MILHAUD Xavier
VOISIN Basile

TP 2 : Interpolation, matrices de Vandermonde

Question 1

1. *question 1.1*

   La fonction de Vandermonde nous permet donc de retourner les coefficients nécessaires à l'interpolation.

2. *question 1.2*

   Fonction calculant a sans utiliser de boucles :
   //Fonction vandermondebis.sci
   function a = vandermondebis(x,y)
   n = length(x) ;
   *V = (x*ones(1,n)).^(ones(n,1)*[0 :n-1]) ;*
   a = V\y ;

   *En effet, pour le calcul de la matrice de Vandermonde, x*ones(1,n) permet de générer une matrice carré, dont toutes les colonnes sont égales au vecteur x ; et ones(n,1)*[0 :n-1] permet de générer une matrice dont les lignes sont toutes [0 :n-1] . Ainsi en élevant les termes de la première matrice aux puissances correspondant aux éléments de la seconde, nous obtenons la matrice de Vandermonde.*

   Pour l'efficacité de la fonction vandermondebis par rapport à la fonction vandermonde, on obtient, à l'aide de tic et toc et en testant sur la fonction sinus :
   temps d'execution de vandermonde : 0,032 s
   temps d'execution de vandermondebis : 0,003 s
   La nouvelle fonction est environ 10 fois plus rapide que l'ancienne.

3. *question 1.3*

   Cette dernière version de vandermonde calcule la matrice de Vandermonde à l'aide d'une boucle for permettant de calculer la i^eme colonne en multipliant terme à terme la i-1^eme colonne par x .
   Commentaires : pour l'efficacité de la fonction vandermondetierce par rapport à la fonction vandermondebis, on obtient, à l'aide de tic et toc et en testant sur la fonction sinus :
   temps d'execution de vandermondebis : 0,004 s
   temps d'execution de vandermondetierce : 0,005 s
   Le temps d'execution de cette dernière fonction est sensiblement superieur à celui de vandermondebis. Il reste donc raisonnable par rapport à celui de la première version de vandermonde.

Question 2

1. *question 2.1*

   La fonction proposée d'évaluation d'un polynôme étant donnés un réel z et un vecteur a calcule l'expression demandée à l'aide d'une boucle :
   //Fonction sompoly.sci
   function s = sompoly(z,a)

n = length(a) ;
s = a(1) ;
for i = 2 :n
s = s + a(i)(z^(i-1)) ;*
end

2. *question 2.2*

   On applique le même raisonnement qu'à la question 1.2 :
   //Fonction sompolybis.sci
   function s = sompolybis(z,a)
   n = length(a) ;
   *s = ((z*ones(1,n)).^[0 :n-1])*a ;*
   En n'utilisant pas de boucle for on permet à notre programme d'aller plus vite, donc de gagner du temps dans la recherche du résultat.

3. *question 2.3*

   En utilisant le schéma de Horner, on obtient le programme suivant :
   //Fonction sompolybis.sci
   function s = sompolybis(z,a)
   n = length(a) ;
   *s = ((z*ones(1,n)).^[0 :n-1])*a ;*

   Le prinicipe consiste à factoriser, donc on crée une sorte d'encapsulation. Le phénomène et donc la forme finale peut se décomposer facilement.

4. *question 2.4*

   La comparaison de ces trois fonctions en prenant a = linspace(0,99999,77777)' et $z = \sqrt{2}$*, on obtient avec tic et toc :*
   pour sompoly, le temps d'execution est : 1,289 s
   pour sompolybis, le temps d'execution est : 0,028 s
   pour sompolyhorner, le temps d'execution est : 0,887 s
   La meilleure façon de calculer cette somme est donc la forme sans boucle utilisée dans le second algorithme, ce qui est un résultat logique puisque l'on sait que les boucles coutent du temps dans un calcul.

