IV - Utilisation de quelques fonctions intégrées

1. La fonction chebyshev

Mis à part les fonctions `T` et `U` du package `orthopoly`, Maple possède un certain nombre de procédures en rapport direct avec les polynômes de Chebyshev. Il s'agit de la fonction `chebyshev`, et de quelques autres qui appartiennent au package `numapprox`.

Rappelons le produit scalaire *intégral* utilisé pour développer des fonctions en série de Chebyshev sur $[-1,1]$. f et g sont supposées être des expressions de la variable x :

```
> restart: scal:=(f,g)->2/Pi*int(f*g/sqrt(1-x^2),x=-1..1);
```

$$scal := (f,\, g) \to 2\,\frac{\displaystyle\int_{-1}^{1} \frac{f\,g}{\sqrt{1-x^2}}\,dx}{\pi}$$

Enfin la fonction `SN` donne la somme partielle d'indice N dans le développement.

Pour calculer les intégrales, nous avons utilisé le nom complet `orthopoly[T]`, ce qui est un moyen d'utiliser la fonction `T` sans la charger en mémoire. C'est utile pour que le résultat apparaisse comme une combinaison linéaire des polynômes $T(0,x)=1$, $T(1,x)$, $T(2,x)$, etc. :

```
> SN:=(f,N)->1/2*scal(f,1)+Sum(scal(f,orthopoly[T](k,x))*T(k,x),k=1..N) ;
```

$$SN := (f,\, N) \to \frac{1}{2}\,\mathrm{scal}(f,\, 1) + \sum_{k=1}^{N} \mathrm{scal}(f,\, orthopoly_T(k,\, x))\,\mathrm{T}(k,\, x)$$

Voici par exemple la forme symbolique de la somme partielle d'indice 2 dans le développement de l'expression $f(x)=\ln(2-x)$:

```
> s2:=SN(ln(2-x),2);
```

$$s2 := \frac{\displaystyle\int_{-1}^{1} \frac{\ln(2-x)}{\sqrt{1-x^2}}\,dx}{\pi} + \sum_{k=1}^{2} 2\,\frac{\displaystyle\int_{-1}^{1} \frac{\ln(2-x)\; orthopoly_T(k,\, x)}{\sqrt{1-x^2}}\,dx\,\mathrm{T}(k,\, x)}{\pi}$$

On voit que Maple ne sait pas calculer les valeurs exactes de ces intégrales :

```
> value(s2);
```

$$\frac{\displaystyle\int_{-1}^{1} \frac{\ln(2-x)}{\sqrt{1-x^2}}\,dx}{\pi} + 2\,\frac{\displaystyle\int_{-1}^{1} \frac{\ln(2-x)\,x}{\sqrt{1-x^2}}\,dx\,\mathrm{T}(1,x)}{\pi} + 2\,\frac{\displaystyle\int_{-1}^{1} \frac{\ln(2-x)\,(2\,x^2-1)}{\sqrt{1-x^2}}\,dx\,\mathrm{T}(2,x)}{\pi}$$

On doit donc se contenter d'une valeur approchée :

```
> evalf(s2);
```

$$.6238107163 - .5358983849\,\mathrm{T}(1,x) - .07179676972\mathrm{T}(2,x)$$

Dans la foulée, voici les sommes partielles d'indice 3 puis 4 du développement de $\ln(2-x)$.
Chaque développement s'obtient à partir du précédent en ajoutant un terme :

```
> evalf(SN(ln(2-x),3));
  evalf(SN(ln(2-x),4));
```

$$.6238107163 - .5358983849\,\mathrm{T}(1,x) - .07179676972\,\mathrm{T}(2,x) - .01282525764\,\mathrm{T}(3,x)$$

$$.6238107163 - .5358983849\mathrm{T}(1,x) - .07179676972\mathrm{T}(2,x) - .01282525764\mathrm{T}(3,x) - .002577388071\mathrm{T}(4,x)$$

Nous allons maintenant utiliser la fonction intégrée `chebyshev`.

Cette fonction donne elle aussi le développement d'une expression en série de polynômes de Chebyshev, mais elle est plus rapide que notre fonction `SN`.

Autre différence : le développement obtenu par la fonction intégrée `chebyshev` est limité par une précision définie par l'utilisateur et non pas, comme notre fonction `SN`, par l'ordre maximum k des polynômes T_k.

