Compte rendu TP d'interpolation numérique

Mohammed Amine Kaabachi

14/01/2016

1 Interpolation de Lagrange

1.1 Implémentation avec MATLAB

L'interpolation de Lagrange en MATLAB est réalisée en utilisant la fonction polyfit(x,y,n) qui retourne les coefficients du polynôme d'interpolation p_n des données du vecteur y aux points de x.

Dans la majorité des applications, on ne cherche pas les coefficients du polynôme d'interpolation mais plutôt ses valeurs sur un intervalle I. On a réalisé cette petite fonction qui nous facilitera la vie dans les prochaines applications.

```
function px = Lagrange(xi, yi, I)
    p = polyfit(xi,yi,numel(xi)-1);
    px = polyval(p,I);
```

1.2 Les Masses volumiques pour différentes températures

Étant donné le tableau ci-dessous contenant masses volumiques du matériau pour différentes températures. On veut extraire le polynôme d'interpolation de Lagrange et y exhiber quelques valeurs.

Température	94	205	371
Masse volumique	929	902	860

Les coefficients du polynôme sont facilement retrouvables à l'aide des commandes suivantes :

```
xi = [94 205 371];
yi = [929 902 860];
p = polyfit(xi,yi,numel(xi)-1)
```

Le résultat de ces commandes nous donne que $p_2(x) = -0.0892x + 17.0901$. Si on dessine la courbe de ce polynôme sur l'intervalle $[100, 1000]$, on aura la figure 1.

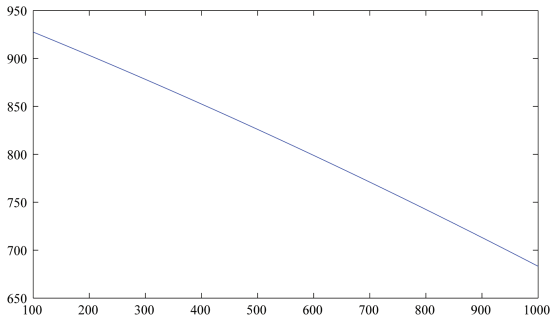

Figure 1: Courbe du polynôme $p_2(x) = -0.0892x + 17.0901$

Enfin, pour déterminer les valeurs de ce polynôme pour un certain nombre de points, on utilise la fonction Lagrange de la manière suivante:

```
xm = [251 305 800];
ym = Lagrange(xi, yi, xm)
```

Les résultats sont donnés par le tableau suivant :

Température	251	305	800
Masse volumique	890.5561	876.9316	742.4559

1.3 Interpolation de la fonction exponentiel

On veut interpoler la fonction $f(x) = e^x$ sur l'intervalle $[-1, 1]$ en n points équidistants pour $n = 2, ..., 8$. Par suite, on veut tracer la fonction considérée et les polynômes d'interpolation sur le même graphique. En utilisant la fonction Lagrange, on propose le code suivant :

```
clr = lines(14);
I = -1:1/50:1;
y = exp(I);
plot(I, y);
hold on;
for n=2:8
        xi = -1:2/n:1;
        yi = exp(xi);
        p = Lagrange(xi, yi, I);
        hold on;
        plot(I,p,'Color',clr(n,:));
 end
```

Les tracés de ces polynômes et de la fonction f sur l'intervalle [-1, 1] et sur un voisinage de 0 sont illustrés sur la figures suivantes .

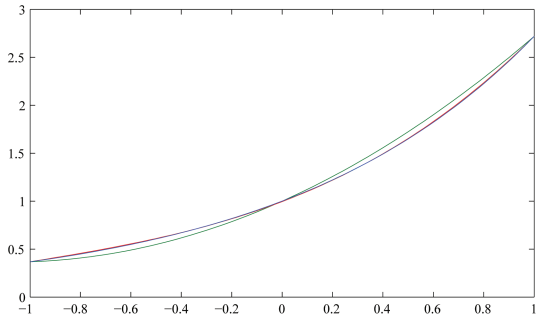

Figure 2: Tracés sur [-1, 1]

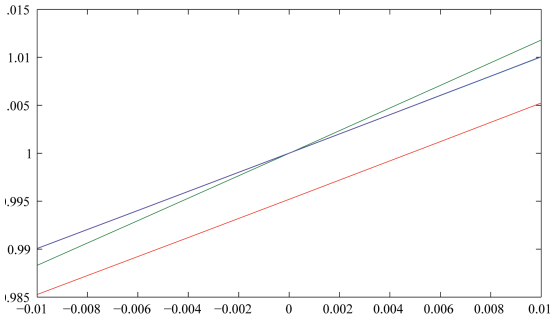

Figure 3: Tracés sur un voisinage de 0

Pour avoir une idée claire sur les limites de cette interpolation, on peut tracer l'erreur du polynôme pour n = 8 et la fonction f. Ce tracé est donnée par la figure 4.

