

## Courbes

Dans toute la suite, nous supposons qu'un arc est toujours donné par une liste d'expressions dépendant d'un paramètre. C'est ainsi que le cercle du plan de centre 0 et de rayon 2 sera donné en Maple par

```
> [2*cos(t), 2*sin(t)];
```

### Invariants affines

#### Tangente et vecteur tangent

Le vecteur tangent au point régulier  $t_0$  à l'arc donné par  $t \mapsto [x(t), y(t)]$  sera donné par

```
> vecteur_tangent := (f, t, t0) -> subs(t=t0, diff(f, t));  
                                vecteur_tangent := (f, t, t0)    subs(t = t0, diff(f, t))
```

```
> vecteur_tangent([a*cos(t), b*sin(t)], t, u);  
                                [-a sin(u), b cos(u)]
```

Cette fonction donne également le vecteur tangent à un arc gauche

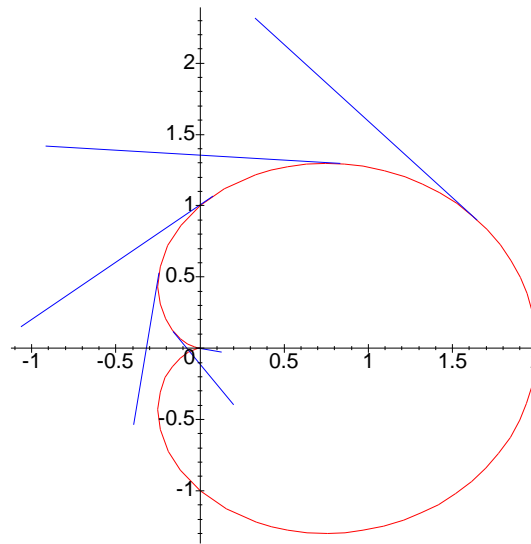
```
> vecteur_tangent([cos(t), sin(t), t^2], t, t0);  
                                [-sin(t0), cos(t0), 2 t0]
```

Ceci nous permettra de dessiner un arc et un certain nombre de ses vecteurs tangents à l'aide de la fonction qui suit:

`courbe_et_tangents(parametrage, liste_de_parametres)` tracera la courbe et les vecteurs tangents aux points demandés de la courbe. La courbe sera une courbe paramétrée au sens usuel de Maple, c'est à dire de la forme  $[x(t), y(t), t = a .. b]$

```
> courbe_et_tangents:=proc(parametrage, liste)  
    local t, i, n, courbe, points, vecteurs, segments, couleurs;  
    t:=op([3,1],parametrage);  
    courbe:=[op(1..2,parametrage)];  
    n:=nops(liste);  
    points:=[seq(subs(t=liste[i], courbe), i=1..n)];  
    vecteurs:=[seq(vecteur_tangent(courbe, t, liste[i]), i=1..n)];  
    segments:=seq([points[i], points[i]+vecteurs[i]],  
                  i=1..nops(liste));  
    couleurs:=[red, seq(blue, i=1..n)];  
    plot([parametrage, segments], color=couleurs,  
         seq(args[i], i=3..nargs));  
end;
```

```
> courbe_et_tangents([(1+cos(t))*cos(t), (1+cos(t))*sin(t), t=-Pi..Pi], [0, 0.5, 1, 1.5, 2, 2.5, 3]);
```



