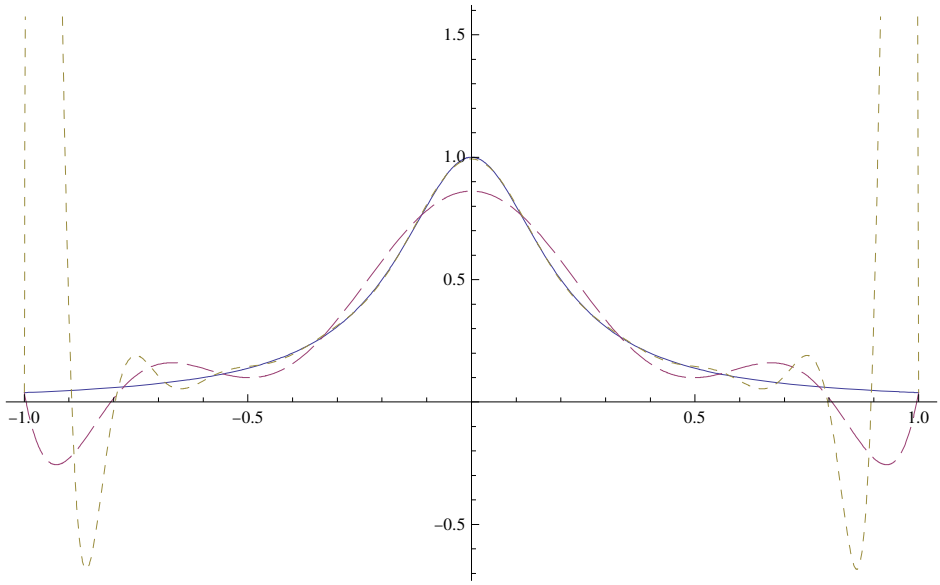
Collège du Sud, Bulle
2-ème année OS PAM
3-ème année OC AM

Applications des mathématiques

Interpolation



Edition 2018-2019
d'après Marcel Délèze et Eugène Pasquier
http://applmaths.collegedusud.ch/

§ 0 Introduction

Soit n points

$$(x_0,\ y_0),\ (x_1,\ y_1),\ \ldots,\ (x_{n-1},\ y_{n-1})$$

d'abscisses distinctes avec $x_0 < x_1 < \ldots < x_{n-1}$. Le but de l'interpolation est de trouver une fonction $g : [x_0,\ x_n] \longrightarrow \mathbb{R}$ dont le graphe passe par tous ces points, c'est-à-dire avec

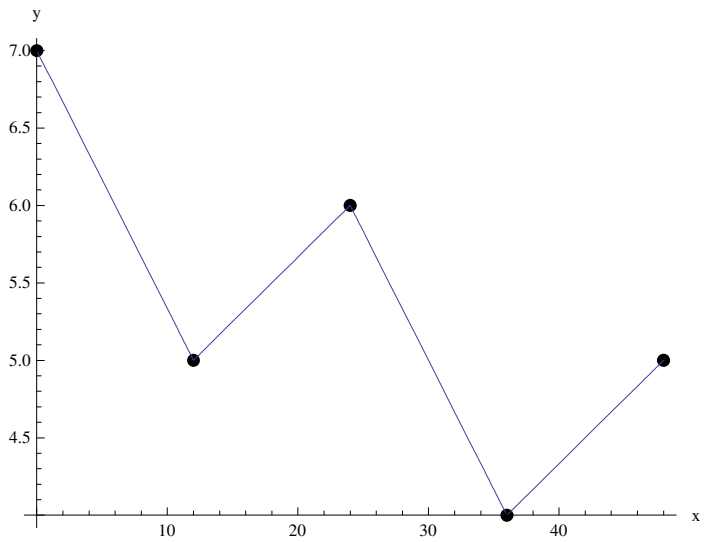
$$g(x_0) = y_0, \quad g(x_1) = y_1, \quad \ldots, \quad g(x_{n-1}) = y_{n-1}.$$

Un tel problème est appelé un *problème d'interpolation*, les points sont appelés *points d'interpolation* et la fonction recherchée un *interpolant* (ou une *fonction d'interpolation*).
Le choix du type d'interpolant varie en fonction du contexte dans lequel se pose le problème d'interpolation. Nous allons voir quelques possibilités dans un cas concret. Nous considérons les 4 points d'interpolation donnés dans le tableau suivant :

x	y
0	7
12	5
24	6
36	4
48	5

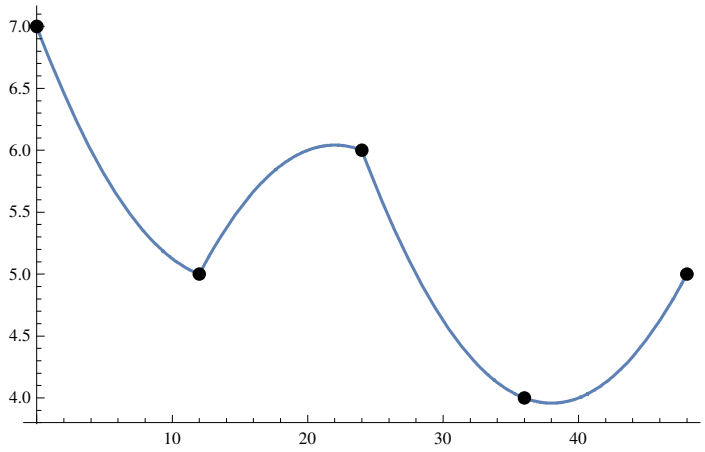
Nous voulons évaluer l'interpolant de ce problème d'interpolation en 42 : nous disons que nous voulons *interpoler* en 42.