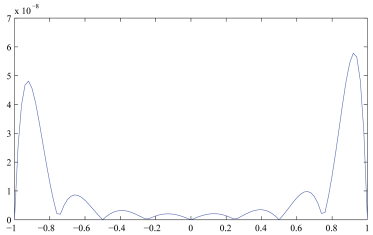

Figure 4: Erreur entre $p_8 et f$

On peut refaire l'étude précédente avec les points de Tchebychev simplement en ajoutant cette ligne après la déclaration de la liste xi :

```
xi = cos(2*(xi-1)*pi/n);
```

Les tracés de ces polynômes et de la fonction f sur l'intervalle [-1, 1] et sur un voisinage de 0 avec les points de Tchebychev deviennent alors :

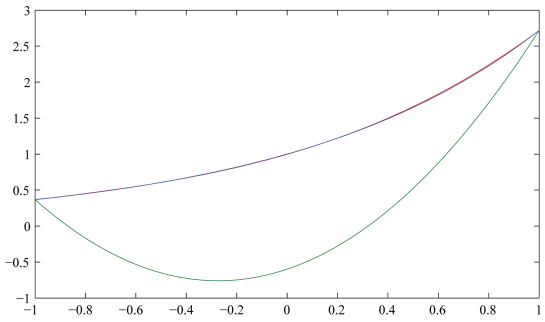

Figure 5: Tracés sur [-1, 1]

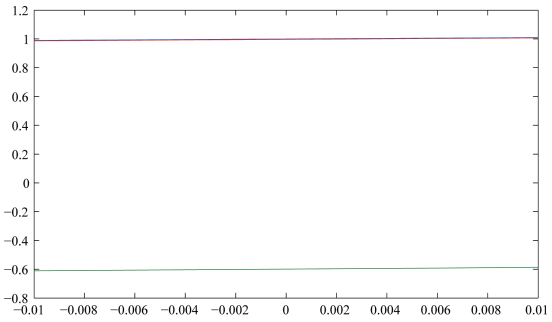

Figure 6: Tracés sur un voisinage de 0

Concernant l'erreur du polynôme pour n = 8 et la fonction f, l'erreur est minimale. La figure suivante montre comment elle approche 0;

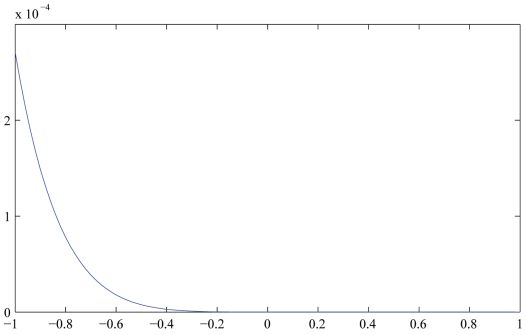

Figure 7: Erreur entre $p_8 et f$

1.4 Phénomène de Runge

Dans le cadre de l'interpolation de Lagrange de certaines fonctions, l'augmentation du nombre n de points d'interpolation ne constitue pas nécessairement une bonne stratégie d'approximation.

En étudiant cette question, le mathématicien allemand Carl Runge découvrit, en 1901, un résultat contraire à l'intuition : il existe des configurations où l'écart maximal entre la fonction et son interpolation augmente indéfiniment avec n.

On considère la fonction de test définie par :

$$f(x) = \frac{1}{25x^2 + 1}$$

On veut étudier pour n = 6, n = 10 et n = 14, le polynôme d'interpolation de degré n de la fonction f; On propose le code suivant :

```
clr = lines(14);
I = -1:1/50:1;
```

```
y =  1./(25.*(I.^2)+1);
plot(I,y);
for n = [6 10 14]
        xi = -1:2/n:1;
        yi = 1./(25.*(xi.^2)+1);
        p = Lagrange(xi, yi, I);
        hold on;
        plot(I,p,'Color',clr(n,:));
end
```

La figure représentative de cette étude est la suivante :