[ Dans le plan, l'équation de la tangente au point  $t_0$ , dans un système de coordonnées *ccords* peut être donnée par

```
[ > tangente:=(courbe,t,t0,coords)->
      linalg[det](linalg[matrix](
          [[op(coords),1],
            [op(subs(t=t0,courbe)),1],
            [op(subs(t=t0,diff(courbe,t))],0]]
          )):
[ > tangente([(1+cos(t))*cos(t),(1+cos(t))*sin(t)],t,t0,[x,y]);
  x sin(t0)2 -x cos(t0) -x cos(t0)2 +2 cos(t0) sin(t0)2 +cos(t0)2 +2 cos(t0)3 +cos(t0)2 sin(t0)2 +cos(t0)4 -2 sin(t0) cos(t0) y
    -sin(t0) y +sin(t0)2
[ > simplify(");
      x -2 x cos(t0)2 -x cos(t0) +2 cos(t0) +cos(t0)2 -2 sin(t0) cos(t0) y -sin(t0) y +1
[ > collect(",[x,y]);
      (1 -2 cos(t0)2 -cos(t0)) x +(-2 sin(t0) cos(t0) -sin(t0)) y +cos(t0)2 +2 cos(t0) +1
[ Suivant le même schéma, on peut obtenir l'équation du plan osculateur à un arc gauche
[ > plan_osculateur:=(courbe,t,t0,coords)->
```

```

linalg[det](linalg[matrix](
    [[op(coords),1],
    [op(subs(t=t0,courbe)),1],
    [op(subs(t=t0,diff(courbe,t))),0],
    [op(subs(t=t0,diff(courbe,t,t))),0]]
)):
> plan_osculateur([cos(t),sin(t),a*t],t,t0,[x,y,z]);

$$x \sin(t_0) a + \sin(t_0)^2 z - \sin(t_0)^2 a t_0 - \cos(t_0) y a + \cos(t_0)^2 z - \cos(t_0)^2 a t_0$$

> collect("[x,y,z]);

$$x \sin(t_0) a - \cos(t_0) y a + (\cos(t_0)^2 + \sin(t_0)^2) z - \sin(t_0)^2 a t_0 - \cos(t_0)^2 a t_0$$


```

## Points remarquables

### Points singuliers

La recherche des points singuliers d'un arc peut se faire sans difficulté à l'aide de la fonction

```

> points_singuliers:=(courbe,t)->
    solve({op(diff(courbe,t))},t):
> points_singuliers([(1+cos(t))*cos(t),(1+cos(t))*sin(t)],t);

$$\{t = \}$$

> points_singuliers([- (u^4+6*u^2-3)/(1+u^2)^2, 8*u^3/(1+u^2)^2],u);

$$\{u = 0\}, \{u = \text{RootOf}(-3 + \_Z^2)\}$$

> allvalues("[2]);

$$\{u = \sqrt{3}\}, \{u = -\sqrt{3}\}$$


```

### Points non biréguliers et points d'inflexion

De même pour la recherche des points non biréguliers (en général des points d'inflexion)

```

> points_non_bireguliers:=proc(courbe,t)
    local x,y;
    x:=courbe[1]; y:=courbe[2];
    solve(diff(x,t$2)*diff(y,t)-diff(x,t)*diff(y,t$2)=0,t)
end:
> points_non_bireguliers([t/(1+t^3),(1+t^2)/(1+t^3)],t);

$$\frac{1}{2} \%2 + \frac{2}{(-4 + 4 I \sqrt{3})^{1/3}}, -\frac{1}{4} \%2 - \frac{1}{(-4 + 4 I \sqrt{3})^{1/3}} + \frac{1}{2} I \sqrt{3} \quad \frac{1}{2} \%2 - \frac{2}{(-4 + 4 I \sqrt{3})^{1/3}},$$


```

$$-\frac{1}{4}\%2 - \frac{1}{(-4 + 4I\sqrt{3})^{1/3}} - \frac{1}{2}I\sqrt{3} \quad \frac{1}{2}\%2 - \frac{2}{(-4 + 4I\sqrt{3})^{1/3}}$$

