Correction du TP 7

B. Landreau

Polynomes et interpolation polynomiale

Exercice 1 : calculs de PGCD

a) Calcul du PGCD

C'est l'algorithme d'Euclide.
Il est interessant de faire afficher chaque division euclidienne enutilisant printf.
Sachant que le PGCD n'est defini qu'a un inversible pres, on convient generalement de retourner un polynome unitaire, en utiliserant**lcoeff**.

```
> euclide:=proc(A,B)
  local a,b,q,r;
  #option trace;
  a:=A;b:=B;
  while b<>0 do
    q:=quo(a,b,X);
    r:=rem(a,b,X);
    printf("\n %a  =  (%a)  * (%a)  +
  %a",a,b,q,r);
    a:=b;
    b:=r;
  od;
  RETURN(a/lcoeff(a));
  end:
> euclide(X^12-2,X^8-1);
```

```
X^12-2  =  (X^8-1)  * (X^4)  +  -2+X^4
X^8-1  =  (-2+X^4)  * (X^4+2)  +  3
-2+X^4  =  (3)  * (-2/3+1/3*X^4)  +  0
```

1

Attention, cet algorithme ne marche pas pour les entiers puisqu'on utilise quo et rem qui sont propres aux polynomes.
Il suffirait de les remplacer par irem et iquo.

b) Calcul d'un couple de Bezout

C'est l'algorithme d'Euclide etendu.

```
> euclide_etendu:=proc(A,B)
  local a,b,q,r,s,u,t,v;
  #option trace;
  a:=A;b:=B;
  u:=1;s:=0;
  while b<>0 do
    q:=quo(a,b,X);
    r:=rem(a,b,X);
    printf("\n %a  =  (%a)  * (%a)  +
  %a",a,b,q,r);
    t:=s;
    s:=u-q*s;
    u:=t;
    a:=b;
    b:=r;
  od;
  # verification
  v:=quo(a-A*u,B,X);
  # normalisation
  u:=sort(u/lcoeff(a));
  v:=sort(v/lcoeff(a));
```

```
  a:=a/lcoeff(a);
  printf("\n\n (%a) * (%a)  + (%a) *
  (%a) =  %a",A,u,B,v,expand(A*u+B*v));
  RETURN([a,u,v]);
  end:
```

```
> euclide_etendu(X^12-2,X^8-1);

 X^12-2  =  (X^8-1)  * (X^4)  +  -2+X^4
 X^8-1  =  (-2+X^4)  * (X^4+2)  +  3
 -2+X^4  =  (3)  * (-2/3+1/3*X^4)  +  0

 (X^12-2) * (-1/3*X^4-2/3)  + (X^8-1) * (1/3*X^8
+2/3*X^4+1/3) =  1
```

$$\left[1, -\frac{1}{3}X^4 - \frac{2}{3}, \frac{1}{3}X^8 + \frac{2}{3}X^4 + \frac{1}{3}\right]$$

Ca marche!

Exercice 2 : interpolation de Lagrange

a) subdvision

Ecrire une fonction sub(n,a,b) qui fabrique une liste (ou un vecteur) de longueur n+1
contenant les nombres a+k(b-a)/n.

Le plus simple est d'utiliser l'instruction **seq**.

```
> sub1:=proc(n,a,b)
  local k;
  RETURN([seq(a+k*(b-a)/n,k=0..n)]);
  end:
```

On peut aussi utiliser la structure de vecteur mais attention dans ce cas
il faut tenir compte du fait que l'indice d'un vecteur

commence a 1 et non 0.

```
> sub2:=proc(n,a,b)
  local k;
  RETURN(vector(n+1,k->a+(k-1)*(b-a)/n))
  ;
  end:
```

Essayons de suite

```
> sub1(5,0,1);
```

$$\left[0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\right]$$

```
> sub2(5,0,1);
```

$$\left[0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\right]$$

Ca baigne!

b) Construction de la liste des ordonnees

Etant donnees une fonction f et une liste x contenant des nombres x[i] construire une liste y contenant f(x[i]).

On pourra utiliser la fonction **map**

On peut utiliser a nouveau l'instruction seq.

Si x est une liste, le nombre d'elements de la liste s'obtient a l'aide de **nops,**

si c'est un vecteur, c'est **vectdim** (qui necessite linalg**).**

Si on ne veut pas charger tout linalg, on ecrit **linalg[vectdim].**

```
> valeurs:=proc(f,x)
```

```
  local n;
  if type(x,list)=true then
    n:=nops(x);
  fi;
  if type(x,vector)=true then
     n:=linalg[vectdim](x);
  fi;
  RETURN([seq(f(x[i]),i=1..n)]);
  end:
>
>
```

```
> f:=x->x^2;
  x1:=sub1(5,0,1);y1:=valeurs(f,x1);
  x2:=sub2(5,0,1);y2:=valeurs(f,x2);
```

$$f := x \to x^2$$

$$x1 := \left[0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\right]$$

$$y1 := \left[0, \frac{1}{25}, \frac{4}{25}, \frac{9}{25}, \frac{16}{25}, 1\right]$$

$$x2 := \left[0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\right]$$

$$y2 := \left[0, \frac{1}{25}, \frac{4}{25}, \frac{9}{25}, \frac{16}{25}, 1\right]$$

On peut aussi apprendre a utilier la fonction **map**(f,x) qui marche indifferement pour les listes et les vecteurs.
Son effet est d'appliquer la fonction f a tous les elements de la structure x.

```
> valeurs:=proc(f,x) RETURN(map(f,x))
  end:
```

ou encore avec la fleche

```
> valeurs:=(f,x) ->map(f,x);
```

$$valeurs := map$$

```
> x1:=sub1(5,0,1);y1:=valeurs(f,x1);
  x2:=sub2(5,0,1);y2:=valeurs(f,x2);
```

$$x1 := \left[0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\right]$$

$$y1 := \left[0, \frac{1}{25}, \frac{4}{25}, \frac{9}{25}, \frac{16}{25}, 1\right]$$

$$x2 := \left[0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1\right]$$

$$y2 := \left[0, \frac{1}{25}, \frac{4}{25}, \frac{9}{25}, \frac{16}{25}, 1\right]$$

c) Construction du polynome d'interpolation

```
> X:=sub1(4,-Pi,Pi);Y:=valeurs(sin,X);
```

$$X := \left[-\pi, -\frac{1}{2}\pi, 0, \frac{1}{2}\pi, \pi\right]$$

$$Y := [0, -1, 0, 1, 0]$$

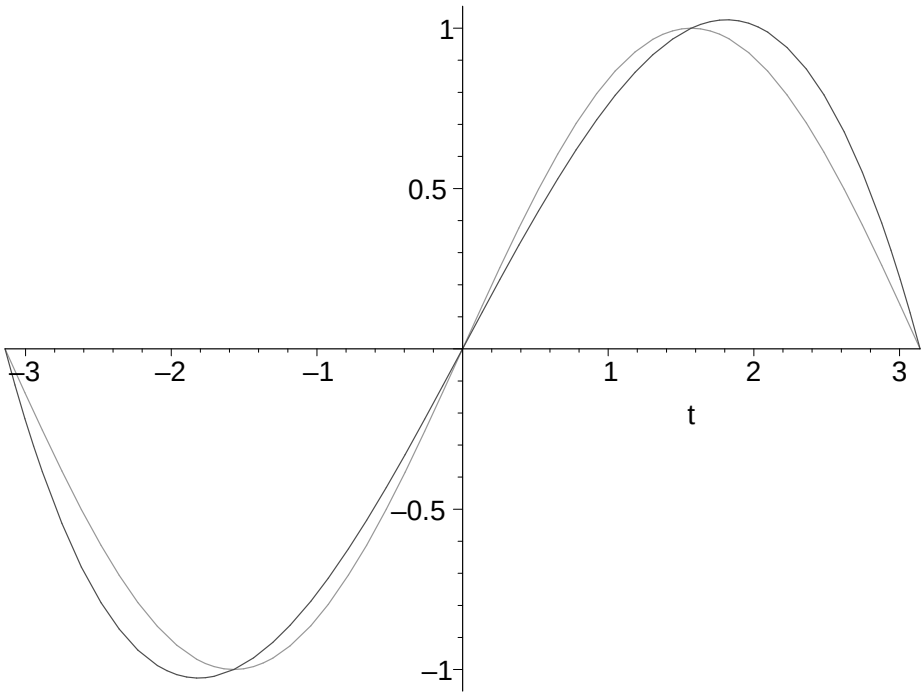
```
> P4:=interp(X,Y,t);
```

$$P4 := -\frac{8}{3}\frac{t^3}{\pi^3} + \frac{\frac{8}{3}t}{\pi}$$

Pour tracer le garphe de f sur [a,b], on met soit plot(f,a..b),

soit plot(f(t),t=a..b).
Pour tracer plusieurs fonctions, on utilise
plot({f(t),g(t)},t=a..b);
Ici, il faut savoir que la valeur retournee pat interp est un polynome formel, donc P4 est une expression en t.
Il ne faut pas mettre P4(t). on peut aussi si on prefere creer une fonction polynomiale p4:=t->P4 mais ce n'est pas vraiment necessaire.

```
> plot({sin(t),P4},t=-Pi..Pi);
```



