Université de Montpellier

Année Universitaire 2026

Faculté des Sciences

Analyse numérique élémentaire.

Pascal Azerad, Michel Cuer

23 février 2026

Table des matières

Les sections précédées d'une étoile ($\star$) sont des compléments facultatifs.

Introduction

Ce fascicule est destiné aux étudiants n'ayant pas suivi de cours d'analyse numérique en L1 ou L2. Il est en quelque sorte un préliminaire au cours d'analyse numérique des équations différentielles de L3. Il est extrait d'un poly rédigé en 2020, en collaboration avec Michel Cuer. Les algorithmes sont écrits en MATLAB mais la compréhension des codes devrait aller de soi si vous êtes habitué à Python ou un autre langage.

0.1 Motivations

Les outils de l'analyse numérique sont utilisés dans toutes les disciplines et sont appliqués dans toutes les technologies : bureau d'étude pour la conception des appareils ou machines que nous rencontrons dans la vie courante (automobiles, avions, smartphones...), appareils médicaux comme les scanners, contrôle dans les machines de travaux publics, prévision du temps, modélisation des épidémies...Une formation mathématique équilibrée devrait comporter une initiation aux techniques du calcul scientifique, quel que soit l'emploi auquel on se destine.

0.2 Signification des termes utilisés dans le programme

Dans le premier thème, représentation des nombres en machine, on s'intéresse aux effets du codage des nombres en machine sur la précision des calculs. Il faut être conscient du fait qu'il n'est pas possible de représenter en machine tous les nombres réels (un nombre réel peut contenir une infinité de décimales) ; il faut faire des approximations et il y a des erreurs d'arrondis. Les systèmes de calculs formels comme `sagemath`, `MAPLE`, `Mathematica` ou `MuPAD` manipulent des symboles sans introduire d'erreur d'arrondis, mais il est impossible de traiter ainsi tous les problèmes de mathématiques ; il faut faire des approximations à un moment ou à un autre.

Dans le deuxième thème, très classique, et déjà rencontré dans le secondaire, on traite les méthodes numériques de calcul des racines d'une équation $f(x) = 0$ où f est une fonction réelle de variable réelle.

Dans le troisième thème, interpolation, on s'intéresse à la question de l'approximation par un polynôme en général, d'une fonction réelle de variable réelle, connue en un nombre fini de points.

Dans le quatrième thème, quadrature, on traite les méthodes numériques pour évaluer des intégrales définies $I = \int_a^b f(x)dx$.

Les sections précédées d'une étoile ($\star$) sont des compléments facultatifs.

Vous trouverez à la fin de ce document une bibliographie non exhaustive de livres que vous pouvez éventuellement consulter.

Ce document contient de nombreuses coquilles, merci de les signaler à l'auteur qui les corrigera au fur et à mesure.

Chapitre 1

Représention des nombres en machines et arithmétiques des ordinateurs

Ce chapitre présente le système de représentation digital des nombres utilisés sur la plupart des ordinateurs et machines à calculer actuels. Ce système repose sur le standard IEEE 754 virgule flottante. L'acronyme IEEE signifie Institution of Electrical and Electronics Engineers. Ce standard a mis de nombreuses années à émerger et comporte de nombreuses subtilités qui sont maintenant mises en oeuvre directement dans le "hardware" des micropresseurs, encore que certains points soient encore traités par des logiciels ("software"). Pour une vision approfondie, consulter le petite livre très agréable et pleins d'exemples "Numerical computing with IEEE Floating Point Arithmetic", Michael L. Overton, SIAM, 2001. Dans ce chapitre, nous allons traiter successivement :

— la représentation des nombres en machine ;
— quelques particularités des calculs en flottants illustrées par des exemples.

1.1 La représentation des nombres en machine

Dans tous les ordinateurs actuels le système de numération utilisé est le système à base 2. Il y a eu des machines utilisant d'autres systèmes de numération, mais les mémoires élémentaires utilisées, appelés ***bits***, étant des dispositifs électroniques à deux états, il est peu probable que cela change. Il faut donc un peu compter en base 2 où par exemple $37\,(\equiv 3*10+7) = 2^5+2^2+1$ (penser à $2^4 = 16$) s'écrit 100101 (en base 2). D'une manière générale on écrira un entier dans une base b (entier ≥ 2) particulière sous la forme d'une suite de chiffres $(a_n a_{n-1}...a_1 a_0)_b \equiv a_n b^n + a_{n-1}b^{n-1} + ...a_1 b + a_0$ où les chiffres a_i sont des entiers compris entre 0 et $b-1$ (bien sûr $a_n > 0$) et les chiffres sont rangées de DROITE à GAUCHE (le chiffre des unités a_0 est à droite)

Pour raccourcir l'écriture en base 2 qui est très longue, on utilise le codage en base 16, hexadécimal, car il est facile de passer des chiffres hexadécimaux $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f$ au binaire (un chiffre hexadécimal occupe 4bits) avec la table :

0 : 0000	4 : 0100	8 : 1000	c : 1100
1 : 0001	5 : 0101	9 : 1001	d : 1101
2 : 0010	6 : 0110	a : 1010	e : 1110
3 : 0011	7 : 0111	b : 1011	f : 1111

La conversion binaire → hexadécimal est très facile, il suffit de grouper les chiffres quatre à quatre en partant de la droite Par exemple `1 0001 0001 1001 1101` s'écrit `1119d` en hexadécimal (ce qui donne en décimal $13+9\times16+1\times16^2+1\times16^3+1\times16^4 = 70045$). Remarquez qu'on a complété par des zéros à gauche pour avoir 5 paquets de 4 chiffres.

La conversion hexadécimal → binaire est plus facile encore, il suffit de remplacer chaque chiffre hexadécimal par son écriture binaire.

Dans la mémoire des ordinateurs les bits sont rassemblés en groupes appelés mots et chaque mot a une adresse : on lit ou on écrit en bloc sur tous les bits du mots. Tous les ordinateurs actuels ont des mots de 8 bits appelés ***octets*** ou bytes en anglais et on peut manipuler des entiers sans signe (on dit ***entiers non signés***) qui occupent $8, 16, 32$ ou 64 bits (un, deux, quatre ou huit octets) et donc :

— un entier non signé sur 8 bits varie de 0 à $2^7+2^6+2^5+2^4+2^3+2^2+2+1=2^8-1=255$ ce qui fait 256 valeurs (penser à $2^2=4$, $2^4=16$, $16^2=256$ et aussi $2^9=512$, $2^{10}=1024$) ;
— un entier non signé sur 16 bits varie de 0 à $2^{15}+...+2+1=2^{16}-1=65535$;
— un entier non signé sur 32 bits varie de 0 à $2^{31}+...+2+1=2^{32}-1=4294967296$
— un entier non signé sur 64 bits varie de 0 à $2^{63}+...+2+1=2^{64}-1=1.8447\,10^{19}$ ce nombre est si grand que son approximation en flottants double précision ... voir plus loin ... ne peut pas être exacte, sa valeur exacte est 18446744073709551615.

On décrit maintenant le codage des entiers signés (qui peuvent être aussi bien positifs que négatifs, donc sont des entiers relatifs) et celui des flottants qui sont des réels représentés par une mantisse et un exposant. On précise cela plus loin.

1.1.1 Le codage des entiers relatifs

Il faut retenir essentiellement que le dernier bit (le plus à gauche) sert à coder le signe : le nombre est positif si ce bit est 0 *et négatif si ce bit est* 1. *De plus si ce nombre est négatif, pour obtenir sa valeur absolue il faut retrancher l'entier obtenu en base* 2 *avec les bits restants à* $2^{\text{nombre de bits}}$. En clair :

— pour un entier signé codé sur 8 bits $b_7b_6...b_1b_0=\begin{cases} b_62^6+b_52^5+...+b_12+b_0 \text{ si } b_7=0 \\ -(2^8-b_72^7-b_62^6-...-b_12-b_0) \text{ si } b_7=1 \end{cases}$ si bien que cet entier peut varier de $-2^7(=-(2^8-2^7))=-128$ à $2^7-1=127$ $((=2^6+2^5+...+2+1)$ (si $b_7=1$, la formule est aussi $-b_72^7+b_62^6+...+b_12+b_0$; *on parle de complément à* 2) ;

— pour un entier signé codé sur 16 bits $b_{15}b_{14}...b_1b_0=\begin{cases} b_{14}2^{14}+b_{13}2^{13}+...+b_12+b_0 \text{ si } b_{15}=0 \\ -(2^{16}-b_{15}2^{15}-b_{14}2^{14}-...-b_12-b_0) \text{ si } b_{15}=1 \end{cases}$ si bien que cet entier peut varier de $-2^{15}(=-(2^{16}-2^{15}))=-32768$ à $2^{15}-1=32767$ $((=2^6+2^5+...+2+1)$ (remarque analogue au premier cas) ;

— pour un entier signé codé sur 32 bits $b_{31}b_{30}...b_1b_0=\begin{cases} b_{30}2^{30}+b_{29}2^{29}+...+b_12+b_0 \text{ si } b_{31}=0 \\ -(2^{32}-b_{31}2^{31}-b_{30}2^{30}-...-b_12-b_0) \text{ si } b_{31}=1 \end{cases}$ si bien que cet entier peut varier de $-2^{31}(=-(2^{32}-2^{31}))=-2147483648$ à $2^{31}-1=2147483647$ $((=2^{30}+2^{29}+...+2+1)$ (remarque analogue au premier cas) ;

— pour un entier signé codé sur 64 bits $b_{63}b_{62}...b_1b_0=\begin{cases} b_{62}2^{62}+b_{61}2^{61}+...+b_12+b_0 \text{ si } b_{63}=0 \\ -(2^{64}-b_{63}2^{63}-b_{62}2^{62}-...-b_12-b_0) \text{ si } b_{63}=1 \end{cases}$ si bien que cet entier peut varier de $-2^{63}(=-(2^{64}-2^{63}))=-9223372036854775808$ à $2^{63}-1=9223372036854775807$ $((=2^{62}+2^{61}+...+2+1)$ (remarque analogue au premier cas).

1.1.2 Le codage des flottants

Rappelons le principe de l'écriture en base b d'un nombre réel x.

$$x=(a_na_{n-1}\dots a_1a_0,a_{-1}a_{-2}\dots)_b\equiv a_nb^n+a_{n-1}b^{n-1}+\dots a_1b+a_0+a_{-1}b^{-1}+a_{-2}b^{-2}+\dots$$

où les chiffres $(a_i)_{i\leq n}$ sont des entiers compris entre 0 et $b-1$ (bien sûr $a_n>0$).
Comme les ordinateurs sont des objets matériels, ils ne disposent que d'une mémoire finie donc quel que soit le système utilisé on ne pourra coder que des suites finies de chiffres donc nécessairement des nombres rationnels [1] La position de la virgule peut etre fixe. Mais on préfère l'écriture en virgule flottante, c'est à dire que l'on convient d'écrire le nombre sous la forme

$$x=(a_n,a_{n-1}\dots a_1a_0a_{-1}a_{-2}\dots a_{-m})_b\cdot b^n=(d_0,d_1d_2\dots d_p)_b\cdot b^n$$

Le nombre $(d_0,d_1d_2\dots d_p)_b$ s'appelle la *mantisse* et b^n l'exposant. Par définition la matisse commence par un chiffre non nul. La mantisse est donc dans l'intervalle $[1,b[$. Cette écriture permet d'emblée d'appréhender l'ordre de grandeur du nombre en question $x\approx a_n\cdot b^n$. Les "décimales" de la mantisse sont appelés les chiffres significatifs.

Précisons cela en base 10. Si x est un réel, x s'écrit donc $x=\pm M\times 10^E$ avec M réel appelée mantisse vérifiant $1\leq M<10$ et E entier relatif appelé exposant.

Ainsi l'approximation de 346.58 avec une mantisse à 4 chiffres est $3.466\,10^2$;

1. et encore pas tous, prenez par exemple 1/3 en base 10 et aussi 1/10 en base 2.

l'écriture de 0.0000456 avec une mantisse à 6 chiffres est $4.56000\,10^{-5}$;

l'approximation de $\pi = 3.141592653589793...$ avec une mantisse à 8 chiffres est $3.1415927\,10^{0}$.

Rappelons la notion essentielle d'*erreur relative*. Soit x un réel et $\tilde{x}$ une approximation de x. L'erreur relative est définie par

$$|\frac{\delta x}{x}| = |\frac{x - \tilde{x}}{x}|$$

et elle s'exprime en % (sans unité), contrairement à l'erreur absolue $|x - \tilde{x}|$ qui a la même unité que x (si x est en mètre par exemple, l'erreur absolue aussi). Si on fixe le nombre de chiffres de la mantisse, en faisant flotter la virgule en conséquence, et qu'on arrondit au plus proche, cela a l'avantage que *la précision relative est à peu près constante* ($\frac{5\,10^{-4}}{1\,\text{à}\,9}$ en base 10 pour une mantisse à 4 chiffres). Cependant l'erreur absolue peut être proportionnelle, en valeur absolue, au nombre qu'on "approxime".

Remarque. Si on codait les réels en virgule fixe, la précision absolue de la représentation serait fixe ; mais la précision relative dépendrait de l'ordre de grandeur. C'est pour garantir une précision relative indépendante de l'ordre de grandeur, c'est-à-dire pour travailler avec un nombre fixe de chiffres significatifs, que l'on préfère une représentation en virgule flottante. Par exemple en base 10, avec 4 chiffres après la virgule, on peut comparer dans la table suivante les deux représentations approchées (obtenues, pour simplifier, par troncature et non par arrondi) : on constate qu'en virgule fixe, plus les valeurs sont petites, moins on conserve de chiffres significatifs, *et plus l'erreur relative est importante !*. □

nombre exact	troncature virgule fixe	erreur relative	troncature virgule flottante	erreur relative
0.0000123456789	0.0000	100%	1.234×10^{-5}	4.5999e-04
0.000123456789	0.0001	19%	1.234×10^{-4}	4.5999e-04
0.00123456789	0.0012	2.8%	1.234×10^{-3}	4.5999e-04
0.0123456789	0.0123	0.0037	1.234×10^{-2}	4.5999e-04
0.123456789	0.1234	4.5999e-04	1.234×10^{-1}	4.5999e-04
1.23456789	1.2345	5.4991e-05	1.234×10^{0}	4.5999e-04
12.3456789	12.3456	6.3909e-06	1.234×10^{1}	4.5999e-04
123.456789	123.4567	7.2090e-07	1.234×10^{2}	4.5999e-04
1234.56789	1234.5678	7.2900e-08	1.234×10^{3}	4.5999e-04

TABLE 1.1 – Comparaison des erreurs relatives en format fixe et flottant.

