

# A Wooden Möbius Trefoil Knot

Daniel Goffinet

A solid trefoil knot is constructed in *Mathematica*, and the graphics object is recreated in wood.

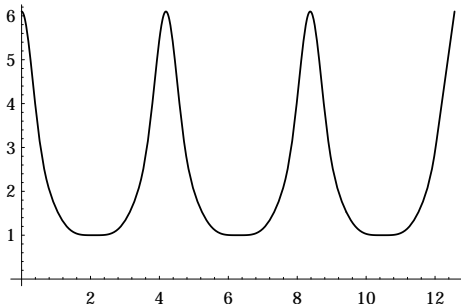
## Motivations

As most of *Mathematica* users, I like that big twisted knot which often greets you when opening a *Mathematica* file or many a book about *Mathematica*. As many others, I decided to make a notebook with “tubing” commands. Unlike most of *Mathematica* users, I am also a marqueteer (not a mousqueteer, but someone who cuts tiny pieces of veneer, usually to inlay pieces of furniture). That combination led me to make an actual wooden veneered knot.

## Drawing the Knot

I had in mind the shape of the knot I wanted, so my first attempt was to use a polar plot to get a plane curve. I would later fit a height function to make my curve go over and under itself. Sound easy? Nobody believes that it can be hard, but the best curve I could get was:

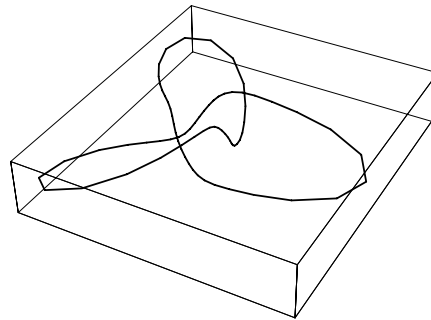
```
In[1]:= r = 1; R = 3;
In[2]:= pointed[u_] := 0.7 u^4 + u^20
In[3]:= rho[t_] := r + R pointed[Cos[3t/4]] // N
In[4]:= Plot[rho[t], {t, 0, 4 Pi}]
```



With the following height function, I indeed had a knot, but not the nice looking one I had dreamed of.

```
In[5]:= z[t_] := Sin[3t/2] // N
In[6]:= p[t_] := {rho[t] Cos[t], rho[t] Sin[t], z[t]}
```

```
In[7]:= ParametricPlot3D[Evaluate[p[t]], {t, 0, 4 Pi},
Axes -> False]
```



I decided to make the three-dimensional curve “by hand” and I started meekly drawing on squared paper. As the shape I wanted had a three-fold symmetry around the line  $x = y = z$ , when putting down the coordinates for my points I discovered that the  $y$  coordinates were merely the  $x$  ones but one third of a period later, and the same for  $z$ : I had only the  $x$  function to define!

With a lot of pencil-and-eraser work I finally got the list I wanted:

```
In[8]:= x = 2.5 {0.9, 0.7, 0, -2.5, -1.85, -0.75, 0.6, 2, 3.5,
1, -0.35, -0.7, -0.6, -0.25, 0.2, 0.7, 0.9, 0.95};
In[9]:= long = Length[x]; third = Quotient[long, 3];
In[10]:= y = RotateLeft[x, third]; z = RotateLeft[y, third];
```

Now comes the smoothing time: a Fourier Fit approximates my data by trigonometric sums.

```
In[11]:= pi = N[Pi];
In[12]:= ptx = Transpose[{2 pi/long Range[0, long - 1], x}];
In[13]:= funs = Flatten[
Table[{Sin[k t], Cos[k t]}, {k, 0, Quotient[long, 2]}]];
In[14]:= cx = Fit[ptx, funs, t]; eqx = Function[t, Evaluate[cx]];
```

As often, instead of getting a nice approximation, you get a rather close one, but with a lot of wriggles coming from the sines of big multiples of  $t$ . So I smoothed it again by averaging my function twice.

Below, the `CleanIt` rule forces *Mathematica* to develop the sums through the sines and cosines, before evaluating, so that it can group together the terms afterwards.