Voici par exemple comment retrouver les résultats précédents :

```
> chebyshev(ln(2-x),x,10^(-1));
```

$$.6238107165\,\mathrm{T}(0,\,x) - .5358983849\,\mathrm{T}(1,\,x) - .07179676972\,\mathrm{T}(2,\,x)$$

```
> chebyshev(ln(2-x),x,5*10^(-2));
```

$$.6238107165\,\mathrm{T}(0,\,x) - .5358983849\,\mathrm{T}(1,\,x) - .07179676972\,\mathrm{T}(2,\,x) - .01282525764\,\mathrm{T}(3,\,x)$$

```
> chebyshev(ln(2-x),x,10^(-2));
```

$$.6238107165\,\mathrm{T}(0,\,x) - .5358983849\,\mathrm{T}(1,\,x) - .07179676972\,\mathrm{T}(2,\,x)$$
$$-.01282525764\,\mathrm{T}(3,\,x) - .002577388072\,\mathrm{T}(4,\,x)$$

Si on charge en mémoire la fonction `orthopoly[T]`, alors la procédure intégrée `chebyshev` renvoie toujours un résultat exprimé suivant les polynômes T_k, mais il suffit d'une simple évaluation pour obtenir le résultat suivant les puissances de x :

```
> with(orthopoly,T) :
  chebyshev(ln(2-x),x,10^(-1));
  eval(chebyshev(ln(2-x),x,10^(-1)));
```

$$.6238107165\,\mathrm{T}(0,\,x) - .5358983849\,\mathrm{T}(1,\,x) - .07179676972\,\mathrm{T}(2,\,x)$$

$$.6956074862 - .5358983849\,x - .1435935394\,x^2$$

Dans la syntaxe de la fonction intégrée `chebyshev`, le troisième argument indique une précision *eps*. Il doit être interprété de la manière suivante :

Le developpement $f(x) = a_0T_0(x) + a_1T_1(x) + \cdots + a_kT_k(x) + \cdots$ est stoppé lorsque le module du coefficient a_k devient inférieur au produit par *eps* du plus grand de tous les coefficients déja calculés.

Le plus souvent d'ailleurs les coefficients a_k forment une suite qui est rapidement décroissante en module. Comme chaque T_k est majoré par 1 en valeur absolue, on a ainsi une idée de la précision obtenue dans le développement sur tout l'intervalle $[-1, 1]$.

Si on n'indique pas explicitement une précision, Maple utilise $10^{-Digits}$, par défaut 10^{-10}.

Voici par exemple le développement de $\ln(2-x)$ à cette précision :

```
> s:=chebyshev(ln(2-x),x);
```

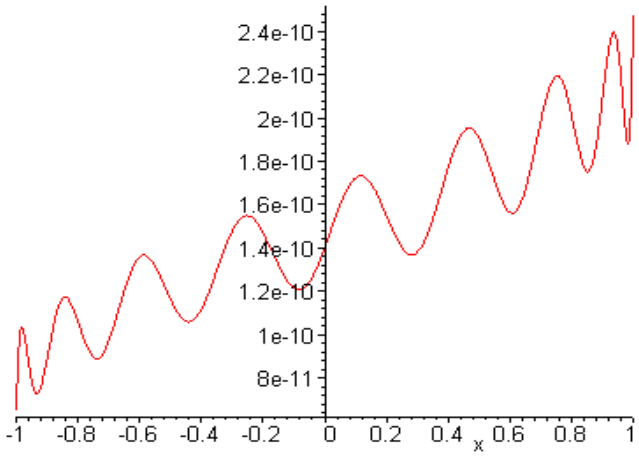
$$\begin{aligned} s := \; & .6238107165\,\mathrm{T}(0,\,x) - .5358983849\,\mathrm{T}(1,\,x) - .07179676972\,\mathrm{T}(2,\,x) \\ & -.01282525764\,\mathrm{T}(3,\,x) - .002577388072\,\mathrm{T}(4,\,x) - .0005524872418\,\mathrm{T}(5,\,x) \\ & -.0001233654252\,\mathrm{T}(6,\,x) - .00002833342806\,\mathrm{T}(7,\,x) \\ & -.6642929272\,10^{-5}\,\mathrm{T}(8,\,x) - .1582193363\,10^{-5}\,\mathrm{T}(9,\,x) \\ & -.3815526906\,10^{-6}\,\mathrm{T}(10,\,x) - .9294248695\,10^{-7}\,\mathrm{T}(11,\,x) \\ & -.2282854384\,10^{-7}\,\mathrm{T}(12,\,x) - .5646365550\,10^{-8}\,\mathrm{T}(13,\,x) \\ & -.1404894267\,10^{-8}\,\mathrm{T}(14,\,x) - .3514312059\,10^{-9}\,\mathrm{T}(15,\,x) \\ & -.8862094583\,10^{-10}\,\mathrm{T}(16,\,x) \end{aligned}$$

Si nous traçons la différence entre $f(x) = \ln(2-x)$ et ce développement $s(x)$, on obtient la courbe suivante.

