DGeometrica

A Mathematica package for differential geometry

Yoshihiko TAZAWA

School of Information Environment
Tokyo Denki University
2−1200 Muzai−gakuendai
Inzai, Chiba Pref. 270−1382
JAPAN
tazawa@cck.dendai.ac.jp

■ 1. Abstract

This notebook is an introduction of the first half of **DGeometrica**, a package for doing differential geometry of curves and surfaces by *Mathematica*.

The geometry of curves and surfaces is a nice topic to deal with computer algebras. Alfred Gray's book [5] is a comprehensive introduction of how to utilize *Mathematica* in differential geometry. He gave a tutorial talk [1] in the first IMS. His *Mathematica* package is still downloadable from [4]. John Oprea published a book [9] based on Maple. Richard Palais and his group created a freeware 3D−XplorMath [5] which works on Macintosh OS and deals with many topics from differential geometry.

In the third IMS the author gave a talk [2] and introduced several examples from the book [10] written in Japanese. The main part of **DGeometrica** is a set of ideas and *Mathematica* commands contained in the [10]. The feature of **DGeometrica** compared to other packages is that the usage of numerical functions such as **NDSolve** or **NIntegrate** to do experimental approach and to extend the range of examples in differential geometry of curves and surfaces.

This article consists of three materials: the present notebook, **DGeometrica** package files contained in a folder named DGeometrica, and a pair of *Mathematica* notebooks **Plane− CurvesTest.nb** and **SpaceCurvesTest.nb**. Set the folder DGeometrica in the standard package folder of *Mathematica*, and input all commands to obtain usage explanations and output of all the commands contained in the package. All commans in **`DGeometrica`** start with the prefix **`dg−`** like **`dgPlot`**. **`dg−`**commands with **`Anim`** like **`dgPlotAnim`** returns an animation, and **`dg−`**commands with **`Fig`** like **`dgCatenaryFig`** returns a figure. **`dg−`**commands without **`Anim`** or **`Fig`** like **`dgCurvature`** are just for calculation.

The present notebook is an outline of the package and introduces several commands in it. All Input cells in this notebook are set non−evaluatable in order to make the appearance stable. The **`dg−`**commands should be input in **PlaneCurvesTest.nb** and **SpaceCurvesTest.nb**.

■ 2. Plane curves

First, we open the package **`'PlaneCurves'`**.

In[1]:= **<<DGeometrica'PlaneCurves'**

□ 2.1 Plane curves

A plane curve is a mapping from an interval $I \subset \mathbb{R}$ into the two dimensional Euclidian plane $\mathbb{R}^2$

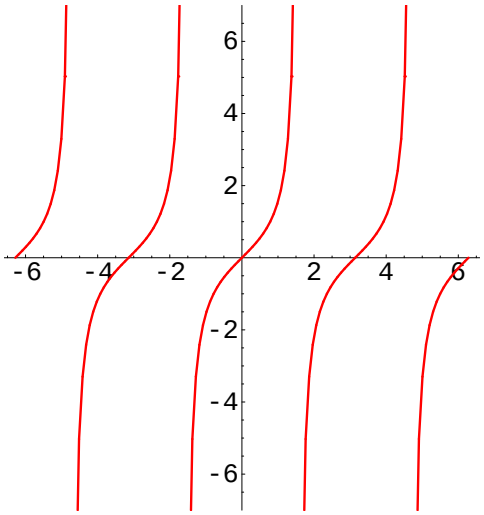
$$c: I \longrightarrow \mathbb{R}^2 ; c(t) = (x(t), y(t)), t \in I,$$

where $x(t)$ and $y(t)$ are differentiable functions of class at least C^2 . If the curve is of the form $c(x)=(x, f(x))$, namely the graph of a function $y = f(x)$, it is plotted by **Plot** of *Mathematica*, but it fails to show the discontinuity of functions. For a function with discontinuity, we can use **dgPlot** as follows.

In[2]:= **?dgPlot**

```
dgPlot[f[x],{x,x0,x1},n,rgb,thk,opts] plots the graph of
  the function f[x], x0<=x<=x1, jointing n segments,
  with RGBColor[rgb], Thickness[thk], and Options opts.
  Segments with slope greater than 1000 are not shown,
  therefore, if the function has points of discontinuity,
  the curve is plotted discontinuous at those points.
```

In[4]:= **dgPlot[Tan[x],{x,−2*Pi,2*Pi},128,{1,0,0},**
0.005,Axes−>True,AspectRatio−>Automatic,PlotRange−>{−7,7}];



Likewise, we can use **dgParametricPlot** for parametrized curves with possible points of discontinuity;

In[5]:= **?dgParametricPlot**

```
dgParametricPlot[c[t],{t,t0,t1},n,rgb,thk,opts] plots
  tthe parametrized curve c[t], t0<=t<=t1, jointing n
  segments, with RGBColor[rgb], Thickness[thk], and
  Options opts. Segments with length greater than
  the 0.1 times width or the 0.1 times height of the
  box containing the curve are not shown. Therefore,
  if the curven has points of discontinuity, the
  curve is plotted discontinuous at those points.
```

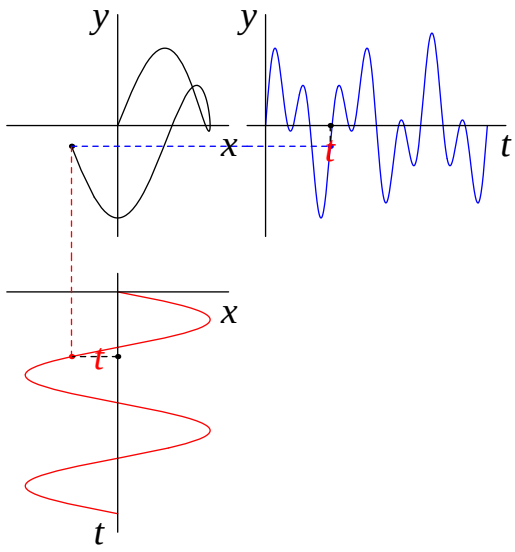
dgPlotAnim returns an animation that shows how a parametrized curve is generated by given component functions $x(t)$ and $y(t)$.

In[6]:= **?dgPlotAnim**

```
dgPlotAnim[c,a,b] returns an animation
  of a curve c[t]={x[t],y[t]},    a<=t<=b,
  together with the curves x=x[t] and y=y[t].
```

In this notebook, all animations are represented by a single scene or several scenes as follows. The full animation can be observed by inputting the commands in **PlaneCurvesTest.nb** and **SpaceCurvesTest.nb**.

In[7]:=
```
c[t_]={Sin[2t],Sin[5t]*Cos[2t]};
dgPlotAnim[c,0,2Pi];
```



Similar animations are obtained by **dgPolarPlotAnim** for curves with polar parametrization.

2.2 Well−known curves

Some well−known curves are contained in the package. Some of them are shown below.

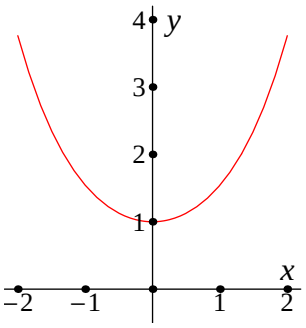
In[9]:= **?dgCatenaryFig**

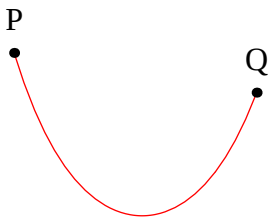
```
dgCatenaryFig returns a figure of a catenary together
  with  the parametric expression of a catenary.
```

In[10]:= **dgCatenaryFig**

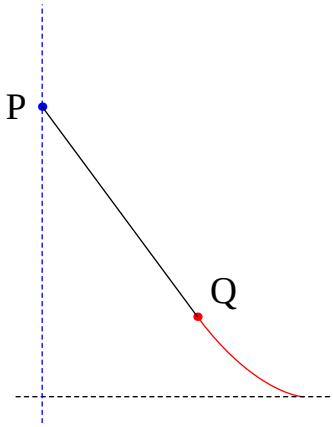
Catenary: c[t]={t,a∗Cosh[t/a]}

Fig: catenary with a=1, −2<=t<=2





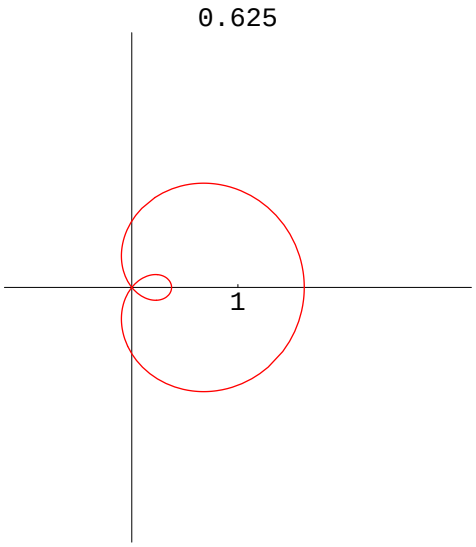
In[11]:= **dgTractrixAnim;**



In[14]:= **?dgLimaconAnim**

```
dgLimaconAnim[b0,b1] returns the animation of the limacons
  c[t]=(Cos[t]+b)*{Cos[t],Sin[t]}, 0<=t<=2Pi, b0<=b<=b1.
  The value of the marameter b is also printed at the top.
```

In[15]:= **dgLimaconAnim[0.5,2]**



□ 2.3 Generic curves

Traditionally, examples in differential geometry of curves and surfaces are confined to a small group of calculable objects: integration of the functions, like the curvature function, are beyond our hand−calculation except for some simple curves. The main purpose of using *Mathematica* in differential geometry is, I suppose to utilize its numerical functions like **NIntegrate** or **NDSolve** to extend the range of examples. If we know the parametric expression of curves or surfaces, we can treat them differential geometrically, even if the expression is fairly complicated.

