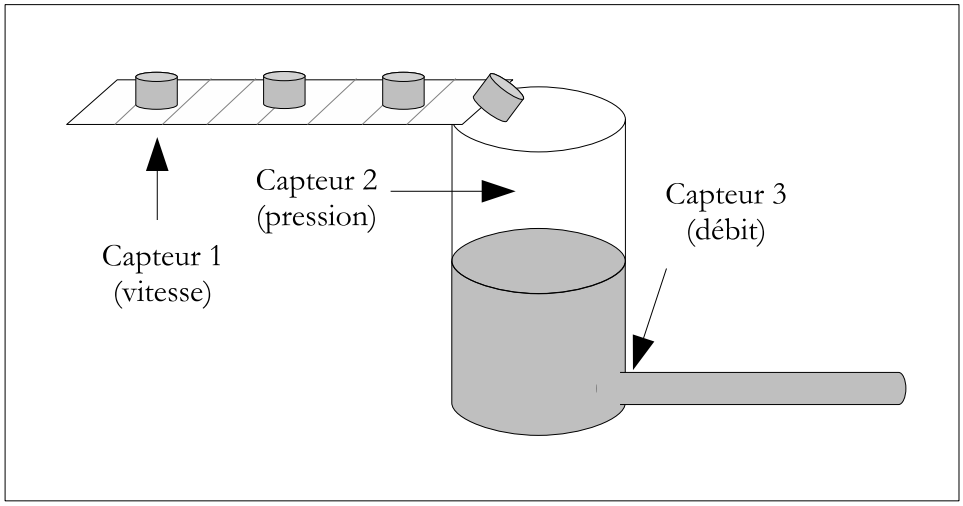
ESIAL

Maths Numériques

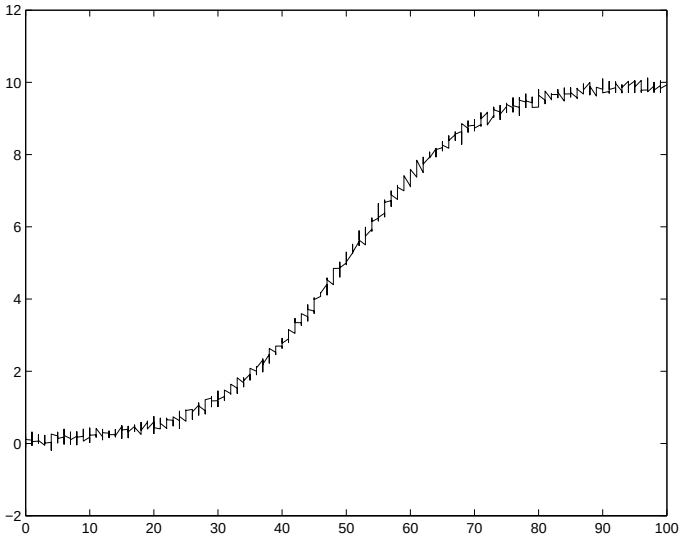
TP interpolation polynômiale et moindres carrés

L'entreprise BIKAREFOUL est spécialisée dans la surveillance des systèmes de production. En d'autres termes, elle est chargée de détecter les pannes, les ruptures ou la déterioration des systèmes de production de ses entreprises clientes. Pour cela, elle met en place un certain nombre de capteurs dans le système à surveiller et analyse en temps réel les données envoyées par ces capteurs. Prenons l'exemple d'une partie du système de production de la société PADEGAFFE consacrée à la transformation de blocs de pétrole solide en pétrole liquide :



Exercice 1 *Interpolation*

Vous êtes dans la peau d'un ingénieur informaticien fraîchement diplômé de l'ESIAL, brillante école d'ingénieurs située à Nancy et venez d'être accepté au sein de l'entreprise BIKAREFOUL. Votre première mission est de développer une application destinée à stocker dans une base de données les informations fournie par les capteurs :



Données envoyées par le capteur 1 pendant 100 secondes (1000 valeurs)

Seulement, vous vous apercevez que le stockage de l'intégralité des informations pose problème ! En effet, le système dispose de 120 capteurs différents fournissant chacun une valeur toute les 100 ms. L'intégralité des données sur une période d'un mois représente près de 26 millions de valeurs. Vous en concluez que la seule solution viable consiste à stocker uniquement une partie des données. Malheureusement, vos collègues du département de surveillance sont verts de rage en apprenant cela et vous somment de trouver une solution pour conserver le maximum d'information. Fort de votre enseignement de mathématiques numériques, vous avez l'ingénieuse idée de conserver dans la base de données une partie de l'information mais suffisament pour reconstituer les informations non retenues. Dans un premier temps vous optez alors pour le stockage dans la base du polynôme d'interpolation des données fournies par période de 100 secondes au lieu de stocker les 1000 valeurs initiales. Vous vous attelez alors à coder un programme fournissant le polynôme d'interplation p de degré $n-1$ s'écrivant dans la base de Newton :

$$p(t) = c_1 + c_2(t - x_1) + c_3(t - x_1)(t - x_2) + ... + c_n(t - x_1)(t - x_2)...(t - x_{n-1})$$

en utilisant l'algorithme vu en cours s'appuyant sur le calcul des différences divisées.