Il apparait bien que la précision est de l'ordre de 10^{-10}, mais paradoxalement cette précision excellente sur tout l'intervalle implique que le calcul des différences $s(x) - \ln(2-x)$ est sujet à une incertitude relative importante.

Le tracé ci-dessous est donc inexact :

```
> plot(s-ln(2-x),x=-1..1);
```

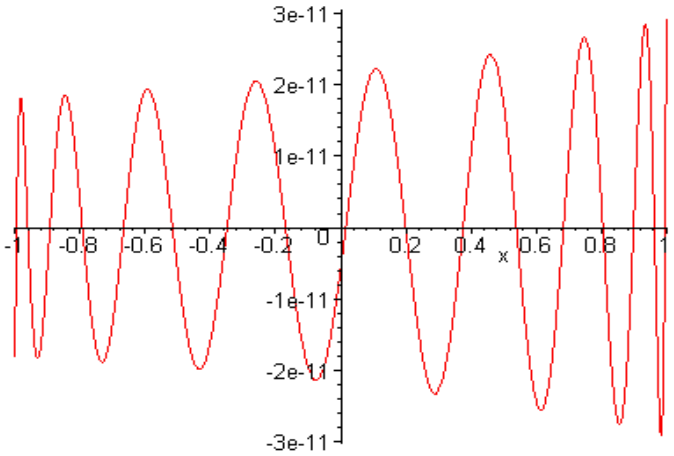


Pour obtenir un tracé fiable, il faut augmenter la précision interne des calculs (ici on va passer à 15 décimales pour avoir de la marge), avant de demander à Maple le développement $s(x)$ de $\ln(2-x)$ à la précision 10^{-10}.

On trace ensuite $s(x) - \ln(2-x)$, toujours avec une précision interne de 15 décimales : ce n'est qu'après le tracé qu'on peut revenir à la valeur par défaut `Digits`= 10.

Voici donc le tracé, correct cette fois, de la différence entre $f(x) = \ln(2-x)$ et son développement en série de polynômes de Chebyshev à la précision 10^{-10}.

```
> Digits:=15: s:=chebyshev(ln(2-x),x,10^(-10)) :
  plot(s-ln(2-x),x=-1..1); Digits:=10 :
```



Nous avons utilisé ici un exemple qui n'est pas tout à fait innocent puisqu'on démontre que le développement en série de polynômes de Chebyshev de $f(x) = \ln(2-x)$ répond à la formule suivante :

```
> T:='T' : SN:=N->ln(1+sqrt(3)/2)-2*Sum((2-sqrt(3))^n/n*T(n,x),n=1..N) :
  ln(2-x)=SN(infinity);
```

$$\ln(2-x) = \ln(1+\frac{1}{2}\sqrt{3}) - 2\sum_{n=1}^{\infty} \frac{(2-\sqrt{3})^n \, \mathrm{T}(n,\, x)}{n}$$

Voici par exemple le développement exact de $\ln(2-x)$ à l'ordre 4...

```
> s4:=value(SN(4));
```

$$s4 := \ln(1+\frac{1}{2}\sqrt{3}) - 2\,(2-\sqrt{3})\,\mathrm{T}(1,\, x) - (2-\sqrt{3})^2\,\mathrm{T}(2,\, x) - \frac{2}{3}\,(2-\sqrt{3})^3\,\mathrm{T}(3,\, x) - \frac{1}{2}\,(2-\sqrt{3})^4\,\mathrm{T}(4,\, x)$$

...Et la forme numérique de ce développement. On retrouve un résultat déjà obtenu :

```
> evalf(s4);
```

$$.6238107165 - .535898384\mathrm{T}(1,\, x) - .07179676949\mathrm{T}(2, x) - .01282525758\mathrm{T}(3, x) - .002577388055\mathrm{T}(4, x)$$

La fonction suivante permet de calculer les coefficients $a_0, a_1, a_2, \ldots$ de ce développement :

```
> a:=n->`if`(n=0,ln(1+sqrt(3)/2),-2*(2-sqrt(3))^n/n);
```

$$a := n \to \text{`if`}\left(n = 0,\, \ln(1+\frac{1}{2}\sqrt{3}),\, -2\,\frac{(2-\sqrt{3})^n}{n}\right)$$

Le fait qu'on puisse calculer ces coefficients, visiblement décroissants en module, permet de confirmer la signification du troisième paramètre de la fonction `chebyshev`.

Supposons par exemple qu'on veuille écrire le développement de $\ln(2-x)$ à la précision 10^{-6}. Combien de termes la procédure `chebyshev` va-t-elle écrire ?

