Math 4530

4.2.8 - HW solutions to set 7

Computations related to the shape operator

Here is a list of procedures to calculate the matrix of the shape operator, the principle curvatures, the mean curvature and the Gauss curvature, using a given patch X. The procedures are illustrated with computations and pictures for the helicoid and the torus.

```
> restart:
  with(linalg):
  with(plots):
Warning, the protected names norm and trace have been redefined and unprotected

Warning, the name changecoords has been redefined

> assume(u,real); #this gets rid of that annoying "csgn" fcn
  assume(v,real);
> #dot product
  dp:=proc(X,Y)
  X[1]*Y[1]+X[2]*Y[2]+X[3]*Y[3];
  end:
> #2-norm, i.e. magnitude.
  nrm:=proc(X)
  sqrt(dp(X,X));
  end:
> #cross product:
  xp:=proc(X,Y)
  local a,b,c;
  a:=X[2]*Y[3]-X[3]*Y[2];
  b:=X[3]*Y[1]-X[1]*Y[3];
  c:=X[1]*Y[2]-X[2]*Y[1];
  [a,b,c];
  end:
> #Derivative matrix for mapping X:
  DXq:=proc(X)
  local Xu,Xv;
  Xu:=matrix(3,1,[diff(X[1],u),diff(X[2],u),diff(X[3],u)]);
  Xv:=matrix(3,1,[diff(X[1],v),diff(X[2],v),diff(X[3],v)]);
  simplify(augment(Xu,Xv),radical,symbolic,trig);
  end:
> #Matrix of first fundamental form:
  gij:=proc(X)
  local g11,g12,g22,Y;
  Y:=evalm(DXq(X));
  simplify(evalm(transpose(Y)&*Y),
     radical,symbolic,trig);
  end:
> #unit normal:
  U:=proc(X)
  local Y,Z,s;
```

```
Y:=DXq(X);
Z:=xp(col(Y,1),col(Y,2));
s:=nrm(Z);
simplify(evalm((1/s)*Z),radical,symbolic,trig);
end:
> #matrix of second fundamental form:
hij:=proc(X)
local Y,Xu,Xv,Xuu,Xuv,Xvv,U1,h11,h12,h22;
Y:=DXq(X);
U1:=U(X);
Xu:=col(Y,1);
Xv:=col(Y,2);
Xuu:=[diff(Xu[1],u),diff(Xu[2],u),diff(Xu[3],u)];
Xuv:=[diff(Xu[1],v),diff(Xu[2],v),diff(Xu[3],v)];
Xvv:=[diff(Xv[1],v),diff(Xv[2],v),diff(Xv[3],v)];
h11:=dp(Xuu,U1);
h12:=dp(Xuv,U1);
h22:=dp(Xvv,U1);
simplify(matrix(2,2,[h11,h12,h12,h22]),
  radical,symbolic,trig);
end:

> #matrix of shape operator wrt basis {Xu,Xv}:
aij:=proc(X)
local Y,H,G;
H:=hij(X);
G:=gij(X);
simplify(evalm(inverse(G)&*H),
   radical,symbolic,trig);
end:
> #Gauss curvature
GK:=proc(X)
local A;
A:=aij(X);
simplify(det(A),radical,symbolic,trig);
end:
> #Mean curvature
MK:=proc(X)
local A;
A:=aij(X);
simplify(1/2*trace(A),radical,symbolic,trig);
end:
> #Principle curvatures and directions:
PK:=proc(X)
local Y;
Y:=aij(X);
eigenvects(Y);
end:
> test:=[u,v,u^2-v^2];
```

$$test := [u\sim, v\sim, u\sim^2 - v\sim^2]$$

```
> DXq(test);
```

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2\,u\sim & -2\,v\sim \end{bmatrix}$$

```
> gij(test);
  hij(test);
  subs({u=0,v=0},aij(test));
  subs({u=0,v=0},GK(test));
  subs({u=0,v=0},MK(test));
  subs({u=0,v=0},aij(test));
```

$$\begin{bmatrix} 1+4\,u\sim^2 & -4\,u\sim\,v\sim \\ -4\,u\sim\,v\sim & 1+4\,v\sim^2 \end{bmatrix}$$

$$\begin{bmatrix} \frac{2}{\sqrt{4\,u\sim^2+4\,v\sim^2+1}} & 0 \\ 0 & -\frac{2}{\sqrt{4\,u\sim^2+4\,v\sim^2+1}} \end{bmatrix}$$

$$\begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

$$-4$$

$$0$$

$$\begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

4.2.8 begins here:

