Calcul formel

8. Polynômes

8.1. Manipuler les polynômes

Travaux pratiques 1.
Soient $P(X) = X^4 - 3X^2 - 1$ et $Q(X) = (X+1)^4$ des polynômes de $\mathbb{Q}[X]$. Après avoir déclaré l'anneau des polynômes $\mathbb{Q}[X]$ par `R.<X> = QQ[]` (où `QQ` désigne le corps des rationnels), répondre aux questions suivantes :

1. Est-ce que $\deg(P \cdot Q) = \deg P + \deg Q$?
2. Est-ce que $\deg(P - Q) = \max(\deg P, \deg Q)$?
3. Développer Q. Quel est le coefficient de X^3 ?
4. Quel est le quotient de la division euclidienne de $P(X)$ par $(X+1)^2$? Et le reste ?
5. Quelles sont les racines de Q ? Et celles de P ?

1. (et 2.) La commande `degree()` permet d'obtenir le degré ; ainsi, après avoir déclaré l'anneau de polynômes $\mathbb{Q}[X]$ et avoir défini les deux polynômes P et Q, on teste les égalités sur les degrés.

   Code 1 (*intro-polynome.sage (1)*).
   ```
   R.<X> = QQ[]
   P = X^4-3*X^2-1
   Q = (X+1)^4
   (P*Q).degree() == P.degree() + Q.degree()
   (P-Q).degree() == max(P.degree(),Q.degree())
   ```

   La première égalité est vraie, par contre la seconde est fausse (il n'y a pas toujours égalité des degrés). Les énoncés vrais en toute généralité pour des polynômes non nuls sont :

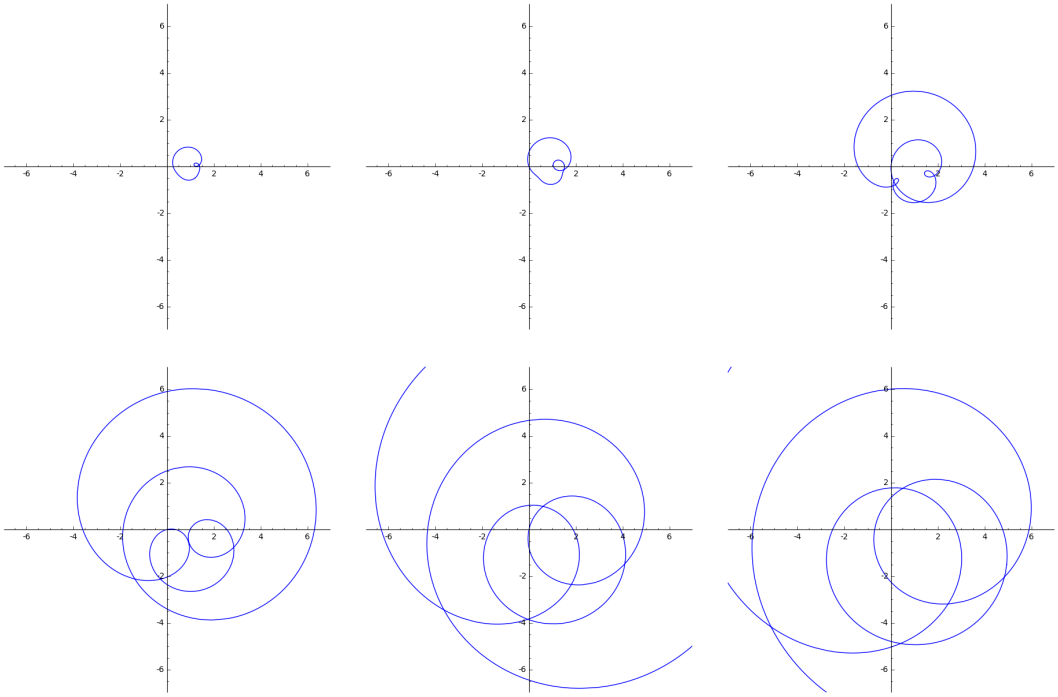
   $$\deg(P \cdot Q) = \deg P + \deg Q \quad \text{et} \quad \deg(P+Q) \leqslant \max(\deg P, \deg Q).$$