1. Pour cela, vous écrivez la fonction Matlab d'entête :

   ```
   function c = diff_div(x, y)
   ```

   calculant les différences divisées des points d'interpolation $(x_i, y_i)_{1 \leq i \leq n}$.

   On rappelle que pour $\{x_i\}_{1 \leq i \leq n}$ des points deux à deux distincts, les différences divisées de la fonction $f : x_i \mapsto y_i$ sont définies par récurrence :
   – différences divisées d'ordre 0 :
   $$f[x_i] = f(x_i) = y_i$$
   – différences divisées d'ordre 1 :
   $$f[x_i, x_j] = \frac{f(x_i) - f(x_j)}{x_i - x_j} = \frac{y_i - y_j}{x_i - x_j}$$
   – différences divisées d'ordre k :
   $$f[x_l, ..., x_{l+k}] = \frac{f[x_{l+1}, ..., x_{l+k}] - f[x_l, ..., x_{l+k-1}]}{x_{l+k} - x_l}, k \neq 0$$

2. puis la fonction d'entête :

   ```
   function p = horner(t, x, c)
   ```

   évaluant le polynôme d'interpolation (écrit donc dans une base de Newton) en les points contenus dans le vecteur `t` (la sortie `p` est de même dimension que `t` avec $p_i = p(t_i)$).

3. Pour calculer le polynôme d'interpolation correspondant aux 1000 premières données fournies par le capteur 1, vous commencez par importer les données via la commande :

   ```
   >> [t s] = import_('capteur_1.dat');
   ```

   et vérifiez grâce à la commande `plot(t,s)` que les données sont conformes au graphe de la page 1.

4. Maintenant que tout est prêt pour effectuer le calcul du polynôme, vous vous apercevez que vous n'avez toujours pas choisi le nombre de points n que vous souhaitez conserver (*i.e.* le degré $n-1$ du polynôme d'interpolation). Après quelques instants de reflexion, vous décidez de calculer les polynômes d'interpolation de degré $n = 0 ... 15$ et de conserver celui qui sera le "plus proche" de f (au sens des moindres carrés). Pour cela, on définit l'erreur d'interpolation d'ordre n par : $e_n(t) = f(t) - p_n(t)$ où p_n est le polynôme d'interpolation de degré n de la fonction f. Le polynôme p le "plus proche" sera celui pour lequel la norme de la fonction erreur est minimale :
   $$p = \min_{p_n} || e_n ||^2 = \min_{p_n} || f - p_n ||^2$$
   Vous écrivez alors l'algorithme permettant de déterminer le nombre de points d'interpolation (*i.e.* le degré du polynôme d'interpolation) qu'il est préférable de conserver :

```
for n=1:16
    k = floor(linspace(1,length(t),n)); % génère n entiers
                                        % équidistants entre 1 et length(t)
    x = t(k); % calcul des abscisses x_i des points d'interpolation
    ...
end
[norme_err n] = min(...)
...
subplot(2,1,1)
plot(x, y, 'bo', t, p, 'r-', t, s, 'g-')
legend('points d''interpolation','polynôme d''interpolation','données d''origine')
title('Interpolation')

subplot(2,1,2)
plot(t, e, 'b-')
title('% Erreur d''interpolation : e(t) = f(t)-p(t)')
```

5. Fier d'avoir apparemment déterminé le meilleur polynôme d'interpolation de f, une question vous intrigue. Est-il réellement judicieux de choisir les points d'interpolation équidistants ? Vous décidez d'essayer la répartition dite de Tchébychev, sur l'intervalle $[a, b]$, donnée par :

$$x_i = \frac{(a+b)}{2} + \frac{(b-a)}{2} \cos\left(\frac{(2i+1)\pi}{2(n+1)}\right), \quad i = 0, \ldots, n.$$

