

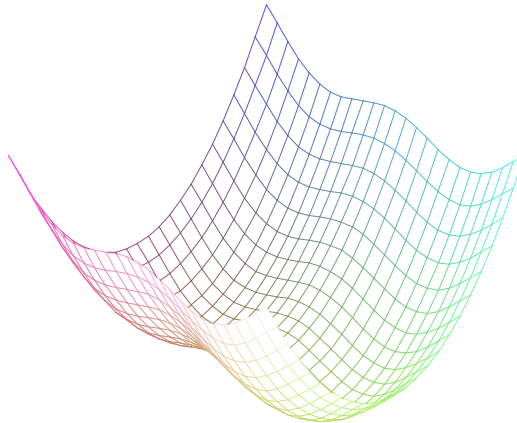
# Surfaces

## ☐ Tracé de surfaces

### ☐ Différents modes de tracés simples

☐ La fonction `plot3d` permet de tracer soit des surfaces cartésiennes  $z=f(x,y)$ , soit des surfaces paramétriques  $x=f(u,v)$ ,  $y=g(u,v)$ ,  $z=h(u,v)$ , de diverses manières. Soit sous forme de réseau de lignes de coordonnées

> `plot3d(x^2-y*sin(y),x=-4..4,y=-4..4);`



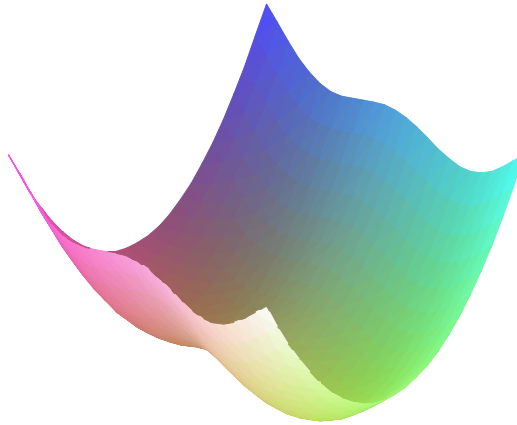
☐ soit sous forme de quadrilatères

> `plot3d(x^2-y*sin(y),x=-4..4,y=-4..4,style=PATCH);`

Insufficient memory  
to display plot

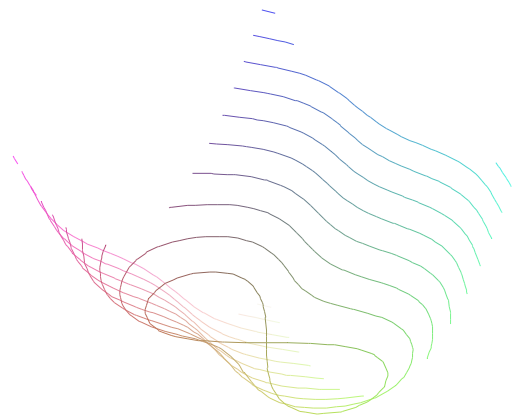
☐ soit sous forme pleine

> `plot3d(x^2-y*sin(y),x=-4..4,y=-4..4,style=PATCHNOGRID);`



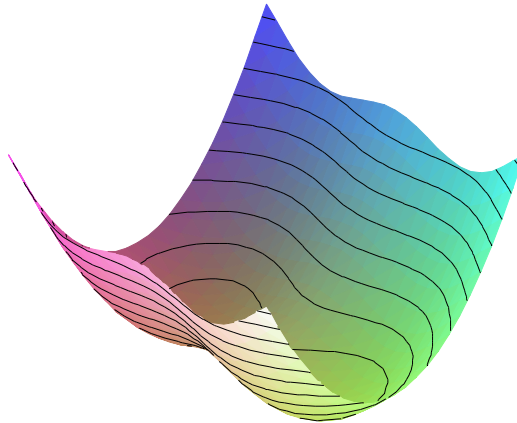
[ soit encore sous forme de lignes de niveau

```
[ > plot3d(x^2-y*sin(y),x=-4..4,y=-4..4,style=CONTOUR);
```