Pour le savoir, on forme la quotient des valeurs absolues de a_n et de a_0 :

```
> q:=n->abs(evalf(a(n)/a(0)));
```

$$q := n \to \left|\text{evalf}(\frac{\text{a}(n)}{\text{a}(0)})\right|$$

On constate que le premier quotient inférieur à 10^{-6} est obtenu pour $n = 10$.

Cela signifie que la fonction `chebyshev`, pour cette précision, va afficher des composantes sur $T(10, x)$ au maximum :

```
> q(9), q(10);
```

$$.2536335620\,10^{-5},\ .6116481720\,10^{-6}$$

En voici la confirmation :

```
> chebyshev(ln(2-x),x,10^(-6));
```

$$\begin{aligned}
&.6238107165\,\text{T}(0,\,x) - .5358983849\,\text{T}(1,\,x) - .07179676972\,\text{T}(2,\,x)\\
&-.01282525764\,\text{T}(3,\,x) - .002577388072\,\text{T}(4,\,x) - .0005524872418\,\text{T}(5,\,x)\\
&-.0001233654252\,\text{T}(6,\,x) - .00002833342806\,\text{T}(7,\,x)\\
&-.6642929272\,10^{-5}\,\text{T}(8,\,x) - .1582193363\,10^{-5}\,\text{T}(9,\,x)\\
&-.3815526906\,10^{-6}\,\text{T}(10,\,x)
\end{aligned}$$

2. Développement sur tout intervalle

Dans cette étude, on écrit surtout des développements en polynômes de Chebyshev sur $[-1, 1]$, que ce soit avec le produit scalaire *intégral* ou avec le produit scalaire *discret*.

Si $t \mapsto f(t)$ est définie sur le segment $[a, b]$, le changement de variable $x = \frac{2t-a-b}{b-a}$, s'inversant en $t = \frac{a+b}{2} + x\,\frac{b-a}{2}$, permet de lui associer l'application $x \mapsto g(x)$ définie sur $[-1, 1]$.

Il suffit alors de développer $g(x)$ sur ce segment puis de revenir à t dans le résultat.

On obtient ainsi le développement de f sur $[a, b]$.

La fonction `chebyshev` permet d'effectuer directement un développement sur un segment quelconque. Il suffit de le préciser dans le deuxième argument.

Voici par exemple le développement de $f(t) = \exp t$ sur $[1, 4]$, à la précision $\frac{1}{10}$:

```
> chebyshev(exp(t),t=1..4,10^(-1));
```

$$\begin{aligned}
&20.06119532\,\text{T}(0,\,\frac{2}{3}t - \frac{5}{3}) + 23.91829068\,\text{T}(1,\,\frac{2}{3}t - \frac{5}{3}) + 8.231336395\,\text{T}(2,\,\frac{2}{3}t - \frac{5}{3})\\
&+ 1.968060288\,\text{T}(3,\,\frac{2}{3}t - \frac{5}{3})
\end{aligned}$$

Dans l'exemple précédent, voici le calcul effectué par Maple :
Il effectue le changement de variable $t = \frac{3x+5}{2}$; quand x parcourt $[-1, 1]$, t parcourt $[1, 4]$.
On en déduit la nouvelle expression $g(x) = \exp\frac{3x+5}{2}$ qu'il faut développer sur $[-1, 1]$, le segment par défaut pour l'instruction `chebyshev` :

```
> chebyshev(exp((3*x+5)/2),x,10^(-1));
```

$$20.06119532\,\mathrm{T}(0,\,x) + 23.91829068\,\mathrm{T}(1,\,x) + 8.231336395\,\mathrm{T}(2,\,x) + 1.968060288\,\mathrm{T}(3,\,x)$$

Ensuite Maple fait le changement de variable inverse, c'est-à-dire $x = \frac{2t-5}{3}$.
On retouve bien le résultat déjà obtenu :

```
> subs(x=(2*t-5)/3,%);
```

$$20.06119532\,\mathrm{T}(0,\,\frac{2}{3}t-\frac{5}{3}) + 23.91829068\,\mathrm{T}(1,\,\frac{2}{3}t-\frac{5}{3}) + 8.231336395\,\mathrm{T}(2,\,\frac{2}{3}t-\frac{5}{3})$$
$$+1.968060288\,\mathrm{T}(3,\,\frac{2}{3}t-\frac{5}{3})$$