On verifie bien que P4 rejoint sin aux 5 points imposes.

```
>
>
```

Exercice 3 : interpolation avec points

equidistants

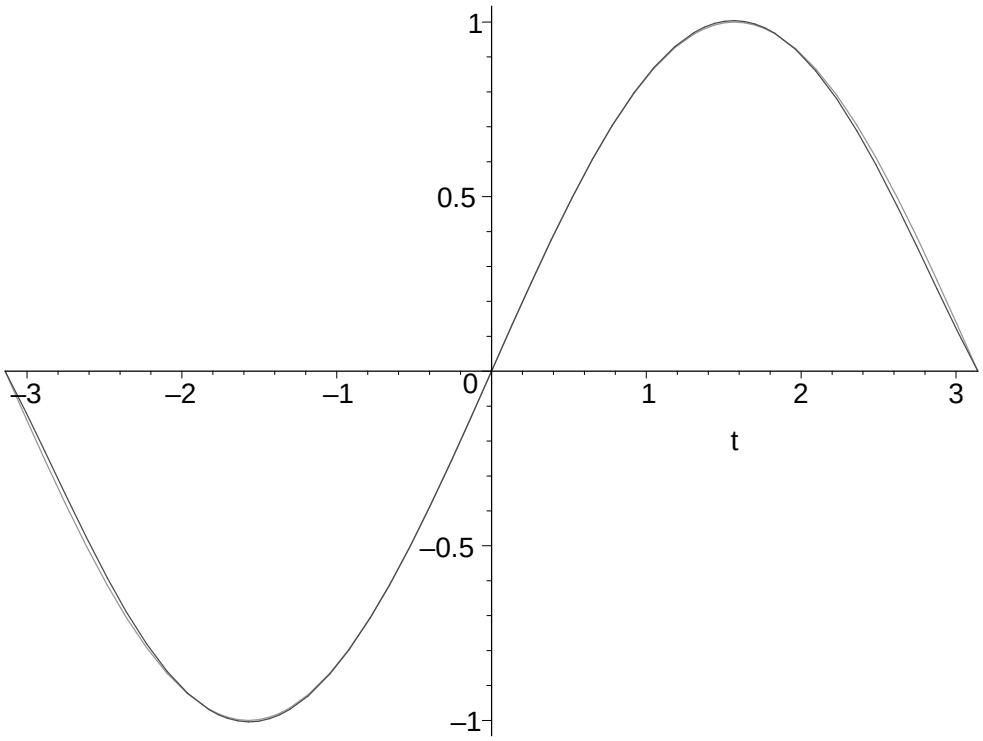
```
>
>
```

Il suffit d'automatiser ce que l'on a fait dans un cas particulier a l'exercie precedent.

```
> interequid:=proc(f,n,a,b)
  local X,Y,p,P;
  X:=sub1(n,a,b);
  Y:=map(f,X);
  P:=interp(X,Y,t);
  plot({f(t),P},t=a..b);
  end:
```

Essayons.

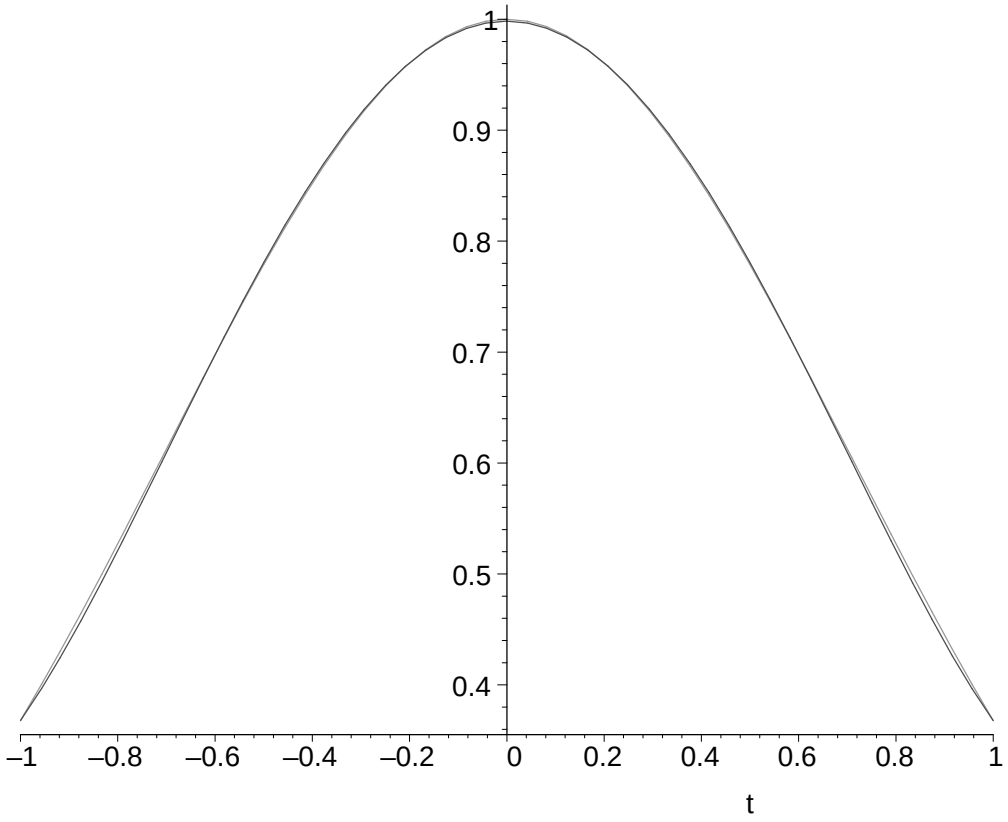
```
> interequid(sin,5,-Pi,Pi);
```



Ca marche, deja avec 6 points, on voit a peine la difference entre les 2 courbes.

```
> f:=x->exp(-x^2);interequid(f,5,-1,1);
```

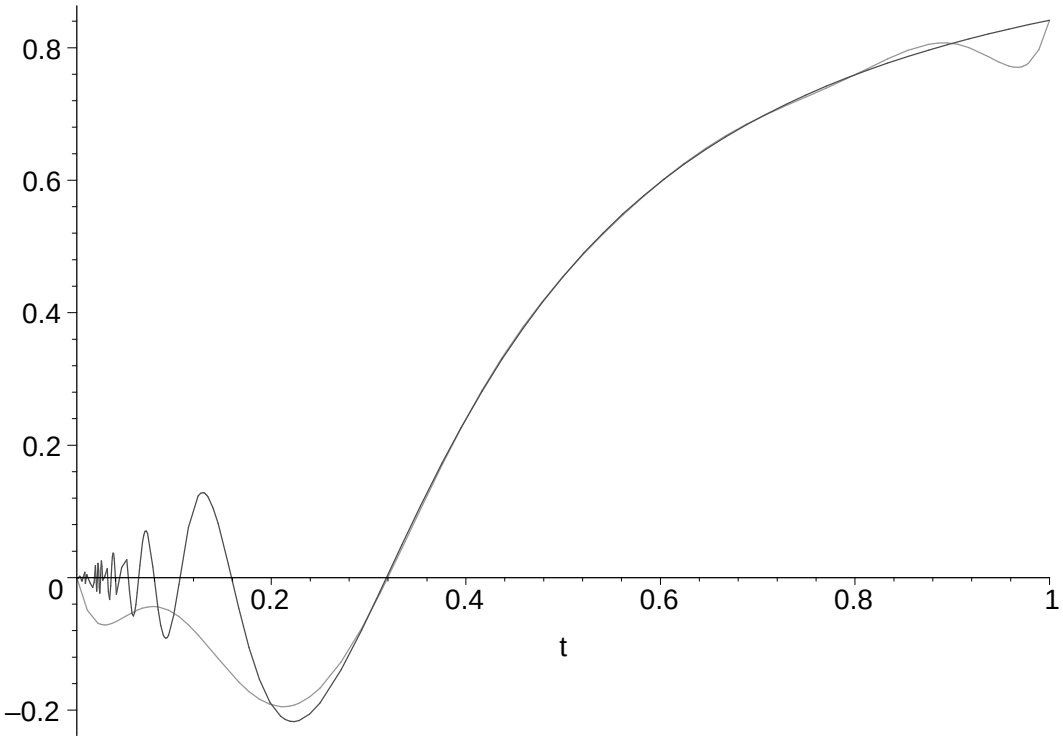
$$f := x \to \mathbf{e}^{(-x^2)}$$



Prenons un exemple pas facile a interpoler.

```
> f:=x->x*sin(1/x);interequid(f,10,0.0001,1
  );
```

$$f := x \to x \sin\left(\frac{1}{x}\right)$$



Eh oui, la c'est dur!

Evaluation de l'erreur commise

Essayons de calculer l'erreur sup|P-sin|.

```
> pi:=evalf(Pi);
```

$$\pi := 3.141592654$$

```
> X:=sub1(5,-pi,pi);Y:=map(sin,X);P:=interp
  (X,Y,t);
```

$X := [-3.141592654, -1.884955592, -.628318530, .628318530,$

$1.884955592, 3.141592654]$

$Y := [.4102067615\ 10^{-9}, -.9510565163, -.5877852517,$

$.5877852517, .9510565163, -.4102067615\ 10^{-9}]$

$$P := .005970529338\,t^5 - .2\,10^{-10}\,t^4 - .1160589566\,10^{-8} - .1600180846\,t^3 + .9977313564\,t + .3\,10^{-9}\,t^2$$

```
> maximize(sin(t)-P,t=-Pi..Pi);
```

$$.0005212509238$$

```
> test_convergence:=proc(f,n,a,b)
  local X,Y,p,P;
  X:=sub1(n,a,b);
  Y:=map(f,X);
  P:=interp(X,Y,t);
  #plot({f(t),P},t=a..b);
  printf("\n%d points ecart
  :%g",n+1,maximize(abs(evalf(f(t)-P)),t=a.
  .b));
  end:


> for n to 15 do
  test_convergence(sin,n,-pi,pi);od;

2 points ecart :1
3 points ecart :1
4 points ecart :.026807
5 points ecart :.082605
6 points ecart :.000521
7 points ecart :.002723
8 points ecart :7.002311e-06
9 points ecart :5.136573e-05
10 points ecart :4.923378e-07
11 points ecart :9.188587e-07
12 points ecart :3.950000e-08
13 points ecart :1.521979e-08
14 points ecart :1.029246e-08
15 points ecart :1.415800e-09
16 points ecart :3.394320e-09
```

On remarque un phenomene interessant et tout a fait typique : au debut de n=1 a 8 environ on ameliore sensiblement l'interpolation
puis apres le gain est minime et lent.

```
>
```

Exercice 4 : interpolation avec points aleatoires

Il faut d'abord mettre au point une procedure de subdivision aleatoire de [a,b].
On divise [a,b] en un grand nombre de points (N=1000)
et on fait n +1 tirages aleatoires d'un nombre entier entre 0 et N a l'aide de la fonction rand().
Attention rand(0..N) retourne en fait une procedure, il faut donc ecrire par exemple tir:=rand(0..N)
puis appeler tir().

```
> subalea:=proc(n,a,b)
  local N,L,tir;
  N:=10000;
  tir:=rand(0..N);
  L:=[seq(evalf(a+tir()*(b-a)/N),i=1..n+1)]
  ;
  L:=sort(L);
  RETURN(L);
  end:
```

```
> subalea(10,0,1);
```