Exercice. Quelle est l'erreur relative commise en approchant π par 3.14 ? *Réponse :* $\frac{\pi - 3.14}{\pi} \approx 5 \cdot 10^{-4} = 0.05\%$

La distance moyenne de la terre à la lune vaut 385 000 km, la distance exacte le 11 septembre 2014 était de 368 372 km, alors que le 11 septembre 2001 elle était de 380 155 km. En fait la distance terre lune oscille entre 363 100 et 405 700 km. Quelle est la variation relative de cette distance ? *Réponse :* $\frac{405700 - 363100}{385000} \approx 0.11 = 11\%$

Quelle est l'erreur relative commise en approchant la constante de gravité g par 10 sachant qu'elle vaut environ 9.8069 à Montpellier. *Réponse :* $\frac{10 - 9.8069}{9.8069} \approx 0.0197 \approx 2\%$

Quelle est l'approximation la plus précise parmi les suivantes $C = 300000$ km/s (valeur exacte dans le vide 299 792 km/s), durée de l'année= 365 jours (valeur moyenne 365.25) et taille de Teddy Riner=2.04 m (au demi-centimètre près) ?

Réponse : l'erreur relative sur l'année $= 0.25/365.25 = 0.00068$ est plus petite que l'erreur relative sur $C \approx 0.00069$ qui est beaucoup plus petite que l'erreur relative sur taille de Teddy $= 0.005/2.04 \approx 0.0025$.

Depuis 1985 environ [2], tous les ordinateurs utilisent ce qu'on appelle la norme de l'arithmétique flottante IEEE 754 dont on va décrire le codage le plus fréquent. Cette norme permet encore des différences entre marques de processeurs mais elles sont petites.

Avant de rentrer dans les détails on peut dire que :

— *le dernier bit sert encore à coder le signe* ;

— qu'un certain nombre de bits (8 en simple précision, 11 bits en double précision) servent à coder l'*exposant signe compris*, mais les *valeurs extrêmes de l'exposant ne sont pas utilisées pour coder des*

2. la situation avant cette date était très variable d'une marque à une autre

nombres "ordinaires" mais servent à coder des résultats d'opérations impossibles comme $\frac{0}{0}$, $0 \times \infty$ (NaN : "*not a number*") ou des dépassements de capacité par le haut en valeur absolue comme $\frac{1}{0}$ (*overflow* : $\pm\infty$) ou des dépassements de capacité vers le bas (*underflow* : flottants dits non normalisés) ;

— le nombre zéro peut être codé de deux façons 0+ lorsque le bit de signe est 0 et tous les autres bits égaux à 0, (resp. 0− lorsque le bit de signe est 1. et tous les autres bits égaux à 0

— lorsque l'exposant E est strictement compris entre ses bornes le nombre vaut $\underbrace{(1+f)}_{\text{mantisse}} 2^E$ où f qui vérifie $0 \leq f < 1$ est codé dans les bits restants (qui sont les premiers : 23 bits en simple précision, 52 bits en double précision).

Voici les détails.

En **simple précision** ces flottants sont codés en général sur 32 bits (4 octets de 8 bits) en une mantisse et un exposant : si ces bits sont $b_{31}, b_{30}, ..., b_0$ le premier code le signe, l'exposant est codé sur les 8 bits suivants, enfin la mantisse est codée sur 23 bits.

La valeur du nombre est définie comme suit :

— si $b_{30}2^7 + ... + b_{23} = 255$ mais $b_{22}...b_0 \neq 0$, la machine dit NaN (not a number : cas d'overflow) ;

— si $b_{30}2^7 + ... + b_{23} = 255$ et $b_{22}...b_0 = 0$, on convient de dire que le nombre est $+\infty$ si $b_{31} = 0$ et $-\infty$ si $b_{31} = 1$;

— si $\mathbf{1} \leq b_{30}2^7 + ... + b_{23} \leq \mathbf{254}$, le nombre vaut :

$$\left\{ \begin{array}{l} +\text{ si } b_{31} = 0 \\ -\text{ si } b_{31} = 1 \end{array} \right\} (1 + \frac{b_{22}}{2} + ... + \frac{b_0}{2^{23}}) 2^{b_{30}2^7 + ... + b_{23} - 127}$$

— si $b_{30}...b_{23} = 0$ et $b_{22}...b_0 = 0$, le nombre est zéro (on dit ± zéro selon que b_{31} est 0 ou 1) ;

— si $b_{30}...b_{23} = 0$ et $b_{22}...b_0 \neq 0$, le nombre est dit dénormalisé (underflow), et vaut

$$\left\{ \begin{array}{l} +\text{ si } b_{31} = 0 \\ -\text{ si } b_{31} = 1 \end{array} \right\} (\frac{b_{22}}{2} + ... + \frac{b_0}{2^{23}}) 2^{-126}$$

Résumons cela par un tableau.

signe	exposant	mantisse
b_{31}	$b_{30} \dots b_{23}$	$b_{22} \dots b_0$

si l'exposant $b_{30} \dots b_{23}$ est	la valeur numérique représentée est
$(00000000)_2 = (0)_{10}$	$\pm(0.b_{22} \dots b_0)_2 \times 2^{-126}$
$(00000001)_2 = (1)_{10}$	$\pm(1.b_{22} \dots b_0)_2 \times 2^{-126}$
$(00000010)_2 = (2)_{10}$	$\pm(1.b_{22} \dots b_0)_2 \times 2^{-125}$
$(00000011)_2 = (3)_{10}$	$\pm(1.b_{22} \dots b_0)_2 \times 2^{-124}$
$\vdots$	$\vdots$
$(01111111)_2 = (127)_{10}$	$\pm(1.b_{22} \dots b_0)_2 \times 2^0$
$(10000000)_2 = (128)_{10}$	$\pm(1.b_{22} \dots b_0)_2 \times 2^1$
$\vdots$	$\vdots$
$(11111100)_2 = (252)_{10}$	$\pm(1.b_{22} \dots b_0)_2 \times 2^{125}$
$(11111101)_2 = (253)_{10}$	$\pm(1.b_{22} \dots b_0)_2 \times 2^{126}$
$(11111110)_2 = (254)_{10}$	$\pm(1.b_{22} \dots b_0)_2 \times 2^{127}$
$(11111111)_2 = (255)_{10}$	$\pm\infty$ si $b_{22} = \dots = b_0 = 0$, NaN sinon

TABLE 1.2 – IEEE simple précision

Ainsi en simple précision, le plus petit nombre machine non négligeable devant 1, pour l'addition, est $\epsilon = \frac{1}{2^{23}} \simeq 1.2\,10^{-7}$, le plus petit nombre machine en valeur absolue est $1 * 2^{1-127} \simeq 1.2\,10^{-38}$ (si on ne considère pas les nombres dénormalisés) et le plus grand $\frac{1-2^{-24}}{\frac{1}{2}} 2^{254-127} \simeq 3.4\,10^{38}$.

En **double précision**, les flottants sont codés sur 64 bits (8 octets) et les bits étant $b_{63}b_{62}...b_0$ le principe est le même :

signe	exposant	mantisse
b_{63}	$b_{62}\dots b_{52}$	$b_{51}\dots b_0$

si l'exposant $b_{62}\dots b_{52}$ est	la valeur numérique représentée est
$(00000000000)_2 = (0)_{10}$	$\pm(0.b_{51}\dots b_0)_2 \times 2^{-1022}$
$(00000000001)_2 = (1)_{10}$	$\pm(1.b_{51}\dots b_0)_2 \times 2^{-1022}$
$(00000000010)_2 = (2)_{10}$	$\pm(1.b_{51}\dots b_0)_2 \times 2^{-1021}$
$(00000000011)_2 = (3)_{10}$	$\pm(1.b_{51}\dots b_0)_2 \times 2^{-1020}$
$\vdots$	$\vdots$
$(01111111111)_2 = (1023)_{10}$	$\pm(1.b_{51}\dots b_0)_2 \times 2^{0}$
$(10000000000)_2 = (1024)_{10}$	$\pm(1.b_{51}\dots b_0)_2 \times 2^{1}$
$\vdots$	$\vdots$
$(11111111100)_2 = (2044)_{10}$	$\pm(1.b_{51}\dots b_0)_2 \times 2^{1021}$
$(11111111101)_2 = (2045)_{10}$	$\pm(1.b_{51}\dots b_0)_2 \times 2^{1022}$
$(11111111110)_2 = (2046)_{10}$	$\pm(1.b_{51}\dots b_0)_2 \times 2^{1023}$
$(11111111111)_2 = (2047)_{10}$	$\pm\infty$ si $b_{51} = \dots = b_0 = 0$, NaN sinon

TABLE 1.3 – IEEE double précision

— si $b_{62}2^{10} + ... + b_{52} = 2047$ mais $b_{51}...b_0 \neq 0$, la machine dit NaN (not a number) ;
— si $b_{62}2^{10} + ... + b_{52} = 2047$ et $b_{51}...b_0 = 0$, on convient de dire que le nombre est $+\infty$ si $b_{63} = 0$ et $-\infty$ si $b_{63} = 1$;
— si $\mathbf{1} \leq b_{62}2^{10} + ... + b_{52} \leq \mathbf{2046}$, le nombre vaut :

$$\left\{ \begin{array}{l} +\text{si } b_{63} = 0 \\ -\text{si } b_{63} = 1 \end{array} \right\} (1 + \frac{b_{51}}{2} + ... + \frac{b_0}{2^{52}})2^{b_{62}2^{10}+...+b_{52}-1023}$$

— si $b_{62}...b_{52} = 0$ et $b_{51}...b_0 = 0$, le nombre est zéro (on dit $\pm$ zéro selon que b_{63} est 0 ou 1) ;
— si $b_{62}...b_{52} = 0$ et $b_{51}...b_0 \neq 0$, le nombre est dit dénormalisé, et vaut

$$\left\{ \begin{array}{l} +\text{si } b_{63} = 0 \\ -\text{si } b_{63} = 1 \end{array} \right\} (\frac{b_{51}}{2} + ... + \frac{b_0}{2^{52}})2^{-1022}$$

Ainsi en double précision, le plus petit nombre machine non négligeable devant 1, pour l'addition, est $\epsilon = \frac{1}{2^{52}} \simeq 2.2\,10^{-16}$, ce nombre est appelé la **précision machine** et est noté `eps` en MATLAB. Le plus petit nombre machine en valeur absolue est $2^{1-1023} \simeq 2.25\,10^{-308}$ (si on ne considère pas les nombres dénormalisés) et le plus grand $2(1 - \frac{1}{2^{53}})2^{1023} \simeq 1.8\,10^{308}$.

La table 1.4 compare la portée des flottants en simple et double précision, ainsi que la précision machine.

Précision	Exposant min	Exposant max	flottant min	flottant max	eps
simple	-126	127	$2^{-126} \approx 1.2 \times 10^{-38}$	$2^{128} \approx 3.4 \times 10^{38}$	$2^{-23} \approx 1.2\,10^{-7}$
double	-1022	1023	$2^{-1022} \approx 2.2 \times 10^{-308}$	$2^{1024} \approx 1.8 \times 10^{308}$	$2^{-52} \approx 2.2\,10^{-16}$

TABLE 1.4 – Bornes des flottants simple et double précision

Il faut bien insister sur le fait qu'en double (respectivement simple) précision $1 + \frac{1}{2^{52}} > 1$ (respectivement $1 + \frac{1}{2^{23}} > 1$) alors que $1 + \frac{1}{2^{53}}$ (respectivement $1 + \frac{1}{2^{24}}$) vaut 1 en machine.
Mais attention si $1 +$ `eps/2` fait 1 en machine $1 -$ `eps/2` est différent de 1 en machine.

À noter aussi qu'il existe aussi des précisions encore plus grande (précision étendue sur 80 bits, quadruple précision sur 128 bits en FORTRAN 2008).

Exercice. tester le code Matlab `function Controle_du_code_des_flottants` disponible sur l'ENT.

On peut également tester le code Matlab `floatgui.m`, extrait du livre de C.B. Moler (disponible sur internet, Cf bibliographie) pour observer la distribution des flottants dans des cas très simples de petites mantisses et petits exposants. [3]

3. Le codage en virgule flottante correspond à une graduation variable selon la zone où se trouve le réel x à représenter : lorsque x est près de zéro, le maillage est très fin, tandis que lorsque x est grand le maillage est très grossier. Cependant en prenant une échelle logarithmique, le maillage devient uniforme.

1.2 Quelques particularités des calculs en flottants illustrées par des exemples

On convient de noter $fl(x)$ le codage en flottant IEEE du réel x. On vient de voir que le codage des flottants assure une précision relative à peu près constante $\frac{|x-fl(x)|}{|x|} \leq \texttt{eps}/2$ pour $x \neq 0$.
Notant `eps` la *précision machine* qui est le plus petit nombre > 0 tel que $fl(1+\texttt{eps}) > 1$, alors on a :

$$fl(1+\texttt{eps}) \neq 1,\ fl(1+\tfrac{\texttt{eps}}{2}) = 1,\ fl(1-\tfrac{\texttt{eps}}{2}) \neq 1,\ fl(1-\tfrac{\texttt{eps}}{2}-\tfrac{\texttt{eps}}{4}) = fl(1-\texttt{eps})$$

Il en résulte par exemple que $fl((1+\frac{\texttt{eps}}{2})+\frac{\texttt{eps}}{2})$ n'est pas égal à $fl((1+(\frac{\texttt{eps}}{2}+\frac{\texttt{eps}}{2}))$ si bien que par exemple, l'*ensemble des flottants ne forme pas un corps* : il n'y a pas toujours associativité des opérations bien sûr à cause des erreurs d'arrondi.

