1 Prog Edit Ajouter nxt OK Save

```
Rect(f,N,a,b):= {
local RV,RR,R,k,av,ap,C;
RV:=NULL; RR:=NULL;
for(k:=0;k<=N;k++) {
av:=a+k*(b-a)/N; //avant...
ap:=a+(k+1)*(b-a)/N; //aprŁs...
R:=polygone(point(av,0),point(av,f(av)),point(ap,f(av)),point(ap,0));
RR:=RR,couleur(R,cyan+rempli); //on remplit les rectangles
RV:=RV,couleur(R,bleu+line_width_3); //le contour en plus Øpais
}
C:=plot(f(x),x=a..b,affichage=rouge+line_width_3); //la courbe
return(RV,RR,C);//les 3 graphes superposØs
```

```
// Parsing Rect
// Warning: x declared as global variable(s) compiling Rect
Done
```

2 On choisit un ØlØment n qui nous donnera le nombre de rectangles.

3 Fig Edit Pointeur Move / Save

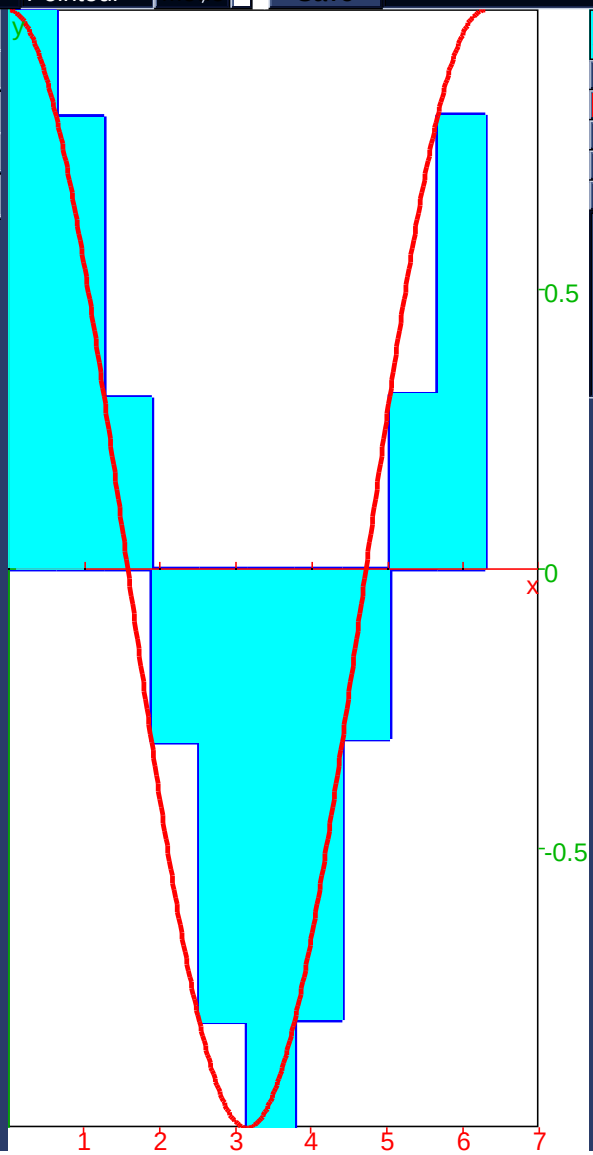
1 n:=element(1 .. 10,9.99

parameter(n,1,10,9.9

2 Rect(x->cos(x),n,0,2*P

Too large for display

3



x:3.2904
v:1.0312

in out cfg Menu

9.99 n

4 On fait la mŒme chose en modifiant juste les coordonnØes d'un point du polygone.

5 Prog Edit Ajouter nxt OK Save

```
Trap(f,N,a,b):= {
local RV,RR,R,k,av,ap,C;
RV:=NULL; RR:=NULL;
for(k:=0;k<=N;k++) {
av:=a+k*(b-a)/N;
ap:=a+(k+1)*(b-a)/N;
R:=polygone(point(av,0),point(av,f(av)),point(ap,f(ap)),point(ap,0));    //
RR:=RR,couleur(R,cyan+rempli);
RV:=RV,couleur(R,bleu+line_width_3);
}
C:=plot(f(x),x=a..b,affichage=rouge+line_width_3);
return(RV,RR,C);
}:;
```

```
// Parsing Trap
// Warning: x  declared as global variable(s) compiling Trap
```