La méthode la plus simple consiste à faire une interpolation linéaire. Le graphe de l'interpolant sera alors une ligne brisée :



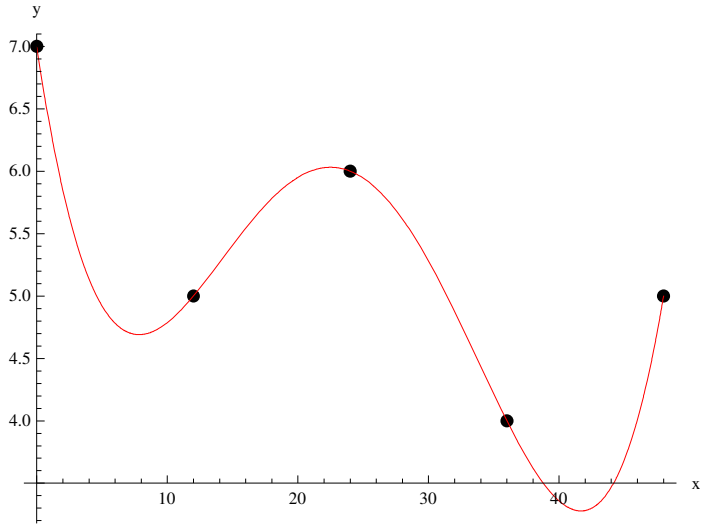
L'évaluation de l'interpolant en 42 donne 4.5.

Une autre possibilité est de faire une interpolation quadratique. Le graphe de l'interpolant est alors constitué d'arcs de parabole (chaque parabole passerait par trois points consécutifs):

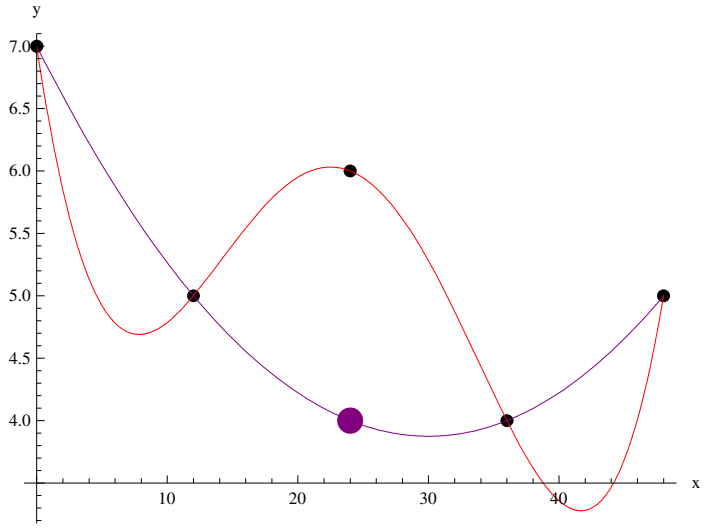


L'évaluation de l'interpolant en 42 donne 4.125. Cette façon de procéder n'est pas satisfaisante car bien que plus compliquée que la précédente, le résultat donne une courbe qui n'est pas lisse. Dans l'idéal, il faudrait "coller de façon lisse" un arc de parabole que passe les troisième et quatrième points ainsi qu'un dernier arc qui passe par les quatrième et le cinquième points (ceci est possible grâce aux calculs des dérivées).

Une dernière possibilité qui utilise des polynômes est de rechercher le polynôme qui passe par les cinq points d'interpolation. Nous obtenons alors le graphe suivant:

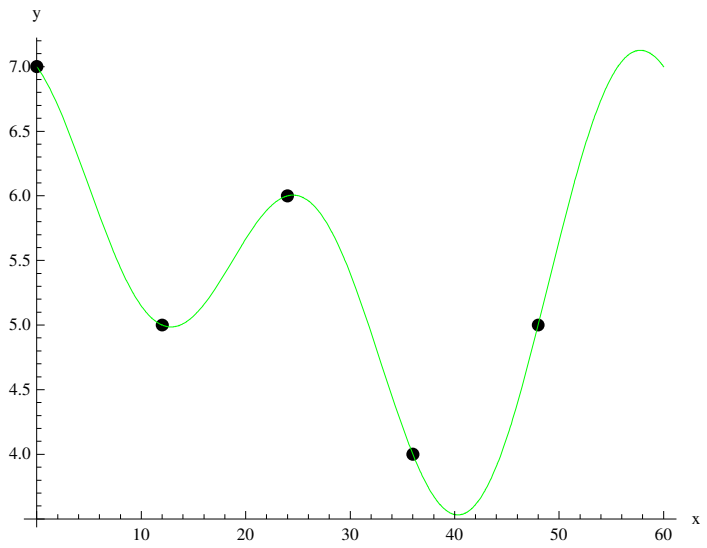


L'évaluation de l'interpolant en 42 donne approximativement 3.28. Cette méthode semble plus satisfaisante que les deux précédentes dans la mesure où le graphe de l'interpolant donne une courbe lisse. Cependant la faiblesse de cette méthode est sa sensibilité aux points d'interpolation : le changement d'un point modifie totalement l'interpolant (ce qui n'était pas le cas dans les autres exemples). Si dans la liste des points d'interpolation le point (24; 6) est remplacé par le point (24; 4) le résultat est très différent:

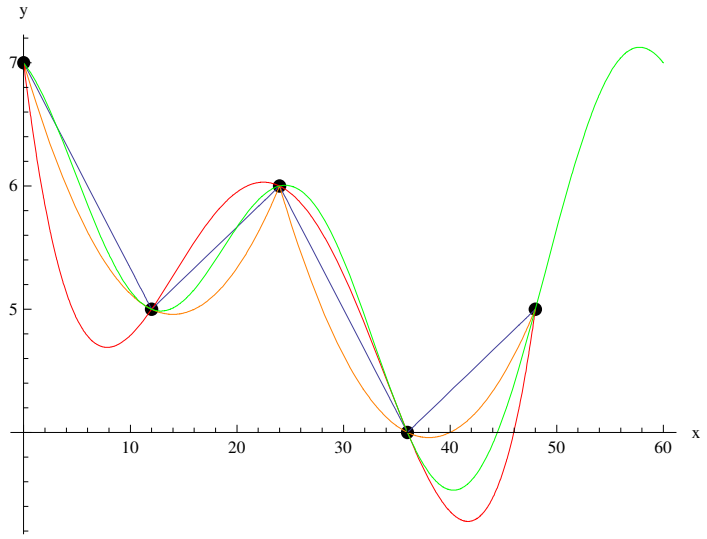


L'évaluation de l'interpolant en 42 avec cet autre point donne 4.375!

