in 6 equations. Hence the Jacobian matrix has size 6×6.



```
\begin{tikzpicture}
\liftarm{0,0}{15}{0}
\liftarm{0,5}{15}{0}
\foreach\k in {0,...,3}{
  \begin{scope}[shift={(\k*4,0)}]
    \begin{liftarmconnect}[connect stop={iterations=\k},liftarm style=ultra thick,type=line segment]
      \liftarm[coordinate=3/A]{1,0}{3}{90}
      \liftarm[coordinate=3/B]{3,0}{3}{90}
      \liftarm[coordinate=1/B]{A}{1}{0}
      \liftarm[coordinate=1/C]{A}{1}{70}
      \liftarm[coordinate=1/C]{B}{1}{110}
      \liftarm[coordinate=2/C]{0,5}{2}{0}
    \end{liftarmconnect}
    \node at (1.5,-1) {\texttt{iterations=\k}};
  \end{scope}
}
\end{tikzpicture}
```

The example below shows the regular heptagon from [1]. In the first environment `liftarmconnect` there are 8 liftarms and 4 conditions resulting in 8 equations. Hence the Jacobian matrix has size 8×8.



```
\begin{tikzpicture}[scale=0.4]
\pgfkeys{/liftarm,scalefactor=1}
\liftarm{-4,0}{8}{0}
\begin{liftarmconnect}
  \liftarm[coordinate={1/A,7/B,8/G}]{-4,0}{8}{135}
  \liftarm[coordinate=11/F]{A}{11}{50}
  \liftarm[coordinate=11/F]{B}{11}{20}
  \liftarm[coordinate={1/C,7/D,8/H}]{4,0}{8}{45}
  \liftarm[coordinate=11/E]{C}{11}{130}
  \liftarm[coordinate=11/E]{D}{11}{160}
  \liftarm[coordinate=8/E]{G}{8}{30}
  \liftarm[coordinate=8/F]{H}{8}{150}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=8/I]{E}{8}{70}
  \liftarm[coordinate=8/I]{F}{8}{110}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=8/J]{G}{8}{70}
  \liftarm[coordinate=8/J]{I}{8}{210}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=8/K]{H}{8}{110}
  \liftarm[coordinate=8/K]{I}{8}{-30}
\end{liftarmconnect}
\end{tikzpicture}
```

4 Describing a construction

If a construction involves many liftarms then it is convenient to describe this construction in separate steps. Then the content of previous steps would need to be copied in each new step. This process can be automated by using the command `\liftarmconstruct` below.

`\liftarmconstruct{⟨commands⟩}`

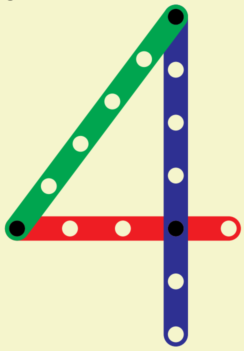
This command appends ⟨*commands*⟩ to an internal token list. Then it uses this token list.

`\liftarmconstructclear`

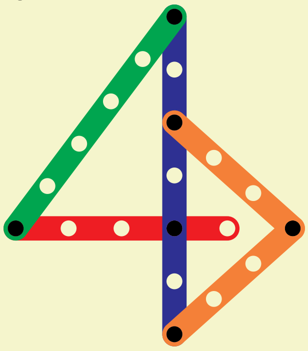
This command clears the token list which is used by the command `\liftarmconstruct`.

As an example, we describe below the construction of a regular pentagon from [1].

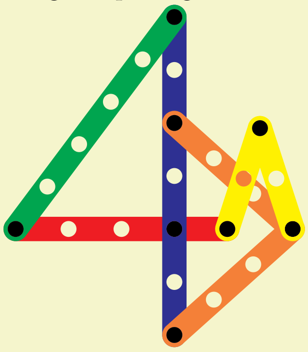
1. First we form a rectangular triangle with 3 liftarms.



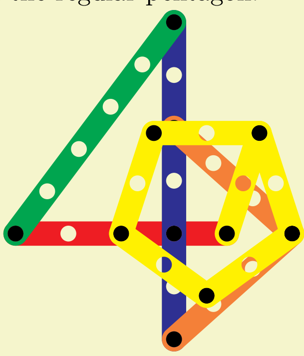
2. Then we add 2 liftarms of length 3.



3. Here appears the first side of the regular pentagon.



4. Now we end the construction of the regular pentagon.



```
\begin{minipage}{5.5cm}%only for usage in this manual
\liftarmconstructclear
\begin{enumerate}
\item First we form a rectangular triangle with 3 liftarms.
\begin{center}
\begin{tikzpicture}[scale=0.7]
\liftarmconstruct{
  \liftarm[mark holes=3]{-3,0}{4}{0}
  \begin{liftarmconnect}
    \liftarm[coordinate=6/A,origin=2]{0,0}{6}{90}
    \liftarm[coordinate=5/A,mark holes={0,5}]{-3,0}{5}{60}
  \end{liftarmconnect}
}
\end{tikzpicture}
\end{center}
\item Then we add 2 liftarms of length $3$.
\begin{center}
\begin{tikzpicture}[scale=0.7]
\liftarmconstruct{
  \begin{liftarmconnect}
    \liftarm[coordinate=3/B,mark holes={0,3}]{0,-2}{3}{45}
    \liftarm[coordinate=3/B,mark holes=0]{0,2}{3}{-45}
  \end{liftarmconnect}
}
\end{tikzpicture}
\end{center}
\item Here appears the first side of the regular pentagon.
\begin{center}
\begin{tikzpicture}[scale=0.7]
\liftarmconstruct{
  \begin{liftarmconnect}
    \liftarm[coordinate=2/C]{B}{2}{100}
    \liftarm[coordinate=2/C,mark holes={0,2}]{1,0}{2}{80}
  \end{liftarmconnect}
}
\end{tikzpicture}
\end{center}
\item Now we end the construction of the regular pentagon.
\begin{center}
\begin{tikzpicture}[scale=0.7]
\liftarmconstruct{
  \begin{liftarmconnect}
    \liftarm[coordinate=2/D]{C}{2}{180}
    \liftarm[coordinate=2/D,mark holes={0,2}]{-1,0}{2}{80}
  \end{liftarmconnect}
  \begin{liftarmconnect}
    \liftarm[coordinate=2/E,mark holes=2]{-1,0}{2}{-80}
    \liftarm[coordinate=2/E]{B}{2}{210}
  \end{liftarmconnect}
}
\end{tikzpicture}
\end{center}
\end{enumerate}
\end{minipage}
```

5 Animations

`\liftarmanimate[⟨options⟩]{⟨frame rate⟩}{⟨list⟩}{⟨command⟩}`

This command shows an animation using the `animateinline` environment of the package `animate`. The package `animate` is *not* loaded by default and needs to be loaded to use the command `\liftarmanimate`. The ⟨*options*⟩ are passed to the `animateinline` environment. The ⟨*frame rate*⟩ of the animation is described in the documentation of the package `animate`. The ⟨*command*⟩ must be a previously defined command with one mandatory argument. The ⟨*list*⟩ is passed to a `\foreach` loop. The frames of the animation consist of the ⟨*command*⟩ evaluated one by one in the result of the `\foreach` loop. The command `\liftarmanimate` creates a timeline which is used in the `animateinline` environment. This timeline is stored in the file ⟨*job name*⟩⟨*number of the animation in the document*⟩`.tln`. It requires two compiler runs to create and use this timeline correctly.

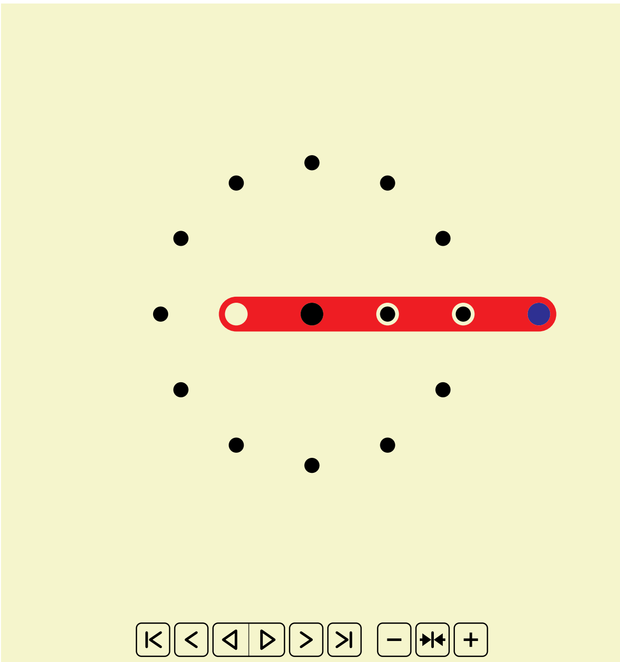
`/liftarm/trace={`⟨*number/number of frames/code*⟩`}...` (no default)

This key draws ⟨*code*⟩ at hole ⟨*number*⟩ of the liftarm on the frames of the animation determined by ⟨*number of frames*⟩.

If ⟨*number of frames*⟩ is 0 then the ⟨*code*⟩ is drawn starting at the current frame until the end of the animation. If ⟨*number of frames*⟩ is an integer greater than or equal to 1 then the ⟨*code*⟩ is drawn starting at the current frame and remaining during the next frames determined by ⟨*number of frames*⟩. If ⟨*number of frames*⟩ is left empty then the ⟨*code*⟩ is drawn starting at the beginning of the animation until the end of the animation.

The ⟨*code*⟩ can be some Ti*k*Z code. In this ⟨*code*⟩, $(0,0)$ is positioned at hole ⟨*number*⟩ of the liftarm. If ⟨*code*⟩ is left empty then a black circle with radius $\frac{2}{3}$ times the `hole radius` is used.