3. On développe un polynôme par la commande `expand`. On récupère les coefficients comme si le polynôme était une liste : `Q[k]` renvoie le coefficient a_k devant X^k.
4. On obtient le quotient et le reste comme avec les entiers. Le quotient `P // (X+1)^2` vaut $X^2 - 2X$. Le reste `P % (X+1)^2` vaut $2X - 1$.
5. La question est ambigüe ! Il faut préciser dans quel ensemble on cherche les racines. Est-ce dans $\mathbb{Q}$, $\mathbb{R}$ ou $\mathbb{C}$? Souhaitons-nous une racine exacte ou approchée ?
   - `P.roots()` : renvoie les racines du polynôme appartenant au corps de base (ici $\mathbb{Q}$). La liste renvoyée ici est vide car P n'a pas de racines rationnelles.
   - `Q.roots()` renvoie `[(-1, 4)]` car -1 est une racine de multiplicité 4.
   - `P.roots(QQbar)` : racines exactes dans $\mathbb{C}$ (pour un polynôme à coefficients dans $\mathbb{Q}$). Ici renvoie deux racines réelles et deux racines imaginaires pures : `-1.817354021023971?`, `1.817354021023971?`,

`-0.5502505227003375?*I`, `0.5502505227003375?*I`. Le point d'interrogation en fin d'écriture signifie que Sage calcule avec les valeurs exactes, mais n'affiche que les premières décimales.
- `P.roots(RR)` : racines réelles *approchées* : `-1.81735402102397`, `1.81735402102397`. On quitte donc la résolution formelle pour une résolution numérique approchée.
- `P.roots(CC)` : racines complexes *approchées*.

Travaux pratiques 2.
- Pour un polynôme $P \in \mathbb{C}[X]$ et un réel $r > 0$, tracer l'image par P du cercle centré à l'origine du plan complexe $\mathbb{C}$ et de rayon r.
- Faites varier r (ou mieux faites une animation). En quoi cela illustre-t-il le théorème de d'Alembert-Gauss ?
- Application à $P(X) = X^4 - X^3 + X^2 - \mathrm{i}X + 1 \in \mathbb{C}[X]$.

Voici quelques images de l'animation pour des valeurs de r valant successivement : $r_0 = 0.5$, $r_1 = 0.6176\ldots$, $r_2 = 0.9534\ldots$, $r_3 = 1.2082\ldots$, $r_4 = 1.4055\ldots$ et $r_5 = 1.5$.



Quand la courbe $\mathscr{C}_r$ passe-t-elle par l'origine ? La courbe passe par l'origine s'il existe un nombre complexe $re^{\mathrm{i}t}$ tel que $P(re^{\mathrm{i}t}) = 0$, c'est-à-dire lorsque P admet une racine de module r. On sait qu'un polynôme de degré n a au plus n racines. Donc il y a au plus n valeurs $r_1, \ldots, r_n$ pour lesquelles $\mathscr{C}_{r_i}$ passe par l'origine. Pour notre exemple de degré 4, il y a 4 racines, qui conduisent ici à 4 modules distincts, $r_1, \ldots, r_4$.

Voici comment procéder :
- On commence par définir l'anneau $\mathbb{C}[X]$ par `R.<X> = CC[]`. Les calculs seront donc des calculs approchés avec des nombres complexes. Notre polynôme P est défini par `P = X^4-X^3+X^2-I*X+1`.
- Le cercle de rayon r est l'ensemble des complexes de la forme $re^{\mathrm{i}t}$ pour $t \in [0, 2\pi]$. La courbe $\mathscr{C}_r$ cherchée est donc l'ensemble

$$\mathscr{C}_r = \left\{P(re^{\mathrm{i}t}) \mid t \in [0, 2\pi]\right\}.$$

Voici la fonction qui renvoie ce tracé :

Code 2 (*intro-polynome.sage (2)*).
```
def plot_image_cercle(P,r):
    var('t')
    zreal = P(X=r*exp(I*t)).real()
    zimag = P(X=r*exp(I*t)).imag()
    G = parametric_plot( (zreal,zimag), (t,0,2*pi) )
    return G
```

- Une animation est une juxtaposition d'images. On la définit, en visualisant à chaque fois le pavé $[-3,3]\times[-3,3]$, par :

```
A = animate( [plot_image_cercle(P,r) for r in srange(0.5,1.5,0.05)],
             xmin=-3,xmax=3,ymin=-3,ymax=3 )
```

puis on l'affiche par `A.show()`.