5. *question 2.5*

   On généralise la fonction de la question 2.2 en créant le vecteur de la façon suivante :
   //Fonction sompolygen.sci
   function v = sompolygen(z,a)
   n = length(a) ;
   m = length(z) ;
   *v = ((z*ones(1,n)).^(ones(m,1)*[0 :n-1]))*a ;*
   Le but est donc de pouvoir se placer en plusieurs points représentés par le vecteur.

Question 3 *question3*

La fonction affiche doit prendre en compte deux grilles différentes : une grille d'interpolation et une grille de découpage afin de représenter sur notre graphe nos résultats. On donne également un titre au

graphe par xtitle. Elle utilise donc la grille de découpage pour afficher la fonction à interpoler et l'autre grille nécessaire à l'interpolation elle-meme.
x = linspace(a,b,1000)' :grille de découpage : valeur constante de n=1000.
y = linspace(a,b,n)' : grille d'interpolation.
On utilise la commande z = sompolygen(x,vandermondebis(y,f(y))) pour nous retourner le polynôme d'interpolation associé à cette fonction.On utilise donc la plus rapide des méthodes pour Vandermonde.

Question 4

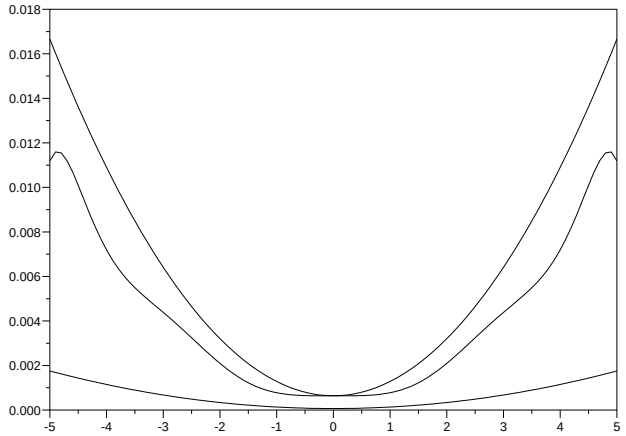
1. *question 4.1*

   Pour tracer f sur $[-5, 5]$*, on tape les commandes :*
   deff("y=f(x)","y=(1)./(1+x^2)") : sert à définir une fonction.
   x=linspace(-5,5,100) : grille de 100 points équirépartis sur $[-5, 5]$

2. *question 4.2*

   Pour tracer p_5 *et* $p_1 1$*, on définit les grilles d'interpolation :*
   y=linspace(-5,5,6) : 6 points équirépartis .
   z=linspace(-5,5,12) : 12 points équirépartis.
   Et on trace les courbes :
   plot2d(x,sompolygen(x',vandermondebis(y',f(y)')))
   plot2d(x,sompolygen(x',vandermondebis(z',f(z)')))

3. *question 4.3*



On constate qu'ici, l'approximation par les polynômes interpolateurs n'est pas très bonne à cause des oscillation de celui-ci. D'ailleurs plus le degré est grand et plus les oscillations sont imprécises autour de la fonction à interpoler. On en conclut que l'augmentation du degré ne donne pas forcément une plus grande précision.

Question 5 *question5*

Affichage de $f(x) = \sin(2\pi x)$ *:*
x=linspace(-1,1,100) : grille de 100 points équirépartis entre $[-1, 1]$ *;*
Affichage du polynôme d'interpolation aux 17 points :
y=linspace(-1,1,17) :grille de 17 points équirépartis entre $[-1, 1]$

plot2d(x,sompolygen(x',vandermondebis(y',f(y)')),2)
Affichage du polynôme d'interpolation perturbé :
*plot2d(x,sompolygen(x',vandermondebis(y',f(y)'+(rand(17,1)*9.5*10^(-4)))),3)*

On constate que pour cette fonction, l'approximation par le polynôme d'interpolation est excellente (les deux courbes se superposent). En outre, si on perturbe légèrement les valeurs d'interpolation, cette approximation reste très bonne.