Méthodes numériques : Notes complémentaires

Département de génie électrique et de génie informatique
Faculté de génie
Université de Sherbrooke

Automne 2019

Note : En vue d'alléger le texte, le masculin est utilisé pour désigner les femmes et les hommes.

Document Méthodes_Numériques_Notes_JdeL.docx
Rédigé par JdeL 08 juin 2008
Révisé par JdeL 18 juin 2009 avec contributions de JF Trégouët
Révisé par JdeL 20 mai 2010
Révisé par JdeL 04 mai 2013
Révisé par JdeL 12 juillet 2013
Révisé par JdeL 10 mai 2014 Version 6
Révisé par JdeL 28 mars 2016 Version 7
Révisé par JdeL 25 mars 2017 Version 8 Repris 14 novembre 2019

Table des matières

1 APPROXIMATION DE DONNEES 5

1.1 APPROXIMATION DISCRETE 5
1.1.1 Concepts de base 5
1.1.2 Approximation de données avec fonction à 2 paramètres 9
1.1.3 Qualité de l'approximation 11
1.1.4 Approximation de données avec une fonction mixte à M paramètres 13
1.1.5 Méthode de la projection orthogonale avec la pseudo-inverse 15
1.1.6 Cas de l'interpolation 18
1.2 APPROXIMATION CONTINUE 21
1.2.1 Les équations normales 21
1.2.2 Approximation polynomiale de Taylor 22
1.2.3 Approximation minimax polynomiale aux nœuds de Tchebychev 23

2 INTERPOLATION DE DONNEES 26

2.1 LE POLYNOME D'INTERPOLATION PAR INVERSION DE MATRICE 26
2.2 LE POLYNOME D'INTERPOLATION DE LAGRANGE 30
2.3 SPLINES CUBIQUES 31

3 DIFFERENTIATION NUMERIQUE 33

3.1 DIFFERENCE AVANT ET DIFFERENCE ARRIERE 33
3.2 UTILISATION DES DERIVEES DANS LE CALCUL D'ERREUR D'INTEGRATION NUMERIQUE 34
3.3 DIFFERENTIATION PAR LA FORMULE DE TUSTIN 35
3.4 DIFFERENCES CENTRALES 35

4 INTEGRATION NUMERIQUE 37

4.1 METHODE TRAPEZOÏDALE 37
4.1.1 Développement de la formule 37
4.1.2 Calcul de l'erreur d'approximation 38
4.2 METHODE DE SIMPSON 39
4.2.1 Développement de la formule 39
4.2.2 Calcul de l'erreur d'approximation 40
4.3 METHODE DE GAUSS 40
4.3.1 Développement de la formule, version normalisée 40
4.3.2 Développement de la formule, version non normalisée 41

5 SOLUTION NUMERIQUE D'EQUATIONS NONLINEAIRES 43

5.1 LA METHODE DE BISSECTION 43
5.2 LA METHODE DE NEWTON-RAPHSON 44
5.3 RACINES MULTIPLES 45
5.4 LA METHODE DE LA SECANTE 46

6 SOLUTION NUMERIQUE D'EQUATIONS DIFFERENTIELLES 48

6.1 DEFINITION 48
6.2 RAPPEL DES METHODES D'INTEGRATION DE EULER 48
6.3 ERREUR LOCALE ET ERREUR GLOBALE 49
6.4 CONVERGENCE ET STABILITE 50
6.5 LA METHODE RUNGE-KUTTA ET ODE45 52

NOTES IMPORTANTES

- **Les lettres « E » suivies d'un numéro représentent des exercices à faire en classe.**

- **Les lettres « D » suivies d'un numéro représentent des démonstrations mathématiques à consulter et à comprendre. Ces démonstrations sont données en annexe, à la fin du document.**

- **Les exercices seront faits avec les outils numériques de MATLAB.**

- **Le but de ces exercices est de coder des relations mathématiques et des outils numériques sur MATLAB afin de développer les compétences en méthodes numériques. L'utilisation des outils symboliques de MATLAB pour résoudre les exercices est strictement interdite. De la même façon, l'usage des fonctions MATLAB existantes qui intègrent déjà ces méthodes numériques ne sont pas permises. Les outils symboliques et les fonctions MATLAB peuvent cependant être utilisés pour valider une réponse déjà obtenue avec les outils numériques enseignés dans ce document.**

- **Il faut éviter au maximum les opérations 'manuelles' (lecture de données sur graphique, transferts de données manuels vers MATLAB, etc.) pour maintenir la précision numérique des opérations MATLAB.**

1 APPROXIMATION DE DONNÉES

Définition et utilité

- *Approximation de données* (*curve fitting*) : problème algébrique qui consiste à développer et résoudre les équations algébriques permettant de représenter des données expérimentales ou une relation fonctionnelle complexe par une fonction continue simple et facile à évaluer. Contrairement à l'*interpolation* (prochain chapitre), on ne tente pas ici de représenter exactement les données puisque, typiquement, elles contiennent du bruit et un lissage des données est requis.
- Elle permet de trouver la tendance de données expérimentales tout en excluant le bruit de mesure.
- Elle permet une représentation simplifiée d'une fonction difficile à évaluer, ce qui permet un calcul plus rapide de la fonction ou d'en déduire plus facilement certaines propriétés (e.g. points minimal/maximal).

Types d'approximation

- Il y a deux approches fondamentales selon la nature des données :
  - approximation de données discrètes ⇒ *approximation discrète*
    la variable dépendante est disponible en des points discrets de la variable indépendante
  - approximation de données continues ⇒ *approximation continue*
    la variable dépendante est disponible de façon continue en fonction de la variable indépendante

1.1 Approximation discrète

Définition

- Une fonction *g(x)* fait l'approximation d'un ensemble de paires de données $(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$ si son tracé se rapproche de ces points, sans nécessairement passer par ces points, tout en minimisant un critère d'erreur. Plusieurs choix existent pour la fonction *g(x)* ; il faut choisir la *meilleure* candidate.
- Problème à résoudre: transformer ce problème géométrique en un problème algébrique et le solutionner.

1.1.1 Concepts de base

Le critère d'erreur à minimiser est le critère du moindre carré, défini comme suit :

- L'erreur de l'approximation *g(x)* à un point x_n, y_n est donnée par : $\delta_n = g(x_n) - y_n$
- La somme des erreurs au carré sur tous les points est appelée l'*erreur quadratique E* :

$$E = \sum_{n=1}^{N} \delta_n^2 = \sum_{n=1}^{N} [g(x_n) - y_n]^2 \qquad \textbf{(1)}$$

- La somme des erreurs au carré est un des meilleurs indicateurs de la qualité d'une approximation.
- Une petite valeur de *E* implique que la fonction *g(x)* représente "bien" les données (x_n, y_n), $n = 1, N$.

Approche générale pour le design de la fonction *g(x)*:

- identifier une fonction candidate *g(x)*, avec *M* coefficients libres a_i à déterminer
- trouver l'équation algébrique de la somme des erreurs au carré : *E*
- calculer les conditions pour avoir un minimum de *E* i.e. $\frac{\partial E}{\partial a_i} = 0, i = 1, \ldots, M$
- ces équations sont appelées les *équations normales*
- solutionner ces *équations normales* pour les *M* coefficients libres a_i de la fonction *g(x)*
- reconstruire l'approximation et vérifier la qualité de l'approximation
- recommencer avec une autre fonction candidate si le résultat n'est pas satisfaisant

Deux exemples ci-dessous démontrent l'application de cette approche.

D1 : Développer l'équation normale pour l'approximation linéaire simple : $g(x) = mx$

- Cette fonction candidate possède un seul paramètre à déterminer (m) donc $M=1$
- Pour N données à approximer, l'erreur quadratique est donnée par :

$$E = \sum_{k=1}^{N} [g(x_k) - y_k]^2 = \sum_{k=1}^{N} [mx_k - y_k]^2$$

- On trouve le minimum de E par rapport au coefficient m :

$$\frac{dE}{dm} = 0 \qquad \Rightarrow \qquad \sum_{k=1}^{N} 2[mx_k - y_k]\, x_k = 0$$

- On veut trouver la solution pour m. On sépare les deux termes de la sommation :

$$2m \sum_{k=1}^{N} x_k^2 - 2 \sum_{k=1}^{N} y_k x_k = 0$$

- On trouve la solution pour m :

$$m = \frac{\sum_{k=1}^{N} y_k x_k}{\sum_{k=1}^{N} x_k^2}.$$

- En remplaçant cette valeur de m dans la fonction $g(x) = mx$, on obtient la meilleure ligne droite qui passe par les points (x_n, y_n), $n = 1, N$.

D2 : Développer les équations normales pour l'approximation linéaire générale : $g(x) = mx + b$

- On généralise le problème précédant pour avec une ligne droite qui ne passe pas nécessairement à l'origine, donc avec 2 paramètres à déterminer : m et b ($M = 2$).
- L'erreur quadratique est donnée par :

$$E = \sum_{k=1}^{N} [g(x_k) - y_k]^2 = \sum_{k=1}^{N} [mx_k + b - y_k]^2$$

- Il faut minimiser E par rapport aux deux paramètres m et b:

$$\frac{\partial E}{\partial m} = 0 \qquad \Rightarrow \qquad \sum_{k=1}^{N} 2[mx_k + b - y_k]\, x_k = 0$$

$$\frac{\partial E}{\partial b} = 0 \qquad \Rightarrow \qquad \sum_{k=1}^{N} 2[mx_k + b - y_k] = 0$$

- Les deux équations normales doivent être solutionnées pour m et b :

$$m \sum_{k=1}^{N} x_k^2 + b \sum_{k=1}^{N} x_k = \sum_{k=1}^{N} y_k x_k \qquad\qquad m \sum_{k=1}^{N} x_k + bN = \sum_{k=1}^{N} y_k$$

- On réécrit sous forme d'équation matricielle (l'ordre des équations est inversé ci-dessous):

$$\begin{bmatrix} N & \sum_{k=1}^{N} x_k \\ \sum_{k=1}^{N} x_k & \sum_{k=1}^{N} x_k^2 \end{bmatrix} \begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{N} y_k \\ \sum_{k=1}^{N} y_k x_k \end{bmatrix}$$

- La solution est donc :

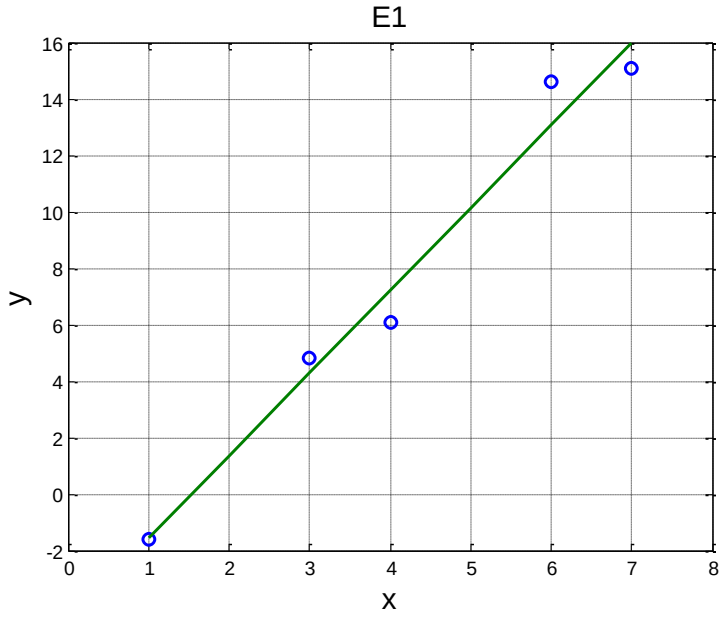
$$\begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} N & \sum_{k=1}^{N} x_k \\ \sum_{k=1}^{N} x_k & \sum_{k=1}^{N} x_k^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{N} y_k \\ \sum_{k=1}^{N} y_k x_k \end{bmatrix}$$

- En utilisant ces valeurs de *m* et *b* dans l'équation d'approximation $g(x) = mx + b$, on obtient la meilleure approximation au sens du moindre carré des données (x_n, y_n), $n = 1, N$.

E1 Trouver l'approximation linéaire $g(x) = mx + b$, au sens du moindre carré, des données suivantes :

x_n	1.0	3.0	4.0	6.0	7.0
y_n	-1.6	4.8	6.1	14.6	15.1

Réponse : $g(x) = 2.93\,x - 4.49$. Voir aussi la figure ci-dessous.



NOTE MATLAB :

- La fonction **`inv`** est utilisée pour inverser une matrice.
- La fonction MATLAB **`sum`** est utilisée pour calculer les sommes. On peut aussi utiliser le produit matriciel correspondant. Par exemple, pour calculer l'erreur quadratique $E = \sum_{n=1}^{N} [g_n - y_n]^2$, on peut utiliser :

`sum((g-y).*(g-y))` ou **`(g-y)'*(g-y)`** si g et y sont des colonnes.

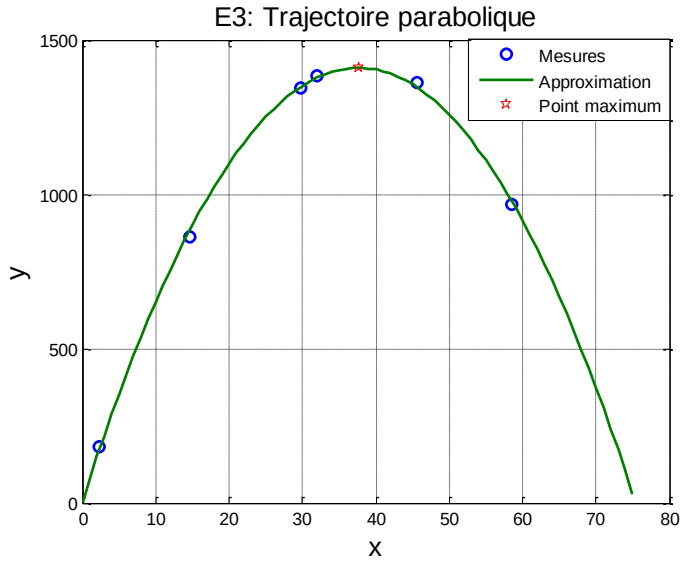
E2 En vous inspirant des démonstrations D1 et D2 ci-dessus, développer les équations normales pour une fonction parabolique$g(x) = a_2x^2 + a_1x + a_0$. À noter qu'il y a *M=3* paramètres à déterminer, donc 3 équations normales à développer.
Réponse :

$$\begin{bmatrix} N & \sum x_n & \sum x_n^2 \\ \sum x_n & \sum x_n^2 & \sum x_n^3 \\ \sum x_n^2 & \sum x_n^3 & \sum x_n^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum y_n \\ \sum y_n x_n \\ \sum y_n x_n^2 \end{bmatrix}$$

* * * * *

E3 À la fin de sa phase de propulsion, la fusée d'un feu d'artifice décrit approximativement une trajectoire parabolique (on néglige la trainée aérodynamique). Quelques mesures de son altitude *h* en fonction de son déplacement horizontal *x* ont été obtenues dans le tableau ci-dessous. Écrire un code MATLAB qui calcule la fonction parabolique *h(x)* représentant la trajectoire et qui en fait le graphique. Calculer aussi la hauteur maximum de sa trajectoire et calculer à quel point de son déplacement *x* ce maximum se trouve. NOTE : Les coordonnées du point maximum doivent être calculées sur MATLAB et non pas lues sur le graphique.

x_n	2.3	14.7	29.7	31.9	45.7	58.6
h_n	184	860	1345	1385	1360	965



Réponse : (x_{max} , h_{max}) = (37.66, 1409)

* * * * *

NOTE MATLAB :

- Il faut réaliser qu'il y a deux séries de points qui sont utilisés sur MATLAB : (1) les données originales discrètes *(x_n, y_n), n = 1, N* et (2) les données utilisées pour générer le graphique de l'approximation *g(x)*.
- Dans le 1er cas, on n'a que des valeurs discrètes *(x_n, y_n)*; on les met sur le graphique avec des symboles.
- Dans le 2e cas, on peut générer le graphique avec la résolution voulue puisqu'on a accès à une fonction *g(x)* que l'on peut évaluer au nombre de points désirés et mettre sur le graphique avec une ligne continue.
- Dans le code MATLAB, il est recommandé d'utiliser par exemple **`x,y`** pour les premières valeurs (discrètes) et **`xx,yy`** pour les deuxièmes valeurs (continues), celles-ci ayant un intervalle entre les points assez petit pour créer une courbe lisse sur MATLAB.

1.1.2 Approximation de données avec fonction à 2 paramètres

- L'approximation linéaire générale $g(x) = mx + b$ est facile et rapide d'utilisation. C'est pourquoi elle est souvent généralisée à d'autres fonctions candidates non linéaires ayant deux paramètres. Cette généralisation consiste à transformer une fonction non linéaire à deux paramètres en une fonction d'approximation linéaire générale de la forme $g(x) = mx + b$. Une fois l'approximation linéaire obtenue, on applique la transformation inverse pour régénérer la fonction non linéaire originale. Voici le problème.

- Dans une approximation de type « boîte noire », on recherche une fonction qui semble bien représenter la tendance des données à approximer et on ajuste ses paramètres pour minimiser l'erreur entre les données et la fonction. Pour ce faire, les fonctions non linéaires $g(x)$ à deux paramètres les plus utilisées sont :

  - exponentielle : $g(x) = \alpha e^{\beta x}$
  - puissance : $g(x) = \alpha x^{\beta}$
  - logarithmique : $g(x) = \alpha + \beta \ln x$
  - réciproque : $g(x) = \alpha + \frac{\beta}{x}$
  - hyperbolique : $g(x) = \frac{\alpha}{\beta + x}$
  - hyperbolique-linéaire : $g(x) = \frac{\alpha x}{\beta + x}$

- Le choix de la fonction candidate non linéaire dépend de la tendance des données à approximer. On choisit la fonction qui semble mieux représenter la tendance des mesures obtenues.
- Dans une approche « boîte grise », on peut se fier aussi au principe physique à la base des mesures, si ce principe est connu. Par exemple, il peut être démontré par l'équilibre hydrostatique que la densité de l'atmosphère a une variation de forme généralement exponentielle en fonction de l'altitude. Ainsi, l'utilisation de la forme $g(x) = \alpha e^{\beta x}$ représenterait bien des mesures de densité en fonction de l'altitude.

- Le problème d'approximation par moindre carré d'une fonction candidate non linéaire $g(x)$ avec deux paramètres α et β est :

$$\text{Minimiser } E = \sum_{n=1}^{N} [g(x_n) - y_n]^2 \text{ par la solution des } \textit{équations normales} : \frac{\partial E}{\partial \alpha} = 0, \frac{\partial E}{\partial \beta} = 0.$$

- Si on appliquait directement les équations normales $\frac{\partial E}{\partial \alpha} = 0$, $\frac{\partial E}{\partial \beta} = 0$ aux fonctions candidates non linéaires ci-dessus, les équations qui en résulteraient seraient non linéaires et souvent impossibles à résoudre comme on a pu le faire dans les démonstrations D1 et D2. On peut éviter ce problème par une transformation.

- En effet, il est possible de transformer ces équations non linéaires $y = g(x)$ sous une forme linéaire générale $Y = mX + b$ et utiliser les solutions aux équations normales de D2 pour calculer la solution pour α et β.

- NOTE SUR LA NOTATION : Les lettres majuscules X, Y pour les variables et les lettres romaines m, b pour les paramètres sont utilisés dans le domaine des variables transformées dans le domaine linéaire alors que les lettres minuscules x, y pour les variables et les lettres grecques α, β pour les paramètres sont utilisés dans le domaine des variables originales du domaine non linéaire.

- Pour faire la transformation, on utilise par exemple la fonction logarithmique qui permet de transformer les multiplications en additions, les divisions en soustractions et les puissances en multiplications et divisions.

- Exemple : Supposons que des données de mesures bruitées ont une tendance exponentielle. On veut donc une approximation de la forme : $y = g(x) = \alpha e^{\beta x}$. En prenant le logarithme naturel de chaque côté, on obtient : $\ln y = \ln \alpha + \beta x$ qui est exactement de forme linéaire générale $Y = mX + b$. En faisant la comparaison, on voit que la correspondance est :

$$Y = \ln y, \quad m = \beta, \quad X = x, \quad b = \ln \alpha.$$

- On applique donc la méthode analytique D2 avec les données transformées $Y_k = \ln y_k$, $X_k = x_k$ pour trouver m, b et on applique les transformations inverses pour trouver α, β :

$$\beta = m, \quad \alpha = e^b.$$

- Le tableau suivant donne la transformation des fonctions candidates non linéaires vers la forme $Y = mX + b$.

$y = g(x)$	$Y = mX + b$	$X =$	$Y =$	$b =$	$m =$	$\alpha =$	$\beta =$
$y = \alpha e^{\beta x}$	$ln\, y = ln\, \alpha + \beta x$	x	$\ln y$	$\ln \alpha$	β	e^b	m
$y = \alpha x^{\beta}$		$\ln x$	$\ln y$				
$y = \alpha + \beta \ln x$	$y = \alpha + \beta(\ln x)$						
$y = \alpha + \frac{\beta}{x}$	$y = \alpha + \beta\left(\frac{1}{x}\right)$	$\left(\frac{1}{x}\right)$	y	α	β	b	m
$y = \frac{\alpha}{\beta + x}$	$y = \frac{\alpha}{\beta} + \frac{-1}{\beta}(xy)$					$\frac{-b}{m}$	$\frac{-1}{m}$
$y = \frac{\alpha x}{\beta + x}$				α	$-\beta$		

La première ligne du tableau démontre l'exemple décrit précédemment où la fonction $y = \alpha e^{\beta x}$ est transformée en une équation linéaire de la forme $Y = mX + b$ de façon à ce que la pente m soit β et l'ordonnée à l'origine b soit ln α. En transformant les données originales avec le logarithme, on peut utiliser les équations de D2 pour trouver la pente m et l'ordonnée à l'origine b et ensuite trouver α et β dans la fonction d'approximation finale $y = \alpha e^{\beta x}$.

E4 Remplir les 20 cases manquantes du tableau ci-dessus.