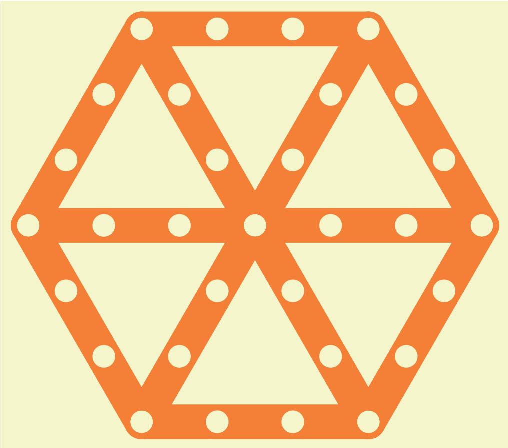
A list of multiple triples ⟨*number/number of frames/code*⟩ can be given to the key `trace`.



```
\usepackage {animate}
\newcommand{\exampleliftarmanimate}[1]{
  \liftarm[
    origin=1,
    mark holes=1,
    trace={
      2/0/,
      3//,
      4/3/{\fill[Blue] (0,0)
        circle[radius=0.15];}
    }
  ]{0,0}{4}{#1}
}
\liftarmanimate[
  autoplay,
  controls,
  loop,
  begin={
    \begin{tikzpicture}
    \useasboundingbox (-4,-4)
      rectangle (4,4);
  },
  end={\end{tikzpicture}}
]
{5}
{0,30,...,330}
{\exampleliftarmanimate}
```

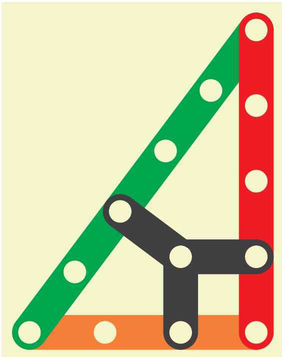
6 Additional examples

The following example shows a regular hexagon.



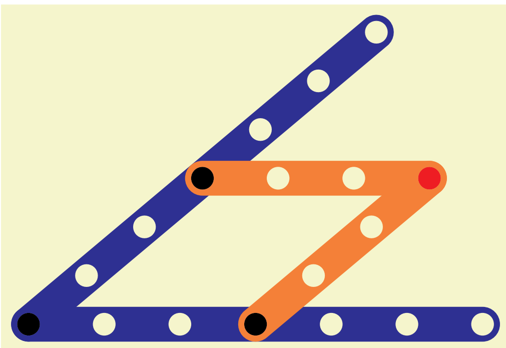
```
\begin{tikzpicture}
\def\r{3}
\foreach\m in {1,...,6}{
  \begin{liftarmconnect}
    \liftarm[coordinate=\r/A]{0,0}{\r}{(\m+1)*60}
    \liftarm[coordinate=\r/A]{\m*60:\r}{\r}{(\m+2)*60}
  \end{liftarmconnect}
}
\end{tikzpicture}
```

The following example illustrates that $2\,\mathrm{atan}(\frac{1}{2}) = \mathrm{atan}(\frac{4}{3})$.



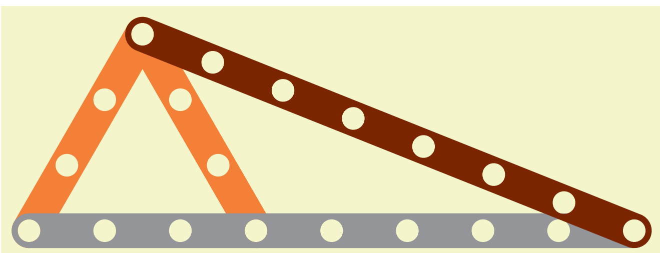
```
\begin{tikzpicture}
\liftarm{0,0}{3}{0}
\liftarm{0,0}{5}{atan(4/3)}
\liftarm{3,0}{4}{90}
\liftarm{2,0}{1}{90}
\liftarm{2,1}{1}{0}
\liftarm{2,1}{1}{90+atan(4/3)}
\end{tikzpicture}
```

The following example illustrates an angle bisection.



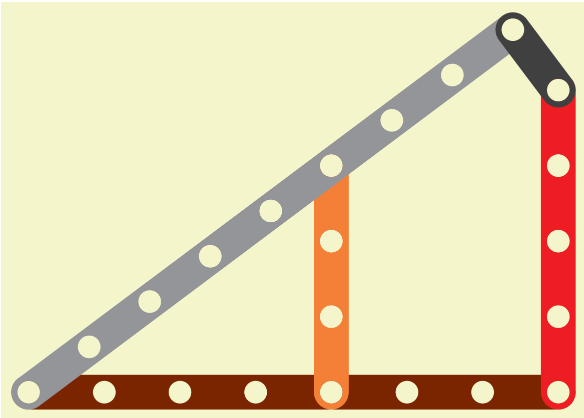
```
\begin{tikzpicture}
\def\ang{40}
\def\r{3}
\liftarm[mark holes={0,\r}]{0,0}{2*\r}{0}
\liftarm[mark holes=\r]{0,0}{2*\r}{\ang}
\liftarm[
  mark holes=\r,
  mark style=Red
]{\r,0}{\r}{\ang}
\liftarm{\ang:\r}{\r}{0}
\end{tikzpicture}
```

The following example illustrates that $7^2 = 3^2 + 8^2 - 2\cdot 3\cdot 8\cos(\frac{\pi}{3})$.



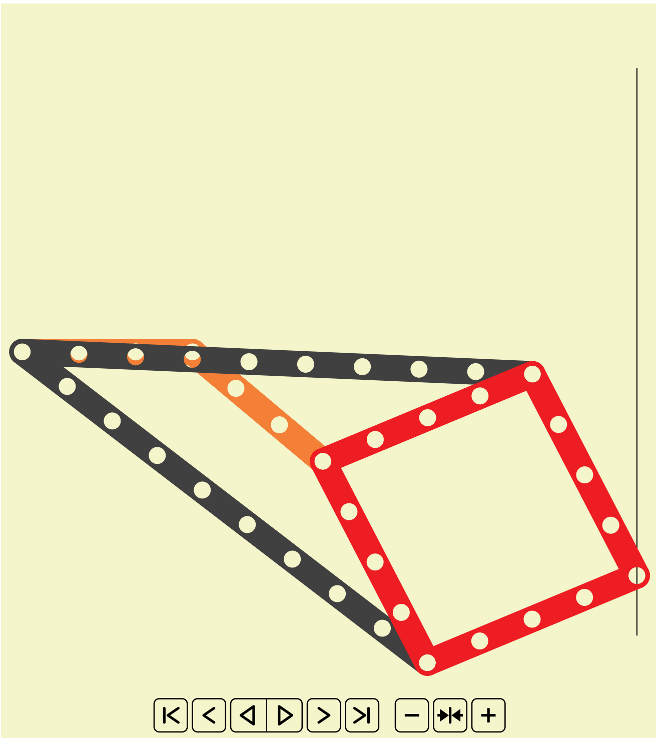
```
\begin{tikzpicture}
\begin{liftarmconnect}
  \liftarm[coordinate=3/A]{0,0}{3}{80}
  \liftarm[coordinate=3/A]{3,0}{3}{100}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=8/B]{0,0}{8}{0}
  \liftarm[coordinate=7/B]{A}{7}{0}
\end{liftarmconnect}
\end{tikzpicture}
```

The following example illustrates that $7^2 + 4^2 = 8^2 + 1^2$.



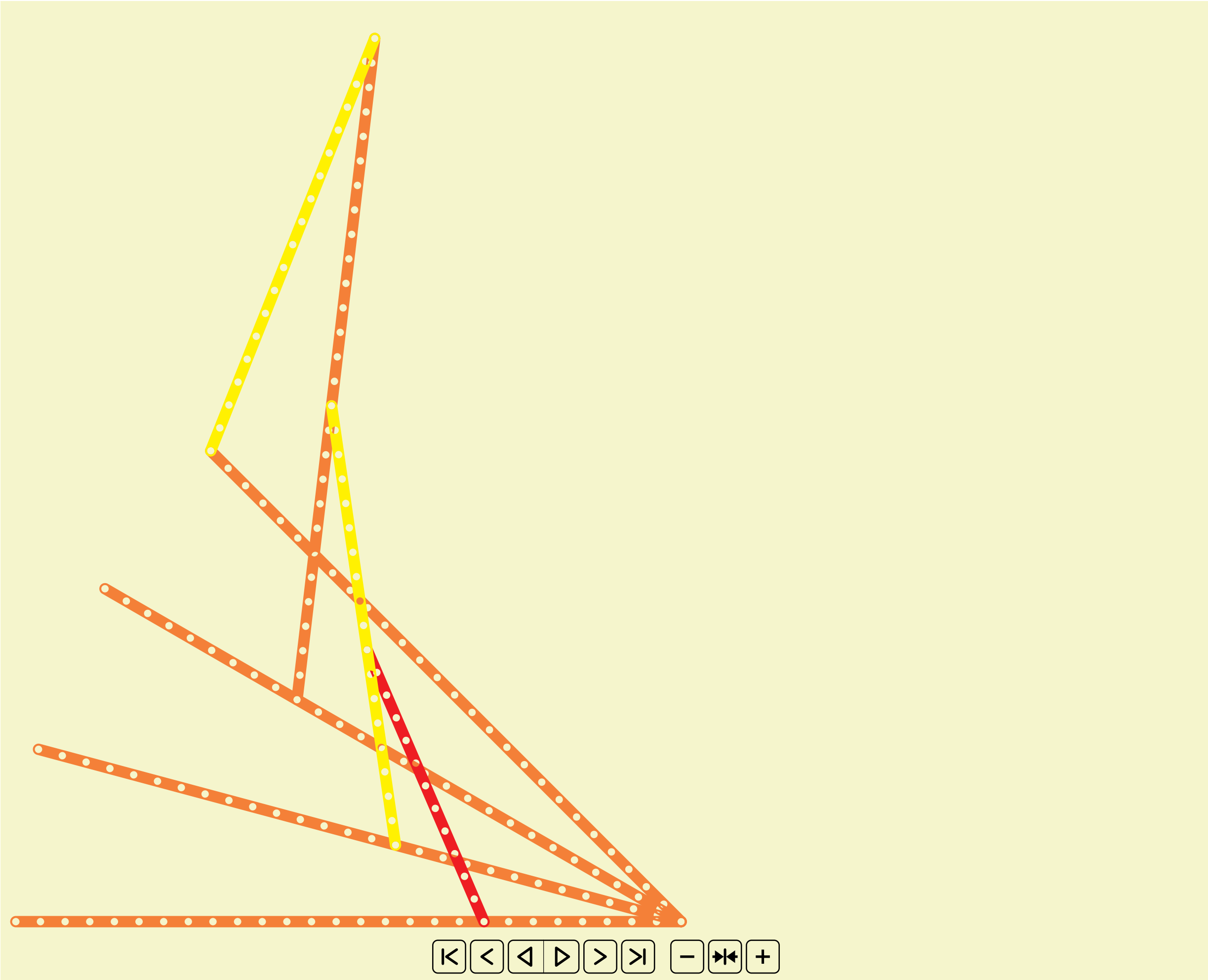
```
\begin{tikzpicture}
\def\a{4}
\def\b{7}
\def\c{1}
\def\d{8}
%\liftarm{0,0}{\b}{0}
%\liftarm{\b,0}{\a}{90}
\begin{liftarmconnect}
  \liftarm[coordinate=\b/A]{0,0}{\b}{0}
  \liftarm[coordinate=\a/A]{\b,\a}{\a}{-90}
\end{liftarmconnect}
\liftarm{4,0}{3}{90}
%\liftarm{\b,\a}{1}{atan(\a/\b)+atan(\c/\d)+90}
%\liftarm{0,0}{\d}{atan(\a/\b)+atan(\c/\d)}
\begin{liftarmconnect}
  \liftarm[coordinate=\d/B]{0,0}{\d}{45}
  \liftarm[coordinate=\c/B]{\b,\a}{\c}{135}
\end{liftarmconnect}
\end{tikzpicture}
```

Below is an animation of the Peaucellier-Lipkin linkage, see e.g. [4].