(a) Helcat:

```
> assume(t,real);
> helcat:=[cos(t)*sinh(v)*sin(u)+sin(t)*cosh(v)*cos(u),
  -cos(t)*sinh(v)*cos(u)+sin(t)*cosh(v)*sin(u),
  u*cos(t)+v*sin(t)];
```

$$helcat := [\cos(t\sim)\sinh(v\sim)\sin(u\sim)+\sin(t\sim)\cosh(v\sim)\cos(u\sim),\\ -\cos(t\sim)\sinh(v\sim)\cos(u\sim)+\sin(t\sim)\cosh(v\sim)\sin(u\sim),\ u\sim\cos(t\sim)+v\sim\sin(t\sim)]$$

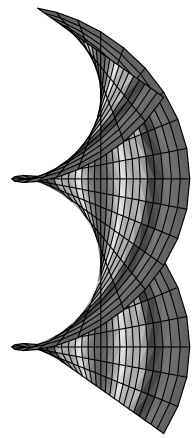
```
> GK(helcat);
  MK(helcat);
```

$$-\frac{1}{\cosh(v\sim)^4}$$

$$0$$

```
> animate3d(helcat,u=0..2*Pi,v=-1..1,t=0..Pi/2,color=GK(helcat),
```

```
scaling=constrained);#this lets you see the helicoid lowering
#itself into a catenoid.  By coloring with GK it is easier
#to see corresponding points.
```

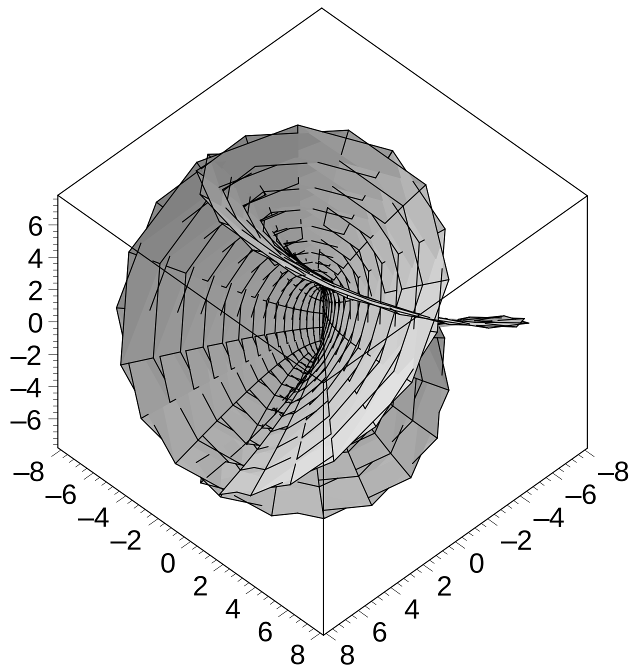


b) Henneberg. (Maple could not compute MK or GK)

```
> henne:=[2*sinh(u)*cos(v)-2/3*sinh(3*u)*cos(3*v),
  2*sinh(u)*sin(v)+2/3*sinh(3*u)*sin(3*v),
  2*cosh(2*u)*cos(2*v)];
```

$$henne := \left[2\,\sinh(u\sim)\cos(v\sim) - \frac{2}{3}\sinh(3\,u\sim)\cos(3\,v\sim),\ 2\,\sinh(u\sim)\sin(v\sim) + \frac{2}{3}\sinh(3\,u\sim)\sin(3\,v\sim),\ 2\cosh(2\,u\sim)\cos(2\,v\sim)\right]$$

```
> plot3d(henne,u=-1..1,v=0..2*Pi,grid=[30,30],
  scaling=constrained,axes=boxed);
```

