TP : Interpolation et approximation

Les fichiers nécessaires à ce TP sont disponibles à l'adresse :
`https://espaces-collaboratifs.grenet.fr/share/page/site/UJFueDDMAP110120/dashboard`
sous la rubrique `Espace documentaire` *dans le dossier* `Documents/Thème_Courbes/TP4`.

Sinon, aller à l'adresse `https://espaces-collaboratifs.grenet.fr`.
Se loguer, puis dans l'onglet `Sites`, *sélectionner* `Rechercher un site` *et entrer* `MAP110-120`.

Attention : les fichiers contenant le programme d'une fonction sont destinés à être utilisés, et en aucun cas à être modifiés.

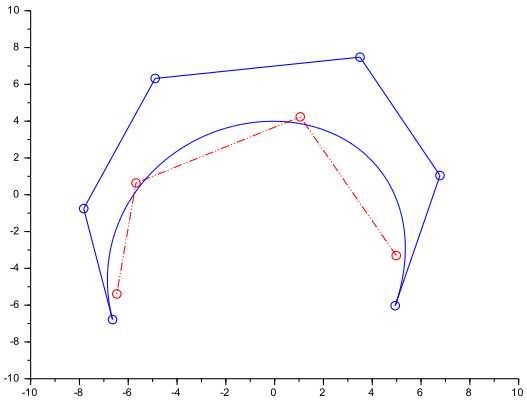
Etant donnée une suite de points (x_i, y_i) du plan, nous nous intéressons dans la première partie de ce TP à la construction d'une courbe polynomiale qui passe par ces point. Il s'agit d'un problème d'interpolation.

1 Préambule : tentative d'interpolation par une courbe de Bézier

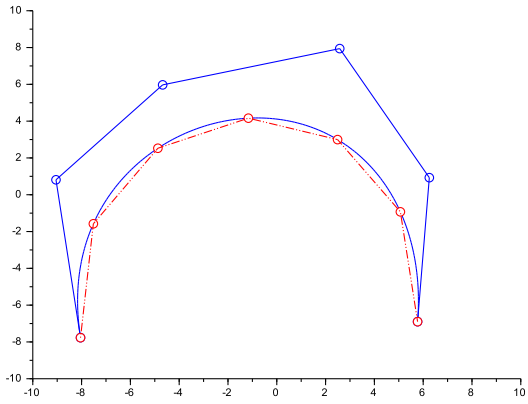
La définition d'une courbe paramétrée polynomiale sous forme Bézier permet de *controler* sa forme à l'aide de son polygone de contrôle. Aussi, peut-on envisager d'utiliser ce formalisme pour astreindre la courbe à passer par des points pré-définis...

Exercice 1 *Récupérer le fichier* Preambule.sce *qui contient plusieurs fonctions.*
La fonction AcquisitionPolygone() *permet d'acquérir à la souris et tracer un polygone, le dernier point étant associé à un click droit. — Exécuter ce fichier afin de :*

1. *définir un polygone de quelques (3 ou 4) points dans la fenêtre graphique* $[-10, 10] \times [-10, 10]$ *(en pointillé rouge sur la figure ci-dessous),*
2. *définir une courbe de Bézier (à l'aide de son polygone de contrôle) interpolant les sommets du polygone précédent.*



pas loin...



parfait ! — mais bien sûr, j'ai triché.

2 Interpolation par une courbe explicite $x \mapsto y(x)$

On suppose donnée une suite de $n+1$ points (x_i, y_i), $i = 0, 1, ..., n$, les abscisses x_i étant deux à deux distinctes et comprises dans un intervalle $[a, b] = [-10, 10]$ dans ce TP. Nous cherchons à déterminer une courbe polynomiale $y(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ (on note que le degré est égal au nombre total de points moins un), vérifiant :

$$y(x_i) = y_i, \quad i = 0, 1, ..., n.$$

2.1 Deux cas simples

◇ Le cas de deux points (donc $n = 1$) conduit à chercher une courbe $y(x) = a_0 + a_1 x$ (autrement dit une droite) passant par deux points d'abscisses distinctes.