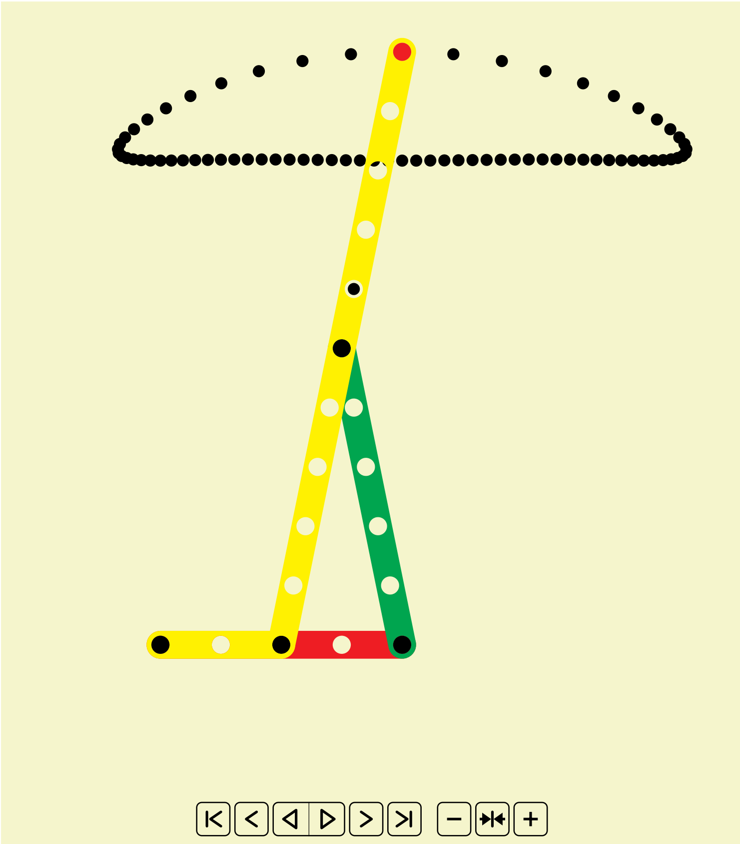
```
\usepackage {animate}
\newcommand{\PLlinkage}[1]{
\begin{tikzpicture}[scale=0.75]
\def\a{3}
\def\b{4}
\def\c{9}
\edef\l{
  \fpeval{2*\a+(\c^2-\b^2-(2*\a)^2)/(2*\a)}
}
\useasboundingbox (-0.23,-6) rectangle
  (\l+0.23,6);
\draw (\l,-5)--(\l,5);
\liftarm{0,0}{\a}{0}
\liftarm[coordinate=\a/A]{\a,0}{\a}{#1}
\begin{liftarmconnect}
  \liftarm[coordinate=\c/B]{0,0}{\c}{0}
  \liftarm[coordinate=\b/B]{A}{\b}{90}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=\c/C]{0,0}{\c}{0}
  \liftarm[coordinate=\b/C]{A}{\b}{-90}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=\b/D]{C}{\b}{0}
  \liftarm[coordinate=\b/D]{B}{\b}{0}
\end{liftarmconnect}
\end{tikzpicture}
}
\begin{animateinline}[
  autoplay,
  controls,
  palindrome
]{30}
\multiframe{80}{rAng=-40+1}{
  \PLlinkage{\rAng}
}
\end{animateinline}
```

Below is an animation of Kempe's trisector, as shown in [3].



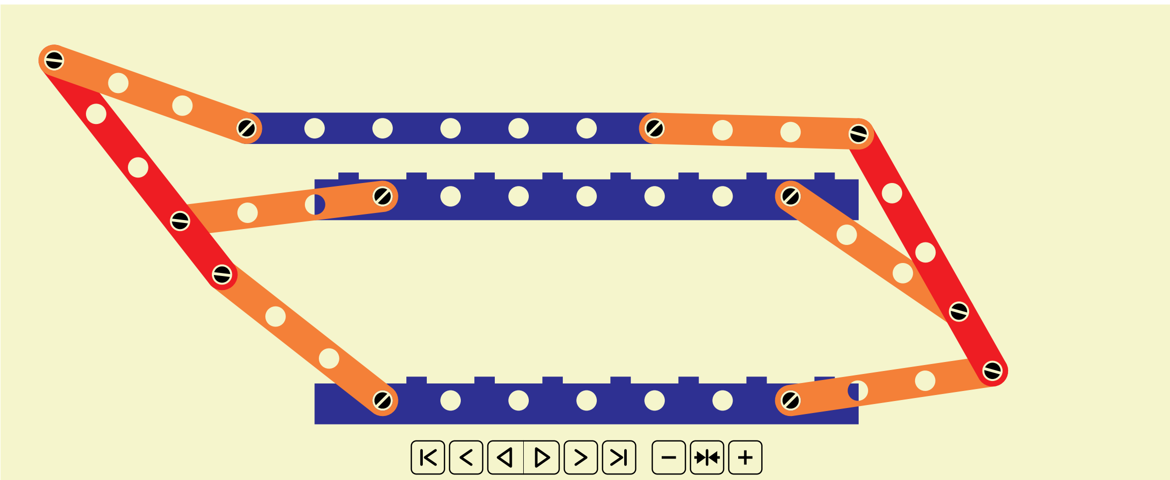
```
\usepackage {animate}
\newcommand{\trisector}[1]{
\begin{tikzpicture}[scale=0.33]
\useasboundingbox (-27.3,-0.5) rectangle (21.2,37);
\liftarm[coordinate=8/A]{0,0}{27}{180}
\liftarm[coordinate=12/B]{0,0}{27}{180-(#1)}
\liftarm[coordinate=18/C]{0,0}{27}{180-2*(#1)}
\liftarm[coordinate=27/D]{0,0}{27}{180-3*(#1)}
\begin{liftarmconnect}
  \liftarm[coordinate=27/E]{C}{27}{0}
  \liftarm[coordinate=18/E]{D}{18}{0}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=12/F]{A}{12}{0}
  \liftarm[coordinate=8/F]{B}{18}{0}
\end{liftarmconnect}
\end{tikzpicture}
}
\begin{animateinline}[autoplay,controls,palindrome]{5}
\multiframe{20}{rAng=15+1}{
  \trisector{\rAng}
}
\end{animateinline}
```

Below is an animation of Chebyshev's Lambda Mechanism.



```
\usepackage {animate}
\newcommand{\CL}[1]{
\liftarm{0,0}{4*\r}{0}
\liftarm[
  mark holes={0,2*\r}
]{0,0}{2*\r}{#1}
\begin{liftarmconnect}
  \liftarm[
    coordinate=5*\r/A,
    mark holes={0,5*\r}
  ]{4*\r,0}{5*\r}{90}
  \liftarm[
    coordinate=5*\r/A,
    mark holes=10*\r,
    mark style=Red,
    trace={6*\r/0/,10*\r//}
  ]{#1:2*\r}{10*\r}{90}
\end{liftarmconnect}
}
\liftarmanimate[
  autoplay,
  controls,
  loop,
  begin={
    \begin{tikzpicture}[scale=0.8]
    \def\r{1}
    \useasboundingbox
      (-2*\r-0.5,-2*\r-0.5)
      rectangle
      (10*\r-0.5,10*\r+0.5);
  },
  end={\end{tikzpicture}}
]
{20}
{0,5,...,355}
{\CL}
```

Below is an animation of a multilink steering mechanism.



```
\usepackage {animate}
\newcommand{\multilink}[1]{
\begin{tikzpicture}[scale=0.9]
\useasboundingbox (-8.5,-0.5) rectangle (8.5,5.7);
\liftarm[brick,screw holes={0,6}]{-3,0}{6}{0}
\liftarm[brick,screw holes={0,6}]{-3,3}{6}{0}
\liftarm[coordinate={0/X,6/Y},screw holes={0,6}]{{-3+(#1)*0.1},4}{6}{0}
\begin{liftarmconnect}
  \liftarm[coordinate=3/A]{-3,0}{3}{160}
  \liftarm[coordinate=3/B]{-3,3}{3}{200}
  \liftarm[coordinate={1/B,4/C},screw holes={0,1,4}]{A}{4}{90}
  \liftarm[coordinate=3/C]{X}{3}{180}
\end{liftarmconnect}
\begin{liftarmconnect}
  \liftarm[coordinate=3/D]{3,0}{3}{20}
  \liftarm[coordinate=3/E]{3,3}{3}{-20}
  \liftarm[coordinate={1/E,4/F},screw holes={0,1,4}]{D}{4}{90}
  \liftarm[coordinate=3/F]{Y}{3}{0}
\end{liftarmconnect}
\end{tikzpicture}
}
\begin{animateinline}[autoplay,controls,palindrome]{10}
\multiframe{41}{rAng=-20+1}{
  \multilink{\rAng}
}
\end{animateinline}
```

7 Version history

Version 1.0 (2022/03/08) First version.

Version 2.0 (2022/04/07) Removed some redundant `;` in the code.[1] Added the command `\liftarmanimate` and the key `trace`.