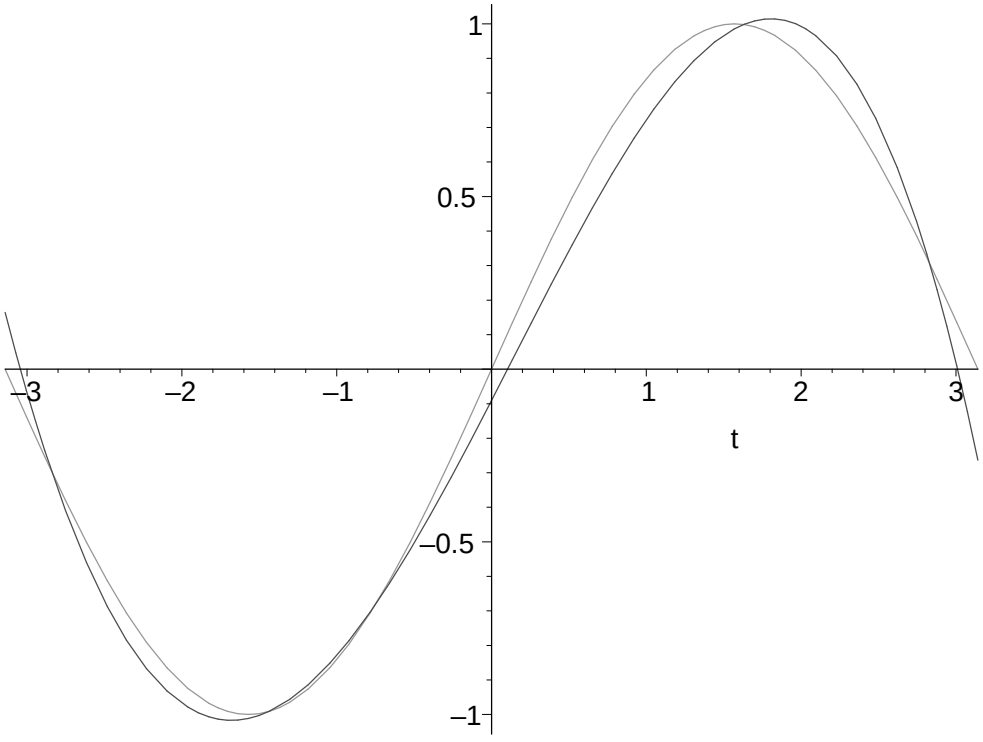
[.1388000000, .1585000000, .2622000000, .2691000000,

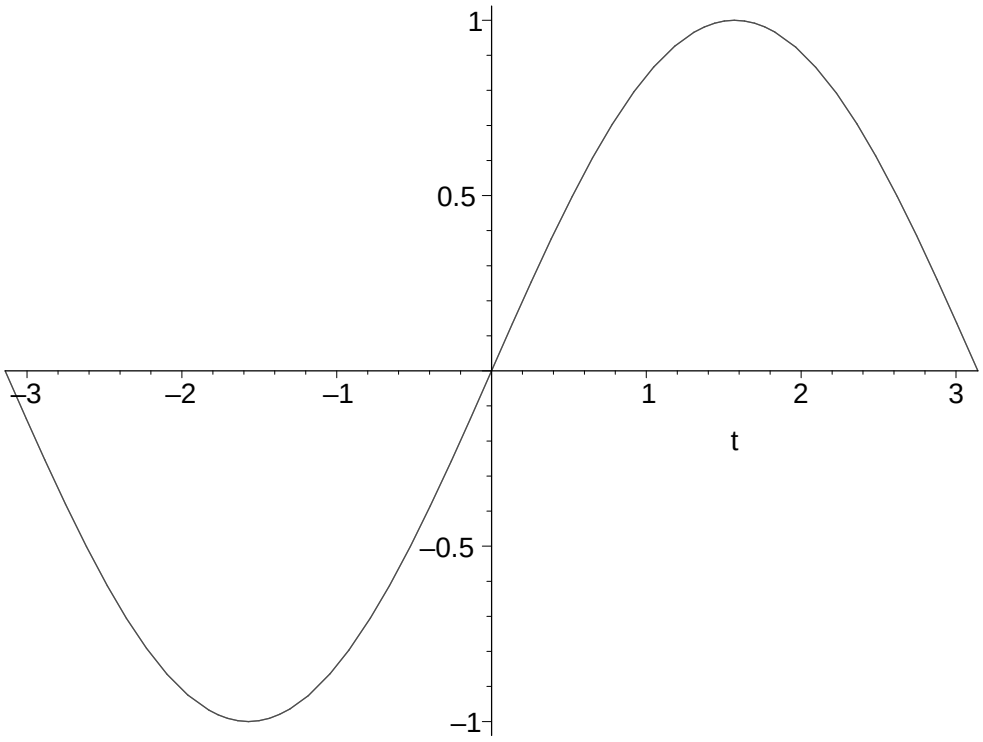
.3826000000, .4058000000, .5412000000, .6184000000,
.6861000000, .8689000000, .9403000000]

```
> interquelc:=proc(f,points,a,b)
  local X,Y,P;
  X:=points;
  Y:=map(f,X);
  P:=interp(X,Y,t);
  plot({f(t),P},t=a..b);
  end:
```

Essayons

```
> points:=subalea(4,-Pi,Pi):interquelc(sin,
  points,-Pi,Pi);
  points:=subalea(10,-Pi,Pi):interquelc(sin
  ,points,-Pi,Pi);
```



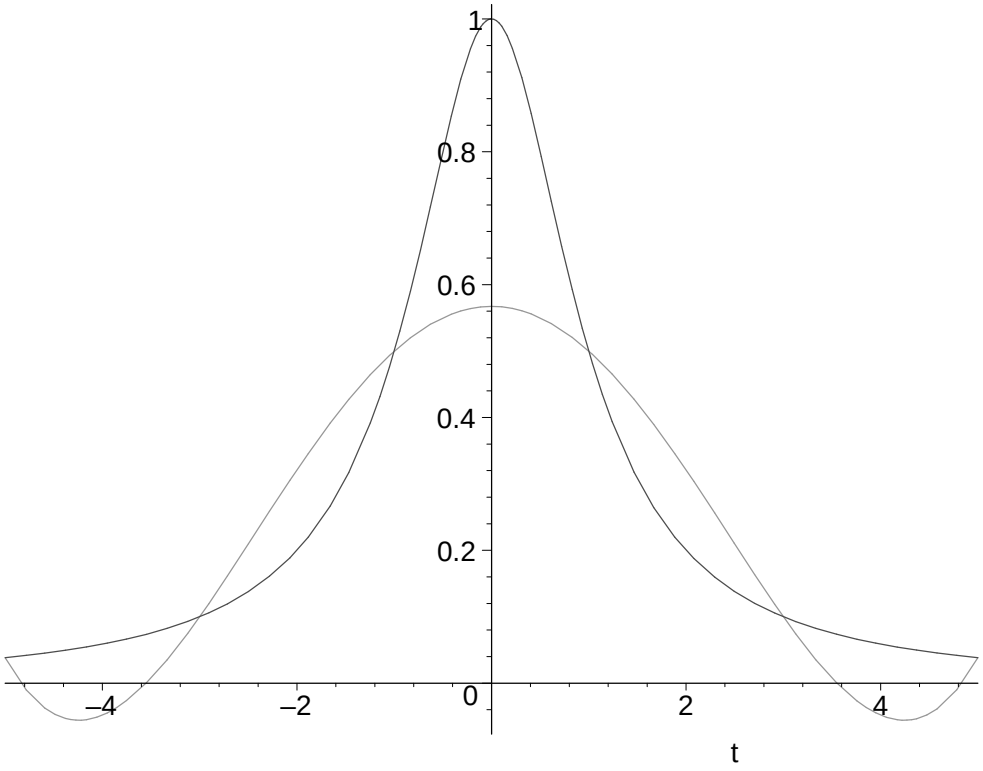


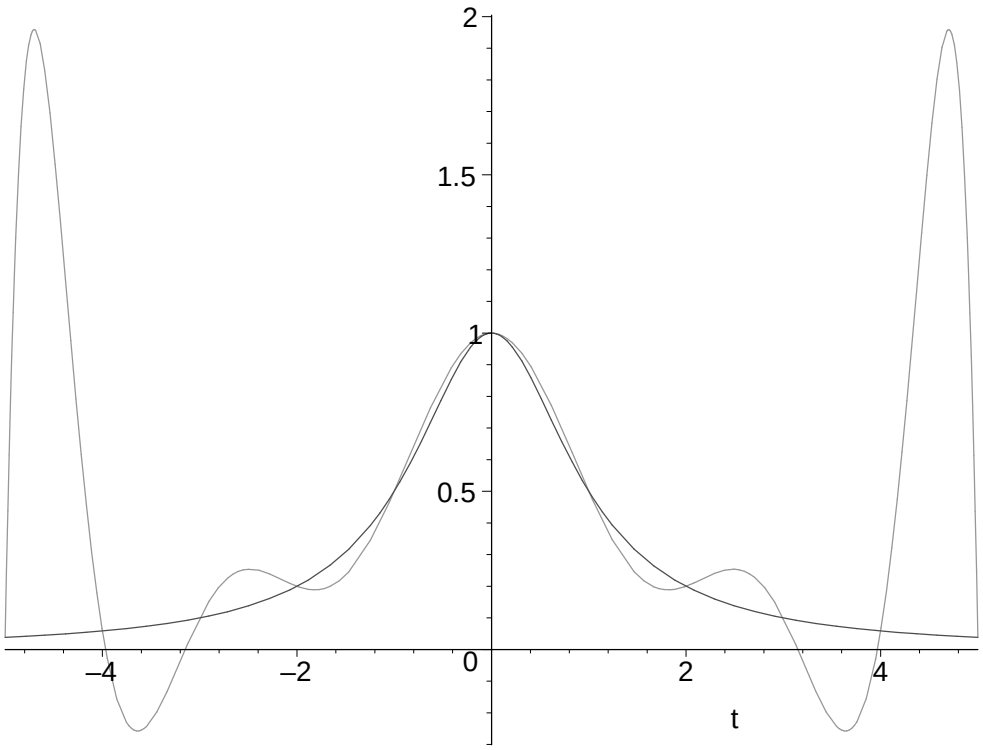
Avec 5 points,le choix des points est important mais avec 10 tout marche bien.