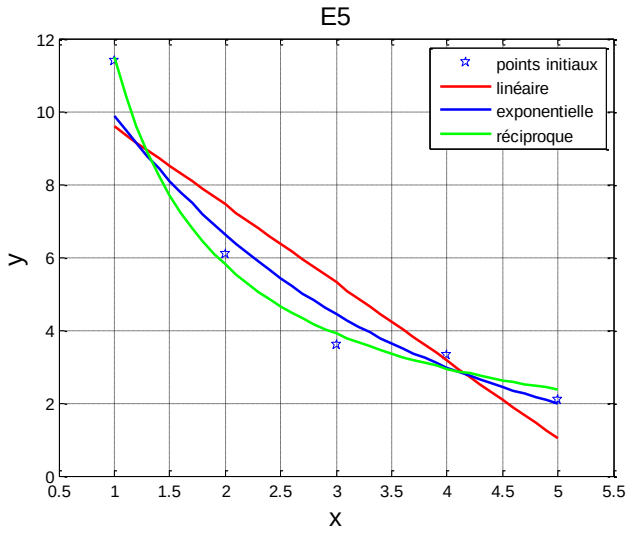
E5 Faire l'approximation des données suivantes avec les fonctions candidates :
(a) $y = mx + b$, (b) $y = \alpha e^{\beta x}$ (c) $y = \alpha + \frac{\beta}{x}$. Calculer et comparer l'erreur dans chaque cas.

x_n	1.0	2.0	3.0	4.0	5.0
y_n	11.4	6.1	3.6	3.3	2.1

Réponses :
(a) $m = -2.140$, $b = 11.72$, $E = 9.184$
(b) $\alpha = 14.75$, $\beta = -0.3998$, $E = 3.394$
(c) $\alpha = 0.0992$, $\beta = 11.39$, $E = 0.3907$

Le calcul de l'erreur E quadratique a été présenté à l'équation (1).

Note : Ce problème a été créé avec la fonction (c) $y = \alpha + \beta / x = 12 / x + erreurs$ et on a obtenu $y = 0.0992 + 11.39 / x$. C'est bien cette fonction qui donne la plus petite erreur quadratique (E). Comparer avec les résultats ci-dessous. Il est donc important de bien analyser la tendance des données afin de choisir la bonne fonction candidate pour l'approximation.



1.1.3 Qualité de l'approximation

- L'erreur quadratique *E*, équation (1), est l'indice de performance qui est utilisé pour qualifier la précision de l'approximation. Sa forme mathématique permet de rapidement développer les équations normales. Cependant, cette erreur est une somme d'erreurs individuelles mises au carré et elle grandit avec le nombre de données. Pour qualifier la qualité de l'approximation, il serait préférable de prendre la moyenne de ces erreurs au carré. De plus, vu que cette moyenne représente des erreurs au carré, il est aussi préférable de prendre la racine carrée pour obtenir une erreur moyenne dans les unités de mesure originales.

- Donc, bien que l'erreur quadratique *E* soit utile dans la formulation mathématique du problème, la racine carrée de la moyenne de ces erreurs, dénotée *erreur RMS* (pour '*root mean square error*'), donne une meilleure mesure physique de l'erreur d'approximation :

$$RMS = \sqrt{\frac{1}{N}E} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}[g(x_n) - y_n]^2} \qquad \textbf{(2)}$$

- Une autre façon de déterminer la qualité de l'approximation est l'utilisation d'un indice qui normalise les erreurs autour de valeurs moyennes. C'est le *coefficient de détermination* R^2. C'est un rapport de variances :

$$R^2 = \frac{\sum_{n=1}^{N}[g(x_n) - \bar{y}]^2}{\sum_{n=1}^{N}[y_n - \bar{y}]^2} \qquad \text{avec} \qquad \bar{y} = \frac{1}{N}\sum_{n=1}^{N} y_n \qquad \textbf{(3)}$$

- Une bonne approximation correspond à $R^2 \approx 1$, une mauvaise à $R^2 \approx 0$.

Attention :

- $R^2 \approx 0$ indique toujours une mauvaise approximation.
- $R^2 \approx 1$ indique une bonne approximation seulement si le nombre de points *N* est suffisamment plus grand que le nombre de paramètres *M* dans l'équation candidate *g(x)*.
- La différence $N - M$ est le nombre de degré de liberté de l'approximation. Plus cette différence est grande, plus les données seront lissées. Un minimum de 3 est recommandé. Plus *M* se rapproche de *N*, plus la courbe d'approximation tend à interpoler les *N* données.
- Quand $M = N$, il y a interpolation des données par *g(x)* : l'erreur quadratique *E* est nulle et le coefficient de détermination R^2 est égal à 1. Cependant, si les données originales sont des mesures bruitées, l'interpolation n'est pas désirée. De plus, avec l'interpolation, le comportement de *g(x)* entre les points initiaux peut être complètement

inacceptable. Ce comportement non désiré, appelé le '*polynomial wiggle problem*' est démontré plus loin les exercices E6 et E7.

- Recommandation : Une bonne qualité de l'approximation de données bruitées vise $\boldsymbol{R^2 \approx 1}$ **et** $\boldsymbol{N - M \geq 3}$.

Vérification :

- Pour l'exercice E1, vérifier que E = 4.6532, RMS = 0.9647, R^2 = 0.9767 et $N - M = 3$.
- Pour l'exercice E3, vérifier que E = 1301.2, RMS = 14.726, R^2 = 0.9988 et $N - M = 3$.
- Pour l'exercice E5(a), vérifier que E = 9.184, RMS = 1.3553, R^2 = 0.8330.
- Pour l'exercice E5(b), vérifier que E = 3.394, RMS = 0.8239, R^2 = 0.9498.
- Pour l'exercice E5(c), vérifier que E = 0.391, RMS = 0.2795, R^2 = 0.9929.

NOTE MATLAB :

- L'erreur quadratique sur MATLAB peut être calculée avec la fonction **`sum`** : **`E = sum((g-y).^2)`** mais l'utilisation du produit scalaire est plus efficace et équivalente : **`E = (g-y)'*(g-y)`**.
- L'erreur RMS peut être calculée avec la fonction **`mean`** : **`err_rms = sqrt(mean((g-y).*(g-y)))`** ou avec la fonction **`rms`** de MATLAB : **`err_rms = rms(g-y)`**.

1.1.4 Approximation de données avec une fonction mixte à *M* paramètres

- Dans D1, D2 et E2, les équations normales pour les cas *M = 1* (linéaire simple), *M = 2* (linéaire générale) et *M=3* (parabolique) ont été développées dans le cas où la forme générale de l'approximation *g(x)* était :

$$g(x) = \sum_{m=1}^{M} a_m x^{m-1}, \quad M = 1, 2, 3$$

- Attention :
  - On utilise ici la sommation de *n = 1, .. N* quand la sommation est sur les données (x_n, y_n).
  - On utilise ici la sommation de *m = 1,…M* pour les termes de la fonction d'approximation *g(x)*.

- Les fonctions élémentaires qui composent la fonction d'approximation *g(x)* s'appellent les *fonctions de base* ('*basis functions*'). De la même façon qu'un vecteur générique dans l'espace en 3 dimensions peut être représenté par la somme de 3 vecteurs unitaires multipliés par les composantes du vecteur :

$$\vec{v} = v_1\vec{\imath} + v_2\,\vec{\jmath} + v_3\vec{k}$$

  une fonction générique *g(x)* peut être représentée par une somme de fonctions de base $\phi_m(x)$ (équivalent aux vecteurs unitaires) multipliés par des coefficients a_m (équivalents aux composantes du vecteur) :

$$g(x) = a_1\phi_1(x) + a_2\phi_2(x) + a_3\phi_3(x) + \cdots\ a_M\phi_M(x) = \sum_{m=1}^{M} a_m\phi_m(x)$$

- La seule différence est que l'espace physique des vecteurs est limité à 3 dimensions alors que l'espace des fonctions de base peut être infini.
- Un exemple est une série de Fourier dont les fonctions de bases sont $\phi_m(x) \sim \sin\left(\frac{2\pi mx}{L}\right), \cos(\frac{2\pi mx}{L})$.
- Un autre exemple est la série de MacLaurin dont les fonctions de bases sont $\phi_m(x) = x^{m-1}$.
- La série de Taylor est le cas de la série de MacLaurin décentrée : $\phi_m(x) = (x - x_c)^{m-1}$.

- Ainsi, dans D1, D2 et E2, les fonctions de bases sont de la forme $\phi_m(x) = x^{m-1}$.

- On généralise maintenant D1, D2 et E2 de deux façons :
  - l'ordre de l'approximation (=nombre de paramètres à déterminer a_m) est générique : *M*
  - les *M* fonctions d'approximation sont aussi génériques : $\phi_m(x), m = 1, \ldots M$.

- Les fonctions de base $\phi_m(x)$ sont au choix de l'utilisateur, pourvu qu'elles ne soient pas des combinaisons linéaires l'une des autres (i.e. elles doivent être indépendantes).

  - Dans un problème de nature périodique dont on connait la période, on prendra des fonctions trigonométriques (série de Fourier).
  - Dans le cas d'un problème à tendance exponentielle, on prendra des fonctions exponentielles.
  - Dans l'incertitude, en se basant sur le théorème de Taylor, les fonctions en puissance de la forme $\phi_m(x) = x^{m-1}$ sont toujours une option à considérer.
  - Une autre solution possible est d'utiliser un mélange de ces fonctions de base, générant ainsi une approximation mixte.

- Dans le cas général d'approximation, le problème est donc:

- Minimiser $E = \sum_{n=1}^{N}[g(x_n) - y_n]^2$avec la fonction candidate :

$$g(x) = a_1\phi_1(x) + a_2\phi_2(x) + \cdots\ a_M\phi_M(x) = \sum_{m=1}^{M} a_m\phi_m(x) \qquad \textbf{(4)}$$

par la solution des *M* équations normales : $\frac{\partial E}{\partial a_m} = 0, m = 1, \ldots, M$

D3 : Démontrer que les *équations normales généralisées* pour l'approximation à M paramètres sont :

$$\begin{bmatrix} \langle\phi_1,\phi_1\rangle & \langle\phi_1,\phi_2\rangle & \cdots & \langle\phi_1,\phi_M\rangle \\ \langle\phi_2,\phi_1\rangle & \langle\phi_2,\phi_2\rangle & & \langle\phi_2,\phi_M\rangle \\ \vdots & & \ddots & \vdots \\ \langle\phi_M,\phi_1\rangle & \langle\phi_M,\phi_2\rangle & \cdots & \langle\phi_M,\phi_M\rangle \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} \langle\phi_1,y\rangle \\ \langle\phi_2,y\rangle \\ \vdots \\ \langle\phi_M,y\rangle \end{bmatrix} \qquad \textbf{(5)}$$

où les définitions suivantes ont été utilisées :

$$\langle\phi_i,\phi_j\rangle = \sum_{n=1}^{N} \phi_i(x_n)\ \phi_j\ (x_n) \qquad \langle\phi_i,y\rangle = \sum_{n=1}^{N} \phi_i(x_n)\ y_n \qquad \textbf{(6)}$$

La démonstration est donnée en détail dans l'annexe à ce document.

- L'équation (5) est de la forme :$\Phi A = \Psi$ avec :

$$\Phi = \begin{bmatrix} \langle\phi_1,\phi_1\rangle & \langle\phi_1,\phi_2\rangle & \cdots & \langle\phi_1,\phi_M\rangle \\ \langle\phi_2,\phi_1\rangle & \langle\phi_2,\phi_2\rangle & & \langle\phi_2,\phi_M\rangle \\ \vdots & & \ddots & \vdots \\ \langle\phi_M,\phi_1\rangle & \langle\phi_M,\phi_2\rangle & \cdots & \langle\phi_M,\phi_M\rangle \end{bmatrix} \quad A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} \quad \Psi = \begin{bmatrix} \langle\phi_1,y\rangle \\ \langle\phi_2,y\rangle \\ \vdots \\ \langle\phi_M,y\rangle \end{bmatrix}$$

et la solution est donc trouvée par inversion de matrice :

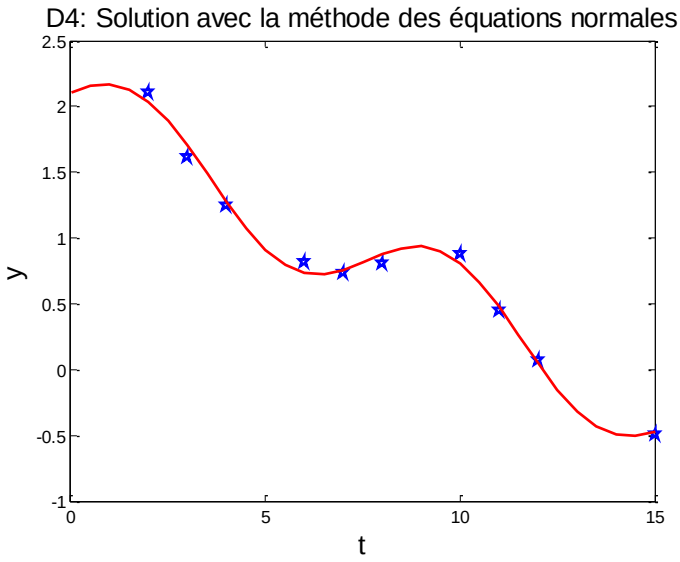
$$\Phi A = \Psi \quad \Longrightarrow \quad A = \Phi^{-1}\Psi.$$

- L'inverse existe si les fonctions de base sont indépendantes aux points x_n.
- Il est utile de comparer l'équation (5) avec le résultat de l'exercice E2 et de retrouver les *M* fonctions $\phi_m(x)$ qui y ont été utilisées.

D4 La trajectoire *y(t)* en mètres de la masse de la grue est une superposition d'une translation linéaire et d'une oscillation dont la période *P* est de 8 s. Une caméra a permis de capter quelques positions de la masse en fonction du temps et elles sont reportées dans le tableau ci-dessous. En utilisant MATLAB et les *équations normales généralisées*, faire le code qui permet de trouver l'équation approximative qui représente le mieux la trajectoire. Calculer l'erreur quadratique, l'erreur RMS et le coefficient R^2. Avec cette fonction, calculer la vitesse moyenne de la masse et l'amplitude des oscillations.

t_n	2	3	4	6	7	8	10	11	12	15
y_n	2.11	1.61	1.25	0.820	0.737	0.810	0.880	0.443	0.070	–0.493

Réponses : $y(x) = 1.9998 - 0.1538\ t + 0.1074\cos(2\pi/P) + 0.3434\sin(2\pi/P)$
E = 0.035, RMS = 0.0593, R = 0.993, $N - M$ = 6, vitesse = –15.4 cm/s, amplitude = 36 cm



- Cet exemple démontre l'utilisation de fonctions de base mixtes dans une approximation.
- Dans ce cas-ci, il fallait connaitre la période des fonctions trigonométriques. Si cette période n'était pas connue, ce problème pourrait se résoudre par exemple avec des fonctions en puissance de la forme $\phi_m(x) = x^{m-1}$. C'est ce qui est fait plus loin à l'exercice E7.

1.1.5 Méthode de la projection orthogonale avec la pseudo-inverse

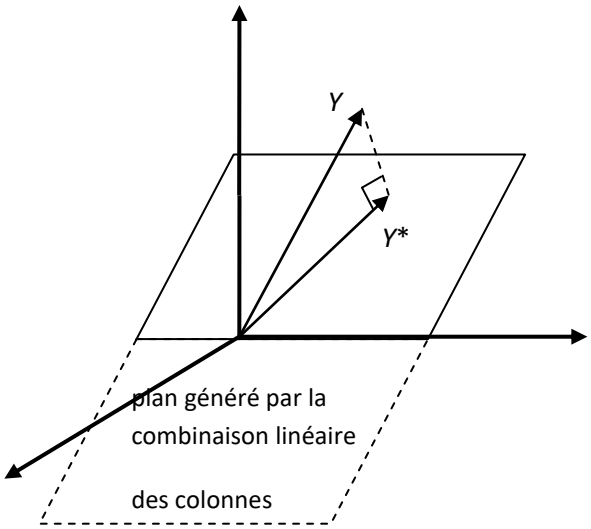
- Les équations normales développées jusqu'ici donnent une façon d'obtenir la solution pour les coefficients a_m d'une approximation. Il existe une autre façon de les obtenir : la méthode de la pseudo-inverse (ou méthode de la projection orthogonale).
- Ainsi, l'approximation par les moindres carrés peut être interprétée en termes de projection orthogonale.
- Pour expliquer ce principe, supposons le cas d'une approximation linéaire $y = a_1 + a_2 x$ avec trois points de mesure $x_n = -1, 0, +1$ pour lesquels on a mesuré $y_n = 0, 2, 5$, $n = 1, 2, 3$. Si on remplace les 3 coordonnées dans l'équation d'approximation $y = a_1 + a_2 x$, on obtient 3 équations pour les $N = 3$ points en fonction des $M = 2$ coefficients (a_1, a_2) à déterminer.
- Le système est donc sur-contraint : 3 équations et 2 paramètres inconnus. Il n'y pas de solution exacte d'où l'emploi du terme approximation pour 'passer près des points' sans pouvoir les interpoler.
- Les 3 équations peuvent s'écrire sous forme matricielle ainsi :

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = Y \Rightarrow \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & +1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 5 \end{bmatrix} \quad \textbf{(7)}$$

- Il faut noter que la matrice rectangulaire ci-dessus, de dimensions N x M = 3 x 2, n'est pas équivalente à la matrice carrée M x M de l'équation (5). Les N équations ci-dessus représentent l'*équation d'approximation* évaluée aux N points. Les équations (5) représentent les M équations normales issues de la minimisation du critère du moindre carré par dérivées partielles.
- Pour expliquer le principe de projection orthogonale, on peut réécrire la partie droite de l'équation (7) sous une forme où les coefficients inconnus forment une combinaison linéaire des colonnes de la matrice 3 x 2 :

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} a_1 + \begin{bmatrix} -1 \\ 0 \\ +1 \end{bmatrix} a_2 = \begin{bmatrix} 0 \\ 2 \\ 5 \end{bmatrix} = Y \quad \textbf{(8)}$$

- Ces deux colonnes $[1\ 1\ 1]^T$ et $[-1\ \ 0\ +1]^T$sont des vecteurs dans l'espace de dimension 3 mais, n'ayant que 2 vecteurs seulement, la combinaison linéaire avec les coefficients a_1 et a_2 sera projetée dans un plan dans l'espace de 3 dimensions, plan généré par ces vecteurs de base $[1\ 1\ 1]^T$ et $[-1\ \ 0\ +1]^T$.
- Ce plan est l'espace généré par les deux vecteurs de base.
- Cependant, les mesures $y_n = [0, 2, 5]^T$ forment un vecteur *Y* dans l'espace de dimension 3 qui, à cause des erreurs de mesures, n'est pas exactement dans cet espace généré par les vecteurs de base.
- Le mieux que les deux vecteurs de base $[1\ 1\ 1]^T$ et $[-1\ 0\ +1]^T$ peuvent réaliser par combinaison linéaire, c'est de générer un vecteur Y^* dans l'espace 2D qui est « le plus près possible » *Y* en 3D.
- Il sera démontré que la solution qui minimise l'erreur quadratique, Y^*, est la projection de *Y* sur ce plan.
- La figure suivante illustre la géométrie : les colonnes de la matrice 3 x 2 génèrent un espace à 2 dimensions qui est un plan dans l'espace à 3 dimensions. Le vecteur *Y* n'est pas dans ce plan en général. La projection de *Y* sur le plan, Y^*, est la meilleure approximation à *Y*.



- Les détails qui suivent vont démontrer comme on trouve cette projection.
- On généralise maintenant le problème géométrique simple en 3D ci-dessus en partant de l'équation (4). Avec *N* paires de points (x_n, y_n), il est possible d'écrire *N* équations avec les *M* inconnues de la forme :

$$\begin{bmatrix} g(x_1) \\ g(x_2) \\ \vdots \\ g(x_N) \end{bmatrix} = \begin{bmatrix} \sum_{m=1}^{M} a_m \phi_m(x_1) \\ \sum_{m=1}^{M} a_m \phi_m(x_2) \\ \vdots \\ \sum_{m=1}^{M} a_m \phi_m(x_N) \end{bmatrix} = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_M(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & & \phi_M(x_2) \\ \vdots & & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \cdots & \phi_M(x_N) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \textbf{(9)}$$

- On définit la matrice *P*, de dimension *N* x *M*, la matrice de coefficients *A*, de dimensions *M* x *1*, et la matrice *Y* de coordonnées y_n, de dimension *N* x *1*.

$$P = \begin{bmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_M(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & & \phi_M(x_2) \\ \vdots & & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \cdots & \phi_M(x_N) \end{bmatrix} = [P_1 \quad P_2 \quad \ldots \quad P_M] \qquad A = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} \quad \text{Y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \textbf{(10)}$$

- Les *M* colonnes de *P*, dénotées par P_i, sont formées par les *M* fonctions candidates évaluées aux *N* points et représentent les vecteurs de base dont la combinaison linéaire avec les a_m va former l'approximation polynomiale *g(x)* (en analogie à l'espace en 2 dimensions de l'exemple précédant où il n'y avait seulement que *M=2* colonnes pour former le plan en 2D). Avec ces définitions, l'équation (9) est réécrite:

$$\begin{bmatrix} g(x_1) \\ g(x_2) \\ \vdots \\ g(x_N) \end{bmatrix} = P \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \Longrightarrow \qquad PA = Y \qquad \textbf{(11)}$$

- Cette équation ressemble beaucoup aux équations normales, équations (5), de la forme $\Phi A = \Psi$. Cependant, il est important de bien voir les différences entre ces *équations normales*, équations (5), et les *équations de projection orthogonales* ci-dessus. En voici les principales différences :

  - Dans les équations normales, $\Phi A = \Psi$, la matrice Φ est carrée, de dimensions *M* x *M* et chaque élément contient une sommation de fonctions de base évaluées aux *N* points de mesures. La matrice Ψ est de dimension *M* x 1 et chaque élément contient aussi une sommation de fonctions de base évaluées aux *N* points de mesures.
  - Dans les équations orthogonales, $PA = Y$, la matrice P n'est pas carrée, elle est de dimensions *N* x *M* et chaque élément contient directement les fonctions de base évaluées aux *N* points de mesures. La matrice Y est de dimension *N* x 1 et chaque élément contient les coordonnées y_n directement.
  - La solution des équations normales, $A = \Phi^{-1} \Psi$ se calcule par inversion de la matrice Φ.
  - La solution des équations orthogonales, $PA = Y$ ne peut se calculer par l'inversion de la matrice *P* puisque celle-ci n'est pas carrée (*N* x *M*). Il faudra plutôt utiliser la pseudo-inverse de la matrice *P* tel que démontré ci-après.