This section shows how to generate plane curves randomly together with their expressions. We use these curves fully in this package. Space curves and surfaces are also generated randomly.

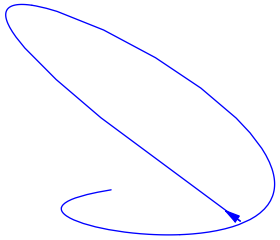
The idea is to generate first finite points on the plane by **Random** and find expression of the curve approximating the points by **Fit**. If we use **FourierTrigSeries** we can obtain periodic expression of closed curves.

In[16]:= **?dgRandomCurve**

```
dgRandomCurve[a,m,n] returns a curve c(t)=(x(t),y(
  t)) approximating m rondom points in (-a,a)×(-a,a),
  where x(t) and y(t) are polynomials of t of degree
  n. A small arrow shows the orientation of the curve.
```

In[17]:= **dgRandomCurve[1,6,5]**

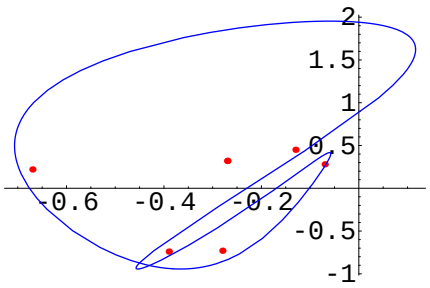
$$\{-96.2706\,t^5 + 324.738\,t^4 - 384.211\,t^3 + 185.808\,t^2 - 30.9683\,t + 0.612926,$$
$$31.6692\,t^5 - 119.345\,t^4 + 165.748\,t^3 - 99.4505\,t^2 + 21.5974\,t - 0.215208\}$$



In[18]:= **?dgRandomClosedCurve**

```
dgRandomClosedCurve[a,m,n] returns a closed curve c(t)=(x(
  t),y(t)) approximating m rondom points in (-a,a)×(-a,a),
  where x(t) and y(t) are periodic functions expressed
  as Fourier trigonometric series expansion to order n.
```

In[19]:= **dgRandomClosedCurve[1,6,6,3];**



$$\{0.04\cos(2(t+0.5)\pi) + 0.18\cos(4(t+0.5)\pi) +$$
$$0.14\sin(2(t+0.5)\pi) - 0.22\sin(4(t+0.5)\pi) + 0.08\sin(6(t+0.5)\pi) - 0.28,$$
$$-0.94\cos(2(t+0.5)\pi) + 0.93\cos(4(t+0.5)\pi) + 0.18\cos(6(t+0.5)\pi) +$$
$$0.07\sin(2(t+0.5)\pi) - 0.08\sin(4(t+0.5)\pi) + 0.03\sin(6(t+0.5)\pi) + 0.24694\}$$

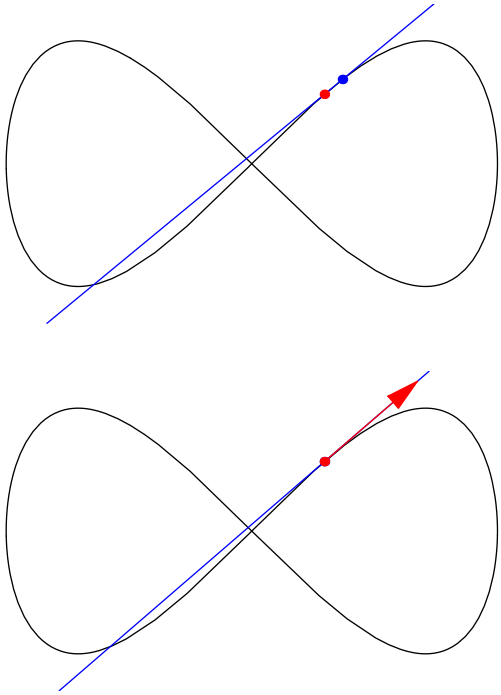
□ 2.4 Arclength and moving frames

For a curve $c(t)=(x(t), y(t))$, its derivative is obtained by differentiating component−wise: $c'(t) = (x'(t), y'(t))$. If $c'(t_0) \neq 0$ at a point $c(t_0)$, then the tangent line is determined to be the limit of the straight line passing through $c(t_0)$ and a close point $c(t)$ on the curve as $t \to t_0$. The tangent line is spanned by the tangent vector $c'(t_0)$, as can be seen in the following animation.

In[20]:= **?dgTangentLineAnim**

```
dgTangentLineAnim[c,t0,h0,a,b] returns an animation
  of the straight line passing through a fixed point
  c[t0] and a moving point c[t0+h] on a curve c[t],
  a<=t<=b, as h tends from h0 to 0. The limit of the
  lines is the tangent line of c[t] at c[t0]. The red
  arrow is the unit tangent vector of c[t]  at c[t0].
```

In[23]:=
```
c[t_]=2*{Sin[t],Sin[t]*Cos[t]};
dgTangentLineAnim[c,0.3,2,0,2Pi]; (* case h0>0 *)
```



The length of a curve $c(t)=(x(t), y(t))$, $a \le t \le b$, is defined by the integration $\int_a^b |c'(t)|\,dt$, and we can calculate it by **dgArcLength**. We can calculate the length of a fairly complicated curve by **dgNumArcLength** numerically.

In[31]:=
```
c[t_]=
 {1.18445+0.215722*t-10.2437*t^2-13.3893*t^3+
   25.9661*t^4+30.553*t^5-36.084*t^6,
  -0.6248+6.4332*t+0.151186*t^2-52.3013*t^3+
   11.1881*t^4+139.806*t^5-105.033*t^6};
dgNumArcLength[0,1][c]
```

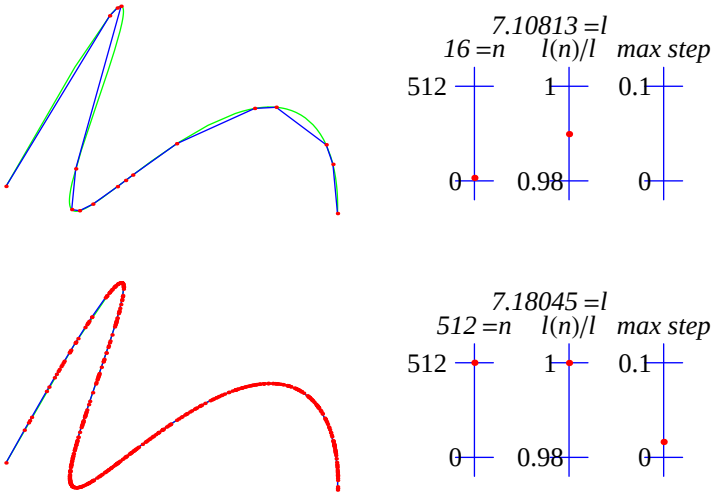
Out[32]= 7.18077

The geometric meaning of the length of a curve is the limit of the length of the broken line jointing points on the curve, as the number of the points tends to infinity. We can see it in the following animation:

In[33]:=
```
?dgArcLengthAnim
```

```
dgArcLengthAnim[c,a,b,m] returns an animation of the
  approximation of the curve c[t],a<=t<=b, by broken lines
  connecting up to 2^m points on the curve, together with
  the number of the points, length of the broken lines,
  the ratios of the length of the brolen lines by the
  length of the curve, and the maximal length of the
  subintervals. The length of the broken line tends to the
  length of the curve as the number of the points increases.
```

In[34]:=
```
c[t_]=
 {1.18445+0.215722*t-10.2437*t^2-13.3893*t^3+
   25.9661*t^4+30.553*t^5-36.084*t^6,
  -0.6248+6.4332*t+0.151186*t^2-52.3013*t^3+
   11.1881*t^4+139.806*t^5-105.033*t^6};
dgArcLengthAnim[c,0,1,9];
```

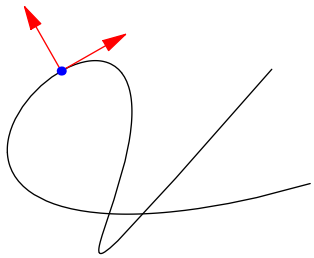


If the derivative $c'(t_0)$ does not vanish anywhere, then the curve $c(t)$ is called a regular curve, and we define the unit tangent vector field $e_1(t)=\frac{c'(t)}{|c'(t)|}$ along the curve $c(t)$. We also define the unit normal vector field $e_2(t)$ by rotating $e_1(t)$ in the positive direction by the angle $\frac{\pi}{2}$. The pair $\{e_1(t), e_2(t)\}$ is called the moving frame of the curve $c(t)$.

In[38]:=
```
?dgMovingFrameAnim
```

```
dgMovingFrameAnim[c,a,b,n] returns an animation of the
  moving frame of a curve c(t), a<=t<=b, with n scenes.
```

In[36]:=
```
c11[t_]={
 3.93+2.97*(1/4+t)-34.70*(1/4+t)^2-142.11*(1/4+t)^3+
 530.59*(1/4+t)^4-535.24*(1/4+t)^5+173.94*(1/4+t)^6,
1/2*(1.01-2.35*(1/4+t)-29.04*(1/4+t)^2-71.76*(1/4+t)^3+
497.63*(1/4+t)^4-635.64*(1/4+t)^5+239.683*(1/4+t)^6)};
dgMovingFrameAnim[c11, 0, 1, 48]
```



2.5 Curvature

The length of a portion of the curve c from the stating point $c(a)$ to a point $c(t)$ on the curve is a function of t which we denote by $s=s(t)=\int_a^t |c'(t)|\,dt$. For a regular curve, $s=s(t)$ is a increasing function and has its inverse function $t = t(s)$. Hence, we can take s as a parameter of c. In this case, s is called the arclength parameter of c.

If c is parametrized by the arclength as $c(s)$, then $c'(s)$ is a unit vector field and is equal to $e_1(s)$. The curvature function $\kappa(s)$ of $c(s)$ is defined by $\kappa(s) = e'_1(s)\cdot e_2(s)$, and the so called Frenet's Formula holds.