Version 3.0 (2024/05/20)

- The package now mainly uses LaTeX3 syntax. The package `etoolbox` is not loaded anymore.
- Improved the code for the key `axle holes`. In particular, the combinations with the keys `contour` and `hole radius` are fixed.
- Improved the path for the shape of a liftarm if the key `brick` is used.
- Changed the key `color` to accept two arguments. The color can no longer be specified without a key.
- Removed the keys `color 0`, `color 1`, `color 2`, `color 3`, `color 4`, `color 5`, `color 6` and `color 7`.
- In v2.0, the colors could only be defined up to length 7. In v3.0, this is not a restriction anymore.
- Changed some initial colors from `Black` to `black`.
- Added the keys `contour style` and `liftarm style`.
- Removed the keys `mark color`, `screw color` and `screw holes angle`. Added the keys `mark radius`, `mark style`, `screw angle`, `screw radius` and `screw style`.
- Improved the algorithm to connect liftarms in multiple ways. In v2.0, transformations such as `x={(0.8,0.5)},y={(-0.6,1.2)}` were not taken into account correctly. This is fixed in v3.0. In v2.0, only 2 liftarms could be connected automatically. In v3.0, this is not a restriction anymore. Therefore the command `\liftarmconnect` and the keys `connect`, `connect coordinate`, `connect reverse`, `liftarm 1` and `liftarm 2` are removed. Instead, the environment `liftarmconnect` and the key `connect stop` were added in v3.0.
- Changed the command `\liftarmconstruct` to allow more customization. Removed the environment `liftarmconstruction` and added the command `\liftarmconstructclear`.

[1] Thanks to Denis Bitouzé for pointing this out.

References

[1] Gerard 't Hooft, *Meccano Math I*, https://webspace.science.uu.nl/~hooft101/lectures/meccano.pdf, 2006.

[2] Gerard 't Hooft, *Meccano Math II*, https://webspace.science.uu.nl/~hooft101/lectures/meccano2.pdf, 2008.

[3] Gerard 't Hooft, *Meccano Math III*, https://webspace.science.uu.nl/~hooft101/lectures/meccano3.pdf, 2014.

[4] Alfred Bray Kempe, *On a general method of producing exact rectilinear motion by linkwork*, 1875.

[5] Till Tantau, *The TikZ and PGF Packages*, Manual for version 3.1.10, https://ctan.org/pkg/pgf, 2023.

Index

A The source code

```
%% liftarm.sty
%% Copyright 2022-2024 Matthias Floré
%
% This work may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.3c
% of this license or (at your option) any later version.
% The latest version of this license is in
%   http://www.latex-project.org/lppl.txt
% and version 1.3c or later is part of all distributions of LaTeX
% version 2005/12/01 or later.
%
% This work has the LPPL maintenance status `maintained'.
%
% The Current Maintainer of this work is Matthias Floré.
%
% This work consists of the files liftarm.pdf, liftarm.sty,
% liftarm.tex and README.md.
\NeedsTeXFormat{LaTeX2e}
\RequirePackage[dvipsnames]{xcolor}
\RequirePackage{tikz}
\usetikzlibrary{calc}
\ProvidesExplPackage{liftarm}{2024/05/20}{3.0}{Geometric constructions with liftarms using TikZ and LaTeX3}
```

A.1 Variables

```
\newcounter { g__liftarm_animate_frame_number_counter }
\newcounter { g__liftarm_animate_number_of_animation_counter }
\newcounter { g__liftarm_animate_number_of_steps_counter }
\newcounter { g__liftarm_animate_step_number_counter }

\bool_new:N \l__liftarm_animate_bool
\bool_new:N \l__liftarm_brick_bool
\bool_new:N \l__liftarm_contour_bool
\bool_new:N \l__liftarm_LU_bool

\clist_new:N \l__liftarm_trace_clist
```

```
\fp_new:N \l__liftarm_angle_fp
\fp_const:Nn \c__liftarm_axle_hole_angle_fp { 21.76702028497987 }%asind ( 1.78 / ( 16 * 0.3 ) )
\fp_new:N \l__liftarm_connect_det_fp
\fp_new:N \l__liftarm_connect_norm_fp
\fp_new:N \l__liftarm_connect_start_constant_x_fp
\fp_new:N \l__liftarm_connect_start_constant_y_fp
\fp_new:N \l__liftarm_connect_stop_value_fp
\fp_new:c { l__liftarm_connect_two_1_option_0_angle_fp }
\fp_new:c { l__liftarm_connect_two_1_option_1_angle_fp }
\fp_new:c { l__liftarm_connect_two_2_option_0_angle_fp }
\fp_new:c { l__liftarm_connect_two_2_option_1_angle_fp }
\fp_new:N \l__liftarm_connect_two_angle_fp
\fp_new:N \l__liftarm_connect_two_A_angle_fp
\fp_new:N \l__liftarm_connect_two_A_length_fp
\fp_new:N \l__liftarm_connect_two_A_x_fp
\fp_new:N \l__liftarm_connect_two_A_y_fp
\fp_new:N \l__liftarm_connect_two_B_angle_fp
\fp_new:N \l__liftarm_connect_two_B_length_fp
\fp_new:N \l__liftarm_connect_two_B_x_fp
\fp_new:N \l__liftarm_connect_two_B_y_fp
\fp_new:N \l__liftarm_connect_two_length_fp
\fp_new:N \g__liftarm_coord_x_fp
\fp_new:N \g__liftarm_coord_y_fp
\fp_new:N \l__liftarm_half_thickness_fp
\fp_new:N \l__liftarm_hole_radius_fp
\fp_new:N \l__liftarm_length_fp
\fp_new:N \l__liftarm_LU_maxA_fp
\fp_new:N \l__liftarm_LU_tmp_fp
\fp_new:N \l__liftarm_mark_radius_fp
\fp_new:N \l__liftarm_origin_fp
\fp_new:N \l__liftarm_origin_connect_initial_fp
\fp_new:N \l__liftarm_scalefactor_fp
\fp_new:N \l__liftarm_screw_angle_fp
\fp_new:N \l__liftarm_screw_radius_fp

\int_new:N \l__liftarm_connect_count_int
\int_new:N \l__liftarm_connect_equation_int
\int_new:N \l__liftarm_LU_count_int
\int_new:N \l__liftarm_LU_imax_int
```

```
\int_new:N \l__liftarm_LU_j_int
\int_new:N \l__liftarm_LU_N_int

\iow_new:N \g__liftarm_animate_write_timeline_iow

\seq_new:N \l__liftarm_connect_coordinate_seq
\seq_new:N \l__liftarm_connect_start_arg_seq
\seq_new:N \l__liftarm_connect_start_coeff_seq
\seq_new:N \l__liftarm_coordinate_seq
\seq_new:N \l__liftarm_trace_item_seq

\str_new:N \l__liftarm_connect_stop_type_str
\str_new:N \l__liftarm_type_str

\tl_new:N \g__liftarm_animate_frames_tl
\tl_new:N \g__liftarm_animate_frames_trace_tl
\tl_new:N \l__liftarm_animate_value_tl
\tl_new:N \l__liftarm_color_tl
\tl_new:N \g__liftarm_construct_tl
\tl_const:Nn \c__liftarm_cos_sin_diff_x_tl { - sin }
\tl_const:Nn \c__liftarm_cos_sin_diff_y_tl { cos }
\tl_const:Nn \c__liftarm_cos_sin_x_tl { cos }
\tl_const:Nn \c__liftarm_cos_sin_y_tl { sin }
\tl_new:N \l__liftarm_holes_tl
\tl_new:N \l__liftarm_shape_tl
\tl_new:N \l__liftarm_tmp_tl
```

A.2 Pgfkeys

```
\pgfkeys
  {
    / liftarm /. is~family ,
    / liftarm ,
    axle~holes /. initial = {} ,
    brick /. code = { \bool_set:Nn \l__liftarm_brick_bool { \cs:w c_#1_bool\cs_end: } } ,
    brick /. default = true ,
    brick = false ,
```

```
color /. code~2~args =
  {
    \tl_clear_new:c { l__liftarm_color_\int_eval:n {#1}_tl }
    \tl_set:cn { l__liftarm_color_\int_eval:n {#1}_tl } {#2}
  } ,
color = { 0 } { Gray } ,
color = { 1 } { darkgray } ,
color = { 2 } { Yellow } ,
color = { 3 } { Orange } ,
color = { 4 } { Red } ,
color = { 5 } { Green } ,
color = { 6 } { Blue } ,
color = { 7 } { Brown } ,
color~modulo /. initial = 8 ,
connect~stop /. is~choice ,
connect~stop / 1-norm /. code =
  {
    \str_set:Nn \l__liftarm_connect_stop_type_str { 1-norm }
    \fp_set:Nn \l__liftarm_connect_stop_value_fp {#1}
  } ,
connect~stop / 1-norm /. default = 0.001 ,
connect~stop / 2-norm /. code =
  {
    \str_set:Nn \l__liftarm_connect_stop_type_str { 2-norm }
    \fp_set:Nn \l__liftarm_connect_stop_value_fp {#1}
  } ,
connect~stop / 2-norm /. default = 0.001 ,
connect~stop / iterations /. code =
  {
    \str_set:Nn \l__liftarm_connect_stop_type_str { iterations }
    \fp_set:Nn \l__liftarm_connect_stop_value_fp {#1}
  } ,
connect~stop / iterations /. default = 10 ,
connect~stop = 1-norm ,
contour /. code = { \bool_set:Nn \l__liftarm_contour_bool { \cs:w c_#1_bool\cs_end: } } ,
contour /. default = true ,
contour = false ,
contour~style /. style = { contour_style /. style = {#1} } ,
contour_style /. style = {} ,
```

```
coordinate /. initial = {} ,
hole~radius /. initial = 0.3 ,
liftarm~style /. style = { liftarm_style /. style = {#1} } ,
liftarm_style /. style = {} ,
liftarm~thickness /. initial = 0.92 ,
mark~holes /. initial = {} ,
mark~radius /. code =
  {
    \pgfmathparse {#1}
    \fp_set:Nn \l__liftarm_mark_radius_fp { \pgfmathresult }
  } ,
mark~radius = 1 ,
mark~style /. style = { mark_style /. style = {#1} } ,
mark_style /. style = {} ,
origin /. code =
  {
    \pgfmathparse {#1}
    \fp_set:Nn \l__liftarm_origin_fp { \pgfmathresult }
  } ,
origin = 0 ,
scalefactor /. code =
  {
    \pgfmathparse {#1}
    \fp_set:Nn \l__liftarm_scalefactor_fp { \pgfmathresult }
  } ,
scalefactor = 0.5 ,
screw~angle /. code =
  {
    \pgfmathparse {#1}
    \fp_set:Nn \l__liftarm_screw_angle_fp { \pgfmathresult }
  } ,
screw~angle = 10 ,
screw~holes /. initial = {} ,
screw~radius /. code =
  {
    \pgfmathparse {#1}
    \fp_set:Nn \l__liftarm_screw_radius_fp { \pgfmathresult }
  } ,
screw~radius = 0.8 ,
```

```
    screw~style /. style = { screw_style /. style = {#1} } ,
    screw_style /. style = {} ,
    trace /. code = { \clist_set:Nn \l__liftarm_trace_clist {#1} } ,
    type /. is~choice ,
    type / liftarm /. code = { \str_set:Nn \l__liftarm_type_str { liftarm } } ,
    type / liftarm /. value~forbidden ,
    type / line~segment /. code = { \str_set:Nn \l__liftarm_type_str { line~segment } } ,
    type / line~segment /. value~forbidden ,
    type = liftarm ,
  }

\pgfkeys
  {
    / liftarm / connect_algorithm /. is~family ,
    / liftarm / connect_algorithm /. unknown /. code = {} ,
    / liftarm / connect_algorithm ,
    coordinate /. initial = {} ,
    origin /. code =
      {
        \pgfmathparse {#1}
        \fp_set:Nn \l__liftarm_origin_fp { \pgfmathresult }
      } ,
  }
```