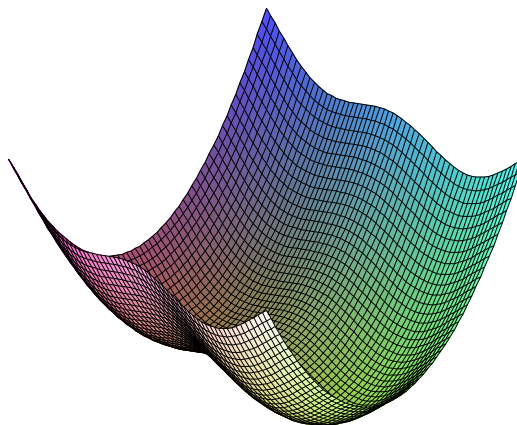


[ soit enfin sous forme pleine avec lignes de niveau

```
[ > plot3d(x^2-y*sin(y),x=-4..4,y=-4..4,style=PATCHCONTOUR);
```



[ enfin l'option grid permet de jouer sur la précision de l'échantillonnage (par défaut 25x25)  
 [ > `plot3d(x^2-y*sin(y),x=-4..4,y=-4..4,grid=[60,60],style=PATCH);`



[ >

### Autres types de tracés

[ La plupart des autres fonctions de tracés se trouvent dans la bibliothèque plots, qu'il convient donc de charger avant toute utilisation.

[ > `with(plots):`

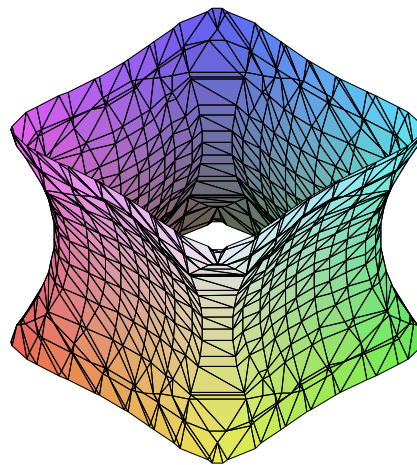
[ La fonction `contourplot3d` permet de tracer les lignes de niveau d'une surface

[ > `contourplot3d(x^2-y*sin(y),x=-4..4,y=-4..4,filled=true);`



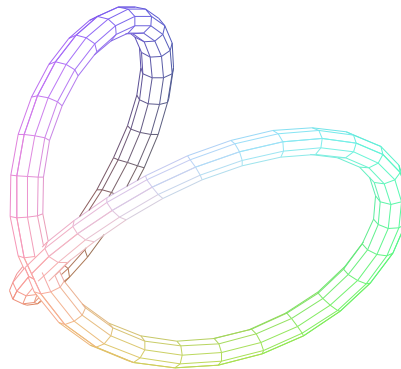
La fonction `implicitplot3d` se charge de tracer les surfaces cartésiennes  $f(x,y,z)=0$ .

```
> implicitplot3d(x^4-x^2*y^2+y^4-z^2=3,x=-2..2,y=-2..2,z=-2..2,
                style=PATCH);
```



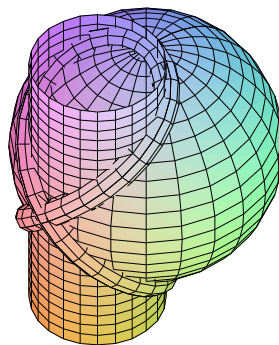
Le rôle de la fonction `tubeplot` est de construire une surface tubulaire autour d'une courbe donnée, ce qui permet (entre autres) une bonne visualisation des courbes en trois dimensions

```
> tubeplot([cos(2*t),sin(t),sin(2*t),t=0..2*Pi,radius=0.1]);
```