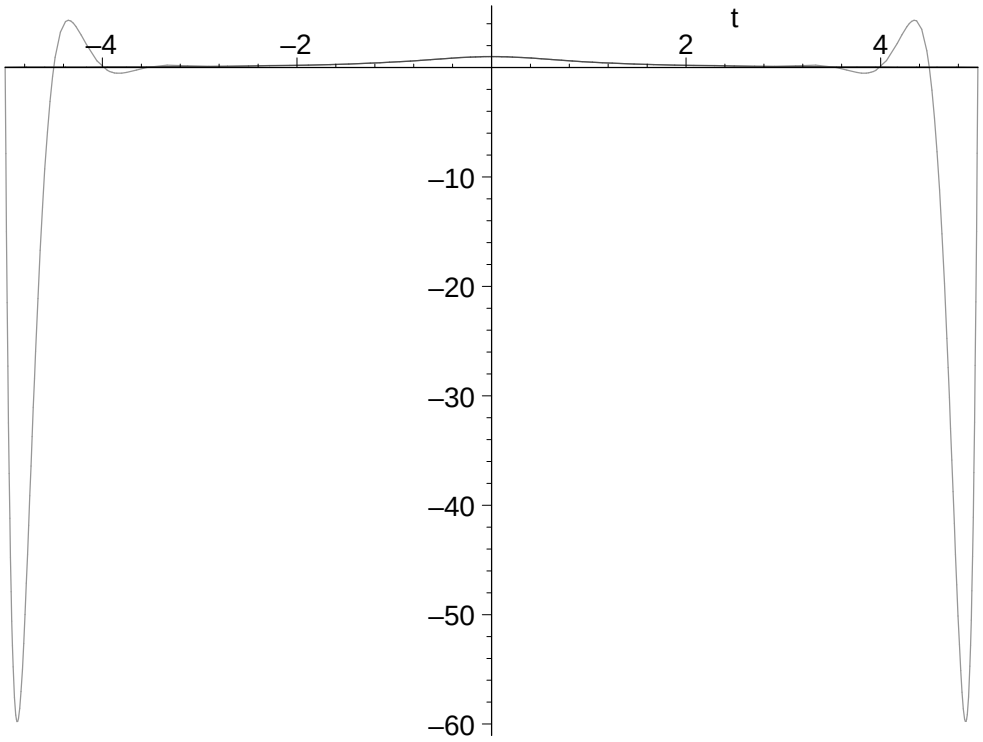
```
> g:=x->1/(1+x^2);
```

$$g := x \to \frac{1}{1+x^2}$$

```
> interequid(g,5,-5,5);interequid(g,10,-5,5
  );interequid(g,20,-5,5);
```


1
0.8
0.6
0.4
0.2
0
–4
–2
2
4
t


2
1.5
1
0.5
0
–4
–2
2
4
t



On decouvre le phenomene de Runge : on a beau augmente le nombre de points, il y a (de plus en plus) d'effets de bord. En tout cas, il est clair que le polynome interpolateur ne converge pas au sens de la norme infinie vers g sur [-5,5]. Calculons l'ecart et verifions la divergence.

```
> for n to 20 do
  test_convergence(g,n,-5,5);od;

2 points ecart :.961538
3 points ecart :.646229
4 points ecart :.707014
5 points ecart :.438357
6 points ecart :.432692
7 points ecart :.616948
8 points ecart :.247359
9 points ecart :1.045177
```

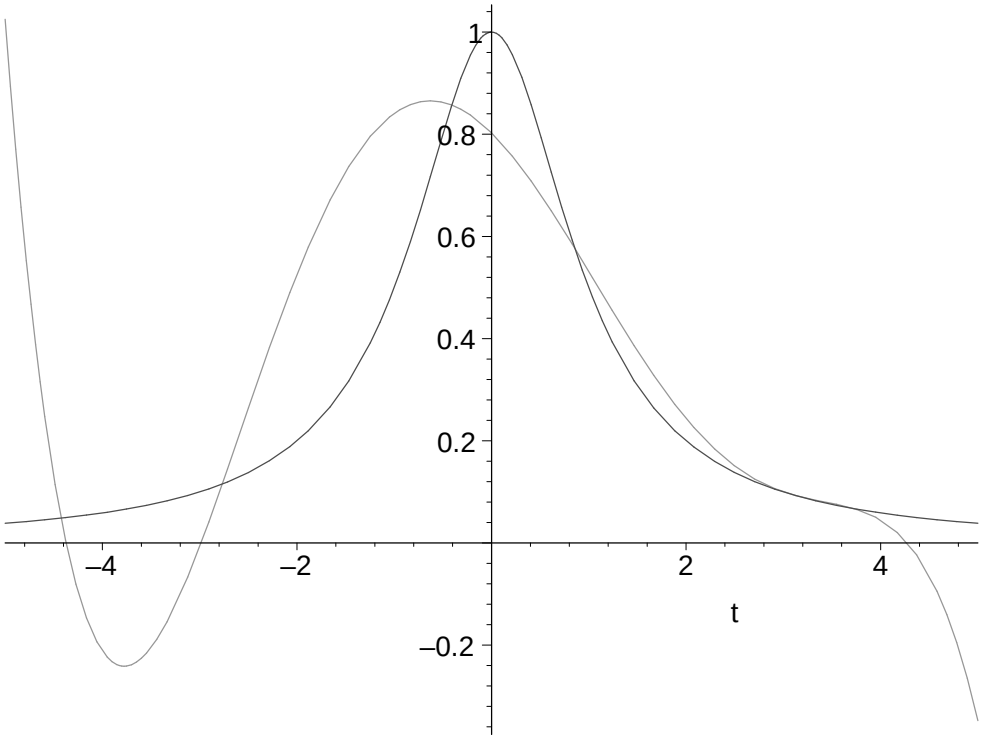
```
10 points ecart :.300298
11 points ecart :1.915659
12 points ecart :.556775
13 points ecart :3.663395
14 points ecart :1.070105
15 points ecart :7.194881
16 points ecart :2.107562
17 points ecart :14.39386
18 points ecart :4.224295
19 points ecart :29.1906
20 points ecart :8.579111
21 points ecart :59.822
```

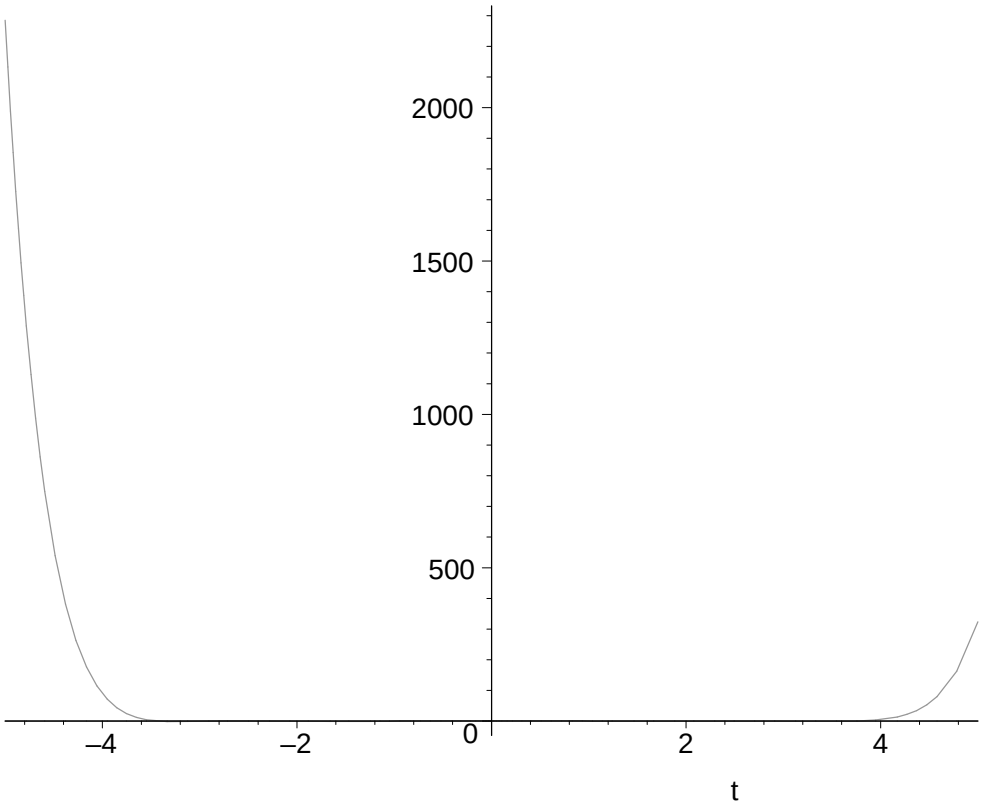
C’est clair!