Voici quelques exemples de calculs douteux, ainsi que des remèdes simples quand c'est possible. Les calculs sont effectués en Matlab (où les réels sont codés sur 64 bits).

```
>> (0.1+0.2)-0.3

ans =

     5.5511e-17

>> 0.1 +(0.2-0.3)

ans =

   2.7756e-17
```

il n'y a pas associativité et les résultats ne sont qu'approchés (à la précision machine.)

```
>> (1234567890123456.0 + 0.1) + 0.6

ans =

     1.234567890123456e+15
```

Le dernier chiffre du résultat est mal arrondi. Alors que si on regroupe les deux derniers termes avant d'additionner on trouve bien

```
>> 1234567890123456.0 + (0.1 + 0.6)

ans =

     1.234567890123457e+15
```

L'arrondi est correct.

```
>> 12345678901234568000-12345678901234567000
ans =

     0
```

Les nombres entiers sont trop grands et sont convertis automatiquement en flottants et il y a perte des derniers chiffres significatifs. Ici il n'y pas de remède, seul le calcul à la main est fiable et donne 1000 (au lieu de 0).
Le calcul suivant conduit à un dépassement de capacité.

```
>> 10^(309)/(9.99)^(309)
ans =

   NaN
```

alors que si on effectue d'abord le rapport avant d'élever à la puissance, il n'y pas de problème. On travaille avec des nombres proches de 1 et il n'y a pas de perte de précision.

```
>> (10/9.99)^(309)
ans =

   1.362272965816352
```

Voici une *règle d'hygiène du calcul* très simple : Dans un calcul essayer de **grouper entre eux les nombres qui ont le même ordre de grandeur**
car un grand nombre + un tout petit nombre = Warning ! le petit nombre est considéré nul.
Soustraire deux nombres extrêmement proches = Risque de perte de précision par élimination catastrophique, voir exemples plus bas.

Dans le même ordre d'idée, on veillera à ne jamais arrêter une boucle avec un test d'égalité à zéro : utiliser `WHILE ABS(X) > EPS` plutôt que `WHILE X <> 0`.

1.2.1 Calcul d'une somme évidente

Un autre exemple concerne le calcul de la somme $s_n = 1 + \sum_{k=1}^{n} \frac{1}{k(k+1)}$ qui puisque $\frac{1}{k} - \frac{1}{k+1} = \frac{1}{k(k+1)}$ vaut $s_n = 2 - \frac{1}{n+1}$. Le code MATLAB suivant montre que pour $n = 1000$, le calcul de s_n dans le sens des indices k croissants ne donne pas le même résultat que dans l'autre sens. *Il n'y a pas associativité des sommes.* La sommation dans le sens des indices décroissants donne ici le résultat exact alors que la somme dans le sens des indices croissants est légèrement fausse. Cela vient du fait que lorsqu'on somme suivant les indices décroissants, les termes successifs sont de tailles comparables au cours du calcul alors que lorsqu'on somme suivant les indices croissants on ajoute des nombres de plus en plus petits à un nombre de l'ordre de 2. Les derniers termes sont de l'ordre de la précision machine en simple précision et sont donc négligés. Evidemment, en double précision, la différence est beaucoup plus faible.

```
function [sc,sd,se] = somme_evidente(n)
%SOMME_EVIDENTE
% On calcule ici 1+somme(1/k/(k+1)) où k=1:n
% dans le sens des k croissants (sc) puis décroissants (sd)
% et en simple précision
% Cette somme vaut 2-1/(n+1) (se) car 1/k-1/(k+1)=1/k/(k+1)
% Les résultats sont affichés en format long et hexadécimal
% (faire par exemple somme_evidente(1000))
se=single(2)-1/(n+1);
sc=single(1); for k=1:n, sc=sc+1/k/(k+1); end;
sd=single(0); for k=n:-1:1, sd=sd+1/k/(k+1); end; sd=sd+1;
disp('Affichage décimal'), format long, disp([sc,sd,se])
disp('Affichage HEXADÉCIMAL'), format hex, disp([sc,sd,se])
```

Le résultat affiché est :

```
Affichage décimal
1.9990020 1.9990010 1.9990010
```

```
Affichage hexadécimal
3fffdf4c 3fffdf44 3fffdf44
```

1.2.2 Calcul de π par la méthode d'Archimède

Dans ce cas précédent seules les dernières décimales sont touchées et cela n'est pas grave, mais voici un exemple très connu de phénomène qu'on peut qualifié de *"cancellation"* où c'est beaucoup plus grave. Il s'agit du calcul de π par la méthode d'Archimède. Le demi-périmètre d'un polygone régulier de 2^k cotés inscrit dans un disque de rayon 1 est $y_k = \underbrace{2^k}_{1/2\,2^k.2} \sin(\underbrace{\frac{\pi}{2^k}}_{\frac{1}{2}\frac{2\pi}{.2^k}})$ et bien sûr $y_k \to_{k\to\infty} \pi$. On a $y_1 = 2$ et les formules de trigonométrie $1 - 2\sin^2(\frac{\alpha}{2}) = \cos(\alpha)$ et $|\cos(\alpha)| = \sqrt{1-\sin^2(\alpha)}$, donnent $y_{k+1} = 2^{k+1}\sin(\frac{\pi}{2^{k+1}}) = 2^{k+1}\sqrt{\frac{1}{2}(1-\cos(\frac{\pi}{2^k}))} = 2^{k+1}\sqrt{\frac{1}{2}(1-\sqrt{1-\sin^2(\frac{\pi}{2^k})})}$ et donc la suite $\{y_k\}_{k\geq 1}$ vérifie :

$$y_1 = 2,\ y_{k+1} = 2^{k+1}\sqrt{\frac{1}{2}(1-\sqrt{1-(\frac{y_k}{2^k})^2})}, \tag{1.1}$$

et aussi, puisque, procédé bien connu de multiplication et division par la quantité "conjuguée", $1-\sqrt{1-(\frac{y_k}{2^k})^2} = \frac{(1-\sqrt{1-(\frac{y_k}{2^k})^2})(1+\sqrt{1-(\frac{y_k}{2^k})^2})}{1+\sqrt{1-(\frac{y_k}{2^k})^2}} = \frac{(\frac{y_k}{2^k})^2}{1+\sqrt{1-(\frac{y_k}{2^k})^2}}$:

$$y_1 = 2,\ y_{k+1} = y_k\sqrt{\frac{2}{1+\sqrt{1-(\frac{y_k}{2^k})^2}}}. \tag{1.2}$$

Voici un code MATLAB où les deux formules sont utilisées.

```
function calcul_de_pi_par_Archimede
%CALCUL_DE_PI_PAR_ARCHIMEDE
% La suite y_k=2^k sin(pi/2^k) converge évidemment vers pi si k tend vers
% l'infini ... c'est le demi périmètre d'un polygone de 2^k cotés inscrit
% dans un cercle de rayon 1 ...
% Au moyen des formules de trigonométrie, cos(\alpha)=1-2 sin(\alpha/2)^2,
% abs(cos(\alpha))=sqrt(1-sin(\alpha)^2), on peut démontrer que cette
% suite peut être calculée par les formules de récurrence
% y_1=2; y_(k+1)=2^(k+1) sqrt((1-sqrt(1-(y_k/2^k)^2))/2)
% ou encore, ce qui est équivalent si les calculs sont exacts:
% y_1=2; y_(k+1)=y_k/sqrt(2/(1+sqrt(1-(y_k/2^k)^2)))
% Et bien à cause de l'accumulation des erreurs d'arrondi les formules
% ne donnent pas les mêmes résultats : la première est très mauvaise.
clear, close all
n=34; ya(1)=2; z=2;
for k=1:n, ya(k+1)=2*z*sqrt((1-sqrt(1-(ya(k)/z)^2))/2); z=2*z; end
yb(1)=2; z=2;
for k=1:n, yb(k+1)=yb(k)*sqrt(2/(1+sqrt(1-(ya(k)/z)^2))); z=2*z; end
format long e;
[ya',yb'], pi
plot(1:35,ya,1:35,yb)
legend('formule instable','formule stable','Location','NorthWest')
```

L'affichage obtenu, figure non comprise, est :

```
2.000000000000000e+00 2.000000000000000e+00
2.828427124746190e+00 2.828427124746190e+00
3.061467458920719e+00 3.061467458920719e+00
3.121445152258053e+00 3.121445152258053e+00
3.136548490545941e+00 3.136548490545940e+00
```

```
3.140331156954739e+00 3.140331156954753e+00
3.141277250932757e+00 3.141277250932773e+00
3.141513801144145e+00 3.141513801144301e+00
3.141572940367883e+00 3.141572940367091e+00
3.141587725279961e+00 3.141587725277160e+00
3.141591421504635e+00 3.141591421511200e+00
3.141592345611077e+00 3.141592345570118e+00
3.141592576545004e+00 3.141592576584872e+00
3.141592633463248e+00 3.141592634338563e+00
3.141592654807589e+00 3.141592648776985e+00
3.141592645321215e+00 3.141592652386591e+00
3.141592607375720e+00 3.141592653288992e+00
3.141592910939673e+00 3.141592653514593e+00
3.141594125195191e+00 3.141592653570993e+00
3.141596553704820e+00 3.141592653585093e+00
3.141596553704820e+00 3.141592653588618e+00
3.141674265021758e+00 3.141592653589499e+00
3.141829681889202e+00 3.141592653589719e+00
3.142451272494134e+00 3.141592653589774e+00
3.142451272494134e+00 3.141592653589788e+00
3.162277660168380e+00 3.141592653589792e+00
3.162277660168380e+00 3.141592653589793e+00
3.464101615137754e+00 3.141592653589793e+00
4.000000000000000e+00 3.141592653589793e+00
0                     3.141592653589793e+00
0                     3.141592653589793e+00
0                     3.141592653589793e+00
0                     3.141592653589793e+00
0                     3.141592653589793e+00
0                     3.141592653589793e+00
ans =
3.141592653589793e+00
```

La première formule parvient difficilement à l'approximation `3.141596553704820e+00` de $\pi \approx 3.141592653589793$ et finit par donner 0(!!!), alors que la seconde formule donne la bonne approximation. Ceci est tout à fait normal puisque $\{y_k\}_{k\geq 1}$, qui devrait converger vers π, reste borné et donc $\frac{y_k}{2^k} \to_{k\to\infty} 0$, si bien que dans la première formule $fl(\sqrt{1-(\frac{y_k}{2^k})^2}) \to_{k\to\infty} 1$ donc $fl(1-\sqrt{1-(\frac{y_k}{2^k})^2})$ finit par s'annuler (le 2^{k+1} ne suffit pas pour rétablir la bonne valeur). Par contre dans la seconde formule $fl(\sqrt{\frac{2}{1+\sqrt{1-(\frac{y_k}{2^k})^2}}}) \to_{k\to\infty} 1$ et y_k finit par se stabiliser à une bonne valeur. Tout s'explique par le fait dans la première formule, la valeur de $(\frac{y_k}{2^k})^2$ est, à partir d'un certain rang non significative devant 1 (comme $\frac{\texttt{eps}}{8}$ devant 1) et donc dans la soustraction $1-\sqrt{1-(\frac{y_k}{2^k})^2}$ il y a "cancellation" des décimales de 1 et $\sqrt{1-(\frac{y_k}{2^k})^2}$.

1.2.3 Un autre exemple d'élimination catastrophique de décimales.

Soit la fonction élémentaire $f(x) = \frac{(1+x)-1}{x}$ qui se simplifie algébriquement en $f(x) = 1$. Nous allons étudier le comportement de la première expression lorsque x est très petit. Voici un petit code MATLAB qui trace la fonction pour x est très petit.

```
function cancel(digits)
clf;
d=(1:digits);
deltax=10.^(-d);
cancel=((1+deltax) -1)./deltax;
```

```
plot(deltax,cancel);
xlim([10.^(-digits) 10.^(-digits+3)]);
pause;
disp('frappez une touche pour continuer');
semilogx(deltax,cancel);
err_rel= abs(cancel -1);
pause;
disp('frappez une touche pour continuer');
semilogx(deltax,err_rel);
end
```

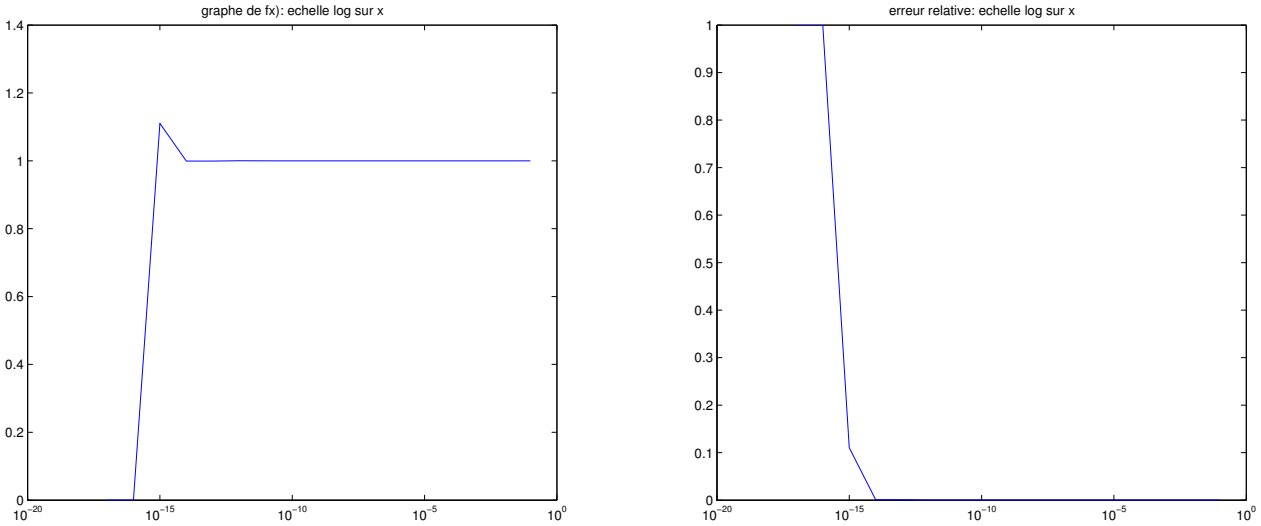


FIGURE 1.1 – gauche : graphe de $f(x) = \frac{(1+x)-1}{x}$ droite : l'erreur $|f(x) - 1|$, échelle log sur x.

Que s'est-il passé ? En dessous du seuil de précision de la machine, $1 + x = 1$ et donc $f(x) = 0$ au lieu de 1, on commet une erreur relative de 100% ! Pour x s'approchant du seuil de précision, les résultats fluctuent notablement autour de la valeur exacte. Evidemment pour $x > 10^{-15}$, le résultat est correct.