Daniel Goffinet teaches mathematics at Lycee Claude Fauriel of Saint-Etienne, and computing at the E.N.T.P.E., a college of civil engineering in Vaulx-en-Velin. His article on fractals appeared recently in the American Mathematical Monthly. In addition to his work in mathematics and marquetry, he plays the alto sax.

```

In[15]:= << Algebra`Trigonometry`
In[16]:= dt = 0.2; ddt = 0.05;
In[17]:= smooth[t_] := eqx[t - dt] + eqx[t] + eqx[t + dt]
In[18]:= supersmooth[t_] :=
  (smooth[t - ddt] + smooth[t] + smooth[t + ddt])/ 9
In[19]:= CleanIt = {Sin[a_(b_ + c_)] -> Sin[a b + a c],
  Cos[a_(b_ + c_)] -> Cos[a b + a c]};
In[20]:= supersmooth[t] /. CleanIt //
  TrigReduce // Simplify // Expand // Chop;

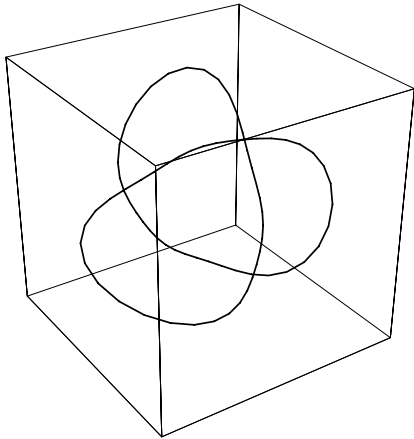
```

Here is my curve:

```

In[21]:= smoothready[t_] := Evaluate[%]
In[22]:= smoothready[t]
Out[22]= 0.618056 - 0.774697 Cos[t] + 2.45214 Cos[2 t] +
  0.359303 Cos[3 t] - 0.101425 Cos[4 t] -
  0.162542 Cos[5 t] - 0.321502 Cos[6 t] +
  0.0821954 Cos[7 t] - 0.0400922 Cos[8 t] +
  0.0507027 Cos[9 t] - 0.508012 Sin[t] - 2.75958 Sin[2 t] +
  1.67713 Sin[3 t] - 0.101057 Sin[4 t] + 0.570234 Sin[5 t] -
  0.275075 Sin[6 t] - 0.0379206 Sin[7 t] - 0.113349 Sin[8 t]
In[23]:= knot[t_] := 0.65 {smoothready[t],
  smoothready[t + 2 pi/3], smoothready[t + 4 pi/3]} // N
In[24]:= ParametricPlot3D[knot[t], {t, 1, 2 pi + 1},
  ViewPoint -> {2, 3, 2}, Axes -> False]

```



## Tubing Commands

I tried it first with the Frenet basis: tangent vector, normal vector, binormal vector. I thought that if I followed the motion of the cross made by the normal and binormal vectors, I would get a square tube around my curve. It works, yes, but in many places, even with a very ordinary curve, the torsion is such that the tube is twisted on a very short distance and it is not at all the kind of tube I wished.

I also tried it with knots on a surface (a torus), using the Darboux basis (the first vector is the tangent, the second one is tangent to the surface and normal to the first one). Instead of sudden complete somersaults, my tube had only sudden half-turns, which is almost as ugly.



I decided to do it “by hand,” which led me to the following commands. First, some general geometric commands: the vector product and a command to normalize vectors:

```

In[254]:= vp[x_, y_] :=
  {x[[2]] y[[3]] - x[[3]] y[[2]],
  x[[3]] y[[1]] - x[[1]] y[[3]],
  x[[1]] y[[2]] - x[[2]] y[[1]]}
In[26]:= makeunit[v_] := v/Sqrt[Apply[Plus, v^2]] // N

```

I choose a constant  $n$  to get my subdivision step, and I sample the points along the curve to get a list of vertices:

```

In[27]:= n = 120;
In[28]:= coreList[n_] := Table[knot[t], {t, 1, 2 pi + 1, 2 pi/n}]

```

The command `cut` takes two vectors as input (these vectors are thought of as the axes of radii 1 tubes) and returns the two half-axes of the ellipse intersection of the tubes.

```

In[29]:= cut[v1_, v2_] :=
  Module[{v1u = makeunit[v1], v2u = makeunit[v2]},
    {makeunit[vp[v1, v2]],
     makeunit[v1u - v2u]/Sin[N[Pi - ArcCos[v1u.v2u]]/2]} ]

