Developpements_Limites

April 10, 2020

1 Développements limités

Marc Lorenzi

7 avril 2020

L'objectif de ce qui va suivre est de nous amuser avec des développements limités. Nous allons nous cantonner à des DLs **en 0** à coefficients **rationnels** pour les deux raisons suivantes :

- La manipulation de DLs à coefficients quelconque demande de mettre en place des techniques de calcul formel qui dépasseraient largement le cadre de ce notebook. En prenant des coefficients rationnels on dispose déjà d'une large classe de développements limités sur lesquels on peut faire des calculs exacts. Python dispose d'une classe `Fraction` qui permet de calculer sur les nombres rationnels.
- Faire des développements limités en un réel a quelconque ne surcharge pas énormément le code, mais notre restriction sur les coefficients rationnels fait que cela ne serait pas forcément très intéressant. En effet, la plupart des fonctions usuelles ont des DLs avec des coefficients irrationnels si l'on se place ailleurs qu'en 0. Par exemple, le DL de $\ln(1+x)$ au point 1 a un coefficient constant égal à $\ln 2$.

Nous allons définir tout au long du notebook des opérations sur l'ensemble $\mathcal{D}$:

- Addition
- Multiplication par un rationnel.
- Multiplication de deux DLs
- Intégration, dérivation.
- Composition $F \circ G$ (dans certains cas)

Nous nous demanderons quels sont les DLs inversibles pour la multiplication et la composition, et nous calculerons leur inverse pour ces opérations lorsque c'est possible.

Remarque : Nous chercherons dans ce qui suit à écrire un code qui s'exécute en un temps raisonnable, mais pas forcément le plus efficace possible. Pour améliorer la lisibilité j'ai parfois choisi d'écrire un code plus simple, mais moins rapide. Nous nous dispenserons de toute façon de faire des DLs à l'ordre 1000 …

1.1 1. Préliminaires

Importons la classe `Fraction` qui permet de manipuler des rationnels.

[1]:
```
from fractions import Fraction
```

Pour avoir un affichage sympathique, les quelques lignes qui suivent importent un “convertisseur LaTeX” et définissent une fonction qui transforme une fraction en chaîne de caractères représentant le code LaTeX de cette fraction. Voir plus loin pour les résultats.

[2]:
```
from IPython.display import display, Math
```

[3]:
```
def frac_to_latex(x):
    p = x.numerator
    q = x.denominator
    if q == 1: return str(p)
    else:
        return r'\frac{%d}{%d}' % (p, q)
```

[4]:
```
frac_to_latex(Fraction(13, 17))
```

[4]:
```
'\\frac{13}{17}'
```

1.2 2. Une classe pour les développements limités

Voici la classe `DL`. Un objet de cette classe représente un DL en 0 à l'ordre m.

$$F(x) = \sum_{k<m} F_k x^k + O(x^m)$$

Remarquons que, contrairement à l'habitude, nos DLs finiront par des **grands** O.

Un tel objet F possède trois champs :

- Un champ `ord` qui contient l'ordre m du DL.
- Un champ data qui contient la liste $[F_0, F_1, \ldots, F_{m-1}]$. `F.data[k]` est le coefficient F_k de degré k du DL F.
- Un champ `val` qui contient la valuation du DL, c'est à dire le plus petit entier $k < m$ tel que $F_k \neq 0$. La valuation est calculée lors de la construction de l'objet. Si tous les coefficients sont nuls, la valuation est égale à l'ordre.

Le constructeur prend en paramètre une liste de fractions et un paramètre optionnel `ord`. Si `ord` n'est pas donné, on le prend égal à la longueur de la liste moins 1.

Tout à la fin de la classe, on trouve la définition des méthodes `__add__`, …, `__rtruediv__`. Ces méthodes font appel à des fonctions pas encore écrites, `add`, `mul`, etc. Le but du notebook est précisément d'écrire ces fonctions ! Ces méthodes nous permettront d'écrire des expressions du genre `F + G`, …, `F / G` lorsque F et G sont des DLs.

Remarque : Le code de la classe `DL` a l'air bien long. Lisez éventuellement la méthode `__init__` et la méthode `Valuation`. Le reste est sans importance (si j'ose dire).

[5]:
```
class DL:

    def __init__(self, data, ord=None):
        if ord == None: self.ord = len(data)
        else: self.ord = ord
        self.data = data[: self.ord] # Tronquer si data est trop longue
        for k in range(len(data), self.ord):
            self.data.append(0)          # Rajouter des zéros si data est trop␣
↪courte
        for k in range(self.ord): self.data[k] = Fraction(self.data[k]) # Tout␣
↪convertir en Fractions
        self.val = self.valuation()      # Calculer la valuation du DL

    def valuation(self):
        k = 0
        while k < self.ord and self.data[k] == 0: k += 1
        return k

    def __str__(self):
        s = ''
        for k in range(self.ord):
            ak = self.data[k]
            if ak < 0: s += '%s*x^%d' % (ak, k)
            elif ak > 0: s += '+%s*x^%d' % (ak, k)
        s += '+O(x^%d)' % self.ord
        return s

    def to_latex(self):
        s = ''
        for k in range(self.ord):
            ak = self.data[k]
            if ak < 0: s += '-' + frac_to_latex(-ak) + (r'x^{%d}' % k)
            elif ak > 0: s += '+' + frac_to_latex(ak) + (r'x^{%d}' %  k)
        s += '+O(x^{%d})' % self.ord
        return s

    def copie(self):
        return DL(self.data.copy(), self.ord)

    def __add__(self, g):
        if type(g) == Fraction or type(g) == int: return add_scal(g, self)
        else: return add(self, g)
    def __radd__(self, g): return self + g
    def __neg__(self): return oppose(self)
    def __sub__(self, g): return self + (-g)
    def __rsub__(self, g): return -(self - g)
    def __mul__(self, g):
```

```
        if type(g) == Fraction or type(g) == int: return mul_scal(g, self)
        else: return mul(self, g)
    def __rmul__(self, g): return self * g
    def __pow__(self, n):
        if type(n) == int: return pow(self, n)
        elif type(n) == Fraction and self.data[0] == 1:
            g = self.copie()
            g.data[0] = Fraction(0)
            g.val = g.valuation()
            return comp(puiss(n, g.ord), g)
        else:
            raise Exception('puissance incorrecte')
    def __truediv__(self, g):
        if type(g) == Fraction or type(g) == int: return Fraction(1, g) * self
        else: return quotient(self, g)
    def __rtruediv__(self, g):
        return inverse(self / g)
```

La fonction `pprint` prend un DL en paramètre et elle l'affiche presque joliment. Il y aurait encore quelques améliorations à apporter (le cas des coefficients égaux à ± 1, une signe $+$ parasite en début de DL), mais cela nous suffira pour ce que nous voulons en faire.