- L'erreur quadratique de l'équation (1) peut être exprimée comme suit :

$$E = \sum_{n=1}^{N} [g(x_n) - y_n]^2 = [g(x_1) - y_1]^2 + [g(x_2) - y_2]^2 + \cdots + [g(x_N) - y_N]^2 \qquad \textbf{(12)}$$

  et, avec les définitions ci-dessus, il est possible de la transformer sous la forme d'une norme de vecteur:

$$E = \begin{bmatrix} g(x_1) - y_1 \\ g(x_2) - y_2 \\ \vdots \\ g(x_N) - y_N \end{bmatrix}^T \begin{bmatrix} g(x_1) - y_1 \\ g(x_2) - y_2 \\ \vdots \\ g(x_N) - y_N \end{bmatrix} = [PA - Y]^T [PA - Y] = \|PA - Y\|^2 \qquad \textbf{(13)}$$

- Par analogie au problème géométrique à 3 dimensions, le 'vecteur' *PA* est la combinaison des colonnes de *P* qui génèrent dans l'espace un 'plan' de dimension *M* < *N*. Dans cette analogie, le vecteur *PA* représente donc la solution *Y** de la figure.
- De retour à l'équation ci-dessus, minimiser l'erreur quadratique *E* est équivalent à minimiser la distance $\|PA - Y\| = \|Y * - Y\|$ entre le vecteur *Y**=*PA* et le vecteur *Y*, dans l'espace de dimension *N*.
- Se référant à la figure précédente, minimiser la distance entre les vecteurs *PA* et *Y* consiste à calculer le vecteur *Y** = *PA* qui est la projection du vecteur *Y* dans le 'plan' généré par les colonnes de *P*.
- La projection de *Y* sur ce plan est $Y^* = P^T Y$. Selon l'équation (11), cette projection est donc :

$$Y * = P^T Y = P^T PA \qquad \textbf{(14)}$$

- La solution pour les *M* coefficients *A* devient donc :

$$P^T PA = P^T Y \quad \Rightarrow \quad A = (P^T P)^{-1} P^T Y \tag{15}$$

- Il peut être vérifié que l'équation$P^T PA = P^T Y$ci-dessus correspond à l'équation (5) $\Phi A = \Psi$.

- La matrice :

$$P^{\perp} = (P^T P)^{-1} P^T \tag{16}$$

est appelée la *pseudo-inverse* de *P* et son existence est assurée en choisissant les colonnes de *P* pour qu'elles soient indépendantes.

D5 Refaire le problème D4 avec la projection orthogonale et vérifier que les coefficients a_m sont les mêmes.

NOTE MATLAB :

- Sur MATLAB, la pseudo-inverse d'une matrice P peut se calculer au long avec **`inv(P'*P)*P'`**.
- La fonction MATLAB **`pinv(P)`** est par contre un meilleur choix parce que certains problèmes numériques rencontrés avec l'usage de **`inv`** dans **`inv(P'*P)*P'`** sont évités avec **`pinv`**.
- Bien que **`pinv`** soit une fonction 'toute faite' très utile sur MATLAB, il est important de reconnaitre son origine qui démontre que cette inversion trouve la 'projection' multidimensionnelle d'une fonction dans l'espace généré par les fonctions de base choisies par le concepteur/conceptrice de l'approximation.

1.1.6 Cas de l'interpolation

- Jusqu'à maintenant, le problème d'approximation consistait en un problème de *N* équations à *M* inconnues où *N* > *M*. Ayant plus d'équations que de coefficients à déterminer, le système d'équations était sur-contraint et une solution exacte de ces équations était impossible. On ne pouvait qu'approximer.

- Dans le cas où *N* = *M*, le nombre de points et donc d'équations *N* est égal au nombre *M* de coefficients a_i. Le système n'est plus sur-contraint et il y a donc possibilité d'une solution exacte. Les points Y sont donc dans le 'plan' généré par les colonnes de *P* et la projection *Y** est en fait coïncidente avec le vecteur *Y*. La matrice *P* devient carrée et la pseudo-inverse de *P* devient l'inverse de *P* :

$$\text{Cas } M = N : P^{\perp} = (P^T P)^{-1} P^T = P^{-1} P^{-T} P^T = P^{-1} \tag{17}$$

La solution et l'erreur quadratique deviennent :

$$A = P^{-1} Y \qquad E = \|PA - Y\|^2 = \|Y * - Y\|^2 = 0 \tag{18}$$

La fonction $g(x)$ passe donc par toutes les paires de points (x_n, y_n), $n = 1, N$. Il n'y a plus de lissage, il y a **interpolation.** L'erreur quadratique est donc nulle. L'interpolation sera le sujet du prochain chapitre.

- Dans le cas de mesures bruitées, il a déjà été dit que l'interpolation n'est pas la solution à prendre. Le degré de filtrage ou de lissage des données dépend de l'ordre du polynôme d'approximation (et donc du nombre M de coefficients a_m à déterminer). De façon générale :
  - Plus *M* est petit par rapport à *N*, plus les données seront lissées, au point où de l'information utile dans le signal pourrait être filtrée par l'approximation.
  - Plus *M* se rapproche de *N*, plus l'approximation prend en compte les données présentes jusqu'au point où, quand $M = N$, toutes les informations dans les mesures sont conservées et il y a interpolation.
  - Une règle du pouce est de commencer avec $M \leq N - 3$ et d'ajuster au besoin.

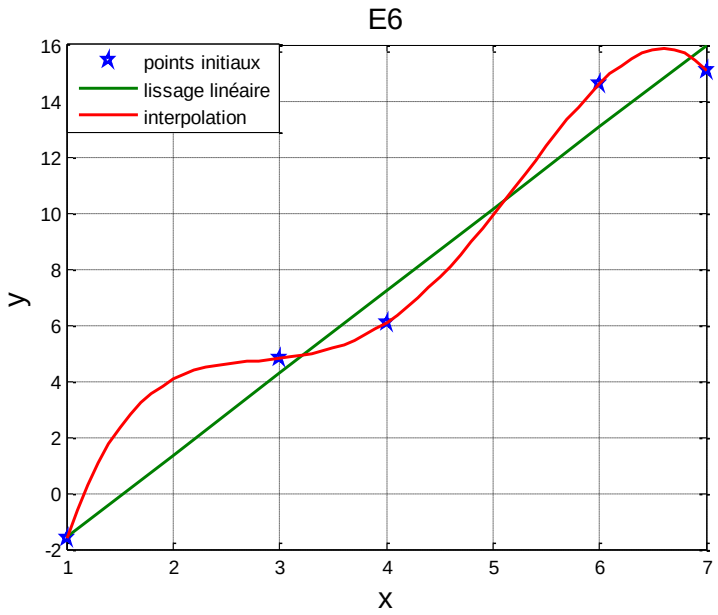
- Toute information sur le principe physique à l'origine des données bruitées doit aussi être prise en compte. Par exemple, si des mesures de position ont été créées par une accélération constante, l'utilisation d'un polynôme d'ordre 2 (fonction parabolique) serait le bon choix.

E6 Cet exemple donne un exemple où l'interpolation de mesures bruitées ne donne pas les résultats attendus. Reprendre les données de l'exercice E1. Utiliser la méthode de projection orthogonale pour:

(a) faire une approximation linéaire ($M = 2$), comme pour E1 (qui a été fait avec les équations normales : vérifier que les coefficients sont les mêmes),
(b) une interpolation à tous les points ($M = N$).

Calculer l'erreur RMS, le coefficient R^2 et la différence $N - M$ dans chaque cas. Il s'agit ici d'un contre-exemple qui démontre que, dans le cas de mesures bruitées, même si *RMS = 0* et *R=1.0* dans le cas $M = N$, cette approximation (interpolation en fait) interpole le bruit de mesure et est donc pire que l'approximation linéaire ($M = 2$) obtenue en (a) qui fait un meilleur lissage des données. Voir le graphique ci-dessous. Comme indiqué plus haut, il est important de maintenir $N - M \geq 3$ pour un lissage des données et éviter d'interpoler le bruit de mesure. Le cas avec $N - M = 3$ et R = 0.9767 représente donc une meilleure approximation de ces données.

Réponses : Cas avec $N - M = 3$ $(N = 5, M = 2)$: E = 4.653, RMS = 0.9647 R = 0.9767
Cas avec $N - M = 0$ $(N = 5, M = 5)$: E = 1.07e-13, RMS = 1.46e-07 R = 1.0000



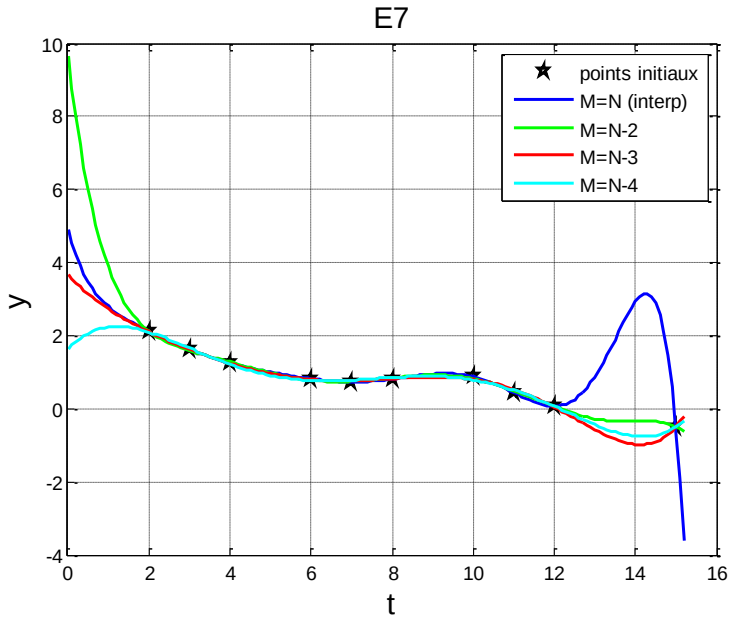
E7 En utilisant la méthode de la projection orthogonale et un polynôme de la forme :

$$g(x) = a_1 + a_2 x + a_3 x^2 + \ldots + a_M x^{M-1} = \sum_{i=1}^{M} a_i x^{i-1}$$

calculer la fonction d'approximation de la trajectoire de la grue du problème D4. Ce calcul sera fait par une fonction MATLAB générique prenant comme entrées les abscisses et les ordonnées des points de la trajectoire ainsi le nombre de termes M dans le polynôme d'approximation. La sortie de cette fonction sera un vecteur contenant les coefficients des polynômes. Utiliser cette fonction pour calculer les polynômes dans les cas où :

(a) $N = M$ (interpolation)
(b) $N = M + 2$
(c) $N = M + 3$
(d) $N = M + 4$.

Simuler et afficher ensuite ces polynômes sur la même figure accompagnée des points initiaux. Le résultat est illustré ci-dessous, à comparer avec le graphique de D4.



- Dans le cas de l'interpolation ($M = N$), l'oscillation observée après $t = 12$ est appelée le '*Polynomial Wiggle Problem*' dans les livres de référence. C'est le nom donné aux situations où la condition $M < N - 3$ n'est pas satisfaite (le pire cas étant l'interpolation avec $N = M$) et où il y a danger de déviation importante entre la tendance générale des données et l'approximation polynomiale.

- Par contre, si on veut forcer une trajectoire à passer à travers certains points (appelés '*way points*'), il faut choisir judicieusement d'autres points d'interpolation pour éviter le '*Polynomial Wiggle Problem*'. Une façon de choisir ces points de façon judicieuse sera présentée à la section 1.2.3.

- On peut remarquer qu'il est dangereux d'utiliser le polynôme d'approximation à l'extérieur du domaine des données originales. On voit que l'approximation entre $t = 0$ et $t = 2$ et au-delà de t = 15 est très différente selon le nombre M de coefficients utilisés dans le polynôme.

1.2 Approximation continue

Définition

- Une fonction *g(x)* fait l'approximation continue d'une autre fonction *f(x)* si son tracé se rapproche de cette fonction *f(x)* à une précision spécifiée sur un intervalle $I = [a, b]$.
- L'idée générale d'une telle approximation est que *g(x)* est habituellement plus facile et rapide à évaluer que la fonction originale *f(x)*, permettant une meilleure rapidité (e.g. dans des calculs en temps réel) ou permettant une analyse locale de certaines propriétés de *f(x)* (e.g. minimum, maximum, racines, etc).
- Par exemple, les approximations de Tchebychev et les polynômes rationnels sont utilisés dans les calculatrices pour calculer les fonctions tan^{-1} et logarithme naturel dont les séries de Taylor ne convergent pas aussi rapidement.

1.2.1 Les équations normales

- Les équations normales sont obtenues par analogie avec le cas discret.
- Comme avant, la fonction *g(x)* est construite à partir d'une combinaison linéaire de fonctions candidates où les coefficients a_i doivent être déterminés afin de minimiser l'erreur quadratique entre *g(x)* et *f(x)* sur *I* :

$$g(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + \ldots + a_M\varphi_M(x) = \sum_{i=1}^{M} a_i\varphi_i(x) \qquad \textbf{(19)}$$

- Déviation de *g(x)* à *x* : $\delta(x) = g(x) - f(x)$

- On vise à minimiser l'intégrale de la déviation au carré sur l'intervalle *I* (à comparer à la somme des déviations au carré dans le cas de l'approximation discrète – voir l'équation (1)) :

$$E = \int_a^b \delta(x)^2 dx = \int_a^b [g(x) - f(x)]^2 dx \qquad \textbf{(20)}$$

- Les équations normales sont obtenues avec : $\frac{\partial E}{\partial a_i} = 0, i = 1, \ldots, M.$

- On obtient la même forme que celle de l'équation (5), $\Phi A = \Psi$:

$$\begin{bmatrix} \langle\varphi_1,\varphi_1\rangle & \langle\varphi_1,\varphi_2\rangle & \cdots & \langle\varphi_1,\varphi_M\rangle \\ \langle\varphi_2,\varphi_1\rangle & \langle\varphi_2,\varphi_2\rangle & \cdots & \langle\varphi_2,\varphi_M\rangle \\ \vdots & \vdots & \ddots & \vdots \\ \langle\varphi_M,\varphi_1\rangle & \langle\varphi_M,\varphi_2\rangle & \cdots & \langle\varphi_M,\varphi_M\rangle \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} \langle\varphi_1, y\rangle \\ \langle\varphi_2, y\rangle \\ \vdots \\ \langle\varphi_M, y\rangle \end{bmatrix} \qquad \textbf{(21)}$$

- La seule modification est le sens du produit scalaire de fonctions,$\langle\cdot,\cdot\rangle$, dont les sommations (de l'approximation discrète) sont remplacées par des intégrales (pour l'approximation continue):

$$\langle\varphi_i,\varphi_j\rangle = \int_a^b \varphi_i(x)\varphi_j(x)\,dx \langle\varphi_i, y\rangle = \int_a^b \varphi_i(x) f(x)\,dx \qquad \textbf{(22)}$$

- La solution est donc de la même forme que pour l'équation (5) : $\mathrm{A} = \Phi^{-1}\Psi$

E8 Calculer une approximation de la fonction $f(x) = \sqrt{x}$sur l'intervalle [1, 5] avec une approximation parabolique $f(x) \approx a_1 + a_2 x + a_3 x^2$. Afficher l'erreur et évaluer approximativement son maximum et son minimum. Rappel : $\int_a^b x^n dx = \left[\frac{x^{n+1}}{n+1}\right]_a^b$.

- Il est important de noter que le produit scalaire de deux vecteurs $\vec{u} \cdot \vec{v}$ ou, sous forme de matrices-colonnes de composantes $u^T v$, donne la longueur de la projection d'un vecteur sur l'autre. De la même façon, le produit scalaire de deux fonctions donne la '*projection*' d'une fonction sur l'autre. Ainsi, $\int_a^b \varphi_n(x) f(x)\, dx$ donne la 'projection' de $f(x)$ sur $\varphi_n(x)$ ou la 'présence' de $f(x)$ dans $\varphi_n(x)$. Par exemple, si $\varphi_n(x) = \sin(nx)$, le produit scalaire $\int_a^b \sin(nx) f(x)\, dx$donne l'amplitude du contenu de la $n^{ième}$ harmonique sinusoïdale dans de *f(x)*. C'est le principe utilisé dans la série de Fourier.

1.2.2 Approximation polynomiale de Taylor

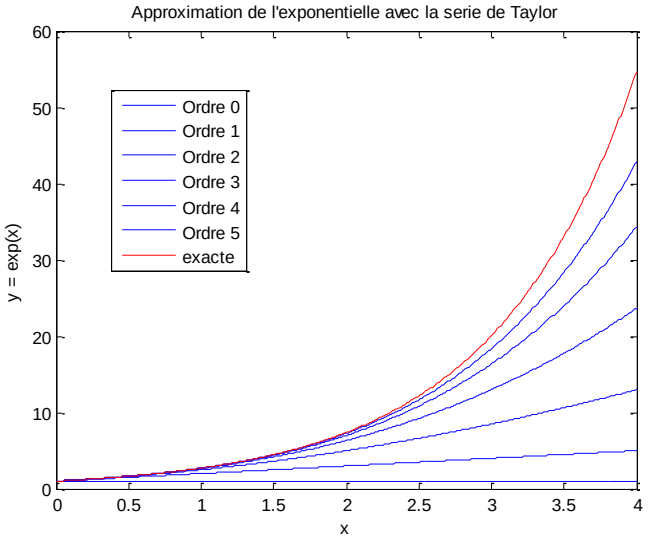
- Quand les dérivées de la fonction *f(x)* sont disponibles, la **série de Taylor** peut être utilisée pour faire l'approximation de la fonction *f(x)*.
- Une série de Taylor autour du point x_e représente la fonction *f(x)* dans une région autour de :

$$f(x) = f(x_e) + \frac{f'(x_e)}{1!}(x - x_e) + \frac{f''(x_e)}{2!}(x - x_e)^2 + \ldots = \sum_{i=1}^{\infty} \frac{f^{(i)}(x_e)}{i!}(x - x_e)^i \qquad \textbf{(23)}$$

- La notation x_e est utilisée pour éviter de confondre le point de linéarisation x_e et la condition (ou valeur) initiale de la fonction, x_0. Dans plusieurs problèmes, le point de linéarisation x_e est souvent un point d'équilibre dynamique, d'où l'origine de la notation.

- Quand $x_e = 0$, la série de Taylor prend le nom de **série de Maclaurin**.

- Le graphique suivant démontre différentes approximations de la fonction exponentielle autour de $x_e = 0$ avec la série de Taylor avec des ordres variant de 0 à 5.



- L'approximation d'ordre *M* de *f(x)* est donnée par :

$$f(x) \approx g_M(x) = f(x_e) + \frac{f'(x_e)}{1!}(x - x_e) + \ldots + \frac{f^{(M)}(x_e)}{M!}(x - x_e)^M = \sum_{i=1}^{M} \frac{f^{(i)}(x_e)}{i!}(x - x_e)^i \qquad \textbf{(24)}$$

- L'erreur de l'approximation de la série de Taylor d'ordre M est appelée le résidu d'ordre *M*, $r_M(x)$:

$$f(x) = g_M(x) + r_M(x) \qquad i.e. \qquad r_M(x) = f(x) - \sum_{i=1}^{M} \frac{f^{(i)}(x_e)}{i!}(x - x_e)^i \qquad \textbf{(25)}$$

- La forme de Lagrange du résidu d'ordre *M* est donnée par :

$$r_M(x) = \frac{f^{(M+1)}(\xi)}{(M+1)!}(x - x_e)^{M+1} \qquad \textbf{(26)}$$

où ξ est entre x_e et x.

- L'approximation par série de Taylor permet donc d'obtenir une limite supérieure sur l'erreur maximale entre la fonction originale $f(x)$ et son approximation $g_M(x)$.

E9 Calculer une approximation de la fonction $f(x) = \sqrt{x}$ sur l'intervalle [1, 5] avec une approximation de Taylor autour de $x = 1$ et de $x = 3$. Essayer deux ordres différents d'approximation. Comparer avec la fonction originale. Laquelle donne la meilleure approximation et pourquoi? Comparer aussi avec E8 en termes d'erreurs maximum et minimum d'interpolation.

1.2.3 Approximation minimax polynomiale aux nœuds de Tchebychev

- Une autre façon de faire l'approximation d'une fonction continue consiste à échantillonner la fonction en un certain nombre de points discrets et d'utiliser les techniques d'approximation discrète décrites à la section précédente.
- Vu que les données ainsi échantillonnées sont normalement exactes, sans erreurs de mesures, l'approximation de la fonction se fait normalement par interpolation des points échantillonnés.
- On fait donc l'approximation de l'infinité de points de la fonction continue par l'interpolation d'un sous-ensemble fini des ces points.
- Cependant, l'exercice E7 a clairement démontré le danger de l'interpolation (*Polynomial Wiggle Problem*): une interpolation polynômiale de données peut causer des déviations importantes dans les segments entre les points d'interpolation.
- Quand les données sont discrètes, comme ce fut le cas dans E7, une des façons d'éviter ce problème est d'utiliser des splines cubiques (voir section 2.3).
- Quand les données peuvent être obtenues de façon continue pour toute valeur de la variable indépendante, comme c'est le cas ici, il est possible de choisir les points d'interpolations de façon à *équilibrer* les déviations dans le sens *minimax*, c'est-à-dire de façon a *mini*miser la déviation *max*imale.
- Cet équilibrage assurera que les déviations de la fonction d'approximation *g(x)* seront d'amplitude égale d'un côté comme de l'autre de la fonction à approximer.