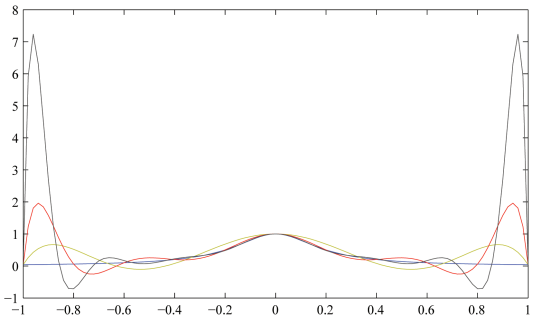

Figure 8: Phénomène de Runge

Si on interpolent avec les points de Tchebychev, on peut minimiser l'oscillation des polynômes interpolateurs. Ce résultat est illustré par la figure suivante :

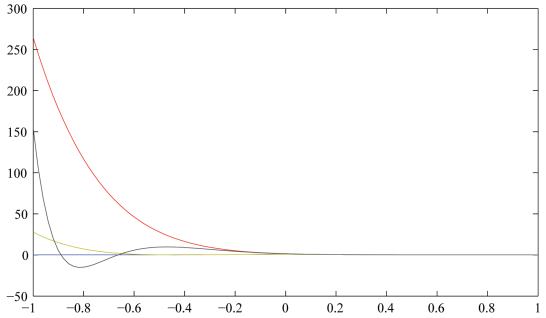

Figure 9: Phénomène de Runge avec les points de Tchebychev

1.5 Interface graphique

On a implémenté à l'aide de Matlab une interface graphique permettant l'affichage et l'interaction avec les figures présentées sur ce compte rendu. Les fichiers .fig et .m sont inclus avec ce pdf.

2 Interpolation par morceaux

2.1 Implémentation avec Matlab

Pour réaliser des interpolations par morceaux avec Matlab, on utilise les fonctions interp1, pchip et spline; ils permettent de faire des interpolations linéaires, cubiques et en splines cubiques.

On peut aussi n'utiliser que la fonction interp1 en spécifiant la méthode 'spline' ou 'cubic'.

2.2 Interpolation linéaire par morceaux

Etant donnée une distribution de nœuds $x_0 < x_1 < \cdots < x_n$. On note I_i l'intervalle $[x_i, x_{i+1}]$ avec $i = 0, ..., n-1$.

On approche f par une fonction continue qui, sur chaque intervalle Ii est définie par le segment joignant les deux points $(x_i, f(x_i))$ et $(x_{i+1}, f(x_{i+1}))$.

Soit intervalle Ii, on veut déterminer l'expression de la fonction interpolatrice $\tilde{f}$ sur ce dernier. On pose $\tilde{f}(x) = ax + b$, $\tilde{f}(x_i) = f(x_i) = y_i$ et $\tilde{f}(x_{i+1}) = f(x_{i+1}) = y_{i+1}$.

Par identification avec les points précédents, on aura le système suivant :

$$\begin{cases} yi \ = \ ax_i + b \\ y_{i+1}{=}ax_{i+1} + b \end{cases}$$

La résolution de ce système a deux équations nous donne :

$$\begin{cases} a{=} \ \frac{y_{i+1}-y_i}{x_{i+1}-x_i} \\ b{=}\frac{y_i x_{i+1}-y_{i+1}x_i}{x_{i+1}-x_i} \end{cases}$$

Par suite, sur chaque intervalle Ii, $\tilde{f}(x) = \frac{y_{i+1}-y_i}{x_{i+1}-x_i}x + \frac{y_i x_{i+1}-y_{i+1}x_i}{x_{i+1}-x_i}$.

On peut aussi trancher sur l'erreur d'interpolation au pire de cas. En particulier, si la fonction f que l'on souhaite interpoler est deux fois continûment dérivable, une majoration de l'erreur existe.

En effet, on sait déjà montrer que si f est n fois continûment dérivable

$$\exists \zeta \in]x_i, x_{i+1}[, f(x) - P(x) = \frac{\prod_1^n (x - x_i)}{n!} f''(\zeta)$$

Pour n = 2 et$P = \tilde{f}$,

$$\exists \zeta \in]x_i, x_{i+1}[, |f(x) - \tilde{f}(x)| \leq \frac{(x - x_i)(x_{i+1} - x)}{2} |f''(\zeta)|$$

En notant $M = sup_{y \in [x_i, x_{i+1}]} |f''(y)|$, on aura :

$$\forall x \in [x_i, x_{i+1}], |f(x) - \tilde{f}(x)| \leq \frac{M}{8}(x_{i+1} - x_i)^2$$

car $(x - x_i)(x_{i+1} - x) \leq \frac{(x_{i+1}-x_i)^2}{2}$.