Enfin la fonction `display3d` permet de réunir dans un même tracé différentes surfaces, dont les tracés ont été sauvegardés dans des variables.

```
> sphere:=plot3d([cos(u)*cos(v),sin(u)*cos(v),sin(v)],
                 u=-Pi..Pi,v=-Pi/2..Pi/2):
cylindre:=plot3d([(cos(u)+1)/2,sin(u)/2,v],u=-Pi..Pi,v=-1.
2..1.2):
fenetre:=tubeplot([cos(u)^2,sin(u)*cos(u),sin(u),u=-Pi..Pi
],radius=0.1):
display3d({sphere,cylindre,fenetre},style=PATCH,scaling=CO
NSTRAINED);
```



## Plan tangent

### Plan tangent à une surface paramétrique

Le plan tangent à une surface paramétrée  $x=f(u,v)$ ,  $y=g(u,v)$ ,  $z=h(u,v)$  en un point  $(u_0,v_0)$  s'obtient facilement, soit sous forme paramétrée

```
> plan_tangent_par:=proc(surface,par_surface,point,par_plan)
    local Fu,Fv,F;
    F:=subs(par_surface[1]=point[1],
            par_surface[2]=point[2],surface);
    Fu:=subs(par_surface[1]=point[1],
            par_surface[2]=point[2],
            diff(surface,par_surface[1]));
    Fv:=subs(par_surface[1]=point[1],
            par_surface[2]=point[2],
            diff(surface,par_surface[2]));
    [F[1]+par_plan[1]*Fu[1]+par_plan[2]*Fv[1],
     F[2]+par_plan[1]*Fu[2]+par_plan[2]*Fv[2],
     F[3]+par_plan[1]*Fu[3]+par_plan[2]*Fv[3]]
end;
```

```
> plan_tangent_par([cos(u)*v,sin(u)*v,2*u],[u,v],[u0,v0],[lambda,mu]);
```

```
[cos(u0) v0 - sin(u0) v0 + μ cos(u0), sin(u0) v0 + cos(u0) v0 + μ sin(u0),
 2 u0 + 2 ]
```

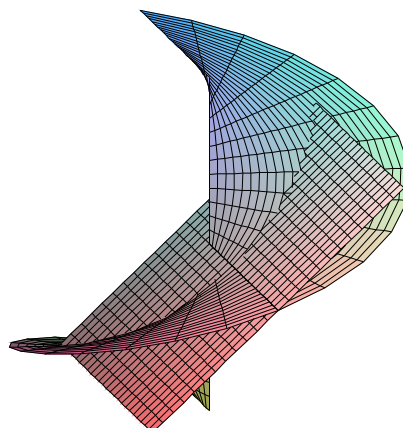
Ceci permet un tracé simple de la surface et de son plan tangent en un point:

```
> surface:=[cos(u)*v,sin(u)*v,u/2];
plan:=simplify(plan_tangent_par(surface,[u,v],[0,1],[s,t])
);
```

$$surface := \cos(u) v, \sin(u) v, \frac{1}{2} u$$

$$plan := 1 + t, s, \frac{1}{2} s$$

```
> g1:=plot3d(surface,u=-Pi..Pi,v=0..3):
g2:=plot3d(plan,s=-2..2,t=-2..2):
display3d({g1,g2},style=PATCH);
```