A.3 Functions

```
\cs_generate_variant:Nn \clist_if_in:nnTF { enTF }
\cs_generate_variant:Nn \clist_map_inline:nn { en }
\cs_generate_variant:Nn \seq_map_indexed_inline:Nn { cn }
\cs_generate_variant:Nn \tl_build_begin:N { c }
\cs_generate_variant:Nn \tl_build_gbegin:N { c }
\cs_generate_variant:Nn \tl_build_end:N { c }
\cs_generate_variant:Nn \tl_build_gend:N { c }
\cs_generate_variant:Nn \tl_build_put_right:Nn { ce , cn }
\cs_generate_variant:Nn \tl_build_gput_right:Nn { ce , cn }

\cs_new_protected:Npn \__liftarm_connect:nnnn #1#2#3#4
```

```
{
  \int_incr:N \l__liftarm_connect_count_int
  \fp_zero_new:c { l__liftarm_connect_angle_\int_use:N \l__liftarm_connect_count_int _fp }
  \pgfmathparse {#4}
  \fp_set:cn { l__liftarm_connect_angle_\int_use:N \l__liftarm_connect_count_int _fp } { \pgfmathresult * deg }
  \fp_set_eq:NN \l__liftarm_origin_fp \l__liftarm_origin_connect_initial_fp
  \pgfkeys
    {
      / liftarm / connect_algorithm ,
      coordinate = \pgfkeysvalueof { / liftarm / coordinate } ,
      #1
    }
  \seq_if_in:NnTF \l__liftarm_connect_coordinate_seq {#2}
    {
      \fp_set_eq:Nc \l__liftarm_connect_start_constant_x_fp { l__liftarm_connect_constant_x_coord_#2_fp }
      \fp_set_eq:Nc \l__liftarm_connect_start_constant_y_fp { l__liftarm_connect_constant_y_coord_#2_fp }
      \seq_set_eq:Nc \l__liftarm_connect_start_arg_seq { l__liftarm_connect_arg_coord_#2_seq }
      \seq_set_eq:Nc \l__liftarm_connect_start_coeff_seq { l__liftarm_connect_coeff_coord_#2_seq }
    }
    {
      \__liftarm_def_coord:n {#2}
      \fp_set_eq:NN \l__liftarm_connect_start_constant_x_fp \g__liftarm_coord_x_fp
      \fp_set_eq:NN \l__liftarm_connect_start_constant_y_fp \g__liftarm_coord_y_fp
      \seq_clear:N \l__liftarm_connect_start_arg_seq
      \seq_clear:N \l__liftarm_connect_start_coeff_seq
    }
  \clist_map_inline:en { \pgfkeysvalueof { / liftarm / connect_algorithm / coordinate } }
    {
      \seq_set_split:Nnn \l__liftarm_coordinate_seq { / } {##1}
      \pgfmathparse { \seq_item:Nn \l__liftarm_coordinate_seq { 1 } }
      \fp_set:Nn \l__liftarm_length_fp { \pgfmathresult - \l__liftarm_origin_fp }
      \seq_if_in:NeTF \l__liftarm_connect_coordinate_seq { \seq_item:Nn \l__liftarm_coordinate_seq { 2 } }
        {
          \clist_map_inline:nn { x , y }
            {
              \int_incr:N \l__liftarm_connect_equation_int
              \tl_clear_new:c { l__liftarm_connect_F_\int_use:N \l__liftarm_connect_equation_int _tl }
              \tl_build_begin:c { l__liftarm_connect_F_\int_use:N \l__liftarm_connect_equation_int _tl }
              \int_step_inline:nn { \l__liftarm_LU_N_int }
```

```
  {
    \tl_clear_new:c { l__liftarm_connect_Jacobian_\int_use:N \l__liftarm_connect_equation_int _########1_tl }
    \tl_build_begin:c { l__liftarm_connect_Jacobian_\int_use:N \l__liftarm_connect_equation_int _########1_tl }
    \tl_build_put_right:cn { l__liftarm_connect_Jacobian_\int_use:N \l__liftarm_connect_equation_int _########1_tl }
      { 0 }
    \fp_zero_new:c { l__liftarm_LU_A_\int_use:N \l__liftarm_connect_equation_int _########1_fp }
  }
\tl_build_put_right:ce { l__liftarm_connect_F_\int_use:N \l__liftarm_connect_equation_int _tl }
  {
    \fp_use:c { l__liftarm_connect_constant_####1_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_fp }
    - \fp_use:c { l__liftarm_connect_start_constant_####1_fp }
    - \fp_use:N \l__liftarm_length_fp * \cs:w c__liftarm_cos_sin_####1_tl\cs_end:
    ( \exp_not:N \cs:w l__liftarm_connect_angle_\int_use:N \l__liftarm_connect_count_int _fp \exp_not:N \cs_end: )
  }
\tl_build_put_right:ce
  {
    l__liftarm_connect_Jacobian_\int_use:N \l__liftarm_connect_equation_int _
    \int_use:N \l__liftarm_connect_count_int _tl
  }
  {
    - \fp_use:N \l__liftarm_length_fp * \cs:w c__liftarm_cos_sin_diff_####1_tl\cs_end:
    ( \exp_not:N \cs:w l__liftarm_connect_angle_\int_use:N \l__liftarm_connect_count_int _fp \exp_not:N \cs_end: )
  }
\seq_map_indexed_inline:cn { l__liftarm_connect_arg_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_seq }
  {
    \tl_build_put_right:ce { l__liftarm_connect_F_\int_use:N \l__liftarm_connect_equation_int _tl }
      {
        + \seq_item:cn { l__liftarm_connect_coeff_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_seq } {########1}
        * \cs:w c__liftarm_cos_sin_####1_tl\cs_end:
        ( \exp_not:N \cs:w l__liftarm_connect_angle_########2_fp \exp_not:N \cs_end: )
      }
    \tl_build_put_right:ce { l__liftarm_connect_Jacobian_\int_use:N \l__liftarm_connect_equation_int _########2_tl }
      {
        + \seq_item:cn { l__liftarm_connect_coeff_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_seq } {########1}
        * \cs:w c__liftarm_cos_sin_diff_####1_tl\cs_end:
        ( \exp_not:N \cs:w l__liftarm_connect_angle_########2_fp \exp_not:N \cs_end: )
      }
  }
\seq_map_indexed_inline:Nn \l__liftarm_connect_start_arg_seq
```

```
                {
                  \tl_build_put_right:ce { l__liftarm_connect_F_\int_use:N \l__liftarm_connect_equation_int _tl }
                    {
                      - \seq_item:Nn \l__liftarm_connect_start_coeff_seq {########1} * \cs:w c__liftarm_cos_sin_####1_tl\cs_end:
                      ( \exp_not:N \cs:w l__liftarm_connect_angle_########2_fp \exp_not:N \cs_end: )
                    }
                  \tl_build_put_right:ce { l__liftarm_connect_Jacobian_\int_use:N \l__liftarm_connect_equation_int _########2_tl }
                    {
                      - \seq_item:Nn \l__liftarm_connect_start_coeff_seq {########1} * \cs:w c__liftarm_cos_sin_diff_####1_tl\cs_end:
                      ( \exp_not:N \cs:w l__liftarm_connect_angle_########2_fp \exp_not:N \cs_end: )
                    }
                }
              \tl_build_end:c { l__liftarm_connect_F_\int_use:N \l__liftarm_connect_equation_int _tl }
              \int_step_inline:nn { \l__liftarm_LU_N_int }
                { \tl_build_end:c { l__liftarm_connect_Jacobian_\int_use:N \l__liftarm_connect_equation_int _########1_tl } }
            }
        }
        {
          \clist_map_inline:nn { x , y }
            {
              \fp_zero_new:c { l__liftarm_connect_constant_####1_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_fp }
              \fp_set_eq:cc { l__liftarm_connect_constant_####1_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_fp }
                { l__liftarm_connect_start_constant_####1_fp }
            }
          \clist_map_inline:nn { arg , coeff }
            {
              \seq_clear_new:c { l__liftarm_connect_####1_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_seq }
              \seq_set_eq:cc { l__liftarm_connect_####1_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_seq }
                { l__liftarm_connect_start_####1_seq }
            }