1.2.4 ($\star$) Un dernier exemple : un calcul rapide d'intégrales

Il s'agit de calculer la suite d'intégrales $E_k = \int_0^1 \underbrace{x^k}_{v} \underbrace{e^{x-1}dx}_{du}$, quantité positive puisque la fonction intégrée est positive. Il est clair que (formule de récurrence obtenue par une intégration par parties) :

$$E_0 = e^{x-1}|_{x=0}^{x=1} = 1 - \frac{1}{e},\ E_k = x^k e^{x-1}|_{x=0}^{x=1} - k\int_0^1 x^{k-1}e^{x-1}dx = 1 - kE_{k-1} \text{ pour } k \geq 1 \tag{1.3}$$

On doit comprendre aussi que $E_k \to_{k\to\infty} 0$. Pour justifier cela on peut d'abord remarquer que $x^k e^{x-1} \to_{k\to\infty} 0$ quel que soit x vérifiant $0 \leq x < 1$. Alors dans le cours d'analyse de L2 on découpe $[0, 1]$ en deux morceaux, un premier $[0, 1 - \frac{1}{N}]$ avec $N > 1$ entier sur lequel $x^k e^{x-1}$ converge uniformément vers zéro et donc pour lequel $\int_0^{1-\frac{1}{N}} x^k e^{x-1}dx \to_{k\to\infty} 0$ et un second $[1 - \frac{1}{N}, 1]$ sur lequel $x^k e^{x-1}$est bornée, en valeur absolue, par 1, si bien que $\int_{1-\frac{1}{N}}^1 x^k e^{x-1}dx \leq \frac{1}{N}$, la longueur de l'intervalle. En choisissant d'abord N assez grand puis k assez grand on voit que E_k peut être rendu aussi petit qu'on veut donc converge vers 0 si k tend vers l'infini. (En L3, cours de théorie de la mesure, vous allez apprendre le théorème de convergence dominée de Lebesgue qui permet de prouver la convergence plus facilement : $x^k e^{x-1} \to_{k\to\infty} 0$ presque partout sur $[0, 1]$ et est borné par quelque chose dont l'intégrale existe conc $E_k \to_{k\to 0} 0$.

On peut maintenant revenir à notre petit problème de calcul numérique. Dans le code `MATLAB` suivant, on a fait le calcul de E_k $0 \leq k \leq 20$, avec un programme de quadrature (voir chapitre Quadrature), puis avec

la formule de récurrence dans le sens des k croissants et enfin on a fait l'approximation $E_{32} = 0$ et utilisé la formule de récurrence dans le sens des k décroissants.

```
function Exemple_d_integrale
%EXEMPLE_D_INTEGRALE
% E_k= Intégrale de 0 à 1 de x^k exp(x-1) pour k=0,1,...,20
% on peut penser à utiliser la formule de récurrence:
% E_0=1-1/e; E_k=1-k E_(k-1)
% Il se trouve que E_k tend vers zéro si k tend vers l'infini, alors
% la formule est mauvaise dans le sens des k croissants, mais elle est
% très bonne dans l'autre sens au point qu'on peut poser ''brutalement''
% E_32=0 et ...
format long e
tic, e(1,1)=1-1/exp(1); for k=2:21, e(k,1)=quad(@(x)f(x,k),0,1,1e-12); end, toc
tic, e(1,2)=1-1/exp(1); for k=2:21, e(k,2)=1-(k-1)*e(k-1,2); end, toc
tic, e(33,3)=0; for k=33:-1:2, e(k-1,3)=(1-e(k,3))/(k-1); end, toc
[e(1:21,1) e(1:21,2) e(1:21,3)]
function y=f(x,k)
y=x.^(k-1).*exp(x-1);
```

et le résultat affiché est :

```
Elapsed time is 0.286808 seconds.
Elapsed time is 0.000008 seconds.
Elapsed time is 0.000007 seconds.
ans =
6.321205588285577e-01  6.321205588285577e-01 6.321205588285577e-01
3.678794411714425e-01  3.678794411714423e-01 3.678794411714423e-01
2.642411176571156e-01  2.642411176571153e-01 2.642411176571153e-01
2.072766470286543e-01  2.072766470286540e-01 2.072766470286539e-01
1.708934118853855e-01  1.708934118853840e-01 1.708934118853843e-01
1.455329405730884e-01  1.455329405730801e-01 1.455329405730786e-01
1.268023565615501e-01  1.268023565615195e-01 1.268023565615284e-01
1.123835040695699e-01  1.123835040693635e-01 1.123835040693008e-01
1.009319674456464e-01  1.009319674450921e-01 1.009319674455933e-01
9.161229298979662e-02  9.161229299417073e-02 9.161229298966059e-02
8.387707010432331e-02  8.387707005829270e-02 8.387707010339417e-02
7.735222886293644e-02  7.735222935878028e-02 7.735222886266420e-02
7.177325364810797e-02  7.177324769463667e-02 7.177325364802957e-02
6.694770257575389e-02  6.694777996972334e-02 6.694770257561571e-02
6.273216394145026e-02  6.273108042387321e-02 6.273216394138015e-02
5.901754088001835e-02  5.903379364190187e-02 5.901754087929777e-02
5.571934593393736e-02  5.545930172957014e-02 5.571934593123560e-02
5.277111916989722e-02  5.719187059730757e-02 5.277111916899477e-02
5.011985495840781e-02 -2.945367075153627e-02 5.011985495809426e-02
4.772275579632686e-02  1.559619744279189e+00 4.772275579620910e-02
4.554488407640699e-02 -3.019239488558378e+01 4.554488407581805e-02
```

Il est clair que la meilleure façon de faire est la récurrence dans le sens des k décroissants à partir de $E_{32} = 0$: c'est plus précis et plus rapide. De plus la récurrence dans le sens des k croissants donnent des résultats aberrants.

La conclusion de ces exemples est qu'avant de programmer une formule en flottants il faut prendre la précaution de choisir si possible, la forme la moins sensible à la propagation des erreurs d'arrondis.

Chapitre 2

Résolution des équations scalaires non linéaires

Dans ce chapitre nous allons décrire successivement :
— quelques rappels d'analyse réelle ;
— la méthode des approximations successives
— la méthode de Newton
— les méthodes de la fausse position, de la sécante et l'algorithme de Dekker-Brent ;
— en complément facultatif, quelques notions sur le calcul de zéros multiples.

2.1 Quelques rappels d'analyse réelle.

Ces quelques précisions sont l'occasion d'insister sur les propriétés fondamentales de l'ensemble des nombres réels $\mathbb{R}$: convergence des suites monotones bornées, existence de la borne supérieure (respectivement inférieure) d'un ensemble de réels non vide et majoré (respectivement minoré), propriété des suites d'intervalles emboîtés et complétude.

2.1.1 Démonstration du théorème des valeurs intermédiaires

Le théorème des valeurs intermédiaires dit qu'étant donnée une fonction réelle h définie continue sur un intervalle réel fermé borné $I = [a, b]$ et un réel y compris entre $h(a)$ et $h(b)$ alors il existe au moins un réel $c \in I$ tel que [1] $h(c) = y$.

Nous allons donner une démonstration élémentaire de ce résultat mais utilisant une propriété assez profonde de $\mathbb{R}$: la ***propriété des intervalles emboîtés*** qui dit que toute intersection $\bigcap_{k=0}^{\infty} I^{(k)}$ d'intervalles fermés $I^{(k)} = [a^{(k)}, b^{(k)}]$ emboîtés ($I^{(k+1)} \subseteq I^{(k)}$) est non vide

2.1.1.1 démonstration constructive par dichotomie.

Par une simple translation, on se ramène au cas où $y = 0$. Comme précédemment, après avoir traité les cas triviaux $h(a) = 0$ ou $h(b) = 0$, on peut toujours se ramener aux cas où $h(a) < 0 < h(b)$.
On définit alors une suite d'intervalles emboîtés $I^{(k)} = [a^{(k)}, b^{(k)}]$, $k \geq 0$ au moyen de l'algorithme de dichotomie (on dit bisection en anglais) :

$$
\begin{array}{l}
[a^{(0)}, b^{(0)}] = [a, b] \text{ et pour } k = 0, 1, ... \\
\text{on calcule } c' = \frac{a^{(k)}+b^{(k)}}{2} \text{ et le nouveau intervalle est défini par :} \\
\left\{ \begin{array}{c} \text{si } h(c') = 0,\ a^{(k+1)} = b^{(k+1)} = c'; \\ \text{si } h(a^{(k)}) \cdot h(c') < 0,\ a^{(k+1)} = a^{(k)},\ b^{(k+1)} = c'; \\ \text{si } h(a^{(k)}) \cdot h(c') > 0,\ a^{(k+1)} = c',\ b^{(k+1)} = b^{(k)} \end{array} \right. .
\end{array}
$$

1. Il n'est pas inutile de préciser à ce propos que ce théorème est un cas particulier d'un théorème de topologie plus général qui dit que, étant donnés deux espaces topologiques E et F et une application continue h de E dans F, l'image $h(C)$ d'un ensemble connexe $C \subset E$ par h est un ensemble connexe.

On notera que les suites $\{a^{(k)}\}_{k\geq 0}$ et $\{b^{(k)}\}_{k\geq 0}$ sont telles que ou bien $a^{(k)} = b^{(k)} = c'$ avec $h(c') = 0$ à partir d'un certain k, ou bien sont telles que pour tout $k \geq 0$, on a $h(a^{(k)}) < 0 < h(b^{(k)})$. Il est clair que le théorème est démontré avec $c = c'$ tel que $h(c) = 0$ si $a^{(k)} = b^{(k)} = c'$ à partir d'un certain rang. Si non on remarque que $[a^{(k+1)}, b^{(k+1)}] \subset [a^{(k)}, b^{(k)}]$ et que $(b^{(k+1)} - a^{(k+1)}) = \frac{1}{2}(b^{(k)} - a^{(k)}) = \frac{1}{2^2}(b^{(k-1)} - a^{(k-1)}) = ... = \frac{1}{2^{k+1}}(b-a) \to_{k\to\infty} 0$. La suite d'intervalles $[a^{(k)}, b^{(k)}]$ est donc emboîtée et a une intersection non vide, forcément réduite à un point $c = \lim_{k\to\infty} a^{(k)} = \lim_{k\to\infty} b^{(k)}$. Mais alors la continuité de h et la propriété $h(a^{(k)}) < 0 < h(b^{(k)})$ montrent que $h(c) \leq 0 \leq h(c)$ et donc $h(c) = 0$.

Remarque. Si on code cet algorithme sur une machine, pour que l'algorithme termine en temps fini, il faut remplacer le test d'arrêt si $h(c') = 0$, par une condition du style si $|h(c')| < TOL$ ou encore si $|a^{(k)} - b^{(k)}| < TOL$ où TOL est une tolérance fixée par l'utilisateur. □

Remarque. L'algorithme de dichotomie converge aussi vite que la suite géométrique de raison 1/2 :

$$a^{(k)} \leq c \leq b^{(k)} \text{ et } (b^{(k)} - a^{(k)}) = O(1/2^k).$$

En gros, à chaque itération, on gagne une "décimale" binaire. On dit que la (vitesse de) convergence est *linéaire* ou géométrique. □

2.1.2 Le théorème du point fixe de Banach et la méthode des approximations successives

Commençons par remarquer que si $f(x) = 0$ alors $x - f(x) = x$ et aussi $x - \lambda f(x) = x$ pour λ un paramètre réel. Le zéro d'une fonction $f(x)$ peut donc se voir comme *le point fixe* d'une autre fonction, à savoir la fonction $g(x) = x - \lambda f(x)$ au sens où $f(x) = 0$ équivaut à $g(x) = x$. L'intérêt de cette remarque est qu'il y a un algorithme itératif très simple pour trouver les points fixes, le schéma des approximations successives. C'est l'objet du théorème qui suit.

Théorème 1 *Soit une fonction g continue sur une partie I* fermée *de $\mathbb{R}$ ayant les propriétés :*

$$I \text{ est stable par } g : \quad \forall x \in I, \quad g(x) \in I \tag{2.1}$$

La fonction g est contractante : Il existe une constante L avec $0 \leq L < 1$ telle que

$$\forall x, y \in I, \quad |g(x) - g(y)| \leq L|x - y| \tag{2.2}$$

alors l'équation $x = g(x)$ a une racine ξ et une seule dans I, cette racine étant limite de la suite $\{x^{(k)}\}_{k\geq 0}$ définie par :

$$x^{(k+1)} = g(x^{(k)}),\ k \geq 0, \quad x^{(0)} \in I \text{donné} \tag{2.3}$$

et on a la majoration d'erreur :

$$|x^{(k)} - \xi| \leq L^k\, |x^{(0)} - \xi| \tag{2.4}$$

Preuve. Cette démonstration utilise la propriété de $\mathbb{R}$ d'être complet qui dit que toute suite de Cauchy de réels est convergente ; une suite $\{r^{(k)}\}_{k\geq 0}$ est dite de Cauchy si quel que le nombre $\epsilon > 0$ donné (par exemple $\epsilon = 10^{-p}$ où $p \geq 1$), on peut trouver un entier $N(\epsilon) \geq 0$ tel que $m \geq N(\epsilon)$ et $n \geq N(\epsilon)$ impliquent $|r^{(m)} - r^{(n)}| \leq \epsilon$ (autrement dit quel que soit le rang p on peut trouver un entier $N(10^{-p})$ tel que les $p-1$ premières décimales de tous les nombres $r^{(k)}$ avec $k \geq N(10^{-p})$ sont les mêmes.)

La partie I étant stable par g, par récurrence sur k on a $x^{(k)} \in I$. Montrons que la suite $(x^{(k)})$ est de Cauchy. La fonction g étant lipschitzienne de rapport L, $|x^{(k+1)} - x^{(k)}| = |g(x^{(k)}) - g(x^{(k-1)})| \leq L \cdot |x^{(k)} - x^{(k-1)}|$ et donc en itérant $|x^{(k+1)} - x^{(k)}| \leq L^k |x^{(1)} - x^{(0)}|$. Cela montre aussi que $|x^{(k+p)} - x^{(k)}| \leq |x^{(k+p)} - x^{(k+p-1)}| + |x^{(k+p-1)} - x^{(k+p-2)}| + ... + |x^{(k+1)} - x^{(k)}| \leq (L^{k+p-1} + L^{k+p-1} + ... + L^k)|x^{(1)} - x^{(0)}| \leq \frac{L^k}{1-L}\, |x^{(1)} - x^{(0)}|$ en sommant la série géométrique de raison $L < 1$. On a donc $|x^{(k+p)} - x^{(k)}| \to_{k\to\infty} 0$ quel que soit l'entier $p \geq 0$ ce qui signifie bien que $\{x^{(k)}\}_{k\geq 0}$ est de Cauchy. Donc $\{x^{(k)}\}_{k\geq 0}$ converge vers un réel ξ appartenant forcément à I puisque $x^{(k)} \in I$ et que I est fermé. Maintenant, grace à la continuité

de g, en passant à la limite dans la relation $x^{(k+1)} = g(x^{(k)})$ on obtient $\xi = g(\xi)$. Enfin remarquant que $|x^{(k+1)} - \xi| = |g(x^{(k)}) - g(\xi)| \leq L\,|x^{(k)} - \xi|$, en itérant on déduit par récurrence $|x^{(k)} - \xi| \leq L^k\,|x^{(0)} - \xi|$.