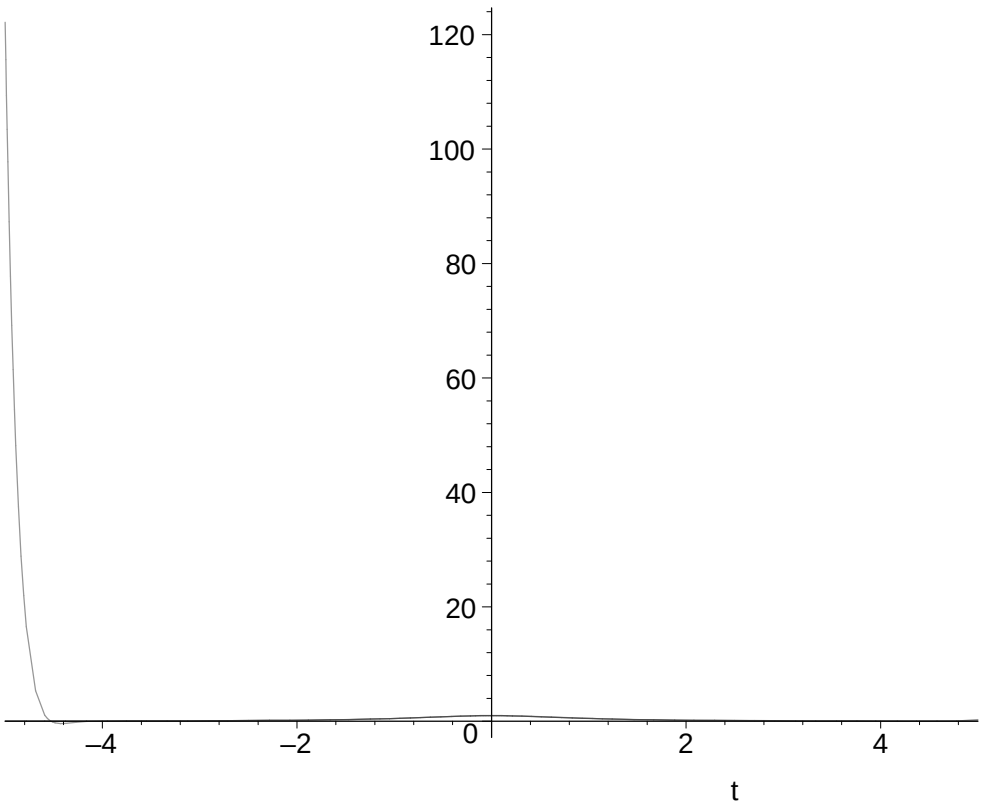
```
>
```

Essayons avec des points aleatoires.

```
> X:=subalea(5,-5,5):interquelc(g,X,-5,5);
  X:=subalea(10,-5,5):interquelc(g,X,-5,5);
  X:=subalea(20,-5,5):interquelc(g,X,-5,5);
```


1
0.8
0.6
0.4
0.2
–0.2
–4
–2
2
4
t


2000
1500
1000
500
0
–4
–2
2
4
t



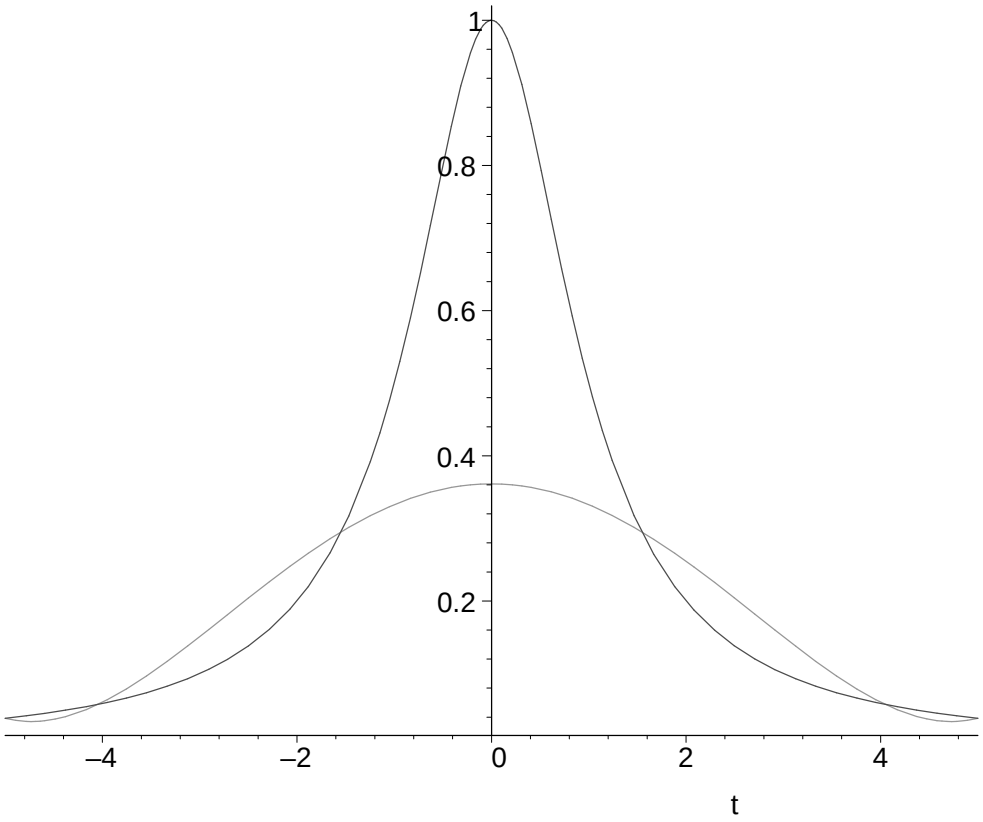
Ca ne marche pas mieux.

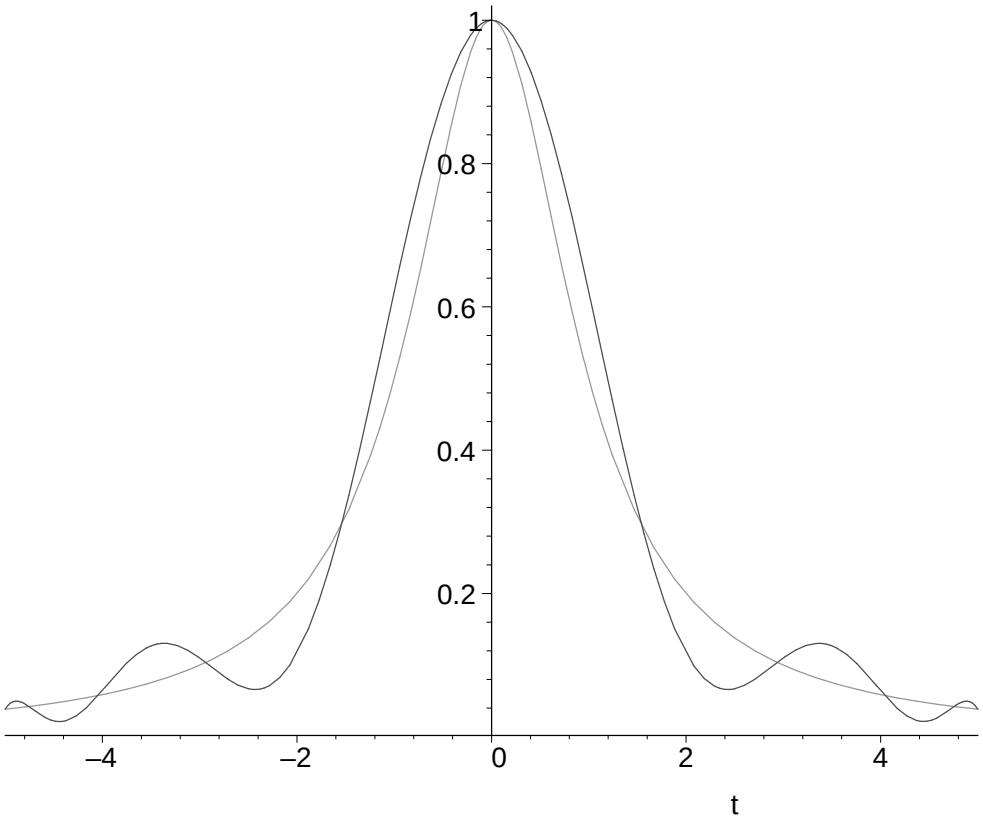
Essayons les points suggeres.

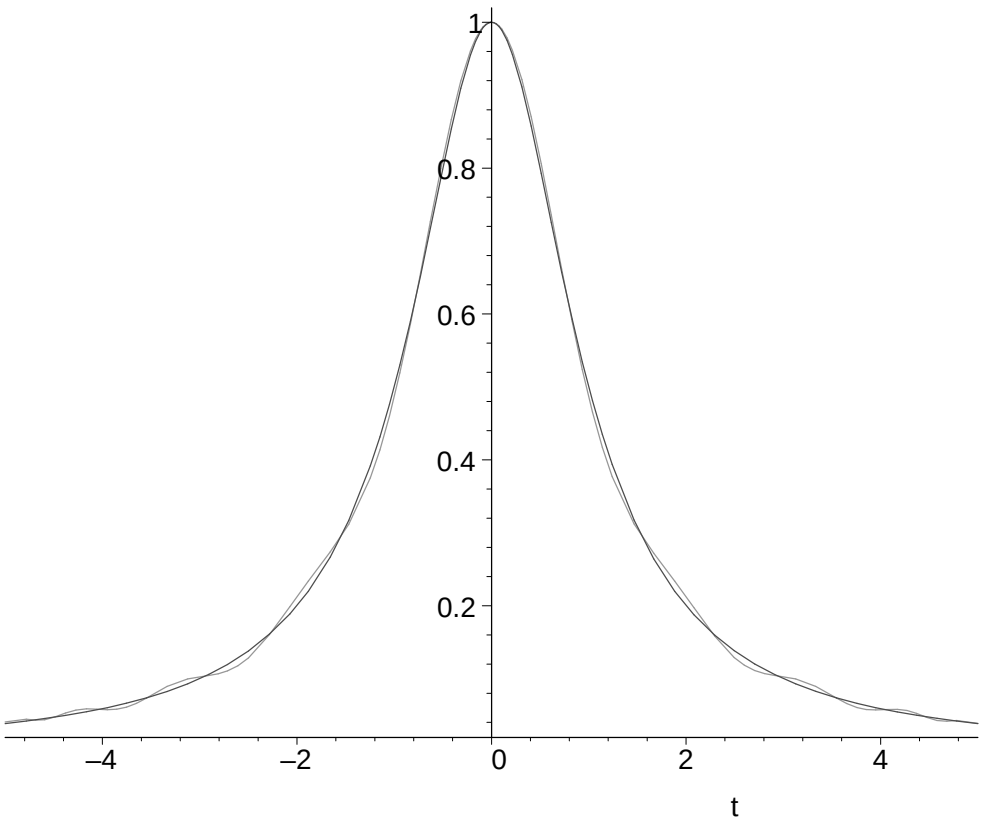
```
> spec:=n->[seq(evalf(5*cos(j*Pi/n)),j=0..n
  )];
```

$$spec := n \rightarrow \left[\mathrm{seq}\left(\mathrm{evalf}\left(5 \cos\left(\frac{j \pi}{n} \right) \right), j = 0 \,..\, n \right) \right]$$

```
> interquelc(g,spec(5),-5,5);
  interquelc(g,spec(10),-5,5);
  interquelc(g,spec(20),-5,5);
```


1
0.8
0.6
0.4
0.2
–4
–2
0
2
4
t


1
0.8
0.6
0.4
0.2
–4
–2
0
2
4
t



C’est beaucoup mieux!