Il se peut que le phénomène étudié soit periodique. Dans ce cas, nous cherchons un interpolant qui est lui aussi périodique. Voici le graphe que nous pourrions avec un interpolant de période 60:



Finalement nous constatons que le choix de l'interpolant varie passablement en fonction des contraintes que nous nous imposons:



§ 1 Calculs d'interpolant

Nous allons voir dans cette section la façon dont se calcule l'interpolant. Nous traiterons d'abord les cas où l'interpolant est construit à partir de polynômes, puis le cas où l'interpolant utilise des fonctions trigonométriques et finalement le cas général.

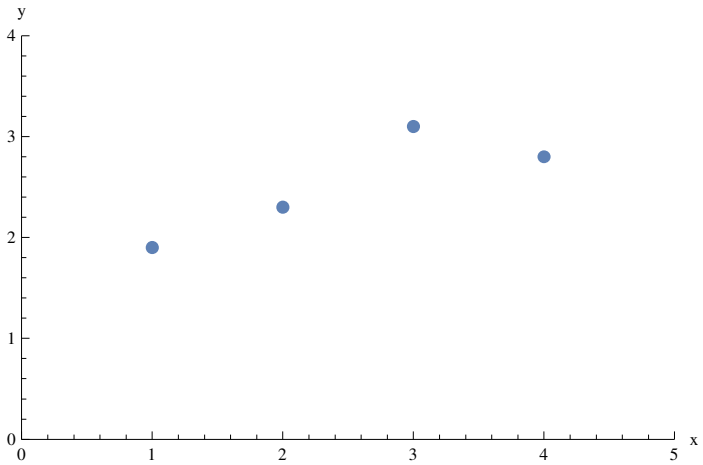
§ 1.1 Interpolation linéaire

Nous considérons les points d'interpolation donnés dans le tableau suivant:

x	1	2	3	4
y	1.9	2.3	3.1	2.8

Nous voulons interpoler en $a = 2.7$.
Graphiquement, la situation se présente comme suit

```
pts = {{1, 1.9}, {2, 2.3}, {3, 3.1}, {4, 2.8}};
abs = Transpose[pts][[1]]; (* abscisses d'interpolation *)
      transposée
ord = Transpose[pts][[2]]; (* ordonnées d'interpolation *)
      transposée
a = 2.7; (* abscisse à interpoler *)
ListPlot[pts, PlotRange → {{0, 5}, {0, 4}}, PlotStyle → PointSize[0.02], AxesLabel → {"x", "y"}]
tracé de liste  zone de tracé  style de tracé  taille des points  titre d'axe
```



Nous avons vu que la méthode la plus simple consiste à effectuer une interpolation linéaire, c'est-à-dire à joindre les points par des segments de droite. Comme nous voulons interpoler en $a = 2.7$, il faut trouver l'équation du segment de droite g passant par les points suivants (2; 2.3) et (3; 3.1). La fonction g est affine donc

```
g[t_] := c0 + c1 t
```

Comme g doit passer par les deuxième et troisième points d'interpolation nous avons les équations suivantes :

```
eqns = Table[g[ abs[[i]] ] == ord[[i]], {i, 2, 3}]
       table
```

$\{c_0 + 2\,c_1 == 2.3,\ c_0 + 3\,c_1 == 3.1\}$

Mathematica nous donne alors les solutions suivantes :

```
sol = Solve[eqns, {c0, c1}]
      résous
```

$\{\{c_0 \to 0.7,\ c_1 \to 0.8\}\}$

Nous pouvons alors interpoler en a :

```
g[a] /. sol[[1]]
```

2.86

Mathematica permet cependant d'avoir directement l'interpolant linéaire grâce à la fonction **Interpolation** en précisant l'ordre de l'interpolant:

```
? Interpolation
```

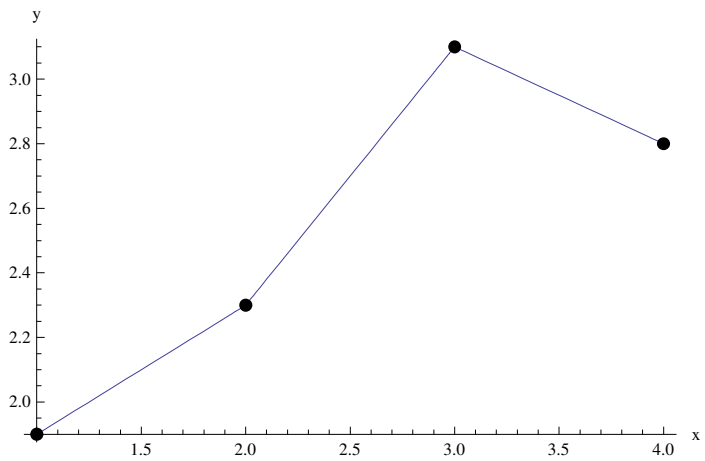
Interpolation[{f_1, f_2, …}] constructs an interpolation of the function values f_i, assumed to correspond to x values 1, 2, … .
Interpolation[{{x_1, f_1}, {x_2, f_2}, …}] constructs an interpolation of the function values f_i corresponding to x values x_i.
Interpolation[{{{x_1, y_1, …}, f_1}, {{x_2, y_2, …}, f_2}, …}] constructs an interpolation of multidimensional data.
Interpolation[{{{x_1, …}, f_1, df_1, …}, …}] constructs an interpolation that reproduces derivatives as well as function values.
Interpolation[*data*, x] find an interpolation of *data* at the point x. >>

```
Options[Interpolation]
options  interpolation
```

```
{InterpolationOrder → 3, Method → Automatic, PeriodicInterpolation → False}
```