          \seq_put_right:ce { l__liftarm_connect_arg_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_seq }
            { \int_use:N \l__liftarm_connect_count_int }
          \seq_put_right:ce { l__liftarm_connect_coeff_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_seq }
            { \fp_use:N \l__liftarm_length_fp }
          \seq_put_right:Ne \l__liftarm_connect_coordinate_seq { \seq_item:Nn \l__liftarm_coordinate_seq { 2 } }
        }
    }
}
```

```
\cs_new_protected:Npn \__liftarm_connect_stop_criterion:
  {
    \int_step_inline:nn { \l__liftarm_LU_N_int }
      { \fp_set:cn { l__liftarm_LU_b_##1_fp } { \cs:w l__liftarm_connect_F_##1_tl\cs_end: } }
    \str_case:Vn \l__liftarm_connect_stop_type_str
      {
        { 1-norm }
          {
            \fp_zero:N \l__liftarm_connect_norm_fp
            \int_step_inline:nn { \l__liftarm_LU_N_int }
              { \fp_add:Nn \l__liftarm_connect_norm_fp { abs ( \cs:w l__liftarm_LU_b_##1_fp\cs_end: ) } }
            \bool_set:Nn \l__liftarm_LU_bool
              { \fp_compare_p:nNn { \l__liftarm_connect_norm_fp } > { \l__liftarm_connect_stop_value_fp } }
          }
        { 2-norm }
          {
            \fp_zero:N \l__liftarm_connect_norm_fp
            \int_step_inline:nn { \l__liftarm_LU_N_int }
              { \fp_add:Nn \l__liftarm_connect_norm_fp { ( \cs:w l__liftarm_LU_b_##1_fp\cs_end: ) ^ 2 } }
            \bool_set:Nn \l__liftarm_LU_bool
              { \fp_compare_p:nNn { sqrt ( \l__liftarm_connect_norm_fp ) } > { \l__liftarm_connect_stop_value_fp } }
          }
        { iterations }
          {
            \bool_set:Nn \l__liftarm_LU_bool
              { \fp_compare_p:nNn { \l__liftarm_LU_count_int } < { \l__liftarm_connect_stop_value_fp } }
          }
      }
  }

\cs_new_protected:Npn \__liftarm_def_coord:n #1
  {
    \path let \p { l__liftarm_coord } = (#1) in
      [
        / utils / exec =
          {
            \fp_gset:Nn \g__liftarm_coord_x_fp
              { ( \pgf@yy * \x { l__liftarm_coord } - \pgf@yx * \y { l__liftarm_coord } ) / \l__liftarm_connect_det_fp }
            \fp_gset:Nn \g__liftarm_coord_y_fp
```

```
              { ( \pgf@xx * \y { l__liftarm_coord } - \pgf@xy * \x { l__liftarm_coord } ) / \l__liftarm_connect_det_fp }
          }
      ] ;
  }

\cs_new_protected:Npn \__liftarm_default:nnnn #1#2#3#4
  {
    \pgfmathparse {#3}
    \fp_set:Nn \l__liftarm_length_fp { \pgfmathresult }
    \fp_compare:nNnTF { \l__liftarm_length_fp } < { 0 }
      { \PackageWarning { liftarm } { The~length~( \fp_use:N \l__liftarm_length_fp )~of~the~liftarm~is~smaller~than~0. } }
      {
        \pgfmathparse {#4}
        \fp_set:Nn \l__liftarm_angle_fp { \pgfmathresult }
        \begin { scope }
          [
            shift = { (#2) } ,
            rotate = \fp_use:N \l__liftarm_angle_fp
          ]
          \pgfkeys { / liftarm , #1 }
          \tl_set:Ne \l__liftarm_color_tl
            {
              \cs:w
                l__liftarm_color_
                \int_mod:nn { \fp_eval:n { trunc ( \l__liftarm_length_fp , 0 ) } } { \pgfkeysvalueof { / liftarm / color~modulo } }_tl
              \cs_end:
            }
          \begin { scope } [ shift = { ( - \fp_use:N \l__liftarm_origin_fp , 0 ) } ]
            \str_case:Vn \l__liftarm_type_str
              {
                { liftarm }
                  {
                    \pgfmathparse { \pgfkeysvalueof { / liftarm / liftarm~thickness } }
                    \fp_set:Nn \l__liftarm_half_thickness_fp { 0.5 * \l__liftarm_scalefactor_fp * \pgfmathresult }
                    \pgfmathparse { \pgfkeysvalueof { / liftarm / hole~radius } }
                    \fp_set:Nn \l__liftarm_hole_radius_fp { \l__liftarm_scalefactor_fp * \pgfmathresult }
                    \bool_if:NTF \l__liftarm_brick_bool
                      {
                        \tl_build_begin:N \l__liftarm_shape_tl
```

```
  \tl_build_put_right:Ne \l__liftarm_shape_tl
    {
      ( -1 , \fp_eval:n { - \l__liftarm_scalefactor_fp * 0.7 } )
      -- ( -1 , \fp_eval:n { \l__liftarm_scalefactor_fp * 0.5 } )
    }
  \int_step_inline:nnn { -1 } { \fp_eval:n { trunc ( \l__liftarm_length_fp , 0 ) } }
    {
      \tl_build_put_right:Ne \l__liftarm_shape_tl
        {
          -- (
            \fp_eval:n { ##1 + 0.5 - \l__liftarm_scalefactor_fp * 0.3 } ,
            \fp_eval:n { \l__liftarm_scalefactor_fp * 0.5 }
          )
          --++ ( 0 , \fp_eval:n { \l__liftarm_scalefactor_fp * 0.2 } )
          --++ ( \fp_eval:n { \l__liftarm_scalefactor_fp * 0.6 } , 0 )
          --++ ( 0 , \fp_eval:n { - \l__liftarm_scalefactor_fp * 0.2 } )
        }
    }
  \tl_build_put_right:Ne \l__liftarm_shape_tl
    {
      -- ( \fp_eval:n { \l__liftarm_length_fp + 1 } , \fp_eval:n { \l__liftarm_scalefactor_fp * 0.5 } )
      --++ ( 0 , \fp_eval:n { - \l__liftarm_scalefactor_fp * 1.2 } )
      -- cycle
    }
  \tl_build_end:N \l__liftarm_shape_tl
}
{
  \tl_set:Ne \l__liftarm_shape_tl
    {
      ( 0 , \fp_use:N \l__liftarm_half_thickness_fp )
      arc
        [
          start~angle = 90 ,
          end~angle = 270 ,
          radius = \fp_use:N \l__liftarm_half_thickness_fp
        ]
      -- ( \fp_use:N \l__liftarm_length_fp , - \fp_use:N \l__liftarm_half_thickness_fp )
      arc
        [
```

```
      start~angle = -90 ,
      end~angle = 90 ,
      radius = \fp_use:N \l__liftarm_half_thickness_fp
     ]
    -- cycle
   }
  }
\tl_build_begin:N \l__liftarm_holes_tl
\int_step_inline:nnn { 0 } { \fp_eval:n { trunc ( \l__liftarm_length_fp , 0 ) } }
 {
  \clist_if_in:enTF { \pgfkeysvalueof { / liftarm / axle~holes } } {##1}
   {
    \int_step_inline:nn { 4 }
     {
      \tl_build_put_right:Ne \l__liftarm_holes_tl
       {
        (
         \fp_eval:n
          {
           ##1 + sqrt ( 2 ) * \l__liftarm_hole_radius_fp * sind ( \c__liftarm_axle_hole_angle_fp )
           * cosd ( ####1 * 90 - 45 )
          } ,
         \fp_eval:n { sqrt ( 2 ) * \l__liftarm_hole_radius_fp * sind ( \c__liftarm_axle_hole_angle_fp )
         * sind ( ####1 * 90 - 45 ) }
        )
        -- (
         \fp_eval:n
          {
           ##1
           + \l__liftarm_hole_radius_fp * cosd ( ####1 * 90 - \c__liftarm_axle_hole_angle_fp )
          } ,
         \fp_eval:n { \l__liftarm_hole_radius_fp * sind ( ####1 * 90 - \c__liftarm_axle_hole_angle_fp ) }
        )
        arc
         [
          start~angle = \fp_eval:n { ####1 * 90 - \c__liftarm_axle_hole_angle_fp } ,
          end~angle = \fp_eval:n { ####1 * 90 + \c__liftarm_axle_hole_angle_fp } ,
          radius = \fp_use:N \l__liftarm_hole_radius_fp
         ]
```

```
        --
       }
     }
    \tl_build_put_right:Nn \l__liftarm_holes_tl { cycle }
   }
   {
    \tl_build_put_right:Ne \l__liftarm_holes_tl
     { ( ##1 , 0 ) circle [ radius = \fp_use:N \l__liftarm_hole_radius_fp ] }
   }
 }
\tl_build_end:N \l__liftarm_holes_tl