Quant à l'équation cartésienne, elle est facile à obtenir à l'aide d'un déterminant

```

> plan_tangent_car:=proc(surface,par_surface,point,coord)
    local Fu,Fv,F;
    F:=subs(par_surface[1]=point[1],
            par_surface[2]=point[2],surface);
    Fu:=subs(par_surface[1]=point[1],
            par_surface[2]=point[2],
            diff(surface,par_surface[1]));
    Fv:=subs(par_surface[1]=point[1],
            par_surface[2]=point[2],
            diff(surface,par_surface[2]));
    linalg[det](linalg[matrix](
        [[coord[1]-F[1],coord[2]-F[2],coord[3]-F[3]],
         Fu,Fv]))
    end:
> plan_tangent_car([cos(u)*v,sin(u)*v,2*u],[u,v],[u0,v0],[x,
y,z]);

$$-2 \sin(u_0) x - \sin(u_0)^2 v_0 z + 2 \sin(u_0)^2 v_0 u_0 + 2 \cos(u_0) y - \cos(u_0)^2 v_0 z$$


$$+ 2 \cos(u_0)^2 v_0 u_0$$

> simplify("");

$$-2 \sin(u_0) x - v_0 z + 2 v_0 u_0 + 2 \cos(u_0) y$$

Le recherche de la position de la surface par rapport à son plan tangent se fait simplement
en remplaçant x,y et z par les valeurs en fonction de u,v, soit
> position:=(surface,par_surface,point)->plan_tangent_car(su
rface,par_surface,point,surface):
> simplify(position([cos(u)*v,sin(u)*v,2*u],[u,v],[u0,v0]));

$$-2 \sin(u_0) \cos(u) v - 2 v_0 u + 2 v_0 u_0 + 2 \cos(u_0) \sin(u) v$$

> combine("");

$$2 v \sin(-u_0 + u) - 2 v_0 u + 2 v_0 u_0$$

Ceci permet de rechercher la courbe intersection de la nappe et de son plan tangent (en
dehors de la solution évidente u=u0 qui donne la génératrice de la nappe réglée passant par
le point)
> solve(",v");

$$\frac{v_0 (-u_0 + u)}{\sin(-u_0 + u)}$$

> subs(v=",surface");

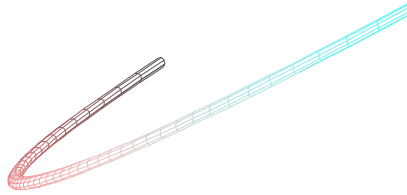
$$\frac{\cos(u) v_0 (-u_0 + u)}{\sin(-u_0 + u)}, \frac{\sin(u) v_0 (-u_0 + u)}{\sin(-u_0 + u)}, \frac{1}{2} u$$

Avec un tracé dans un cas particulier
> courbe:=subs(u0=0,v0=1,"");

$$courbe := \frac{\cos(u) u}{\sin(u)}, u, \frac{1}{2} u$$

> tubeplot(courbe,u=-2..2,radius=0.05);

```



Le signe de l'expression  $rt-s^2$  (avec les notations de Monge) permet facilement la classification d'un point en point elliptique, hyperbolique ou parabolique

```
> type_de_point:=proc(surface,par_surface,point)
    local equation,r,s,t;

    equation:=position(surface,par_surface,point);
    r:=subs(par_surface[1]=point[1],
            par_surface[2]=point[2],
            diff(equation,par_surface[1]$2));
    t:=subs(par_surface[1]=point[1],
            par_surface[2]=point[2],
            diff(equation,par_surface[2]$2));
    s:=subs(par_surface[1]=point[1],
            par_surface[2]=point[2],

    diff(equation,par_surface[1],par_surface[2]));
    r*t-s^2

end;
```

Sur notre exemple précédent, tout point est hyperbolique

```
> simplify(type_de_point([cos(u)*v,sin(u)*v,2*u],[u,v],[u0,v0]));
```

-4

alors que la surface suivante possède à la fois des points des divers types

```
> simplify(type_de_point([u^2+v^2,u^2-v^2,u+v],[u,v],[u0,v0]));
```

$64 v^0 u^0$

[ >



## Types particuliers de surfaces

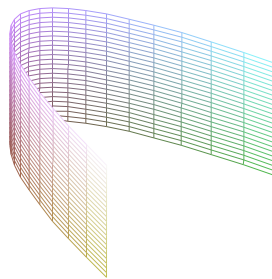
### Cylindres

Une paramétrisation du cylindre ayant pour directrice un arc paramétré et de direction un vecteur est facile à obtenir