[6]:
```
def pprint(F): display(Math(F.to_latex()))
```

Voici un exemple de DL, affiché de façon "standard" avec `print` puis affiché avec `pprint`.

[7]:
```
F = DL([Fraction(1, 2), -Fraction(3,4), Fraction(5, 6), 0, 7], 10)
print(F)
pprint(F)
```

```
+1/2*x^0-3/4*x^1+5/6*x^2+7*x^4+O(x^10)
```

$$+\frac{1}{2}x^0 - \frac{3}{4}x^1 + \frac{5}{6}x^2 + 7x^4 + O(x^{10})$$

Le second affichage est clairement préférable. Voici maintenant la fonction "grand O" :

[8]:
```
def O(n): return DL([], n)
```

[9]:
```
pprint(O(5))
```

$$+O(x^5)$$

Définissons aussi $X_{k,n} = x^k + O(x^n)$. Remarquons que si $k \geq n$, $x^k + O(x^n) = O(x^n)$. Inutile de s'en soucier, le constructeur de la classe `DL` fait ce qu'il faut pour simplifier les choses.

[10]:
```
def X(k, n):
    s = k * [0]
    s.append(1)
    return DL(s, n)
```

[11]:
```
pprint(X(3, 4))
```

$+1x^3 + O(x^4)$

[12]:
```
pprint(X(4, 4))
```

$+O(x^4)$

1.3 3. Développements limités usuels

Nous allons passer en revue les DLs usuels.

1.3.1 3.1 Exponentielle

Rappelons qu'un DL de l'exeponentielle à l'ordre n en 0 est

$$e^x = \sum_{k=0}^{n-1} \frac{x^k}{k!} + O(x^n)$$

[13]:
```
def exp(n):
    s = n * [Fraction(1)]
    for k in range(1, n):
        s[k] = s[k - 1] / k
    return DL(s)
```

[14]:
```
pprint(exp(6))
```

$+1x^0 + 1x^1 + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + O(x^6)$

1.3.2 3.2 Logarithme

On a

$$\ln(1+x) = \sum_{k=1}^{n-1} \frac{(-1)^{k-1}}{k} x^k + O(x^n)$$

[15]:
```
def ln(n):
    return DL([0] + [Fraction((-1) ** (k - 1), k) for k in range(1, n)])
```

[16]:
```
pprint(ln(10))
```

$+1x^1 - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \frac{1}{4}x^4 + \frac{1}{5}x^5 - \frac{1}{6}x^6 + \frac{1}{7}x^7 - \frac{1}{8}x^8 + \frac{1}{9}x^9 + O(x^{10})$

1.3.3 3.3 Inverse

On a

$$\frac{1}{1+x} = \sum_{k=0}^{n-1} (-1)^k x^k + O(x^n)$$

[17]:
```
def inv(n):
    return DL([(-1) ** k for k in range(n)])
```

[18]:
```
pprint(inv(10))
```

$+1x^0 - 1x^1 + 1x^2 - 1x^3 + 1x^4 - 1x^5 + 1x^6 - 1x^7 + 1x^8 - 1x^9 + O(x^{10})$

1.3.4 3.4 Puissances

Soit $a \in \mathbb{Q}$. On a

$$(1+x)^a = \sum_{k=0}^{n-1} \binom{a}{k} x^k + O(x^n)$$

où

$$\binom{a}{k} = \frac{a(a-1)\dots(a-k+1)}{k!}$$

[19]:
```
def puiss(a, n):
    s = n * [Fraction(1)]
    for k in range(1, n):
        s[k] = s[k - 1] * (a - k + 1) / k
    return DL(s)
```

[20]:
```
pprint(puiss(6, 10))
```

$+1x^0 + 6x^1 + 15x^2 + 20x^3 + 15x^4 + 6x^5 + 1x^6 + O(x^{10})$

On peut aussi prendre un exposant fractionnaire. Voici $\sqrt{1+x}$.

[21]:
```
pprint(puiss(Fraction(1, 2), 10))
```

$+1x^0 + \frac{1}{2}x^1 - \frac{1}{8}x^2 + \frac{1}{16}x^3 - \frac{5}{128}x^4 + \frac{7}{256}x^5 - \frac{21}{1024}x^6 + \frac{33}{2048}x^7 - \frac{429}{32768}x^8 + \frac{715}{65536}x^9 + O(x^{10})$

Ou aussi un exposant négatif. Voici $\frac{1}{(1+x)^{1/3}}$.