Nous constatons que par défaut *Mathematica* fait une interpolation avec des polynômes de degré 3 or nous désirons une interpolation linéaire, c'est-à-dire des polynômes de degré 1 :

```
Clear[g]
efface
g[x_] := Interpolation[pts, x, InterpolationOrder → 1]
         interpolation          ordre d'interpolation
Plot[
tracé de courbes
 g[x], {x, Min[abs], Max[abs]},
           minimum    maximum
 Epilog → {PointSize[0.02], Point[pts]},
           taille des points  point
 AxesLabel → {"x", "y"}
 titre d'axe
]
```



Il possible d'interpoler directement en a sans passer par la définition de l'interpolant:

```
Interpolation[pts, a, InterpolationOrder → 1]
interpolation          ordre d'interpolation
```

```
2.86
```

§ 1.2 Interpolation polynomiale

Nous considérons le même problème d'interpolation qu'au point précédent, cependant, nous voulons cette fois faire une *interpolation polynomiale*, c'est-à-dire à faire passer par les 4 points donnés un polynôme. Nous allons, à nouveau, dans un premier temps, ne pas utiliser les méthodes d'interpolation proposées par *Mathematica*. Pour trouver le polynôme g, nous aurons 4 équations et par conséquent le polynôme recherché sera de degré 3 et sera de la forme

$$g(t) = c_o + c_1\, t + c_2\, t^2 + c_3\, t^3.$$

En fait, g est une combinaison linéaire des fonctions suivantes appelées *fonctions de base* : $b_0(t) = 1$, $b_1(t) = t$, $b_2(t) = t^2$, $b_3(t) = t^3$. La fonction f a alors la forme suivante (forme que nous utiliserons dans la prochaine section pour généraliser le problème) :

$$g(t) = c_o\, b_0(t) + c_1\, b_1(t) + c_2\, b_2(t) + c_3\, b_3(t).$$

Il faut déterminer la valeur des nombre c_0, c_1, c_2, c_3 tel que l'interpolant g passe par les quatre points donnés donc

$$g(1) = 1.9, \qquad g(2) = 2.3, \qquad g(3) = 3.1\ , \qquad g(4) = 2.8.$$

Ceci nous donne les quatre équations suivantes:

$$c_o + c_1\,1 + c_2\,1^2 + c_3\,1^3 = 1.9\,, \qquad c_o + c_1\,2 + c_2\,2^2 + c_3\,2^3 = 2.3,$$
$$c_o + c_1\,3 + c_2\,3^2 + c_3\,3^3 = 3.1\ \ , \qquad c_o + c_1\,4 + c_2\,4^2 + c_3\,4^3 = 2.8.$$

Ces conditions constituent un système d'équations linéaires. Pour que *Mathematica* puisse résoudre le plus efficacement possible ce système, nous le mettons sous forme matricielle :

$$\begin{pmatrix} 1 & 1 & 1^2 & 1^3 \\ 1 & 2 & 2^2 & 2^3 \\ 1 & 3 & 3^2 & 3^3 \\ 1 & 4 & 4^2 & 4^3 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 1.9 \\ 2.3 \\ 3.1 \\ 2.8 \end{pmatrix}$$

Nous constatons que chaque colonne de la matrice constitue l'évaluation de chacune des fonction de base aux abscisses d'interpolation. Cette constatation permet de facilement générer la matrice du système linéaire :

```
Clear["Global`*"]
efface
b[t_] = {1, t, t^2, t^3}; (* liste des fonctions de base *)
m = Map[b, abs];
    applique
MatrixForm[m]
apparence matricielle
```

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{pmatrix}$$

Nous pouvons alors utiliser **LinearSolve** qui est la fonction de *Mathematica* spécialisée dans la résolution de systèmes linéaires :

```
c = LinearSolve[m, ord]
    résous équation linéaire
```

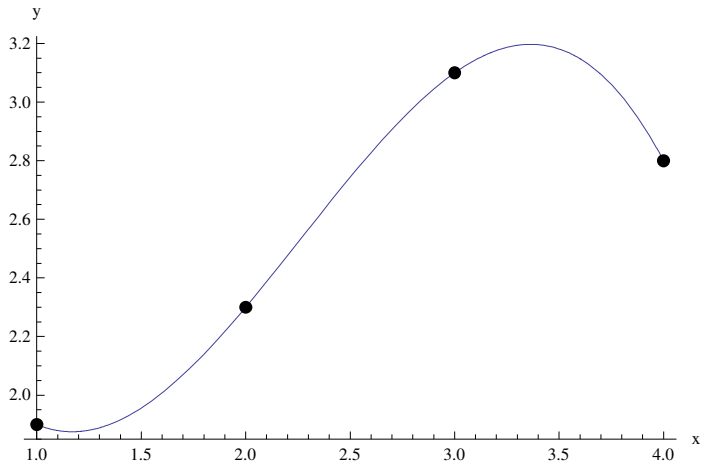
{3.4, -2.95, 1.7, -0.25}

Nous obtenons alors l'interpolant g en faisant le produit scalaire de **c** et **b[t]** :

```
g[t_] := c.b[t]
g[t]
```

$3.4 - 2.95\,t + 1.7\,t^2 - 0.25\,t^3$

```
Plot[
tracé de courbes
 g[x], {x, Min[abs], Max[abs]},
            minimum     maximum
 Epilog → {PointSize[0.02], Point[pts]},
            taille des points     point
 AxesLabel → {"x", "y"}
 titre d'axe
]
```



Nous pouvons maintenant interpoler en a :

```
g[a]
```

```
2.90725
```

Mathematica peut nous donner directement l'interpolant polynomial grâce à la fonction **Interpolatingpolynomial**:

```
? InterpolatingPolynomial
```