$$\%1 := \frac{1}{(-4 + 4I\sqrt{3})^{1/3}}$$

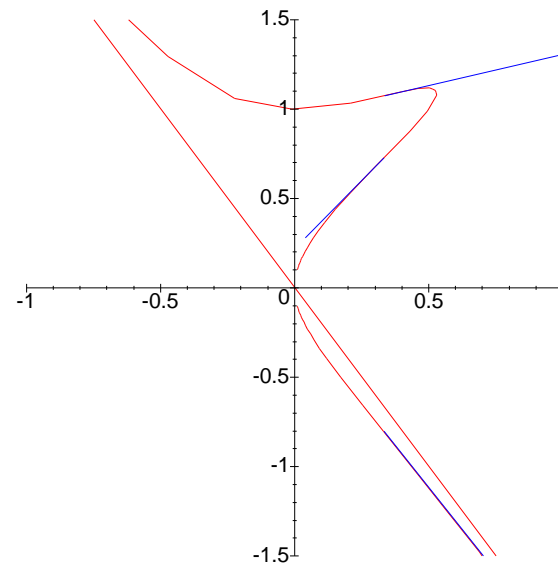
$$\%2 := (-4 + 4I\sqrt{3})^{1/3}$$

> **evalf([""]);**

[1.532088886 -1.10<sup>-9</sup>I, -1.879385242, .3472963560]

Voilà par exemple la courbe avec ses trois tangentes d'inflexion

> **courbe\_et\_tangents([t/(1+t^3), (1+t^2)/(1+t^3), t=-10..10],  
[1.53, -1.88, 0.35], view=[-1..1, -1.5..1.5]);**



### **Points multiples**

Pour la recherche des points multiples, nous vous proposons la méthode suivante, tout au moins en ce qui concerne une courbe paramétrée par des fonctions algébriques

```

> points_multiples:=proc(courbe,t,tpprime)
    local x1,y1,x2,y2;
    x1:=courbe[1];
    x2:=subs(t=tpprime,courbe[1]);
    y1:=courbe[2];
    y2:=subs(t=tpprime,courbe[2]);
    solve({(x1-x2)/(t-tpprime),(y1-y2)/(t-tpprime)},{t,tpprime})
end:
> pm:=[points_multiples([(u^4-12*u^2+3)/(1+u^2)^2, 2*u*(5*u^2-3)/(1+u^2)^2],u,v)];
pm := { u = -RootOf(5 _Z^2 - 3), v = RootOf(5 _Z^2 - 3) },
      { u =  $\frac{1}{3}$  RootOf(_Z^4 - 54 _Z^2 + 9) (-54 + RootOf(_Z^4 - 54 _Z^2 + 9)^2), v = RootOf(_Z^4 - 54 _Z^2 + 9) }
> map(allvalues,"dependent");
{ v =  $\frac{1}{5}\sqrt{15}$ , u =  $-\frac{1}{5}\sqrt{15}$  }, { u =  $\frac{1}{5}\sqrt{15}$ , v =  $-\frac{1}{5}\sqrt{15}$  }, { v =  $\sqrt{15} + 2\sqrt{3}$ , u =  $\frac{1}{3}(\sqrt{15} + 2\sqrt{3})(-54 + (\sqrt{15} + 2\sqrt{3})^2)$  },
      { v =  $-\sqrt{15} - 2\sqrt{3}$ , u =  $\frac{1}{3}(-\sqrt{15} - 2\sqrt{3})(-54 + (-\sqrt{15} - 2\sqrt{3})^2)$  },
      { v =  $\sqrt{15} - 2\sqrt{3}$ , u =  $\frac{1}{3}(\sqrt{15} - 2\sqrt{3})(-54 + (\sqrt{15} - 2\sqrt{3})^2)$  },
      { v =  $-\sqrt{15} + 2\sqrt{3}$ , u =  $\frac{1}{3}(-\sqrt{15} + 2\sqrt{3})(-54 + (-\sqrt{15} + 2\sqrt{3})^2)$  }
> evalf("");
[ { v = .7745966692, u = -.7745966692 }, { v = -.7745966692, u = .7745966692 }, { u = -.4088817066, v = 7.337084962 },
  { v = -7.337084962, u = .4088817066 }, { v = .408881730, u = -7.337084943 }, { u = 7.337084943, v = -.408881730 } ]
donc 6 solutions, ce qui correspond à trois points doubles (à cause de la symétrie entre u et v) dont nous obtenons ainsi les coordonnées
> map(x->subs(x,[(u^4-12*u^2+3)/(1+u^2)^2, 2*u*(5*u^2-3)/(1+u^2)^2]),"");
[[-1.500000000, 0], [-1.500000000, 0], [.7500001969, 1.299038133], [.7500001969, -1.299038133],
  [.7499999989, -1.299038109], [.7499999989, 1.299038109]]