[22]:
```
pprint(puiss(Fraction(-1, 3), 8))
```

$$+1x^0 - \frac{1}{3}x^1 + \frac{2}{9}x^2 - \frac{14}{81}x^3 + \frac{35}{243}x^4 - \frac{91}{729}x^5 + \frac{728}{6561}x^6 - \frac{1976}{19683}x^7 + O(x^8)$$

1.3.5 3.5 sinus, cosinus, arc tangente

On a

$$\sin x = \sum_{k<m} \frac{(-1)^k}{(2k+1)!} x^{2k+1} + o(x^n)$$

où m est le plus petit entier tel que $2m + 1 \geq n$. On a donc

$$2m - 1 < n \leq 2m + 1$$

ou encore

$$m - 1 < \frac{n-1}{2} \leq m$$

et ainsi,

$$m = \left\lceil \frac{n-1}{2} \right\rceil = -\left\lfloor \frac{1-n}{2} \right\rfloor$$

[23]:
```
def sin(n):
    s = n * [Fraction(0)]
    if n > 1: s[1] = Fraction(1)
    m = -((1 - n) // 2)
    for k in range(1, m):
        s[2 * k + 1] = -s[2 * k - 1] / (2 * k * (2 * k + 1))
    return DL(s)
```

[24]:
```
pprint(sin(9))
```

$$+1x^1 - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + O(x^9)$$

On a

$$\cos x = \sum_{k<m} \frac{(-1)^k}{(2k)!} x^{2k} + O(x^n)$$

où m est le plus petit entier tel que $2m \geq n$. On a donc

$$2m - 2 < n \leq 2m$$

ou encore

$$m - 1 < \frac{n}{2} \leq m$$

et ainsi,

$$m = \left\lceil \frac{n}{2} \right\rceil = -\left\lfloor -\frac{n}{2} \right\rfloor$$

[25]:
```
def cos(n):
    s = n * [Fraction(0)]
    if n > 0: s[0] = Fraction(1)
    for k in range(1, -((-n) // 2)):
        s[2 * k] = -s[2 * k - 2] / ((2 * k - 1) * 2 * k)
    return DL(s)
```

[26]:
```
pprint(cos(9))
```

$$+1x^0 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + \frac{1}{40320}x^8 + O(x^9)$$

On a

$$\arctan x = \sum_{k<m} \frac{(-1)^k}{2k+1} x^{2k+1} + O(x^n)$$

où m est le plus petit entier tel que $2m + 1 > n$. On a donc

$$2m \leq 2m + 1 \leq n + 1 < 2m + 2$$

On est donc dans le même cadre que pour la fonction sinus :

$$m = -\left\lfloor \frac{1-n}{2} \right\rfloor$$

[27]:
```
def arctan(n):
    s = n * [Fraction(0)]
    m = -((1 - n) // 2)
    for k in range(m):
        s[2 * k + 1] = Fraction((-1) ** k, 2 * k + 1)
    return DL(s)
```

[28]:
```
pprint(arctan(10))
```

$$+1x^1 - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \frac{1}{9}x^9 + O(x^{10})$$

1.3.6 3.6 sinus, cosinus et argument tangente hyperboliques

Même chose en enlevant les puissances de -1.

[29]:
```
def sh(n):
    s = n * [Fraction(0)]
    if n > 1: s[1] = Fraction(1)
    m = -((1 - n) // 2)
    for k in range(1, m):
        s[2 * k + 1] = s[2 * k - 1] / (2 * k * (2 * k + 1))
    return DL(s)
```

[30]:
```
pprint(sh(10))
```

$$+1x^{1} + \frac{1}{6}x^{3} + \frac{1}{120}x^{5} + \frac{1}{5040}x^{7} + \frac{1}{362880}x^{9} + O(x^{10})$$

[31]:
```
def ch(n):
    s = n * [Fraction(0)]
    if n > 0: s[0] = Fraction(1)
    for k in range(1, -((-n) // 2)):
        s[2 * k] = s[2 * k - 2] / ((2 * k - 1) * 2 * k)
    return DL(s)
```

[32]:
```
def ch(n):
    s = (n + 1) * [Fraction(0)]
    s[0] = Fraction(1)
    for k in range(1, n // 2 + 1):
        s[2 * k] = s[2 * k - 2] / ((2 * k - 1) * 2 * k)
    return DL(s)
```

[33]:
```
pprint(ch(6))
```

$$+1x^{0} + \frac{1}{2}x^{2} + \frac{1}{24}x^{4} + \frac{1}{720}x^{6} + O(x^{7})$$

[34]:
```
def argth(n):
    s = n * [Fraction(0)]
    m = -((1 - n) // 2)
    for k in range(m):
        s[2 * k + 1] = Fraction(1, 2 * k + 1)
    return DL(s)
```

[35]:
```
pprint(argth(15))
```

$$+1x^{1} + \frac{1}{3}x^{3} + \frac{1}{5}x^{5} + \frac{1}{7}x^{7} + \frac{1}{9}x^{9} + \frac{1}{11}x^{11} + \frac{1}{13}x^{13} + O(x^{15})$$

Après cette longue énumération, passons aux opérations sur les développements limités.

1.4 4. Opérations sur les développements limités

1.4.1 4.1 Ordre et valuation d'un DL

Définition : Soit $F(x) = P(x) + O(x^m)$. L'ordre de F est $\omega(F) = m$.

Lorsque nous effectuerons des opérations sur des DLS (somme, produit, etc.), nous nous apercevrons que l'ordre du résultat dépend bien sur des ordres des deux DLs mais aussi, la plupart du temps, d'une autre quantité qui est leur **valuation**.

Définition Soit $F(x) = \sum_{k<m} P_k x^k + o(X^m)$. - Si au moins l'un des P_k est non nul, on appelle valuation de F, et on note $\nu(F)$ le plus petit entier $k < m$ tel que $P_k \neq 0$. - Si tous les P_k sont nuls, on pose $\nu(F) = m$.

Remarquons que $\nu(F) \leq \omega(F)$, et que l'on a égalité si et seulement si $F(x) = O(x^m)$.

1.4.2 4.1 Somme

Définition : Soient $F(x) = P(x) + O(x^m)$ et $G(x) = Q(x) + O(x^n)$ deux DLs. On a

$$(F + G)(x) = P(x) + Q(x) + O(x^r)$$

où

$$r = \min(m, n)$$

Proposition : Soient F, G deux DLs. On a

- $\nu(F + G) \geq \min(\nu(F), \nu(G))$. Si $\nu(F) \neq \nu(G)$ on a même égalité.
- $\omega(F + G) = \min(\omega(F), \omega(G))$.