```

At each vertex I put the two half-axes of the ellipse section of the two (round) tubes. The `Transpose` associates each cross with the associated vertex.

```

In[30]:= axes[list_] :=
  Module[{ring, vectors},
    ring = Join[{list[[-2]]}, list, {list[[2]]}];
    vectors = Drop[ring - RotateLeft[ring], -1];
    Transpose[
      {list, Apply[cut, Partition[vectors, 2, 1], {1}]} ] ]
In[31]:= axList = axes[coreList[n]];

```

The straighten command assumes that the first cross has the right position and moves the second cross so that it becomes the projection of the first one on the bisector plane of the second vertex, that projection being made parallel to the edge joining the vertices. You can't do it with only one projection! You wouldn't know on which side you have to turn.

```
In[32]:= straighten[{p_, {u1_, u2_}}, {q_, {v1_, v2_}}] :=
Module[{v = q - p, c1, c2, s1, s2},
  {c1, s1} = Drop[LinearSolve[
    Transpose[{v1, v2, v}], v + u1], -1];
  {c2, s2} = Drop[LinearSolve[
    Transpose[{v1, v2, v}], v + u2], -1];
  {q, {{c1, s1}, {c2, s2}} . {v1, v2} }
```

The local variables c1, c2, s1, s2 are cosines and sines. The output {q, {w1, w2}} replaces the old {q, {v1, v2}}.

```
In[33]:= all[z_] := FoldList[straighten, First[z], Rest[z]]
```

After all these projections minimizing the deviation, after a complete turn, what happens? You don't come back where you started! It is the same thing with the "parallel transport" in Riemannian geometry, these differences are used to study the curvature. The difference is calculated below in the angle Arg[u + v I].

```
In[34]:= skewed[z_] :=
Module[
  {a = First[z][[2]], b = Last[all[z]][[2]], u, v},
  {u, v} = LinearSolve[a.Transpose[a], a.b[[1]]];
  Arg[u + v I] Range[0, 1, 1/(Length[z] - 1)] ]
```

The turn command takes the two half-axes and makes a rectangle with them.

```
In[35]:= turn[{p_, {a_, b_}}, t_] := Map[(p + #)&,
  Cos[t] {0, a, b, -a, -b} + Sin[-t] {0, b, -a, -b, a} ]
```

## Fitting the Pieces Together

Everything is ready. I have sampled the points along the curve and built the crosses along the curve in axList. Now I use skewed to know how much error there was. The mobius command makes the list of the angles needed to correct the previous skewness and replaces it by a new skewness which is exactly  $\pi/2$ . (I also tried with  $-\pi/2$ , I got two different objects, I selected the nicer.)

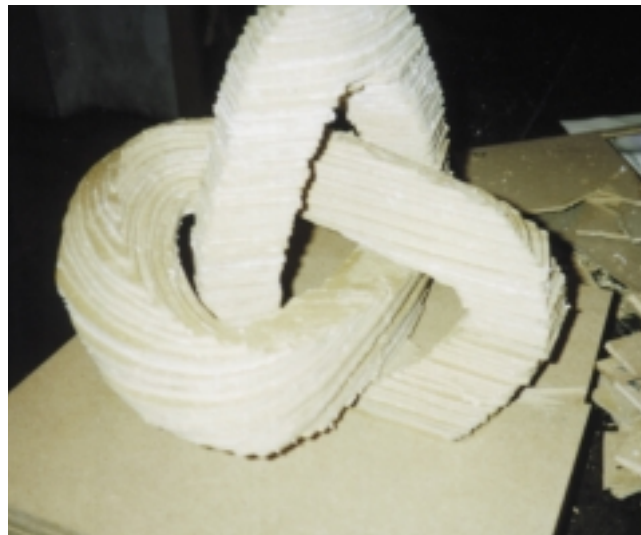
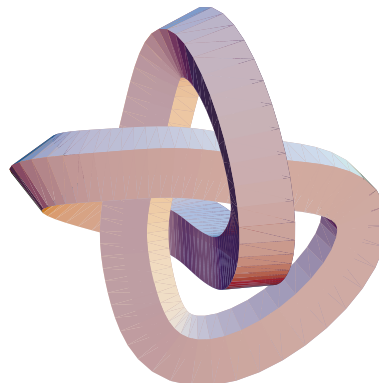
```
In[36]:= mobius[z_] := skewed[z] +
  Range[0, pi/2, pi/2 (Length[z] - 1)]
```