```

### Types de points

Si l'on cherche à trouver le type d'un point (point banal, point d'inflexion, point de rebroussement de première ou deuxième espèce), on pourra utiliser la fonction suivante qui retourne à la fois le type du point et un vecteur tangent

```

> vecteur_nul:= (v,precision) -> evalb((abs(v[1])<=precision) and

```

```

                                (abs(v[2])<=precision)):
vecteurs_lies:=(v1,v2,precision) ->
                                evalb(abs(v1[1]*v2[2]-v2[1]*v1[2])<=precision):

type_point:=proc(courbe,t,t0) local precision,p,q,
                                vect_p,vect_p0,vect_q,le_type;
                                if nargs>3 then precision:=args[4]
                                    else precision:=0 fi;
                                p:=1; vect_p:=diff(courbe,t);
                                while vecteur_nul(eval(subs(t=t0,vect_p)),precision) do
                                    p:=p+1;
                                    vect_p:=diff(vect_p,t)
                                od;
                                vect_p0:=eval(subs(t=t0,vect_p));
                                vect_q:=diff(vect_p,t);
                                q:=p+1;
                                while vecteurs_lies(vect_p0,
                                                        eval(subs(t=t0,vect_q)),
                                                        precision)
                                do
                                    q:=q+1;
                                    vect_q:=diff(vect_q,t)
                                od;
                                if type(p,odd) then
                                    if type(q,even) then le_type:='point banal'
                                    else le_type:='point d'inflexion'
                                    fi
                                else
                                    if type(q,odd) then le_type:='point de rebroussement 1'
                                    else le_type:='point de rebroussement 2'
                                    fi
                                fi;
                                le_type,vect_p0
                            end:
> type_point([sin(t)^3,tan(t)^5/(1+t)],t,0.0,1e-6);
                                point d'inflexion, [6, 0]

```

## Invariants métriques des arcs plans

## [-] Abscisse curviligne

[ La fonction suivante calcule la longueur d'un arc paramétré par t, celui ci variant d'une origine à une extrémité

```
[ > norme_vect:=v->sqr(v[1]^2+v[2]^2):
```

```
    abscisse_curviligne:=(courbe,t,origine,extremite)->
        int(norme_vect(diff(courbe,t)),t=origine..extremite):
```

```
[ > abscisse_curviligne([(1+cos(t))*cos(t),(1+cos(t))*sin(t)],t,0,u);
```

$$\frac{\sqrt{2+2\cos(u)} \sin(u) \sqrt{4}}{1+\cos(u)}$$

## [-] Repère de Frénet

[ Le calcul du repère de Frénet se fait de la manière suivante

```
[ > Frenet:=proc(courbe,t,t0) local derive,tFleche,norme;
    derive:=eval(subs(t=t0,diff(courbe,t)));
    norme:=sqr(derive[1]^2+derive[2]^2);
    tFleche:=[derive[1]/norme,derive[2]/norme];
    [tFleche,[-tFleche[2],tFleche[1]]]
```

```
    end:
```

```
[ > simplify(Frenet([(1+cos(t))*cos(t),(1+cos(t))*sin(t)],t,t));
```

$$-\frac{\sin(t)(2\cos(t)+1)}{\sqrt{2\cos(t)+2}}, \frac{\cos(t)+2\cos(t)^2-1}{\sqrt{2\cos(t)+2}}, -\frac{\cos(t)+2\cos(t)^2-1}{\sqrt{2\cos(t)+2}}, -\frac{\sin(t)(2\cos(t)+1)}{\sqrt{2\cos(t)+2}}$$

```
[ >
```