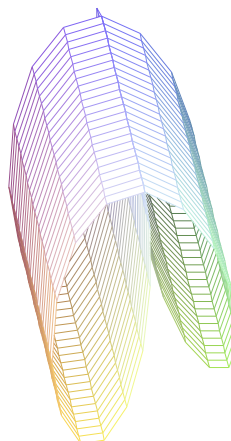
```
> cylindre_par:=proc(directrice,omega,v) local i;  
    [seq(directrice[i]+v*omega[i],i=1..3)]  
end;
```

Avec deux exemples

```
> C1:=cylindre_par([t,t^2,0],[1,1,1],v):  
    plot3d(C1,t=-2..2,v=-1..1);
```



```
> C2:=cylindre_par([sin(t),sin(t)*cos(t),cos(t)^2],[1,1,0],v  
):  
    plot3d(C2,t=-Pi..Pi,v=-1..1);
```



La recherche de l'équation cartésienne d'un cylindre connaissant une directrice d'équations  $f(x,y,z)=0$ ,  $g(x,y,z)=0$  peut se faire en éliminant  $t$  entre les deux équations  $f(x+t a,y+t b,z+t$

c)=0, g(x+t a,y+t b,z+t c)=0, ce qui peut se faire (à condition que toutes les équations soient polynomiales) au moyen d'un résultant (voir l'aide de Maple)

```
> cylindre_car:=proc(courbe,coord,omega) local t,equas;
      equas:=subs(coord[1]=coord[1]+t*omega[1],
                  coord[2]=coord[2]+t*omega[2],
                  coord[3]=coord[3]+t*omega[3],courbe);
      resultant(equas[1],equas[2],t);
end;
```

Voici un exemple avec la parabole d'équations  $x^2-y=0$ ,  $z=0$  et le vecteur (1,1,1).

```
> cylindre_car([x^2-y,z],[x,y,z],[1,1,1]);
```

$$x^2 - y - 2zx + z + z^2$$

et un exemple plus compliqué avec un cylindre construit sur la fenêtre de Viviani (!!)

```
> cylindre_car([x^2+y^2+z^2-1,x^2+y^2-x],[x,y,z],[1,0,1]);
```

$$2x - 1 + 2z^3 + 6zx^2 - 2z + 2x^2y^2 - 4zx^2y^2 + 4y^2 + x^4 - 2x^3 + y^4 + z^4 + 2y^2z^2 - 4z^3x + 6zy^2 - 6y^2x - 6z^2x - 4zx^3 + 6x^2z^2$$

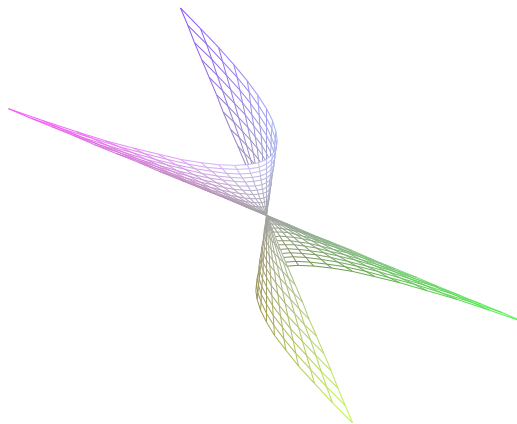
## Cônes

Une paramétrisation du cône ayant pour directrice un arc paramétré et pour sommet un point S peut s'obtenir de la façon suivante

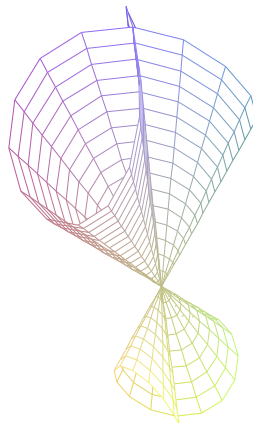
```
> cone_par:=proc(directrice,S,v) local i;
      [seq(v*directrice[i]+(1-v)*S[i],i=1..3)]
end;
```