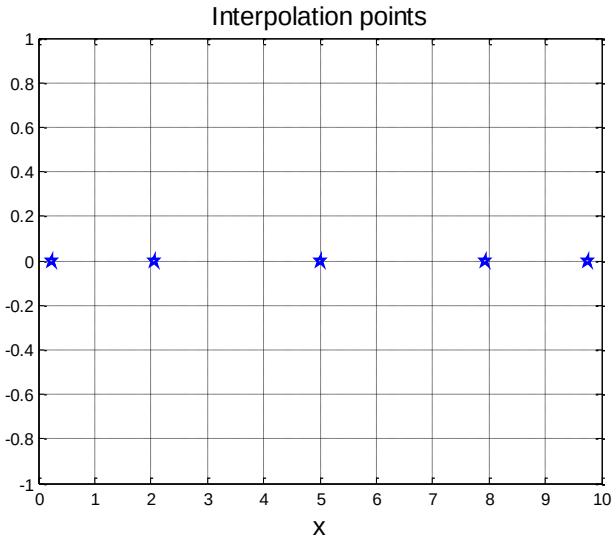
- Pour obtenir une approximation *minimax* d'une fonction continue, il peut être démontré qu'il suffit de choisir *N+1* points d'interpolation aux racines du polynôme de Tchebychev d'ordre *N+1*.
- Ces racines sont données sur l'intervalle normalisé [-1, 1] par :

$$\xi_n = cos\left[\left(\frac{2n+1}{2N+2}\right)\pi\right], n = 0, \ldots, N \qquad \textbf{(27)}$$

- Vu que ces coordonnées sont définies sur un intervalle normalisé entre –1 et +1, il faut appliquer un changement de variable pour une variable indépendante *x* définie sur un intervalle quelconque [*a*, *b*]:

$$x = a + \frac{b-a}{2}(\xi + 1) \qquad \textbf{(28)}$$

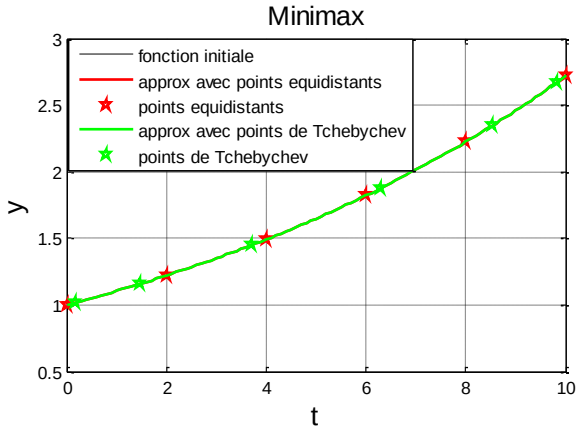
- Par exemple, avec cette équation, les N+1 = 5 points d'interpolation Tchebychev sur un intervalle [*a*, *b*] = [1, 10] serait répartis comme indiqué dans la figure ci-dessous.

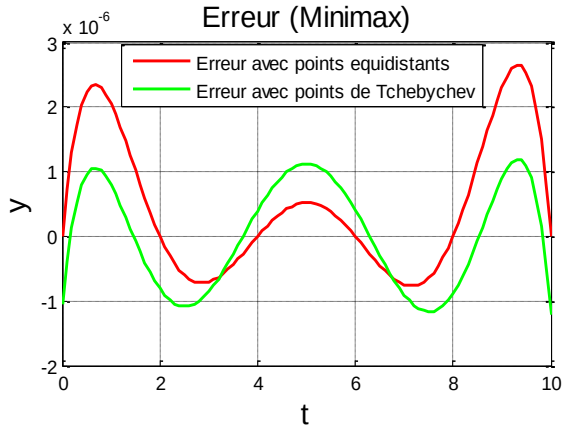


- Les points d'interpolation sont plus denses vers l'extrémité de l'intervalle.

- Une interpolation (e.g. méthode de projection orthogonale) en ces points assure la minimisation des déviations maximales entre l'approximation et la fonction originale.

E10 Faire l'approximation de la fonction $f(t) = e^{t/10}$ dans l'intervalle [0, 10] avec un polynôme d'interpolation d'ordre N=5 en utilisant (1) N+1 échantillons équidistants et (2) N+1 échantillons aux nœuds de Tchebychev. Faire le graphique de l'erreur d'approximation pour vérifier la nature minimax de l'approximation dans le cas (2) comparativement au cas (1), comme illustré ci-dessous. Donner les deux fonctions d'approximation.





APPROXIMATION DE DONNÉES
CHAPITRE 1

DONNÉES DISCRÈTES
(Section 1.1)

Trouver la fonction $y = f(x)$
qui passe près des N données x_i , y_i
au sens des moindres carrés

$$f(x) = \sum_j^M a_j \varphi_j(x)$$

(M paramètres à régler)

Méthode 1

Fonction coût $E = \sum(\,)^2$ (1)
Dérivées partielles : $\frac{\partial E}{\partial a_j} = 0$
ÉQUATIONS NORMALES

$$\begin{bmatrix} \langle \varphi_i \varphi_j \rangle & \cdots \\ \vdots & \ddots \end{bmatrix} \begin{bmatrix} a_j \\ \vdots \end{bmatrix} = \begin{bmatrix} \langle \varphi_i y_i \rangle \\ \vdots \end{bmatrix}$$

$$\langle \cdots \rangle = \sum \cdots$$

Forme : $\Phi A = \Psi$
Solution : $A = \Phi^{-1}\Psi$

Méthode 2

Paramètres a_j apparaissent linéairement
MATRICE DE PROJECTION

$$\begin{bmatrix} \varphi_i & \cdots \\ \vdots & \ddots \end{bmatrix} \begin{bmatrix} a_j \\ \vdots \end{bmatrix} = \begin{bmatrix} y_i \\ \vdots \end{bmatrix}$$

Forme : $PA = Y$
Solution : $A = \left(\mathrm{P}^{\mathrm{T}}\mathrm{P}\right)^{-1}\mathrm{P}^{T}Y$

Éqs. (4), (5), (6)

Éqs. (9), (10), (15)

FONCTIONS CONTINUES
(Section 1.2)

Trouver la fonction $y = f(x)$ qui représente le mieux la fonction plus complexe $g(x)$

$$f(x) = \sum_j^M a_j \varphi_j(x)$$

(M paramètres à régler)

Méthode 1

Fonction coût $E = \int(\,)^2 dx$ (21)
Dérivées partielles : $\frac{\partial E}{\partial a_j} = 0$
ÉQUATIONS NORMALES

$$\begin{bmatrix} \langle \varphi_i \varphi_j \rangle & \cdots \\ \vdots & \ddots \end{bmatrix} \begin{bmatrix} a_j \\ \vdots \end{bmatrix} = \begin{bmatrix} \langle \varphi_i y_i \rangle \\ \vdots \end{bmatrix}$$

$$\langle \cdots \rangle = \int \cdots$$

Forme : $\Phi A = \Psi$
Solution : $A = \Phi^{-1}\Psi$

Éqs. (20), (22), (23)

Méthode 2

Minimax : interpolation aux noeuds deTchebychev
INTERPOLATION

Éqs. (28), (29)

M = N

INTERPOLATION
CHAPITRE 2

Cas particuliers

2.1 $\varphi_i(x) = x^i$ => Vandermonde (34)
2.2 $\varphi_i(x) = L_i(x)$ => Lagrange (36)
2.3 $\varphi(x) = a_0 + a_1 x + a_2 x^2$ splines cubiques (39)-(41)

2 INTERPOLATION DE DONNÉES

Définition et utilité

- *Interpolation de données* : problème algébrique qui consiste à développer et résoudre les équations algébriques permettant d'obtenir une fonction dont le tracé passe exactement par toutes les données.
- Elle permet de trouver les valeurs intermédiaires entre les données tabulées.
- Elle permet une représentation fonctionnelle utile pour déduire certaines propriétés des données (e.g. aire sous la courbe, dérivée ou pente locale, etc.).

2.1 Le polynôme d'interpolation par inversion de matrice

- Il a été déjà démontré à la section précédente qu'un polynôme d'interpolation est un cas particulier d'une approximation fonctionnelle de données discrètes où le nombre de coefficients *M* du polynôme d'interpolation est égal au nombre de données *N* à interpoler.
- Dans ce cas, il y a une solution unique pour les *M* coefficients en fonction des *N=M* données.
- Cette approche peut s'appliquer indifféremment à des données obtenues à des intervalles équidistants ou non-équidistants de la variable indépendante.

- On suppose un polynôme d'interpolation *g(x)* avec *N* coefficients :

$$g(x) = a_1\varphi_1(x) + a_2\varphi_2(x) + \ldots + a_N\varphi_N(x) = \sum_{n=1}^{N} a_n\varphi_n(x) \qquad \textbf{(29)}$$

- En évaluant cette fonction aux *N* données à interpoler, *N* équations algébriques sont obtenues:

$$\begin{bmatrix} g(x_1) \\ g(x_2) \\ \vdots \\ g(x_N) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} a_n\varphi_n(x_1) \\ \sum_{i=1}^{N} a_n\varphi_n(x_2) \\ \vdots \\ \sum_{i=1}^{N} a_n\varphi_n(x_N) \end{bmatrix} = \begin{bmatrix} \varphi_1(x_1) & \varphi_2(x_1) & \cdots & \varphi_N(x_1) \\ \varphi_1(x_2) & \varphi_2(x_2) & \cdots & \varphi_N(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_1(x_N) & \varphi_2(x_N) & \cdots & \varphi_N(x_N) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \qquad \textbf{(30)}$$

- Cette équation est de la forme $PA = Y$ avec la solution:

$$A = P^{-1}Y \qquad \textbf{(31)}$$

- Si les données x_n sont distinctes et les fonctions $\varphi_n(x)$ sont indépendantes (i.e. elles ne sont pas reliées par une combinaison linéaire), l'inverse de *P* existe et donc, la solution aussi.

- La base des fonctions d'interpolation $\varphi_n(x)$) est au choix de l'utilisateur mais doit observer la condition que ces fonctions doivent générer une matrice *P* avec rangées/colonnes indépendantes pour que l'inverse existe.
- Le choix des $\varphi_n(x)$ doit aussi être adapté au problème. Par exemple, si les données ont une tendance périodique, les $\varphi_n(x)$ devraient être, par exemple, des fonctions sinusoïdales et la fonction d'approximation devient une série de Fourier.
- Des choix typiques pour $\varphi_n(x)$ sont les suivants (*n = 1, 2, 3, …*):

- $\varphi_n(x) = x^{n-1}$ polynôme en puissance de *x*
- $\varphi_n(x) = 1, \sin(nx), \cos(nx)$ série de Fourier
- $\varphi_n(x) = e^{\pm nx/c}$ exponentielle
- $\varphi_n(x) = 1, (x+c)^{1/n}$ polynôme en racine de *x*
- $\varphi_n(x) = 1, (x+c)^{-n}$ polynôme en réciproque de *x*

- Dans le cas particulier où les fonctions $\varphi_n(x)$ sont exprimées en puissance de *x* :

$$\varphi_n(x) = x^{n-1} \quad \textbf{(32)}$$

l'équation (30) prend la forme :

$$\begin{bmatrix} 1 & x_1 & \cdots & x_1^{N-1} \\ 1 & x_2 & \cdots & x_2^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \cdots & x_n^{N-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \quad \Rightarrow PA = Y \quad \textbf{(33)}$$

- Dans ce cas, la matrice *P* est appelée la matrice de Vandermonde dont l'existence de l'inverse est assurée si les x_n sont distincts. Cependant, quand les valeurs de la variable indépendante aux points d'interpolation x_n sont d'amplitude élevée et que l'ordre de l'interpolation *N* est aussi élevé, des problèmes numériques peuvent apparaître (pourquoi?).

E11 On veut trouver un polynôme qui interpole les 6 données suivantes.

t	0.0	1.0	2.0	3.0	4.0	5.0
y	0.0	2.608	1.350	-1.909	-2.338	0.6988

Calculer une fonction d'interpolation de ces données avec les fonctions de base suivantes:

(a) $g(x) = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + a_5 x^4 + a_6 x^5 = \sum_{n=1}^{6} a_n x^{n-1}$

(b) $g(x) = a_1 \cos(\omega x) + a_2 \sin(\omega x) + a_3 \cos(2\omega x) + a_4 \sin(2\omega x) + a_5 \cos(3\omega x) + a_6 \sin(3\omega x)$
avec $\omega = 2\pi/4.8$.

(c) $g(x) = a_1 e^{x/c} + a_2 e^{-x/c} + a_3 e^{2x/c} + a_4 e^{-2x/c} + a_5 e^{3x/c} + a_6 e^{-3x/c}$ avec $c = 10$.

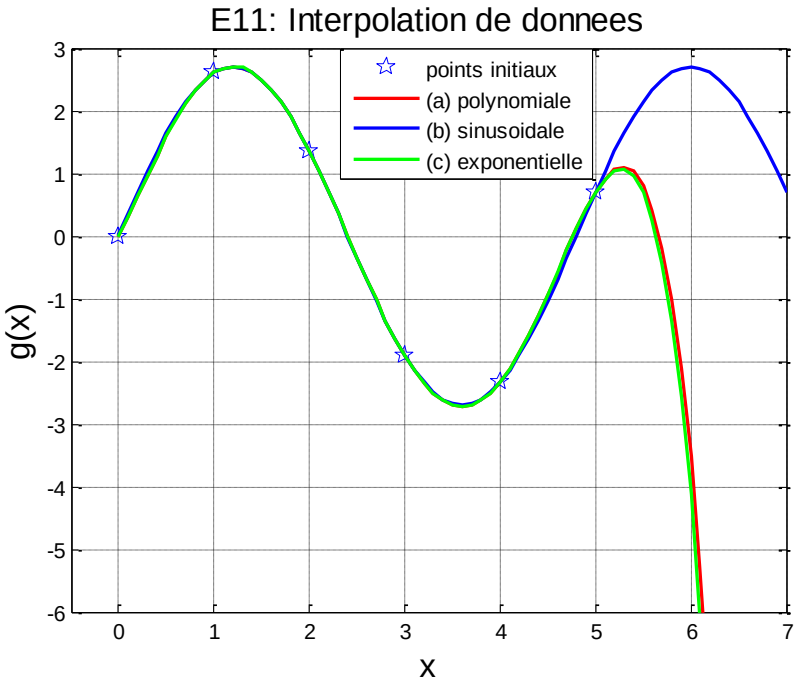
(d) $g(x) = a_1 + \sum_{n=2}^{6} a_n (x+c)^{1/(n-1)}$ avec $c = 10$.

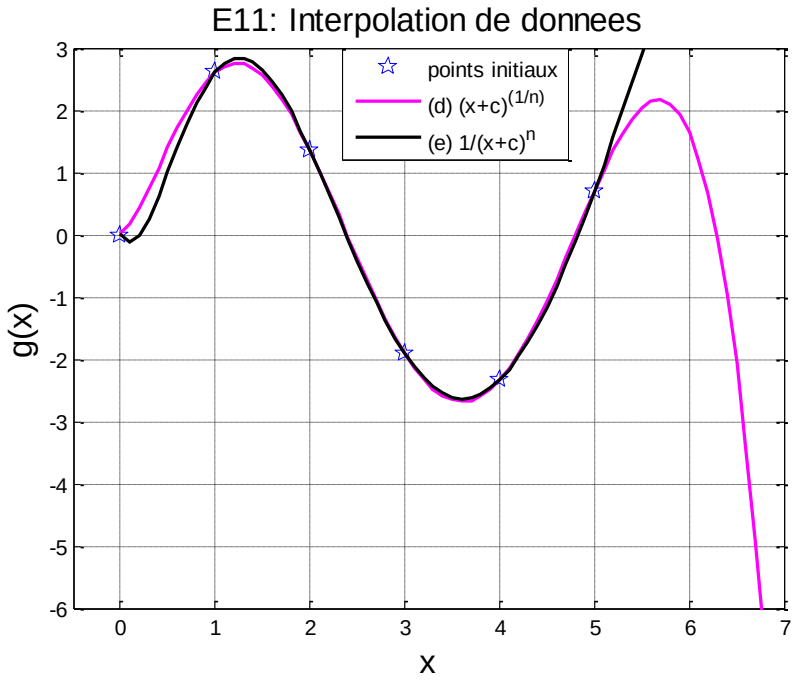
(e) $g(x) = a_1 + \sum_{n=2}^{6} a_n (x+c)^{-(n-1)}$ avec $c = 10$.

Dans l'ordre, ces approximations sont de type (a) polynôme en puissance de x, (b) série de Fourier, (c) exponentielle, (d) réciproque et (e) racine. Dans le cas (b) de la série de Fourier, la fréquence fondamentale du signal $\omega = 2\pi/4.8$ doit être obtenue d'une autre source ou par analyse de Fourier. Dans les cas (c), (d), (e), la constante c est ajustée pour obtenir la meilleure représentation des données. Dans tous les cas, on remarque que le nombre de coefficients ajustables doit être égal au nombre de données à interpoler.

Réponses : Les coefficients obtenus sont :

	a_1	a_2	a_3	a_4	a_5	a_6
(a)	0.000	2.989	1.478	-2.520	0.7204	-0.05968
(b)	0.000	2.700	0.002	0.001	-0.002	-0.001
(c)	-12609	20436	7736.8	-20322	-1460.9	6220.7
(d)	2.6073e+08	-8.7488e+05	1.9756e+08	-2.2826e+09	5.9318e+09	-4.1059e+09
(e)	-12744	8.4436e+05	2.2031e+07	2.8324e+08	1.7960e+09	4.4982e+09





- Les données ont été générées à partir de la fonction $y = 2.7 * \sin(2\pi x/4.8)$. On peut observer que le polynôme à base de fonctions trigonométriques, le cas (b), donne le résultat exact de la fonction, même au-delà de l'intervalle sur lequel les données étaient disponibles. On voit bien dans le tableau de coefficients que seul le coefficient $a_2 = 2.7$ du $\sin(2\pi x/4.8)$ est significatif, tous les autres sont presque zéro (les différences sont dues erreurs d'arrondissement dans les données du tableau (seulement 4 chiffres significatifs ont été retenus dans ces données de départ).

- En pratique, la période fondamentale des données (4.8 ici) n'est pas disponible et l'interpolation sinusoïdale pourrait être moins précise sans cette information. L'interpolation polynomiale (le cas (a)) donne des résultats satisfaisants sans cette information mais seulement dans la plage de données. L'interpolation polynômiale en puissance de *x* au-delà de l'intervalle d'intérêt est rapidement faussée.

- Il en est ainsi pour les interpolations de type exponentielle (c), réciproque (d) et racine (e) qui divergent au-delà de la plage d'intérêt.

- Cet exemple démontre la nécessité de bien choisir la base fonctionnelle $\phi_n(x)$ dans laquelle l'interpolation sera calculée. Il est toujours mieux d'utiliser une fonction qui semble *naturelle* aux données.

Il est possible de combiner différentes fonctions de base pour bien représenter les données disponibles.

L'exercice E11 est repris mais avec des données à interpoler qui ont une composante constante et une composant en rampe (proportionnelle à x) :

t	0.0	1.0	2.0	3.0	4.0	5.0
y	9.2	8.2979	5.6021	7.6021	14.298	19.2

On maintient le polynôme en puissance de *x* comme référence :

(a) $g(x) = a_1 + a_2x + a_3x^2 + a_4x^3 + a_5x^4 + a_6x^5 = \sum_{n=1}^{6} a_n x^{n-1}$

mais on modifie l'interpolation sinusoïdale en y ajoutant une fonction constante et une rampe :

(b) $g(x) = a_1 + a_2x + a_3\cos(\omega x) + a_4\sin(\omega x) + a_5\cos(2\omega x) + a_6\sin(2\omega x)$
avec $\omega = 2\pi/5$.

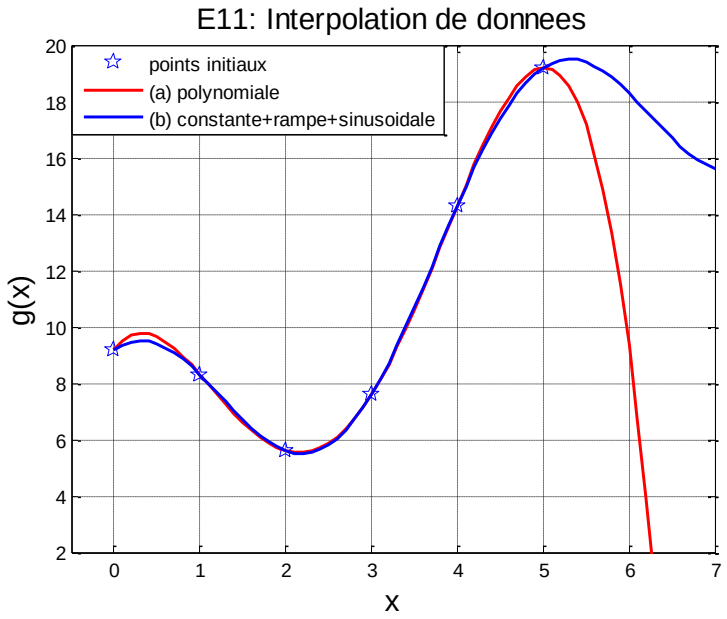
Les coefficients obtenus sont :

	a_1	a_2	a_3	a_4	a_5	a_6
(a)	9.2	3.7801	-7.1158	2.7039	-0.27039	0.000
(b)	5.0	2.0	4.2	0.000	0.000	0.000

On voit par la deuxième rangée et la fonction (b) ci-dessus que les données répondent à l'équation :

$$g(x) = 5 + 2x + 4.2\,cos(2\pi x/5)$$

Le graphique ci-dessous représente les deux fonctions d'interpolation.



2.2 Le polynôme d'interpolation de Lagrange

- La méthode de la section précédente permet de calculer numériquement le polynôme d'interpolation par l'inversion d'une matrice de dimension *N* x *N* où *N* est le nombre de données à interpoler.

- Il existe une façon plus analytique et moins complexe numériquement de créer le polynôme d'interpolation. Le polynôme résultant de cette technique est appelé **polynôme de Lagrange**.

- On veut trouver le polynôme d'interpolation des *N+1* données suivantes :

$$(x_i, y_i), i = 0, \dots N \tag{34}$$

- Le polynôme de Lagrange *p(x)* est exprimé sous la forme :

$$p(x) = y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) + \cdots + y_N L_N(x) \tag{35}$$

où les *fonctions de sélection* $L_i(x)$ sont égales à 1 quand $x = x_i$ et égales à zéro aux autres x_i.

- Ces fonctions de sélection $L_i(x)$ sont d'ordre *N* et définies par :

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{N} \frac{(x - x_j)}{(x_i - x_j)} = \begin{cases} 0, x = x_j, j \neq i \\ 1, x = x_i. \end{cases} \tag{36}$$

- Par exemple, supposant un cas où il y a *N+1=6* données à interpoler, le 2ème polynôme de Lagrange serait :

$$L_2(x) = \frac{(x - x_0)(x - x_1)(x - x_3)(x - x_4)(x - x_5)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)(x_2 - x_4)(x_2 - x_5)} \tag{37}$$

- Sous cette forme, il est plus facile de voir que $L_2(x_0) = L_2(x_1) = L_2(x_3) = L_2(x_4) = L_2(x_5) = 0$ et que $L_2(x_2) = 1$. Il est ainsi plus évident de comprendre que le polynôme de Lagrange à l'équation (35) interpole bien les y_i.

- Finalement, il est aussi clair que pour *N+1* données à interpoler, le polynôme d'interpolation sera d'ordre *N* puisqu'un tel polynôme contient *N+1* degrés de liberté pour l'interpolation des *N+1* points.