[36]:
```
def add(F, G):
    r = min(F.ord, G.ord)
    s = r * [0]
    for k in range(r):
        s[k] = F.data[k] + G.data[k]
    return DL(s)
```

La ligne magique ci-dessous fonctionne parce que nous avons inclus dans la classe `DL` une méthode `__add__` qui appelle la fonction `add` que nous venons d'écrire.

[37]:
```
pprint(sin(5) + cos(7))
```

$$+1x^0 + 1x^1 - \frac{1}{2}x^2 - \frac{1}{6}x^3 + \frac{1}{24}x^4 + O(x^5)$$

La fonction bien utile `add_scal` ajoute le rationnel t au DL F. Si F est un DL à l'ordre m, elle renvoie la somme des deux DLs $(t + O(x^m))$ et $F(x)$

[38]:
```
def add_scal(t, F):
    return add(DL([t], F.ord), F)
```

La méthode `__add__` est suffisamment bien pensée pour nous permettre d'écrire :

[39]:
```
pprint(5 + cos(4))
```

$$+6x^0 - \frac{1}{2}x^2 + O(x^4)$$

1.4.3 4.2 Produit par un réel

Soient $F(x) = P(x) + O(x^m)$ et $t \in \mathbb{Q}$. Nous posons $(tF)(f) = tP(x) + O(x^m)$. Remarquons le résultat peu standard lorsque $t = 0$: $(0F)(x) = O(x^m)$, et pas 0.

[40]:
```
def mul_scal(t, F):
    n = F.ord
    s = n * [0]
    for k in range(n):
        s[k] = t * F.data[k]
    return DL(s)
```

[41]:
```
pprint(3 * cos(7))
```

$$+3x^0 - \frac{3}{2}x^2 + \frac{1}{8}x^4 - \frac{1}{240}x^6 + O(x^7)$$

[42]:
```
pprint(0 * exp(5))
```

$$+O(x^5)$$

Il est maintenant évident de faire des soustractions et, plus généralement, des combinaisons linéaires.

[43]:
```
def oppose(F): return mul_scal(-1, F)
```

[44]:
```
pprint(-sin(5))
```

$$-1x^1 + \frac{1}{6}x^3 + O(x^5)$$

[45]:
```
def sub(F, G):
    return add(F, oppose(G))
```

[46]:
```
pprint(2 * sin(5) - 3 * cos(7))
```

$$-3x^0 + 2x^1 + \frac{3}{2}x^2 - \frac{1}{3}x^3 - \frac{1}{8}x^4 + O(x^5)$$

[47]:
```
pprint(cos(5) - cos(5))
```

$$+O(x^5)$$

Remarquons que maintenant nous pouvons presque écrire nos DLs de façon naturelle :

[48]:
```
pprint(sum([Fraction(1, k ** 2) * X(k, 10) for k in range(1, 11)]))
```

$$+1x^1 + \frac{1}{4}x^2 + \frac{1}{9}x^3 + \frac{1}{16}x^4 + \frac{1}{25}x^5 + \frac{1}{36}x^6 + \frac{1}{49}x^7 + \frac{1}{64}x^8 + \frac{1}{81}x^9 + O(x^{10})$$

Question : L'ensemble $\mathcal{D}$ des DLs est-il un espace vectoriel ? La réponse est non. En revanche, si nous appelons $\mathcal{D}_n$ l'ensemble des DLs d'ordre n, alors $\mathcal{D}_n$ est un espace vectoriel de dimension $n+1$. Remarquons que son vecteur nul n'est pas 0, mais $O(x^n)$. $\mathcal{D} = \bigcup_{n\in\mathbb{N}} \mathcal{D}_n$ est donc une union d'espaces vectoriels disjoints ayant des neutres tous différents, et on a

$$\forall m \leq n, \mathcal{D}_m + \mathcal{D}_n = \mathcal{D}_m$$

1.4.4 4.3 Produit

Définition Soient F et G deux développements limités. On a

$$(FG)(x) = P(x)Q(x) + O(x^r)$$

où $r = \min(\omega(F) + \nu(G), \omega(G) + \nu(F))$.

Proposition : Soient F, G deux DLs. On a

- $\nu(FG) = \nu(F) + \nu(G)$.
- $\omega(FG) = \min(\omega(F) + \nu(G), \omega(G) + \nu(F))$.

La fonction `mul` effectuant le produit de deux DLs ne présente pas de difficulté particulière.

[49]:
```
def mul(F, G):
    m, n = F.ord, G.ord
    mu, nu = F.val, G.val
    r = min(m + nu, n + mu)
    s = r * [0]
    for i in range(m):
        for j in range(min(n, r - i)):
            s[i + j] += F.data[i] * G.data[j]
    return DL(s)
```

[50]:
```
pprint(cos(6) * exp(8))
```

$$+1x^0 + 1x^1 - \frac{1}{3}x^3 - \frac{1}{6}x^4 - \frac{1}{30}x^5 + O(x^6)$$

[51]:
```
pprint(O(4) * O(5))
```

$$+O(x^9)$$

[52]:
```
pprint(exp(5) * 10)
```

$$+10x^0 + 10x^1 + 5x^2 + \frac{5}{3}x^3 + \frac{5}{12}x^4 + O(x^5)$$

Exercice : Quel est un équivalent en 0 de $(2 + \cos x)(2 + \text{ch } x) - 9$? Facile !

[53]:
```
pprint((2 + cos(9))*(2 + ch(9)) - 9)
```

$$+\frac{1}{2016}x^8 + O(x^9)$$

1.4.5 4.3 Puissances

Proposition : Soit $F \in \mathcal{D}$. Soit $n \in \mathbb{N}^*$. On a - $\nu(F^n) = n\nu(F)$. - $\omega(F^n) = \omega(F) + (n-1)\nu(F)$.