Avec deux exemples

```
> C1:=cone_par([t,t^2,0],[1,1,1],v):
plot3d(C1,t=-2..2,v=-2..2);
```



```
> C2:=cone_par([sin(t),sin(t)*cos(t),cos(t)^2],[1,1,0],v):
plot3d(C2,t=-Pi..Pi,v=-1..2);
```



La recherche de l'équation cartésienne d'un cylindre connaissant une directrice d'équations  $f(x,y,z)=0$ ,  $g(x,y,z)=0$  peut se faire en éliminant  $t$  entre les deux équations  $f(t x+(1-t) a, t y+(1-t) b, t z+(1-t) c)=0$ ,  $g(t x+(1-t) a, t y+(1-t) b, t z+(1-t) c)=0$ , ce qui peut se faire (à condition que toutes les équations soient polynomiales) au moyen d'un résultant (voir l'aide de Maple)

```
> cone_car:=proc(courbe,coord,S) local t,equas;
      equas:=subs(coord[1]=t*coord[1]+(1-t)*S[1],
                  coord[2]=t*coord[2]+(1-t)*S[2],
                  coord[3]=t*coord[3]+(1-t)*S[3],courbe);
      resultant(equas[1],equas[2],t);
end;
```

Voici un exemple avec la parabole d'équations  $x^2-y=0$ ,  $z=1$  et le point  $(0,0,0)$ .

```
> cone_car([x^2-y,z-1],[x,y,z],[0,0,0]);
```

$$-z y + x^2$$

et un exemple plus compliqué avec un cône construit sur la fenêtre de Viviani (!!)

```
> cone_car([x^2+y^2+z^2-1,x^2+y^2-x],[x,y,z],[0,0,0]);
```

$$x^2 y^2 - x^2 z^2 + y^4$$

```
>
```

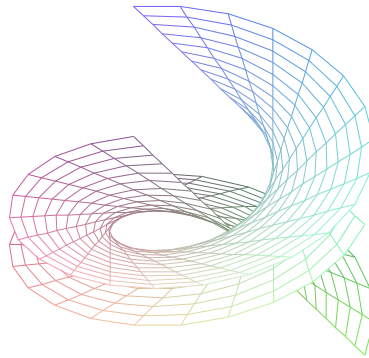
## Autres nappes développables

Le dernier type de nappe réglée développable est constitué de l'ensemble des tangentes à une courbe gauche. Rien de plus facile

```
> surface_tangentes:=proc(courbe,t,u) local tangent,i;
      tangent:=diff(courbe,t);
      [seq(courbe[i]+u*tangent[i],i=1..3)]
end;
```

Essayons avec une hélice circulaire

```
> Sigma:=surface_tangentes([cos(t),sin(t),t],t,u);
      :=[cos(t)-u sin(t), sin(t)+u cos(t), t+u]
> plot3d(Sigma,t=-Pi..Pi,u=-2..2);
```



On voit bien que la surface est pincée le long de la courbe qui est l'ensemble de ses points singuliers.



## Surfaces de révolution

Nous nous contenterons de faire tourner une courbe paramétrique  $t \rightarrow F(t)$  autour d'une droite  $A + \mathbf{R} \cdot u$ . Un paramétrage de la surface est donc  $(t, u) \rightarrow A + \mathbf{R}(u)[AF(t)]$  si  $\mathbf{R}(u)$  est la rotation vectorielle dans  $u$  autour de la droite  $\mathbf{R}$ . Une expression classique de cette rotation est, (lorsque  $\mathbf{R}$  est unitaire)

$$\mathbf{R}(u)(x) = (1 - \cos(u)) \frac{\mathbf{R} \times x}{\|\mathbf{R}\|} + \cos(u) x + \sin(u) \mathbf{R} \times x$$