## [-] Courbure et rayon de courbure

[ Pour ce qui concerne la courbure, on se contente d'appliquer les formules du cours

```
[ > courbure:=proc(courbe,t,t0)
```

```
    local der1,der2;
    der1:=diff(courbe,t);
    der2:=diff(der1,t);
    eval(subs(t=t0,
        (der1[1]*der2[2]-der2[1]*der1[2])
        /((der1[1]^2+der1[2]^2)^(3/2))))
```

```
    end:
```

```
[ > simplify(courbure([(1+cos(t))*cos(t),(1+cos(t))*sin(t)],t,t));
```

$$\frac{3}{2} \frac{1}{\sqrt{2 \cos(t) + 2}}$$

et le rayon de courbure

```
> rayon_de_courbure:=(courbe,t,t0)->1/courbure(courbe,t,t0):
> simplify(rayon_de_courbure([(1+cos(t))*cos(t),(1+cos(t))*sin(t)],t,t));
```

$$\frac{2}{3} \sqrt{2 \cos(t) + 2}$$

## Centre de courbure et développée

Le centre de courbure demande un traitement direct; en effet l'utilisation des fonctions précédentes qui construisent le repère de Frénet et le rayon de courbure introduisent des racines carrées que Maple a ensuite beaucoup de mal à simplifier. On préférera donc la méthode suivante pour la développée

```
> developpee:=proc(courbe,t) local x,y,x1,y1,x2,y2,rapport;
    x:=courbe[1]; y:=courbe[2];
    x1:=diff(x,t); y1:=diff(y,t);
    x2:=diff(x1,t); y2:=diff(y1,t);
    rapport:=(x1^2+y1^2)/(x1*y2-y1*x2);
    [x-rapport*y1,y+rapport*x1]
end:
et pour le centre de courbure
> centre_de_courbure:=(courbe,t,t0)->
    eval(subs(t=t0,developpee(courbe,t))):
> simplify(developpee([(1+cos(t))*cos(t),(1+cos(t))*sin(t)],t));
```

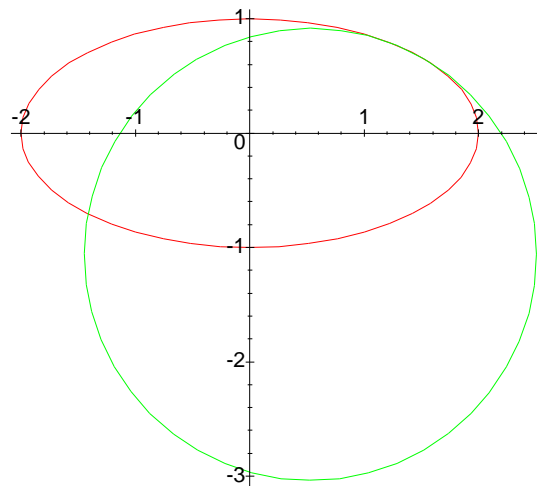
$$-\frac{1}{3} \cos(t)^2 + \frac{1}{3} \cos(t) + \frac{2}{3}, -\frac{1}{3} (\cos(t) - 1) \sin(t)$$

## Cercle osculateur

La fonction suivante calcule un paramétrage du cercle osculateur à un arc en un point t0

```
> cercle_osculateur:=proc(courbe,t,t0,u) local centre,rayon;
    centre:=centre_de_courbure(courbe,t,t0);
    rayon:=rayon_de_courbure(courbe,t,t0);
    [centre[1]+rayon*cos(u),centre[2]+rayon*sin(u)]
end:
> courbe:=[2*cos(t),sin(t)]:
cercle:=cercle_osculateur(courbe,t,Pi/4,u):
plot([op(courbe),t=0..2*Pi],[op(cercle),u=0..2*Pi]);
```