Démonstration : Il suffit de faire une récurrence sur n. Pour $n = 1$ c'est évident. Supposons les égalités vraie pour un certain $n \geq 1$. On a alors

- $\nu(F^{n+1}) = \nu(F^n F) = \nu(F^n) + \nu(F) = n\nu(F) + \nu(F) = (n+1)\nu(F)$.
- $\omega(F^{n+1}) = \omega(F^n F) = \min(\omega(F^n) + \nu(F), \omega(F) + \nu(F^n)) = \min(\omega(F) + n\nu(F), \omega(F) + n\nu(F)) = \omega(F) + n\nu(F)$.

Remarque : Que vaut F^0 ? 1 me direz-vous. Mais 1 n'est pas un DL. Alors $1+O(x^n)$? Mais pour quel n ? Voilà une grande question. Décidons que nous voulons que la formule donnant l'ordre fonctionne pour $n = 0$. On a donc $\omega(F^0) = \omega(F) - \nu(F)$. Nous poserons donc

$$F^0 = 1 + O(x^{\omega(F)-\nu(F)})$$

Ce choix est peut-être discutable, mais nous allons nous y tenir.

[54]:
```
def pow(F, n):
    if n < 0: return inverse(pow(F, -n))
    elif n == 0: return 1 + O(F.ord - F.val)
    else:
        P = X(0, F.ord)
        for k in range(n): P = P * F
        return P
```

Testons avec $\sin x$.

[55]:
```
pprint(sin(6) ** 0)
```

$$+1x^0 + O(x^5)$$

[56]:
```
pprint(sin(6) ** 1)
```

$$+1x^1 - \frac{1}{6}x^3 + \frac{1}{120}x^5 + O(x^6)$$

[57]:
```
pprint(sin(6) ** 10)
```

$$+1x^{10} - \frac{5}{3}x^{12} + \frac{4}{3}x^{14} + O(x^{15})$$

Dernier petit test, $\sin^7 x \sin^4 x - \sin^{11} x$ devrait faire 0, à un O près.

[58]:
```
pprint(sin(10) ** 4 * sin(10) ** 7 - sin(10) ** 11)
```

$+O(x^{20})$

Remarque : Pour l'instant seules les puissances entières positives sont acceptées. Lorsque nous saurons inverser un DL nous pourrons également élever à une puissance négative. Lorsque nous saurons composer, nous pourrons sous certaines conditions) mettre une puissance fractionnaire.

1.4.6 4.4 Composée

Soient $F, G \in \mathcal{D}$. Posons $F(x) = \sum_{k<m} F_k x^k + O(x^m)$. Supposons de plus que $\nu(G) \geq 1$. On a alors

$$(F \circ G)(x) = \sum_{k<m} F_k G(x)^k + O(G(x)^m)$$

Proposition : Posons $\mu = \nu(F - F_0)$. On a $\omega(F \circ G) = \min(\omega(F)\nu(G), \omega(G) + (\mu - 1)\nu(G))$.

Remarque : Dans le cas fréquent où $\mu = 1 = \nu(G)$, on obtient $\omega(F \circ G) = \min(\omega(F), \omega(G))$.

[59]:
```
def comp(F, G):
    if G.val == 0:
        raise Exception('comp : composition impossible')
    mu = (F - F.data[0]).val
    r = min(F.ord * G.val, G.ord + (mu - 1) * G.val)
    H = O(r)
    Gk = X(0, G.ord)
    for k in range(F.ord):
        if F.data[k] != 0:
            H = H + F.data[k] * Gk
        if k < F.ord: Gk = Gk * G
    return H
```

Composons des DLs à l'ordre 20 de e^x et $\ln(1+x)$. On obtient un résultat rassurant.

[60]:
```
pprint(comp(exp(20), ln(20)))
```

$+1x^0 + 1x^1 + O(x^{20})$

Maintenant nous pouvons élever un DL F à une puissance $a \in \mathbb{Q}$, pourvu que $F_0 = 1$. En effet, dans ce cas, $F = 1 + G$ où $\nu(G) \geq 1$. On peut donc calculer $R \circ G$ où $R = \sqrt{1+X}$.

[61]:
```
F = cos(20) ** Fraction(1, 2)
pprint(F)
```

$$+1x^0 - \frac{1}{4}x^2 - \frac{1}{96}x^4 - \frac{19}{5760}x^6 - \frac{559}{645120}x^8 - \frac{29161}{116121600}x^{10} - \frac{2368081}{30656102400}x^{12} - \frac{276580459}{11158821273600}x^{14} - \frac{43947282079}{5356234211328000}x^{16} - \frac{9118829535121}{3278015337332736000}x^{18} + O(x^{20})$$

Vérifions. On devrait avoir $F(x)^2 - \cos x = 0$ à $O(x^{20})$ près.

[62]:
```
pprint(F ** 2 - cos(20))
```

$+O(x^{20})$

1.4.7 4.5 Inverse, quotient

Proposition : Soit $F \in \mathcal{D}$ tel que $\nu(F) = 0$. Il existe $K \in \mathcal{D}$ tel que $FK = o(X^{\omega(F)})$.

Démonstration : $\nu(F) = 0$ signifie que $F_0 \neq 0$. En posant $H = \frac{F}{F_0} - 1$, on obtient

$$F = F_0(1 + H)$$

où $\nu(H) \geq 1$ et $\omega(H) = n = \omega(F)$. Soit $G(x) = 1 - x + x^2 + \ldots + (-1)^n x^n + O(x^n)$. On a alors

$$(1 + H(x))(G \circ H)(x) = 1 + O(x^n)$$

et donc $F(x)K(x) = O(x^n)$ où

$$K(x) = \frac{1}{F_0}(G \circ H)(x)$$

[63]:
```
def inverse(F):
    G = inv(F.ord)
    F0 = F.data[0]
    if F0 == 0: raise Exception('inverse: calcul impossible')
    H = (1 / F0) * F - 1
    return (1 / F0) * comp(G, H)
```

[64]:
```
pprint(1 / cos(10))
```

$+1x^0 + \frac{1}{2}x^2 + \frac{5}{24}x^4 + \frac{61}{720}x^6 + \frac{277}{8064}x^8 + O(x^{10})$

[65]:
```
pprint((1 + sh(10)) * (1 / (1 + sh(10))))
```

$+1x^0 + O(x^{10})$

On peut maintenant écrire des puissances négatives.