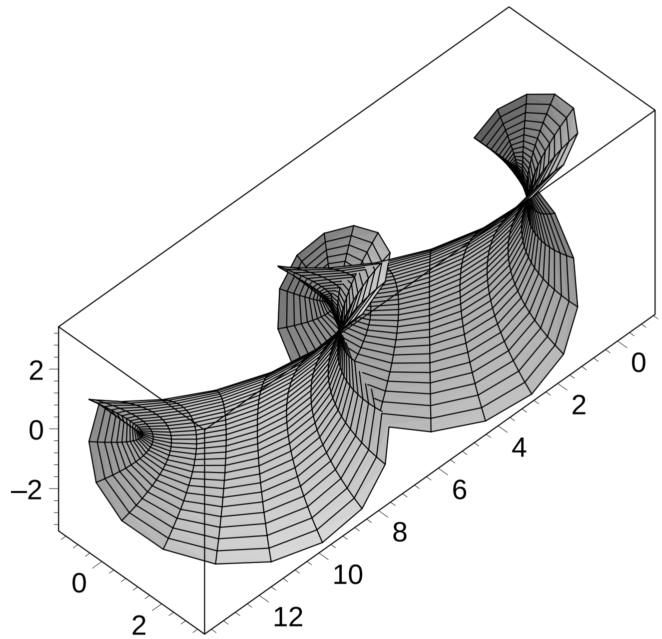


c) Catalan: Maple couldn't compute GK or MK for this!

```
> catalan:=[u-sin(u)*cosh(v),1-cos(u)*cosh(v),
  4*sin(u/2)*sinh(v/2)];
```

$$catalan := \left[u\sim - \sin(u\sim)\cosh(v\sim),\ 1 - \cos(u\sim)\cosh(v\sim),\ 4\sin\left(\frac{u\sim}{2}\right)\sinh\left(\frac{v\sim}{2}\right) \right]$$

```
> plot3d(catalan,u=0..4*Pi,v=-1.5..1.5,grid=[30,30],
  scaling=constrained,axes=boxed);
```



d) Enneper:

```
> enne:=[u-u^3/3+u*v^2,v-v^3/3+v*u^2,u^2-v^2];
```

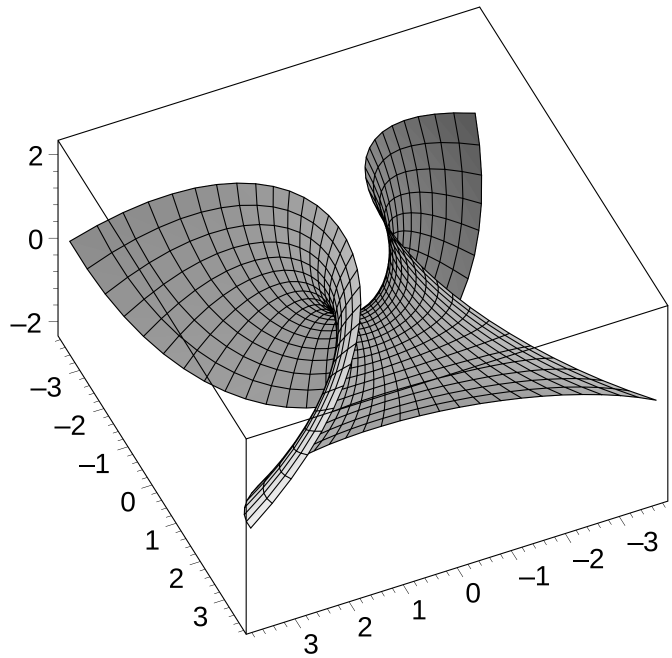
$$enne := \left[u\sim - \frac{1}{3} u\sim^3 + u\sim v\sim^2,\ v\sim - \frac{1}{3} v\sim^3 + v\sim u\sim^2,\ u\sim^2 - v\sim^2 \right]$$

```
> GK(enne);
  MK(enne);
```

