MILHAUD Xavier
VOISIN Basile

TP 3 : Différences divisées, polynômes de Newton

Question 1 *question1.*

Calcul à la main des différences divisées des 5 points $(-3,2);(-1,0);(0,2);(2,-1);(4,1)$ *:*
$f[x_0]] = 2$
$f[x_1]] = 0$ $f[x_0,x_1] = -1$
$f[x_2]] = 2$ $f[x_1,x_2] = 2$ $f[x_0,x_1,x_2] = 1$
$f[x_3]] = -1$ $f[x_2,x_3] = -\frac{3}{2}$ $f[x_1,x_2,x_3] = -\frac{7}{6}$ $f[x_0,x_1,x_2,x_3] = -\frac{13}{30}$
$f[x_4]] = 1$ $f[x_3,x_4] = 1$ $f[x_2,x_3,x_4] = \frac{5}{8}$ $f[x_1,x_2,x_3,x_4] = \frac{43}{120}$ $f[x_0,x_1,x_2,x_3,x_4] = \frac{19}{168}$

Question 2 *question2.*

```
//Fonction diff_div.sci
function m = diff_div(x,y)
n = length(x) ;
m = zeros(n,n) ;
m( :,1) = y ;
for j = 2 :n
for i = j :n
m(i,j) = (m(i,j-1)-m(i-1,j-1))/(x(i)-x(i-j+1)) ;
end
end
```

En testant sur les mêmes points qu'à la question précédente, on obtient :
! 2. 0. 0. 0. 0. !
! 0. - 1. 0. 0. 0. !
! 2. 2. 1. 0. 0. !
! - 1. - 1.5 - 1.1666667 - 0.4333333 0. !
! 1. 1. 0.625 0.3583333 0.1130952 !

Question 3 *question3.*

1. *question 3.1*

   Pour obtenir les différences divisées de la fonction $f_1(x) = 1 + \sin(3x)$*, on tape les commandes :*

```
x=linspace(0,2,6)
deff("y=f(x)","y=(1)+sin(3*x)")
v=f(x)
diff_div(x',v')
```

   Et on obtient :
   ! 1. 0. 0. 0. 0. 0. !
   ! 1.9320391 2.3300977 0. 0. 0. 0. !
   ! 1.6754632 - 0.6414398 - 3.7144218 0. 0. 0. !
   ! 0.5574796 - 2.7949591 - 2.6918991 0.8521023 0. 0. !
   ! 0.0038354 - 1.3841104 1.7635608 3.7128833 1.7879881 0. !
   ! 0.7205845 1.7918728 3.969979 1.8386818 - 1.1713759 - 1.479682 !

2. *question 3.2*
   Rajoutons un point $x = 0,2$ *(sans retrier le vecteur x) :*
   x = [x,[0.2]]
   v = f(x)
   diff_div(x',v')
   Et on obtient :
   ! 1. 0. 0. 0. 0. 0. 0. !
   ! 1.9320391 2.3300977 0. 0. 0. 0. 0. !
   ! 1.6754632 - 0.6414398 - 3.7144218 0. 0. 0. 0. !
   ! 0.5574796 - 2.7949591 - 2.6918991 0.8521023 0. 0. 0. !
   ! 0.0038354 - 1.3841104 1.7635608 3.7128833 1.7879881 0. 0. !
   ! 0.7205845 1.7918728 3.969979 1.8386818 - 1.1713759 - 1.479682 0. !
   ! 1.5646425 - 0.4689211 1.6148528 2.3551262 - 0.8607407 - 1.5531762 - 0.3674708 !

   On constate que le tableau précédent est inchangé, et on a juste rajouté une ligne. Cela signifie qu'on ajoute simplement au polynôme d'interpolation précédent le produit $f[x_0, \cdots, x_6]\,\omega_7(x)$.

3. *question 3.3*

   //Fonction ajoute_noeud.sci
   function a = ajoute_noeud(m,x,y)
   n = size(m,"r") ;
   a = zeros(n+1,n+1) ;
   a(1 :n,1 :n) = m ;
   a(n+1,1) = y ;
   for j in 2 :(n+1)
   a(n+1,j) = (a(n+1,j-1)-a(n,j-1))/(x(n+1)-x(n-j+2)) ;
   end

Question 4 *question4.*

On peut démontrer le résultat de la façon suivante :
Si $f \in \mathcal{P}_{n-1}$*, alors* $\Pi(f)$*, le polynôme a priori de degré n interpolant f en* $n+1$ *points est en fait de degré inférieur ou égal à* $n+1$*. Donc, dans la formule* $\Pi f(x) = f[x_0] + \sum_{j=1}^{n} f[x_0, \cdots, x_j](x - x_0)\cdots(x - x_{j-1})$*, le dernier terme de la somme, responsable su terme de degré n, est forcément nul :* $f[x_0, \cdots x_n] = 0$

Question 5 *question5.*

Pour calculer les différences divisées, on tape :
x=linspace(0,0.0002,6)
v=f(x)
diff_div(x',v')
Et on obtient :
! 1. 0. 0. 0. 0. 0. !
! 1.00012 3. 0. 0. 0. 0. !
! 1.00024 2.9999999 - 0.0005399 0. 0. 0. !
! 1.00036 2.9999999 - 0.0010801 - 4.5016075 0. 0. !
! 1.00048 2.9999997 - 0.0016200 - 4.498716 18.071482 0. !
! 1.0006 2.9999996 - 0.0021599 - 4.4998728 - 7.2300383 - 126507.6 !