Une équation du cercle osculateur dans un système de coordonnées demande par contre de nouveau un traitement direct pour éviter des simplifications hasardeuses de racines carrées

```
> equation_cercle_osculateur:=proc(courbe,t,t0,coords)
    local x,y,x1,y1,x2,y2,rapport;
    x:=courbe[1]; y:=courbe[2];
    x1:=diff(x,t); y1:=diff(y,t);
    x2:=diff(x1,t); y2:=diff(y1,t);
    rapport:=(x1^2+y1^2)/(x1*y2-y1*x2);
    eval(subs(t=t0,
        (coords[1]-x+rapport*y1)^2+(coords[2]-y-rapport*x1)^2
        -(x1^2+y1^2)*rapport^2));
end;
```

```
> equation_cercle_osculateur([2*cos(t),sin(t)],t,Pi/4,[x,y]);
```

$$x - \frac{3}{8}\sqrt{2}^2 + y + \frac{3}{4}\sqrt{2}^2 - \frac{125}{32}$$

 Développante

[ La fonction suivante permet un calcul élémentaire d'une développante

```
> developpante:=proc(courbe,t,s0)
    local tFleche,s,norme,derive;
    derive:=diff(courbe,t);
    norme:=sqrt(derive[1]^2+derive[2]^2);
    s:=int(norme,t);
    [courbe[1]+(s0-s)*derive[1]/norme,
     courbe[2]+(s0-s)*derive[2]/norme]
end:
> simplify(developpante([cos(t),sin(t)],t,s0));
[cos(t) -sin(t) s0 +sin(t) t, sin(t) +cos(t) s0 -cos(t) t]
> simplify(developpante([(1+cos(t))*cos(t),(1+cos(t))*sin(t)],t,0));
[-3 cos(t)^2 +3 cos(t) +2, -3 (cos(t) -1) sin(t)]
```

## Equation intrinsèque

[ La résolution de l'équation intrinsèque  $c = f(s)$  (courbure en fonction de l'abscisse curviligne) se ramène à trois calculs de primitives à condition de se donner une origine de l'arc et l'angle d'un vecteur tangent unitaire avec l'axe Ox. Ceci conduit à la fonction

```
> equation_intrinseque:=proc(f,s,s0,origine,phi0)
    local phi,v;
    phi:=int(f,s=s0..v)+phi0;
    [origine[1]+int(cos(phi),v=s0..s),
     origine[2]+int(sin(phi),v=s0..s)]
end:
> equation_intrinseque(1/s,s,1,[0,0],0);
1/2 s sin(ln(s)) + 1/2 s cos(ln(s)) - 1/2, 1/2 s sin(ln(s)) - 1/2 s cos(ln(s)) + 1/2
```

## Invariants métriques des arcs gauches

### Abscisse curviligne

[ La fonction suivante calcule la longueur d'un arc paramétré par t, celui ci variant d'une origine à une extrémité

```
> norme_vect:=v->sqrt(v[1]^2+v[2]^2+v[3]^2):
abscisse_curviligne:=(courbe,t,origine,extremite)->
    int(norme_vect(diff(courbe,t)),t=origine..extremite):
> abscisse_curviligne([cos(t),sin(t),t],t,0,u);
```

$$u\sqrt{2}$$

## Repère de Frenet

Les formules du cours permettent le calcul du repère de Frénet

```
> produit_vect:=(v1,v2)->[v1[2]*v2[3]-v1[3]*v2[2],
                           v1[3]*v2[1]-v1[1]*v2[3],
                           v1[1]*v2[2]-v1[2]*v2[1]]:

Frenet3D:=proc(courbe,t,t0)
    local der1,der2,norme_t,norme_b,v,
           tFleche,bFleche;
    der1:=eval(subs(t=t0,diff(courbe,t)));
    norme_t:=norme_vect(der1);
    der2:=eval(subs(t=t0,diff(courbe,t$2)));
    v:=produit_vect(der1,der2);
    norme_b:=norme_vect(v);
    tFleche:=[der1[1]/norme_t,der1[2]/norme_t,
              der1[3]/norme_t];
    bFleche:=[v[1]/norme_b,
              v[2]/norme_b,v[3]/norme_b];
    [tFleche,produit_vect(bFleche,tFleche),bFleche]
end:

> simplify(Frenet3D([2*cos(t),sin(t),t],t,Pi/2));
      -2/5*sqrt(5),0,1/5*sqrt(5),[0,-1,0],1/5*sqrt(5),0,2/5*sqrt(5)
```