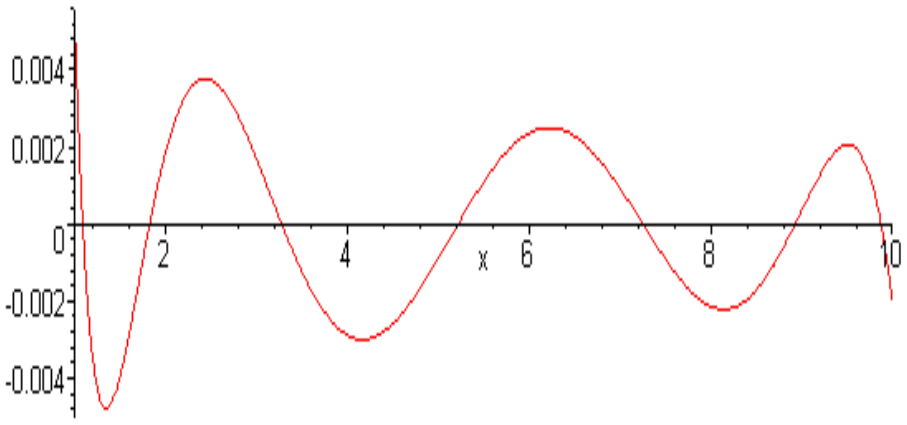
On peut donc effectuer des développements sur des intervalles quelconques, et ainsi obtenir des approximations polynômiales très précises. Voici par exemple celui de $\ln x$ sur l'intervalle $J = [1, 10]$, à la précision $\frac{1}{100}$.
On a directement écrit le résultat P suivant les puissances croissantes de x :

```
> with(orthopoly,T): J:=1..10 :
  P:=chebyshev(ln(x),x=J,10^(-2)): P:=collect(P,x);
```

$$P := -.00002524853208\,x^6 + .0009644274330\,x^5 - .01500875565\,x^4 + .1232488867\,x^3$$
$$-.5856153386\,x^2 + 1.780525649\,x - 1.298687933$$

Voici maintenant le tracé de la différence entre $\ln x$ et son développement P, sur l'intervalle J.
On voit que l'erreur maximum dans l'approximation de $\ln x$ par P est de l'ordre de $0,004$.
On constate comme toujours un certain nombre d'oscillations, assez régulières sans être exactement de même amplitude, et il y a 7 points d'égalité :

```
> plot(P-ln(x),x=J);
```



3. Comparaison avec "minimax"

Il est intéressant de comparer le résultat précédent avec celui de la fonction `minimax` du package `numapprox`, qui donne le polynôme ou plus généralement la fraction rationnelle de meilleure approximation au sens de la norme uniforme pour un degré m donné.

Voici par exemple quel est le polynôme de degré inférieur ou égal à 6 qui est le plus proche de $\ln x$ sur l'intervalle $J = [1, 10]$.

```
> Q:=numapprox[minimax](ln(x),x=J,6) :
  Q:=collect(Q,x);
```

$$Q := -.00003176214739\,x^6 + .001179499664\,x^5 - .01776993046\,x^4 + .1406064376\,x^3$$
$$-.6407833247\,x^2 + 1.862421149\,x - 1.341934002$$

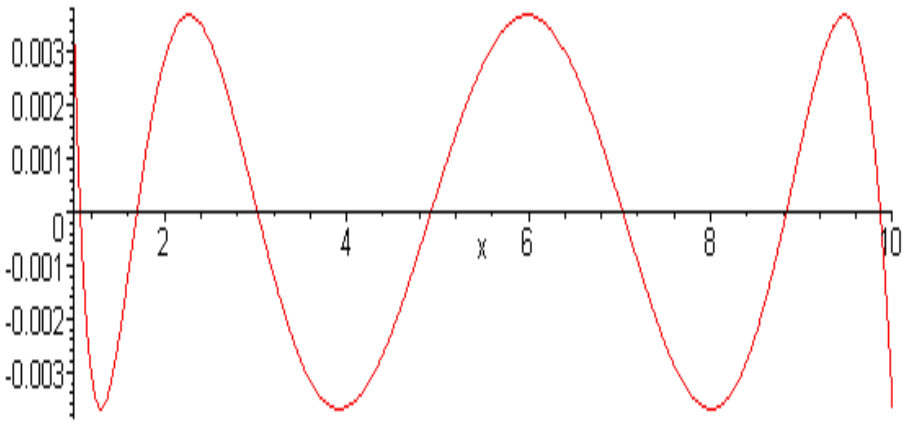
On voit que les coefficients des polynômes P et Q sont loin d'être les mêmes !

```
> P-Q;
```

$$.651361531\,10^{-5}\,x^6 - .0002150722310\,x^5 + .00276117481\,x^4 - .0173575509\,x^3$$
$$+.0551679861\,x^2 - .081895500\,x + .043246069$$