InterpolatingPolynomial[{f_1, f_2, …}, x] constructs an interpolating polynomial in x which reproduces the function values f_i at successive integer values 1, 2, … of x.
InterpolatingPolynomial[{{x_1, f_1}, {x_2, f_2}, …}, x] constructs an interpolating polynomial for the function values f_i corresponding to x values x_i.
InterpolatingPolynomial[{{{x_1, y_1, …}, f_1}, {{x_2, y_2, …}, f_2}, …}, {x, y, …}] constructs a multidimensional interpolating polynomial in the variables x, y, ….
InterpolatingPolynomial[{{{x_1, …}, f_1, df_1, …}, …}, {x, …}] constructs an interpolating polynomial that reproduces derivatives as well as function values. >>

```
g[t_] := InterpolatingPolynomial[pts, t];
```
polynôme d'interpolation

```
Expand[g[t]]
```
développe

$3.4 - 2.95\,t + 1.7\,t^2 - 0.25\,t^3$

Il possible d'interpoler directement en a sans passer par la définition de l'interpolant:

```
InterpolatingPolynomial[pts, a]
```
polynôme d'interpolation

```
2.90725
```

§ 1.3 Interpolation polynomiale : cas général

Nous allons maintenant traiter l'interpolation polynomial de façon générale.

Énoncé du problème d'interpolation polynomiale

Sont donnés n points

$(x_0,\ y_0),\ (x_1,\ y_1),\ \ldots,\ (x_{n-1},\ y_{n-1})$ d'abscisses distinctes $x_0 < x_1 < \ldots < x_{n-1}$.

Déterminez le polynôme g de degré $\leq (n-1)$ passant par ces points, c'est-à-dire

$g(x_0) = y_0,\quad g(x_1) = y_1,\quad \ldots,\quad g(x_{n-1}) = y_{n-1}.$

Proposition

Le problème d'interpolation polynomiale possède une et une seule solution g.

Démonstration de l'existence du polynôme d'interpolation *g*

Voir l'exercice 1-4.

Démonstration de l'unicité du polynôme d'interpolation *g*

Supposons qu'il existe deux polynômes d'interpolation g_1 et g_2 qui sont de degré inférieur ou égal à $n-1$. Nous avons donc $g_1(x_i) = g_2(x_i) = y_i$, $i = 0, 1, \ldots, n-1$.
Définissons $d(x) = g_2(x) - g_1(x)$. d est un polynôme de degré inférieur ou égal à $n-1$ qui s'annule en n abscisses distinctes :

$$d(x_0) = g_2(x_0) - g_1(x_0) = y_0 - y_0 = 0,$$
$$d(x_1) = g_2(x_1) - g_1(x_1) = y_1 - y_1 = 0,$$
$$\ldots$$
$$d(x_{n-1}) = g_2(x_{n-1}) - g_1(x_{n-1}) = y_{n-1} - y_{n-1} = 0$$

Par conséquent, d est divisible par $(x-x_0)\cdot(x-x_1)\cdot\ldots\cdot(x-x_{n-1})$ donc $d(x) = (x-x_0)(x-x_1)\ldots(x-x_{n-1})\,q(x)$, où q est un polynôme. Si q n'est pas identiquement nulle, alors d est de degré supérieur à $n-1$ ce qui n'est pas possible par hypothèse. d est donc identiquement nul ce qui signifie que les polynômes g_1 et g_2 sont égaux.

■

§ 1.4 Interpolation trigonométrique

Nous considérons une tabelle de valeurs numériques qui représente la température moyenne journalière en un lieu imaginaire

t	Début janvier	Fin mars	Mi-juin	Mi-septembre
y	-3	5	15	8

A partir de cette tabelle, nous voulons estimer la température moyenne journalière au début du mois d'août.

Numérotons les 365 jours de l'année de 0 à 364:

0	30	31	58	59	89	90	119	120	150	151	180	181	211	212	242	243	272	273	303	304	333	334	364
1 janvier	31 janvier	1 février	28 février	1 mars	31 mars	1 avril	30 avril	1 mai	31 mai	1 juin	30 juin	1 juillet	31 juillet	1 août	31 août	1 septembre	30 septembre	1 octobre	31 octobre	1 novembre	30 novembre	1 décembre	31 décembre

Ainsi, le temps sera exprimé en jours (unité d = day).

t [d]	0	89	166	258
y [°C]	-3	5	15	8

Montrons d'abord qu'il n'est pas judicieux d'interpoler par un polynôme car le climat est un phénomène cyclique.

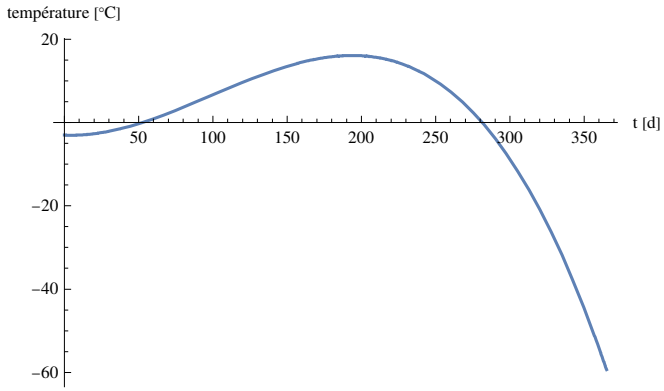
```
Clear["Global`*"]
```
efface

```
pts = {{0, -3.}, {89, 5.}, {166, 15.}, {258, 8.}};
g[t_] = InterpolatingPolynomial[pts, t];
```
polynôme d'interpolation

```
Expand[g[t]]
```
développe

$-3. - 0.0151272\, t + 0.00168343\, t^2 - 5.65713 \times 10^{-6}\, t^3$

```
Plot[g[t], {t, 0, 365}, AxesLabel → {"t [d]", "température [°C]"}]
```
tracé de courbes titre d'axe



Le climat étant un phénomène cyclique, la température au 31 décembre doit être proche de celle du premier janvier. Le premier jour de l'année suivante, la température devrait être de −3 °C, ce qui est loin d'être le cas

```
g[365]
```

-59.3369

Il faut interpoler par des fonctions périodiques dont la période est un sous-multiple de 365 jours, c'est-à-dire 365, $\frac{365}{2}$, $\frac{365}{3}$, ...