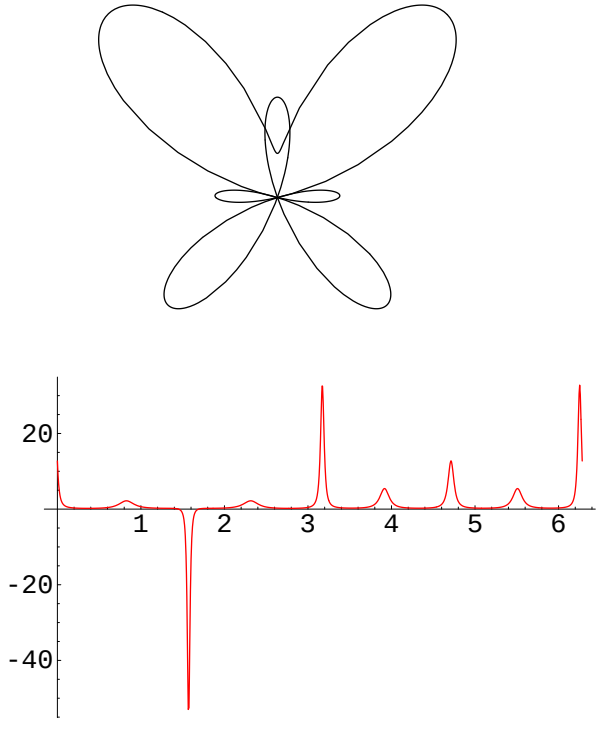
$$\begin{aligned} c'(s) &= e_1(s) \\ e'_1(s) &= \kappa(s)\, e_2(s) \\ e'_2(s) &= -\kappa(s)\, e_1(s) \end{aligned}$$

The curvature is calculated by **dgCurvature.**

In[41]:= **?dgCurvature**

```
dgCurvature[c][t] calculates the
  curvature function of a paramertrized curve c[t].
```

In[46]:=
```
butterflyxxyy[t_]=
  (Exp[Sin[t]]−2Cos[4t])*{Cos[t],Sin[t]};
ParametricPlot[butterflyxxyy[t]//Evaluate,{t,0,2Pi},
  AspectRatio−>Automatic,Axes−>False];
Plot[dgCurvature[butterflyxxyy][t],{t,0,2Pi},PlotStyle−>RGBColor[1,0,0],PlotRange−>
All];
```

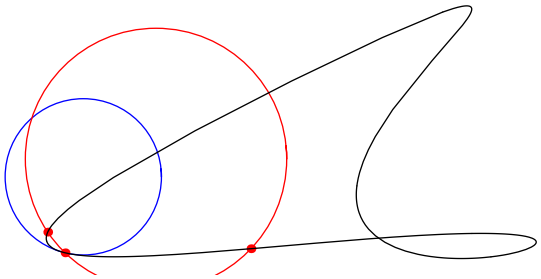


Geometric meaning of the curvature is as follows. Take a point $c(a_0)$ on the curve c, and two other points $c(a_0 - h)$ and $c(a_0 + k)$. The limit of the circle passing through the three points as $h\to 0$ and $h\to 0$ is called the osculating circle of c at $c(a_0)$. The osculating circle

approximates the curve around $c(a_0)$ best among the circles tangential to the curve at $c(a_0)$, and its radius $\rho(a_0)$ is equal to $\frac{1}{\kappa(a_0)}$, if $\kappa(a_0) \neq 0$. **dgOsculatingCircleAnim** show the definition of the osculating circle, and **dgurvatureAnim** shows the relation between the curvature and the osculating circle, and how the circle is curved.

In[52]:=
```
c[t_] ={-2.17*Cos[t]-1.95*Cos[2t]+0.44*Sin[t]-0.05*Sin[2t],
    -0.22*Cos[t]-0.72*Cos[2t]-1.57*Sin[t]+.22*Sin[2t]};
```

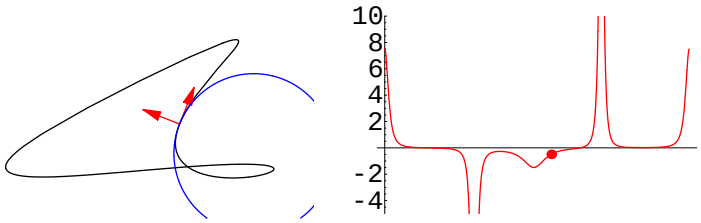
In[53]:=
```
dgOsculatingCircleAnim[c,Pi/16,Pi/6,Pi/3,0,2Pi,24];
```



In[54]:=
```
?dgCurvatureAnim
```

```
dgCurvatureAnim[c,a,b,num,opts] returns the animation
  of the moving frame, the osculating circle, and the
  curvature function of a curve c[t], a<=t<=b with num
  scenes. opts are the options for the curvature graph.
```

In[55]:=
```
dgCurvatureAnim[c,0,2Pi,64,PlotRange->{-5,10}];
```



The Gauss map γ of a curve $c: I \longrightarrow \mathbb{R}^2$ is a mapping from I to the unit circle $S^1 \subset \mathbb{R}^2$ defined by

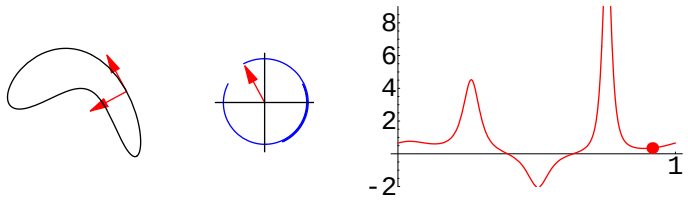
$$\gamma : I \longrightarrow S^1 \subset \mathbb{R}^2 ; \gamma(t)=e_1(t)\ t \in I,$$

where the starting point of the unit vector $\gamma(t)$ is set to be the origin O of $\mathbb{R}^2$.
dgGaussMapAnimt shows the relation between the curvature and the Gauss map.

In[56]:=
```
?dgGaussMapAnim
```

```
dgGaussMapAnim[c,t0,t1,opts] returns the animation of the
  moving frame,Gauss map, and the curvature of the curve c[
  t], t0?t?t1 with options opts for the curvature graphics.
```

In[57]:=
```
c[t_] = {
   0.0634*Cos[2Pi*t] + 0.203*Cos[4Pi*t] −
    1.54*Sin[2Pi*t] − 0.228*Sin[4Pi*t],
   0.419*Cos[2Pi*t] + 0.708*Cos[4Pi*t] +
   0.647*Sin[2Pi*t] + 0.186*Sin[4Pi*t]};
dgGaussMapAnim[c,0,1,PlotRange−>{−2,9}];
```

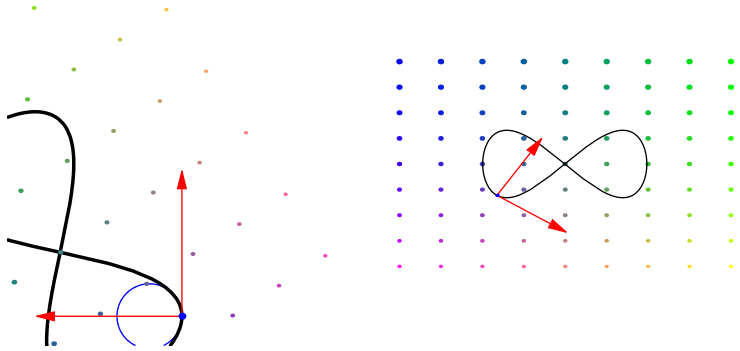


The Frenet's Formula implies how the shape of the curve changes in terms of moving frame, in other words, how the curve is observed from the view point fixed to the moving frame. **dgViewFromMovingFrameAnim** shows it.

In[59]:=
```
?dgViewFromMovingFrameAnim
```

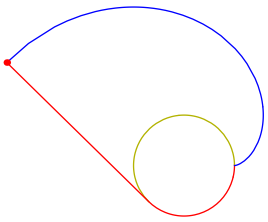
```
dgViewFromMovingFrameAnim[c,t0,t1,n]
  returns the animation of the scene viewed from
  the moving frame of the curve c[t], t0<=t<=t1,
  and the blue osculating circle with n scenes.
```

In[60]:=
```
cc[t_] = {Sin[t + 0.01], Sin[t + 0.01] * Cos[t + 0.01]};
dgViewFromMovingFrameAnim[cc, 0, 2 Pi, 64];
```

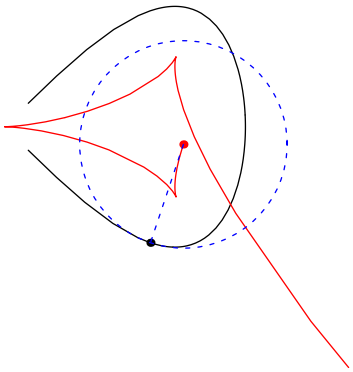


Involutes and evolutes are examples to create curves from given curves, and they are related to the curvature function.

In[62]:=
```
c[t_]:={Cos[t],Sin[t]};
dgInvoluteAnim[c,0,2Pi,0,24];
```



In[64]:=
```
eight[t_]:={Sin[t],Cos[t]*Sin[t]};
dgEvoluteAnim[eight,Pi/32,Pi−Pi/32,16,−0.2,1.5,−1,1]
```



□ 2.6 Fundamental Theorem of plane curves

Plane curves are determined by a given curvature function as stated in the following.

Fundamental Theorem of Plane Curves. *Let $\kappa(s)$ be a continuous function of s. Then, there exist plane curves c(s), such that s is the arclength parameter of c, and the curvature function of c(s) is equal to the given function $\kappa(s)$. These curves are all congruent under isometries of $\mathbb{R}^2$, therefore such a curve is uniquely determined by the initial condition $c(s_0)=p_0$, $c'(s)=v_0$.*

Such a curve described in the theorem is obtained by regarding the Frenet's Formula as a system of differential equations and by solving it. We can utilize **NDSolve** of *Mathematica* for this purpose, to obtain **dgCurveByCurvature**.