Voici maintenant la représentation de la différence entre $\ln x$ et son polynôme Q de meilleure approximation et de degré 6. On constate des oscillations qui sont cette fois-ci de même amplitude. Il en est toujours ainsi pour les approximations de type *minimax*.

```
> plot(Q-ln(x),x=J);
```



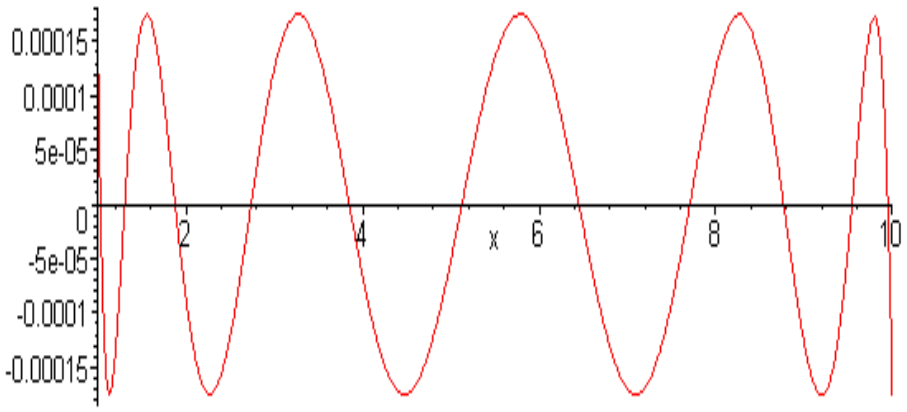
L'erreur maximum dans l'approximation précédente, minimale parmi tous les polynômes de degré inférieur ou égal à 6, est de l'ordre de 0.003 ce qui est à peine mieux que ce que nous avons trouvé avec le développement P de f à l'ordre 6 sur les polynômes de Chebyshev.

D'autre part, la recherche d'un polynôme de degré donné et qui réalise la meilleure approximation uniforme d'une fonction f sur un intervalle J exige beaucoup de calculs.

On s'en rend bien compte ci-dessous où on a calculé la meilleure approximation de $\ln x$ sur $J = [1, 10]$ par un polynôme de degré 10 puis le développement de $\ln x$ en polynômes de Chebyshev à la précision 10^{-4} sur le même intervalle.

Dans le premier cas, l'erreur maximum est de l'ordre de 0.00017, et le temps de calcul, qui est très variable d'un essai à l'autre, est de l'ordre de 14 secondes.

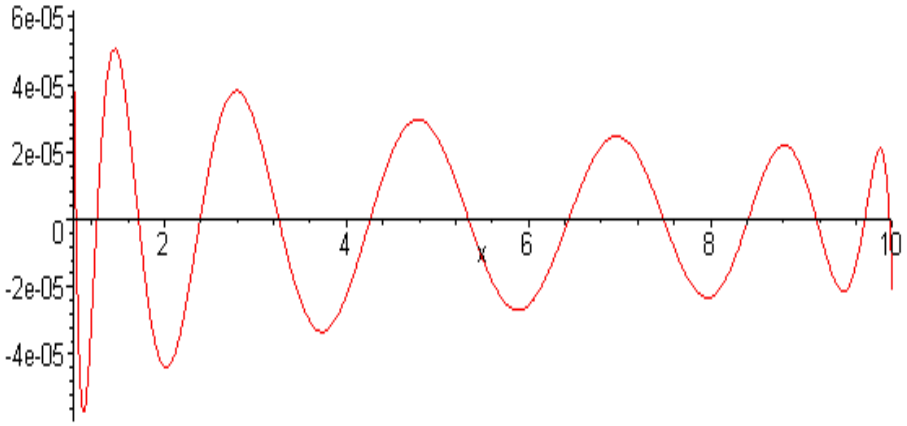
```
> restart: with(orthopoly) :
  J:=1..10: c:='temps de calcul=' :
  t:=time( ): P:=numapprox[minimax](ln(x),x=J,10): t:=time( )-t :
```



temps de calcul =, 13.898

Dans le second cas, et en moins de trois centièmes de secondes, on obtient une approximation polynômiale avec une erreur maximum de l'ordre de 0.00005. On vérifierait que ce deuxième polynôme est de degré 12.

```
> plot(%%-ln(x),x=J); c,t; t:=time( ) : Q:=chebyshev(ln(x),x=J,10^(-4)) :
  t:=time( )-t: plot(%%-ln(x),x=J); c,t;
```



temps de calcul =, .027

4. Les fonctions chebdeg, chebsort, chebmult

Le package `numapprox` contient trois fonctions dont le rôle est de traiter des expressions relatives aux polynômes de Chebyshev, comme celles qui sont renvoyées par la fonction `chebyshev`.