Remarque.
Par défaut `Sage` fait les calculs dans un ensemble appelé `SR` (*Symbolic Ring*). Dans cet ensemble vous pouvez définir des expressions polynomiales et en trouver les racines. L'accès aux fonctions spécifiques aux polynômes (comme le degré, les coefficients...) est un peu différent. Voici comment déclarer un polynôme, récupérer son degré ainsi que le coefficient devant X^2 :

Code 3 (*intro-polynome.sage (3)*).
```
var('X')
P = X^4-3*X^2-1
P.degree(X)
P.coefficient(X,2)
```

8.2. Algorithme de Horner

L'algorithme de Horner permet d'évaluer rapidement un polynôme P en une valeur α. Voyons comment il fonctionne à la main avec l'exemple de $P(X)=X^5+X^4-5X^3-3X-2$ et $\alpha=2$.
Sur la première ligne du tableau, on écrit les coefficients de P, $a_n, a_{n-1}, \dots, a_1, a_0$. On reporte en troisième ligne première colonne, a_n, le coefficient dominant de P. On multiplie ce nombre par α, on l'écrit en deuxième ligne, deuxième colonne, puis on ajoute a_{n-1} que l'on écrit en troisième ligne, deuxième colonne. Et on recommence. À chaque étape, on multiplie le dernier nombre obtenu par α et on lui ajoute un coefficient de P. Le dernier nombre obtenu est $P(\alpha)$.

a_k	1	1	−5	0	−3	−2
$\alpha=2$		2	6	2	4	2
b_k	1	3	1	2	1	0

Ici le dernier nombre est 0, donc $P(2)=0$. Conclusion : 2 est racine de P.
Avec le même polynôme et $\alpha=-1$ le tableau se complète ainsi :

a_k	1	1	−5	0	−3	−2
$\alpha=-1$		−1	0	5	−5	8
b_k	1	0	−5	5	−8	6

ce qui donne $P(-1)=6$.
Ce qui suit formalise cette procédure et montre que l'algorithme de Horner permet des calculs efficaces avec les polynômes.

Travaux pratiques 3.
On fixe un corps K et $\alpha\in K$. Soit $P(X)=a_nX^n+a_{n-1}X^{n-1}+\cdots+a_kX^k+\cdots+a_1X+a_0\in K[X]$.
On pourra prendre pour les applications $P(X)=X^5+X^4-5X^3-3X-2$ et $\alpha=2$, puis $\alpha=-1$.

1. Compter et comparer le nombre de multiplications nécessaires dans K pour calculer $P(\alpha)$, par :

(a) le calcul direct :
$$P(\alpha) = a_n\alpha^n + a_{n-1}\alpha^{n-1} + \cdots + a_k\alpha^k + \cdots + a_1\alpha + a_0,$$
(b) l'algorithme de Horner :
$$P(\alpha) = \Big(\big((a_n\alpha + a_{n-1})\alpha + a_{n-2}\big)\alpha + \cdots + a_1\Big)\alpha + a_0.$$
(c) Écrire une fonction qui calcule $P(\alpha)$ par l'algorithme de Horner.

2. On formalise le calcul précédent en définissant la suite :
$$b_n = a_n \qquad \text{puis pour} \qquad n-1 \geqslant k \geqslant 0 \ : \ b_k = \alpha b_{k+1} + a_k.$$
(a) Montrer que dans la division euclidienne de $P(X)$ par $X-\alpha$ qui s'écrit :
$$P(X) = (X-\alpha)Q(X) + P(\alpha)$$
le reste $P(\alpha)$ est égal à b_0 et le quotient $Q(X)$ est égal au polynôme $b_nX^{n-1} + \cdots + b_2X + b_1$ dont les coefficients sont les b_k pour $k \geqslant 1$.
Indications. Pour ce faire développer $(X-\alpha)Q(X)$ et retrouver pour les coefficients de Q la même relation de récurrence que celle définissant la suite des b_k.
(b) Modifier votre fonction afin qu'elle calcule la suite $(b_n, b_{n-1}, \ldots, b_1, b_0)$ et qu'elle renvoie le quotient Q et le reste $b_0 = P(\alpha)$.
(c) Écrire une fonction qui calcule l'expression d'un polynôme $P(X)$ dans la base des $(X-\alpha)^k$, c'est-à-dire calculer les $c_k \in K$ tels que :
$$P(X) = c_n(X-\alpha)^n + c_{n-1}(X-\alpha)^{n-1} + \cdots + c_1(X-\alpha) + c_0.$$