Au voisinage de 0*, la fonction f est quasi-linéaire : l'équivalent est* $1 + 3x$*. On devrait donc avoir des différences divisées proche de* 0 *à partir de la dérivé seconde. Ici, la dérivé seconde est bien nulle, mais*

les suivantes ne le sont pas. Ceci est dû à la propagation des erreurs lors du calcul des différences divisés dans l'algorithme.

Question 6 *question6.*

1. *question 6.1*

   ```
   //Fonction newton.sci
   function w = newton(x,y,z)
   l = length(z) ;
   n = length(x) ;
   m = diff_div(x,y) ;
   w = (z-(x(n-1)*ones(l,1)))*m(n,n) ;
   for i = (n-1) :-1 :2
   w = (z-(x(i-1)*ones(l,1))).*(w+m(i,i)*ones(l,1)) ;
   end
   w = w + ones(l,1)*m(1,1) ;
   ```

2. *question 6.2*

   Pour comparer les différentes mt́hodes d'interpolation, nous avons tracé les courbes et les polynômes d'interpolation de Vandermonde et de Newton avec 6 points d'interpolation pour 3 fonctions :
   L'important reste de définir deux grilles différentes : une pour l'interpolation et une pour le dessin.
   Pour f(x)=(1+x.ˆ2)")
   x = linspace(-5,5,100) ;
   y = linspace(-5,5,6) : interpolation ;
   *Pour f(x)=sin(2pi*x) x = linspace(-1,1,100) ;*
   y = linspace(-1,1,6) :interpolation ;
   *Pour la fonction f(x)=1+sin(3*x)*
   x = linspace(0,2,100) ;
   y = linspace(0,2,6) : interpolation ;
   La courbe de newton s'obtient en faisant la commande suivante dans chaque cas :
   plot2d(x,newton(y',fonction(y)',x'),3)

   Et on obtient le graphique :

   On constate que les courbes de l'approximation de Newton et de Vandermonde sont confondues.
   Il s'agit donc de deux méthodes donnant un résultat très proche.

Question 7 *question7.*

1. *question 7.1*

   Cette méthode, qui utilise le fait que les différences divisées sont invariantes par permutation, est également valable pour calculer le polynôme d'interpolation dans la base de Newton car seules les différences divisés de la forme $f[x_0 \cdots x_k]$ servent effectivement au calcul de ce polynôme. Ces termes qui se trouvent sur la diagonale, sont identiques à ceux du premier tableau.

2. *question 7.2*
   Cette méthode diminue le temps de calcul du programme car, pour calculer les éléments d'une même colonne de la matrice, on fait toujours appel au même premier élément de la colonne précédente, et à l'élément (variable) de ligne courante de la colonne précédente. Ceci nous permet de gagner un temps précieux en n'utilisant qu'une boucle for.
   Exemple de calcul effectué dans cette méthode :
   $f[x_0, x_1, x_2, x_5] = f[x_2, x_0, x_1, x_5]$
   $= \frac{f[x_0,x_1,x_5]-f[x_2,x_0,x_1]}{x_5-x_2}$
   $= \frac{f[x_0,x_1,x_5]-f[x_0,x_1,x_2]}{x_5-x_2}$

3. *question 7.3*

   Nouvelle méthode :
   //Fonction diff_div_bis.sci
   function m = diff_div_bis(x,y)
   n = length(x) ;
   m = zeros(n,n) ;
   m(:,1) = y ;
   for j = 2 :n
   *m(j :n,j) = (m(j :n,j-1)-(m(j-1,j-1)*ones(n-j+1,1))./(x(j :n)-x(j-1)*ones(n-j+1,1)) ;*
   end

   On peut vérifier que l'on obtient bien la même diagonale qu'au premier tableau de différences divisées de la question 2 :
   diff_div_bis(x,y)
   ! 2. 0. 0. 0. 0. !
   ! 0. - 1. 0. 0. 0. !
   ! 2. 0. 1. 0. 0. !
   ! - 1. - 0.6 0.1333333 - 0.4333333 0. !
   ! 1. - 0.1428571 0.1714286 - 0.2071429 0.1130952 !