In[66]:= **?dgCurveByCurvature**

```
dgCurveByCurvature[kappa,{s,s0,s1},{x0,y0},theta0] returns
  the curve c[s], s0<=s<=s1, where s is the arclength,
  kappa is the curvature function of c[s], c[s0]={x0,y0},
  and theta0 is the argument of the direction c'[s0].
```

In[67]:= **c[s_]=dgCurveByCurvature[0.5*s*Sin[s],{s,0,4*Pi},{0,0},0];**

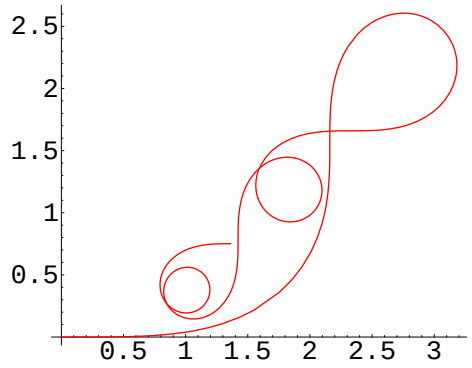
In[68]:= **c[s]**

Out[68]= {InterpolatingFunction[{ 0. 12.5664 }, <>][*s*],
 InterpolatingFunction[{ 0. 12.5664 }, <>][*s*]}

In[69]:= **c[4]**

Out[69]= {2.33586, 2.38073}

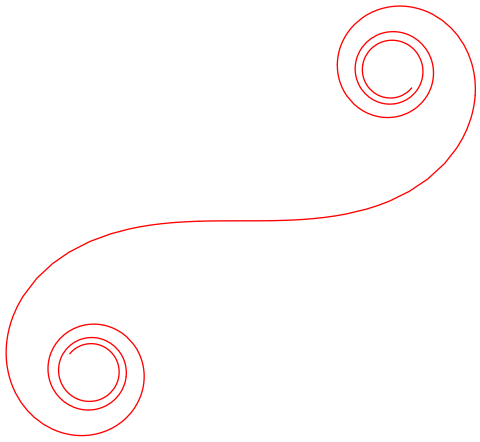
In[70]:= **ParametricPlot[c[s],{s,0,4*Pi},AspectRatio->Automatic,PlotRange->All,PlotStyle->RGBColor[1,0,0]]**



Out[70]= - Graphics -

More practically, we can input as follows.

In[72]:= **dgPlotByCurvature[s,{s,-2Pi,2Pi},{0,0},0,PlotStyle->RGBColor[1,0,0],Axes->False];**



From geometrical view point, to determine a curve by the curvature function can be said to approximate the curve by jointing small pieces of circles whose radii are determined by the curvature function κ, in other words, to approximate the curve locally around a

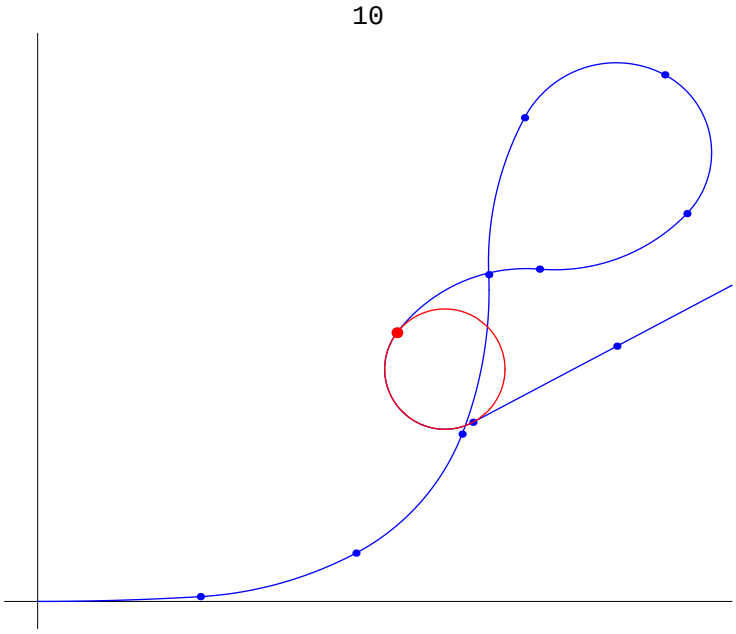
point $c(a_0)$ by a circle tangential to the curve at this point whose radius is $\frac{1}{\kappa(a_0)}$, and jointing such small segments of circles.

We can observe the above by **dgJointCircleSegmentsAnim** as follows. This shows how we can use *Mathematica* as a tool for *experimental approach* to mathematics, which is one of the important features of this package.

In[73]:= **?dgJointCircleSegmentsAnim**

```
dgJointCircleSegmentsAnim[f,{s,s0,s1},n] returns the
  animation of the approximation of the curve  c(s) ,s0<=
  s<=s1,determined by the curvature function f of s and
  the initial condition c(0)=(0,0) and c'(0)=(0,0), where
  the approximation is obtained by joiting n small pieces
  of circles. The solution curve  c(s) is also plotted in
  green color, which is obtained by solving the system
  of differential equations in the Fundamental Theorem.
```

In[74]:= **dgJointCircleSegmentsAnim[0.5*s*Sin[s],{s,0,4*Pi},16]**



□ 2.7 Global properties of plane curves

The total curvature of a closed plane curve $c(s)$, $a \le s \le b$, parametrized by the arclength is defined by the integration $\int_a^b \kappa(s)\, ds$. It is equal to 2π times the rotation number of c, namely how many times the tangent vector rotate when it goes along the curve from the starting point to the end point, that is in this case equal to the starting point.

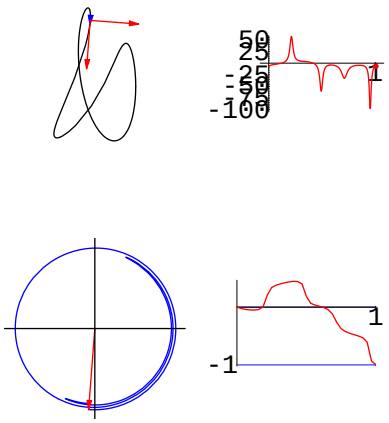
The total absolute curvature of $c(s)$ is defined by $\int_a^b |\kappa(s)|\, ds$.

dgTotalCurvatureAnim shows how the value of the total curvature is determined.

In[75]:= **?dgTotalCurvatureAnim**

```
dgTotalCurvatureAnim[c[t],{t,t0,t1},n] returns
  an animation of the curve c(t), t0 <= t <= t1,
  with n scenes, together with the moving frame,
  the  curvature function, Gauss map, and the
  integration of the curvature. The last value of the
  integration of the curvature is the total curvature.
```

In[76]:= **c11[t_]={−0.02+0.54Cos[2Pi(0.5+t)]+0.28Cos[4Pi(0.5+t)]−0.16Cos[6Pi(0.5+t)]+0.37Sin[2Pi(0.5+t)]+0.08Sin[4Pi(0.5+t)]−0.03Sin[6Pi(0.5+t)],0.128036−0.29Cos[2Pi(0.5+t)]+1.06Cos[4Pi(0.5+t)]+0.05Cos[6Pi(0.5+t)]+0.19Sin[2Pi(0.5+t)]−0.45Sin[4Pi(0.5+t)]+0.16Sin[6Pi(0.5+t)]};**
dgTotalCurvatureAnim [c11[t],{t,0,1},32];



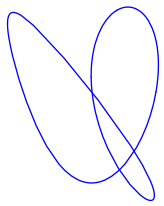
dgTotalCurvatureAnim shows how the value of total curvature is determined.

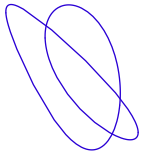
Since the total curvature is equal to 2π times of an integer, its value is discrete and therefore stable under a homotopy of the closed curve, if all homotopy curves are regular curves. First, we show an example of a homotopic deformation of closed curves.

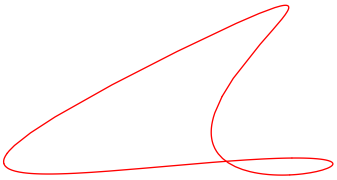
In[78]:= **?dgHomotopyAnim**

```
dgHomotopyAnim[c11[t],c22[t],{t,t0,t1},num] returns
  the animation of the linear homotopy between
  c11(t) and c22(t), t0<=t<=t1, with num+1 scenes.
```

In[79]:=
```
c22[t_]={
 0.51Cos[2Pi*t]−0.31Cos[4Pi*t]+0.026Cos[6Pi*t]−
 0.17Sin[2Pi*t]+0.34Sin[4Pi*t]+ 0.12Sin[6Pi*t],
0.001Cos[2Pi*t]+0.93Cos[4Pi*t]+0.011Cos[6Pi*t]+
 0.13Sin[2Pi*t]+0.12Sin[4Pi*t]+0.042Sin[6Pi*t]};
c33[t_]:={
 −1.08*Cos[2*Pi*t]− 0.974*Cos[4*Pi*t]+
 0.223*Sin[2*Pi*t]−0.0267*Sin[4*Pi*t],
 −0.11*Cos[2*Pi*t]− 0.361*Cos[4*Pi*t]−
 0.783*Sin[2*Pi*t]+ 0.11*Sin[4*Pi*t]}
dgHomotopyAnim[c22[t],c33[t],{t,0,1},12];
```