1. (a) Le calcul d'un monôme $a_k \cdot \alpha^k$ de degré k nécessite k multiplications, donc pour calculer $P(\alpha)$ il faut $n + (n-1) + \cdots + k + \cdots + 1 + 0 = \frac{n(n+1)}{2}$ multiplications (et n additions).
(b) La méthode de Horner
$$P(\alpha) = \Big(\big((a_n\alpha + a_{n-1})\alpha + a_{n-2}\big)\alpha + \cdots + a_1\Big)\alpha + a_0.$$
nécessite seulement n multiplications (et n additions). On passe d'un coût d'ordre $\frac{1}{2}n^2$ à un coût d'ordre n ; le gain est donc énorme.
(c) Pour l'implémentation, on note que la formule de récurrence se fait pour des indices k allant en décroissant.

Code 4 (*horner.sage (1)*).
```
def eval_horner(P,alpha):
    n = P.degree()
    val = P[n]
    for k in range(n-1,-1,-1):
        val = alpha*val + P[k]
    return val
```

2. (a) On note la division euclidienne de $P(X)$ par $X-\alpha$ sous la forme $P(X) = (X-\alpha)Q(X) + R$. Le polynôme Q est de degré $n-1$ et on le note $Q(X) = b'_nX^{n-1} + \cdots + b'_kX^{k-1} + \cdots + b'_2X + b'_1$ (on veut montrer $b'_k = b_k$). Le reste R est de degré strictement inférieur à $\deg(X-\alpha) = 1$ donc R est un polynôme constant. En évaluant l'égalité de la division euclidienne en α, on a immédiatement $P(\alpha) = R$, que l'on note pour l'instant b'_0.
L'égalité $P(X) = (X-\alpha)Q(X) + R$ s'écrit :
$$P(X) = (X-\alpha)\big(b'_nX^{n-1} + \cdots + b'_kX^{k-1} + \cdots + b'_2X + b'_1\big) + b'_0.$$
On développe le terme de droite et on regroupe les monômes de même degré :
$$P(X) = b'_nX^n + (b'_{n-1} - \alpha b'_n)X^{n-1} + \cdots + (b'_k - \alpha b'_{k+1})X^k + \cdots + (b'_1 - \alpha b'_2)X + (b'_0 - \alpha b'_1)$$

On identifie coefficient par coefficient :

$$\begin{cases} b'_n &= a_n \\ b'_{n-1} - \alpha b'_n &= a_{n-1} \\ & \cdots \\ b'_k - \alpha b'_{k+1} &= a_k \\ & \cdots \\ b'_0 - \alpha b'_1 &= a_0 \end{cases} \quad \text{donc} \quad \begin{cases} b'_n &= a_n \\ b_{n-1'} &= \alpha b'_n + a_{n-1} \\ & \cdots \\ b'_k &= \alpha b'_{k+1} + a_k \\ & \cdots \\ b'_0 &= \alpha b'_1 + a_0 \end{cases}$$

Les suites (b'_k) et (b_k) sont définies par le même terme initial a_n et la même relation de récurrence, elles sont donc égales.

(b) On modifie légèrement le code précédent. La suite $(b_n, b_{n-1}, \ldots, b_1)$ donne les coefficients de Q (attention au décalage) et b_0 donne le reste qui n'est autre que $P(\alpha)$.

Code 5 (*horner.sage (2)*).

```
def division_horner(P,alpha):
    n = P.degree()
    if n <= 0: return 0,P
    b = [None] * (n+1)     # Liste triviale de longueur n+1
    b[n] = P[n]
    for k in range(n-1,-1,-1):
        b[k] = alpha*b[k+1] + P[k]
    Q = sum( b[k+1]*X^k for k in range(0,n) )
    R = b[0]
    return Q,R
```

- Pour notre polynôme $P(X) = X^5 + X^4 - 5X^3 - 3X - 2$ et $\alpha = 2$, l'appel de la fonction définie ci-dessus : `division_horner(P,alpha)` renvoie un reste R nul et un quotient $Q(X) = X^4 + 3X^3 + X^2 + 2X + 1$. Ce qui signifie que $X - 2$ divise $P(X)$ ou encore que $P(2) = 0$.
- Pour ce même polynôme et $\alpha = -1$, la commande `division_horner(P,alpha)` renvoie $Q(X) = X^4 - 5X^2 + 5X - 8$ et un reste $R = 6$.