\fill [ \l__liftarm_color_tl , even~odd~rule , / liftarm / liftarm_style ]
 \l__liftarm_shape_tl \l__liftarm_holes_tl ;
\bool_if:NT \l__liftarm_contour_bool
 { \draw [ \l__liftarm_color_tl ! 75 ! black , ultra~thick , / liftarm / contour_style ] \l__liftarm_shape_tl ; }
\clist_map_inline:en { \pgfkeysvalueof { / liftarm / mark~holes } }
 {
  \fill [ black , / liftarm / mark_style ]
   ( {##1} , 0 ) circle [ radius = \fp_eval:n { \l__liftarm_mark_radius_fp * \l__liftarm_hole_radius_fp } ] ;
 }
\clist_map_inline:en { \pgfkeysvalueof { / liftarm / screw~holes } }
 {
  \clist_map_inline:nn { -1 , 1 }
   {
    \fill [ black , shift = { ( {##1} , 0 ) } , rotate = 45 , / liftarm / screw_style ]
     (
      \fp_eval:n { ####1 * \l__liftarm_screw_angle_fp }
      \c_colon_str
      \fp_eval:n { \l__liftarm_screw_radius_fp * \l__liftarm_hole_radius_fp }
     )
     arc
      [
       start~angle = \fp_eval:n { ####1 * \l__liftarm_screw_angle_fp } ,
       end~angle = \fp_eval:n { ####1 * ( 180 - \l__liftarm_screw_angle_fp ) } ,
       radius = \fp_eval:n { \l__liftarm_screw_radius_fp * \l__liftarm_hole_radius_fp }
      ]
     ;
   }
 }
```

```
      }
    { line~segment }
      {
        \draw [ \l__liftarm_color_tl , / liftarm / liftarm_style ] ( 0 , 0 ) -- ( \fp_use:N \l__liftarm_length_fp , 0 ) ;
      }
  }
\clist_map_inline:en { \pgfkeysvalueof { / liftarm / coordinate } }
  {
    \seq_set_split:Nnn \l__liftarm_coordinate_seq { / } {##1}
    \coordinate ( \seq_item:Nn \l__liftarm_coordinate_seq { 2 } )
      at ( { \seq_item:Nn \l__liftarm_coordinate_seq { 1 } } , 0 ) ;
  }
\bool_if:NT \l__liftarm_animate_bool
  {
    \clist_map_inline:Nn \l__liftarm_trace_clist
      {
        \seq_set_split:Nnn \l__liftarm_trace_item_seq { / } {##1}
        \stepcounter { g__liftarm_animate_frame_number_counter }
        \tl_build_gput_right:Ne \g__liftarm_animate_frames_trace_tl
          {
            \exp_not:n { \newframe \begin } { scope }
            [ shift = { (#2) } , rotate = \fp_use:N \l__liftarm_angle_fp ]
            \exp_not:N \begin { scope }
            [ shift = { ( \fp_eval:n { \seq_item:Nn \l__liftarm_trace_item_seq { 1 } - \l__liftarm_origin_fp } , 0 ) } ]
            \tl_if_empty:eTF { \seq_item:Nn \l__liftarm_trace_item_seq { 3 } }
              {
                \exp_not:N \fill
                [ black ] ( 0 , 0 ) circle [ radius = \fp_eval:n { \l__liftarm_hole_radius_fp * 2 / 3 } ] ;
              }
              { \seq_item:Nn \l__liftarm_trace_item_seq { 3 } }
            \exp_not:n { \end { scope } \end { scope } }
          }
        \tl_if_empty:eTF { \seq_item:Nn \l__liftarm_trace_item_seq { 2 } }
          {
            \tl_build_gput_right:ce { g__liftarm_animate_timeline_0_tl }
              { \theg__liftarm_animate_frame_number_counter x 0 , }
          }
          {
            \pgfmathparse { \use:e { \seq_item:Nn \l__liftarm_trace_item_seq { 2 } } }
```

```
                        \tl_build_gput_right:ce { g__liftarm_animate_timeline_\theg__liftarm_animate_step_number_counter _tl }
                          { \theg__liftarm_animate_frame_number_counter x \fp_eval:n { \pgfmathresult } , }
                      }
                  }
              }
          \end { scope }
        \end { scope }
      }
  }

\cs_new_protected:Npn \__liftarm_LU_decomposition:
  {
    \int_step_inline:nn { \l__liftarm_LU_N_int }
      {
        \int_zero_new:c { l__liftarm_LU_P_##1_int }
        \int_set:cn { l__liftarm_LU_P_##1_int } {##1}
      }
    \int_step_inline:nn { \l__liftarm_LU_N_int }
      {
        \fp_zero:N \l__liftarm_LU_maxA_fp
        \int_set:Nn \l__liftarm_LU_imax_int {##1}
        \int_step_inline:nnn {##1} { \l__liftarm_LU_N_int }
          {
            \fp_set:Nn \l__liftarm_LU_tmp_fp { abs ( \cs:w l__liftarm_LU_A_####1_##1_fp\cs_end: ) }
            \fp_compare:nNnT { \l__liftarm_LU_tmp_fp } > { \l__liftarm_LU_maxA_fp }
              {
                \fp_set_eq:NN \l__liftarm_LU_maxA_fp \l__liftarm_LU_tmp_fp
                \int_set:Nn \l__liftarm_LU_imax_int {####1}
              }
          }
        \int_compare:nNnF { \l__liftarm_LU_imax_int } = {##1}
          {
            \int_set_eq:Nc \l__liftarm_LU_j_int { l__liftarm_LU_P_##1_int }
            \int_set_eq:cc { l__liftarm_LU_P_##1_int } { l__liftarm_LU_P_\int_use:N \l__liftarm_LU_imax_int _int }
            \int_set_eq:cN { l__liftarm_LU_P_\int_use:N \l__liftarm_LU_imax_int _int } \l__liftarm_LU_j_int
            \int_step_inline:nn { \l__liftarm_LU_N_int }
              {
                \fp_set_eq:Nc \l__liftarm_LU_tmp_fp { l__liftarm_LU_A_##1_####1_fp }
                \fp_set_eq:cc { l__liftarm_LU_A_##1_####1_fp } { l__liftarm_LU_A_\int_use:N \l__liftarm_LU_imax_int _####1_fp }
```

```
                \fp_set_eq:cN { l__liftarm_LU_A_\int_use:N \l__liftarm_LU_imax_int _####1_fp } \l__liftarm_LU_tmp_fp
              }
          }
        \int_step_inline:nnn { ##1 + 1 } { \l__liftarm_LU_N_int }
          {
            \fp_set:cn { l__liftarm_LU_A_####1_##1_fp }
              { \cs:w l__liftarm_LU_A_####1_##1_fp\cs_end: / \cs:w l__liftarm_LU_A_##1_##1_fp\cs_end: }
            \int_step_inline:nnn { ##1 + 1 } { \l__liftarm_LU_N_int }
              {
                \fp_sub:cn { l__liftarm_LU_A_####1_########1_fp }
                  { \cs:w l__liftarm_LU_A_####1_##1_fp\cs_end: * \cs:w l__liftarm_LU_A_##1_########1_fp\cs_end: }
              }
          }
      }
  }

\cs_new_protected:Npn \__liftarm_LU_solve:
  {
    \int_step_inline:nn { \l__liftarm_LU_N_int }
      {
        \fp_zero_new:c { l__liftarm_LU_x_##1_fp }
        \fp_set_eq:cc { l__liftarm_LU_x_##1_fp } { l__liftarm_LU_b_\int_use:c { l__liftarm_LU_P_##1_int }_fp }
        \int_step_inline:nn { ##1 - 1 }
          {
            \fp_sub:cn { l__liftarm_LU_x_##1_fp }
              { \cs:w l__liftarm_LU_A_##1_####1_fp\cs_end: * \cs:w l__liftarm_LU_x_####1_fp\cs_end: }
          }
      }
    \int_step_inline:nnnn { \l__liftarm_LU_N_int } { -1 } { 1 }
      {
        \int_step_inline:nnn { ##1 + 1 } { \l__liftarm_LU_N_int }
          {
            \fp_sub:cn { l__liftarm_LU_x_##1_fp }
              { \cs:w l__liftarm_LU_A_##1_####1_fp\cs_end: * \cs:w l__liftarm_LU_x_####1_fp\cs_end: }
          }
        \fp_set:cn { l__liftarm_LU_x_##1_fp } { \cs:w l__liftarm_LU_x_##1_fp\cs_end: / \cs:w l__liftarm_LU_A_##1_##1_fp\cs_end: }
      }
  }
```

```
\cs_new:Npn \__liftarm_Mod:nn #1#2
  {
    min
      (
        Mod
          (
            \fp_eval:n { \cs:w l__liftarm_connect_two_#1_option_#2_angle_fp\cs_end: - \cs:w l__liftarm_connect_angle_#1_fp\cs_end: } ,
            360
          ) ,
        Mod
          (
            \fp_eval:n { \cs:w l__liftarm_connect_angle_#1_fp\cs_end: - \cs:w l__liftarm_connect_two_#1_option_#2_angle_fp\cs_end: } ,
            360
          )
      )
  }
```