Pour les utiliser, il vaut mieux que la fonction `T` du package `orthopoly` ne soit pas chargée en mémoire ! On commence par charger le package `numapprox` :

```
> restart :with(numapprox);
```

[*chebdeg*, *chebmult*, *chebpade*, *chebsort*, *chebyshev*, *confracform*, *hermite_pade*, *hornerform*, *infnorm*, *laurent*, *minimax*, *pade*, *remez*, *taylor*]

La fonction `chebsort` trie les $T(k,x)$ suivant les valeurs croissantes du premier argument k.

C'est l'équivalent de la fonction `sort` qui ordonne les fonctions polynômes en x suivant les puissances de x. On trie ici un polynôme de degré 11 exprimé sur la base des T_k :

```
> r:=rand(12): P:=add(r()*T(r(),x),k=0..5);
  P:=chebsort(P);
```

$$P := 13\,\mathrm{T}(2,\,x) + 9\,\mathrm{T}(11,\,x) + 5\,\mathrm{T}(8,\,x) + \mathrm{T}(1,\,x) + 7\,\mathrm{T}(3,\,x)$$

$$P := \mathrm{T}(1,\,x) + 13\,\mathrm{T}(2,\,x) + 7\,\mathrm{T}(3,\,x) + 5\,\mathrm{T}(8,\,x) + 9\,\mathrm{T}(11,\,x)$$

La fonction `chebmult` permet de multiplier deux expressions polynômiales écrites sur la base des T_k. Les seconds arguments de la fonction `T` doivent être tous identiques.

```
> P:=1+T(3,z)+T(2,z); Q:=T(1,z)+T(4,z);'P*Q'=chebmult(P,Q);
```

$$P := 1 + \mathrm{T}(3,\,z) + \mathrm{T}(2,\,z)$$

$$Q := \mathrm{T}(1,\,z) + \mathrm{T}(4,\,z)$$

$$P\,Q = 2\,\mathrm{T}(1,\,z) + \mathrm{T}(2,\,z) + \frac{1}{2}\,\mathrm{T}(3,\,z) + \frac{3}{2}\,\mathrm{T}(4,\,z) + \frac{1}{2}\,\mathrm{T}(6,\,z) + \frac{1}{2}\,\mathrm{T}(7,\,z)$$

La fonction `chebmult` permet de retrouver une relation bien connue.

Il suffirait de préciser avec `assume(n>=1)` que l'entier n est positif pour se débarrasser de la valeur absolue dans le résultat :

```
> '2*x*T(n,x)'=chebmult(2*T(1,x),T(n,x));
```

$$2\,x\,\mathrm{T}(n,\,x) = \mathrm{T}(1+n,\,x) + \mathrm{T}(|n-1|\,,\,x)$$

La fonction `chebdeg` donne le degré maximum d'une expression algébrique sur les polynômes de Chebyshev. Voici par exemple le développement de Chebyshev de la fonction exponentielle, à la précision $\frac{1}{100}$:

```
> Digits:=2: p:=chebyshev(exp,x); Digits:=10 :
```

$$p := 1.3\,\mathrm{T}(0,\,x) + 1.1\,\mathrm{T}(1,\,x) + .27\,\mathrm{T}(2,\,x) + .044\,\mathrm{T}(3,\,x) + .0054\,\mathrm{T}(4,\,x)$$

La fonction `chebdeg` nous apprend maintenant que le polynôme précédent est de degré 4 relativement à la base des polynômes T_k.

Bien sûr ce résultat saute aux yeux, mais on doit penser à l'utilité de la fonction `chebdeg` dans une optique de programmation.

On ne sait pas en effet à l'avance jusqu'à quel degré la fonction `chebyshev` va devoir pousser le développement pour atteindre la précision voulue par l'utilisateur :

```
> chebdeg(p);
```

$$4$$

5. La fonction chebpade

Enfin le package `numapprox` contient une fonction `chebpade` dont le rôle est d'approcher une expression par un polynôme ou une fraction rationnelle, de degrés donnés à l'avance, exprimés en fonction des T_k :

```
> restart: with(numapprox,chebpade);
```

$$[chebpade]$$

Voici par exemple le développement polynômial, de degré 3 suivant les polynômes T_k, de la fonction exponentielle sur l'intervalle $[-1, 1]$, qui est le segment par défaut, comme avec la fonction `chebyshev`.