(c) On obtient les c_k comme restes de divisions euclidiennes successives par $X - \alpha$. On commence par la division euclidienne de $P(X)$ par $X - \alpha$. On a vu comment calculer le quotient Q_1 et le reste $b_0 = P(\alpha)$. En évaluant l'expression $P(X) = c_n(X-\alpha)^n + \cdots + c_1(X-\alpha) + c_0$ en α, on voit que l'on a aussi $c_0 = P(\alpha)$ donc c_0 est le reste de la première division par $X - \alpha$.
On a donc $P(X) = (X-\alpha)Q_1(X) + c_0$. On recommence en divisant Q_1 par $X - \alpha$, on obtient un quotient Q_2 et un reste qui va être c_1. Donc $P(X) = (X-\alpha)\big((X-\alpha)Q_2 + c_1\big) + c_0$.
On continue ainsi de suite jusqu'à ce que le quotient soit nul (au bout de $n+1$ étapes), on a alors

$$P(X) = \Big(\big((c_n(X-\alpha) + c_{n-1})(X-\alpha) + c_{n-2}\big)(X-\alpha) + \cdots c_1\Big)(X-\alpha) + c_0$$

soit en développant :

$$P(X) = c_n(X-\alpha)^n + \cdots + c_1(X-\alpha) + c_0.$$

Le code suivant renvoie les coefficients $(c_0, c_1, \ldots, c_n)$.

Code 6 (*horner.sage (3)*).

```
def developpe_horner(P,alpha):
    Q = P
    coeff = []
    while Q != 0:
        Q,R = division_horner(Q,alpha)
        coeff.append(R)
    return coeff
```

- Pour $P(X) = X^5 + X^4 - 5X^3 - 3X - 2$ et $\alpha = 2$, la commande `developpe_horner(P,alpha)` renvoie la liste `[0, 49, 74, 43, 11, 1]`, ce qui signifie :

$$P(X) = 1 \cdot (X-2)^5 + 11 \cdot (X-2)^4 + 43 \cdot (X-2)^3 + 74 \cdot (X-2)^2 + 49 \cdot (X-2).$$

- Pour le même polynôme et $\alpha = -1$, la fonction renvoie `[6, -17, 11, 1, -4, 1]`, donc
$$P(X) = 1\cdot(X+1)^5 - 4\cdot(X+1)^4 + 1\cdot(X+1)^3 + 11\cdot(X+1)^2 - 17\cdot(X+1) + 6.$$

8.3. Interpolation de Lagrange

Théorème 1 (Interpolation de Lagrange)**.**
Soient (x_i, y_i), $i = 0, \dots, n$, une suite de $n+1$ points, d'abscisses deux à deux distinctes. Il existe un unique polynôme P de degré inférieur ou égal à n tel que
$$P(x_i) = y_i \quad \text{pour } i = 0, \dots, n.$$

En particulier, pour toute fonction f continue sur un intervalle contenant $x_0, \dots, x_n$, il existe un unique polynôme P de degré inférieur ou égal à n tel que
$$P(x_i) = f(x_i) \quad \text{pour } i = 0, \dots, n.$$

Travaux pratiques 4.

1. Montrer l'unicité du polynôme P dans le théorème d'interpolation de Lagrange.
2. Pour l'existence, étant donnés $x_0, \dots, x_n$, on définit les polynômes de Lagrange $L_0, L_1, \dots, L_n$:
$$L_i(X) = \prod_{j \neq i} \frac{X - x_j}{x_i - x_j}.$$
Montrer que le polynôme :
$$P(X) = \sum_{i=0}^{n} y_i L_i(X)$$
répond au problème : il est de degré inférieur ou égal à n et $P(x_i) = y_i$ pour $i = 0, \dots, n$.
3. Écrire une fonction qui, étant donnée une liste de points, renvoie le polynôme d'interpolation P.
4. Interpoler la fonction définie par $f(x) = \sin(2\pi x)e^{-x}$ sur l'intervalle $[0, 2]$ pour une subdivision régulière de $n+1$ points. Tracer les graphes de la fonction f et des polynômes d'interpolation correspondant à différentes valeurs de n.
5. Faire le même travail avec la fonction définie par $f(x) = \frac{1}{1+8x^2}$ sur l'intervalle $[-1, 1]$ pour une subdivision régulière de $n+1$ points. Quel problème apparaît ? C'est le *phénomène de Runge*.