E12 Écrire un code MATLAB permettant d'afficher la courbe du polynôme de Lagrange obtenue à partir des données de l'exercice E11 (les coefficients du polynôme ne sont pas demandés).

2.3 Splines cubiques

- Le problème le plus commun avec l'interpolation d'un grand nombre *N+1* de données avec un polynôme d'ordre élevé *N* est le *Polynomial Wiggle Problem* discuté à l'Exercice E7.
- Une façon d'éviter ce problème est de faire une interpolation par segments avec un polynôme d'ordre inférieur à N.
- Les splines cubiques utilisent des polynômes $g(x)$ d'ordre 3 avec 4 coefficients indéterminés sur chaque intervalle $x_n \leq x \leq x_{n+1}$:

$$g(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \qquad \textbf{(38)}$$

- Deux conditions sont imposées par le besoin d'interpoler à x_n et x_{n+1}. Avec quatre coefficients à déterminer, il reste donc deux degrés de liberté à utiliser en imposant deux contraintes additionnelles.
- Ces contraintes sont d'assurer la continuité de la première dérivative $g'(x)$ et de la deuxième dérivative $g''(x)$ à chaque interconnexion.
- La cubique d'interpolation est construite en suivant les étapes suivantes :
  - La deuxième dérivée de la cubique est une droite à deux coefficients inconnus : $g''(x) = 2a_2 + 6a_3 x$.
  - Sur les $N+1$ données à interpoler, il y a $N+1$ valeurs inconnues de$g''(x)$. Il y a $N-1$ interconnexions et donc $N-1$ conditions de continuité sur les $g''(x)$. En supposant une interpolation de splines cubiques *naturelle*, on pose $g''(x_0) = g''(x_N) = 0$ pour ainsi avoir seulement $N-1$ inconnues $g''(x)$ avec $N-1$ conditions de continuité.
  - En intégrant $g''(x)$ une première fois pour obtenir les $g'(x)$ et une deuxième fois pour les $g(x)$, on applique les $N-1$ conditions de continuité de $g'(x)$ aux interconnexions et les $N+1$ conditions d'interpolations $g(x)$ aux $N+1$ données.
  - On obtient ainsi les équations linéaires permettant de solutionner pour tous les coefficients des splines cubiques.

D6 Développer les équations linéaires des splines cubiques et démontrer que l'équation d'interpolation est :

$$g(x) = \frac{g_k''}{6}\left[\frac{(x_{k+1}-x)^3}{h} - h(x_{k+1}-x)\right] + \frac{g_{k+1}''}{6}\left[\frac{(x-x_k)^3}{h} - h(x-x_k)\right] + y_k\frac{(x_{k+1}-x)}{h} + y_{k+1}\frac{(x-x_k)}{h} \qquad \textbf{(39)}$$

$$\text{pour } x_k \leq x \leq x_k + 1, \qquad k = 0. \ldots, N-1$$

où $h = x_{k+1} - x_k$ et les $N-1$ inconnues $g_k'', k = 1, \ldots, N-1$ ($g''(x_0) = g''(x_N) = 0$) sont données par la solution de:

$$\begin{bmatrix} 4 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} g_1'' \\ g_2'' \\ \vdots \\ g_{N-2}'' \\ g_{N-1}'' \end{bmatrix} = \begin{bmatrix} \frac{6(y_2 - 2y_1 + y_0)}{h^2} \\ \frac{6(y_3 - 2y_2 + y_1)}{h^2} \\ \vdots \\ \frac{6(y_{N-1} - 2y_{N-2} + y_{N-3})}{h^2} \\ \frac{6(y_N - 2y_{N-1} + y_{N-2})}{h^2} \end{bmatrix} \qquad \textbf{(40)}$$

La matrice à inverser est de dimension $N-1 \times N-1$.

E13 Faire l'interpolation des données avec des splines cubiques et l'utiliser pour trouver la valeur de $g(3.4)$.

x	1	2	3	4	5
$g(x)$	11	9	12	15	11

3 DIFFERENTIATION NUMÉRIQUE

Définition et utilité

- *Différentiation numérique* : problème algébrique qui consiste à calculer numériquement la dérivée d'une fonction pour laquelle une différentiation analytique est difficile ou impossible; en particulier, problème qui consiste ou à calculer la dérivée d'une fonction dont la valeur n'est connue qu'à des points discrets.
- Elle permet de calculer des propriétés liées à la différentiation d'une fonction. Par exemple, la dérivée temporelle d'une position donne la vitesse et la dérivée temporelle de la vitesse donne l'accélération.
- La série de Taylor, présentée à la section 1.2.2, sera utilisée.

3.1 Différence avant et différence arrière

- On considère une fonction analytique $f(x)$ i.e. qui peut être représentée par une série de Taylor autour de x :

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(x) + \cdots \quad \textbf{(41)}$$

- La 1ère dérivée est mise en évidence :

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(x) - \frac{h^2}{6}f'''(x) + \cdots \quad \textbf{(42)}$$

- Avec la notation $f(x+h) = f_{j+1}$ et $f(x) = f_j$, la *1ère différence avant* de *f(x)* à x_j est définie par :

$$\Delta f_j = f(x+h) - f(x) = f_{j+1} - f_j \quad \textbf{(43)}$$

- L'approximation d'ordre *h* de la dérivée $f'(x)$, utilisant la 1ère différence *avant*, est définie par :

$$f'(x) = \frac{\Delta f_j}{h} + \text{termes d'ordre } \leq h \quad \textbf{(44)}$$

- En prenant la série de Taylor de $f(x-h)$, la même procédure donne l'approximation d'ordre *h* de la dérivée $f'(x)$, utilisant la 1ère différence *arrière*:

$$f'(x) = \frac{\nabla f_j}{h} + \text{termes d'ordre } \leq h \quad \textbf{(45)}$$

où la *1ère différence arrière* a été définie par :

$$\nabla f_j = f(x) - f(x-h) = f_j - f_{j-1} \quad \textbf{(46)}$$

- De façon similaire, en calculant la série de Taylor de chaque terme dans la différence $2f(x+h) - f(x+2h)$, le terme f' s'annule. En mettant f'' en évidence, on obtient la 2ème dérivée avec une 2ème différence avant:

$$f''(x) = \frac{f_{j+2} - 2f_{j+1} + f_j}{h^2} + (\text{termes} \leq h) = \frac{\Delta^2 f_j}{h^2} + (\text{termes} \leq h) \quad \textbf{(47)}$$

- En répétant le processus avec $f(x-h)$ et $f(x-2h)$, on obtient aussi f'' avec une différence arrière :

$$f''(x) = \frac{f_j - 2f_{j-1} + f_{j-2}}{h^2} + (\text{termes} \leq h) = \frac{\nabla^2 f_j}{h^2} + (\text{termes} \leq h) \quad \textbf{(48)}$$

3.2 Utilisation des dérivées dans le calcul d'erreur d'intégration numérique

- À la prochaine section sur l'intégration numérique, le calcul de l'erreur d'intégration sera fait à partir des dérivées d'une fonction évaluées aux extrémités de l'intervalle d'intégration.
- Pour calculer approximativement les dérivées d'une fonction $f(x_i)$ évaluées à une série de points discrets x_i aux extrémités $x = [a, b]$ d'un intervalle, on utilise la différence avant au point avant *a* et la différence arrière au point arrière *b*. On suppose une distance entre les points x_i égale à h.

- Première dérivée au point *a* (différence avant) :

$$f'(a) \approx \frac{f_{j+1} - f_j}{h}$$

fpa = (y(2)–y(1))/ delx;

- Première dérivée au point *b* (différence arrière) :

$$f'(b) \approx \frac{f_j - f_{j-1}}{h}$$

fpb = (y(end)–y(end-1))/delx;

- Deuxième dérivée au point *a* (différence avant) :

$$f''(a) \approx \frac{f_{j+2} - 2f_{j+1} + f_j}{h^2}$$

fppa = (y(3) –2*y(2)+y(1))/ delx^2;

- Deuxième dérivée au point *b* (différence arrière) :

$$f''(b) \approx \frac{f_j - 2f_{j-1} + f_{j-2}}{h^2}$$

fppb = (y(end) –2*y(end-1)+y(end-2))/ delx^2;

- Pour la 3e dérivée, on utilise la double dérivée appliquée à la simple dérivée.

- Troisième dérivée au point *a* (différence avant) :

$$f'''(a) \approx \frac{f_{j+3} - 3f_{j+2} + 3f_{j+1} - f_j}{h^3}$$

fpppa = (y(4) − 3*y(3) + 3*y(2) − y(1))/delx^3;

- Troisième dérivée au point *b* (différence arrière):

$$f'''(b) \approx \frac{f_j - 3f_{j-1} + 3f_{j-2} - f_{j-3}}{h^3}$$

fpppb = (y(end) – 3*y(end-1) + 3*y(end-2) – y(end-3))/delx^3;

3.3 Différentiation par la formule de Tustin

- Avec la différence *avant*, il a été démontré que l'approximation de la dérivée d'une fonction à un point *j* est donnée par :

$$f_j^{'}(x) = \frac{f_{j+1} - f_j}{h} \quad \textbf{(49)}$$

- Avec la différence *arrière*, l'approximation de la dérivée d'une fonction à un point *j* est donnée par :

$$f_j^{'}(x) = \frac{f_j - f_{j-1}}{h} \quad \textbf{(50)}$$

- En évaluant cette différence arrière au prochain point *j+1* (*j* $\Rightarrow$ *j+1*), on obtient :

$$f_{j+1}'(x) = \frac{f_{j+1} - f_j}{h} \quad \textbf{(51)}$$

- En comparant les équations (49) et (51), il est évident que leur côté droit étant identique, la différence arrière assigne le calcul de la dérivée au point *j+1* alors que la différence avant assigne cette même valeur de dérivée au point *j*.
- Il serait donc naturel de considérer que le côté droit de ces équations correspond non pas à la dérivée au point *j+1* ou au point *j* mais à la moyenne de ces dérivées :

$$\frac{f_{j+1}'(x) + f_j'(x)}{2}.$$

E14 En utilisant la moyenne des dérivées comme illustrée ci-dessus, démontrer que l'approximation correspondante de la dérivée sous la forme d'une équation discrète est donnée par :

$$f_j^{'}(x) = \frac{2}{h}\frac{(z-1)}{(z+1)} f_j \quad \textbf{(52)}$$

- Cette forme de la différentiation est utilisée dans la discrétisation de fonction de transfert avec la méthode de Tustin.

3.4 Différences centrales

- On considère une fonction analytique *f(x)* i.e. qui peut être représentée par une série de Taylor autour de *x*. On calcule les deux séries de Taylor suivantes :

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2}f^{''}(x) + \frac{h^3}{6}f''(x) + \cdots \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2}f^{''}(x) - \frac{h^3}{6}f''(x) + \cdots \end{aligned} \quad \textbf{(53)}$$

- En soustrayant la deuxième de la première expression, les termes de puissance paire de *h* s'annulent :

$$f_{j+1} - f_{j-1} = 2hf'(x) + \frac{h^3}{3}f''(x) + \cdots \quad \textbf{(54)}$$

- On met la première dérivée en évidence :

$$f^{'}(x) = \frac{f_{j+1} - f_{j-1}}{2h} - \frac{h^2}{6}f''(x) + \cdots \quad \textbf{(55)}$$

- La première dérivée ainsi obtenue avec une *différence centrale* est d'ordre h^2:

$$f'(x) = \frac{f_{j+1} - f_{j-1}}{2h} + (\text{termes} \leq h^2) \quad \textbf{(56)}$$

- Avec une même approche, la deuxième dérivée avec une *différence centrale* :

$$f''(x) = \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2} - \frac{h^2}{12} f'''' + (\text{termes} \leq h^4) \quad \textbf{(57)}$$

est d'ordre h^2 :

$$f''(x) = \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2} + (\text{termes} \leq h^2) \quad \textbf{(58)}$$

E15 A l'aide des approximations de dérivées de type différences centrales, obtenir la version discrète (en fonction de z) de la fonction de transfert suivante :

$$H(s) = \frac{2s + 5}{s^2 + 9}.$$

1. A partir de la fonction de transfert $H(s)$, établir l'équation différentielle liant l'entrée *u(t)* et la sortie *y(t)* et ses dérivées.
2. Utiliser les approximations de dérivées de type *différences centrales* pour obtenir une équation aux différences (voir note ci-dessous).
3. Introduire l'opérateur z défini par l'équation suivante :

$$y_{k-n} = y_k z^{-n} \quad (\text{retard de } n \text{ échantillons})$$

4. En déduire la fonction de transfert discrète :

$$H(z) = \frac{y_k}{u_k}$$

Note : La forme générale d'une équation aux différences est la suivante :

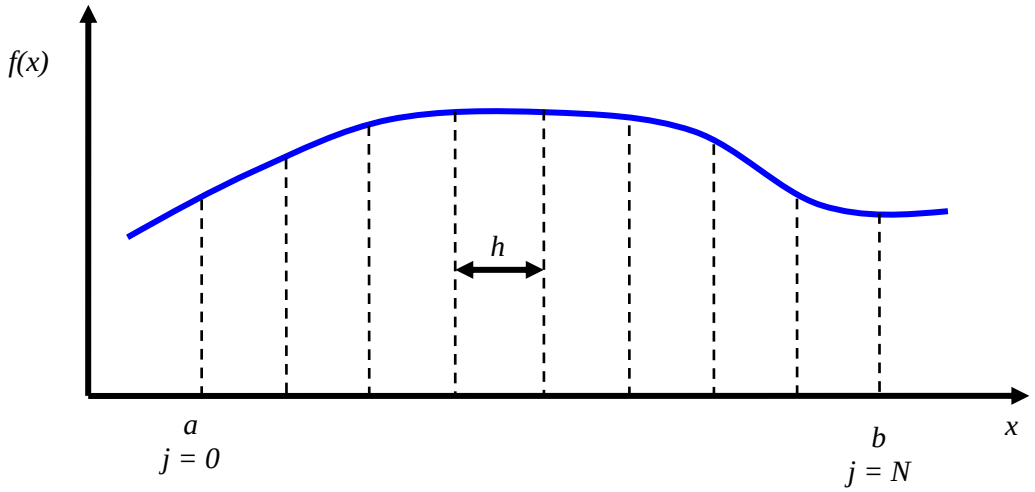
$$a_0 y_k + a_1 y_{k-1} + \cdots + a_n y_{k-n} = b_0 u_k + b_1 u_{k-1} + \cdots + b_m u_{k-m}$$

4 INTÉGRATION NUMÉRIQUE

Définition et utilité

- *Intégration numérique* : problème algébrique qui consiste à calculer numériquement l'intégrale d'une fonction pour laquelle une intégration analytique est difficile ou impossible, en particulier, problème qui consiste ou à calculer l'intégrale d'une fonction dont la valeur n'est connue qu'à des points discrets.
- Elle permet de calculer des propriétés liées à l'intégration d'une fonction. Par exemple, l'intégration d'une accélération en fonction du temps donne la variation de la vitesse et l'intégration de la vitesse donne la variation de position.
- On considère l'intégration d'une fonction *f(x)* sur un intervalle $a \leq x \leq b$:

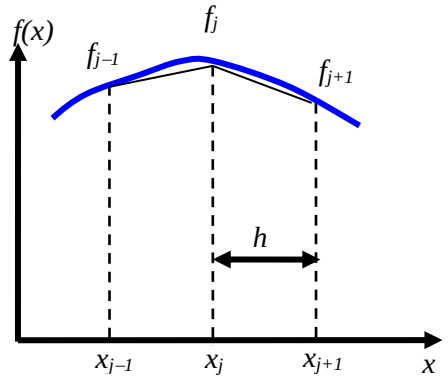
$$F = \int_a^b f(x)dx \qquad \textbf{(59)}$$



- Généralement, les méthodes numériques d'intégration consistent à subdiviser l'intervalle [*a*, *b*] en segments de largeur *h* qui subdivisent la surface à calculer en une série d'éléments de surface, ou *partitions*, délimités par les points aux coordonnées$(x_j, f_j), j = 0, \ldots, N$où$f_j = f(x_j)$.
- La surface de chaque partition est calculée et la somme de ces éléments de surface donne l'intégrale.
- Différentes méthodes existent pour calculer la surface des partitions, chacune avec sa propre précision et sa charge de calcul, ces deux caractéristiques étant habituellement proportionnelles.
- Les principales techniques sont présentées dans les prochaines sections.

4.1 Méthode trapézoïdale

4.1.1 Développement de la formule



- Dans la méthode trapézoïdale, les partitions sont créées en joignant les points $f(x_j) = f_j$avec des lignes droites.
- La surface de deux partitions consécutives devient :

$$\int_{x_{j-1}}^{x_j} f(x)dx \approx \frac{f_{j-1} + f_j}{2} h \int_{x_j}^{x_{j+1}} f(x)dx \approx \frac{f_j + f_{j+1}}{2} h$$

- En faisant la somme :

$$\int_{x_{j-1}}^{x_{j+1}} f(x)dx \approx \frac{h}{2}\left(f_{j-1} + 2f_j + f_{j+1}\right)$$

et en généralisant à tout l'intervalle :

$$F = \int_a^b f(x)dx \approx \frac{h}{2}(f_0 + 2f_1 + \cdots + 2f_{N-1} + f_N) = \frac{h}{2}\left(f_0 + f_N + 2\sum_{j=1}^{N-1} f_j\right) \quad \textbf{(60)}$$

- Le nom de la méthode vient du fait que chaque partition a la forme d'un trapèze.

E16 À partir de $\int_{x_j}^{x_{j+1}} f(x)dx \approx \frac{f_j + f_{j+1}}{2} h$ sur un intervalle, démontrer l'équation (60).

4.1.2 Calcul de l'erreur d'approximation

- Il est important de calculer l'erreur d'intégration de cette méthode.
- La série de Taylor de la section 1.2.2 est utilisée.

- Il peut être démontré que la méthode d'intégration trapézoïdale est une méthode d'ordre 2, c'est-à-dire que l'erreur de la méthode est proportionnelle à h^2 et que la forme de l'erreur est donnée par le terme encadré:

$$F = \int_a^b f(x)dx = \frac{h}{2}\left(f_0 + f_N + 2\sum_{j=1}^{N-1} f_j\right) - \frac{h^2}{12}(b-a)f''(\bar{x}) + \text{termes d'ordre plus élevé} \quad \textbf{(61)}$$

où $f''(\bar{x}) = \frac{1}{N}\sum_{j=0}^{N-1} f''(x_j)$ est la moyenne de la 2ème dérivée de la fonction sur tout l'intervalle. En faisant l'approximation de la 2ème dérivée du terme d'erreur par une simple différence $f''(\bar{x}) \approx \frac{f'(b) - f'(a)}{b-a}$, le terme d'erreur devient :

$$\text{Erreur} = \frac{h^2}{12}[f'(b) - f'(a)]$$

où $f'(a)$ et $f'(b)$ sont les dérivées de la fonction aux extrémités de l'intervalle (voir Section 3.2).

* * * * *

- Une correction peut être appliquée à (61) pour obtenir une méthode trapézoïdale avec correction:

$$F = \int_a^b f(x)dx = \frac{h}{2}\left(f_0 + f_N + 2\sum_{j=1}^{N-1} f_j\right) - \frac{h^2}{12}[f'(b) - f'(a)] + \text{termes d'ordre} \leq h^4 \quad \textbf{(62)}$$

- Sa précision est d'ordre 4 mais requiert le calcul de la 1ère dérivée de la fonction.

- La méthode trapézoïdale peut aussi être utilisée pour l'intégration d'une fonction évaluée seulement en des points discrets espacés également.

D7 : Calculer $\int_0^{0.5} e^x dx$ et l'erreur d'intégration avec la méthode trapézoïdale avec N = 2 et N = 4.

- Pour *N=2*, on a $h = 0.25$ et pour *N=4*, on a $h = 0.125$
- Selon (60) pour $N = 2$: $F = \int_0^{0.5} e^x dx \approx \frac{0.25}{2}(e^0 + 2e^{0.25} + e^{0.50}) = 0.652097$
- L'erreur d'intégration prédite est $\frac{0.25^2(e^{0.5} - e^{0.0})}{12} = 0.00338$.
- Selon (60) pour $N = 4$: $F = \int_0^{0.5} e^x dx \approx \frac{0.125}{2}(e^0 + 2e^{0.125} + 2e^{0.250} + 2e^{0.375} + e^{0.50}) = 0.649566$

- L'erreur d'intégration prédite est $\frac{0.125^2(e^{0.5}-e^{0.0})}{12} = 0.00085$.

- La valeur exacte est $e^{0.5} - e^{0} = 0.648721$ donc des erreurs respectives de 0.003376 et de 0.000845.

E17 La force appliquée F sur un véhicule en fonction de sa position x a été mesurée à des intervalles de 0.5 m. (a) Avec la méthode trapézoïdale, calculer le travail effectué par cette force sur le véhicule. (b) Faire le même calcul si l'intervalle entre les mesures avait été de 1.0 m plutôt que de 0.5 m. Dans les deux cas, calculer approximativement l'erreur d'intégration en calculant approximativement les dérivées premières $f'(a)$ et $f'(b)$ aux extrémités de l'intervalle.

x (m)	-2.0	-1.5	-1.0	-0.5	0.0	0.5	1.0	1.5	2.0	2.5	3.0
F (N)	14	8.75	5.00	2.75	2.00	2.75	5.00	8.75	14.00	20.75	29.00

Note : En intégrant la fonction analytique qui a généré ces données, on obtient la réponse exacte de *45 Joules*. Pour l'intervalle de 0.5, l'erreur prédite est 0.5625 et l'erreur réelle est 0.6250. Pour l'intervalle de 1.0, l'erreur prédite est 2.00 et l'erreur réelle est 2.50. La différence est due à l'approximation de la dérivée aux extrémités.