>

Exercice 5 : interpolation de Tchebycheff

Il faut d’abord creer une subdvision qui calcule les abcisses de Tchebycheff.

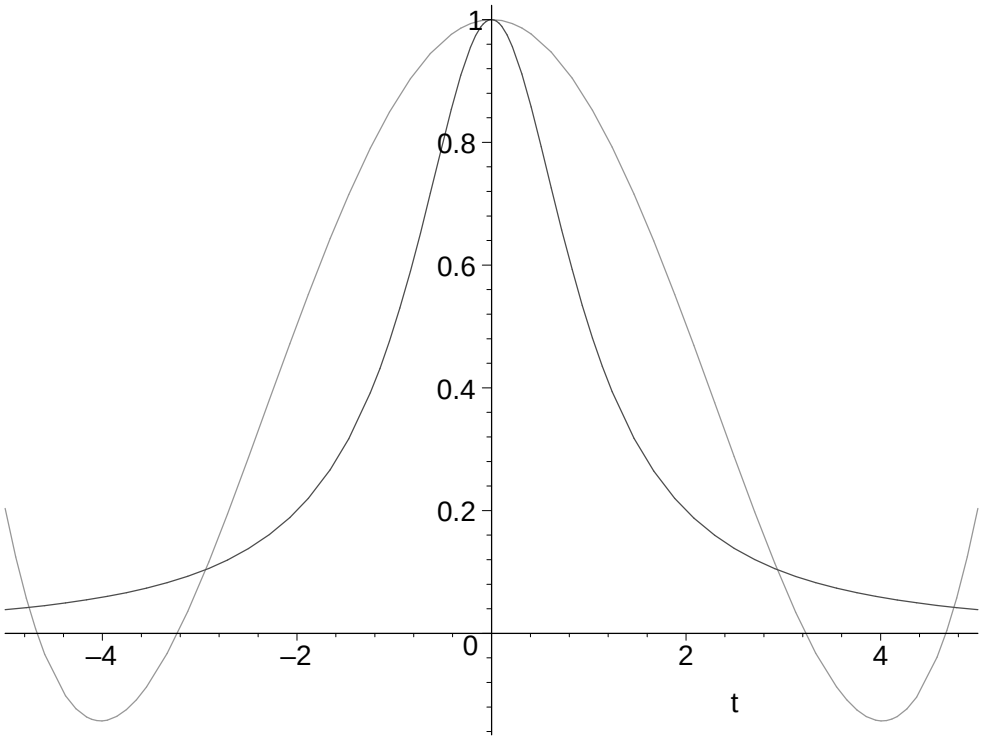
```
> subtche:=(n,a,b)->[seq(evalf((a+b)/2+cos(
  (2*k+1)*Pi/(2*n))*(a-b)/2),k=0..n-1)];
```

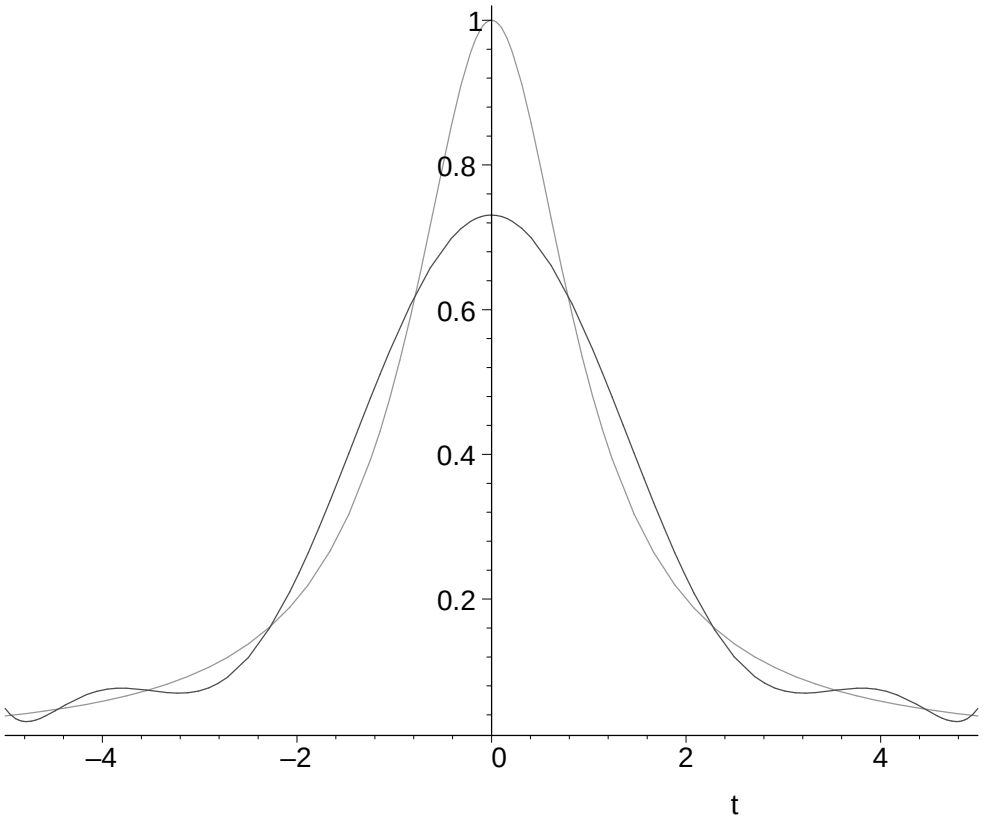
$$subtche := (n, a, b) \rightarrow \left[\mathrm{seq}\left(\mathrm{evalf}\left(\frac{1}{2} a + \frac{1}{2} b + \frac{1}{2} \cos\left(\frac{1}{2} \frac{(2k+1)\pi}{n} \right) (a-b) \right), k = 0 \,..\, n-1 \right) \right]$$

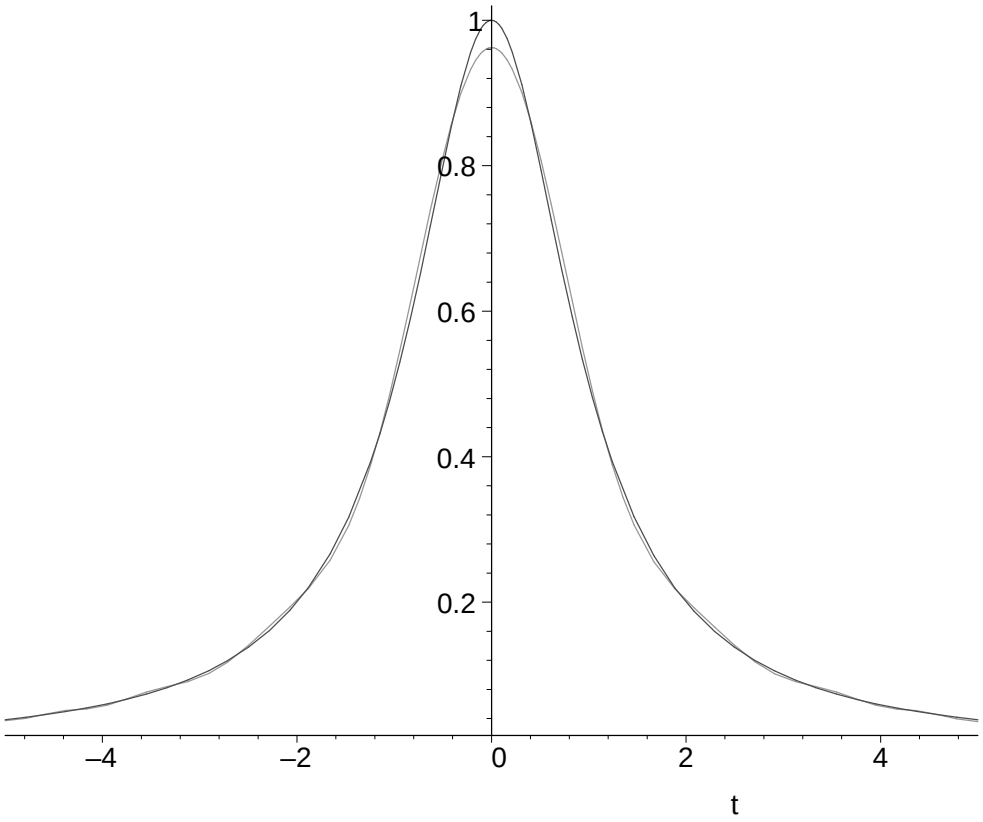
```
> intertche:=proc(f,n,a,b)
  local X,Y,P;
  X:=subtche(n,a,b);
  Y:=map(f,X);
  P:=interp(X,Y,t);
  plot({f(t),P},t=a..b);
  end:
```

Essayons.

```
> g:=x->1/(1+x^2):
  intertche(g,5,-5,5);
  intertche(g,10,-5,5);
  intertche(g,20,-5,5);
```