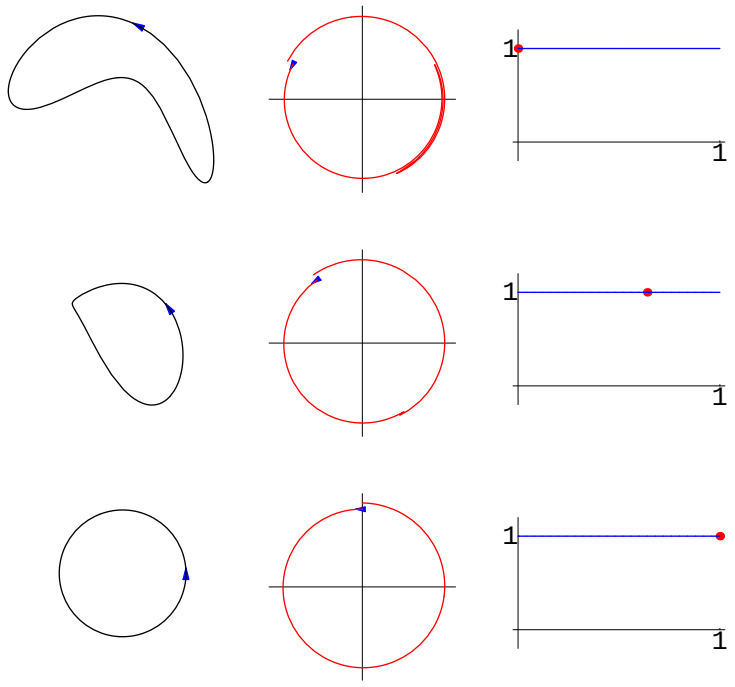
Here our homotopy between c_1 and c_2 is obtained linearly, i.e.,$c_s(t) = (1-s)c_1(t) + sc_2(t)$, so some of the homotopy curves c_s might fail to be regular. Since the total curvature is equal to 2π times of an integer, its value is discrete and therefore stable under a homotopy of the closed curve, if all homotopy curves are regular curves.

dgTotalCurvatureUnderHomotopyAnim show the stability of the value of the total curvature under homotopy.

In[82]:= **?dgTotalCurvatureUnderHomotopyAnim**

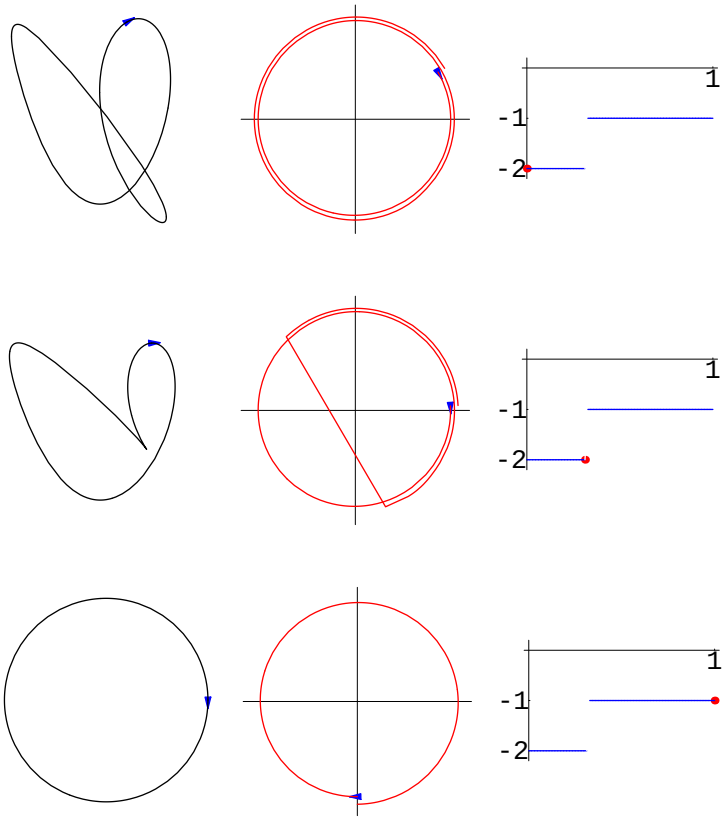
```
dgTotalCurvatureUnderHomotopyAnim[c11[t],c22[
  t],{t,t0,t1},n] returns the animation of the
  homotopy between two closed curves cur11(t) and
  cur22(t) , t0<=t<=t1, together with the change
  of Gauss map and total curvatur, with n+1 scenes.
```

In[83]:=
```
curve11 = {
 −0.029 + 0.0634*Cos[2Pi*t] + 0.203*Cos[4Pi*t] −
       1.54*Sin[2Pi*t] − 0.228*Sin[4Pi*t],
  −0.177 + 0.419*Cos[2Pi*t] + 0.708*Cos[4Pi*t] +
       0.647*Sin[2Pi*t] + 0.186*Sin[4Pi*t]};
circlepos = {Cos[2*Pi*t], Sin[2*Pi*t]};
dgTotalCurvatureUnderHomotopyAnim[curve11,circlepos,{t,0,1},25];
```



The following animation shows that if some of the homotopy curves are not regular, the value of total curvature fails to be regular.

In[86]:=
```
curve22={0.51*Cos[2*Pi*t]−
0.31*Cos[4*Pi*t]+
0.026*Cos[6*Pi*t]−0.17*Sin[2*Pi*t]+
0.34*Sin[4*Pi*t]+
0.12*Sin[6*Pi*t],
0.001*Cos[2*Pi*t]+0.93*Cos[4*Pi*t]+
0.011*Cos[6*Pi*t]+
0.13*Sin[2*Pi*t]+
0.12*Sin[4*Pi*t]+
0.042*Sin[6*Pi*t]};
circleneg={Cos[−2*Pi*t],Sin[−2*Pi*t]};
dgTotalCurvatureUnderHomotopyAnim[curve22,circleneg,{t,0,1},64];
```



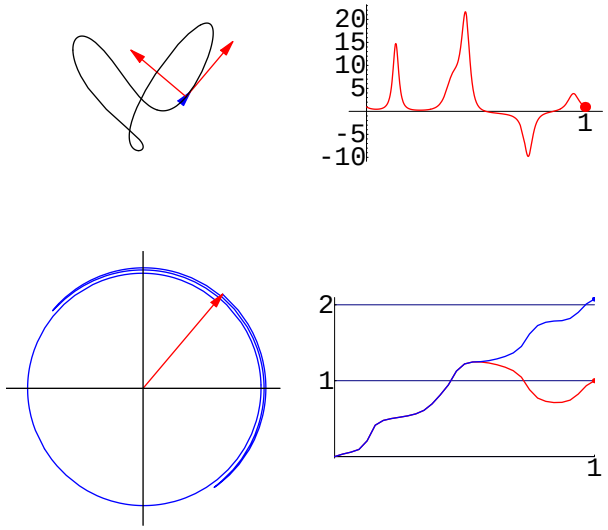
dgSingularCueveInHomotopyAnim is useful to detect the possible non−regular curves among the homotpopy curves.

On the other hand, the total absolute curvature of a closed curve $c(s)$ is defined by $\int_a^b |\kappa(s)|\,ds$. **dgTotalAbsoluteCurvatureAnim** shows how the value of the total absolute curvature is determined.

In[89]:=
```
?dgTotalAbsoluteCurvatureAnim
```

```
dgTotalAbsoluteCurvatureAnim[c,t0,t1,n]
  returns an animation of the curve c(t),t0<=t<=
  t1, with n scenes, together with the change of
  the  curvature, Gauss map, and the integration of
  the curvature (red curve) and the integration of
  the absolute value of the curvature (blue curve).
```

In[90]:=
```
cur[t_]={
 0.46*Cos[2*Pi*t]+0.31*Cos[4*Pi*t]− 0.14*Cos[6*Pi*t]+
 0.56*Sin[2*Pi*t]+0.13*Sin[4*Pi*t]+0.042*Sin[6*Pi*t],
 0.45*Cos[2*Pi*t]−0.44*Cos[4*Pi*t]− 0.15*Cos[6*Pi*t]−
 0.12*Sin[2*Pi*t]+0.41*Sin[4*Pi*t]+ 0.14*Sin[6*Pi*t]};
dgTotalAbsoluteCurvatureAnim[cur[t],{t,0,1},32];
```



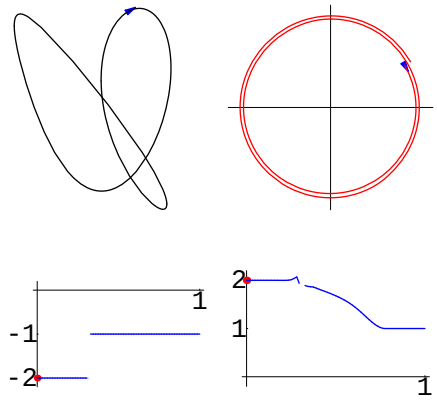
Unlike the total curvature, the value of the total absolute curvature is not discrete. According to a well−known theorem, the total absolute curvature is greater than or equal to 2π, and it is equal to 2π if and only if the curve is an ovaloid.
dgTotalAbsoluteCurvatureAnim shows the above using a homotopy.

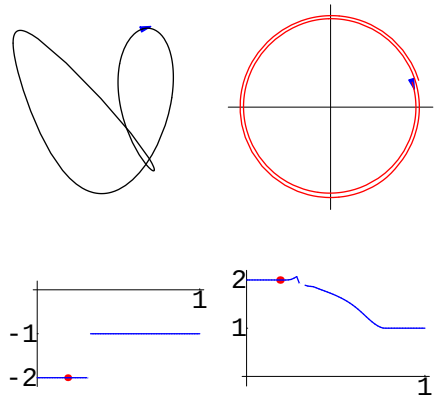
In[92]:=
```
?dgTotalAbsoluteCurvatureUnderHomotopyAnim
```

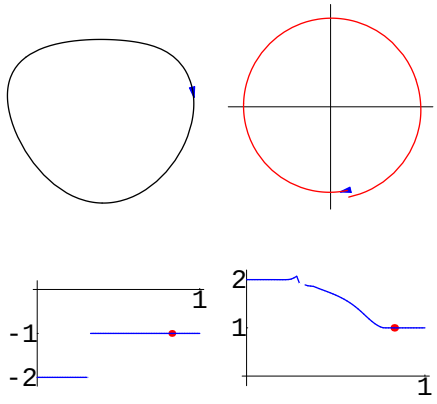
```
dgTotalAbsoluteCurvatureUnderHomotopyAnim[cur11,cur22,t0,t1,
  n] returns the animation of the homotopy between the
  curves cur11(t) and cur22(t) , t0<=t<=t1, together
  with the change of Gauss map, total curvature, and
  total absolute curvature, with n scenes. All homotopy
  curves are not necessarili regular even if  cur11(
  t) and cur22(t) are regular, so the values of the
  total curvature and total absolute curvature are not
  necessarily continuous in the homotopy parameter.
```

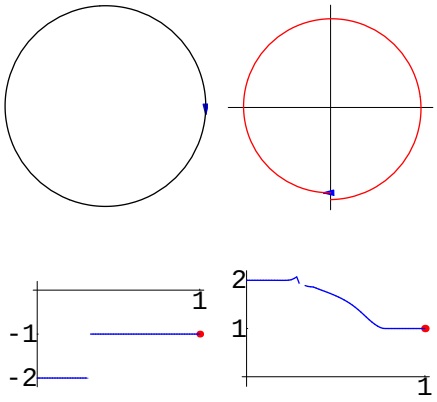
Example 1 (case: there is a singular cueve in the homotopy.)