[66]:
```
pprint(cos(10) ** (-1))
```

$+1x^0 + \frac{1}{2}x^2 + \frac{5}{24}x^4 + \frac{61}{720}x^6 + \frac{277}{8064}x^8 + O(x^{10})$

Faire un quotient est bien entendu immédiat.

[67]:
```
def quotient(f, g):
    return mul(f, inverse(g))
```

Quel est le DL à l'ordre 13 en 0 de la fonction tangente ?

[68]:
```
def tan(n):
    return sin(n) / cos(n)
```

[69]:
```
pprint(tan(13))
```

$$+1x^1 + \frac{1}{3}x^3 + \frac{2}{15}x^5 + \frac{17}{315}x^7 + \frac{62}{2835}x^9 + \frac{1382}{155925}x^{11} + O(x^{13})$$

Vérification : calculons $\tan x \cos x - \sin x$, à l'ordre 20. Cela devrait donner $O(x^{20})$.

[70]:
```
pprint(tan(20) * cos(20) - sin(20))
```

$$+O(x^{20})$$

Quel est un équivalent en 0 de $\sin(\tan x) - \tan(\sin x)$?

[71]:
```
pprint(comp(sin(8), tan(8)) - comp(tan(8), sin(8)))
```

$$-\frac{1}{30}x^7 + O(x^8)$$

1.4.8 4.6 Primitives

Soit $F(x) = \sum_{k<n} F_k x^k + O(x^n)$. On pose

$$\int_0^x F = \sum_{k<n} \frac{F_k}{k+1} x^{k+1} + O(x^{n+1})$$

La primitive de F qui vaut $c \in \mathbb{Q}$ en 0 est alors $c + \int_0^x F$.

[72]:
```
def primitive(F, c):
    n = F.ord
    s = (n + 1) * [0]
    for k in range(n):
        s[k + 1] = F.data[k] / (k + 1)
    s[0] = c
    return DL(s)
```

Par exemple, si on intègre un DL de $\frac{1}{x^2+1}$, on obtient un DL de la fonction arc tangente.

[73]:
```
pprint(primitive(1 / (1 + X(2, 10)), 0))
```

$$+1x^1 - \frac{1}{3}x^3 + \frac{1}{5}x^5 - \frac{1}{7}x^7 + \frac{1}{9}x^9 + O(x^{11})$$

pour calculer un DL à l'ordre n de la fonction arcsin, il suffit de calculer un DL de $\frac{1}{\sqrt{1-x^2}}$ à l'ordre $n-1$, puis d'intégrer.

[74]:
```
def arcsin(n):
    if n == 0: return o(0)
    else:
        F = - X(2, n - 1)
        G = comp(puiss(Fraction(-1, 2), n - 1), F)
        return primitive(G, 0)
```

[75]:
```
pprint(arcsin(20))
```

$$+1x^1+\frac{1}{6}x^3+\frac{3}{40}x^5+\frac{5}{112}x^7+\frac{35}{1152}x^9+\frac{63}{2816}x^{11}+\frac{231}{13312}x^{13}+\frac{143}{10240}x^{15}+\frac{6435}{557056}x^{17}+\frac{12155}{1245184}x^{19}+O(x^{20})$$

Vérifions tout cela en calculant un DL de $\sin\circ\arcsin$.

[76]:
```
pprint(comp(sin(20), arcsin(20)))
```

$$+1x^1+O(x^{20})$$

pour calculer un DL à l'ordre n de la fonction argsh, il suffit de calculer un DL de $\frac{1}{\sqrt{1+x^2}}$ à l'ordre $n-1$, puis d'intégrer.

[77]:
```
def argsh(n):
    if n == 0: return o(0)
    else:
        F = -X(2, n - 1)
        G = comp(puiss(Fraction(-1, 2), n - 1), F)
        return primitive(G, 0)
```

[78]:
```
pprint(argsh(20))
```

$$+1x^1+\frac{1}{6}x^3+\frac{3}{40}x^5+\frac{5}{112}x^7+\frac{35}{1152}x^9+\frac{63}{2816}x^{11}+\frac{231}{13312}x^{13}+\frac{143}{10240}x^{15}+\frac{6435}{557056}x^{17}+\frac{12155}{1245184}x^{19}+O(x^{20})$$

1.5 5. retour aux DLs usuels

1.5.1 5.1 L'exponentielle

Dans le paragraphe sur les développements usuels nous avons *parachuté* le DL à l'ordre n de $f(x)=\exp x$. Serait-il possible, à ce stade du notebook, de le faire *calculer* par Python ? La réponse est oui. Tout tient en deux faits : 1. $f'=f$ 2. $f(0)=1$.

Deux c'est encore trop. On peut compacter cela en une seule propriété :

$$f(x)=1+\int_0^x f(t)\,dt$$

Exercice : Montrer que 1. et 2. équivalent à la ligne ci-dessus.

Ainsi, en intégrant un DL de l'exponentielle et en ajoutant une constante convenable (1 en l'occurrence), on obtient à nouveau un DL de l'exponentielle, à un ordre 1 de plus. Notons $DL_n(\exp)$ le DL de l'exponentielle à l'ordre n. Nous avons donc

$$DL_n(\exp)(x) = 1 + \int_0^x DL_{n-1}(\exp)(t)\,dt$$

Or nous savons que $DL_0(\exp)(x) = 1 + o(1)$. En intgérant n fois cette égalité on obtient le DL à l'ordre n. Voici notre nouvelle fonction exp qui **calcule** le DL de l'exponentielle.