◇ Le cas de trois points conduit à chercher une courbe $y(x) = a_0 + a_1 x + a_1 x^2$ (autrement dit une parabole) passant par trois points d'abscisses distinctes. Les 3 contraintes $y(x_i) = y_i$ pour $i = 0, 1, 2$, s'écrivent matriciellement sous la forme :

$$\begin{pmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix} \quad \text{noté} \quad A\,X = b, \quad \text{conduisant à :} \quad X = A^{-1}\,b.$$

Exercice 2 *Récupérer le fichier Explicite3Points.sce à l'adresse du lien donné ci-dessus et compléter les lignes suivantes :*
lignes 47,48 : construction de la matrice A,
ligne 55 : calcul du polynôme d'interpolation p pour l'affichage,
puis expérimenter ce programme.

2.2 Cas général : $n + 1$ points

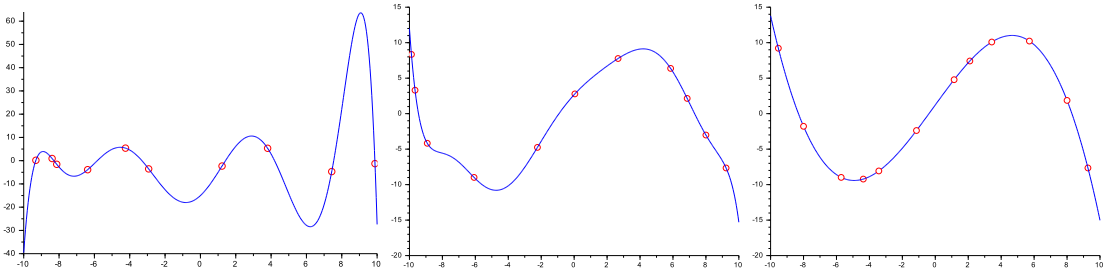
Le problème d'interpolation de $n + 1$ points par un polynôme $y(x)$ de degré n conduit à résoudre le système suivant dont les inconnues sont les coefficients a_i du polynôme

$$\left\{ \begin{array}{ccc} y(x_0) & = & y_0 \\ y(x_1) & = & y_1 \\ \vdots & & \vdots \\ y(x_n) & = & y_n \end{array} \right. \Leftrightarrow \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad \text{noté } A\,X = b.$$

Exercice 3 *Récupérer le fichier ExpliciteGeneral.sce à l'adresse du lien donné ci-dessus. L'acquisition des points se fait dans une fenêtre donnée à l'aide de clicks gauches, interdit les points de même abscisse et se termine uniquement par un click droit.*
a) Compléter la ligne 38 consistant à définir la matrice de Vandermonde A donné ci-dessus, selon l'algorithme proposé ci-dessous, où la variable Scilab xi (vecteur ligne à $n + 1$ composantes) contient les points d'interpolation $x_0, x_1, ..., x_n$.

```
initialisation
    A = ones(n+1,n+1)
pour k = 1 jusqu'à n faire
    colonne(k+1)_de_A = colonne(k)_de_A .* vecteur_colonne_xi'
    // produit terme à terme des deux colonnes
fin pour
```

b) Expérimenter ce programme.
c) Tenter ensuite de construire une courbe en forme de "S" couché et lisse (voir ci-dessous, à droite) par interpolation d'une douzaine de points... si possible sans oscillations ! (voir ci-dessous, au milieu). Que peut-on en conclure quant au "contrôle" de la forme de la courbe par interpolation de quelques points ?



La figure de droite est suspecte... car bien sur j'ai encore triché.

2.3 Evaluation d'un polynôme

Les coefficients $a_0, a_1, ..., a_n$ étant donnés, nous considérons la question de l'évaluation du polynôme

$$p(x) = a_0 + a_1\, x + a_2\, x^2 + a_3\, x^3 + \cdots + a_n\, x^n$$

pour une valeur réelle x donnée. Notons déjà que l'algorithme suivant, basé sur l'exponentiation, est à proscrire.

```
p = a(0)
pour k = 1  jusqu'à n faire
    p = p + a(k) * x^k
fin pour
```

Nous donnons ci-dessous deux autres algorithmes permettant d'évaluer $p(x)$.