## Courbure

On applique de nouveau les formules du cours

```
> courbure3D:=proc(courbe,t,t0)
    local der1,der2,norme_t,norme_b,v;
    der1:=eval(subs(t=t0,diff(courbe,t)));
    norme_t:=norme_vect(der1);
    der2:=eval(subs(t=t0,diff(courbe,t$2)));
    v:=produit_vect(der1,der2);
    norme_b:=norme_vect(v);
    norme_b/norme_t^3
end:
```

```
> simplify(courbure3D([a*cos(t),a*sin(t),b*t],t,t));
```

$$\frac{\sqrt{a^2(b^2+a^2)}}{(b^2+a^2)^{3/2}}$$

## Torsion

Encore une fois les formules du cours

```
> torsion:=proc(courbe,t,t0)
    local der1,der2,der3,norme,v;
    der1:=eval(subs(t=t0,diff(courbe,t)));
    der2:=eval(subs(t=t0,diff(courbe,t$2)));
    der3:=eval(subs(t=t0,diff(courbe,t$3)));
    v:=produit_vect(der1,der2);
    norme:=norme_vect(v);
    linalg[det](linalg[matrix]([der1,der2,der3]))/norme^2
end:
> simplify(torsion([a*cos(t),a*sin(t),b*t],t,t));
```

$$\frac{b}{b^2+a^2}$$

```
>
```

## Equations intrinsèques

Les formules donnant les dérivées des vecteurs du repère de Frénet conduisent, si l'on connaît courbure et torsion en fonction de l'abscisse curviligne, à un système différentiel, dont la résolution conduit, moyennant un choix de repère de Frénet initial, à la connaissance du repère de Frénet en fonction de l'abscisse curviligne, puis par intégration de t(s) en choisissant une origine, à la connaissance d'un paramétrage de l'arc. La fonction suivante, étant données courbures et torsion en fonction de l'abscisse curviligne, cherche à calculer l'arc correspondant, sachant qu'à l'instant s0 son repère de Frénet est le repère canonique

```
> intrinseque3D:=proc(c,tau,s,s0) local sol1,sol2,sol3,
    systeme,inconnues,t,n,b,t1,t2,t3;
    systeme:=diff(t(s),s)=c*n(s),
    diff(n(s),s)=-c*t(s)+tau*b(s),
    diff(b(s),s)=-tau*n(s);
    inconnues:={t(s),n(s),b(s)};
    sol1:=dsolve({systeme,t(s0)=1,n(s0)=0,b(s0)=0},inconnues);
    sol2:=dsolve({systeme,t(s0)=0,n(s0)=1,b(s0)=0},inconnues);
    sol3:=dsolve({systeme,t(s0)=0,n(s0)=0,b(s0)=1},inconnues);
    t1:=eval(subs(sol1,t(s)));
    t2:=eval(subs(sol2,t(s)));
    t3:=eval(subs(sol3,t(s)));
end;
```

```

t3:=eval(subs(sol3,t(s)));
[int(t1,s),int(t2,s),int(t3,s)]
end:

```

```

> f:=intrinsic3D(3,2,s,0);

```

$$f := \frac{4}{13}s + \frac{9}{169}\sqrt{13}\sin(\sqrt{13}s), -\frac{3}{13}\cos(\sqrt{13}s), -\frac{6}{169}\sqrt{13}\sin(\sqrt{13}s) + \frac{6}{13}s$$

```

[ Avec une vérification:

```

```

> simplify(courbure3D(f,s,s));

```

3

```

> simplify(torsion(f,s,s));

```

2

# 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