4.2 Méthode de Simpson

4.2.1 Développement de la formule

- Dans la méthode de Simpson, les partitions sont créées en joignant les points $f(x_j) = f_j$ avec des paraboles plutôt que des lignes droites.
- La formulation est obtenue avec la série de Taylor. On utilise l'intégrale indéfinie :

$$F(x) = \int_a^x f(x)dx \qquad \textbf{(63)}$$

- En développant la série de Taylor de $F_{j+1} = F(x_j + h)$ et de $F_{j-1} = F(x_j - h)$ autour de $x_j$et en soustrayant la deuxième de la première, on obtient :

$$F_{j+1} - F_{j-1} = 2hf_j + \frac{h^3}{3}f_j'' + \frac{h^5}{60}f_j'''' + \cdots \qquad \textbf{(64)}$$

- La deuxième dérivée f_j''est remplacée par sa représentation par différence centrale, équation (57):

$$F_{j+1} - F_{j-1} = \frac{h^3}{3}\left[f_{j-1} + 4f_j + f_{j+1}\right] - \frac{h^5}{90}f_j'''' + \text{ (termes} \leq h^7) \qquad \textbf{(65)}$$

- Cette expression ci-dessus représente la surface de deux partitions contiguës entre x_{j-1} et x_{j+1} utilisant la règle de Simpson. En faisant la somme sur toutes les paires de partitions (ce qui implique que le nombre de partitions doit être pair et le nombre de points impairs), i.e. :

$$F(b) = \int_a^b f(x)dx = (F_2 - F_0) + (F_4 - F_2) + (F_6 - F_4) + \cdots (F_N - F_{N-2}) \qquad \textbf{(66)}$$

on obtient, après quelques manipulations, la formule d'intégration de Simpson :

$$F = \int_a^b f(x)dx \approx \frac{h}{3}\left(f_0 + f_N + 4 \sum_{\substack{j=1 \\ jimpair}}^{N-1} f_j + 2 \sum_{\substack{j=2 \\ jpair}}^{N-1} f_j \right) + \text{(termes } \leq h^4) \quad \textbf{(67)}$$

4.2.2 Calcul de l'erreur d'approximation

- Le terme d'erreur est :

$$F - \frac{h}{3}\left(f_0 + f_N + 4 \sum_{\substack{j=1 \\ jimpair}}^{N-1} f_j + 2 \sum_{\substack{j=2 \\ jpair}}^{N-1} f_j \right) = -\frac{h^4}{180}(b-a) f''''(\overline{x}) + \text{(termes } \leq h^6) \quad \textbf{(68)}$$

  ce qui démontre que la méthode est d'ordre 4.

- Si la 4ème dérivée de la fonction à intégrer est nulle, l'erreur d'approximation est nulle. En faisant l'approximation de la 4ème dérivée du terme d'erreur par une simple différence, le terme d'erreur devient :

$$\text{Erreur} = \frac{h^4}{180}[f'''(b) - f'''(a)]$$

  où $f'''(a)$ et $f'''(b)$ sont les triples dérivées de la fonction aux extrémités de l'intervalle (voir Section 3.2).

E18 Recalculer le travail fait par la force F du problème E17 mais cette fois avec la méthode de Simpson. Calculer approximativement l'erreur d'intégration. Comparer la précision des résultats des deux méthodes (trapèze et Simpson) par rapport à la réponse exacte de 45 J.

4.3 Méthode de Gauss

4.3.1 Développement de la formule, version normalisée

- La méthode d'intégration de Gauss, aussi connue sous le nom de quadrature de Gauss, est développée à partir des étapes suivantes :
  - changement de variable de *x* à ξ et normalisation de l'intégrale sur l'intervalle [–1, +1]
  - utilisation des nœuds de Tchebychev comme points d'interpolation (section 1.2.3)
  - interpolation de ces points par une fonction d'interpolation (e.g. polynômes de Lagrange, section 2.2)
  - intégration analytique de cette fonction d'interpolation.

- Une fois ces étapes accomplies, la formule de quadrature de Gauss à *N* points prend la forme:

$$\int_{-1}^{+1} f(\xi)d\xi \approx \alpha_1 f(\xi_1) + \alpha_2 f(\xi_2) + \cdots + \alpha_N f(\xi_N) \quad \textbf{(69)}$$

  où les poids α_n et les valeurs des ξ_n sont des constantes qui peuvent être calculées et tabulées en fonction de N. Le tableau suivant donne les valeurs pour $N = 2, \ldots, 6$.

N	ξ_n	α_n	N	ξ_n	α_n
2	$\pm 1/\sqrt{3}$	1			
3	$\pm\sqrt{0.6}$ 0	5/9 8/9	**5**	± 0.9061798459 ± 0.5384693101 0	0.2369268850 0.4786286705 0.5688888889
4	± 0.8611363116 ± 0.3399810436	0.3478548451 0.6521451549	**6**	± 0.9324695142 ± 0.6612093865 ± 0.2386191861	0.1713244924 0.3607615730 0.4679139346

4.3.2 Développement de la formule, version non normalisée

- À partir d'une intégrale de la forme générale ci-dessous, il faut faire le changement de variable approprié pour obtenir la forme normalisée de l'équation (69).

$$F = \int_a^b f(x)dx \qquad \textbf{(70)}$$

- Le changement de variable de *x* variant dans [*a, b*] pour ξ variant dans [–1, +1] est :

$$x = a + \frac{b-a}{2}(\xi + 1) \qquad dx = \frac{b-a}{2}d\xi \qquad \textbf{(71)}$$

- En remplaçant (71) dans (70) :

$$F = \int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{+1} f\left(a + \frac{b-a}{2}(\xi + 1)\right)d\xi \qquad \textbf{(72)}$$

et en y insérant la formule de quadrature de Gauss (69), on obtient:

$$F = \int_a^b f(x)dx = \frac{b-a}{2}[\alpha_1 f(x_1) + \alpha_2 f(x_2) + \cdots + \alpha_N f(x_N)] \qquad \textbf{(73)}$$

où les x_n sont obtenus des valeurs tabulées de ξ_n avec :

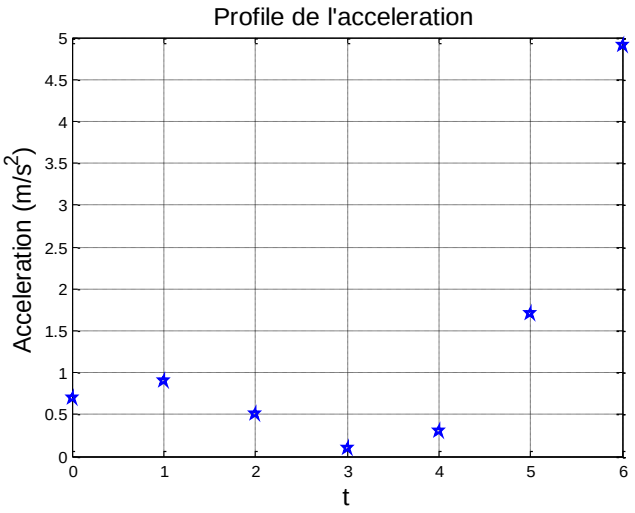
$$x_n = a + \frac{b-a}{2}(\xi_n + 1) \qquad \textbf{(74)}$$

E19 : Calculer $\int_0^{0.5} e^x dx$ avec la quadrature de Gauss avec N =2 et N =3.

Solution :

- Selon (70) : $x_n = 0 + \frac{0.5-0}{2}(\xi_n + 1) = \frac{1}{4}(\xi_n + 1)$
- Selon (72) : $F = \int_0^{0.5} e^x dx = \frac{1}{4}[\alpha_1 f(x_1) + \alpha_2 f(x_2) + \cdots + \alpha_N f(x_N)]$
- Pour *N=2* , on obtient les valeurs des ξ_n $(\pm 1/\sqrt{3})$ et des α_n (1, 1) selon le tableau ci-dessus :
  $x_{1,2} = \frac{1}{4}(\pm 1/\sqrt{3} + 1) = 0.105662, 0.394338 \qquad \int_0^{0.5} e^x dx \approx \frac{1}{4}[1e^{x_1} + 1e^{x_2}] = 0.648712$
- Pour *N=3* , selon le même tableau ci-dessus, on obtient :
  $x_{1,2,3} = \frac{1}{4}(\pm\sqrt{0.6} + 1{,}0 + 1) = 0.25 \pm 0.193649, 0.25$ $\int_0^{0.5} e^x dx \approx \frac{1}{4}\left[\frac{5}{9}e^{x_1} + \frac{5}{9}e^{x2} + \frac{8}{9}e^{x_3}\right]$ =0.648721.
- Pour *N=2*: 0.648712, pour *N=3*: 0.648721. La valeur exacte (analytique) est **0.648721**.

E20 L'accélération en m/s2 d'un véhicule est mesurée avec un accéléromètre à toutes les secondes (voir données ci-dessous et figure).



(a) Si la vitesse initiale du véhicule était de *4 m/s*, quelle est sa vitesse à $t = 6$ *s*? Utiliser la méthode trapézoïdale et la méthode de Simpson pour calculer l'intégrale. Calculer approximativement l'erreur d'intégration de la méthode trapézoïdale.

(b) Vérifier votre résultat en intégrant analytiquement le polynôme qui interpole les données.

(c) Selon l'analyse des coefficients d'interpolation, quel est l'ordre du polynôme qui a généré les données ?

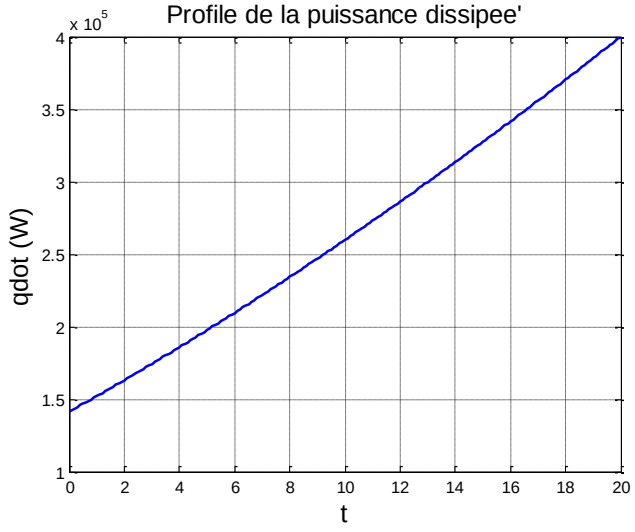
(d) Calculer l'erreur d'intégration de la méthode trapézoïdale et de Simpson en utilisant les dérivées du polynôme d'interpolation.

t (s)	0	1	2	3	4	5	6
a (m/s^2)	0.7	0.9	0.5	0.1	0.3	1.7	4.9

E21 La puissance en chaleur ($\dot{q}$ en Watt) dissipée par un véhicule se déplaçant à haute vitesse dans l'atmosphère est proportionnelle à sa vitesse v selon l'équation $\dot{q} = 50v^{3/2}$. Quand le véhicule accélère avec une accélération constante $a =$ 10 m/s2 de 200 m/s à 400 m/s, quelle est la chaleur dissipée en Joules. Utiliser la quadrature de Gauss (1) à 3 points et (2) à 5 points.

Note :

$$q = \int \dot{q}dt = \int kv^{3/2}dt = \int kv^{3/2}\frac{dt}{dv}dv = \int kv^{3/2}\frac{1}{a}dv$$



5 SOLUTION NUMÉRIQUE D'ÉQUATIONS NONLINÉAIRES

Définition et utilité

- *La solution numérique d'équations non linéaires* consiste à calculer numériquement les racines d'une équation *non linéaire transcendantale*, c'est-à-dire une équation de la forme $g(x) = 0$ pour laquelle il est impossible de trouver une solution algébrique de la forme $x = g^{-1}(0)$, par exemple :

$$a_3x^3 + a_2x^2 + a_1x + a_0 = 0 \qquad \cos x = x \qquad e^x \cos 2\pi x = 1 \tag{75}$$

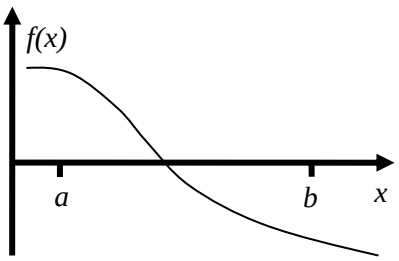
- Elle permet donc de calculer les racines d'une fonction ou, en général, le ou les points d'intersection entre deux fonctions non linéaires.

- Par exemple, si on recherche l'intersection entre $\cos x$ et x, on pose $f(x) = \cos x - x$ et on recherche la solution numérique au problème général :

$$f(x) = 0 \tag{76}$$

Une contrainte souvent rencontrée dans ce type de problème est que l'évaluation de la fonction $f(x)$ impose une charge de calcul considérable. L'évaluation de cette fonction avec un maillage de valeurs de x sur une grande plage de valeurs dans le but de rechercher la solution de façon brute est donc à proscrire. Elle serait aussi mal adaptée à des algorithmes devant opérer en temps réel. Il est donc essentiel de minimiser le nombre d'évaluations de la fonction $f(x)$ et de rechercher des façons efficaces de trouver la solution de $f(x) = 0$.

5.1 La méthode de bissection



- La méthode de bissection est la technique la plus élémentaire de trouver la racine d'une fonction *f(x)*.
- On suppose la recherche d'une racine dans l'intervalle $[a, b]$.
- L'algorithme est le suivant :
  1. Calculer le point milieu de l'intervalle $x_m = \frac{a+b}{2}$.
  2. Si $|f(x_m)| \leq \varepsilon$ où ε est une petite valeur positive choisie selon la précision requise, on arrête les itérations.
  3. Sinon, calculer$f(x_m)f(a)$: si ce produit est négatif, la racine est dans l'intervalle $[a, x_m]$ puisque la fonction a changé de signe. L'intervalle de recherche devient donc $[a, x_m]$. Si ce produit est positif, l'intervalle de recherche devient $[x_m, b]$.
  4. On retourne à l'étape 1 avec le nouvel intervalle de recherche.

- S'il y a plusieurs racines dans l'intervalle $[a, b]$, la condition$f(x_m)f(a) < 0$ implique qu'il y a un nombre impair de racines dans l'intervalle alors que la condition $f(x_m)f(a) > 0$ implique qu'il y a un nombre pair de racines, ou aucune. La méthode de bissection devient alors plus complexe à réaliser parce que l'algorithme doit reconnaître ces situations de racines multiples.
- Une façon brute d'identifier les cas de racines multiples est de faire un maillage fin de la variable indépendante *x* et d'observer les conditions où $f(x_n)f(x_{n+1}) < 0$. Si le maillage $|x_{n+1} - x_n|$est assez fin, chaque produit négatif indiquera la présence d'une seule racine dans $[x_n, x_{n+1}]$.
- Une autre difficulté de cette méthode est son incapacité à détecter les racines multiples en des points tangents à l'abscisse. Par exemple, la fonction $f(x) = (x-4)^2 = 0$ a deux racines à $x = 4$. Cependant, la fonction n'est jamais négative et donc il n'y aura jamais de produit négatif $f(x_m)f(a)$ à détecter.

5.2 La méthode de Newton-Raphson

- La méthode de Newton-Raphson (dénotée NR et parfois appelée simplement méthode de Newton) utilise une approximation de Taylor pour trouver la racine de l'équation.
- Selon la série de Taylor (23) de la section 1.2.2, évaluée autour d'un point arbitraire x_e, le calcul des racines de la fonction $f(x)$ implique la solution de :

$$f(x) = f(x_e) + \frac{f'(x_e)}{1!}(x - x_e) + \frac{f''(x_e)}{2!}(x - x_e)^2 + \ldots = 0 \quad \textbf{(77)}$$

- Le polynôme étant une série infinie transcendantale sans solution analytique, la méthode de NR utilise les deux premiers termes de la série de Taylor:

$$f(x_e) + f'(x_e)(x - x_e) = 0 \quad \textbf{(78)}$$

que l'on solutionne pour x :

$$x = x_e - \frac{f(x_e)}{f'(x_e)} \quad \textbf{(79)}$$

- À part le cas où la fonction originale $f(x)$ était une droite (auquel cas, la méthode de NR aurait été efficace mais inutile pour en trouver la racine), la solution obtenue avec (79) ne sera pas exactement la racine, à cause des termes d'ordre supérieur à 2 qui ont été négligés dans la série de l'équation (77).

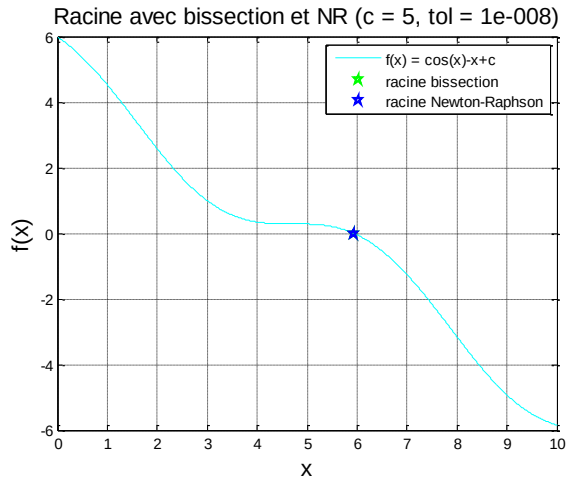
- Il faut donc itérer pour converger vers la solution :

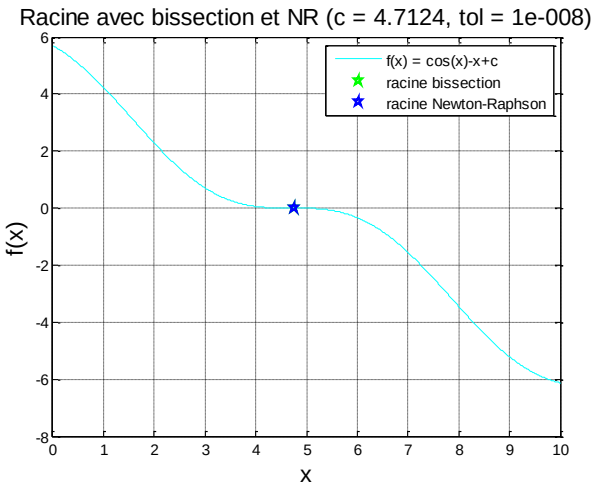
$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \quad \textbf{(80)}$$

- Comme pour la méthode de la bissection, les itérations sont arrêtées quand la précision requise est atteinte avec le test : $|f(x_m)| \leq \varepsilon$.

E22 Avec les méthodes de la bissection et de NR, calculer l'intersection des courbes $f_1(x) = \cos x$ et $f_2(x) = x - c$ (c est une constante), dans l'intervalle [0, 10] avec un seuil de précision de 1.0e-08. Comparer le nombre d'itérations requis par chacune des méthodes dans le cas où $c = 5$ et $c = 4.7124$ (une itération est un passage à l'intérieur de la boucle de l'algorithme). Expliquer le nombre différent d'itérations pour ces deux valeurs de c. Il ne faut pas oublier que l'on cherche à minimiser le nombre d'appel aux fonctions $f(x)$ et $f'(x)$. Il faut donc construire un algorithme en conséquence.

Réponses :
$c = 5$: Bissection prend 27 itérations, NR prend 4 itérations pour un départ à 0
$c = 4.7124$: Bissection prend 19 itérations, NR prend 12 itérations pour un départ à 0





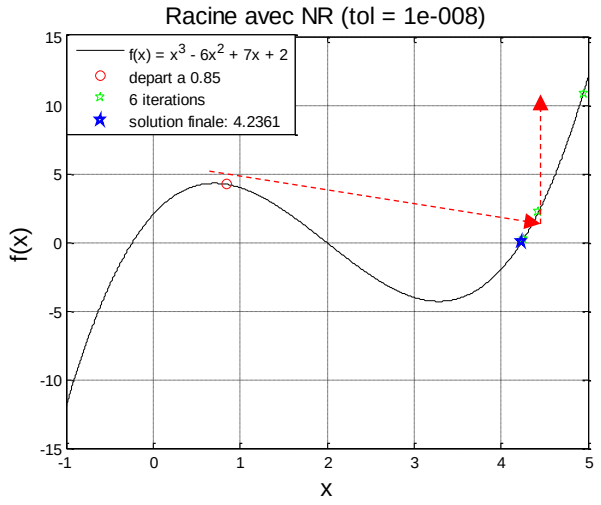
5.3 Racines multiples

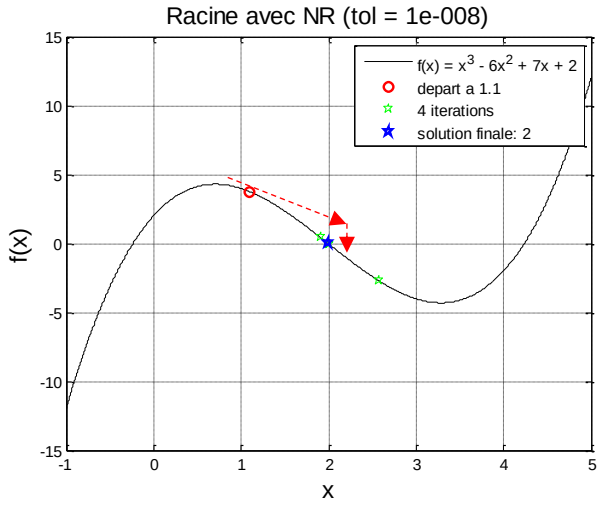
- En général, la méthode de NR prend moins d'itérations que la méthode de bissection pour obtenir la même précision dans le calcul de la racine.
- Cependant, elle n'est pas sans faiblesse pour certains types de fonctions.
- Dans le cas d'une fonction avec plusieurs racines et donc avec plusieurs solutions possibles, la solution obtenue dépend du point de départ des itérations. De plus, un point de départ près d'une racine ne signifie pas nécessairement que les itérations vont se terminer à cette racine. Voir l'exemple suivant.

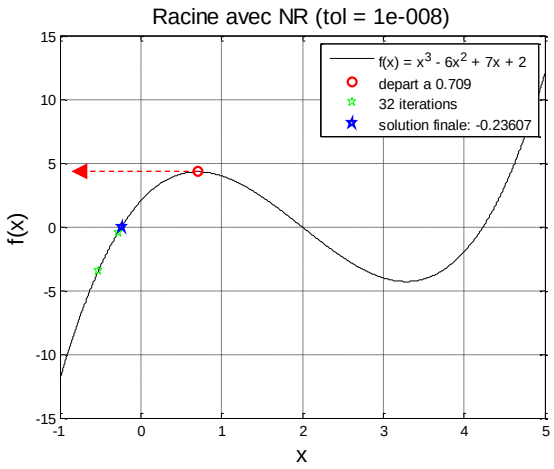
E23 Avec la méthode de NR, trouver les racines de la fonction $f(x) = x^3 - 6x^2 + 7x + 2$en démarrant les itérations aux valeurs initiales suivantes (1) $x = 0.85$, (2) $x = 1.10$, (3) $x = 0.709$ et (4) $x = 1.0$.