Algorithme " classique " :

```
// données : un réel x et les coefficients
// a(0),a(1),...,a(n) du polynôme
xk = 1
p = a(0)
pour i = 1 jusqu'à n faire
    xk = xk * x
    p = p + a(i) * xk
fin pour
```

Algorithme d'évaluation de Horner :

```
// données : un réel x et les coefficients
// a(0),a(1),...,a(n) du polynôme
p = a(n)
pour i = n-1 jusqu'à 0 avec un pas de -1 faire
    p = p * x + a(i)
fin pour
```

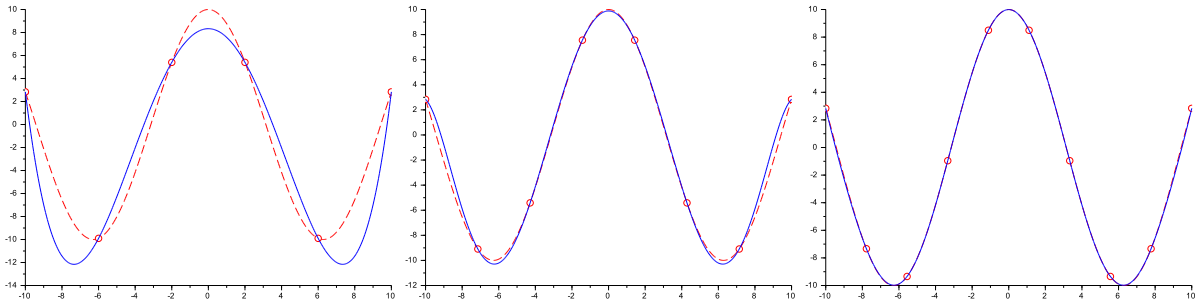
Exercice 4 *a) Utiliser l'algorithme de Horner pour évaluer le polynôme $p(x) = -2+3\,x+2\,x^2-x^3+2\,x^4$, pour la valeur $x = 5$. — b) Comparer le coût (nombre de multiplications et d'additions) de ces deux algorithmes. Lequel vous semble le plus performant ?*

3 Polynôme d'interpolation d'une fonction

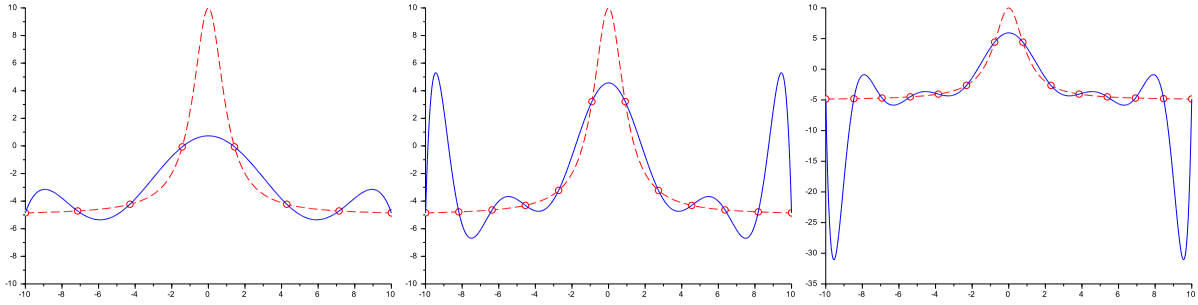
3.1 Principe

Etant donnée une fonction f suffisamment régulière, définie sur l'intervalle $[a, b] = [-10, 10]$, ainsi qu'une suite d'abscisses x_i, $i = 0, 1, ..., n$, deux à deux distinctes, on appelle polynôme d'interpolation de f en $x_0, x_1, ..., x_n$, l'unique polynôme $p(x)$ de degré n, noté $P(x, f)$, qui interpole les données $\big(x_i, f(x_i)\big)$, pour $i = 0, 1, ..., n$. Autrement dit, les ordonnées y_i ne sont plus ici "choisies" par l'utilisateur, mais proviennent de l'échantillonnage d'une fonction.