Prouvons enfin l'unicité du point fixe ξ dans I : si η est un autre point fixe dans I, de $\xi = g(\xi)$ et $\eta = g(\eta)$ et de 2.2 on déduit $|\xi - \eta| = |g(\xi) - g(\eta)| \leq L|\xi - \eta| < |\xi - \eta|$ si $\xi \neq \eta$ ce qui est absurde et donc la seule possibilité est $\xi = \eta$ ■

.

Remarques. (i) La méthode consiste donc simplement à itérer une fonction. La vitesse de convergence est la même que celle de la suite géométrique de raison L vers 0 car

$$|x^{(k)} - \xi| = O(L^k)$$

La convergence est donc d'autant plus rapide que L est petit devant 1. Si $L \geq 1/2$ la méthode est moins performante que la simple dichotomie.

Si on passe au logarithme on voit que le nombre de décimales exactes augmente linéairement avec k : on dit que l'algorithme du point fixe converge *à vitesse linéaire.* Si par exemple $L = 10^{-d}$, $L^k = 10^{-k\,d}$, à chaque itération on gagne d décimales.

(ii) Lorsque la fonction g est C^1, on peut estimer précisément la vitesse de convergence :

$$\frac{(x^{(k+1)} - \xi)}{(x^{(k)} - \xi)} = \frac{g(x^{(k)}) - g(\xi)}{x^{(k)} - \xi} \to g'(\xi), \quad k \to +\infty$$

La convergence est d'autant plus rapide que $|g'(\xi)|$ est petit devant 1. Si $g'(\xi) = 0$, la convergence est très rapide, on a dans ce cas

$$(x^{(k+1)} - \xi) \ll (x^{(k)} - \xi), \quad k \to +\infty$$

On dit que la vitesse de convergence est *superlinéaire*, i.e. mieux que linéaire. On dit aussi qu'il y a *super-convergence.*

(iii) Lorsque g est dérivable, on peut donner une condition suffisante très simple pour qu'elle soit contractante à l'aide du *théorème des accroissements finis.* Si $\forall x \in I$ on a $|g'(x)| \leq M < 1$ alors $\forall x, y \in I, \quad |g(x) - g(y)| \leq M|x - y|$ donc g est contractante.

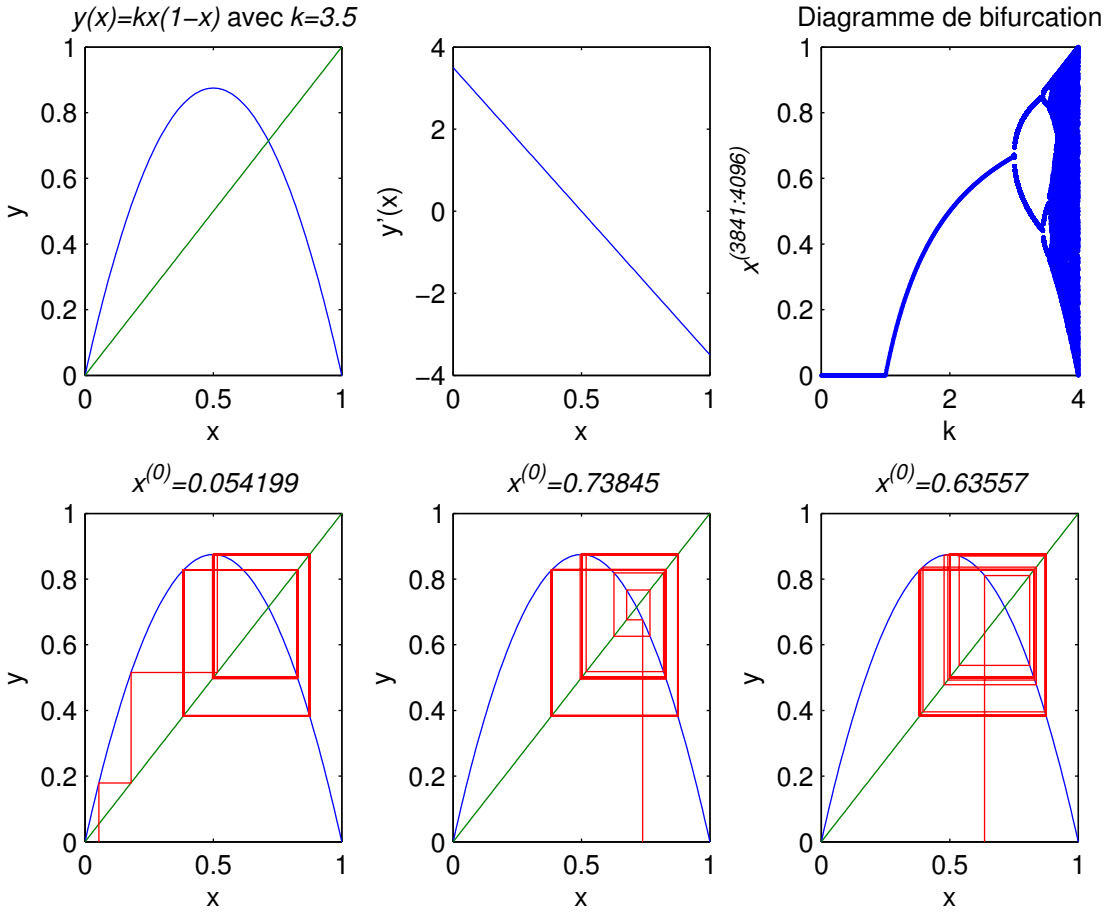
iv) La généralisation de la preuve au cas d'une fonction g d'un espace métrique complet, en particulier $\mathbb{R}^n$ ou un espace vectoriel normé complet (un espace de Banach) dans lui même est simple ; il suffit de remplacer partout les valeurs absolues de différences $|x - y|$ par $d(x, y)$ (ou $\|\mathbf{x} - \mathbf{y}\|$). Le théorème du point fixe de Banach a une portée considérable et permet en particulier d'établir un théorème d'existence et d'unicité de la majorité des problèmes d'équations différentielles ordinaires avec conditions initiales posées en pratique.

v) Dans notre cas d'une fonction g réelle de variable réelle il peut être plus élémentaire de distinguer les cas g croissante de pente majorée par $L < 1$ (cas de l'escalier) qui conduit à une suite $\{x^{(k)}\}_{k\geq 0}$ monotone (croissante majorée ou décroissante minorée selon que $x^{(0)} < \xi$ ou $x^{(0)} > \xi$) et g décroissante de pente minorée par $-L > -1$ (cas de l'escargot) qui conduit à une suite $\{x^{(k)}\}_{k\geq 0}$ constituée de deux suites adjacentes de même limite ξ. On est conduit ainsi à d'autres démonstrations utilisant la propriété de $\mathbb{R}$ qui dit que toute suite de réels monotone bornée est convergente.

vi) Il faut insister sur le fait qu'il ne suffit pas que la fonction g soit une application continue de I dans lui même pour que la méthode des approximations successives converge. Il faut aussi que g soit contractante ($0 \leq L < 1$). ($\star$) On peut évoquer ici, bien que les détails de ce sujet soient complètement en dehors du programme de L3 de mathématiques, les questions relatives au comportement chaotique de certaines suites ou systèmes dynamiques. L'exemple de Feigenbaum [2] concerne la fonction dite logistique $x \to g_k(x) = kx(1 - x)$. Lorsque le paramètre k est compris entre 0 et 4, la fonction g_k est une application de $[0, 1]$ dans lui même, mais selon les valeurs de k, le comportement de la suite $\{x^{(n)}\}_{n\geq 0}$, où $x^{(0)} \in [0, 1]$ est quelconque et $x^{(n+1)} = kx^{(n)}(1 - x^{(n)})$ si $n \geq 0$, change. Lorsque $0 < k \leq k_1 = 1$, $x = 0$ est l'unique point fixe et la suite $x^{(n)} \to_{n\to\infty} 0$. Si $1 < k$, il y a deux points fixes dans $[0, 1]$, $x = 0$ et $x = 1 - \frac{1}{k}$, mais pour $1 < k < k_2 = 3$, le point fixe $x = 0$ est répulsif (dérivée $g'(0) = k$ de module > 1) alors que le point fixe $x = 1 - \frac{1}{k}$ est attractif (dérivée $g'(1 - \frac{1}{k}) = 2 - k$ de module < 1) si bien que pour $0 < x^{(0)} < 1$, la suite $x^{(n)} \to_{n\to\infty} 1 - \frac{1}{k}$

2. La référence est M.J. Feigenbaum, 1979, The universal metric properties of nonlinear transformations : J. Stat. Phys., 21, 669 – 706.

(évidemment, si $x^{(0)} = 0$ ou 1, $x^{(n)} = 0$ pour $n \geq 1$). Pour $k = k_2 = 3$, il y a encore convergence (très lente) de $\{x^{(n)}\}_{n\geq 0}$ vers $1 - \frac{1}{3}$ (pour $0 < x^{(0)} < 1$) mais pour $k > k_2 = 3$, les deux points fixes sont répulsifs et la suite $\{x^{(n)}\}_{n\geq 0}$ qui reste bornée n'est plus convergente.



Revenant à notre sujet principal, pour trouver des racines d'équations $f(x) = 0$, la méthode la plus rapide lorsqu'elle converge est celle de Newton. C'est l'objet de ce qui suit.

2.2 La méthode de Newton (ou de Newton-Raphson)

La méthode de Newton pour résoudre $f(x) = 0$, où f est une fonction de $\mathbb{R}$ (ou d'un intervalle $I \subset \mathbb{R}$) dans $\mathbb{R}$ dérivable, consiste à définir, à partir d'une première approximation $x^{(0)}$ de la racine, une suite $\{x^{(k)}\}_{k\geq 0}$ telle que $x^{(k+1)}$ soit le zéro de "l'approximation au premier ordre de f en $x^{(k)}$" $x \to f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)})$, ou encore l'abscisse où la tangente au graphe de f passant par le point $(x^{(k)}, f(x^{(k)}))$ rencontre l'axe des x, si bien que :

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}. \tag{2.5}$$

Remarque. Remarquons que c'est une méthode itérative de la forme $x^{(k+1)} = g(x^{(k)}) = x^{(k)} - \lambda_k f(x^{(k)})$ où le paramètre $\lambda_k = \frac{1}{f'(x^{(k)})}$ dépend de $x^{(k)}$ est choisi de façon à ce que $g'(\xi) = 0$. Nous reviendrons sur ce point dans les remarques après l'énoncé du théorème. □

Nous allons donner deux démonstrations assez élémentaires de convergence de cette méthode avec différents domaines de validité.

2.2.1 Première démonstration. Cas où f est de classe C^2, strictement monotone et convexe ou concave et change de signe sur un intervalle $[a,b]$

Théorème 2 *Si est une fonction réelle f définie sur un intervalle $[a,b]$, de classe C^2 (deux fois continûment dérivable), change de signe sur $[a,b]$ (vérifie $f(a)f(b)<0$) a une dérivée $f'(x)\neq 0$ pour tout $x\in[a,b]$ (c'est à dire est strictement monotone sur $[a,b]$) et est soit convexe ($f''(x)\geq 0$ pour tout $x\in[a,b]$) soit concave ($f''(x)\leq 0$ pour tout $x\in[a,b]$) sur $[a,b]$, alors, étant donné $x^{(0)}\in[a,b]$ tel que $x^{(1)}=x^{(0)}-\frac{f(x^{(0)})}{f'(x^{(0)})}\in[a,b]$, la suite $\{x^{(k)}\}_{k\geq 0}$ engendrée par la méthode de Newton (formule (2.5)) est bien définie (tout entière contenue dans $[a,b]$), converge vers l'unique racine ξ de f sur $[a,b]$ et de plus $(x^{(k+1)}-\xi)\sim_{k\to\infty}\frac{1}{2}\frac{f''(\xi)}{f'(\xi)}(x^{(k)}-\xi)^2$ (on dit que la convergence est* quadratique*).*

Preuve. En changeant au besoin [3] le signe de x et (ou) f, on peut toujours supposer que $f(a)<0$, $f(b)>0$, $f'(x)>0$ pour $x\in[a,b]$ et $f''(x)\geq 0$ pour $x\in[a,b]$.

Bien évidemment la fonction continue strictement croissante f a un zéro unique $\xi\in]a,b[$ et on peut remarquer que $\{x\in[a,b],\ f(x)\geq 0\}\Leftrightarrow\{\xi\leq x\leq b\}$. Comme par hypothèse $x^{(1)}=x^{(0)}-\frac{f(x^{(0)})}{f'(x^{(0)})}\in[a,b]$, on peut écrire, en appliquant la formule de Taylor-Lagrange à l'ordre deux, $f(x^{(1)})=\underbrace{f(x^{(0)})+f'(x^{(0)})(-\frac{f(x^{(0)})}{f'(x^{(0)})})}_{0}$ $+\frac{1}{2}f''(\eta^{(0)})(x^{(1)}-x^{(0)})^2\geq 0$ où $\eta^{(0)}$ est entre $x^{(0)}$ et $x^{(1)}$, et donc $\xi\leq x^{(1)}\leq b$.
Montrons par récurrence que $\xi\leq x^{(k)}\leq b$ pour $k\geq 1$. Supposons donc cette propriété vraie au rang k ; $x^{(k+1)}=x^{(k)}-\frac{f(x^{(k)})}{f'(x^{(k)})}$ or $\xi\leq x^{(k)}\leq b$ donc $\frac{f(x^{(k)})}{f'(x^{(k)})}\geq 0$, ainsi on a bien $x^{(k+1)}\leq x^{(k)}\leq b$. D'autre part la convexité de f donne que le graphe de f est au dessus de la tangente au point $x^{(k)}$: $f(x)\geq f(x^{(k)})+f'(x^{(k)})(x-x^{(k)})$, et en prenant $x:=x^{(k+1)}$ on obtient : $f(x^{(k+1)})\geq f(x^{(k)})+f'(x^{(k)})(x^{(k+1)}-x^{(k)})=0$ d'où $x^{(k+1)}\geq\xi$. Ainsi la propriété est vraie pour tout $k\geq 1$. De plus on a montré que $x^{(k+1)}\leq x^{(k)}$.