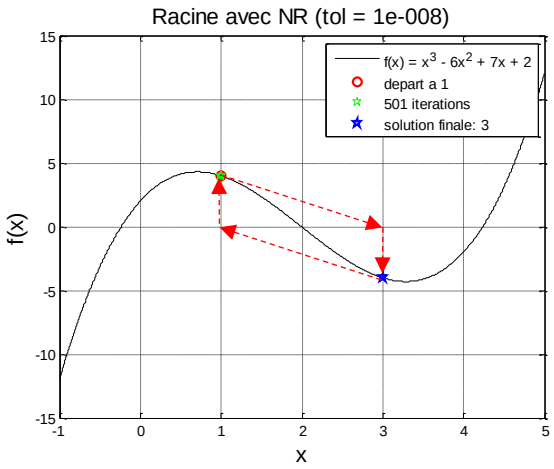
Réponses :

- $x = 0.85$ converge vers la racine à $x = +4.2361$ après 6 itérations
- $x = 1.10$ converge vers la racine à $x = +2.0000$ après 4 itérations
- $x = 0.709$ converge vers la racine à $x = -0.2361$ après 32 itérations
- $x = 1.00$ ne converge pas (arrêtée après 501 itérations): oscillations entre $x = 1$ et $x = 3$.









- Les flèches rouges indiquent la ou les premières itérations.
- La pente au point de départ des itérations détermine laquelle des racines sera obtenue.
- Un départ près d'un minimum ou d'un maximum de la fonction (case (3)) cause un grand nombre d'itérations et peut même porter les itérations vers $\pm\infty$ (pourquoi?).
- La présence de racines multiples implique des minima et maxima et donc, la possibilité d'oscillations sans convergence (cas (4)).

5.4 La méthode de la sécante

- Il arrive parfois que la dérivée de la fonction $f(x)$ soit difficile ou trop complexe à évaluer. Dans ce cas, la dérivée $f'(x)$ est approximée par une différence arrière, équations (45) et (46) :

$$f'(x) = \frac{\nabla f}{h} = \frac{f(x) - f(x-h)}{h} \qquad \textbf{(81)}$$

- Les itérations prennent donc la forme :

$$x_{k+1} = x_k + h_{k+1}$$

$$h_{k+1} = -\frac{f(x_k)h_k}{f(x_k) - f(x_{k-1})} \qquad \textbf{(82)}$$

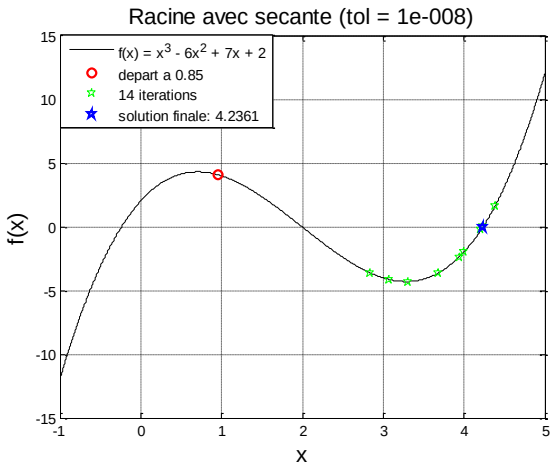
$$h_0 = x_0 - x_{00}$$

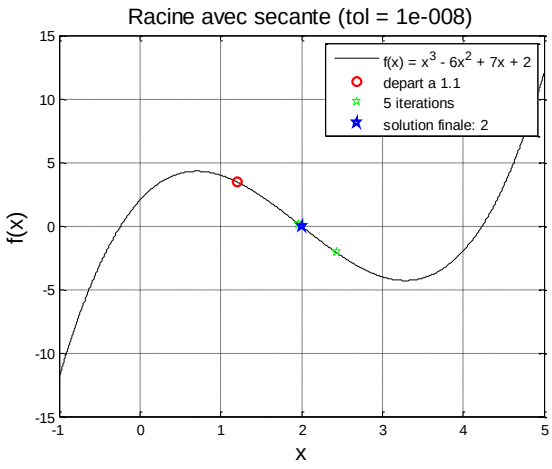
- Vu que la dérivée doit être évaluée avec une différence, il faut initialiser les itérations avec deux valeurs de la variable indépendante x, dénotées par x_0 et x_{00} ci-dessus.

E24 Démontrer les équations (82) à partir de l'algorithme de NR dont la dérivée est remplacée par la différence arrière.

E25 Trouver les racines de la fonction $f(x) = x^3 - 6x^2 + 7x + 2$ du problème E23 avec la méthode de la sécante en démarrant les itérations aux valeurs initiales suivantes (1) x_0 = 0.85, (2) x_0 =1.10. Prendre $x_{00} = x_0 - 0.1$.

Réponses :
- $x = 0.85$ converge vers la racine à $x = +4.2361$ après 14 itérations (8 de plus que la méthode NR)
- $x = 1.10$ converge vers la racine à $x = +2.0000$ après 5 itérations (1 de plus que la méthode NR)


Racine avec secante (tol = 1e-008)
f(x) = x3 - 6x2 + 7x + 2
depart a 0.85
14 iterations
solution finale: 4.2361
f(x)
x


Racine avec secante (tol = 1e-008)
f(x) = x3 - 6x2 + 7x + 2
depart a 1.1
5 iterations
solution finale: 2
f(x)
x

6 SOLUTION NUMÉRIQUE D'ÉQUATIONS DIFFÉRENTIELLES

6.1 Définition

- La solution numérique d'une équation différentielle consiste à trouver la solution *y(t)* en fonction du temps *t* à partir d'une *équation différentielle ordinaire* (EDO) de la forme :

$$\frac{dy}{dt} = f(y,t) \qquad \textbf{(83)}$$

où *f* est en général une fonction non linéaire de *y* et où le temps *t* peut apparaître explicitement (système variant dans le temps) ou non (système invariant dans le temps).

- La variable dépendante *y* est en général une matrice contenant plusieurs variables dépendantes inter reliées par la fonction *f*.
- La plupart des algorithmes d'intégration numérique traitent un système d'équations d'ordre 1 plutôt que des équations d'ordre supérieur à 1 (certains algorithmes dédiés aux systèmes mécaniques acceptent des équations d'ordre 2).
- Donc, tout système qui contient des dérivées d'ordre $N > 1$ doit être transformé de façon à ce que chaque équation d'ordre *N* soit réduite à *N* équations d'ordre 1. La procédure est la même que celle requise pour transformer une équation différentielle d'ordre *N* en un modèle variables-d'état *A, B, C, D* d'ordre *N* (voir un exemple à l'Annexe B, section B2.2)

- Dans le cas **où *f* ne dépend pas de *y* mais seulement de la variable indépendante *t***, le problème se réduit à un problème d'intégration numérique (i.e. de quadrature) tel que vu à la Section 4.
- La dépendance de la dérivée ou 'pente locale' de *y* sur la valeur même de *y* implique une solution récursive où la valeur de *y* est propagée en direction de sa pente locale mais cette pente doit être recalculée puisqu'elle dépend de la valeur de *y* propagée.

- Avant de poursuivre dans la lecture de ces notes, la lecture de l'Annexe B du cours est recommandée.

6.2 Rappel des méthodes d'intégration de Euler

- La méthode de Euler a été vue à l'APP3; on la revoit rapidement ici.
- Selon la section B.4 de l'Annexe B, la méthode de Euler est une méthode :
  - très simple mais très imprécise de solution numérique
  - rarement utilisée en pratique
  - mais souvent utilisée pour illustrer des concepts de base.

- Les méthodes de *Euler explicite* (*EulerEX*) et de *Euler prédiction-correction* (*EulerPC*) seront utilisées ici comme démonstration.
- Il est démontré à la section B4.1 que la méthode d'intégration EulerEX est la une série de Taylor tronquée au deuxième terme :

$$\frac{dy}{dt} = f(y,t) \qquad \Rightarrow \qquad \hat{y}_{n+1} = y_n + f(y_n, t_n)\Delta t_n \qquad \textbf{(84)}$$

où Δt_n est le pas d'intégration de y_n à y_{n+1} et où la notation (^) est utilisée pour une valeur approximative.

- À la section B4.3, il est démontré que la méthode d'intégration EulerPC est en fait une méthode trapézoïdale d'intégration numérique :

Prédiction de la sortie avec EulerEX : $\hat{y}_{n+1}^{P} = y_n + f(y_n, t_n)\Delta t_n$

Prédiction de la pente à ce point : $f(\hat{y}_{n+1}^{P}, t_{n+1})$

Calcul de la pente moyenne : $\bar{f} = \frac{1}{2}[f(\hat{y}_{n+1}^{P}, t_{n+1}) + f(\hat{y}_n, t_n)]$ **(85)**

Correction avec la pente moyenne: $\hat{y}_{n+1} = y_n + \frac{1}{2}[f(y_n, t_n) + f(\hat{y}_{n+1}^{P}, t_{n+1})]\Delta t_n$

- Ces deux techniques seront utilisées pour évaluer les erreurs d'intégration et la stabilité des solutions numériques des EDO. Il est à noter que ces techniques sont plutôt imprécises et sont rarement utilisées dans la résolution de problèmes numériques. Leur intérêt est plutôt académique, pour fins de démonstration.

6.3 Erreur locale et erreur globale

- Les erreurs d'intégration numérique sont traitées à la section B.5.
- Pour une méthode d'intégration d'ordre *p*, l'*erreur locale* d_n est proportionnelle à Δt^{p+1}:

$$|d_n| \leq C\Delta t^{p+1} \quad \textbf{(86)}$$

où *C* est une constante arbitraire. L'*erreur globale* est d'ordre *p* par rapport au pas d'intégration :

$$|e_n| \leq K\Delta t^{p} \quad \textbf{(87)}$$

- En diminuant le pas d'intégration d'un facteur 2 par exemple, l'erreur globale sera diminuée d'un facteur 2^p.
- Le rapport d'erreur globale avec deux pas d'intégration différent est donc :

$$\frac{|e_n|}{|e_m|} \leq \left(\frac{\Delta t_n}{\Delta t_m}\right)^p \quad \textbf{(88)}$$

- Ce rapport d'erreur globale peut être transformée de façon a extraire explicitement l'ordre de la méthode :

$$\frac{|e_n|}{|e_m|} \leq \left(\frac{\Delta t_n}{\Delta t_m}\right)^p \quad \Rightarrow \quad \ln\left(\frac{e_n}{e_m}\right) \approx p \ln\left(\frac{\Delta t_n}{\Delta t_m}\right) \quad \textbf{(89)}$$

- Quand l'erreur globale *e* est mesurable, il possible de voir si celle-ci se comporte bien selon l'équation (87) en observant l'ordre *p* de la méthode tel que calculé par l'équation (89). Pour une série de pas d'intégration d'essai, si l'ordre *p* calculé selon (89) demeure constant, la solution numérique est dite *stable* ou *convergente* pour ces pas d'intégration. L'ordre de la méthode d'intégration peut être ainsi déduit si elle n'est pas connue.

- Quand l'erreur globale *e* n'est pas directement mesurable par la méthode d'intégration, l'erreur globale peut être approximée par la différence $e \approx \hat{y} - \hat{y}*$où $\hat{y}*$est la solution numérique obtenue avec un pas d'intégration beaucoup plus petit que celui utilisé pour obtenir $\hat{y}$.

E26 Cinq simulations numériques d'un problème dynamique ont été effectuées avec 5 pas d'intégration Δt différents. Les erreurs globales e correspondantes ont été mesurées :

e	Δt
0.08443	0.050
0.02603	0.040
0.01048	0.030
0.00319	0.020
0.00040	0.010

À partir de ces données, déduire l'ordre de la méthode d'intégration numérique. Déduire aussi à partir de quel pas d'intégration la solution numérique est convergente.

6.4 Convergence et stabilité

Les concepts de convergence et de stabilité sont décrits à la section B.6 de l'Annexe B. Ces concepts seront assimilés ici à partir d'exemples concrets sur MATLAB. En premier lieu, les codes MATLAB nécessaires seront développés.

E27 Développer les programmes MATLAB tels que décrits ci-dessous et les utiliser pour permettre d'atteindre les objectifs suivants :

- Savoir programmer et utiliser un intégrateur numérique sur MATLAB
- Traduire un problème aux équations différentielles ordinaires en une fonction MATLAB
- Connaître les méthodes d'intégration explicite et prédiction-correction
- Savoir reconnaître une instabilité numérique et en trouver la cause
- Connaître et savoir utiliser le lien entre l'erreur de discrétisation et le pas d'intégration
- Reconnaître l'ordre d'une méthode numérique

Les deux intégrateurs numériques utilisés sont:

(i) la méthode de Euler explicite (EULEREX)
(ii) la méthode de Euler avec prédiction et correction (EULERPC – moyenne des pentes)

Le code MATLAB de ces intégrateurs numériques est fourni.

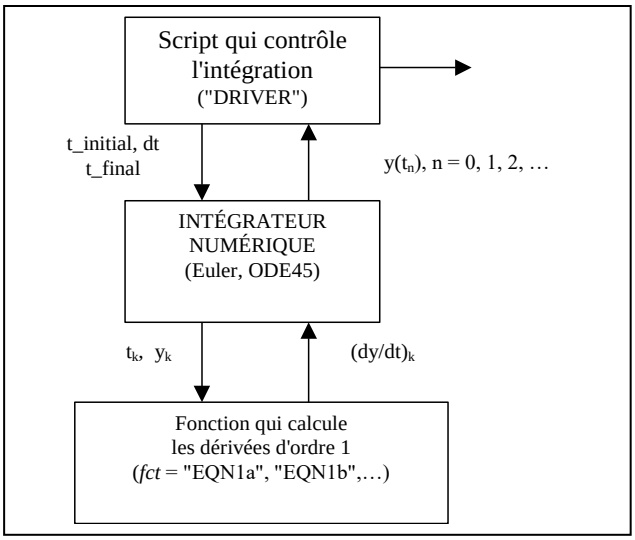
Chaque intégrateur est une fonction MATLAB et reçoit les variables suivantes à l'entrée:

fct = variable qui contient le nom de la fonction MATLAB décrivant l'EDO à solutionner
y_0 = conditions initiales
t_0 = temps initial
dt = pas d'intégration (fixe)
t_f = temps de fin de simulation

La fonction retourne les variables dépendantes *y* et indépendante *t*. La fonction MATLAB *feval* permet d'évaluer une fonction à partir d'une variable contenant son nom, par exemple :

```
fct = 'EDO'
f   = feval(fct,t,y)
```

qui est équivalent à *f = EDO(t,y)* où *EDO* fait référence au fichier fonction EDO.m qui contient l'équation différentielle ordinaire à intégrer. Un troisième intégrateur fourni par MATLAB (ODE45) est aussi utilisé pour fins de comparaison. La façon d'utiliser un intégrateur numérique est illustrée ci-dessous.



Pour toutes les questions du type : "Vérifier que la solution analytique est…", on ne demande pas de retrouver cette solution à partir de l'équation différentielle mais, dans le sens inverse, de démontrer que la solution proposée satisfait l'équation différentielle (par substitution de la solution dans l'EDO).

Problème 1a : Système marginalement stable
L'EDO d'un système est donnée par:

$$\frac{d^2y}{dt^2} + 9y = 0.$$

(a) Avec $y_0 = [1, 0]$, vérifier que la solution analytique exacte est $y(t) = \cos(3t)$. La disponibilité d'une solution exacte pour ce problème permet de calculer et d'évaluer ainsi la précision des solutions numériques.

(b) Écrire une fonction MATLAB (appelée 'eqn1a') sous forme standard qui retourne les dérivées d'ordre 1 $f(t, y)$ de ce système dynamique:

```
function f = eqn1a(t,y).
```

Les dérivées $f(t, y)$ doivent toujours être sous la forme d'une colonne (standard MATLAB).

(c) Utiliser ensuite le script prob1a.m pour faire le reste du problème. Le script proba1a.m est le "driver" de la figure ci-dessus qui appelle les intégrateurs numériques (EULEREX, EULERPC, ODE45) qui eux, à leur tour, appellent la fonction-script *eqn1a.m* développé en (b) ci-dessus. Le script proba1a.m exécute de multiples intégrations numériques avec les trois intégrateurs avec, à chaque fois, un pas d'intégration plus petit (divisé par 2 à chaque nouvelle intégration).

(d) Analyser les graphiques obtenus et conclure sur l'ordre des méthodes d'intégration et sur leur domaine de convergence en fonction du pas d'intégration.

Problème 1b. Système stable '*stiff*' couplé homogène

Pour le système linéaire stationnaire suivant :

$$\frac{dy_1}{dt} = +998\, y_1 + 1998\, y_2 \qquad y_1(0) = 1$$
$$\frac{dy_2}{dt} = -999\, y_1 - 1999\, y_2 \qquad y_2(0) = 1$$

(a) Démontrer que la solution exacte y_E est donnée par :

$$y_{E1}(t) = +4\, e^{-t} - 3\, e^{-1000t} \qquad y_{E2}(t) = -2\, e^{-t} + 3\, e^{-1000t}$$

(b) Écrire une fonction MATLAB (appelée 'eqn1b') sous forme standard qui donne les dérivées.
(c) Utiliser ensuite le SCRIPT prob1b.m (le "driver") pour faire le reste du problème. Le script calcule 5 solutions numériques avec EULEREX et EULERPC avec 5 pas d'intégration différents.
(d) Conclure sur la stabilité des solutions numériques en fonction du pas d'intégration.

Problème 1c. Système non-homogène
Pour le système suivant :

$$\frac{dy}{dt} = 5(y - t^2)$$

(a) Démontrer que la solution exacte *y* est donnée par :

$$y\,(t) \;=\; Ae^{5t} \;+\; t^2 \;+\; 0.4\,t \;+\; 0.08$$

où A est une constante d'intégration qui dépend des conditions initiales. Démontrer que dans le cas où $y_0 \;=\; 0.08$, la solution devient : $y\,(t) \;=\; t^2 \;+\; 0.4\,t \;+\; 0.08$.

(b) Écrire une fonction MATLAB (appelée 'eqn1c') sous forme standard qui donne les dérivées.
(c) Utiliser ensuite le SCRIPT prob1c.m (le "driver") pour faire le reste du problème. Le script calcule 3 solutions numériques avec EULEREX et EULERPC avec 3 pas d'intégration différents.
(d) Conclure sur la stabilité des solutions numériques en fonction du pas d'intégration.

6.5 La méthode Runge-Kutta et ODE45

E28 Développer le programme MATLAB tel que décrit ci-dessous et l'utiliser pour permettre d'atteindre les objectifs suivants :

- Traduire un problème aux équations différentielles ordinaires en une fonction MATLAB
- Initier à l'utilisation d'un intégrateur numérique à pas variable de type Runge-Kutta.
- Vérifier l'effet de la tolérance d'intégration sur la solution numérique et sur le nombre de pas requis.

- Utiliser l'intégrateur ODE45 sur MATLAB. Faire **help ode45** sur MATLAB pour avoir l'information.
- [T,Y] = ODE45('F',TSPAN,Y0,OPTIONS) avec TSPAN = [T0 TFINAL]
- Intègre le système d'équations dy/dt = F(t,y) de T0 à TFINAL avec conditions initiales Y0.
- 'F' est le nom du fichier qui calcule les EDO d'ordre 1 à intégrer.
- La fonction F(T,Y) doit retourner un vecteur colonne.
- Chaque rangée de la solution Y correspond au temps dans la colonne T.

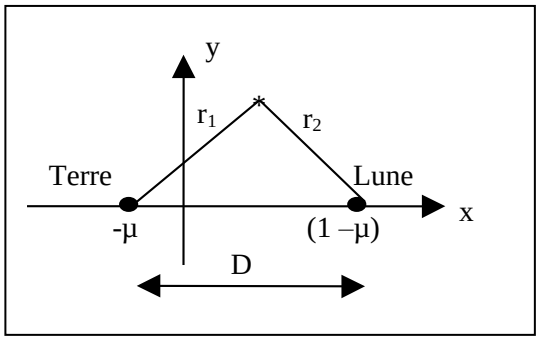
- Sans l'argument "OPTIONS", les paramètres d'intégration nominaux sont utilisés.
- Pour introduire de nouveaux paramètres, il faut les passer par la variable OPTIONS, qui est elle-même créée par la fonction ODESET (voir help odeset).
- Pour changer la tolérance d'intégration relative ("RelTol") de 1.0e-03 (valeur par défaut) à 1.0e-06, on utilise: options = odeset('RelTol',1e-6);.

Le problème du calcul de la trajectoire d'un véhicule (la capsule Apollo) autour de deux corps beaucoup plus massifs (la Terre et la Lune) fait partie du "problème gravitationnel restreint à trois corps". Bien que ce problème soit bien connu, aucune solution analytique n'a été obtenue. Toute solution requiert l'usage d'un intégrateur numérique. On suppose la trajectoire dans un plan xy avec l'origine placée au centre de masse du système Terre-Lune et avec l'axe des x selon la direction Terre-Lune. En utilisant des variables (position et vitesse) normalisées par la distance D entre la Terre et la Lune (D ≈ 384 000 km), on obtient :

$$\frac{d^2x}{dt^2} = +2\frac{dy}{dt} + x - \frac{\mu^*(x+\mu)}{r_1^3} - \frac{\mu(x-\mu^*)}{r_2^3} \qquad \frac{d^2y}{dt^2} = -2\frac{dx}{dt} + y - \frac{\mu^* y}{r_1^3} - \frac{\mu y}{r_2^3}$$

où

$$\mu = \frac{\text{masse Lune}}{\text{masse Terre} + \text{masse Lune}} = \frac{1}{82.45} \text{ et } \mu^* = 1 - \mu$$



Les distances r_1 et r_2 entre la capsule et les centres de masse sont :

$$r_1 = \sqrt{(x+\mu)^2 + y^2}$$
$$r_2 = \sqrt{(x-\mu^*)^2 + y^2}$$

Avec les variables normalisées, la position de la Terre est $x = -\mu$ et celle de la Lune est $x = \mu^* = (1-\mu)$.

On utilisera les conditions initiales (normalisées) suivantes :

$$x(0) = 1.2,\ y(0) = 0.0, \quad v_x(0) = \left(\frac{dx}{dt}\right)_0 = 0.0, v_y(0) = \left(\frac{dy}{dt}\right)_0 = -1.04935751.$$

La capsule part du côté caché de la Lune à une distance d'environ 0.2D de la Lune, avec une vitesse initiale dans la direction $-y$.

(a) Transformer les équations en 4 équations d'ordre 1.

(b) Écrire le fichier "apollo.m" qui calcule les dérivées $f(t,z)$ où z contient les quatre variables d'état.