Exercice 5 *Récupérer le fichier InterpolFonction.sce à l'adresse du lien donné ci-dessus. Ce programme permet de tracer (en pointillés rouge) le graphe d'une fonction définie sur l'intervalle $[-10, 10]$, puis de déterminer et tracer (en bleu continu) le polynôme d'interpolation de cette fonction en $n + 1$ points équidistants $x_0, x_1, ..., x_n$ sur l'intervalle $[a, b] = [-10, 10]$, $x_i = a + i\,(b - a)/n$ pour $i = 0, 1, ..., n$, soit en Scilab : xi = linspace(a,b,n+1).*
a) Compléter les lignes 47 et 53 et expérimenter ce programme pour chacune des fonctions suivantes $f_1(x) = 10\,\cos(x/2)$ et $f_2(x) = -5 + 15/(1 + x^2)$, $x \in [-10, 10]$.
b) On notera les différences de comportement (convergence pour la fonction cosinus et phénomène d'oscillations dit "de Runge" pour la seconde fonction) lorsque le nombre de points d'interpolation augmente.
c) Vous expérimenterez ce programme pour une nouvelle fonction de votre choix que vous définirez en ligne 33.



6 points — 8 points — 10 points

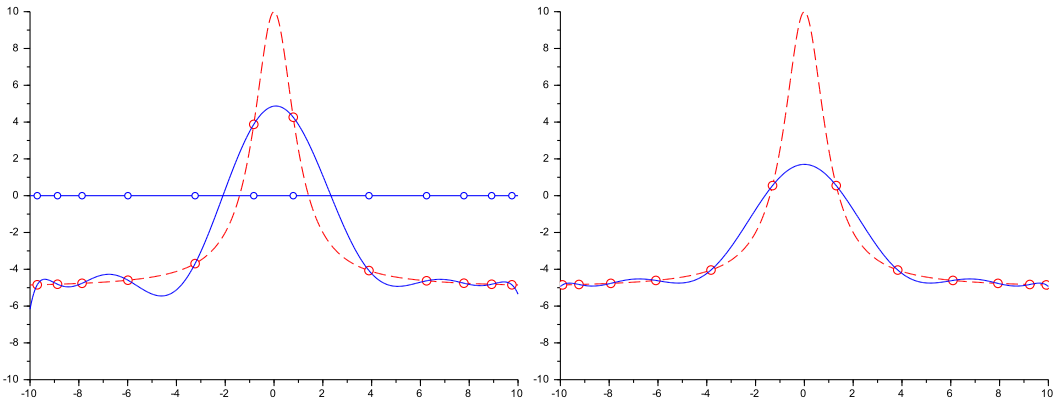


8 points — 12 points — 14 points

3.2 Points de Tchebychev (ou Chebyshev)

Le phénomène d'oscillations observé ci-dessus provient en grande partie de la régularité de l'échantillonnage des points d'interpolation x_i (c'est-à-dire, du fait que les x_i sont équidistants). Reprenons donc l'exemple de la fonction $f_2(x)$ et essayons de limiter les oscillations de sa fonction interpolante par un choix plus judicieux des points x_i (le nombre de points étant fixé par exemple à $n+1=12$).

Exercice 6 *Dans un premier temps nous proposons de choisir les abscisses d'interpolation x_i à la souris afin d'essayer de définir une stratégie optimale. Expérimenter une telle stratégie à l'aide du fichier* Choix-PointsSouris.sce. *Ce programme trace la fonction à interpoler, permet d'acquérir à la souris $n+1=12$ abscisses qui sont visualisées sur l'axe des abscisses, puis trace la fonction interpolante associée. On pourra bien sur faire varier le nombre de points d'interpolation.*
Conseil : il est préférable d'avoir une "densité" de points plus importante sur les bords de l'intervalle.



Gauche : choix de 12 abscisses d'interpolation "à la souris" — Droite : les 12 points de Tchebychev.

Pour une fonction définie sur un intervalle $[a,b]$, Tchebychev (1821-1894) a montré que le choix optimal de $n+1$ points d'interpolation était donné par

$$x_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\Big(\frac{(2k+1)\pi}{2n+2}\Big)\,, \qquad k=0,1,...n.$$

Ces points sont appelés points de Tchebychev.