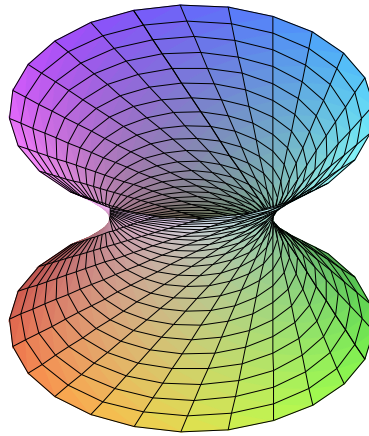
ce qui nous conduit à la fonction suivante (les paramètres  $\omega$  et  $A$  étant pris par défaut comme  $[0,0,1]$  et  $[0,0,0]$ )

```
> rotation:=proc(point,angle) local AF,i,omega,A,omega1,res;
    if nargs<4 then A:=[0,0,0] else A:=args[4] fi;
    if nargs<3 then omega:=[0,0,1] else omega:=args[3] fi;

    omega1:=evalm(omega/sqrt(linalg[dotprod](omega,omega)));
    AF:=linalg[vector]([seq(point[i]-A[i],i=1..3)]);
    res:=evalm(A
        +(1-cos(angle))*
        linalg[dotprod](AF,omega1,orthogonal)*omega1
        + cos(angle)*AF
        +sin(angle)*linalg[crossprod](omega1,AF)
    );
    simplify(convert(res,list));
end;
```

Un essai, en faisant tourner la droite  $x=1, y=z$  autour de l'axe  $Oz$

```
> Sigma:=rotation([1,t,t],u);
           := [cos(u) - sin(u) t, cos(u) t + sin(u), t]
> plot3d(Sigma,t=-2..2,u=-Pi..Pi,style=PATCH);
```



[ Maintenant faisons tourner un cercle autour d'une droite non coplanaire bien choisie

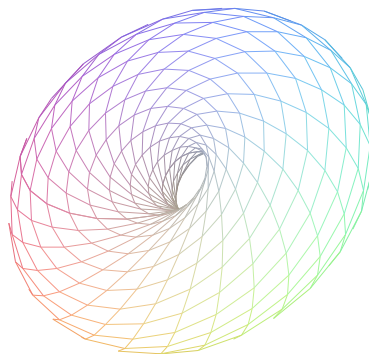
[ > **Sigma:=rotation([0,1+sqrt(2)\*cos(u),sqrt(2)\*sin(u)],v,[1,0,1]);**

$$:= \frac{1}{2}\sqrt{2}\sin(u) - \frac{1}{2}\cos(v)\sqrt{2}\sin(u) - \frac{1}{2}\sin(v)\sqrt{2} - \sin(v)\cos(u),$$

$$\cos(v) + \cos(v)\sqrt{2}\cos(u) - \sin(v)\sin(u),$$

$$\frac{1}{2}\sqrt{2}\sin(u) + \frac{1}{2}\cos(v)\sqrt{2}\sin(u) + \frac{1}{2}\sin(v)\sqrt{2} + \sin(v)\cos(u)$$

[ > **plot3d(Sigma,u=-Pi..Pi,v=-Pi..Pi);**



[ et on obtient un tore, qui contient donc beaucoup plus de cercles que ce que l'on pourrait penser a priori (c'est la théorie des cercles de Villarceau).

[ >

[ >

# Cours de mathématiques

par Denis Monasse

Ed. Vuibert

## Table des matières

- Plan général
- Algèbre générale
- Algèbre linéaire
- Réduction des endomorphismes
- Topologie des espaces métriques
- Espaces vectoriels normés
- Comparaison des fonctions
- Suites et séries numériques
- Fonctions d'une variable réelle
- Intégration
- Suites et séries de fonctions

- Séries entières
- Formes quadratiques
- Formes hermitiennes
- Séries de Fourier
- Calcul différentiel
- Equations différentielles
- Espaces affines
- Courbes
- Surfaces
- Intégrales multiples
- Index