Contrairement à la fonction `chebyshev`, qui donnait un développement à une précision donnée, le résultat est ici fonction d'un degré maximum précisé par l'utilisateur :

```
> chebpade(exp(x),x,3);
```

$$1.266065878\,\mathrm{T}(0,\,x) + 1.130318208\,\mathrm{T}(1,\,x) + .2714953396\,\mathrm{T}(2,\,x) + .04433684985\,\mathrm{T}(3,\,x)$$

On retrouve ainsi le résultat obtenu avec les fonctions `scal` et `SN` au début de cette section :

```
> scal:=(f,g)->2/Pi*int(f*g/sqrt(1-x^2),x=-1..1) :
  SN:=(f,N)->1/2*scal(f,1)+Sum(scal(f,orthopoly[T](k,x))*T(k,x),k=1..N) :
  s:=evalf(SN(exp(x),3));
```

$$s := 1.266065878 + 1.130318208\,\mathrm{T}(1,\,x) + .2714953395\,\mathrm{T}(2,\,x) + .04433684985\,\mathrm{T}(3,\,x)$$

Voici le développement de $\exp x$, toujours sur $[-1, 1]$, au moyen d'une fraction rationnelle $R = \frac{P}{Q}$, où le numérateur P et le dénominateur Q sont exprimés sur les T_k et de degrés respectifs 3 et 2 :

```
> R:=chebpade(exp(x),x,[3,2]);
```

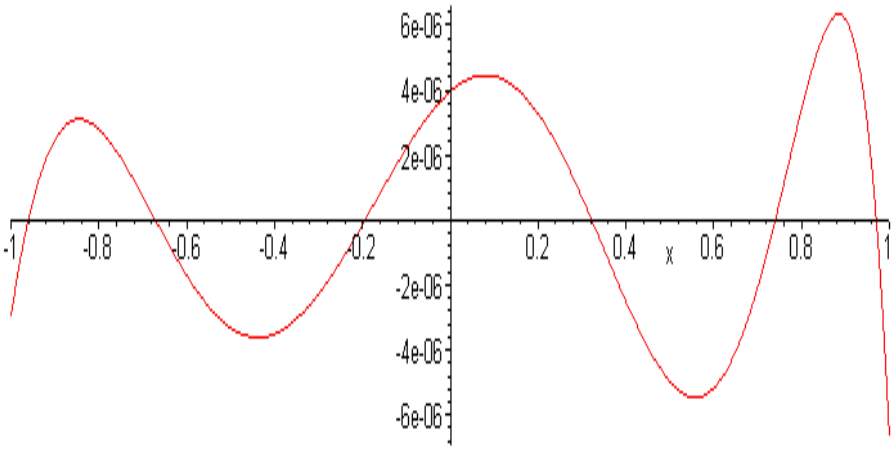
$$\begin{aligned} R := \; & (1.050531166\,\mathrm{T}(0,\,x) + .6016362122\,\mathrm{T}(1,\,x) + .07417897149\,\mathrm{T}(2,\,x) \\ & + .004109558353\,\mathrm{T}(3,\,x))/(\mathrm{T}(0,\,x) - .3870509565\,\mathrm{T}(1,\,x) + .02365167312\,\mathrm{T}(2,\,x)) \end{aligned}$$

Après avoir chargé en mémoire la fonction `T` du package `orthopoly`, nous pouvons afficher l'expression de R en fonction des puissances de x et tracer la différence entre la fraction rationnelle $R(x)$ et la fonction $\exp x$ sur l'intervalle $[-1, 1]$.

On voit que l'erreur maximum sur tout le segment est de l'ordre de $6 \cdot 10^{-6}$, ce qui est remarquable car on n'utilise que des polynômes de degré inférieur ou égal à 3 :

```
> with(orthopoly,T): 'R'=R; plot(R-exp(x),x=-1..1);
```

$$R = \frac{.9763521945 + .5893075371\,x + .1483579430\,x^2 + .01643823341\,x^3}{.9763483269 - .3870509565\,x + .04730334624\,x^2}$$



On voit par exemple que pour espérer atteindre une précision équivalente mais en se limitant à un développement polynômial, il faut aller jusqu'au degré 6 :

```
> P:=chebpade(exp(x),x,6); plot(P-exp(x),x=-1..1);
```

$$\begin{aligned} P := \; & 1.266065878\,\mathrm{T}(0,\,x) + 1.130318208\,\mathrm{T}(1,\,x) + .2714953396\,\mathrm{T}(2,\,x) \\ & + .04433684985\,\mathrm{T}(3,\,x) + .005474240442\,\mathrm{T}(4,\,x) + .0005429263119\,\mathrm{T}(5,\,x) \\ & + .00004497732296\,\mathrm{T}(6,\,x) \end{aligned}$$