Exercice 7 *Récupérer le fichier* Tchebychev.sce *qui permet de comparer le polynôme d'interpolation (en vert) obtenu dans le cas de points équidistants avec le polynôme d'interpolation (en bleu) obtenu dans le cas des points de Tchebychev.*
a) Compléter les lignes 67 et 68 permettant de calculer les points de Tchebychev.
b) Expérimenter ce programme, en particulier en faisant maintenant varier le nombre de points d'interpolation nbxi. *Peut-on dire que (expérimentalement) le polynôme d'interpolation se "rapproche" de la courbe f_2 lorsque le nombre de points d'interpolation augmente ? Tester la fonction f_3 donnée dans le fichier.*

Bonus :
Afin de vérifier (expérimentalement) l'optimalité des points de Tchebychev, nous allons introduire une légère perturbation sur ces points et comparer les graphes des 2 polynômes d'interpolation associés.

Exercice 8 *Récupérer le fichier* TchebychevRandom.sce *où l'on définit une nouvelle suite de points d'interpolation* xkPertub *par perturbation des points de Tchebychev* xk *déterminés dans l'exercice précédent. L'interpolant associé aux points de Tchebychev est tracé en bleu et celui associé aux points perturbés est tracé en vert. — Expérimenter ce programme en faisant varier le coefficient d'amplitude et le nombre de points d'interpolation* nbxi. *— Comparer les différents graphes. — Tester d'autres fonctions. — Tester dans la console Scilab l'instruction* rand(xk) *pour un vecteur* xk *afin de comprendre les lignes 65 à 67.*

4 Interpolation par une courbe paramétrique $t \mapsto \big(x(t), y(t)\big)$

Considérons maintenant une suite ordonnée de $n+1$ points $M_i = (x_i, y_i)$ du plan. Nous nous intéressons au problème de la détermination d'une courbe paramétrée polynomiale de degré n interpolant ces points. Par défaut, cette courbe sera paramétrée sur l'intervalle $[0, 1]$. Ainsi, nous cherchons deux polynômes $x(t)$ et $y(t)$, chacun de degré n, tels que la courbe paramétrée

$$t \in [0,1] \mapsto M(t) = \big(x(t), y(t)\big)$$

passe par les points (x_i, y_i).
Nous devons donc choisir une suite strictement croissante de $n+1$ paramètres t_i dans l'intervalle $[0, 1]$, tels que

$$M(t_i) = M_i \quad \Leftrightarrow \quad \big(x(t_i), y(t_i)\big) = (x_i, y_i), \quad \text{pour} \quad i = 0, 1, ..., n.$$

Ce problème se ramène donc au cas explicite (vu ci-dessus) et revient donc à résoudre les deux problèmes d'interpolation suivants

$$x(t_i) = x_i, \quad i = 0, 1, ..., n \qquad \text{et} \qquad y(t_i) = y_i, \quad i = 0, 1, ..., n.$$

Nous considérons deux méthodes pour le choix des paramètres t_i : la paramétrisation équidistante et la paramétrisation cordale.

4.1 Paramétrisation équidistante

La paramétrisation équidistante (ou unifome) consiste à choisir $n+1$ paramètres t_i équirépartis dans l'intervalle $[0, 1]$: $t_i = i/n,\ i = 0, 1, ..., n$.

Exercice 9 *Récupérer le fichier* InterpolParam.sce *qui permet l'interpolation paramétrique d'une suite de points acquis à la souris. La paramétrisation proposée est ici uniforme : les paramètres t_i sont choisis équidistants en ligne 57. Les lignes 60 et 61 (incomplètes) concernent les 2 problèmes d'interpolation $x(t_i) = x_i$ et $y(t_i) = y_i$. — Compléter ces deux lignes 60 et 61 et expérimenter ce programme.*