In[93]:=
```
c11[t_]=N[{
 0.51*Cos[2*Pi*t]−0.31*Cos[4*Pi*t]+
          0.026*Cos[6*Pi*t]−
 0.17*Sin[2*Pi*t]+0.34*Sin[4*Pi*t]+
           0.12*Sin[6*Pi*t],
0.001*Cos[2*Pi*t]+0.93*Cos[4*Pi*t]+
          0.011*Cos[6*Pi*t]+
 0.13*Sin[2*Pi*t]+0.12*Sin[4*Pi*t]+
          0.042*Sin[6*Pi*t]}];
c22[t_]={Cos[−2*Pi*t],Sin[−2*Pi*t]};
Timing[dgTotalAbsoluteCurvatureUnderHomotopyAnim[c11,c22,0,1,64];]
```


1
-1
-2
2
1
1


1
-1
-2
2
1
1


1
-1
-2
2
1
1



Out[95]= {31.5 Second, Null}

■ 3. Space curves

First, open the **'SpaceCurves'** package.

In[1]:= **<<DGeometrica'SpaceCurves'**

□ 3.1 Space curves

A space curve is a mapping from an interval $I \subset \mathbb{R}$ into the three dimensional Euclidian space $\mathbb{R}^3$

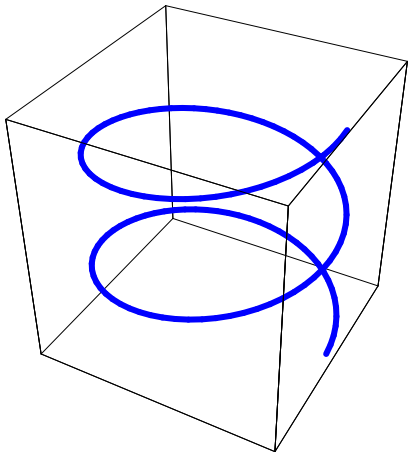
$$c: I \longrightarrow \mathbb{R}^3 \; ; \; c(t)=(x(t), y(t), z(t)) \; t \in I,$$

where $x(t)$, $y(t)$ and $y(t)$ are differentiable functions of class at least C^3 . We assume all space curves are regular, i.e., the derivative $c'(t)$ never vanishes. A space curve is plotted usually by **ParametricPlot3D**, but it has some difficulty to set options.
dgPlotCurve3D plots space curves with given thickness and RGBColor.
dgArrow3D is useful to draw 3–dimnsional arrows.
We can obtain randomly generated space curves and closed curves by similar commands for plane curves.

In[2]:= **?dgPlotCurve3D**

```
dgPlotCurve3D[c[t],{t,t0,t1},n,col,thk,opts]
  plots the curve c(t), a<=t<=b, with PlotPoints->n,
  RGBColor -> col, Thickness-> thk,a and options->opt.
```

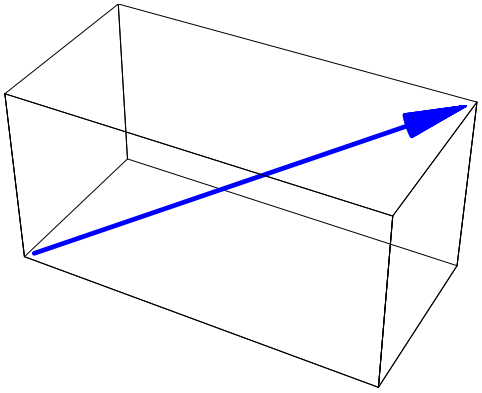
```
In[3]:= hel[t_] = {Cos[t], Sin[t], t / 6};
        dgPlotCurve3D[hel[t], {t, 0, 4 Pi},
         128, {0, 0, 1}, 0.015, Axes -> None];
```



```
In[5]:= ?dgArrow3D
```

```
dgArrow3D[p,q,rgb,m,thk11,thk22,n] returns
  the 3-dimensional arrow from p to q with color
  rgb and thickness of the arrow thk22, where
  the arrow-head consists of n line-segments with
  thickness thk11 and the size of the head is m.
```

```
In[6]:= Show[dgArrow3D[{0, 0, 0}, {2, 1, 1},
          {0, 0, 1}, 0.3, 0.005, 0.01, 42], PlotRange -> All]
```



```
Out[6]= - Graphics3D -
```

□ 3.2 Moving frame, curvature, and torsion

Length of space curves and arclength parameter for space curves are defined in a similar way for plane curves. For a curve $c(s)$ parametrized by arclength s, the moving frame $\{e_1(s), e_2(s), e_3(s)\}$ is defined as follows:

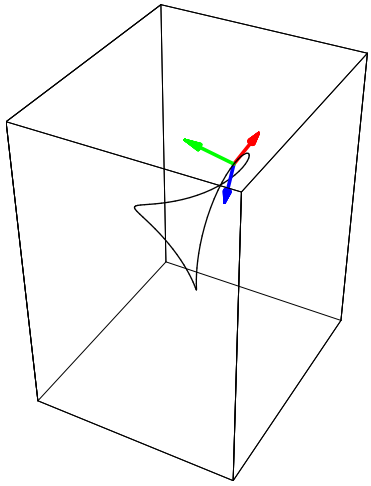
$$e_1(s) = c'(s),\ e_2(s) = \frac{e'_1(s)}{|e'_1(s)|},\ e_3(s) = e_1(s) \times e_2(s),$$

where $\times$ denotes the exterior product in $\mathbb{R}^3$. $e_1(s)$, $e_2(s)$, $e_3(s)$ are called the unit tangent, normal, and binormal vector field of $c(s)$, respectively.

In[7]:= **?dgMovingFrame3DAnim**

```
dgMovingFrame3DAnim[cur,t0,t1,n] returns an animation
  of the moving frame of a curve c[t], t0<=t<=t1,
  with n scenes. The unit tangent vector is shown
  in red color, the unit principal normal vector
  in blue, ant the unit binormal vector in green.
```

In[8]:=
```
c44[t_]=1.5*{0.25−0.38*Cos[2Pi*t]+0.34*Cos[4Pi]+
0.30*Sin[2Pi*t]−0.02*Sin[4Pi*t],
−0.17−0.20*Cos[2Pi*t]+0.52*Cos[4Pi*t]+
0.10*Sin[2Pi*t]+0.27*Sin[4Pi*t],
0.10−0.80*Cos[2Pi*t]−0.20*Cos[4Pi*t]−
0.29*Sin[2Pi*t]−0.54*Sin[4Pi*t]};
dgMovingFrame3DAnim[c44,0,1,64];
```



The curvature $\kappa(s)$ and torsion $\tau(s)$ of a space curve $c(s)$ parametrized by arclength are defined by

$$\kappa(s) = |\, e'_1(s)\,|, \quad \tau(s) = e'_2(s)\cdot e_3(s),$$

where the dot $\cdot$ denotes the inner product of vectors. In terms of an arbitrary parameter t, they are expressed as

$$\kappa(t) = \frac{|\, c'(t)\times c''(t)\,|}{|\, c'(t)\,|^3}, \quad \tau(s) = \frac{\det(\, c'(t)\; c''(t)\; c'''(t)\,)}{|\, c'(t)\times c''(t)\,|^2},$$

where det denotes the determinant. They are calculated by **dgCurvature3D** and **dgTorsion**. Frenet's Formula for space curves is as follows.