Choisissons comme *fonctions de base*

$$b_0(t) = 1; \quad b_1(t) = \cos\left(\frac{2\pi}{365} t\right); \quad b_2(t) = \sin\left(\frac{2\pi}{365} t\right); \quad b_3(t) = \cos\left(\frac{4\pi}{365} t\right).$$

La fonction d'interpolation est

$g(t) = c_0\, b_0(t) + c_1\, b_1(t) + c_2\, b_2(t) + c_3\, b_3(t) =$

$$c_0 + c_1 \cos\left(\frac{2\pi}{365} t\right) + c_2 \sin\left(\frac{2\pi}{365} t\right) + c_3 \cos\left(\frac{4\pi}{365} t\right).$$

Les conditions à remplir sont

$$g(0) = -3, \quad g(89) = 5, \quad g(166) = 15, \quad g(258) = 8$$

$$c_0 + c_1 \cos\left(\frac{2\pi}{365} 0\right) + c_2 \sin\left(\frac{2\pi}{365} 0\right) + c_3 \cos\left(\frac{4\pi}{365} 0\right) = -3,$$

$$c_0 + c_1 \cos\left(\frac{2\pi}{365} 89\right) + c_2 \sin\left(\frac{2\pi}{365} 89\right) + c_3 \cos\left(\frac{4\pi}{365} 89\right) = 5,$$

$$c_0 + c_1 \cos\left(\frac{2\pi}{365} 166\right) + c_2 \sin\left(\frac{2\pi}{365} 166\right) + c_3 \cos\left(\frac{4\pi}{365} 166\right) = 15,$$

$$c_0 + c_1 \cos\left(\frac{2\pi}{365} 258\right) + c_2 \sin\left(\frac{2\pi}{365} 258\right) + c_3 \cos\left(\frac{4\pi}{365} 258\right) = 8.$$

Le système d'équations est linéaire. Écrivons-le sous la forme matricielle :

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & \cos\left(\frac{178\pi}{365}\right) & \sin\left(\frac{178\pi}{365}\right) & \cos\left(\frac{356\pi}{365}\right) \\ 1 & \cos\left(\frac{332\pi}{365}\right) & \sin\left(\frac{332\pi}{365}\right) & \cos\left(\frac{664\pi}{365}\right) \\ 1 & \cos\left(\frac{516\pi}{365}\right) & \sin\left(\frac{516\pi}{365}\right) & \cos\left(\frac{1032\pi}{365}\right) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} -3 \\ 5 \\ 15 \\ 8 \end{pmatrix}.$$

Résolvons le système avec *Mathematica* :

```
Clear[b, t]; x = Transpose[pts][[1]];
efface          transposée
```

```
b[t_] = {1, Cos[2 π/365 t], Sin[2 π/365 t], Cos[4 π/365 t]};
        cosinus             sinus           cosinus
```

```
m = Map[b, x]; MatrixForm[m]
    applique   apparence matricielle
```

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & \text{Sin}\left[\frac{9\pi}{730}\right] & \text{Cos}\left[\frac{9\pi}{730}\right] & -\text{Cos}\left[\frac{9\pi}{365}\right] \\ 1 & -\text{Cos}\left[\frac{33\pi}{365}\right] & \text{Sin}\left[\frac{33\pi}{365}\right] & \text{Cos}\left[\frac{66\pi}{365}\right] \\ 1 & -\text{Sin}\left[\frac{63\pi}{730}\right] & -\text{Cos}\left[\frac{63\pi}{730}\right] & -\text{Cos}\left[\frac{63\pi}{365}\right] \end{pmatrix}$$

Les coefficients de la matrice sont des valeurs numériques; aussi, la résolution du système ne présente pas de difficulté particulière.

```
y = Transpose[pts][[2]];
    transposée
```

```
c = LinearSolve[m, y] // N;
    résous équation linéaire   val
```

```
MatrixForm[c]
apparence matricielle
```

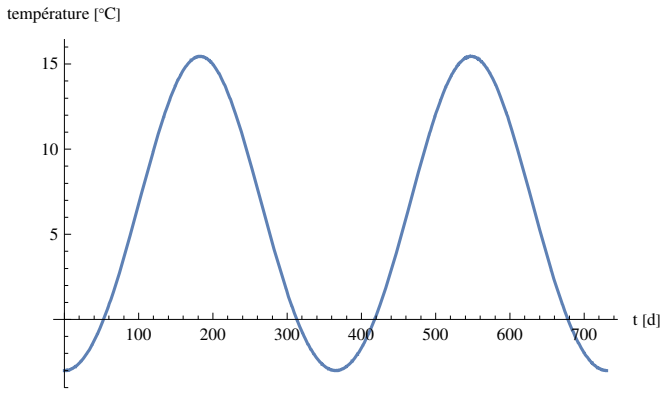
$$\begin{pmatrix} 5.81979 \\ -9.22485 \\ -0.0587724 \\ 0.405065 \end{pmatrix}$$

```
Clear[g, t];
efface
```

```
g[t_] = c.b[t];
g[t]
```

$$5.81979 - 9.22485\,\text{Cos}\left[\frac{2\pi t}{365}\right] + 0.405065\,\text{Cos}\left[\frac{4\pi t}{365}\right] - 0.0587724\,\text{Sin}\left[\frac{2\pi t}{365}\right]$$

```
Plot[g[t], {t, 0, 365*2}, AxesLabel → {"t [d]", "température [°C]"}]
```
tracé de courbes titre d'axe