4.2 Paramétrisation cordale

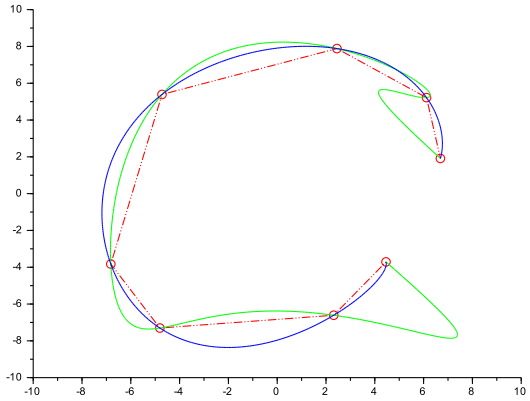
La paramétrisation cordale consiste à choisir les $n+1$ paramètres t_i de sorte que les distances entre les paramètres t_i et t_{i+1} soient proportionnelles aux distances entre les points à interpoler associés M_i et M_{i+1}. Précisément :

$$t_0 = 0 \quad \text{et} \quad t_i = \frac{\sum_{j=0}^{i-1} ||\overrightarrow{M_j M_{j+1}}||}{\sum_{j=0}^{n-1} ||\overrightarrow{M_j M_{j+1}}||}, \quad i = 1, ..., n.$$

Soit en Scilab :

```
// Paramétrisation cordale
tk = zeros(xi);
for i = 2 : length(xi)
    tk(i) = tk(i-1) + sqrt( (xi(i) - xi(i-1))^2 + (yi(i) - yi(i-1))^2 );
end
tk = tk / tk(length(xi));
```

Exercice 10 *Compléter le fichier* InterpolParam.sce *à partir de la ligne 66 afin d'appliquer la paramétrisation cordale. — Expérimenter ce programme et comparer les deux méthodes (c'est-à-dire, les 2 paramétrisations).*



La courbe verte est l'interpolant associée à la paramétrisation uniforme tandis que la courbe bleue est l'interpolant associée à la paramétrisation cordale.

La seconde partie de ce TP est consacré au problème de l'approximation. Précisément, étant donnée une suite de points (x_i, y_i) du plan, nous nous intéressons maintenant à la construction d'une courbe polynomiale qui passe à " proximité " de ces points. Ce problème n'a bien sur un sens que si la courbe n'a pas la capacité à passer par l'ensemble des points.

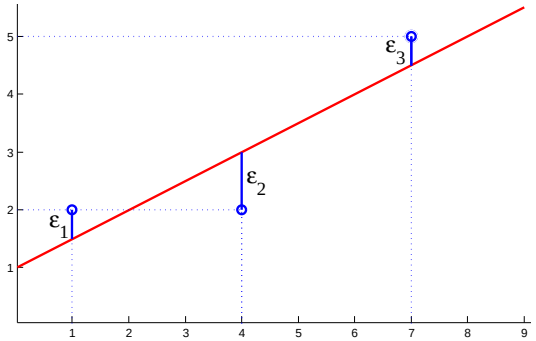
5 Approximation par une courbe explicite $x \mapsto y(x)$

Imposer à une courbe de passer par des points pré-définis par un procédé d'interpolation peut générer des formes et des effets indésirables (oscillations). L'approximation privilégie plutôt la forme globale de la courbe recherchée (définie par un modèle mathématique) et " approche au mieux " (selon un critère à définir) les données dont le nombre est en général beaucoup plus grand que le nombre de paramètres libres dans le modèle mathématique.

5.1 Droite de régression linéaire

◇ Exemple introductif

On donne les trois points de coordonnées $(1, 2)$, $(4, 2)$, $(7, 5)$ (voir figure ci-jointe) et on se propose de déterminer une droite passant par ces points. Clairement la mission est impossible. Néanmoins, on peut estimer que les données (c'est-à-dire, les points) sont bruitées et que le *modèle* qui doit les représenter est réellement celui d'une droite. La question se pose alors de déterminer une droite qui "approche au mieux" les points.



