# 1 Procédure `Lmin`

## 1.1 `Lmin` en cartésiennes

C'est une simple boucle qui somme la longueur des $n$ segments de droites.
On remarquera que $\delta$ est pris directement en calcul approché, on a en effet intérêt à passer en calcul approché le plus rapidement possible.

```
> Lmin:=proc(x,y,t0,t1,n)
> local delta,L,ti,i;
> delta:=evalf(t1-t0)/n;
> L:=0;
> for i from 1 to n do
>      ti:=t0+i*delta;
>      L:=L+sqrt((x(ti)-x(ti-delta))^2+(y(ti)-y(ti-delta))^2)
> od;
> L
> end:
```

L'essai pour le demi cercle est concluant :

```
> Lmin(cos,sin,0,Pi,10);
```

3.128689300

## 1.2 Procédure `LminApp`

Ici, il s'agit d'une boucle « tant que », qui tourne tant que deux valeurs succéssives ne sont pas assez proches.
L'astuce de travailler en $2^p$ est de limiter les calculs sachant que maple se souvient de ses calculs précédents...

```
> LminApp:=proc(x,y,t0,t1,e)
> local p;
> for p from 1 while abs(Lmin(x,y,t0,t1,2^p)-Lmin(x,y,t0,t1,2^(p+1)))>e
>    do od;
> Lmin(x,y,t0,t1,2^(p+1))
> end:
```

On essaye encore une fois :

```
> LminApp(cos,sin,0,Pi,0.01);
```

3.140331143

## 1.3 En polaires

L'idée est d'utiliser les procédures précédentes en créant en fontion de $\rho$, cos et sin les fonctions qui nous manquent, à savoir $x$ et $y$.

```
> LminPol:=proc(rho,t0,t1,n)
> local x,y;
> x:=t->rho(t)*cos(t);
> y:=t->rho(t)*sin(t);
> Lmin(x,y,t0,t1,n)
> end:
```

On essaye :

```
> LminPol(cos,0,Pi,10);
```

3.090169942

La deuxième procédure est aussi facile à écrire :

```
> LminPolApp:=proc(rho,t0,t1,e)
> local x,y;
> x:=t->rho(t)*cos(t);
> y:=t->rho(t)*sin(t);
> LminApp(x,y,t0,t1,e)
> end:
```

Elle fonctionne aussi correctement :

```
> LminPolApp(cos,0,Pi,0.01);
```

3.140331143

# 2 Procédure `Lmaj`

## 2.1 Coordonnées des $B_i$

Pour obtenir les coordonées de $B_i$, il faut calculer les équations des tangentes à la courbe en $A_{i-1}$ et $A_i$ et chercher, par un simple système linéaire de deux équations à deux inconnues, le point d'intersection de ces deux droites.
La tangente en $A_i$ est :

$$x'(t_i)(y - y(t_i)) - y'(t_i)(x - x(t_i)) = 0$$

Pour le point d'intersection, on obtient pour $x$ :

$$x = \frac{y'(t_i)\,x'(t_{i-1})\,x(t_i) - y'(t_{i-1})\,x'(t_i)\,x(t_{i-1}) - (y(t_i) - y(t_{i-1}))\,x'(t_i)\,x'(t_{i-1})}{y'(t_i)\,x'(t_{i-1}) - y'(t_{i-1})\,x'(t_i)}$$

On inverse les rôles de $x$ et $y$ pour obtenir $y$.

IL faut un peu ici expliquer le sens des variables choisies :
– `h` représente $i - 1$ ;
– `i` représente $i$ ;
– un 1 représente une dérivée première. Ainsi, `x1h` représente $x'(t_{i-1})$.

Rappelons aussi que `D` est l'opérateur de dérivation.

```
> Bx:=proc(x,y,t0,t1,n,i)
> local delta,th,ti,xh,xi,yh,yi,x1h,x1i,y1h,y1i,num,den;
> delta:=evalf(t1-t0)/n;
> th:=t0+(i-1)*delta;ti:=t0+i*delta;
> xh:=x(th);yh:=y(th);xi:=x(ti);yi:=y(ti);
> x1h:=D(x)(th);y1h:=D(y)(th);x1i:=D(x)(ti);y1i:=D(y)(ti);
> num:=y1i*x1h*xi-y1h*x1i*xh-(yi-yh)*x1i*x1h;
> den:=y1i*x1h-y1h*x1i;
> num/den
> end:
```

La procédure `By` n'est que la copie de la précédente :

```
> By:=proc(x,y,t0,t1,n,i)
> local delta,th,ti,xh,xi,yh,yi,x1h,x1i,y1h,y1i,num,den;
> delta:=evalf(t1-t0)/n;
> th:=t0+(i-1)*delta;ti:=t0+i*delta;
> xh:=x(th);yh:=y(th);xi:=x(ti);yi:=y(ti);
> x1h:=D(x)(th);y1h:=D(y)(th);x1i:=D(x)(ti);y1i:=D(y)(ti);
> num:=x1i*y1h*yi-x1h*y1i*yh-(xi-xh)*y1i*y1h;
> den:=x1i*y1h-x1h*y1i;
> num/den
> end:
```

### 2.2 Procédure `Lmaj`

Il faut ici calculer la longueur du segment $A_0B_1$, y ajouter dans une boucle les longueurs des segments $A_1A_{i+1}$ de $i = 1$ à $n-1$, et enfin ajouter la longueur du dernier segment $B_nA_n$.

```
> Lmaj:=proc(x,y,t0,t1,n)
> local delta,L,ti,i;
> delta:=evalf(t1-t0)/n;
> L:=sqrt((Bx(x,y,t0,t1,n,1)-x(t0))^2+(By(x,y,t0,t1,n,1)-y(t0))^2);
> for i from 1 to n-1 do
>    L:=L+sqrt((Bx(x,y,t0,t1,n,i+1)-Bx(x,y,t0,t1,n,i))^2
>              +(By(x,y,t0,t1,n,i+1)-By(x,y,t0,t1,n,i))^2)
> od;
> L:=L+sqrt((x(t1)-Bx(x,y,t0,t1,n,n))^2+(y(t1)-By(x,y,t0,t1,n,n))^2);
> L
> end:
```

On fait un essai avec les fonctions indiquées :

```
> Lmaj(cos,sin,0,Pi,10);
```

3.167688806

### 2.3 Procédure `LmajApp`

C'est le même procédé exactement que dans le cas de `LminApp`.

```
> LmajApp:=proc(x,y,t0,t1,e)
> local p;
> for p from 1 while abs(Lmaj(x,y,t0,t1,2^p)-Lmaj(x,y,t0,t1,2^(p+1)))>e
> do od;
> Lmaj(x,y,t0,t1,2^(p+1))
> end:
```

Un petit essai…

```
> LmajApp(cos,sin,0,Pi,0.01);
```

3.144118388

## 3 Utilisation des deux procédés

La structure de la procédure `L` est très simple :

```
> L:=proc(x,y,t0,t1,e)
> local p;
> for p from 1 while abs(Lmaj(x,y,t0,t1,2^p)-Lmin(x,y,t0,t1,2^p))>e do
> od;
> Lmin(x,y,t0,t1,2^p),Lmaj(x,y,t0,t1,2^p)
> end:
```

Un dernier essai :

```
> L(cos,sin,0,Pi,0.0001);
```

3.141572888, 3.141632087