Donc la suite $\{x^{(k)}\}_{k\geq 1}$ est monotone décroissante minorée par ξ, donc converge vers un nombre $\mu\in[\xi,b]$, mais comme la limite de (2.5) donne (car f et f' sont continues et $f'(x)>0$ sur $[a,b]$) $\mu=\mu-\frac{f(\mu)}{f'(\mu)}$ forcément $f(\mu)=0$ ce qui prouve $\lim_{k\to\infty}x^{(k)}=\xi$. Enfin la formule de Taylor-Lagrange à l'ordre deux donne aussi, avec $\zeta^{(k)}\in[\xi,x^{(k)}]$: $0=f(\xi)=f(x^{(k)})+f'(x^{(k)})(\xi-x^{(k)})+\frac{1}{2}f''(\zeta^{(k)})(\xi-x^{(k)})^2=f'(x^{(k)})(\underbrace{\frac{f(x^{(k)})}{f'(x^{(k)})}+\xi-x^{(k)}}_{\xi-x^{(k+1)}})+\frac{1}{2}\frac{f''(\zeta^{(k)})}{f'(x^{(k)})}(\xi-x^{(k)})^2)$. Comme f' ne s'annule pas par hypothèse,

$$\xi-x^{(k+1)}=-\frac{1}{2}\frac{f''(\zeta^{(k)})}{f'(x^{(k)})}(\xi-x^{(k)})^2)$$

et donc on a $\frac{x^{(k+1)}-\xi}{(x^{(k)}-\xi)^2}=\frac{1}{2}\frac{f''(\zeta^{(k)})}{f'(x^{(k)})}\to_{k\to\infty}\frac{1}{2}\frac{f''(\xi)}{f'(\xi)}$. cqfd.

Remarques.

i) On peut voir la méthode de Newton comme l'application de la méthode des approximations successives à la fonction $x\to g(x)=x-\frac{f(x)}{f'(x)}$ et le fait que la convergence est quadratique (donc superlinéaire, voir remarque alinéa précédent) vient de l'annulation de la constante de Lipschitz en $x=\xi$, puisque $g'(\xi)=1-\frac{f'(\xi)}{f'(\xi)}+\frac{f(\xi)f''(\xi)}{(f'(\xi))^2}=0$.

ii) La convergence quadratique signifie qu'en pratique le nombre de décimales exactes "double" à chaque itération. En effet, si $x^{(k)}$ a d décimales exactes, on a $|(x^{(k)}-\xi)|\sim 10^{-d}$, alors $|(x^{(k+1)}-\xi)|=O(x^{(k)}-\xi)^2$ sera de l'ordre de 10^{-2d} et donc aura en gros $2d$ décimales exactes.
Pour illustrer cette vitesse « exponentielle », rien ne vaut un exemple. On veut calculer de façon approchée $\xi=\sqrt{2}$ la racine positive de l'équation $f(x):=x^2-2=0$. L'algorithme de Newton devient dans ce cas $x^{(k+1)}=1/2(x^{(k)}+2/x^{(k)})$. [4]
Le code

3. Si $f''(x)\geq 0$, dans le cas où $f'(x)>0$ il n'y a rien à changer et dans le cas où $f'(x)<0$, il suffit de changer x en $-x$ pour obtenir sur l'intervalle $[-b,-a]$ une fonction $x\to f(-x)$ qui est bien convexe strictement croissante. Si $f''(x)\leq 0$, après avoir changé f en $-f$, on est ramené au cas précédent.

4. algorithme utilisé par les Babyloniens pour calculer les racines carrées.

```
x=1;
for i=1:6
  x=1/2*(x+2/x);
  disp(x);
end
```

produit le résultat

$$\begin{aligned} x &= \underline{1}.500000000000000 \\ x &= \underline{1.41}6666666666667 \\ x &= \underline{1.414215}686274510 \\ x &= \underline{1.41421356237}4690 \\ x &= \underline{1.414213562373095} \\ x &= \underline{1.414213562373095} \end{aligned}$$

la valeur $\sqrt{2} = 1.414213562373095\ldots$ exacte à la précision machine est obtenue en seulement 5 itérations ! Lorsqu'il converge l'algorithme de Newton a besoin de moins de 10 itérations en général.

iii) Cette démonstration ne se généralise pas directement au cas d'une application $\mathbf{f}$ de $\mathbb{R}^n$ dans $\mathbb{R}^n$ avec $n > 1$ et ne permet pas de traiter le cas d'un zéro ξ qui est un point d'inflexion, cas où par exemple $f(\xi) = 0$, $f'(\xi) > 0$, $f''(x) \begin{cases} < 0 \,\text{si}\, x > \xi \\ = 0 \,\text{si}\, x = \xi \\ > 0 \,\text{si}\, x < \xi \end{cases}$.

2.2.2 Deuxième démonstration : convergence locale au voisinage d'un zéro ξ où $f'(\xi) \neq 0$ qui englobe le cas d'un zéro qui est un point d'inflexion

Théorème 3 *Étant donnée une fonction f continûment différentiable de $\mathbb{R}$ (ou d'un intervalle I de $\mathbb{R}$) dans $\mathbb{R}$ ayant un zéro ξ en lequel $f'(\xi) \neq 0$, il existe $\delta > 0$ tel que si $|x^{(0)} - \xi| < \delta$ la suite $\{x^{(k)}\}_{k\geq 0}$ engendrée par la méthode de Newton (formule (2.5)) converge vers ξ.*

Preuve. Choisissons $0 < \alpha < 1$, montrons par récurrence que la suite $\{x^{(k)}\}_{k\geq 0}$ engendrée par la méthode de Newton vérifie $|x^{(k)} - \xi| \leq \alpha^k |x^{(0)} - \xi|$ ce qui prouvera bien que $x^{(k)} \to_{k\to\infty} \xi$ puisque $\alpha^k \to_{k\to\infty} 0$ pour $0 < \alpha < 1$. Cette inégalité est vraie pour $k = 0$ et si elle est vraie jusqu'au rang k ($|x^{(k)} - \xi| \leq \alpha^k |x^{(0)} - \xi|$), on peut écrire :

$$\begin{aligned} x^{(k+1)} - \xi &= x^{(k)} - (f'(x^{(k)}))^{-1} f(x^{(k)}) - \xi \\ &= -(f'(x^{(k)}))^{-1} \left(f(x^{(k)}) - \underbrace{f(\xi)}_{0} - f'(x^{(k)})(x^{(k)} - \xi) \right) \end{aligned}$$

Ecrivons sous forme d'intégrale l'expression

$$f(x^{(k)}) - f(\xi) - f'(x^{(k)})(x^{(k)} - \xi) = \int_{\xi}^{x^{(k)}} (f'(t) - f'(x^{(k)}))\, dt$$

On obtient ainsi :

$$x^{(k+1)} - \xi = -(f'(x^{(k)}))^{-1} \int_{\xi}^{x^{(k)}} (f'(t) - f'(x^{(k)}))\, dt$$

On considère la fonction

$$(x, y) \in \mathbb{R}^2 \to \varphi(x, y) = |(f'(x))^{-1}||f'(y) - f'(x)| \in \mathbb{R}.$$

Cette fonction est définie continue au voisinage de (ξ, ξ) et vérifie $\varphi(\xi, \xi) = 0$, donc, définition de la continuité en (ξ, ξ), il existe $\delta = \delta(\alpha)$ tel que $\{|x - \xi| < \delta,\ |y - \xi| < \delta\} \Rightarrow |\varphi(x, y)| < \alpha$. Donc si $|x^{(k)} - \xi| \leq \alpha^k |x^{(0)} - \xi|$ et $0 < \alpha < 1$, on a

$$|-(f'(x^{(k)}))^{-1}(f'(t) - f'(x^{(k)}))| = \varphi(x^{(k)}, t) < \alpha$$

puisque $|x^{(k)} - \xi| \leq \alpha^k |x^{(0)} - \xi| < \alpha^k \delta < \delta$ et $|t - \xi| \leq |x^{(k)} - \xi| < \delta$ et :

$$|x^{(k+1)} - \xi| \leq \int_{\xi}^{x^{(k)}} \varphi(x^{(k)}, t)\, dt \leq \alpha |x^{(k)} - \xi| \leq \alpha^{k+1} |x^{(0)} - \xi|$$

si bien que l'inégalité est vraie au rang $k+1$ donc à tous les rangs. cqfd.

Remarques. (i) Si on suppose f de classe C^2, dans le cas de convergence de la méthode, on a la vitesse de convergence quadratique, par le même raisonnement que dans la preuve du théorème précédent. Cependant, même si f est seulement de classe C^1, on voit que la convergence est plus rapide que toute suite géométrique α^k, car α est arbitraire dans la preuve. On retrouve la superconvergence de la méthode, même dans ce cas.

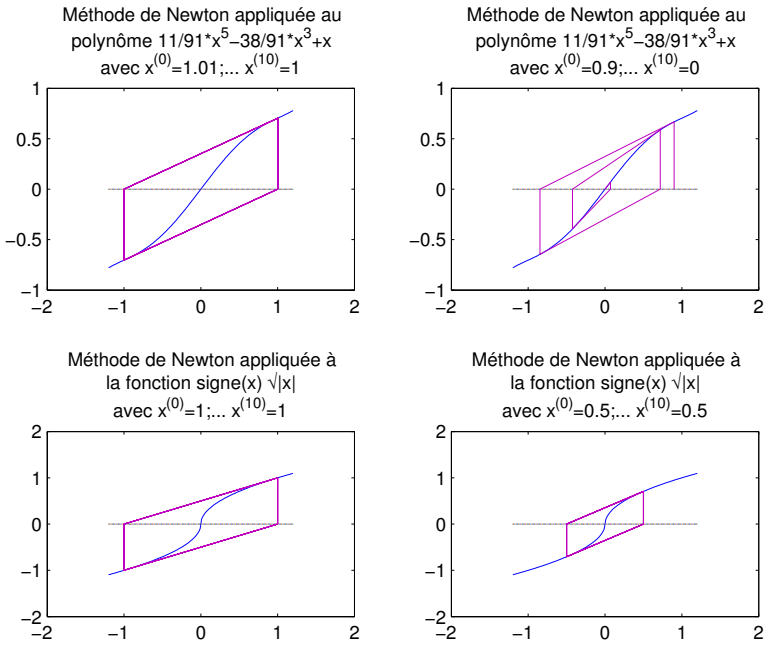
ii) Cette démonstration se généralise au cas d'une application $\mathbf{f}$ de $\mathbb{R}^n$ dans $\mathbb{R}^n$ avec $n > 1$, la méthode de Newton devenant dans ce cas, si $\mathbf{f}(x_1, x_2, ..., x_n) = (f_1(x_1, x_2, ..., x_n), f_2(x_1, x_2, ..., x_n), ..., f_n(x_1, x_2, ..., x_n))^T$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - (D\mathbf{f}(\mathbf{x}^{(k)}))^{-1} \mathbf{f}(\mathbf{x}^{(k)}),$$

où $(D\mathbf{f}(\mathbf{x}^{(k)}))^{-1}$ est l'inverse de la matrice Jacobienne de $\mathbf{f}$ en $\mathbf{x}^{(k)}$ (matrice $n \times n$)

$$D\mathbf{f}(\mathbf{x}^{(k)}) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} (\mathbf{x}^{(k)})$$ et bien sûr $\mathbf{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, ..., x_n^{(k)})^T$. Il suffit de remplacer les valeurs absolues $|.|$ par une norme $\|.\|$ dans $\mathbb{R}^n$.

iii) Cette démonstration permet de traiter le cas d'un zéro ξ qui est un point d'inflexion ; il y aura convergence si f' est continue sous réserve que l'approximation initiale soit assez proche de ξ. Mais si l'approximation initiale n'est pas assez proche de ξ, il y a des cas où on peut observer des suites avec plusieurs valeurs d'adhérences. Par exemple dans le cas de la fonction polynôme [5] $f(x) = \frac{11}{91}x^5 - \frac{38}{91}x^3 + x$, $x = 0$ est un zéro qui est un point d'inflexion ($f(0) = 0$,$f'(0) = 1$, $f''(0) = 0$) et on a $x^{(0)} = 1 \Rightarrow x^{(1)} = 1 - \frac{f(1)}{f'(1)} = 1 - \frac{\frac{64}{91}}{\frac{32}{91}} = -1 \Rightarrow x^{(2)} = -1 - \frac{f(-1)}{f'(-1)} = -1 - \frac{-\frac{64}{91}}{\frac{32}{91}} = 1$. Alors si $x^{(0)} = 1.01$ par exemple la suite $x^{(k)}$ présente deux points d'accumulation $x^{\infty} = \pm 1$, alors que si $x^{(0)} = 0.9$ il y a convergence vers 0. À noter que si la fonction $f(x)$ n'est pas dérivable en le zéro ξ il se peut qu'il y ait toujours plusieurs points d'accumulation quel que soit $x^{(0)} \neq \xi$. C'est ce qu'on observe [6] avec la fonction $f(x) = \begin{cases} \sqrt{x} \text{ si } x \geq 0 \\ -\sqrt{(-x)} \text{ si } x < 0 \end{cases}$



5. cf R. Fletcher, 1980, Practical methods of optimization, Vol 1, Unconstrained Optimization : Wiley, New York
6. "exemple pervers" cf C.B. Moler, 2004, Numerical computing with MATLAB, http ://www.mathworks.fr/moler).

2.3 ($\star$) Quelques procédés d'accélération de convergence de suites.

2.3.1 ($\star$) La méthode d'accélération de convergence appelée Δ^2 d'Aitken

Ce procédé consiste, à partir d'une suite de réels $\{x^{(k)}\}_{k\geq 0}$ qui converge vers une limite ξ mais vérifie $x^{(k)} \neq \xi$ pour tout k, à calculer une suite $\{x'^{(k)}\}_{k\geq 0}$ définie par :

$$x'^{(k)} = x^{(k)} - \frac{(x^{(k+1)} - x^{(k)})^2}{x^{(k+2)} - 2x^{(k+1)} + x^{(k)}}. \tag{2.6}$$

On note d'ordinaire $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$ et donc $\Delta^2 x^{(k)} = \Delta(x^{(k+1)} - x^{(k)}) = x^{(k+2)} - 2x^{(k+1)} + x^{(k)}$ d'où la terminologie Δ^2. Ce procédé permet d'accélérer la convergence des suites qui convergent à vitesse linéaire. Montrons que si la suite convergente $\{x^{(k)}\}_{k\geq 0}$ de limite ξ est telle que la suite $\{\theta^{(k)}\}_{k\geq 0}$ définie par $x^{(k+1)} - \xi = \theta^{(k)}(x^{(k)} - \xi)$ vérifie $\theta^{(k)} \to_{k\to\infty} \theta$ avec $|\theta| < 1$, alors :

$$\lim_{k\to\infty} \frac{x'^{(k)} - \xi}{x^{(k)} - \xi} = 0, \tag{2.7}$$

Ainsi la suite $\{x'^{(k)}\}_{k\geq 0}$ converge vers ξ à vitesse superlinéaire donc plus rapidement que $\{x^{(k)}\}_{k\geq 0}$.