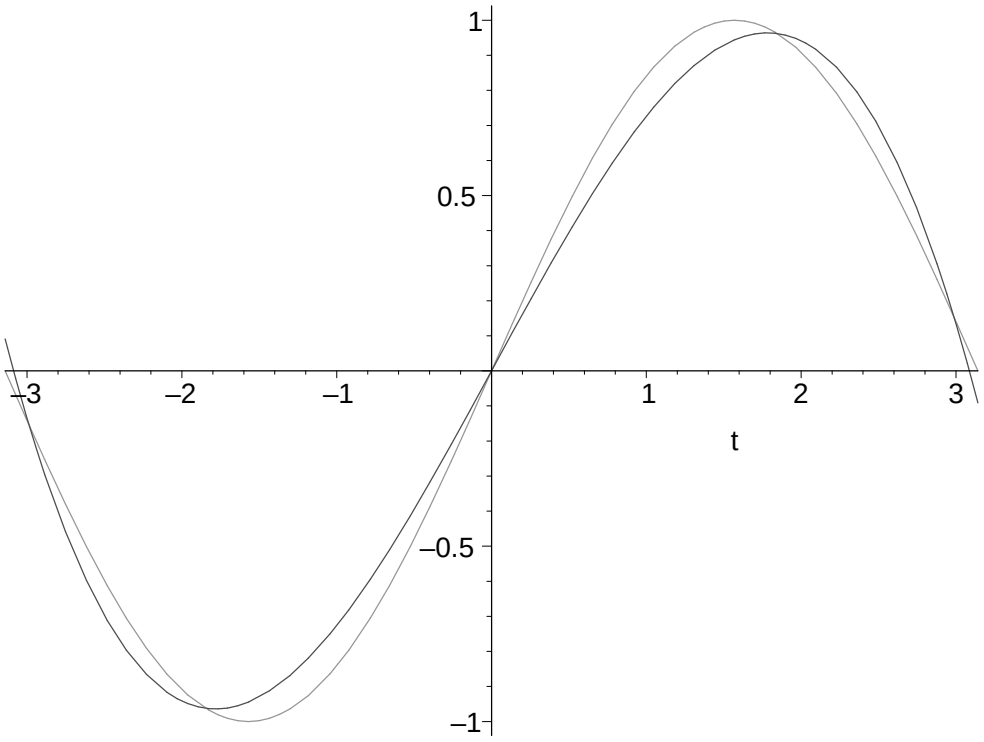

1
0.8
0.6
0.4
0.2
–4
–2
0
2
4
t

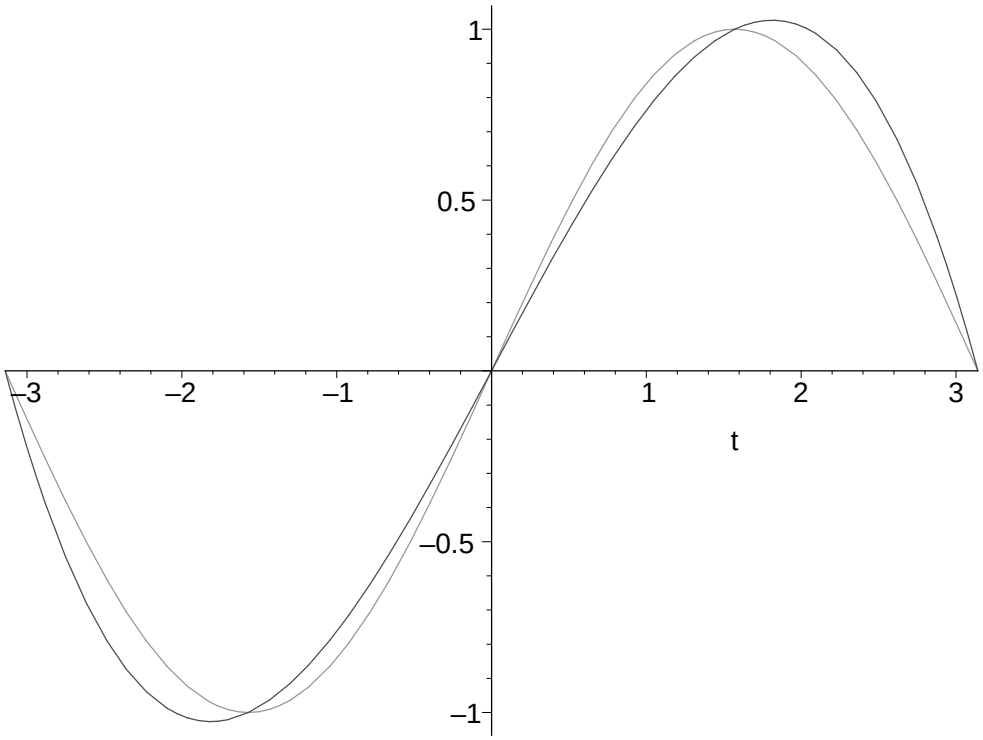


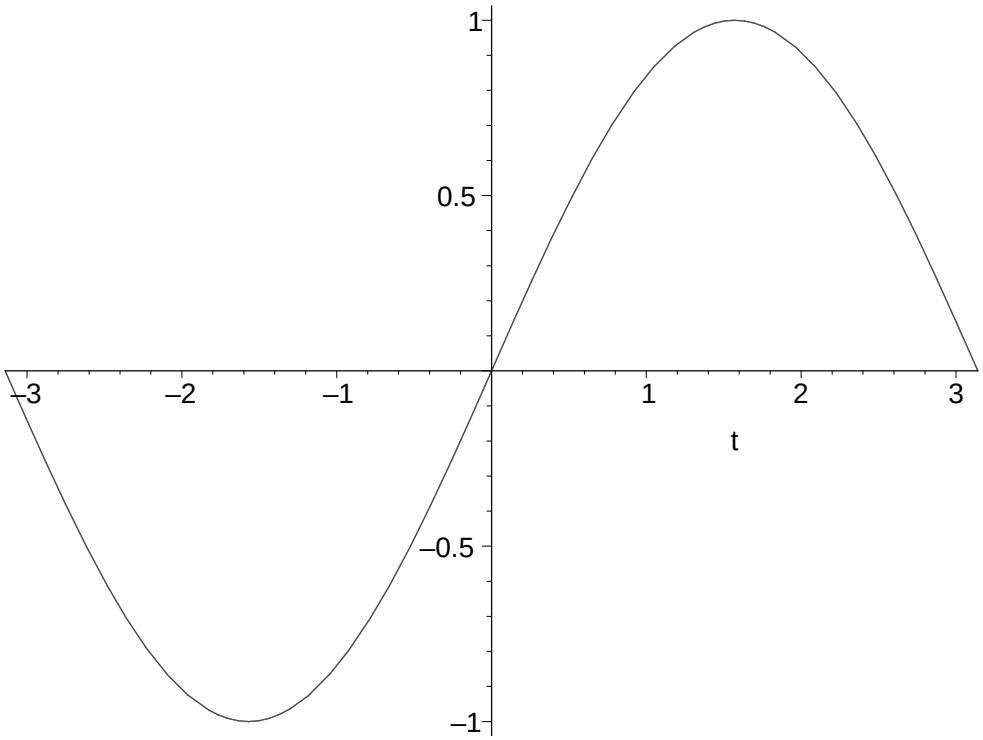
C’est quand meme beaucoup mieux que interquid et interquelc

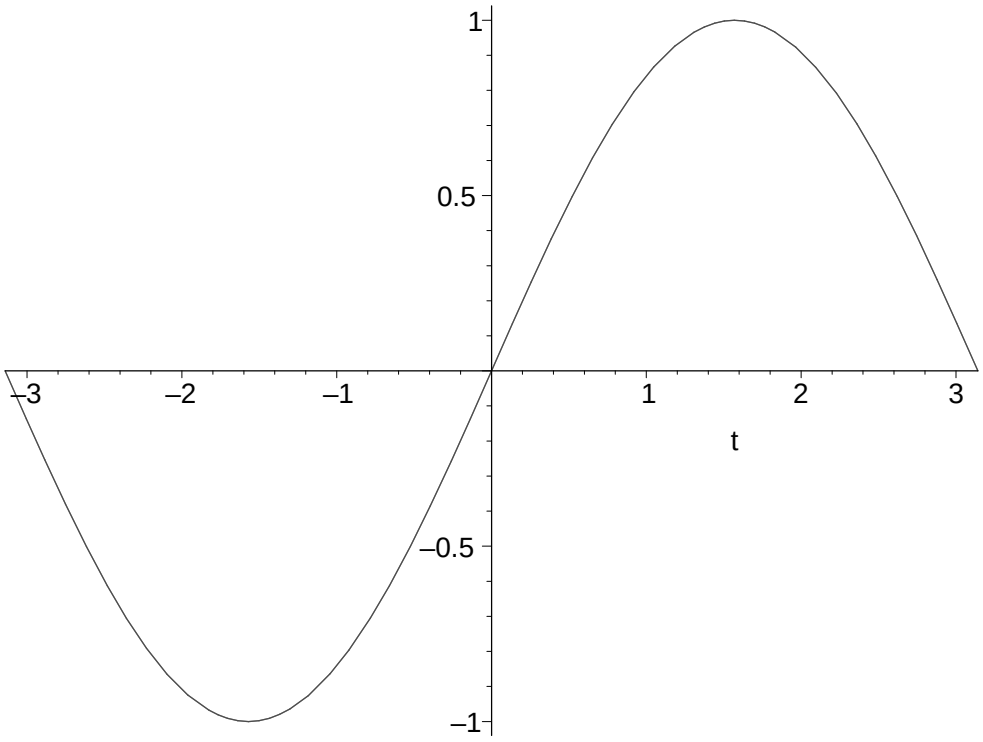
Essayons avec la fonction sinus.

```
> intertche(sin,5,-Pi,Pi);interequid(sin,4,
  -Pi,Pi);
  intertche(sin,10,-Pi,Pi);interequid(sin,9
  ,-Pi,Pi);
```


1
0.5
–0.5
–1
–3
–2
–1
1
2
3
t


1
0.5
–3
–2
–1
1
2
3
t
–0.5
–1


1
0.5
–0.5
–1
–3
–2
–1
1
2
3
t



Dans ce cas, la difference est moins marquee. La fonction sinus se laisse bien approcher (sur [-Pi,Pi]) par des polynoemes.