Here is the result: the curve with a cross at each vertex, and it takes you four turns to come back.

```
In[37]:= crosses = Thread[turn[all[axList], mobius[axList]]];
```

And the last preparation: link links two squares by polygons. (The corners of the squares provided must already be in matching order.)

```
In[38]:= link[{p_, q_}] := Map[Polygon,
  Transpose[{p, RotateLeft[p], RotateLeft[q], q, p}]]
In[39]:= solid = Map[Link, Partition[Map[Rest, crosses], 2, 1]];
In[40]:= Show[Graphics3D[{EdgeForm[], solid}],
  ViewPoint -> {-1, -5, 0}, Boxed -> False]
```



## It Seems Over?

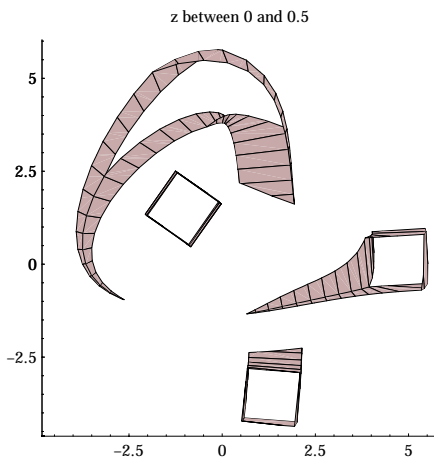
And of course it is not! A long task begins: to slice the virtual object by virtual parallel planes to make actual blueprints, which will be enlarged by photocopying, glued on pieces of wood, cut along the dotted line, glued together, and then begins the real marquetry work (the longest part). I used mahogany, ash, oak, ebony, holy, a gnarl of Norwegian beech, rosewood, some burl of elm, bone glue, and money coins.

How were the virtual cuts done? By an easy *Mathematica* command. I first scaled `coreList` to the size of the cube made of 80 layers of chipwood-like material I used. I got plans of the parts of the knot between planes separated by a height of  $1/2$ , together with labeled axes which served as reference points to know where to glue the photocopies on the boards. The boards were piled up, nailed, and glued (only in the places that were supposed to belong to the knot).

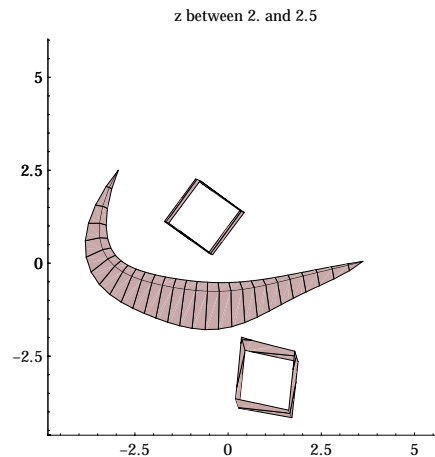
```
In[40]:= label[z_] := "z between " <> ToString[z] <>
         " and " <> ToString[z + 0.5]
```

```
In[41]:= alt[z_] := Show[Graphics3D[solid],
                        PlotRange -> {All, All, {z, z + 0.5}},
                        PlotLabel -> label[z],
                        Boxed -> False, ViewPoint -> {0, 0, 200},
                        AspectRatio -> 1, Axes -> True]
```

```
In[42]:= alt[0]
```



```
In[42]:= alt[2]
```



## The Sentence Inlaid on the Knot

Un arbre bien planté est à moitié scié  
 Une bille bien sciée est à moitié rabotée  
 Une planche bien rabotée est à moitié raclée  
 Un bois bien raclé est à moitié poncé  
 Une pièce bien poncée est à moitié vernie  
 Un meuble bien verni est à moitié vendu  
 Ce qui est bien vendu est à moitié payé  
 Et l'argent encaissé permet de replanter

A well planted tree is half sawn  
 A well sawn bole is half planed  
 A well planed board is half scraped  
 A well scraped wood is half sanded  
 A well sanded piece is half varnished  
 A well varnished cabinet is half sold  
 What is well sold is half paid  
 With the money pocketed you go and seed again

Daniel Goffinet  
 9 rue de la République, 42000 St-Etienne, France



The electronic supplement contains the notebook  
 Trefoil Knot.