Preuve. C'est un simple calcul :

$$x'^{(k)} - \xi = x^{(k)} - \xi - \frac{(x^{(k+1)} - \xi - (x^{(k)} - \xi))^2}{x^{(k+2)} - \xi - 2(x^{(k+1)} - \xi) + (x^{(k)} - \xi)} = (1 - \frac{(\theta^{(k)} - 1)^2}{\theta^{(k+1)} - 2\theta^{(k)} + 1})(x^{(k)} - \xi),$$

et comme $\theta^{(k)} \to_{k\to\infty} \theta \neq 1$, on peut affirmer que $\frac{(\theta^{(k)}-1)^2}{\theta^{(k+1)}-2\theta^{(k)}+1} \to_{k\to\infty} \frac{(\theta-1)^2}{\theta^2-2\theta+1} = 1$, d'où le résultat.

2.3.2 ($\star$) La méthode de Aitken-Steffensen

Dans le cas où la suite initiale $\{x^{(k)}\}_{k\geq 0}$ est de la forme $x^{(k+1)} = g(x^{(k)})$, on peut accélérer la convergence en réinjectant toutes les 3 itérations le résultat du procédé Δ^2 dans la suite d'origine, afin de « l'aider » à converger : Ainsi à partir de $x^{(0)}$, $g(x^{(0)})$, $g(g(x^{(0)}))$ on calcule le terme amélioré

$$x'^{(0)} = x^{(0)} - \frac{(g(x^{(0)}) - x^{(0)})^2}{g(g(x^{(0)})) - 2g(x^{(0)}) + x^{(0)}}.$$

Ensuite on itère deux fois $g(x'^{(0)})$, $g(g(x'^{(0)}))$ et on prend comme terme suivant le terme amélioré

$$x'^{(0)} - \frac{(g(x'^{(0)}) - x'^{(0)})^2}{g(g(x'^{(0)})) - 2g(x'^{(0)}) + x'^{(0)}}.$$

Et ainsi de suite. On remarque que la suite des termes améliorés est obtenue en appliquant la méthode des approximations successives non pas à la fonction g mais à la fonction G qui est :

$$G(x) = x - \frac{(g(x) - x)^2}{g(g(x)) - 2g(x) + x} \tag{2.8}$$

La méthode qui, pour résoudre $x = g(x)$, consiste à appliquer la méthode des approximations successives à $x = G(x)$ avec $G(x) = x - \frac{(g(x)-x)^2}{g(g(x))-2g(x)+x}$, s'appelle méthode de Aitken-Steffensen et en général elle converge aussi rapidement que la méthode de Newton. Cette rapidité de convergence s'explique assez facilement une fois qu'on a pu calculer les dérivées $G'(x)$ et $G''(x)$ en $x = \xi$ le point fixe de g ($g(\xi) = \xi$). Ce calcul n'est pas si facile "à la main" et l'utilisation d'un logiciel de calcul formel est pratique. On obtient :

$$G'(\xi) = 0,\ G''(\xi) = \frac{g'(\xi)g''(\xi)}{g'(\xi) - 1}$$

Ainsi la méthode de Aitken-Steffensen a en général le même ordre de convergence que la méthode de Newton et ne nécessite pas le calcul d'une dérivée. Cette méthode peut même converger si le point fixe de g n'est pas attractif (cas $g(\xi) = \xi$ avec $|g'(\xi)| > 1$).Mais il n'y a pas de généralisation à plusieurs dimensions.

2.4 Les méthodes de la fausse position, de la sécante et l'algorithme de Dekker-Brent

Le but de ce paragraphe est de décrire les principes de construction des algorithmes de calcul des zéros des fonctions réelles d'une variable réelle lorsqu'on connaît a priori un intervalle d'encadrement de la racine sur lequel la fonction change de signe et lorsqu'on préfère ou lorsqu'on ne peut pas calculer de dérivée. On trouve ces algorithmes [7] dans le logiciel `MATLAB` (fonction `fzero`) et dans la plupart des bibliothèques de calcul comme `SLATEC`, `NAG`, `IMSL`. Le calcul d'un zéro d'ordre pair ne peut pas être fait de cette façon et avec `MATLAB` on peut conseiller l'usage de la fonction `fminbnd` pour le calcul d'un minimum local.

2.4.1 Les méthodes de la fausse position et de la sécante

2.4.1.1 La méthode de la fausse position

Étant donnée une fonction réelle f définie continue sur un intervalle $[a,b]$, telle que $f(a)f(b)<0$, on conviendra ici d'appeler méthode de la fausse position la méthode qui consiste à approcher le zéro de f sur $[a,b]$ par le zéro du polynôme d'interpolation de f aux points a et b et de degré inférieur ou égal à 1, ceci permettant soit d'arrêter les calculs soit de trouver un intervalle plus petit contenant le zéro et de recommencer. L'expression de ce polynôme étant $p_1(x)=\frac{x-b}{a-b}f(a)+\frac{x-a}{b-a}f(b)$ (c'est bien une fonction polynôme de degré au plus 1 qui vaut $f(a)$ si $x=a$ et $f(b)$ si $x=b$) ce zéro est :

$$c=\frac{af(b)-bf(a)}{f(b)-f(a)}=\frac{a(f(b)-f(a))+af(a)-bf(a)}{f(b)-f(a)}=a-\frac{f(a)}{\frac{f(b)-f(a)}{b-a}}$$

On obtient ainsi l'algorithme :

$$\begin{array}{l}[a^{(0)},b^{(0)}]=[a,b]\text{ et pour }k=0,1,...\\ \text{on calcule }c=a^{(k)}-\frac{f(a^{(k)})}{\frac{f(b^{(k)})-f(a^{(k)})}{b^{(k)}-a^{(k)}}}\text{ et on définit le nouveau intervalle par :}\\ \left\{\begin{array}{l}a^{(k+1)}=b^{(k+1)}=c\text{ si }f(c)=0\\ a^{(k+1)}=a^{(k)},\ b^{(k+1)}=c\text{ si }f(a^{(k)})f(c)<0\\ a^{(k+1)}=c,\ b^{(k+1)}=b^{(k)}\text{ si }f(a^{(k)})f(c)>0\end{array}\right.\end{array}. \tag{2.9}$$

On pourrait penser que si f est assez régulière c'est à dire ici "convenablement approchée par un polynôme de degré 1, ce qui est le cas si f est de classe C^2 (deux fois continûment dérivable)", alors on obtient ainsi un algorithme plus performant que la méthode de dichotomie ou la méthode des approximations successives appliquée à la recherche d'un point fixe de $g(x)=x+f(x)$ ou $x-f(x)$ (dans les cas où cela fonctionne). Et bien ce n'est pas forcément le cas et si par exemple $f'(x)>0$, $f''(x)\geq 0$ pour $x\in[a,b]$ et si l'algorithme est infini, alors $b^{(k)}=b$ pour tout $k\geq 0$ et $a^{(k+1)}=a^{(k)}-\frac{f(a^{(k)})}{\frac{f(b)-f(a^{(k)})}{b-a^{(k)}}}\to_{k\to\infty}\xi$, l'unique zéro de f sur $[a,b]$, si bien qu'on obtient la méthode des approximations successives appliquée à la fonction $g(x)=x-\frac{f(x)}{\frac{f(x)-f(b)}{x-b}}$ (qui est aussi $g(x)=b-\frac{f(b)}{\frac{f(b)-f(x)}{b-x}}$) et la convergence n'est que linéaire, plus précisément $\frac{a^{(k+1)}-\xi}{a^{(k)}-\xi}\to_{k\to\infty}g'(\xi)$ où $0<g'(\xi)<1$ en général.

Preuve. On utilise d'abord la convexité de f sur $[a,b]$ (le graphe de $f(x)$ sur un intervalle est en dessous de la corde qui joint les extrémités du graphe sur cet intervalle) dont la traduction analytique est $f(\theta x+(1-\theta)y)\leq\theta f(x)+(1-\theta)f(y)$ pour $0\leq\theta\leq 1$ et $x,y\in[a,b]$. Cette inégalité se déduit de $f''(x)\geq 0$ comme suit. On écrit, en utilisant la formule des accroissements finis sur f puis sur f' ($\varphi(v)-\varphi(u)=\varphi'(w)(v-u)$ où $w\in]u,v[$ si $\varphi:[u,v]\to\mathbb{R}$ est continue et $\varphi:]u,v[\to\mathbb{R}$ dérivable) :

$$\begin{array}{rl}f(\theta x+(1-\theta)y)-(\theta f(x)+(1-\theta)f(y))&=\theta(f(\underbrace{\theta x+(1-\theta)y}_{x+(1-\theta)(y-x)})-f(x))+(1-\theta)(f(\underbrace{\theta x+(1-\theta)y}_{y-\theta(y-x)})-f(y))\\ &=\theta(1-\theta)(y-x)(f'(\eta_1)-f'(\eta_2)),\end{array}$$

7. Dans le logiciel `SCILAB` (fonction `fsolve`) l'algorithme utilisé est le même que celui à plusieurs variables et est donc bien plus compliqué (dérivée de `MINPACK`).

avec $x < \eta_1 < \theta x + (1-\theta)y < \eta_2 < y$, et donc, avec $\eta_1 < \eta_3 < \eta_2$:

$$f(\theta x + (1-\theta)y) - (\theta f(x) + (1-\theta)f(y)) = \theta(1-\theta)(y-x)(\eta_1 - \eta_2)f''(\eta_3) \leq 0 \text{ cqfd.}$$

Revenant à notre problème on note que si $a^{(k)} < b^{(k)}$, ce qui est vrai pour $k = 0$, $c = a^{(k)} - \frac{f(a^{(k)})}{\frac{f(b^{(k)}) - f(a^{(k)})}{b^{(k)} - a^{(k)}}}$ qui appartient à $[a^{(k)}, b^{(k)}]$ (car le polynôme de degré 1, p_1 vérifie $p_1(a^{(k)}) < 0 < p_1(b^{(k)})$), s'écrit aussi $c = \underbrace{\frac{b^{(k)} - c}{b^{(k)} - a^{(k)}}}_{\theta \text{ tel que } 0<\theta<1} a^{(k)} + \underbrace{\frac{c - a^{(k)}}{b^{(k)} - a^{(k)}}}_{1-\theta} b^{(k)}$ et comme f est convexe $f(c) \leq \frac{b^{(k)} - c}{b^{(k)} - a^{(k)}} f(a^{(k)}) + \frac{c - a^{(k)}}{b^{(k)} - a^{(k)}} f(b^{(k)}) =$ $p_1(c) = 0$ si bien que, si l'algorithme est infini, cette inégalité est toujours stricte et on a $b^{(k+1)} = b^{(k)} = b$ et $a^{(k+1)} = c = a^{(k)} - \frac{f(a^{(k)})}{\frac{f(b) - f(a^{(k)})}{b - a^{(k)}}}$ (qui est aussi $b - \frac{f(b)}{\frac{f(b) - f(a^{(k)})}{b - a^{(k)}}}$). L'algorithme est bien la méthode des approximations successives appliquée à $x = g(x)$ avec $g(x) = x - \frac{f(x)}{\frac{f(b) - f(x)}{b - x}}$ ou encore $g(x) = b - \frac{f(b)}{\frac{f(b) - f(x)}{b - x}}$.

Il est clair que la suite $\{a^{(k)}\}_{k \geq 0}$ est croissante et converge vers l'unique zéro ξ de f sur $[a, b]$ et comme $a^{(k+1)} - \xi = g(a^{(k)}) - \xi = g(a^{(k)}) - g(\xi) = g'(\eta^{(k)})(a^{(k)} - \xi)$ où $a^{(k)} < \eta^{(k)} < \xi$, il vient $\frac{a^{(k+1)} - \xi}{a^{(k)} - \xi} \to_{k \to \infty} g'(\xi)$. Il ne reste plus qu'à exprimer cette dérivée pour montrer que $0 < g'(\xi) < 1$. Or :

$$g'(\xi) = (x - \frac{f(x)}{\frac{f(b) - f(x)}{b - x}})'_{x=\xi} = 1 - \frac{f'(\xi)}{\frac{f(b) - f(\xi)}{b - \xi}} + \underbrace{(\frac{f(x)(\frac{f(b) - f(x)}{b - x})'}{(\frac{f(b) - f(x)}{b - x})^2})_{x=\xi}}_{0} = \underbrace{1 - (b - \xi)\frac{f'(\xi)}{f(b)}}_{<1},$$

$$= \frac{f(b) - (b - \xi)f'(\xi)}{f(b)} = \frac{f(b) - f(\xi) - (b - \xi)f'(\xi)}{f(b)} = \frac{(b - \xi)^2 f''(\eta)}{2f(b)} \geq 0$$

la dernière assertion résultant de la convexité de f. En général on a $g'(\xi) \neq 0$ donc l'algorithme converge seulement à vitesse linéaire. Ainsi cette méthode n'est pas performante et peut même être moins efficace que la méthode de dichotomie si $g'(\xi) > \frac{1}{2}$.

2.4.1.2 La méthode de la sécante

Dans la méthode de Newton, il n'est pas toujours possible de calculer la dérivée de la fonction $f(x)$ ou bien le calcul de cette dérivée peut être très coûteux en temps de calcul. On peut alors remplacer la dérivée par la pente de la sécante aux deux derniers points calculés. Cette méthode repose sur la même formule que la précédente mais vue cette fois comme une approximation de la méthode de Newton où la dérivée $f'(x^{(k)})$ dans la formule $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$ est approchée par $\frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}$. L'algorithme est donc :

$$x^{(0)} = a,\ x^{(1)} = b,\ x^{(k+1)} = \begin{cases} x^{(k)} - \frac{f(x^{(k)})}{\frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}} \text{ si } f(x^{(k)}) \neq 0 \\ x^{(k)} \text{ si non} \end{cases}, \text{ pour } k \geq 1. \tag{2.10}$$

Il se trouve que cet algorithme simple, dans lequel on abandonne la tâche de conserver un encadrement de la racine, est performant et utilisé en pratique. En fait on peut montrer que *lorsqu'il converge*, l'algorithme converge à *vitesse superlinéaire*, plus précisément si on définit l'erreur à l'étape k par $e_k = \xi - x^{(k)}$, on peut prouver que

$$e_{k+1} \sim \text{Cte} \cdot (e_k)^{\Phi}$$

où $\Phi = \frac{1 + \sqrt{5}}{2}$ est le nombre d'or.