A.4 Document commands and environment


```
\NewDocumentCommand \liftarm { O {} m m m }
  { \__liftarm_default:nnnn {#1} {#2} {#3} {#4} }

\NewDocumentCommand \liftarmanimate { O {} m m m }
  {
    \bool_set_true:N \l__liftarm_animate_bool
    \stepcounter { g__liftarm_animate_number_of_animation_counter }
    \setcounter { g__liftarm_animate_number_of_steps_counter } { -1 }
    \tl_build_gbegin:N \g__liftarm_animate_frames_tl
    \tl_build_gbegin:N \g__liftarm_animate_frames_trace_tl
    \setcounter { g__liftarm_animate_step_number_counter } { -1 }
    \foreach \l__liftarm_animate_value_tl in {#3}
      {
        \stepcounter { g__liftarm_animate_number_of_steps_counter }
        \tl_build_gput_right:Ne \g__liftarm_animate_frames_tl
          {
            \exp_not:n { \newframe \stepcounter { g__liftarm_animate_step_number_counter } #4 }
            { \l__liftarm_animate_value_tl }
          }
```

```
   }
  \tl_build_gend:N \g__liftarm_animate_frames_tl
  \int_step_inline:nnn { 0 } { \theg__liftarm_animate_number_of_steps_counter }
   {
    \tl_clear_new:c { g__liftarm_animate_timeline_##1_tl }
    \tl_build_gbegin:c { g__liftarm_animate_timeline_##1_tl }
   }
  \tl_build_gput_right:cn { g__liftarm_animate_timeline_0_tl } { c , }
  \setcounter { g__liftarm_animate_frame_number_counter } { \theg__liftarm_animate_number_of_steps_counter }
  \file_if_exist:nF { \c_sys_jobname_str \theg__liftarm_animate_number_of_animation_counter . tln }
   {
    \iow_open:Nn \g__liftarm_animate_write_timeline_iow
     { \c_sys_jobname_str \theg__liftarm_animate_number_of_animation_counter . tln }
    \iow_now:Ne \g__liftarm_animate_write_timeline_iow { \c_colon_str \c_colon_str c , 0 }
    \iow_close:N \g__liftarm_animate_write_timeline_iow
   }
  \begin { animateinline } [ #1 , timeline = \c_sys_jobname_str \theg__liftarm_animate_number_of_animation_counter . tln ] {#2}
   \tl_tail:N \g__liftarm_animate_frames_tl%remove the first \newframe
   \tl_build_gend:N \g__liftarm_animate_frames_trace_tl
   \g__liftarm_animate_frames_trace_tl
  \end { animateinline }
  \iow_open:Nn \g__liftarm_animate_write_timeline_iow { \c_sys_jobname_str \theg__liftarm_animate_number_of_animation_counter . tln }
  \int_step_inline:nnn { 0 } { \theg__liftarm_animate_number_of_steps_counter }
   {
    \tl_build_gend:c { g__liftarm_animate_timeline_##1_tl }
    \iow_now:Ne \g__liftarm_animate_write_timeline_iow
     { \c_colon_str \c_colon_str \cs:w g__liftarm_animate_timeline_##1_tl\cs_end: ##1 }
   }
  \iow_close:N \g__liftarm_animate_write_timeline_iow
  \bool_set_false:N \l__liftarm_animate_bool
 }

\NewDocumentCommand \liftarmconstruct { m }
 {
  \tl_gput_right:Nn \g__liftarm_construct_tl {#1}
  \g__liftarm_construct_tl
 }
```

```
\NewDocumentCommand \liftarmconstructclear {}
  { \tl_gclear:N \g__liftarm_construct_tl }

\NewDocumentEnvironment { liftarmconnect } { O {} +b }
  {
    \pgfkeys { / liftarm , #1 }
    %verify that the contents consists only of commands \liftarm because the contents of this environment are processed several times
    \DeclareExpandableDocumentCommand \liftarm { O {} m m m } {}%expandable for usage in \tl_set:Ne
    \tl_set:Ne \l__liftarm_tmp_tl {#2}
    \tl_remove_all:Nn \l__liftarm_tmp_tl { \par }
    \tl_if_blank:VF \l__liftarm_tmp_tl
      { \PackageError { liftarm } { The~environment~liftarmconnect~should~only~consist~of~commands~\protect \liftarm } {} }
    \int_zero:N \l__liftarm_LU_N_int
    \RenewDocumentCommand \liftarm { O {} m m m } { \int_incr:N \l__liftarm_LU_N_int }
    #2
    \fp_set:Nn \l__liftarm_connect_det_fp { \pgf@yy * \pgf@xx - \pgf@yx * \pgf@xy }
    \int_case:nnF { \l__liftarm_LU_N_int }
      {
        { 0 }
          {}
        { 1 }
          {
            \RenewDocumentCommand \liftarm { O {} m m m }
              { \__liftarm_default:nnnn {##1} {##2} {##3} {##4} }
          }
        { 2 }
          {
            \int_zero:N \l__liftarm_connect_count_int
            \int_zero:N \l__liftarm_connect_equation_int
            \seq_clear:N \l__liftarm_connect_coordinate_seq
            \fp_set_eq:NN \l__liftarm_origin_connect_initial_fp \l__liftarm_origin_fp
            \RenewDocumentCommand \liftarm { O {} m m m }
              {
                \int_incr:N \l__liftarm_connect_count_int
                \fp_zero_new:c { l__liftarm_connect_angle_\int_use:N \l__liftarm_connect_count_int _fp }
                \pgfmathparse {##4}
                \fp_set:cn { l__liftarm_connect_angle_\int_use:N \l__liftarm_connect_count_int _fp } { \pgfmathresult }
                \fp_set_eq:NN \l__liftarm_origin_fp \l__liftarm_origin_connect_initial_fp
                \pgfkeys
```

```
      {
        / liftarm / connect_algorithm ,
        coordinate = \pgfkeysvalueof { / liftarm / coordinate } ,
        ##1
      }
    \__liftarm_def_coord:n {##2}
    \fp_set_eq:cN { l__liftarm_connect_two_\int_to_Alph:n { \l__liftarm_connect_count_int }_x_fp } \g__liftarm_coord_x_fp
    \fp_set_eq:cN { l__liftarm_connect_two_\int_to_Alph:n { \l__liftarm_connect_count_int }_y_fp } \g__liftarm_coord_y_fp
    \int_compare:nNnTF { \l__liftarm_connect_count_int } = { 1 }
      {
        \clist_map_inline:en { \pgfkeysvalueof { / liftarm / connect_algorithm / coordinate } }
          {
            \seq_set_split:Nnn \l__liftarm_coordinate_seq { / } {####1}
            \tl_clear_new:c { l__liftarm_connect_two_A_length_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_tl }
            \tl_set:ce { l__liftarm_connect_two_A_length_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_tl }
              { \seq_item:Nn \l__liftarm_coordinate_seq { 1 } - \fp_use:N \l__liftarm_origin_fp }
            \seq_put_right:Ne \l__liftarm_connect_coordinate_seq { \seq_item:Nn \l__liftarm_coordinate_seq { 2 } }
          }
      }
      {
        \clist_map_inline:en { \pgfkeysvalueof { / liftarm / connect_algorithm / coordinate } }
          {
            \seq_set_split:Nnn \l__liftarm_coordinate_seq { / } {####1}
            \seq_if_in:NeT \l__liftarm_connect_coordinate_seq { \seq_item:Nn \l__liftarm_coordinate_seq { 2 } }
              {
                \int_incr:N \l__liftarm_connect_equation_int
                \int_compare:nNnT { \l__liftarm_connect_equation_int } > { 1 }
                  { \PackageError { liftarm } { There~are~too~many~conditions~for~2~liftarms } {} }
                \pgfmathparse
                  { \cs:w l__liftarm_connect_two_A_length_coord_\seq_item:Nn \l__liftarm_coordinate_seq { 2 }_tl\cs_end: }
                \fp_set:Nn \l__liftarm_connect_two_A_length_fp { \pgfmathresult }
                \pgfmathparse { \seq_item:Nn \l__liftarm_coordinate_seq { 1 } }
                \fp_set:Nn \l__liftarm_connect_two_B_length_fp { \pgfmathresult - \l__liftarm_origin_fp }
              }
          }
      }
  }
#2
\fp_set:Nn \l__liftarm_connect_two_length_fp
```

```
    {
      sqrt (
        ( \l__liftarm_connect_two_A_x_fp - \l__liftarm_connect_two_B_x_fp ) ^ 2
        + ( \l__liftarm_connect_two_A_y_fp - \l__liftarm_connect_two_B_y_fp ) ^ 2
      )
    }
  \fp_set:Nn \l__liftarm_connect_two_angle_fp
    {
      atand (
        \l__liftarm_connect_two_B_y_fp - \l__liftarm_connect_two_A_y_fp ,
        \l__liftarm_connect_two_B_x_fp - \l__liftarm_connect_two_A_x_fp
      )
    }
  \fp_set:Nn \l__liftarm_connect_two_A_angle_fp
    {
      acosd (
        (
          ( \l__liftarm_connect_two_A_length_fp ) ^ 2 + ( \l__liftarm_connect_two_length_fp ) ^ 2
          - ( \l__liftarm_connect_two_B_length_fp ) ^ 2
        ) / ( 2 * \l__liftarm_connect_two_A_length_fp * \l__liftarm_connect_two_length_fp )
      )
    }
  \fp_set:Nn \l__liftarm_connect_two_B_angle_fp
    {
      acosd (
        (
          ( \l__liftarm_connect_two_B_length_fp ) ^ 2 + ( \l__liftarm_connect_two_length_fp ) ^ 2
          - ( \l__liftarm_connect_two_A_length_fp ) ^ 2
        ) / ( 2 * \l__liftarm_connect_two_B_length_fp * \l__liftarm_connect_two_length_fp )
      )
    }
  \fp_set:cn { l__liftarm_connect_two_1_option_0_angle_fp }
    { \l__liftarm_connect_two_angle_fp + \l__liftarm_connect_two_A_angle_fp }
  \fp_set:cn { l__liftarm_connect_two_1_option_1_angle_fp }
    { \l__liftarm_connect_two_angle_fp - \l__liftarm_connect_two_A_angle_fp }
  \fp_set:cn { l__liftarm_connect_two_2_option_0_angle_fp }
    { 180 + \l__liftarm_connect_two_angle_fp - \l__liftarm_connect_two_B_angle_fp }
  \fp_set:cn { l__liftarm_connect_two_2_option_1_angle_fp }
    { 180 + \l__liftarm_connect_two_angle_fp + \l__liftarm_connect_two_B_angle_fp }
```

```
        \pgfmathparse
          {
            \__liftarm_Mod:nn { 1 } { 0 } + \__liftarm_Mod:nn { 2 } { 0 }
            >
            \__liftarm_Mod:nn { 1 } { 1 } + \__liftarm_Mod:nn { 2 } { 1 }
          }
        \tl_set:Ne \l__liftarm_tmp_tl { \pgfmathresult }
        \int_zero:N \l__liftarm_connect_count_int
        \RenewDocumentCommand \liftarm { O {} m m m }
          {
            \int_incr:N \l__liftarm_connect_count_int
            \__liftarm_default:nnnn {##1} {##2} {##3}
              { \fp_use:c { l__liftarm_connect_two_\int_use:N \l__liftarm_connect_count_int _option_\l__liftarm_tmp_tl _angle_fp } }
          }
      }
  }
  {
    \int_zero:N \l__liftarm_connect_count_int
    \int_zero:N \l__liftarm_connect_equation_int
    \seq_clear:N \l__liftarm_connect_coordinate_seq
    \fp_set_eq:NN \l__liftarm_origin_connect_initial_fp \l__liftarm_origin_fp
    \RenewDocumentCommand \liftarm { O {} m m m } { \__liftarm_connect:nnnn {##1} {##2} {##3} {##4} }
    #2
    \int_compare:nNnF { \l__liftarm_connect_equation_int } = { \l__liftarm_LU_N_int }
      {
        \PackageError { liftarm }
          {
            The~Jacobian~matrix~is~not~square~
            (the~size~is~\int_use:N \l__liftarm_connect_equation_int \space by~\int_use:N \l__liftarm_LU_N_int )
          } {}
      }
    \int_zero:N \l__liftarm_LU_count_int
    \int_step_inline:nn { \l__liftarm_LU_N_int }
      { \fp_zero_new:c { l__liftarm_LU_b_##1_fp } }
    \__liftarm_connect_stop_criterion:
    \bool_while_do:Nn \l__liftarm_LU_bool
      {
        \int_step_inline:nn { \l__liftarm_LU_N_int }
          {
```

```
            \int_step_inline:nn { \l__liftarm_LU_N_int }
              { \fp_set:cn { l__liftarm_LU_A_##1_####1_fp } { \cs:w l__liftarm_connect_Jacobian_##1_####1_tl\cs_end: } }
          }
        \__liftarm_LU_decomposition:
        \__liftarm_LU_solve:
        \int_step_inline:nn { \l__liftarm_LU_N_int }
          { \fp_sub:cn { l__liftarm_connect_angle_##1_fp } { \cs:w l__liftarm_LU_x_##1_fp\cs_end: } }
        \int_incr:N \l__liftarm_LU_count_int
        \__liftarm_connect_stop_criterion:
      }
    \int_zero:N \l__liftarm_connect_count_int
    \RenewDocumentCommand \liftarm { O {} m m m }
      {
        \int_incr:N \l__liftarm_connect_count_int
        \__liftarm_default:nnnn {##1} {##2} {##3}
          { \fp_eval:n { \cs:w l__liftarm_connect_angle_\int_use:N \l__liftarm_connect_count_int _fp\cs_end: / deg } }
      }
  }
  #2
}
{}

\endinput
```