[79]:
```
def exp2(n):
    f = X(0, 0)
    for k in range(n):
        f = 1 + primitive(f, 0)
    return f
```

[80]:
```
pprint(exp2(10))
```

$$+1x^0 + 1x^1 + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \frac{1}{24}x^4 + \frac{1}{120}x^5 + \frac{1}{720}x^6 + \frac{1}{5040}x^7 + \frac{1}{40320}x^8 + \frac{1}{362880}x^9 + O(x^{10})$$

1.5.2 5.2 Sinus et cosinus

On peut faire de même pour sinus et cosinus en remarquant que

$$\begin{cases} \sin x &= 0 + \int_0^x \cos t\,dt \\ \cos x &= 1 - \int_0^x \sin t\,dt \end{cases}$$

Sachant qu'un DL à l'ordre 0 de $\sin x$ et $\cos x$ est $\sin x = o(1)$ et $\cos x = 1 + o(1)$, n intégrations successives nous fourniront les DLs à l'ordre n. Voici la fonction `sincos` qui fait ce travail.

[81]:
```
def sincos(n):
    fsin = O(0)
    fcos = X(0, 0)
    for k in range(n):
        fsin, fcos = primitive(fcos, 0), 1 - primitive(fsin, 0)
    return fsin, fcos
```

Et voici donc nos DLs de sin et cos *calculés* et pas *parachutés*.

[82]:
```
def sin2(n): return sincos(n)[0]
def cos2(n): return sincos(n)[1]
```

[83]:
```
pprint(sin2(10))
```

$$+1x^1 - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \frac{1}{362880}x^9 + O(x^{10})$$

[84]:
```
pprint(cos2(10))
```

$$+1x^0 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + \frac{1}{40320}x^8 + O(x^{10})$$

1.5.3 5.3 Puissances

Soit $a \in \mathbb{Q}$. Notons $f_a(x) = (1+x)^a$. On a

$$(1+x)^a = 1 + a\int_0^x (1+x)^{a-1}\,dx$$

On peut donc ici obtenir un DL à l'ordre n de f_a en intégrant un DL à l'ordre $n-1$ de f_{a-1}. La situation est un peu plus compliquée que dans le cas de l'exponentielle, du sinus et du cosinus, et une fonction récursive se prête bien à la résolution du problème :

[85]:
```
def puiss2(a, n):
    if n == 0: return X(0, 0)
    else:
        return 1 + a * primitive(puiss2(a - 1, n - 1), 0)
```

[86]:
```
pprint(puiss2(-1, 10))
```

$$+1x^0 - 1x^1 + 1x^2 - 1x^3 + 1x^4 - 1x^5 + 1x^6 - 1x^7 + 1x^8 - 1x^9 + O(x^{10})$$

[87]:
```
pprint(puiss2(6, 10))
```

$$+1x^0 + 6x^1 + 15x^2 + 20x^3 + 15x^4 + 6x^5 + 1x^6 + O(x^{10})$$

[88]:
```
pprint(puiss2(Fraction(1, 2), 10))
```

$$+1x^0 + \frac{1}{2}x^1 - \frac{1}{8}x^2 + \frac{1}{16}x^3 - \frac{5}{128}x^4 + \frac{7}{256}x^5 - \frac{21}{1024}x^6 + \frac{33}{2048}x^7 - \frac{429}{32768}x^8 + \frac{715}{65536}x^9 + O(x^{10})$$

[89]:
```
pprint(puiss(Fraction(1, 2), 10))
```

$$+1x^0 + \frac{1}{2}x^1 - \frac{1}{8}x^2 + \frac{1}{16}x^3 - \frac{5}{128}x^4 + \frac{7}{256}x^5 - \frac{21}{1024}x^6 + \frac{33}{2048}x^7 - \frac{429}{32768}x^8 + \frac{715}{65536}x^9 + O(x^{10})$$

1.5.4 5.4 Les autres DLs usuels

Que reste-t-il à examiner ? Les fonctions hyperboliques (exercice), l'arc sinus, le logarithme et l'arc tangente. Mais ces fonctions s'obtiennent en primitivant $\frac{1}{\sqrt{1-x^2}}$, $\frac{1}{1+x}$ et $\frac{1}{1+x^2}$. Il est donc immédiat de faire *calculer* leur DL par Python.

En conclusion, nous avons maintenant suffisamment de fonctions sur les DLs pour permettre à Python de *calculer* tous les DLs usuels en utilisant des propriétés caractéristiques des fonctions usuelles.

1.6 6. Réciproque d'un développement limité

1.6.1 6.1 Dérivée d'un DL

Soit f une fonction ayant un DL à l'ordre $n \geq 1$ en 0, dérivable au voisinage de 0 et telle que f' a un DL à l'ordre $n-1$ en 0. Alors on obtient un DL de f' à l'ordre $n-1$ en dérivant formellement le DL de f. Formellement,

$$\left(\sum_{k<n} f_k x^k + O(x^n)\right)' = \sum_{k<n-1} (k+1) f_{k+1} x^k + O(x^{n-1})$$

Le cas $n = 1$ est peu intéressant mais pas faux. Il nous dit que si f est dérivable au voisinage de 0 et si f' est bornée au voisinage de 0, alors $(f_0 + O(x))' = O(1)$, ce qui est vrai.

[90]:
```
def derivee(f):
    n = f.ord
    if n == 0:
        raise Exception('derivee: calcul impossible')
    s = (n - 1)  * [0]
    for k in range(n - 1):
        s[k] = (k + 1) * f.data[k + 1]
    return DL(s)
```

Par exemple, en dérivant le DL de $\sin x$ à l'ordre 8, on obtient le DL de $\cos x$ à l'ordre 7.

[91]:
```
pprint(derivee(sin(8)))
```

$$+1x^0 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + O(x^7)$$

1.6.2 6.2 Résolution d'une équation

Étant donnés deux DLs f et h, peut-on trouver un DL g tel que $g \circ f = h$?