Done

6

7 Fig Edit Pointeur Move / Save

1

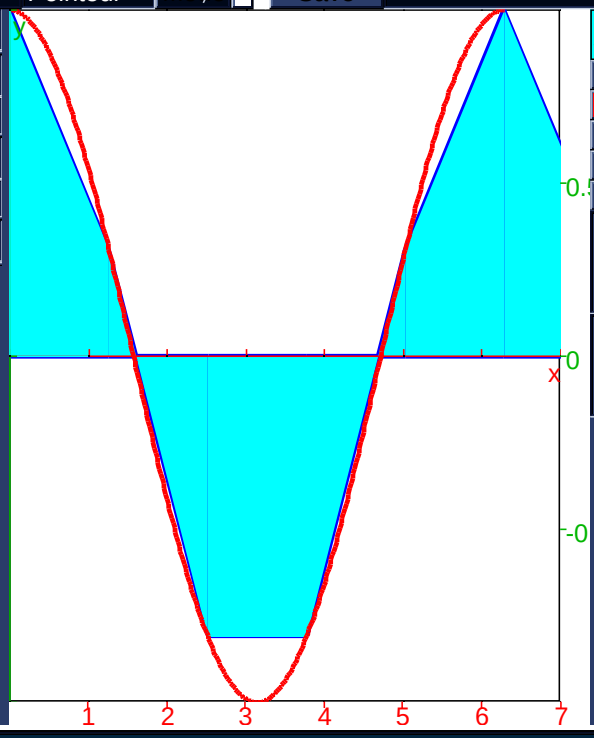
2 n:=element(1 .. 10,5.04

parameter(n,1,10,5.0

3 Trap(x->cos(x),floor(n)

Too large for display

4



x:3.1671
y:1.2844

in

out

cfg

Menu

6.21 n

6.21 n

8 On passe àl'approximation numØrique en utilisant sum.

9 Prog Edit Ajouter nxt OK Save

```
aireRect(f,N,a,b):= {
local k;
evalf(sum((b-a)/N*f(a+k*(b-a)/N),k=1..N));
}:;
```

```
// Parsing aireRect
// Success compiling aireRect
```

Done

10 aireRect(x->x^2,10,0,1)

0.385

11

12 Prog Edit Ajouter nxt OK Save

```
aireTrap(f,N,a,b):= {
local k;
evalf((b-a)/N*((f(a)+f(b))/2+sum(f(a+k*(b-a)/N),k=1..N-1)));
}:;
```

```
// Parsing aireTrap
// Success compiling aireTrap
```

Done

13 aireTrap(x->x^2,10,0,1)

0.335

14 Alors que le rØsultat attendu est

15 evalf(int(x^2,x=0..1))

0.333333333333

16 Pour la somme de Riemann, on utilise sum_riemann

17 Prog Edit Ajouter nxt OK Save

```
Riem(f,a,b):= {
sum_riemann(((b-a)/P)*f(a+p*(b-a)/P),⌈P,p⌉)  }:;
```

```
// Parsing Riem
// Warning: P p  declared as global variable(s) compiling Riem
```

Done

18 Riem(x->x^2,-5,2)

$$\frac{133}{3}$$

19 On vØrifie

20 int(x^2,x=-5..2)

$$\frac{133}{3}$$

21 CALCULS DE PI

22 Avec la mØthode des rectangles appliquØ àx->sqrt(1-x^2) on obtient

23 4*aireRect(x->sqrt(1-x^2),10000,0,1)

Evaluation time: 0.45

3.14139147761

24 Avec la mØthode des trapŁzes appliquØ àx->sqrt(1-x^2) on obtient

25 4*aireTrap(x->sqrt(1-x^2),10000,0,1)

Evaluation time: 0.31

3.14159147761

26 Alors qu'on devrait obtenir

27 evalf(Pi)

3.14159265359

28 On note quand mŒme que la mØthode des trapŁzes est plus efficace.

29 FORMULE DE MACHIN

30 Prog Edit Ajouter nxt OK Save

```
mini(u,p):={
local k;
k:=0;
while(evalf((u^(2*k+3))/(2*k+3))>10^(-p)) {
k:=k+1}
};;
```

```
// Parsing mini
// Success compiling mini
```

Done

31

32 Prog Edit Ajouter nxt OK Save

```
greg(n,a,p):={
local S,k;
S:=0;
for(k:=0;k<=n+1;k++) {
S:=S+evalf(((-1)^k*a^(2*k+1))/(2*k+1),p+1);
};
return(S);
};;
```

```
// Parsing greg
// Success compiling greg
```

Done

33 mini(1/5,100)

69

34 Il faut aller jusqu'au rang 69 pour avoir 100 bonnes dØcimales

35 4*(4*greg(69,1/5,100)-greg(69,1/239,100))

3.14159265359

36 4*(4*greg(69,1/5,100)-greg(69,1/239,100))-evalf(Pi,100)

- 2.85746847821e-101

37 Pas mal...

38