◇ Moindres carrés

La méthode est la suivante dans le cas général. Supposons donnée une suite de n points (x_i, y_i) du plan avec $n > 2$. Puisque en général, aucune droite ne peut passer par l'ensemble des points, on cherche une droite $Y = a\,X + b$ qui minimise la somme des erreurs au carré, à savoir la quantité

$$\epsilon_i^2 + \cdots + \epsilon_n^2 \quad \text{avec} \quad \epsilon_i = y_i - (a\,x_i + b). \tag{1}$$

On montre que les paramètres optimaux $\hat{a}$ et $\hat{b}$ selon ce critère sont solutions du système linéaire

$$\begin{pmatrix} \sum x_i^2 & \sum x_i \\ \sum x_i & n \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum x_i y_i \\ \sum y_i \end{pmatrix}. \tag{2}$$

La droite $Y = \hat{a}\,X + \hat{b}$ obtenue est appelée droite de régression linéaire (ou droite aux moindres carrés) associée à l'ensemble des données (x_i, y_i). La méthode utilisée pour déterminer cette droite est appelée méthode des moindres carrés qui est une méthode très générale et qui s'applique à de nombreuses situations.

Exercice 11 Quelques calculs à la main. — *On considère la situation de l'exemple introductif.*
a) Ecrire " à la main " le système linéaire permettant de déterminer les coefficients $\hat{a}$ et $\hat{b}$ de la droite de régression linéaire associée à ces 3 points.
b) Résoudre le système linéaire à l'aide de Scilab (ou à la main) et en déduire l'équation de cette droite de régression linéaire (on ne demande pas de la tracer).

◇ Moindres carrés - Bis

Nous reprenons la situation de la régression linéaire décrite ci-dessus, c'est-à-dire d'une suite de n points (x_i, y_i) du plan à approcher par une droite $Y = a\,X + b$, et nous allons retrouver le système (2). L'approche est ici matricielle, toujours de type *moindres carrés*, et permet de considérer des situations plus générales que celle d'une droite (ou même d'un polynôme) passant à proximité de points.

Idéalement, la droite devrait passer par l'ensemble des points (x_i, y_i) qui devraient donc satisfaire le système linéaire suivant :

$$\left\{ \begin{array}{rcl} a\,x_1 + b & = & y_1 \\ a\,x_2 + b & = & y_2 \\ & \vdots & \\ a\,x_n + b & = & y_n \end{array} \right. \quad \Leftrightarrow \quad \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad \text{noté} \quad A\,X = B.$$

Ce système linéaire $A\,X = B$ n'admettant généralement pas de solution, on cherche un vecteur $\hat{X}$ qui minimise la " distance " du vecteur $A\,X$ au vecteur B, autrement dit qui minimise la quantité $||A\,X - B||$, ce qui revient (*) à minimiser la quantité $||A\,X - B||^2 = \sum_{i=1}^{n} \big((a\,x_i + b) - y_i\big)^2 = \sum_{i=1}^{n} \epsilon_i^2$. Nous retrouvons donc bien ici la même formulation que ci-dessus.

On donne le résultat suivant : Etant donné un système linéaire $A\,X = B$, où le nombre de lignes de la matrice A est strictement supérieur au nombre de colonnes, le vecteur $\hat{X}$ qui minimise la quantité $||A\,X - B||^2$ est solution du système linéaire carré suivant

$$\big(A^T A\big)\,X = A^T\,B, \tag{3}$$

où A^T est la matrice transposée de A. Si les colonnes de la matrice A sont linéairement indépendantes, la solution $\hat{X}$ est unique.

Exercice 12 — *Justifier la remarque (*) ci dessus :" ce qui revient à minimiser..." — Retrouver le système linéaire (2) à partir du résultat (3) donné ci-dessus.*

Exercice 13 *Récupérer le fichier Regression.sce. Ce programme permet d'acquérir à la souris une suite de points, puis de déterminer ensuite la droite de régression linéaire associée par la méthode des moindres carrés. — Expérimenter ce programme.*

Exercice 14 *Récupérer le fichier MCRandom.sce et analyser ce que fait ce programme. — Et bien sur, expérimenter ce programme en faisant varier les coefficients d'amplitude. — On modifiera aussi l'équation de la fonction test (droite).*