$$-\frac{4}{(1 + 2\,u\sim^2 + u\sim^4 + 2\,v\sim^2 + 2\,v\sim^2\,u\sim^2 + v\sim^4)^2}$$

$$0$$

```
> plot3d(enne,u=-1.5..1.5,v=-1.5..1.5,grid=[30,30],
  scaling=constrained,axes=boxed);
```



e) Sherk's Fifth surface:

```
> sherk:=[arcsinh(u),arcsinh(v),arcsin(u*v)];
```

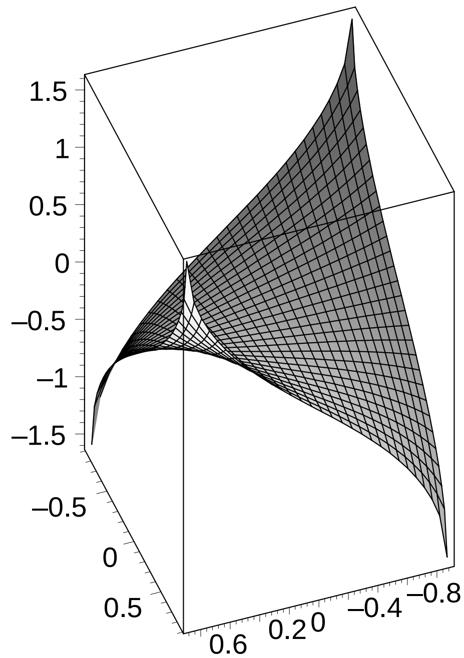
$$sherk := [\operatorname{arcsinh}(u\sim), \operatorname{arcsinh}(v\sim), \arcsin(v\sim\, u\sim)]$$

```
> GK(sherk);
  MK(sherk);
```

$$\frac{1}{(-1-u\sim^2)\,(v\sim^2+1)}$$

$$0$$

```
> plot3d(sherk,u=-1..1,v=-1..1,grid=[30,30],
  scaling=constrained,axes=boxed);
```



f) Planar lines of curvature:

```
> planarlines:=[1/sqrt(1-t^2)*(t*u+sin(u)*cosh(v)),
  1/sqrt(1-t^2)*(v+t*cos(u)*sinh(v))
  ,cos(u)*cosh(v)];
```

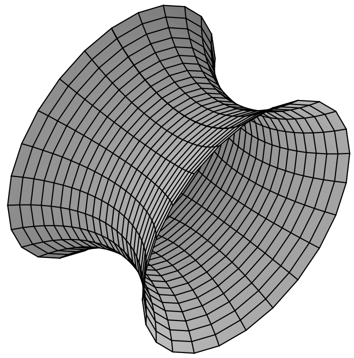
$$planarlines := \left[\frac{t\sim u\sim + \sin(u\sim)\cosh(v\sim)}{\sqrt{1-t\sim^2}}, \frac{v\sim + t\sim \cos(u\sim)\sinh(v\sim)}{\sqrt{1-t\sim^2}}, \cos(u\sim)\cosh(v\sim)\right]$$

```
> GK(planarlines);
  MK(planarlines);
```

$$-(-1+t\sim^2)^2 \operatorname{signum}(-1+t\sim^2)^2 \Big/ (\cosh(v\sim)^4 + 4\,t\sim\cos(u\sim)\cosh(v\sim)^3 + 4\cos(u\sim)^3\,t\sim^3\cosh(v\sim) + 6\cos(u\sim)^2\cosh(v\sim)^2\,t\sim^2 + \cos(u\sim)^4\,t\sim^4)$$

$$0$$

```
> animate3d(planarlines,u=0..2*Pi,
  v=-1..1,t=0..(.5),
  scaling=constrained);  #at t=0 it's a catenoid!
```



```
> animate3d(planarlines,u=0..2*Pi,
  v=-1..1,t=0..(.5),
  scaling=constrained); #which starts
  #getting pulled as t increases!
```