Polynômes : fonctionnalités supplémentaires

Voici quelques fonctions utiles pour manipuler les polynômes :

Dérivation

Proposé et testé par RL, étudiant ba2 2012-2013. <sxh python; title : derivation.py> # -*- coding: utf-8 -*- def polyderiv(a):

```
  """dérivation d'un polynôme
  """
  b=a[:]  #copie de la liste des coefficients du polynôme de départ
  n=len(b)-1  #ordre du polynôme
  for i in range (n+1):
      b[i]=b[i]*i  #on redéfinit chaque coefficient i de la liste par ce
même coefficient*le degré
  b.pop(b[0])  #on supprime le premier élément de la liste (terme
indépendant)
  return b
```

</sxh>

Multiplication par x

Proposition de AP, étudiant ba2 2012-2013 : <sxh python; title : polyx.py> # -*- coding: utf-8 -*- def polyx(a):

```
  """polyx est un fonction qui multiplie un polynôme par x.
  Cela revient à rajouter un 0 à gauche de la liste passée en argument de la
fonction (a).
  """
  b=[0]               #Nouvelle liste dont le premier terme vaut 0.
  for coef in range(len(a)):           #Pour tout les coefficients de la
liste a de degré n, on ajoute la liste b.
      b.append(a[coef])
  return b
```

</sxh> Les listes pouvant être concaténées, on peut écrire cela plus simplement encore : <sxh python; title : polyshift.py> # -*- coding: utf-8 -*- def polyshift (a):

```
  """Multiplication du polynôme par la variable x"""
  b=[0]+a   # cela revient à "shifter" la liste des coefficients en insérant
un 0 "à gauche"
  return b
```

</sxh>

Intégration

Sur base de la proposition de RL, étudiant ba2 2012-2013 : <sxh python; title : polyintegr.py> def polyintegr(a):

```
  """intégration d'un polynôme
  """
  b=[0]  #on indique un coefficient indépendant nul en début de la liste
(constante d'intégration nulle)
  n=len(a)  #ordre du polynôme après intégration == ordre du polynôme avant
intégration +1
  for i in range (n):  # on balaie sur toutes les puissances i successives
      b.append(a[i]/(i+1)) #le coefficient du terme correspondant après
intégration est ajouté
  return b
```

</sxh>

Multiplication de deux polynômes

Proposition de BF, étudiant ba2 2012-2013, complétée par des commentaires et une application sur des puissances d'un binôme générant les coefficients du triangle de Pascal : <sxh python; title : polymult_BF.py> #!/usr/bin/python # -*- coding: UTF-8 -*- def polymult_BF(a,b):

```
  n=len(a)-1  # degré du polynôme a
  m=len(b)-1  # degré du polynôme b
  p=n+m  # degré du polynôme c = a * b
  c=[] # on va créer tous les coefficients du polynôme c, initialisés à 0 :
  while(len(c)<p+1):
      c.append(0)
  for k in range(p+1):  # pour toutes les puissances du polynôme c
      for i in range(len(a)):  # on considère tous les termes de a
          for j in range(len(b)): # on considère tous les termes de b
              if(k==i+j):  # si les degrés combinés valent k, on met à jour
le coefficient k :
                  c[k]=c[k]+(a[i]*b[j])
  return c
```

x=[1,1] prod=[1,1] for i in range(10):

```
  prod=polymult_BF(x,prod)
  print prod
```

</sxh> OK, cela fonctionne, mais il semble possible d'économiser des opérations. Multiplier un

polynômes de degré n par un autre de degré m devrait pouvoir se faire en n*m étapes. Or, il y a 3 structures de répétitions imbriquées, donc n*m*(n+m) étapes ! De plus, on ne tire pas vraiment parti de l'initialisation du polynôme c, puisque ses coefficients sont créés dans l'ordre.

Voici une modification permettant d'en tirer parti, supprimant le balayage des puissances de c, le test, et quelques autres éléments inutiles : <sxh python; title : polymult.py> #!/usr/bin/python # -*- coding: UTF-8 -*- def polymult(a,b):

```
  n=len(a)-1  # degré du polynôme a
  m=len(b)-1  # degré du polynôme b
  p=n+m  # degré du polynôme c = a * b
  c=[] # on va créer tous les coefficients du polynôme c, initialisés à 0 :
  for k in range(p+1):
      c.append(0)
  for i in range(n+1):  # on considère tous les termes de a
      for j in range(m+1): # on considère tous les termes de b
          c[i+j]+=a[i]*b[j]  # on incrémente le coefficient de degré k :
  return c
```

x=[1,1] prod=[1,1] for i in range(10):

```
  prod=polymult(x,prod)
  print prod
```

</sxh>

Génération par récurrence des polynômes de Legendre

Proposition de GH, étudiant ba2 2012-2013, complétée par des commentaires. La fonction nécessite d'autres fonctions vues précédemment, qui sont reprises dans le code du programme : <sxh python; title : polylegendre.py> #!/usr/bin/python # -*- coding: UTF-8 -*- def polylegendre(nmax):

```
  """Fonction générant les coefficients des polynômes de Legendre jusqu'à
l'ordre nmax
  cf. http://fr.wikipedia.org/wiki/Polyn%C3%B4me_de_Legendre pour la formule
de récurrence
  """
  rep=[[1.],[0.,1.]] # les deux premiers polynômes (degrés 0 et 1) pour
l'application de la formule de récurrence
  if nmax < 1: # si nmax est inférieur au degré 1, on renvoie le polynôme de
degré 0
      rep=[[1.]]
  if nmax > 1:  # pour le degré max supérieur à deux, on calcule les
polynômes suivants
      for n in range(2,nmax+1): # P_n(x) = (2n-1)/n x P_n-1(x) + (1-n)/n
P_n-2(x)
rep.append(polyadd(polyscal((2*n-1.)/n,polyshift(rep[n-1])),polyscal((1.-
n)/n,rep[n-2])))
  return rep
```

def polyscal(s,a):

```
  """polynôme multiplié par un scalaire s """
  b=[]
  for coef in a:
      b.append(coef*s)
  return b    # on retourne les coefficients multipliés par s
```

def polyshift(a):

```
  """Multiplication du polynôme par la variable x"""
  b=[0]+a   # cela revient à "shifter" la liste des coefficients en insérant
un 0 "à gauche"
  return b
```

def polyadd(a,b):

```
  """ Addition de deux polynômes de coefficients a et b
  """
  r=a[:] # on travaille sur une copie de a pour ne pas le modifier
  t=b[:] # idem pour b
  g=[]   # polynôme somme
  n1=len(r) # ordre du premier polynôme
  n2=len(t) # ordre du second polynôme
  if n1>n2: # premier polynôme de plus haut degré que le second
      for i in range (n1-n2):
          t.append(0)
  elif n1<n2: # second polynôme de plus haut degré que le premier
       for i in range (n2-n1):
           r.append(0)
  # r et t ont à présent la même longueur
  for i in range (len(r)):
      g.append(r[i]+t[i])
  return g  # on retourne les coefficients additionnés dans la liste g
```

test de la fonction générant les polynômes de Legendre : for k in range(6): # on teste la fonction sur des ordres croissants

```
  print polylegendre(k)[k]  # on imprime juste les coefficients du polynôme
de plus haut ordre !
```

</sxh>

Reste à créer un graphe...

From:
https://dvillers.umons.ac.be/wiki/ - **Didier Villers, UMONS - wiki**

Permanent link:
https://dvillers.umons.ac.be/wiki/teaching:progappchim:polynomes-10?rev=1353573942

Last update: **2012/11/22 09:45**