(c) À partir de la ligne de commande MATLAB, faire une première simulation (z1) avec ODE45 de $t_0 = 0$ à $t_f = 6.19$ en utilisant les paramètres d'intégration nominaux i.e. avec RelTol = 1e-03. Faire le graphique de la trajectoire avec des points bleus par exemple:

```
» plot(z1(:,1),z1(:,2), '.b')
```

Note : Il est possible de placer la Terre et la Lune sur le graphique avec les trois commandes :

```
» hold on        » plot((1-u),0,'o')        » plot(-u,0,'p')
```

Remarquer l'espacement des points d'intégration au départ, près de la Terre, de la Lune et dans l'espace.

(d) Faire une deuxième simulation (z2) avec RelTol = 1e-06. Faire le graphique de la trajectoire sur la même figure avec des marqueurs de couleur verte:

```
» plot(z2(:,1),z2(:,2),'.g')
```

Remarquer l'effet du changement de tolérance sur (1) l'espacement des points d'intégration et (2) l'erreur globale à la fin de la trajectoire.

E29 Des lapins et des renards (… par Jean de Lafontaine)

Un système de commande non linéaire (souvent rencontré de façon naturelle dans les systèmes biologiques et les écosystèmes) permet de régulariser un système instable de la forme $dx_1/dt = x_1$ (dont la solution est exponentielle : $x_1 = x_1(0)\, e^t$) par l'ajout d'une rétroaction non linéaire négative de la forme ($a\, x_1\, x_2$) où x_2 est généré par un système stable. Le modèle mathématique d'un tel système est, par exemple :

$$\frac{dx_1}{dt} = 2x_1 - ax_1x_2 \qquad \frac{dx_2}{dt} = -x_2 + ax_1x_2$$

$$a \geq 0.$$

Avec $a = 0$, x_1 est instable (tend vers l'infini) et x_2 est stable (tend vers zéro). Avec $a > 0$, le couplage du système stable sur le système instable tend à stabiliser ce dernier.

Selon certaines recherches, ce modèle représente bien les variations de population de lapins (x_1) et de leurs prédateurs, les renards (x_2). Quand il n'y a pas de renard ($x_2 = 0$ et $dx_2/dt = 0$), la population des lapins augmente de façon exponentielle, faute d'ennemis naturels (système instable). Quand il n'y a pas de lapin ($x_1 = 0$ et $dx_1/dt = 0$), la population des renards disparaît exponentiellement, faute de nourriture (système stable). Quand les deux coexistent, les renards se nourrissent de lapins avec une probabilité proportionnelle au produit de leur population respective, ax_1x_2 (rétroaction naturelle négative), diminuant ainsi la population de lapins et favorisant l'expansion de la population de renards. Le facteur a est une mesure de cette probabilité de rencontre renard-lapin.

Trouver la solution numérique de ces équations pour prédire la population des lapins et des renards sur une période de 20 ans (t est en année dans ce modèle) en supposant une population initiale de $x_1(0) = 200$ lapins et de $x_2(0) = 100$ renards et une probabilité de rencontre de 1 % ($a = 0.01$). Commenter le résultat graphique.

D8 Des lapins, des renards et de l'environnement

Selon le modèle ci-dessus, s'il n'y a pas de prédateurs ($x_2 = 0$), la population de lapins augmente à un taux de :

$$\frac{dx_1}{dt} = 2x_1$$

La solution de cette équation, $x_1(t) = x_1(0)e^{+2t}$, indique une augmentation sans limite de cette population. En pratique, l'environnement ne peut fournir la nourriture nécessaire à une infinité de lapins et il y a, ici encore, un contrôle automatique qui limite cette croissance. Le modèle mathématique est donc corrigé avec un facteur additionnel dans l'équation pour x_1 :

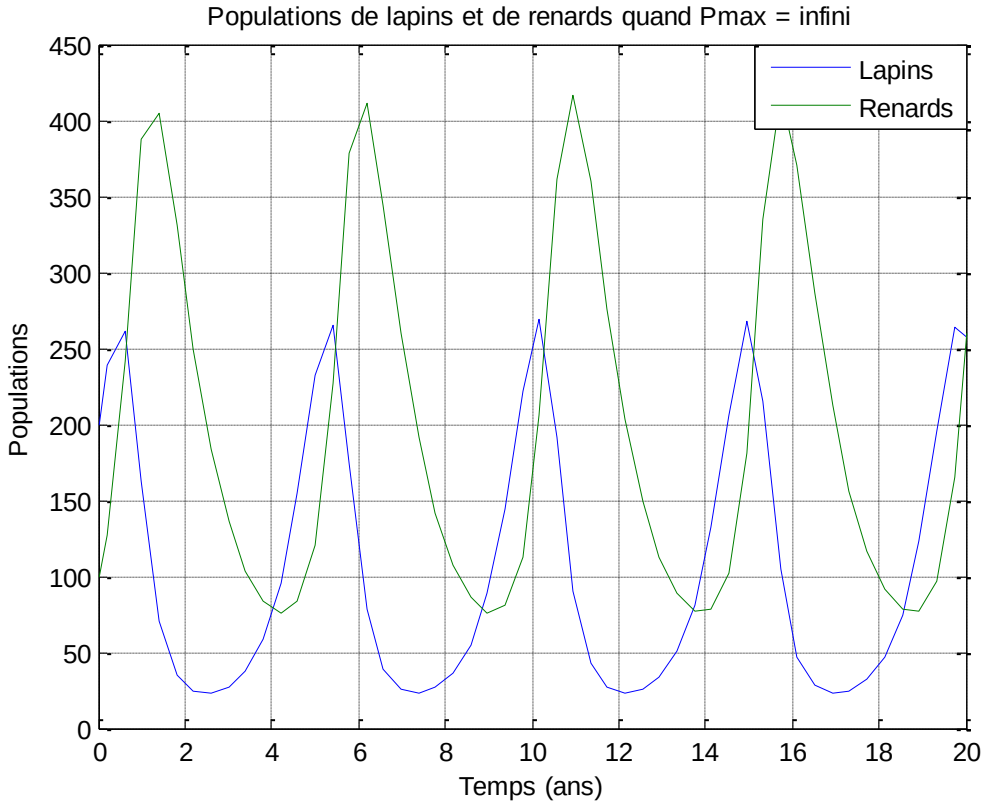
$$\frac{dx_1}{dt} = 2x_1\left(1 - \frac{x_1}{P_{max}}\right) - ax_1x_2 \qquad \frac{dx_2}{dt} = -x_2 + ax_1x_2$$

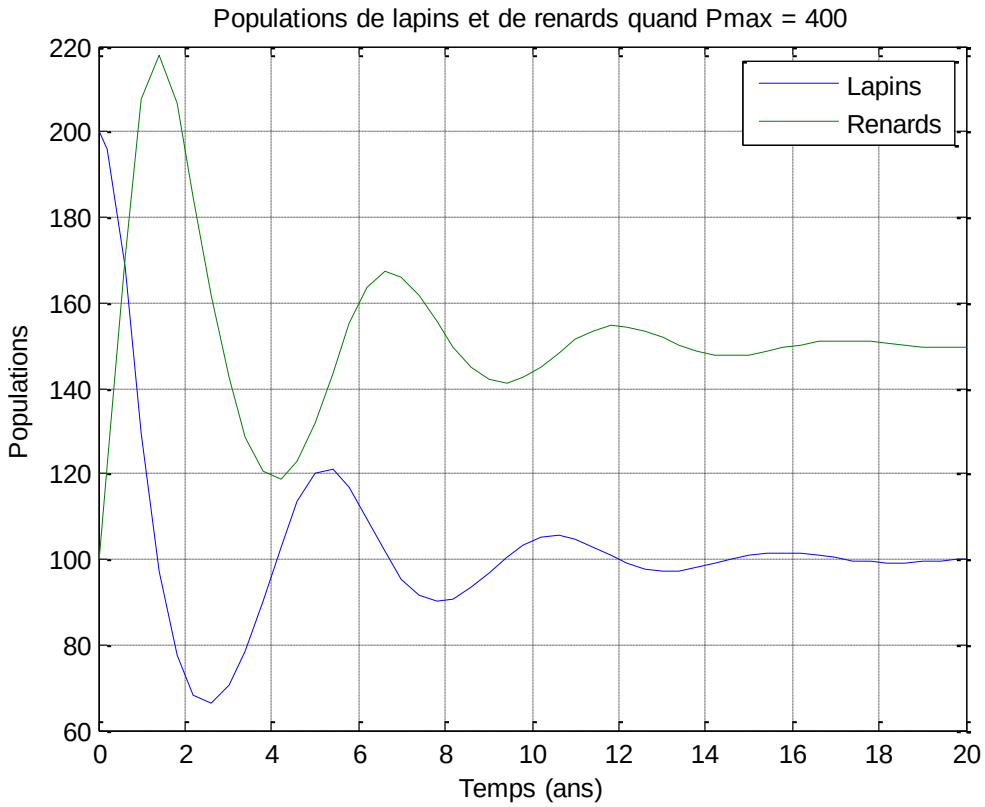
$$a > 0.$$

Il peut être observé que sans renards ($x_2 = 0$), la population de lapins x_1 augmentera jusqu'à la valeur limite P_{max} où dx_1/dt devient nul et la population cesse de croître. Quand $P_{max} \to \infty$, le modèle original est obtenu.

Refaire la simulation numérique avec les mêmes conditions initiales ($x_1(0) = 200$ lapins et $x_2(0) = 100$ renards) et paramètre ($a = 0.01$) ci-dessus et avec une population maximale $P_{max} = 400$ lapins. Commenter le résultat graphique.

Les graphiques obtenus sont sur la page suivante.


Populations de lapins et de renards quand Pmax = infini
Lapins
Renards
Populations
Temps (ans)
0
50
100
150
200
250
300
350
400
450
0
2
4
6
8
10
12
14
16
18
20


Populations de lapins et de renards quand Pmax = 400
Lapins
Renards
Populations
Temps (ans)
60
80
100
120
140
160
180
200
220
0
2
4
6
8
10
12
14
16
18
20

ANNEXE

Démonstrations

1

D.1 Equation normale pour $g(x) = mx$

- $E = \sum_{k=1}^{N} \left[g(x_k) - y_k \right]^2 = \sum_{k=1}^{N} \left[mx_k - y_k \right]^2$
- $\frac{\partial E}{\partial m} = 0 \quad \Rightarrow \quad \sum_{k=1}^{N} 2 \left[mx_k - y_k \right] x_k = 0$
- $2m \sum_{k=1}^{N} x_k^2 - 2 \sum_{k=1}^{N} y_k x_k = 0$
- $m = \dfrac{\sum_{k=1}^{N} y_k x_k}{\sum_{k=1}^{N} x_k^2}$

D.2 Equations normales pour $g(x) = mx + b$

- $E = \sum_{k=1}^{N} \left[g(x_k) - y_k \right]^2 = \sum_{k=1}^{N} \left[mx_k + b - y_k \right]^2$
- $\frac{\partial E}{\partial m} = 0 \quad$ et $\quad \frac{\partial E}{\partial b} = 0 \quad \Rightarrow$ pour minimiser E
- $\frac{\partial E}{\partial m} = 0 \Rightarrow \sum_{k=1}^{N} 2 \left[mx_k + b - y_k \right] x_k = 0$
- $\frac{\partial E}{\partial b} = 0 \Rightarrow \sum_{k=1}^{N} 2 \left[mx_k + b - y_k \right] = 0$
- $m \sum_{k=1}^{N} x_k^2 + b \sum_{k=1}^{N} x_k = \sum_{k=1}^{N} y_k x_k$
- $m \sum_{k=1}^{N} x_k + bN = \sum_{k=1}^{N} y_k$

$$\begin{bmatrix} N & \sum x_k \\ \sum x_k & \sum x_k^2 \end{bmatrix} \begin{bmatrix} b \\ m \end{bmatrix} = \begin{bmatrix} \sum y_k \\ \sum y_k x_k \end{bmatrix}$$

2

D.3 Equations normales pour l'approximation à M paramètres

- $g(x) = a_1 \phi_1(x) + a_2 \phi_2(x) + \ldots + a_M \phi_M(x) = \sum_{i=1}^{M} a_i \phi_i(x)$
- $E = \sum_{n=1}^{N} \left[g(x_n) - y_n \right]^2 = \sum_{n=1}^{N} \left[\sum_{i=1}^{M} a_i \phi_i(x_n) - y_n \right]^2$
- Condition : $\frac{\partial E}{\partial a_j} = 0$ pour $j = 1, M$
- $\frac{\partial E}{\partial a_j} = \sum_{n=1}^{N} 2 \left[\sum_{i=1}^{M} a_i \phi_i(x_n) - y_n \right] \phi_j(x_n) = 0 \qquad j = 1, M$

$$\Rightarrow \sum_{n=1}^{N} \sum_{i=1}^{M} \left[a_i \phi_i(x_n) - y_n \right] \phi_j(x_n) = \sum_{n=1}^{N} y_n \phi_j(x_n)$$

- Change l'ordre de la sommation

$$\Rightarrow \sum_{i=1}^{M} a_i \sum_{n=1}^{N} \phi_i(x_n) \phi_j(x_n) = \sum_{n=1}^{N} y_n \phi_j(x_n) \qquad j = 1, M$$

- Met sous forme matricielle :

$$\begin{matrix} j=1 \Rightarrow \\ j=2 \Rightarrow \\ \\ j=M \Rightarrow \end{matrix} \begin{bmatrix} \sum_{n=1}^{N} \phi_1(x_n)\phi_1(x_n) & \sum_{n=1}^{N} \phi_2(x_n)\phi_1(x_n) & \cdots \\ \sum_{n=1}^{N} \phi_1(x_n)\phi_2(x_n) & \sum_{n=1}^{N} \phi_2(x_n)\phi_2(x_n) & \cdots \\ \vdots & \vdots & \vdots \\ \sum \phi_1(x_n)\phi_M(x_n) & \sum \phi_2(x_n)\phi_M(x_n) & \cdots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} \sum_{n=1}^{N} \phi_1(x_n) y_n \\ \sum_{n=1}^{N} \phi_2(x_n) y_n \\ \vdots \\ \sum_{n=1}^{N} \phi_M(x_n) y_n \end{bmatrix}$$

15/06/08 6:19 PM G:\COURS UdeS\Session S4 Ete2008\APP4\Chap1\D4.m 1 of 1

3

```
% D4

  clc
  close all
  clear

%Preparation des donnees de mesures

  t = [2 3 4 6 7 8 10 11 12 15]';
  y = [2.11 1.61 1.25 0.820 0.737 0.810 0.880 0.443 0.070 -0.493]';

  figure, plot(t, y, 'x')

% Frequence angulaire des oscillations

  P = 8.0;
  w = 2*pi/P;

% Solution du probleme

% y = a1 + a2 t + a3 cos(wt) + a4 sin(wt)
% phi1 = 1
% phi2 = t
% phi3 = cos(wt)
% phi4 = sin(wt)

  phi(:,1) = ones(size(y));
  phi(:,2) = t;
  phi(:,3) = cos(w*t);
  phi(:,4) = sin(w*t);

  A = [phi(:,1)'*phi(:,1), phi(:,1)'*phi(:,2), phi(:,1)'*phi(:,3), phi(:,1)'*phi(:,4);
       phi(:,2)'*phi(:,1), phi(:,2)'*phi(:,2), phi(:,2)'*phi(:,3), phi(:,2)'*phi(:,4);
       phi(:,3)'*phi(:,1), phi(:,3)'*phi(:,2), phi(:,3)'*phi(:,3), phi(:,3)'*phi(:,4);
       phi(:,4)'*phi(:,1), phi(:,4)'*phi(:,2), phi(:,4)'*phi(:,3), phi(:,4)'*phi(:,4) ];

  B = [phi(:,1)'*y, phi(:,2)'*y, phi(:,3)'*y, phi(:,4)'*y]';

  a = inv(A)*B

  tc = [0:.5:15]';
  yc = a(1) + a(2)*tc + a(3)*cos(w*tc) + a(4)*sin(w*tc);

  figure, plot(t, y, 'p', tc, yc, 'r', 'LineWidth',2)
  xlabel('t', 'Fontsize',15)
  ylabel('y', 'Fontsize',15)
  title('D4', 'Fontsize',15)

  vitesse = a(2)
  amplitude = sqrt(a(3)*a(3) + a(4)*a(4))
```

```
% D5

  clc
  close all
  clear

%Preparation des donnees de mesures

  t = [2 3 4 6 7 8 10 11 12 15]';
  y = [2.11 1.61 1.25 0.820 0.737 0.810 0.880 0.443 0.070 -0.493]';

  figure, plot(t, y, 'x')

% Frequence angulaire des oscillations

  P = 8.0;
  w = 2*pi/P;

% Solution du probleme

% y = a1 + a2 t + a3 cos(wt) + a4 sin(wt)
% phi1 = 1
% phi2 = t
% phi3 = cos(wt)
% phi4 = sin(wt)

  phi(:,1) = ones(size(y));
  phi(:,2) = t;
  phi(:,3) = cos(w*t);
  phi(:,4) = sin(w*t);

  P = [phi(:,1), phi(:,2), phi(:,3), phi(:,4)];

  a = inv(P'*P)*P'*y

  tc = [0:.5:15]';
  yc = a(1) + a(2)*tc + a(3)*cos(w*tc) + a(4)*sin(w*tc);

  figure, plot(t, y, 'b', tc, yc, 'r', 'LineWidth',2)
  xlabel('t', 'Fontsize',15)
  ylabel('y', 'Fontsize',15)
  title('D5', 'Fontsize',15)

  vitesse = a(2)
  amplitude = sqrt(a(3)*a(3) + a(4)*a(4))
```

D.6 Interpolation par splines cubiques

- Entre les points x_k et x_{k+1}, $k = 0, 1, \ldots N-1$

  $g(x)$ = cubique (ordre 3)

  $g'(x)$ = parabolique (ordre 2)

  $g''(x)$ = droite (ordre 1)

- On utilise le polynôme de Lagrange pour interpoler entre $g''(x_k) = g''_k$ et $g''(x_{k+1}) = g''_{k+1}$:

$$g''(x) = g''_k \frac{(x_{k+1} - x)}{(x_{k+1} - x_k)} + g''_{k+1} \frac{(x - x_k)}{(x_{k+1} - x_k)} \quad \Rightarrow \quad h \triangleq (x_{k+1} - x_k)$$

$k = 0, \ldots, N-1$

- On intègre une fois :

$$g'(x) = \frac{g''_k}{2} \frac{(x_{k+1} - x)^2}{h} + \frac{g''_{k+1}}{2} \frac{(x - x_k)^2}{h} + a_k$$

a_k : constante d'intégration

- On intègre une autre fois :

$$g(x) = \frac{g''_k}{6} \frac{(x_{k+1} - x)^3}{h} + \frac{g''_{k+1}}{6} \frac{(x - x_k)^3}{h} + a_k x + b_k$$

- On redéfinit les constantes d'intégration pour avoir une forme plus simple (c'est un truc...)

$$a_k x + b_k = A_k (x - x_k) + B_k (x_{k+1} - x)$$

- On remplace dans $g(x)$ ci-dessus.

- On trouve la solution pour A_k et B_k en appliquant les conditions que $g(x)$ doit interpoler y_k et y_{k+1}:

$$g(x_k) = y_k \quad \text{et} \quad g(x_{k+1}) = y_{k+1}$$

- On obtient:

$$g_k'' \frac{h^2}{6} + B_k h = y_k \quad \text{et} \quad g_{k+1}'' \frac{h^2}{6} + A_k h = y_{k+1}$$

- On solutionne pour A_k et B_k:

$$B_k = \frac{y_k}{h} - \frac{g_k'' h}{6} \qquad A_k = \frac{y_{k+1}}{h} - \frac{g_{k+1}'' h}{6}$$

- On remplace dans $g(x)$:

$$g(x) = \frac{g_k''}{6}\left[\frac{(x_{k+1}-x)^3}{h} - h(x_{k+1}-x)\right] + \frac{g_{k+1}''}{6}\left[\frac{(x-x_k)^3}{h} - h(x-x_k)\right]$$
$$+ y_k \frac{(x_{k+1}-x)}{h} + y_{k+1} \frac{(x-x_k)}{h}$$

- Cette équation :
  - ⇒ interpole les coordonnées $(x_k\ y_k)$ et $(x_{k+1}\ y_{k+1})$ grâce au choix de A_k, B_k
  - ⇒ interpole les 2e dérivées $(x_k\ g_k'')$ et $(x_{k+1}\ g_{k+1}'')$
- Cependant, elle contient $N+1$ inconnues:
  g_k'', $k = 0, \ldots, N$.
- On utilise la continuité de la 1ère dérivée $g'(x)$ aux $N-1$ interconnections pour obtenir $N-1$ contraintes sur les $N+1$ inconnues.

- Il va manquer 2 contraintes : on choisira g_0'' et g_N'' en imposant une valeur appropriée.
- Pour une spline 'naturelle' : $g_0'' = g_N'' = 0$
- De retour à $g(x)$, on dérive pour $g'(x)$:

$$g'(x) = \frac{g_k''}{6}\left[\frac{-3(x_{k+1}-x)^2 + h}{h}\right] + \frac{g_{k+1}''}{6}\left[\frac{3(x-x_k)^2 - h}{h}\right] + \frac{(y_{k+1}-y_k)}{h}$$

- On évalue $g'(x)$ at x_k et x_{k+1} :

$$g'(x_k) = -\frac{g_k'' h}{3} - \frac{g_{k+1}'' h}{6} + \frac{(y_{k+1}-y_k)}{h} \qquad (1)$$

$$g'(x_{k+1}) = \frac{g_k'' h}{6} + \frac{g_{k+1}'' h}{3} + \frac{(y_{k+1}-y_k)}{h} \qquad (2)$$

- Pour avoir continuité dans $g'(x)$, on change $k+1$ pour k dans (2) et on égale à (1) :

$$\frac{g_{k-1}'' h}{6} + \frac{g_k'' h}{3} + \frac{(y_k - y_{k-1})}{h} = -\frac{g_k'' h}{3} - \frac{g_{k+1}'' h}{6} + \frac{(y_{k+1}-y_k)}{h}$$

- En simplifiant, on obtient :

$$g_{k+1}'' + 4g_k'' + g_{k-1}'' = \frac{6}{h^2}\left[y_{k+1} - 2y_k + y_{k-1}\right] \qquad k = 1, \ldots, N-1$$

- Sachant que $g_0'' = g_N'' = 0$, on a :

$k=1$ $\quad g_2'' + 4g_1'' = 6\left[y_2 - 2y_1 + y_0\right]/h^2$

$k=N-1$ $\quad 4g_{N-1}'' + g_{N-2}'' = 6\left[y_N - 2y_{N-1} + y_{N-2}\right]/h^2$