La fonction est périodique

```
g[0]
```

```
-3.
```

```
g[365]
```

```
-3.
```

La température au début du mois d'août sera donc, en degrés Celsius,

```
g[212]
```

```
14.1226
```

§ 1.5 Interpolation : cas général

Énoncé du problème

Soit

$x_0, x_1, \ldots, x_{n-1}$ des abscisses distinctes $p \le x_j \le q$

$y_0, y_1, \ldots, y_{n-1}$ des ordonnées quelconques;

$b_0, b_1, \ldots, b_{n-1}$ des fonctions de base

$$[p, q] \xrightarrow{b_j} \mathbb{R}, \quad j = 0, 1, \ldots, n-1$$

Nous cherchons une fonction d'interpolation

$$g = c_0 b_0 + c_1 b_1 + \ldots + c_{n-1} b_{n-1}$$

telle que

$$g(x_0) = y_0, \; g(x_1) = y_1, \quad \ldots, \; g(x_{n-1}) = y_{n-1}$$

Calcul

$$g(x_0) = y_0, \; g(x_1) = y_1, \quad \ldots, \; g(x_{n-1}) = y_{n-1}$$

$$c_0 b_0(x_0) + c_1 b_1(x_0) + \ldots + c_{n-1} b_{n-1}(x_0) = y_0$$
$$c_0 b_0(x_1) + c_1 b_1(x_1) + \ldots + c_{n-1} b_{n-1}(x_1) = y_1$$
$$\ldots$$
$$c_0 b_0(x_{n-1}) + c_1 b_1(x_{n-1}) + \ldots + c_{n-1} b_{n-1}(x_{n-1}) = y_{n-1}$$

Le système d'équations peut s'écrire sous forme vectorielle

$$c_0 \begin{pmatrix} b_0(x_0) \\ b_0(x_1) \\ \ldots \\ b_0(x_{n-1}) \end{pmatrix} + c_1 \begin{pmatrix} b_1(x_0) \\ b_1(x_1) \\ \ldots \\ b_1(x_{n-1}) \end{pmatrix} + \ldots + c_{n-1} \begin{pmatrix} b_{n-1}(x_0) \\ b_{n-1}(x_1) \\ \ldots \\ b_{n-1}(x_{n-1}) \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \ldots \\ y_{n-1} \end{pmatrix}$$

Il s'agit de résoudre un système de *n* équations à *n* inconnues :

$$\boxed{\begin{pmatrix} b_0(x_0) & b_1(x_0) & \ldots & b_{n-1}(x_0) \\ b_0(x_1) & b_1(x_1) & \ldots & b_{n-1}(x_1) \\ \ldots & \ldots & \ldots & \ldots \\ b_0(x_{n-1}) & b_1(x_{n-1}) & \ldots & b_{n-1}(x_{n-1}) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \ldots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \ldots \\ y_{n-1} \end{pmatrix}}$$

- **Hypothèse**

La matrice est régulière. En particulier, ses vecteurs-colonnes sont linéairement indépendants et constituent une base de $\mathbb{R}^n$, cela signifie qu'il est possible de construire tous les vecteurs de $\mathbb{R}^n$ à partir de ceux-ci. Nous posons

$$\overrightarrow{e_0} = \begin{pmatrix} b_0(x_0) \\ b_0(x_1) \\ \dots \\ b_0(x_{n-1}) \end{pmatrix}, \overrightarrow{e_1} = \begin{pmatrix} b_1(x_0) \\ b_1(x_1) \\ \dots \\ b_1(x_{n-1}) \end{pmatrix}, \dots, \overrightarrow{e_{n-1}} = \begin{pmatrix} b_{n-1}(x_0) \\ b_{n-1}(x_1) \\ \dots \\ b_{n-1}(x_{n-1}) \end{pmatrix}.$$

Ces vecteur sont appelé *discrétisation* (ou *digitalisation*) des fonctions de base. De façon générale, pour n'importe quelle fonction (dont on ne connaît pas forcément l'expression analytique)

$$f : [p, q] \to \mathbb{R}$$

nous définissons l'opération d'échantillonnage ou de discrétisation comme suit

$$\overrightarrow{\text{discr}(f)} := \begin{pmatrix} f(x_0) \\ f(x_1) \\ \dots \\ f(x_{n-1}) \end{pmatrix}$$

où $\{x_0, x_1, \dots, x_{n-1}\}$ sont les abscisses de l'échantillonnage.

Avec la notation précédente, les vecteurs-colonnes de la matrice prennent la forme

$$\overrightarrow{e_k} = \overrightarrow{\text{discr}(b_k)} \quad \text{pour} \quad k = 0, 1, \dots, n-1$$

Notre problème consiste donc à exprimer le vecteur $\vec{y}$ dans la base $\left\{\overrightarrow{e_0}, \overrightarrow{e_1}, \dots, \overrightarrow{e_{n-1}}\right\}$, c'est-à-dire à déterminer les coefficients $\{c_0, c_1, \dots, c_{n-1}\}$ tels que

$$c_0 \overrightarrow{e_0} + c_1 \overrightarrow{e_1} + \dots + c_{n-1} \overrightarrow{e_{n-1}} = \vec{y} \qquad \text{où} \qquad \vec{y} = \begin{pmatrix} y_0 \\ y_1 \\ \dots \\ y_{n-1} \end{pmatrix}$$

- **Exemple**

Interpolation polynomiale

$$b_0(t) = 1, \; b_1(t) = t, \; b_2(t) = t^2, \dots, b_{n-1}(t) = t^{n-1}$$

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \dots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \dots \\ y_{n-1} \end{pmatrix}$$

§ 1.6 Exercices

- **Exercice 1-1 [sans/avec *Mathematica*]**

Les points suivants étant donnés

$M_0(-1, 2)$, $M_1(1, -1)$, $M_2(2, 3)$, $M_3(5, -7)$,

calculez sans *Mathematica*, par interpolation linéaire les ordonnées qui correspondent à $t = 1.3$ et $t = 3.2$.
Vérifiez vos solutions en utilisant la fonction **`Interpolation`**.