>

Supplement : trace des polynomes de Tchebycheff

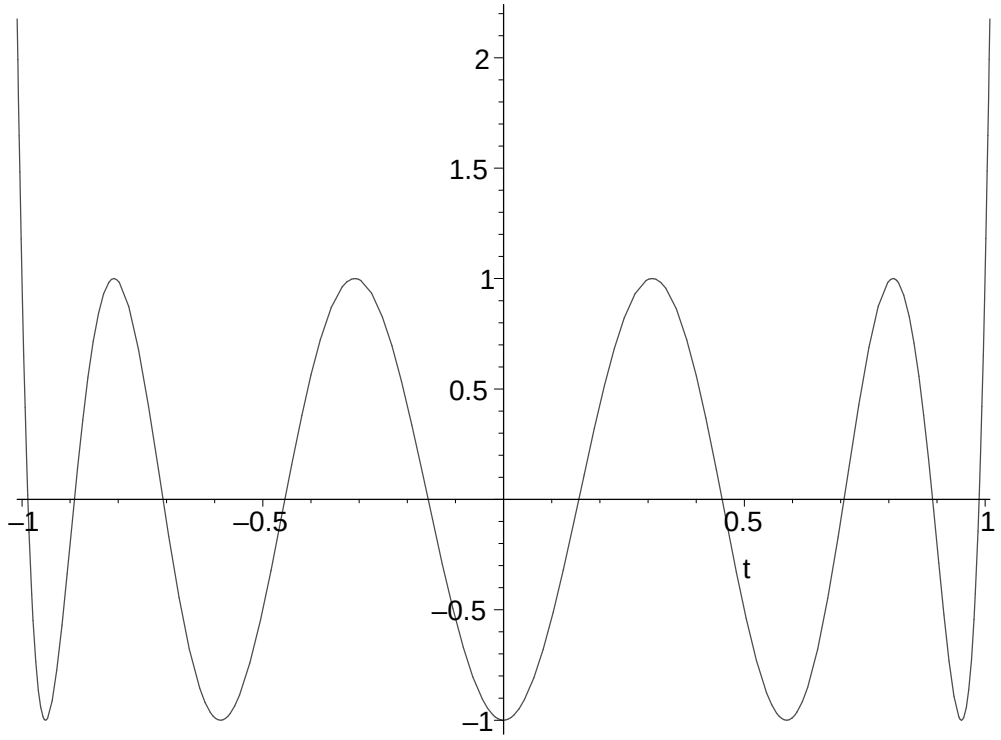
La librairie qui contient les polynomes orthogonaux est orthopoly.

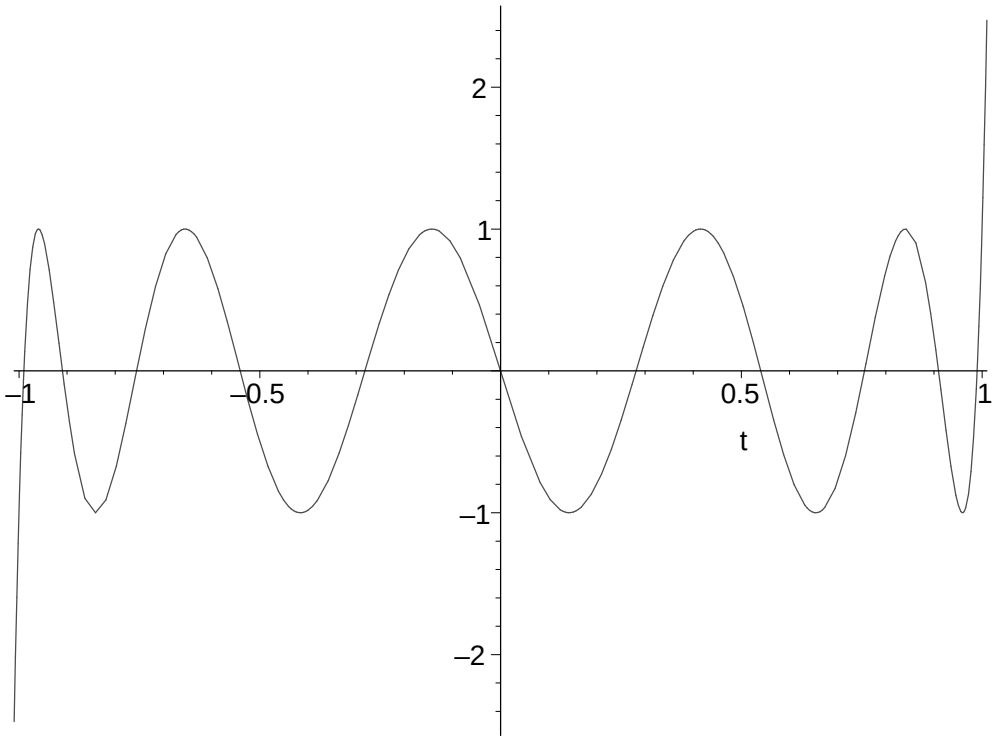
```
> with(orthopoly):T(11,t);
Warning, the name P has been redefined
```

$$1024\,t^{11} - 2816\,t^{9} + 2816\,t^{7} - 1232\,t^{5} + 220\,t^{3} - 11\,t$$

>

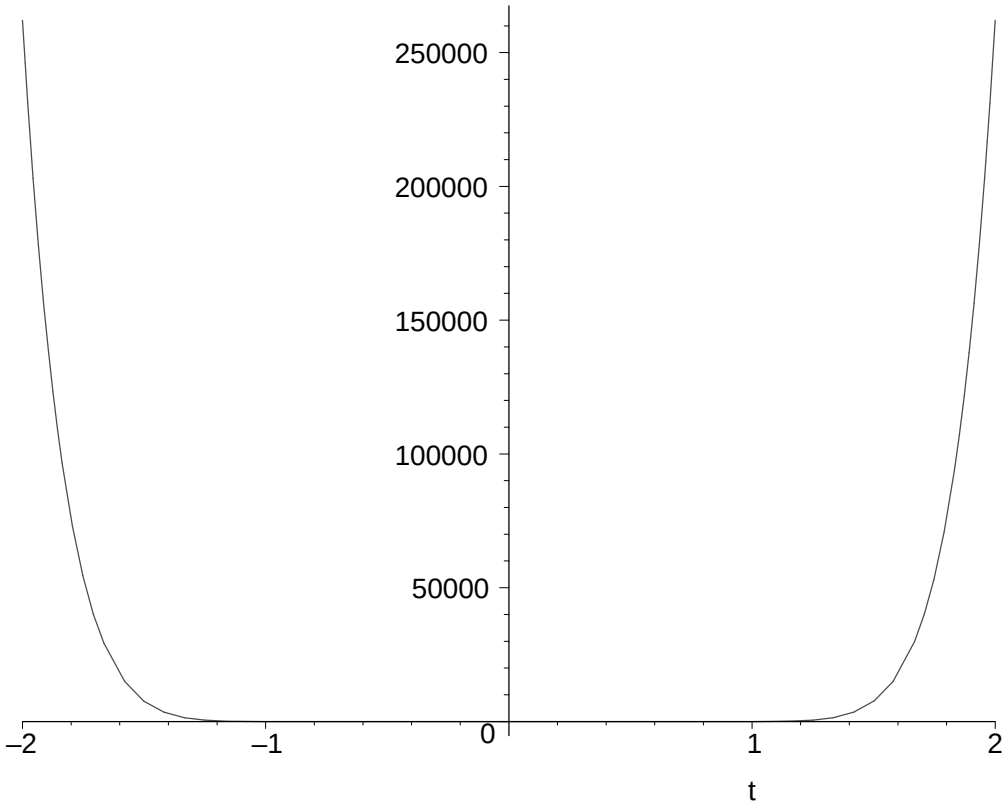
```
> plot(T(10,t),t=-1.01..1.01);
  plot(T(11,t),t=-1.01..1.01);
```

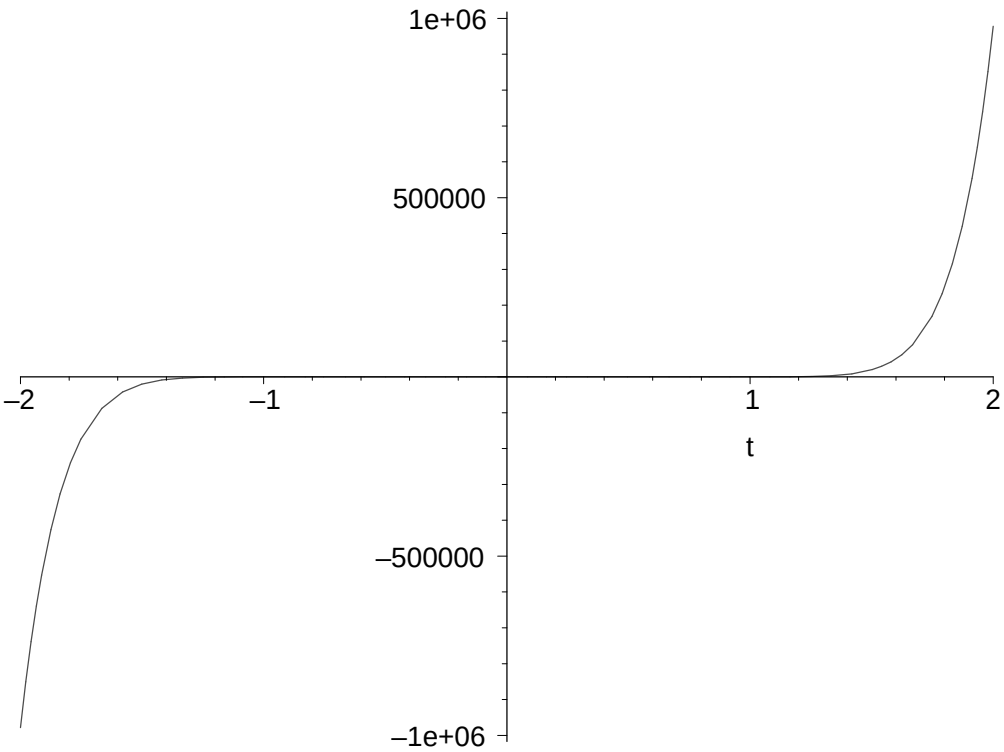




Les polynomes de Tchebycheff sont extra-plats sur [-1,1].

```
> plot(T(10,t),t=-2..2);
  plot(T(11,t),t=-2..2);
```


250000
200000
150000
100000
50000
–2
–1
0
1
2
t



>

FIN