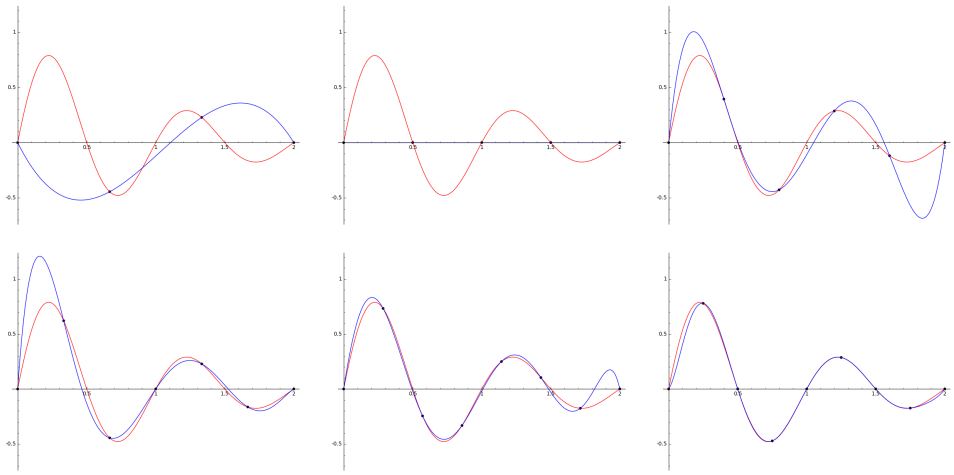
1. Supposons que P et Q soient deux polynômes de degré inférieur à n vérifiant $P(x_i) = y_i$ et $Q(x_i) = y_i$ pour $i = 0, \dots, n$. Alors le polynôme $P - Q$ vérifie $(P-Q)(x_i) = 0$ pour $i = 0, \dots, n$. Ainsi $P - Q$ est un polynôme de degré inférieur ou égal à n ayant $n+1$ racines. C'est donc nécessairement le polynôme nul. Ainsi $P - Q = 0$, donc $P = Q$.
2. Les polynômes L_i sont de degré exactement n et sont définis de sorte que
$$L_i(x_i) = 1 \quad \text{et} \quad L_i(x_j) = 0 \quad \text{pour } j \neq i.$$
Il est alors clair que $P(X) = \sum_{j=0}^{n} y_j L_j(X)$ est aussi de degré inférieur ou égal à n et vérifie pour $i = 0, \dots, n$:
$$P(x_i) = \sum_{j=0}^{n} y_j L_j(x_i) = y_i L_i(x_i) = y_i.$$
3. On commence par définir :
   - `R.<X> = RR[]` : l'anneau de polynômes (les coefficients sont des réels approchés) ;
   - `f = sin(2*pi*x)*exp(-x)` : la fonction ;
   - `liste_points = [ (2*i/n, f(x=2*i/n)) for i in range(n+1) ]` : une liste de points du graphe de f.

   La fonction suivante renvoie le polynôme interpolateur d'une liste de points.

Code 7 (*interpolation.sage*)**.**
```
def interpolation_lagrange(liste_points):
    n = len(liste_points)-1
    allx = [p[0] for p in liste_points] # Abscisses
```

```
    ally = [p[1] for p in liste_points] # Ordonnées
    liste_lagrange = []
    for i in range(n+1):
        A = prod(X-x for x in allx if x != allx[i])          # Numérateur
        B = prod(allx[i]-x for x in allx if x != allx[i]) # Dénominateur
        L = A/B                                          # Polynôme de Lagrange
        liste_lagrange.append(L)
    # Le polynôme interpolateur :
    lagrange = sum( liste_lagrange[i]*ally[i] for i in range(n+1) )
    return lagrange
```

4. Voici les tracés de la fonction (en rouge) définie par $f(x) = \sin(2\pi x)e^{-x}$ et des polynômes d'interpolation (en bleu) pour $n = 3, 4, 5, 6, 7$ et 8. Pour $n = 4$, le polynôme est le polynôme nul. À partir de $n = 8$, il devient difficile de distinguer le graphe de la fonction du graphe du polynôme.



5. Voici l'interpolation de $f(x) = \frac{1}{1+8x^2}$ pour $n = 7, 10$ et 18. La zone centrale de la courbe est bien interpolée mais, autour de -1 et $+1$, des «bosses» apparaissent. Ces bosses deviennent de plus en plus grandes en se décalant vers les extrémités. Il y a convergence simple mais pas convergence uniforme. Pour éviter ce phénomène, il faudrait mieux choisir les x_i.