Rolling Without Slipping on a Mobius Strip

Alexey Ivanov
stitch_design@rambler.ru

Consider the classical equation of a Mobius strip in parametric form. By using animation shows how movement can occur on the non-oriented surface. We will choose the route as the closed curve belonging to the surface. The sphere was selected as moving geometric object that visually continuously rolls over the surface of a Mobius strip.

Description of Algorithm

At each point of the selected curve we restore the normal to the surface and save points as arrays (La - curve and L - normal). First points belong to the curve on the surface, and the other points lie on the normals at the points of the curve. After for each curve point we build the spatial circle by three points. These points are: another point on the curve, the next point on the curve and the point on the normal to any of these two points. (In this program, we get the points of the circle as the intersection of the sphere of radius r centered at the point on the normal (L) with the plane passing through these three points.) The lengths of the curve segments on the Mobius strip and the lengths of segments of the circle we get equal to h. We will follow that the circle must make a full turn along the curve and then begin to count the points on the circle again. h and r are chosen in such a way that the curve has an integer number of complete revolutions of the circle. To simulate rolling over the surface we mark the two points with different colors on the circle (red and green). At the final figure instead of a circle, we draw a graph of the sphere with the help of Maple standard function, and it turns out that the sphere is rolling without slipping on the Mobius strip.

Code

```
> restart : with(LinearAlgebra) : with(plots) : with( geom3d) :
  smin := 0 : h := 0.075 : smax := evalf(4 π) : hh := 0.5 : r := (smax − smin)/(1000 h);
  f1 := (1 + x1/2 · cos(x2/2)) · cos(x2) :
  f2 := (1 + x1/2 · cos(x2/2)) · sin(x2) :
  f3 := x1/2 · sin(x2/2) :
  fxx1 := unapply( diff(f1, x1), x1, x2) : fxx2 := unapply( diff(f1, x2), x1, x2) :
  fyx1 := unapply( diff(f2, x1), x1, x2) : fyx2 := unapply( diff(f2, x2), x1, x2) :
  fzx1 := unapply( diff(f3, x1), x1, x2) : fzx2 := unapply( diff(f3, x2), x1, x2) :
  La1 := convert([seq((1 + hh/2 · cos(x2/2)) · cos(x2), x2 = smin .. smax, h)], array) :
  La2 := convert([seq((1 + hh/2 · cos(x2/2)) · sin(x2), x2 = smin .. smax, h)], array) :
```

```
La3 := convert([seq((hh/2 · sin(x2/2)), x2 = smin .. smax, h)], array) :
 m := numelems(La3) ;
  Sf := plot3d([f1, f2, f3], x1 =-1 ..1, x2 = 0 ..2 π, style = surface, color = green,
    transparency = 0.4) :

 for i from 1 to m do
  U1 := fxx1(hh, smin + (i − 1) · h) · fyx2(hh, smin + (i − 1) · h) − fxx2(hh, smin + (i
    − 1) · h) · fyx1(hh, smin + (i − 1) · h) :
   U2 := (fyx1(hh, smin + (i − 1) · h) · fzx2(hh, smin + (i − 1) · h) − fyx2(hh, smin
    + (i − 1) · h) · fzx1(hh, smin + (i − 1) · h)) :
    U3 := (fzx1(hh, smin + (i − 1) · h) · fxx2(hh, smin + (i − 1) · h) − fzx2(hh, smin
    + (i − 1) · h) · fxx1(hh, smin + (i − 1) · h)) :
    VN := VectorNorm(⟨U1 , U2, U3⟩, 2) :
 L1[i] := La1[i] + r·U2/VN :  L2[i] := La2[i] + r·U3/VN : L3[i] := La3[i] + r·U1/VN :
 od:
 smin := 0 : smax := evalf(2π·r) :   Nmm := floor(smax/h) + 1;

    ############################################################################\
    #####################

    ############################################################################\
    #####################


   jk := 0;
   for jj to m − 1 do
   xx1 := La1[jj]; xx2 := La2[jj]; xx3 := La3[jj];
   xxx1 := L1[jj]; xxx2 := L2[jj]; xxx3 := L3[jj];

 point(O1, xxx1, xxx2, xxx3);
 point(O2, xx1, xx2, xx3) :
 if jj = m then point(O3, La1[jj − 1], La2[jj − 1], La3[jj-1]) else point(O3, La1[jj
    + 1], La2[jj + 1], La3[jj + 1]) : fi:

  plane(VOP1, [O1, O2, O3], [x1, x2, x3]) :


  f1 := (x1 − xxx1)^2 + (x2 − xxx2)^2 + (x3 − xxx3)^2 − r^2 :
  f2 := lhs(Equation(VOP1)) :
 ff11 := unapply( diff (f1, x1), x1, x2, x3) :ff12 := unapply( diff (f1, x2), x1, x2, x3) :
    ff13 := unapply( diff (f1, x3), x1, x2, x3) :
x01 := xx1 : x02 := xx2 : x03 := xx3 :
 n := 2 :
 x := seq(eval(cat('x', i)), i = 1 .. n + 1) : x0 := seq(eval(cat('x0', i)), i = 1 .. n + 1) :
 F := [seq(unapply(eval(cat('f', i)), [x]), i = 1 .. n)] : A := MTM:-jacobian(F(x), [x]) :
    A := MTM:-subs(A, [x], [x(s)]) :

 for i to n do b[i] := simplify(Determinant(DeleteColumn(ColumnOperation(A, [i, n
```

```
+ 1]), n + 1))) od:
        b[n + 1] := simplify( -Determinant(DeleteColumn(A, n + 1))) :
        deqs := seq(diff(x[i](s), s) = b[i]/(b[1]^2 + b[2]^2 + b[3]^2)^0.5, i = 1 ..n + 1):
ics := seq(x[i](0) = x0[i], i = 1 .. n + 1) :
slnn := dsolve([deqs, ics], numeric, method = rkf45, abserr = h/10, maxfun = 10000, range
   = smin .. smax) :
  for j from 1 to n + 1 do   Lg[j] := convert([seq(rhs(slnn(i)[j + 1]), i = smax ..smin,
   -h)], array) : od:


   mm := numelems(Lg[1]);   jk := jk + 1 : if  jk = Nmm then jk := 1 : fi:
 VN := VectorNorm(⟨ff11(Lg[1][jk], Lg[2][jk], Lg[3][jk]) ,  ff12(Lg[1][jk],
   Lg[2][jk], Lg[3][jk]), ff13(Lg[1][jk], Lg[2][jk], Lg[3][jk])⟩, 2) :

  SS[jj] := plottools[sphere]([xxx1, xxx2, xxx3], r + 0., style = surface, color = gray) :
 MP[jj] := pointplot3d([[Lg[1][jk], Lg[2][jk], Lg[3][jk]], [Lg[1][jk]
   - (2· r - 0.0) ·ff11(Lg[1][jk], Lg[2][jk], Lg[3][jk])/VN, Lg[2][jk]
   - (2· r - 0.0) ·ff12(Lg[1][jk], Lg[2][jk], Lg[3][jk])/VN, Lg[3][jk]
   - (2· r - 0.0) ·ff13(Lg[1][jk], Lg[2][jk], Lg[3][jk])/VN]], color = [green, red],
   style = point, symbol = solidsphere, symbolsize = 4) :


       od: mm;

  Se := seq(display(Sf, SS[k], MP[k]), k = 1 ..m - 1) :

 display(Se, view = [-2 ..2, -2 ..2, -2 ..2], insequence = true, axes = normal);
```