Comme exemple, soit la fonction f définie par $f(x) = \frac{x^2+10}{\sin x+1.2}$. On veut interpoler cette fonction linéairement par morceaux suivant la distribution -2, 0, 2, 4, 6, 8. On propose ce code :

```
x = -2:1/1000:8;
f = (x.^2 + 10) ./ (sin(x)+1.2)
plot(x,f);
hold on;
pts = [-2 0 2 4 6 8];
ft = interp1(x,f,pts);
plot(pts,ft);
```

On obtient alors la figure suivante :

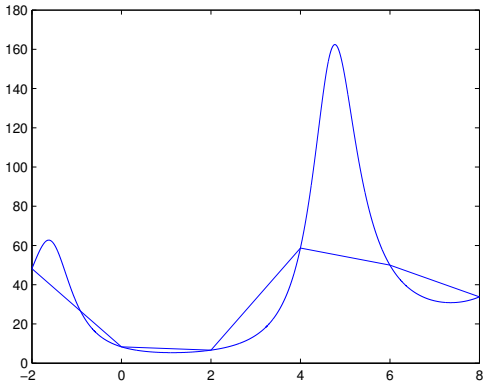

Figure 10: Interpolation linéaire par morceaux de f

2.3 Interpolation cubique par morceaux et splines cubiques

Dans un premier exemple, on dispose du tableau suivant :

i	1	2	3	4	5	6	7
x	-3	-2	-1	0	1	2	3
y	-1	-1	-1	0	1	1	1

On veut calculer et tracer les polynômes cubiques et la spline cubique interpolant les données de ce tableau. On a réalisé le code suivant :

```
xi = -3:1/1000:3;
x = -3:1:3;
y = [-1 -1 -1 0 1 1 1];
cu = pchip(x,y,xi); % interpolation cubique
sp = spline(x,y,xi);
plot(xi, cu, '', xi, sp, '--');
```

On obtient comme résultat la figure 11.

Dans le second exemple, les données suivantes correspondent, en millions, à la taille d'une population entre l'année 1900 et l'année 1990.

Année	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990
Population	75.995	91.972	105.711	123.203	131.669	150.697	179.323	203.212	226.505	249.633

Pour estimer la population en 1975 en interpolant par splines cubiques, on utilise les commandes suivantes :

```
a = 1900:10:1990;
p = [75.995 91.972 105.711 123.203 131.669 150.697 179.323 203.212 226.505 249.633];
sp = spline(a,p,1970)
```

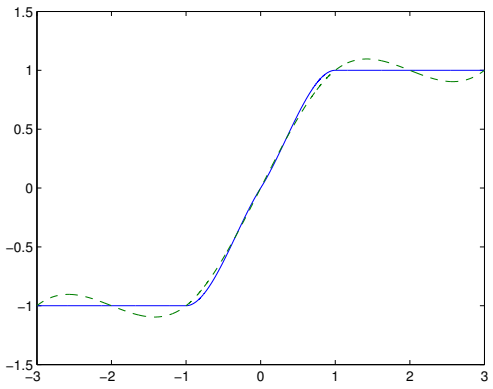

Figure 11: Interpolation cubique par morceaux vs spline cubique

Ceçi nous donne que la population en 1975 est égale à 203.212 . On veut tracer la taille de la population en chaque année de 1900 à 1990, on développe alors le code çi-dessous :

```
a = 1900:10:1990;
p = [75.995 91.972 105.711 123.203 131.669 150.697 179.323 203.212 226.505 249.633];
sp = spline(a,p,1970)
xi = 1900:1:1990;
y = spline(a,p,xi);
plot(a,p,'o', xi,y, '-');
```

Et finalement, on obtient la figure suivante :

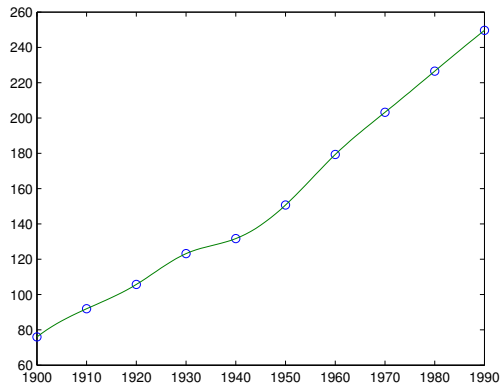

Figure 12: Taille de la population en chaque année