- **Exercice 1-2 [avec *Mathematica*]**

Déterminez le polynôme de degré minimal qui passe par les points suivants:

$M_0(-1, 2)$, $M_1(1, -1)$, $M_2(2, 3)$, $M_3(5, -7)$, $M_4(6, -3)$.

Travaillez dans un premier temps avec **`Solve`**, puis avec **`LinearSolve`** et finalement en utilisant **`InterpolatingPolynomial`**.

■ **Exercice 1-3 [sans *Mathematica*]**

Déterminez le polynôme de degré minimal qui passe par les quatre points suivants

$M_0(-1, 2)$, $M_1(1, -1)$, $M_2(2, 3)$, $M_3(5, -7)$

Prescription d'exercice : écrivez le polynôme cherché comme combinaison linéaire des fonctions de base

$\{1, \quad t - x_0, \quad (t - x_0)(t - x_1), \quad (t - x_0)(t - x_1)(t - x_2)\}$,

où les x_i sont les abscisses des points d'interpolation.
Quel est l'avantage de ces fonctions de base ?
Résolvez le système d'équations et écrivez explicitement la solution g.

■ **Exercice 1-4 [sans *Mathematica*]**

En généralisant la méthode de l'exercice 1-3, démontrez l'existence d'un polynôme d'interpolation de degré n-1 pour un problème d'interpolation à n points.

■ **Exercice 1-5 [sans *Mathematica*]**

Soit les points d'interpolation

$P_0(-1; 3)$, $P_1(2; 5)$, $P_2(3; -1)$,

x_0, x_1, x_2 les abscisses de chacun de ces points et y_0, y_1, y_2 leurs ordonnées. Le but de cet exercice est de déterminer le polynôme d'interpolation p en évitant d'avoir à résoudre un système d'équations. Pour ce faire, procéder de la façon suivante:

1. Déterminer un polynôme qui s'annule en x_1 et x_2.

2. Déterminer tous les polynômes de degré 2 qui s'annulent en x_1 et x_2.

3. Déterminer le polynôme p_0 de degré 2 qui s'annule en x_1 et x_2 et qui vaut 1 en x_0.

4. Déterminer le polynôme p_1 de degré 2 qui s'annule en x_0 et x_2 et qui vaut 1 en x_1.

5. Déterminer le polynôme p_2 de degré 2 qui s'annule en x_0 et x_1 et qui vaut 1 en x_2.

6. Écrire le polynôme p recherché à l'aide des fonctions de base $\{p_0, p_1.p_2\}$.

$\{p_0, p_1, p_2\}$ est la **base de Lagrange** correspondant aux abscisses d'interpolation et p est le **polynôme de Lagrange** correspondant aux points d'interpolation.

7. Déterminez le polynôme d'interpolation passant par les points $P_0(-1; 3)$, $P_1(2; 5)$, $P_2(3; -1)$ et $P_3(5; 4)$ à l'aide de la base de Lagrange.

8. Déterminez le polynôme d'interpolation passant par trois points $P_0(x_0; y_0)$, $P_1(x_1; y_1)$ et $P_2(x_2; y_2)$ à l'aide de la base de Lagrange.

■ **Exercice 1-6 [avec *Mathematica*]**

Soit $(-1; 2)$, $(3; 7)$, $(4; 1)$, $(5, -2)$ des points d'interpolation. Le but de cet exercice est de générer automatiquement la base de Lagrange ainsi que le polynôme d'interpolation correspondant à ces points. Pour cela, procédez en suivant les instruction suivantes :

1. Définissez une liste **pts** contenant les points d'interpolation puis générer automatiquement les listes **abs** et **ord** contenant les liste des abscisses et des ordonnées d'interpolation.
2. Générez une liste contenant les fonctions $t - x_0$, $t - x_1$, ..., $t - x_{n-1}$.
3. Construisez la fonction **n[i,t]** correspondant au numérateur de $L_{i-1}(t)$, le $(i-1)^{\text{ème}}$ polynôme de Lagrange : il s'agit de la multiplication de toutes les fonctions de la liste précédente sauf la $i^{\text{ème}}$. Pour cela, utiliser les fonctions **Apply**, **Delete** et **Times**.
4. Construisez la fonction **d[i]** correspondant au dénominateur de $L_{i-1}(t)$, le $(i-1)^{\text{ème}}$ polynôme de Lagrange, en procédant comme au point précédent.
5. En utilisant les fonctions **n** et **d** construisez la liste **b[t]** contenant les n polynômes de Lagrange.
6. Construisez **p[t]** le polynôme de Lagrange. Vérifiez votre résultat en traçant le graphe de p et en faisant figurer sur celui-ci les points d'interpolation.
7. Définissez un module **lagrange[pts_,t_]** qui détermine le polynôme d'interpolation **p[t]** passant par les points **pts** en utilisant la base de Lagrange. Vérifiez que votre module fonctionne correctement à l'aide de l'exemple utilisé dans cet exercice.
8. Définissez un module **lagrange2[pts_,t_]** qui fonctionne comme le module **lagrange**, mais dont l'output contient en plus le graphe du polynôme **p** avec les points **pts**. Pour cela, utilisez la fonction **Print**.

■ **Exercice 1-7 [avec *Mathematica*]**

Déterminez une fonction périodique de période 7 qui passe par les points suivants:

$M_0(-1, 2)$, $M_1(1, -1)$, $M_2(3, -3)$, $M_3(5, 0)$.

Comparez les résultat obtenus

- en prenant dans les fonctions de bases deux sinus,
- en prenant dans les fonctions de bases deux cosinus,
- en paramétrant la fonction **Interpolation** afin qu'elle génère un interpolant périodique.

Pour cela, tracez dans un même repère les points d'interpolation ainsi que le graphe des trois interpolants.