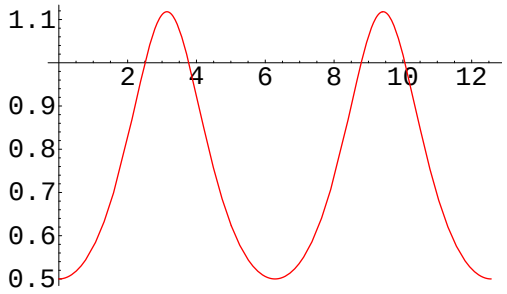
$$\begin{aligned} c'(s) &= e_1(s) \\ e'_1(s) &= \kappa(s)\, e_2(s) \\ e'_2(s) &= -\kappa(s)\, e_1(s) + \tau(s)\, e_3(s) \\ e'_3(s) &= -\tau(s)\, e_2(s) \end{aligned}$$

In[10]:= **viviani[t_]={1+Cos[t],Sin[t],2Sin[t/2]};**

In[11]:= **dgCurvature3D[viviani][t]//Simplify**

Out[11]=
$$\frac{\sqrt{3\cos(t)+13}}{(\cos(t)+3)^{3/2}}$$

In[12]:= **Plot[dgCurvature3D[viviani][t],{t,0,4Pi},PlotStyle->RGBColor[1,0,0]]**

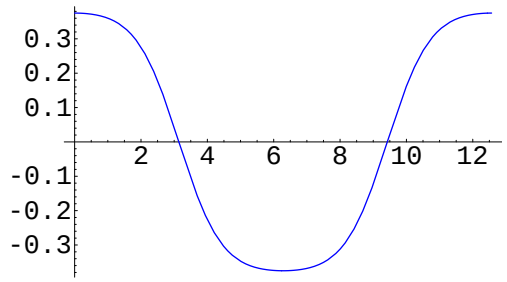


Out[12]= - Graphics -

In[13]:= **dgTorsion[viviani][t]//Simplify**

Out[13]= $\frac{6\cos\left(\frac{t}{2}\right)}{3\cos(t)+13}$

In[14]:= **Plot[dgTorsion[viviani][t],{t,0,4Pi},PlotStyle->RGBColor[0,0,1]]**



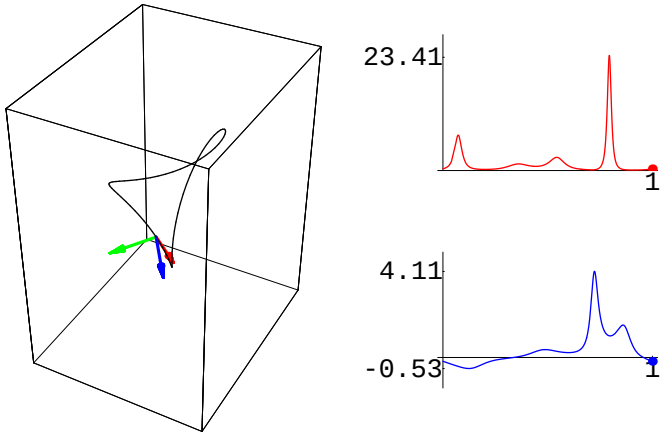
Out[14]= - Graphics -

We can observe the change of the values of curvature and torsion by **dgCurvature-AndTorsionAnim**.

In[15]:= **?dgCurvatureAndTorsionAnim**

```
dgCurvatureAndTorsionAnim[c,t0,t1,n,opts]
  returns an animation of n+1 scenes of a curve c[
  t], t0<=t<=t1, together with the moving frame,
  the curvature function un red color, and the
  torsion function in blue with options opts.
```

In[16]:=
```
c44[t_]=1.5*{0.25−0.38*Cos[2Pi*t]+0.34*Cos[4Pi]+
0.30*Sin[2Pi*t]−0.02*Sin[4Pi*t],
−0.17−0.20*Cos[2Pi*t]+0.52*Cos[4Pi*t]+
0.10*Sin[2Pi*t]+0.27*Sin[4Pi*t],
0.10−0.80*Cos[2Pi*t]−0.20*Cos[4Pi*t]−
0.29*Sin[2Pi*t]−0.54*Sin[4Pi*t]};
dgCurvatureAndTorsionAnim[c44,0,1,32];
```



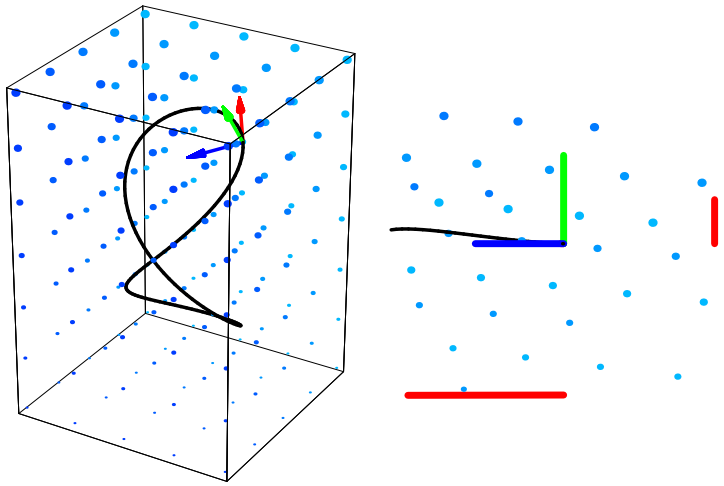
□ 3.3 Geometric meaning of curvature and torsion

dgViewFromMovingFrame3DAnim is the 3−dimensional version of **dgViewFrom Mov− ingFrameAnim** . The animation shows scenes viewed from the coordinate fixed to the moving frame looking at the ambient space into the direction of the unit tangent vector. The change of curvature and torsion is also shown.

In[18]:=
```
?dgViewFromMovingFrame3DAnim
```

```
dgViewFromMovingFrame3DAnim[c,t0,t1,n] returns an animation
  of the graphics array of the graphics of  curve c[t]
  with the movimg frame and the graphics of the view
  from the moving frame of a space curve c(t), t0<=t<=
  t1, with n+1 scenes. The curvature and torsion are
  shown by the vertical and horizontal segments. Lattice
  points are also plotted to clarify the background. opts
  are the options for the graphics of the curve c[t].
```

```
In[19]:= viviani[t_]={1+Cos[t],Sin[t],2Sin[t/2]};
         dgViewFromMovingFrame3DAnim[viviani,0,4Pi,128,ViewPoint->{5,3,3}];
```



□ 3.4 Fundamental Theorem of space curves

Space curves are determined by a given curvature function and a given torsion function as stated in the following.

Fundamental Therem of Space Curves. *Let $\kappa(s)$ and $\tau(s)$ be continuous functions of s. Then, there exist space curves c(s), such that s is the arclength parameter of c, and the curvature function and torsion function of c(s) are equal to the given functions $\kappa(s)$ and $\tau(s)$, respectively. These curves are all congruent under isometries of $\mathbb{R}^3$, therefore such a curve is uniquely determined by the initial condition $c(s_0)=p_0$, $c'(s)=v_0$, $c''(s)=w_0$.*

Such a curve described in the theorem is obtained by regarding the Frenet's Formula for space curves as a system of differential equations and by solving it. We can utilize **NDSolve** of *Mathematica* for this purpose, to obtain **dgCurveByCurvatureAndTorsion**.

```
         ?dgCurveByCurvatureAndTorsion
```

```
In[21]:= kk[s_]=2Sin[s]+2.5;tt[s_]=2Sin[4s]^3;
         c[s_]=dgCurveByCurvatureAndTorsion[kk,tt,{s,0,4Pi},{0,0,0},{1,0,0},{0,0,1}];
```

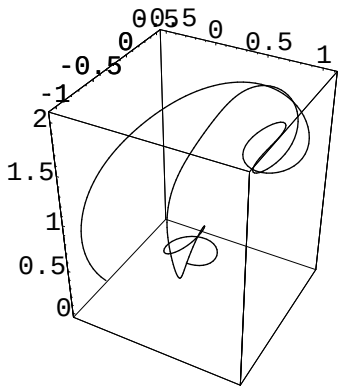
```
In[23]:= c[s]
```

```
Out[23]= {InterpolatingFunction[{ 0.   12.5664 }, <>][s],
           InterpolatingFunction[{ 0.   12.5664 }, <>][s],
           InterpolatingFunction[{ 0.   12.5664 }, <>][s]}
```

```
In[24]:= c[Pi/3]
```

```
Out[24]= {0.0450083, −0.279809, 0.505861}
```

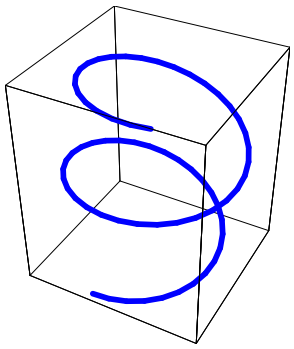
In[25]:= **ParametricPlot3D[c[s],{s,0,4Pi},PlotPoints->256];**



To draw such a curve, **dgPlotByCurvatureAndTorsion** is useful.

In[26]:= **kk[s_]=1;tt[s_]=0.2;**

In[28]:= **dgPlotByCurvatureAndTorsion[kk,tt,{s,0,4Pi},{0,0,0},{1,0,0},{0,1,0},{0,0,1},0.02,64];**

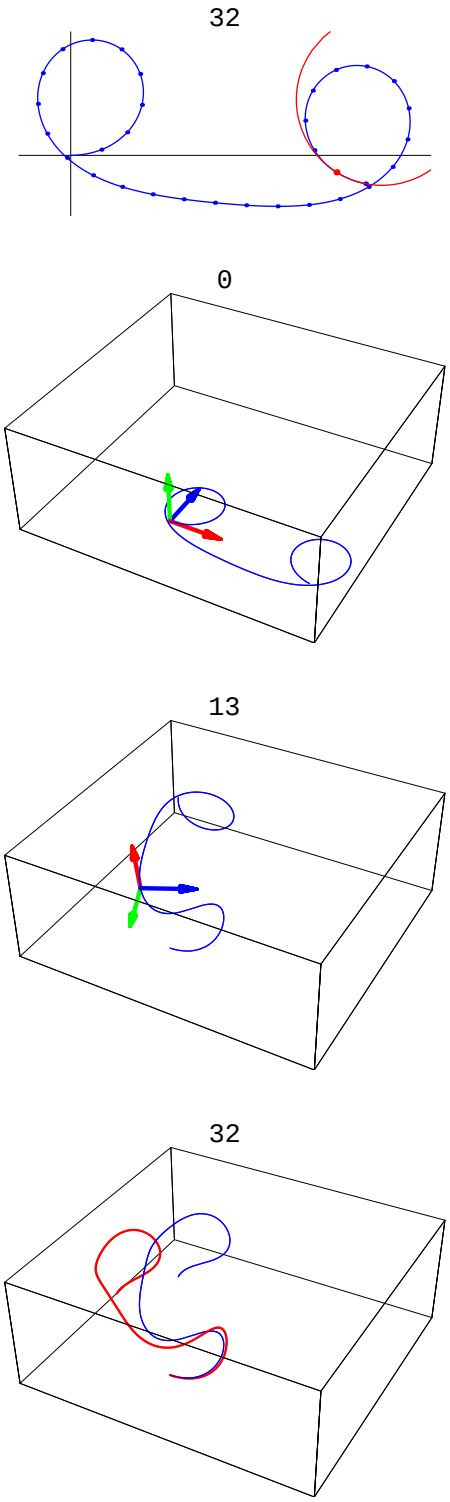


To following **dgJointCircleSegments3DAnim** is the 3−dimensional version of **dg−JointCircleSegmentsAnim**. First, we joint small pieces of circles whose radii are determined by the curvature function and obtain a plane curve. Then, we twist the plane curve at each jointing point according to the torsion function and generates a space curve. The obtained space curve approximates the curve determined by the curvature and torsion, according to the Fundamental Theorem.