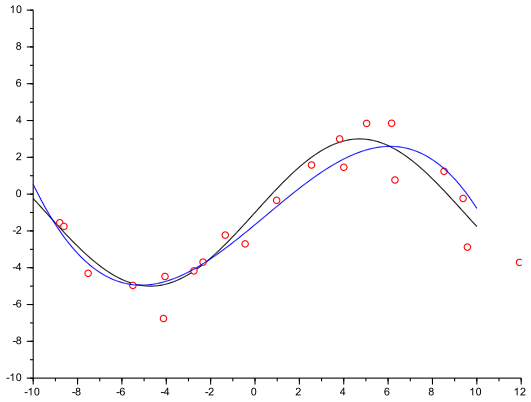
5.2 Généralisation

Exercice 15 *Modifier le fichier Regression.sce afin de déterminer le polynôme de degré d approchant au mieux, au sens des moindres carrés, une suite de N points (x_i, y_i) acquis à la souris ($N > d+1$). — Expérimenter ensuite ce programme.*

Bonus :

Exercice 16 *Modifier le fichier MCRandom.sce afin de considérer des données perturbées issues d'autres fonctions tests, ainsi que des approximants polynomiaux de degré plus élevé. — On approchera par exemple au sens des moindres carrés un échantillonnage de valeurs perturbées issues de la fonction test $y = -1 + 4\sin(x/3)$ par une courbe polynomiale cubique. On fera varier les coefficients d'amplitude. — Généralisation : on pourra aussi considérer N données perturbées issues d'un polynôme de degré d ($d+1 < N$), que l'on approchera au sens des moindres carrés par un polynôme de même degré d. Retrouve-t'on le polynôme initial ?*

La fonction $y(x) = -1 + 4\sin(x/3)$ est représentée en noire. Les points rouges sont les données perturbées issues de cette courbe. L'approximant cubique des données perturbées est tracé en bleu.



5.3 Erreur résiduelle

La méthode des moindres carrés consiste à minimiser la somme des erreurs au carrés définie par la relation (1). Lorsque les paramètres optimaux ont été déterminés, il reste en général une erreur appelée *erreur résiduelle* $E_{res} = \sum_{i=1}^{n} \epsilon_i^2$ qui permet de mesurer l'écart entre les données et le modèle choisi. Afin d'analyser la pertinence d'un modèle, on considère *l'erreur résiduelle moyenne* $\left(\sum_{i=1}^{n} \epsilon_i^2\right)/n$ dont la racine carrée est l'*écart type*.

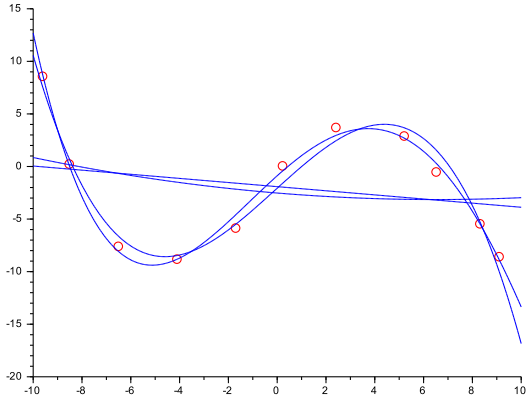
Exercice 17 *Récupérer le fichier ErreurResiduelle.sce qui permet d'acquérir une suite de points à la souris, puis de déterminer le meilleur polynôme de degré d ($1 \leq d \leq d_{max}$) approximant ces points selon la méthode des moindres carrés.*
Vérifier que l'erreur résiduelle moyenne (en fait l'écart type) diminue lorsque le degré du modèle augmente. Cela vous semble-t'il naturel ? Que se passe-t'il si le nombre de points est égal à $d_{max}+1$?
L'erreur résiduelle vous semble-t-elle un outil intéressant pour choisir un modèle mathématique associé à des mesures expérimentales ?

```
Erreur résiduelle moyenne (écart type) :
    5.3102355

Erreur résiduelle moyenne (écart type) :
    5.2894031

Erreur résiduelle moyenne (écart type) :
    1.1155038

Erreur résiduelle moyenne (écart type) :
    0.5593594
```



Les points rouges sont entrés à la souris, puis on détermine selon le critère des moindres carrés, les polynômes d'approximation de degré 1,2,3,4,... (figure ci-dessus à droite), et on affichage l'erreur résiduelle moyenne (en fait l'écart type) (à gauche).