Soit

$$f(x) = \sum_{k<n} f_k x^k + O(x^n)$$

un DL à l'ordre $n \geq 2$. On suppose que $f_0 = 0$ et $f_1 \neq 0$. Soit

$$h(x) = \sum_{k<m} h_k x^k + O(x^m)$$

un DL à l'ordre $m \geq 0$. Nous allons déterminer un DL g, à un ordre aussi grand que possible, tel que $g \circ f = h$. Quel ordre peut-on espérer ? Eh bien nous allons voir par récurrence sur m que c'est $\min(m, n + \nu(h'))$.

- Commençons par le cas évident où $m = 0$. On cherche $g(x) = O(1)$ tel que $O(1) = O(1)$. Il n'y a rien à faire.
- Soit maintenant $m \geq 1$ et supposons que nous ayons résolu le problème pour tous les ordres strictement inférieurs à m. Le DL g convient si et seulement si $g_0 = h_0$ et $(g \circ f)' = h'$, ou encore $f' \times g' \circ f = h'$. Comme on a supposé $f_1 = f'_0 \neq 0$, le DL f' est inversible pour la multiplication. Ainsi, g convient si et seulement si $g_0 = h_0$ et

$$g' \circ f = \frac{h'}{f'}$$

Or $h_1 = \frac{h'}{f'}$ est un DL d'ordre $m_1 = \min(m-1, n-1+\nu(h'))$, donc d'ordre strictement inférieur à m. Par l'hypothèse de récurrence il existe un DL g_1 d'ordre m_1 tel que $g_1 \circ f = h_1$. On en déduit par une primitivation un DL g d'ordre $m_1 + 1 = \min(m, n+\nu(h'))$ tel que $g \circ f = h$.

Il y a donc existence et unicité de g. La fonction `solve_comp` qui calcule g à partir de f et h est facile à écrire récursivement.

[92]:
```
def solve_comp(f, h):
    if h.ord == 0: return O(0)
    else:
        h1 = quotient(derivee(h), derivee(f))
        g1 = solve_comp(f, h1)
        return primitive(g1, h.data[0])
```

Prenons un exemple. Cherchons g tel que $g(\sin x) = \cos x$. L'avantage est que l'on connaît la réponse : $g(x) = \sqrt{1-x^2}$.

Résolvons …

[93]:
```
pprint(solve_comp(sin(10), cos(10)))
```

$$+1x^0 - \frac{1}{2}x^2 - \frac{1}{8}x^4 - \frac{1}{16}x^6 - \frac{5}{128}x^8 + O(x^{10})$$

Puis vérifions.

[94]:
```
pprint((1 - X(2, 10)) ** Fraction(1, 2))
```

$$+1x^0 - \frac{1}{2}x^2 - \frac{1}{8}x^4 - \frac{1}{16}x^6 - \frac{5}{128}x^8 + O(x^{10})$$

1.6.3 6.3 Réciproque d'un DL

Calculer la réciproque du DL f à l'ordre n consiste à prendre $h(x) = x + o(x^n)$.

[95]:
```
def reciproque(f):
    return solve_comp(f, X(1, f.ord))
```

Retrouvons un DL de $\arcsin x$.

[96]:
```
pprint(reciproque(sin(15)))
```

$$+1x^1 + \frac{1}{6}x^3 + \frac{3}{40}x^5 + \frac{5}{112}x^7 + \frac{35}{1152}x^9 + \frac{63}{2816}x^{11} + \frac{231}{13312}x^{13} + O(x^{15})$$

Vérifions.

[97]:
```
pprint(arcsin(15))
```

$$+1x^1 + \frac{1}{6}x^3 + \frac{3}{40}x^5 + \frac{5}{112}x^7 + \frac{35}{1152}x^9 + \frac{63}{2816}x^{11} + \frac{231}{13312}x^{13} + O(x^{15})$$

Terminons par une fonction dont on ne connaît pas explicitement la réciproque. Prenons $f(x) = xe^x$. On a $f(0) = 0$, $f'(0) = 1 \neq 0$ et f est donc une bijection d'un voisinage de 0 sur un voisinage de 0.

[98]:
```
f = X(1, 10) * exp(9)
pprint(f)
```

$$+1x^1 + 1x^2 + \frac{1}{2}x^3 + \frac{1}{6}x^4 + \frac{1}{24}x^5 + \frac{1}{120}x^6 + \frac{1}{720}x^7 + \frac{1}{5040}x^8 + \frac{1}{40320}x^9 + O(x^{10})$$

[99]:
```
g = reciproque(f)
pprint(g)
```

$$+1x^1 - 1x^2 + \frac{3}{2}x^3 - \frac{8}{3}x^4 + \frac{125}{24}x^5 - \frac{54}{5}x^6 + \frac{16807}{720}x^7 - \frac{16384}{315}x^8 + \frac{531441}{4480}x^9 + O(x^{10})$$

Vérification :

[100]:
```
pprint(comp(f, g))
pprint(comp(g, f))
```

$$+1x^1 + O(x^{10})$$

$$+1x^1 + O(x^{10})$$

1.6.4 6.4 Quelques remarques sur les réciproques des DLs usuels

Que nous apporte le calcul de réciproques en ce qui concerne les fonctions usuelles ? En fait, rien. Passons-les en revue :

- sin, tan, circulaires et hyperboliques : nous disposons déjà des DLs de leurs réciproques.
- cos : la fonction arccos n'est pas dérivable en 0, elle n'y possède donc pas de DL (sauf à l'ordre 0).
- exp, ou plus exactement $e^x - 1$, pour que la fonction s'annule en 0. On a

$$y = e^x - 1 \Leftrightarrow x = \ln(y + 1)$$

et donc la réciproque de $e^x - 1$ est $\ln(1 + x)$, dont on possède déjà un DL.

- $(1 + x)^a - 1$. Cette fois,

$$y = (1 + x)^a - 1 \Leftrightarrow x = (1 + y)^{\frac{1}{a}} - 1$$

et donc la réciproque de $(1+x)^a - 1$ est $(1+x)^{\frac{1}{a}} - 1$.