In[2]:= **?dgJointCircleSegments3DAnim**

```
dgJointCircleSegments3D[curv,tors,{s,s0,s1},num] returns
  the animation of the approximation of a curve c[s],
  s0<=s<=s1, with the curvature function curv and the
  torsion function tors first by jointing num circle-
  segments whise radii are determined by the curvature
  function, and then by twisting it at each jointing
  point according to the torsion function. The figure
  of c[t] obtained by solving the system of equations
  of the Fundamental Theorem of Space Curves is
  also shown at the end to compare the approximation.
```

In[3]:= **dgJointCircleSegments3DAnim[Sin[s]+1.1,2Sin[2s],{s,0,3Pi},32]**



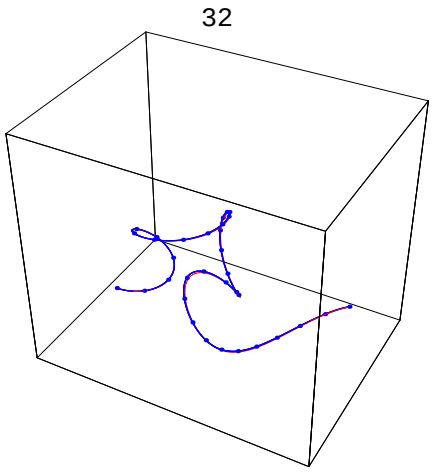
If we increase the number of the segments, then we obtain better approximation. The following **dgJointHelixSegments3DAnim** joints small pieces of helices whose curvature and torsion are constant values determined by the given curvature and torsion

functions. At each jointing points, we joint them so that the moving frames of the two adjoining helices coincide. Comparing the output of this command to that of the previous one, we get better approximation and see that the helices are most basic curves among space curves.

In[4]:= **?dgJointHelixSegmentsAnim**

```
dgJointHelixSegmentsAnim[kappa,tau,lgth,num,
  pltrng] returns an animation of piesewise smooth
  curves obtained by jointing pieces of helices that
  approximates the red curve with assined curvature
  kappa(s), torsion tau(s), and length lgth. The
  number of the segments varies from 2 to numand the
  PlotRange is pltrng. The animation shows a better
  approximation than that of jointing circle segments.
```

In[10]:=
```
kappa22[s_]=Sin[s]+1.5;
tau22[s_]=Sin[s];
plrng22={{-1.6,3},{0,3.5},{-2,2}};
dgJointHelixSegmentsAnim[kappa22,tau22,4*Pi,32,plrng22];
```



□ 3.5 Global properties of space curves

One of the theorems about the global properties of space curves is the following:

Fenchel's Therem. *The total absolute curvature* κ_c *of a closed space curve c satisfies the inequality* $\kappa_c \geq 2\pi$, *and* $\kappa_c = 2\pi$ *if and only if c is an ovaloid contained in a plane.*

We do not go further into this theorem here, but since the proof is based on the Gauss map of space curves, so we introduce here some **dg**−commands related to the Gauss map.

The Gauss map γ of a space curve c: $I \longrightarrow \mathbb{R}^3$ is a mapping to the unit sphere $S^2 \subset \mathbb{R}^3$ defined by

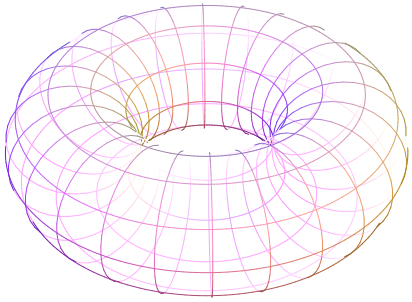
$$\gamma : I \longrightarrow S^2 \subset \mathbb{R}^3 ; \gamma(t)=e_1(t)\ t\in I,$$

where the starting point of the unit vector $\gamma(t)$ is set to the origin O of $\mathbb{R}^3$. Therefore, the image of the Gauss map of a space curve is a curve contained in the unit sphere. If we draw a surface by the usual **ParametricPlot3D**, then curves on the surface cannot be expressed rightly. So we use here **dgWireFrame** to draw a colorful wireframe of a surface with a kind of hidden curves expression, to make a curve on the surface clearer.

In[14]:= **?dgWireFrame**

```
dgWireFrame[f[u,v],{u,u0,u1},{v,v0,v1},{m1,m2},{n1,n2},
  viewpnt,opts] returns a wireframe graphics of  the
  surface f[u,v], u0<=u<=u1, v0<=v<=v1, with (n1+1) u-
  curves and (m1+1)  v-curves. (m1*m2+1) is the plotpoints
  of each u-curve, and (n1*n2+1) is of each v-curve. The
  wireframe is shown with RGBColor similar to that of
  surfaces plotted by ParametricPlot3d. viewpnt shoul be
  the viewpoint from which the wiewframe is observed,
  by default {1.3,-2.4,2}. opts are options for Show
```

In[15]:= **graph11=dgWireFrame[**
{Cos[u]*(Cos[v]+2),Sin[u]*(Cos[v]+2),Sin[v]},
{u,0,2Pi},{v,0,2Pi},{24,8},{10,4},{1.3,−2.4,2.},Boxed−>False]



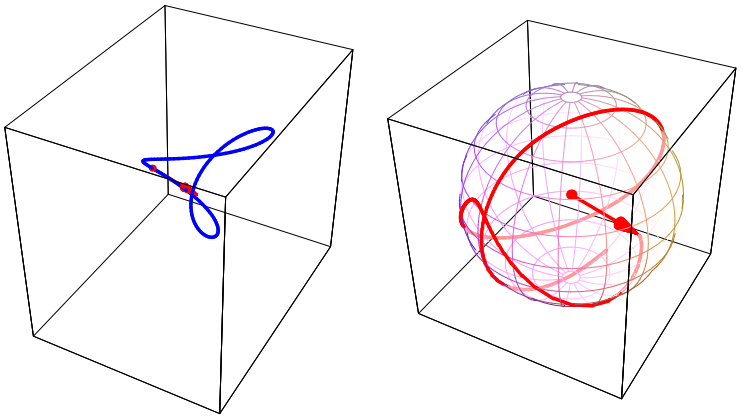
Out[15]= **-** Graphics3D **-**

dgGaussMapOfSpaceCurveAnim creates an animation how the Gauss map of a space curve is generated.

In[16]:= **?dgGaussMapOfSpaceCurveAnim**

```
dgGaussMapOfSpaceCurveAnim[cur,{t,t0,t1}] returns an
  animation of the Gauss map of a space curve cur, t0<=t<=
  t1. The unit sphere is plotted by a colorful wireframe.
```

In[17]:=
```
c44[t_]={0.25−0.38*Cos[6.28*t]+0.34*Cos[12.56*t]+0.30*Sin[6.28*t]−0.02*Sin[12.56*t],
−0.17−0.20*Cos[6.28*t]+0.52*Cos[12.56*t]+0.10*Sin[6.28*t]+0.27*Sin[12.56*t],0.10−0.8
0*Cos[6.28*t]−0.20*Cos[12.56*t]−0.29*Sin[6.28*t]−0.54*Sin[12.56*t]};
dgGaussMapOfSpaceCurveAnim[c44[t],{t,0,1}];
```



■ 4. Conclusion

The author considers that the following things should be done after this article.

(1) The second half of **DGeometrica** about surfaces will be organized soon.
(2) The new version 6 of *Mathematica* is said to include new features of graphics functions, therefore the package should be adjusted accordingly.
(3) Also, some new topics that did not appear in [10] will be included.
(4) Another important thing is to construct an interface for the package, namely, a material that explains the meaning of commands in the package, some thing like the present notebook, but more comprehensive and like the help browser of *Mathematica*.
(5) Visualization of proofs of theorems in differential geometry as presented in [3] should also be included in the package.

As a conclusion, classical differential geometry of curves and surfaces is a nice theme that has an affinity between *Mathematica*; **NDSolve**, **NIntegrate**, and many graphics functions of *Mathematica* are powerful tools to visualize mathematical concepts in differential geometry: it is what the author would like to emphasize in this package.

■ References

IMS talks

[1] Alfred Gray, Tutorial: *Differential Geometry of Surfaces via Mathematica*, IMS 1995.

[2] Yoshihiko Tazawa, *Experiments in the Theory of Surfaces*, IMS 1999.

[3] Yoshihiko Tazawa, *Gauss−Bonnet Theorem by Mathematica*, IMS 2001.

On−line packages

[4] Alfred Gray's Home page: http://math.cl.uh.edu/~gray/
[5] 3D−XplorMath by Richard Palais et al: http://vmm.math.uci.edu/3D−XplorMath/

Books

[6] Michael Spivak, *A comprehensive Introduction to Differential Geometry*, I−V, Publish or Perish, 1975.

[7] M. P. do Carmo, *Differential Geometry of Curves and Surfaces*, I−V, Prentice−Hall, 1976.

[8] Alfred Gray, *Modern differential geometry of curves and surfaces with Mathematica*, Second Edition, CRC Press 1998.

[9] John Oprea, *Differential geometry and its applications*, Second Edition, Prentice Hall, 2004.

[10] Yoshihiko Tazawa, *Introduction to the theory of curves and surfaces with* ***Mathematica*** (in Japanese language) , Pearson Education Japan, 1999.
An English outline: http://math.kn.dendai.ac.jp/cur_face.pdf