Vous remplacez alors dans le script précédent

```
k = floor(linspace(1,length(t),n));
```

par

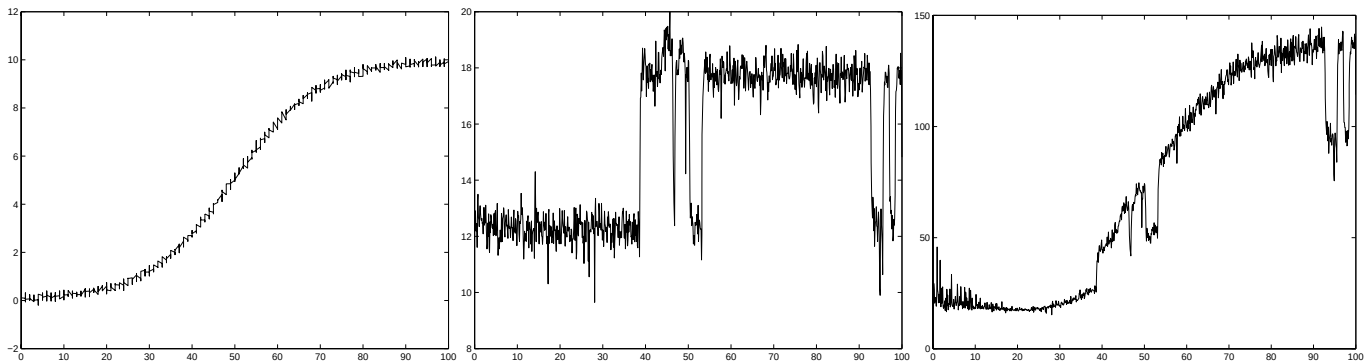
```
a = 1;
b = length(t);
k = floor((a+b)/2 + (b-a)/2*cos(((2*(1:n)-1)*pi)/(2*n)));
```

En concluez quant à l'intérêt de cette nouvelle répartition.

Exercice 2 *Moindres carrés*

La patrone de la société PADEGAFFE, Mme Mélusine ENFAILLITE, souhaiterait réduire le nombre de capteurs car trop coûteux. En outre, elle est persuadée que le capteur 3 n'est pas indispensable et que le débit peut se calculer à l'aide des capteurs 1 et 2. Vous êtes donc chargé d'écrire un algorithme permettant de déterminer la relation entre les données fournies par le capteur 3 et celles fournies par les capteurs 1 et 2.

1. Vous décidez à cet effet de travailler sur un échantillon de 1000 données (comme précédemment) :



```
[t c1] = import_('capteur_1.dat');
[t c2] = import_('capteur_2.dat');
[t c3] = import_('capteur_3.dat');
```

2. Ne sachant pas quel modèle adopter, vous élaborez un algorithme qui, à partir d'un modèle surparamétré, va déterminer les paramètres inutiles. Vous commencez pour cela par considérer le modèle suivant (composé de $p = 13$ paramètres) :

$$\widehat{c_3}(t) = \sum_{i=1}^{p} u_i.\phi_i(t)$$

avec

$\phi_1 : t \mapsto c_1(t)$ $\quad \phi_2 : t \mapsto c_2(t)$ $\quad \phi_3 : t \mapsto c_1^2(t)$ $\quad \phi_4 : t \mapsto c_2^2(t)$
$\phi_5 : t \mapsto c_1^3(t)$ $\quad \phi_6 : t \mapsto c_2^3(t)$ $\quad \phi_7 : t \mapsto c_1^4(t)$ $\quad \phi_8 : t \mapsto c_2^4(t)$
$\phi_9 : t \mapsto c_1(t).c_2(t)$ $\quad \phi_{10} : t \mapsto \sqrt{|c_1(t)|}$ $\quad \phi_{11} : t \mapsto \sqrt{|c_2(t)|}$ $\quad \phi_{12} : t \mapsto \ln|c_1(t)|$
$\phi_{13} : t \mapsto \ln|c_2(t)|$

Vous estimez alors les coefficients u_i du modèle par la méthode des moindres carrés et considérez que le paramètre i doit être retiré du modèle si $|\, u_i \,| < \varepsilon$ avec $\varepsilon = 0.02$. (Pour estimer ces coefficients, vous commencez par formuler le problème sous la forme standard :

$$\min_{u \in \mathbb{R}^p} || \, A.u - c_3 \, ||^2$$

puis résolvez les équations normales

$$A^T A u = A^T c_3$$

à l'aide des deux fonctions dont vous disposez (`factorisation_lu` et `descente_remontee`))

3. Après avoir déterminé un modèle pertinent et les coefficients correspondants, vous décidez de vérifier de visu la correspondance entre votre estimation et la courbe de données de c3.

```
... % c3_chapeau = estimation de c3 en fonction de c1 et c2
    % par la méthode des moindres carrés

subplot(2,1,1)
plot(t, c3_chapeau, 'r-', t, c3, 'g-')
legend('Approximation de c_3','c_3')
title('Approximation de c_3 par la méthode des moindres carrés')

subplot(2,1,2)
plot(t,abs(c3_chapeau-c3));
title('Erreur = |c3_chapeau - c3|')
```