Pour une preuve, voir l'article M. Vianello, Zanovello, 1992, On the superlinear convergence of the secant method : Amer. Math. Monthly, 758 – 761

ainsi que A.M. Ostrowski, 1970, Solution of equations in Euclidean and Banach spaces : Acad. Press., N.Y.

2.4.2 ($\star$) L'algorithme de Dekker-Brent

Étant donnés un intervalle $[a,b] \subset \mathbb{R}$ et une fonction réelle f telle que $f(a)f(b) < 0$, il est possible de modifier la méthode de la sécante, en la combinant à la méthode de dichotomie, afin d'obtenir à chaque itération un encadrement de la racine et ceci sans altérer la vitesse de convergence, ni augmenter le nombre d'évaluations de f. C'est la modification de Dekker [8] qui peut être décrite succinctement comme suit, les détails devant être cherchés dans le code `MATLAB fzerotx.m` placé en appendice et extrait du livre de C.B. Moler disponible sur internet comme indiqué dans le chapitre d'introduction :
à chaque pas de l'algorithme :

1. on dispose d'un encadrement $[a,b]$ de la racine, on fait en sorte que $|f(b)| \leq |f(a)|$, la précédente valeur de b étant conservée et notée c (au départ on pose $c = a$ mais ensuite c est en général différent de b et a) et on fait en sorte que $f(a)f(b) < 0$;
2. ensuite on calcule l'approximation $x^{(k+1)}$ de la racine donnée par la formule (2.10) avec $x^{(k-1)} = c$, $x^{(k)} = b$, puis on pose $c = b$ et selon que $x^{(k+1)}$ est ("approximativement" : voir code pour plus de détails) dans l'intervalle $[a,b]$ ou non on retient comme nouveau b l'approximation $x^{(k+1)}$ ou le point milieu $\frac{a+b}{2}$ donné par la méthode de dichotomie.

On retiendra la simplicité du procédé.

Mais dans les cas où on dispose de trois points a, b, c distincts, on peut obtenir, en général, une approximation encore meilleure que celle donnée par la méthode de la sécante en effectuant une interpolation à trois points. Il existe une méthode, c'est la méthode de Muller, dans laquelle on calcule comme nouvelle approximation de la racine le zéro du polynôme de degré deux au plus qui interpole f en ces trois points, en choisissant le zéro au voisinage de l'approximation précédente. Cette méthode nécessite le calcul d'une racine carrée mais on va voir comment éviter cela sans altérer l'ordre de convergence. Au lieu de chercher une parabole $y = p(x)$ qui passe par trois points, on cherche une parabole $x = q(y)$ qui passe par les mêmes points. L'idée est de chercher à interpoler par une fonction polynôme $y \mapsto q(y)$ de degré deux telle que $x^{(j)} = q(f(x^{(j)})$ pour $j = k, k-1, k-2$. Il suffit alors de calculer $x = q(y)$ pour $y = 0$ pour trouver l'intersection de la parabole avec l'axe des abscisses !

Ce procédé porte le nom d'interpolation quadratique inverse (on a permuté les variables x et y). La formule obtenue ne nécessite pas le calcul de la racine d'un trinôme du second degré comme dans la méthode de Muller. On utilise simplement l'interpolation de Lagrange (voir chapitre suivant Interpolation) : La formule de l'interpolation quadratique inverse est donc simplement :

$$\begin{array}{l} x^{(k+1)} = q_2(0), \text{ où} \\ q_2(y) = \frac{(y-f(x^{(k-1)}))(y-f(x^{(k)}))}{(f(x^{(k-2)})-f(x^{(k-1)}))(f(x^{(k-2)})-f(x^{(k)}))}\, x^{(k-2)} \\ + \frac{(y-f(x^{(k-2)}))(y-f(x^{(k)}))}{(f(x^{(k-1)})-f(x^{(k-2)}))(f(x^{(k-1)})-f(x^{(k)}))}\, x^{(k-1)} \\ + \frac{(y-f(x^{(k-1)}))(y-f(x^{(k-2)}))}{(f(x^{(k)})-f(x^{(k-1)}))(f(x^{(k)})-f(x^{(k-2)}))}\, x^{(k)} \end{array} \tag{2.11}$$

et on appelle algorithme de Dekker-Brent [9] la modification suivante de l'algorithme décrit plus haut :
à chaque pas de l'algorithme :

1. on dispose d'un encadrement $[a,b]$ de la racine, on fait en sorte que $|f(b)| \leq |f(a)|$, la précédente valeur de b étant conservée et notée c (au départ on pose $c = a$ mais ensuite c est en général différent de b et a) et on fait en sorte que $f(a)f(b) < 0$;
2. ensuite on calcule l'approximation $x^{(k+1)}$ de la racine donnée par la formule (2.11) si c'est possible c'est à dire si les trois points a, b, c sont différents et par (2.10) si non (donc avec $x^{(k-2)} = a$, $x^{(k-1)} = c$, $x^{(k)} = b$), puis on pose $c = b$ et selon que $x^{(k+1)}$ est ("approximativement" : voir code pour plus de détails) dans l'intervalle $[a,b]$ ou non on retient comme nouveau b l'approximation $x^{(k+1)}$ ou le point milieu $\frac{a+b}{2}$ donné par la méthode de dichotomie.

Lorsque f est assez régulière (de classe C^3 convient), on peut montrer que l'ordre de convergence est $p \simeq 1.84$, la plus grande racine de $p^3 - p^2 - p - 1 = 0$. Cela signifie que $e^{(k+1)} = O\left((e^{(k)})^p\right)$ lorsque k tend vers l'infini,

8. La référence est : T.J. Dekker, 1969, Finding zeros by means of successive linear interpolation, dans "Constructive Aspects of the Fundamental Theorem of Algebra, B. Dejon and P. Henrici (editors), Wiley-Interscience, New York, pages 37 – 48.
9. R.P. Brent, 1973, Algorithms for minimization without derivatives : Prentice-Hall, Englwood Cliffs, NJ.

où $e^{(k)} = \xi - x^{(k)}$ où ξ est la racine.
On pourra consulter le code `MATLAB fzerotx.m` placé en appendice pour la mise en oeuvre pratique. On trouvera dans le livre déjà cité A.M. Ostrowski, 1970, Solution of equations in Euclidean and Banach spaces : Acad. Press., N.Y. la preuve que les ordres de convergence des méthodes obtenues par interpolation inverse à 4, 5, ..., n points ne dépassent jamais 2.

2.5 ($\star$) Compléments : notions sur le calcul de zéros multiples

Pour le calcul des racines simples des polynômes orthogonaux par exemple qui seront utilisées dans la construction des formules de quadrature de Gauss (deux chapitres plus loin), la méthode de Newton fonctionne très bien et on peut même construire une méthode à convergence cubique [10] :

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}(1 + \frac{f(x^{(k)})}{f'(x^{(k)})}\frac{f''(x^{(k)})}{2f'(x^{(k)})}). \tag{2.12}$$

À noter que pour vérifier rapidement que la méthode de Newton est d'ordre 2 (resp. que (2.12) est d'ordre 3), il "suffit" de calculer les dérivées d'ordre 1 et 2 (resp. 1, 2 et 3) de la fonction $g(x) = x - \frac{f(x)}{f'(x)}$ (resp. $g(x) = x - \frac{f(x)}{f'(x)}(1 + \frac{f(x)}{f'(x)}(1 + \frac{f(x)}{f'(x)}\frac{f''(x)}{2f'(x)}))$ en le zéro de $f(x)$: une dérivée première nulle avec une dérivée seconde non nulle donne une méthode d'ordre 2 et des dérivées première et seconde nulles avec une dérivée troisième non nulle donne une méthode d'ordre 3 (faire la démonstration en exercice, en justifiant pour f suffisamment dérivable, que si la suite $\{x^{(k)}\}_{k\geq 0}$ converge vers le zéro ξ de f, alors $x^{(k+1)} - \xi = g(x^{(k)}) - g(\xi) = \frac{g^{(p)}(\eta^{(k)})}{p!}(x^{(k)} - \xi)^p$ pour un certain entier $p \geq 2$ avec $g^{(p)}(x) \equiv \frac{d^p g}{dx^p}(x)$ et $\eta^{(k)}$ à préciser). Mais à la "main" ces calculs sont fastidieux et il vaut mieux utiliser une système de calcul formel. Avec les instructions `MAPLE` suivantes (on donne aussi ce qui est affiché) :
on vérifie que la méthode de Newton est d'ordre 2 (d'ordre 3 si f'' s'annule en le zéro). De même pour l'autre méthode, on trouve que si ξ est un zéro de f alors $g'(\xi) = g''(\xi) = 0$ et $g'''(\xi) = 3\frac{(f''(\xi))^2}{(f'(\xi))^2} - \frac{f'''(\xi)}{f'(\xi)}$.

Mais lorsque le zéro est multiple (d'ordre $r > 1$) la méthode de Newton n'est plus que d'ordre 1. Si $f(x) = (x - \xi)^r h(x)$ avec $r > 1$, h de classe C^2 et $h(\xi) \neq 0$, on a, lorsque $x^{(k)} \to_{k\to\infty} \xi$, $\lim_{k\to\infty} \frac{|x^{(k+1)} - \xi|}{|x^{(k)} - \xi|} = \frac{r-1}{r}$. Une démonstration de ce résultat [11], fondée sur l'application de la formule de Taylor à l'ordre 2 au voisinage de $x^{(k)}$ à $\frac{f(x)}{(x-\xi)^{r-1}}$, est proposée en td.
On montre aussi en td que dans ce cas d'un zéro d'ordre $r > 1$, si on change l'itération de Newton $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}$ en $x^{(k+1)} = x^{(k)} - r\frac{f(x^{(k)})}{f'(x^{(k)})}$ alors, de nouveau, l'ordre de convergence est 2. Il faut quand même remarquer que les problèmes conduisant à des calculs de racines multiples sont assez singuliers en ce sens que si l'équation $f(x) = 0$ est perturbée en $f(x) + \delta f(x) = 0$ la racine multiple de $f(x) = 0$ va être perturbée en plusieurs racines simples de $f(x) + \delta f(x) = 0$.

Dans ce genre de problèmes plus ou moins singuliers, on ne connaît pas forcément l'ordre de la racine et il est remarquable qu'il existe une méthode de calcul de racines de fonctions d'une variable (réelle ou complexe d'ailleurs) dont l'ordre de convergence est 2 quel que soit l'ordre de la racine. Il s'agit de l'itération [12] :

$$x^{(k+1)} = x^{(k)} - \frac{f(x^k)f'(x^{(k)})}{(f'(x^{(k)}))^2 - f(x^{(k)})f''(x^{(k)})} \tag{2.13}$$

Avec un logiciel de calcul formel, on peut vérifier que l'ordre de convergence est 2 quelque soit l'ordre du zéro.

Pour terminer cette sous section voici un exemple simple de manipulations en `MATLAB` qui illustre la difficulté de calculer des racines multiples. Pour compter le nombre d'itérations que fait la fonction `fzero` il faut l'utiliser sous la forme `[x,fval,exitflag,output] = fzero(...);` et le nombre d'itérations est

10. Je l'ai trouvée dans le livre P.J. Davis, P. Rabinowitz, 1984, Methods of numerical integration (second edition) : Acad. Press, New York, dans le sous-programme FORTRAN GRULE page 487 assez difficile à décoder.

11. qui montre aussi que la méthode de Newton converge encore pourvu que $x^{(0)}$ soit assez voisin de ξ

12. Je l'ai trouvé dans le livre P. Borwein, T. Erdélyi, 1995, Polynomials and polynomial inequalities : Springer, page 365 qui renvoie au livre P. Henrici, 1974, Applied and computational complex analysis, Volume I : John Wiley & Sons, New York, exercice 5 page 533 ;

`output.iterations`. Pour afficher les itérations il faut lancer l'instruction `options=optimset('Display','iter');` avant l'appel de `fzero` qui doit alors être de la forme `fzero(fonction,[a,b],options)`. Alors si les instructions :

```
options=optimset('Display','iter');
[x,fval,exitflag,output] = fzero(@(x) sinh(x),[-2 1],options);
```

donne le résultat en 8 itérations, les instructions :

```
options=optimset('Display','iter');
[x,fval,exitflag,output] = fzero(@(x) (x.^2).*sinh(x),[-2 1],options);
```

donnent le résultat en 151 itérations !

2.6 ($\star$) Appendice : le code fzerotx.m

```
function b = fzerotx(F,ab,varargin)
%FZEROTX Textbook version of FZERO.
% x = fzerotx(F,[a,b]) tries to find a zero of F(x) between a and b.
% F(a) and F(b) must have opposite signs. fzerotx returns one
% end point of a small subinterval of [a,b] where F changes sign.
% Arguments beyond the first two, fzerotx(F,[a,b],p1,p2,...),
% are passed on, F(x,p1,p2,..).
%
% Examples:
% fzerotx(@sin,[1,4])
% MATLAB6: F = inline('sin(x)'); fzerotx(F,[1,4])
% MATLAB7: F = @(x) sin(x); fzerotx(F,[1,4])
% Initialize.
a = ab(1);
b = ab(2);
fa = feval(F,a,varargin{:});
fb = feval(F,b,varargin{:});
if sign(fa) == sign(fb)
    error('Function must change sign on the interval')
end
c = a;
fc = fa;
d = b - c;
e = d;
% Main loop, exit from middle of the loop
while fb ~= 0
% The three current points, a, b, and c, satisfy:
% f(x) changes sign between a and b.
% abs(f(b)) <= abs(f(a)).
% c = previous b, so c might = a.
% The next point is chosen from
% Bisection point, (a+b)/2.
% Secant point determined by b and c.
% Inverse quadratic interpolation point determined
% by a, b, and c if they are distinct.
    if sign(fa) == sign(fb)
        a = c; fa = fc;
        d = b - c; e = d;
    end
    if abs(fa) < abs(fb)
```

```
        c = b; b = a; a = c;
        fc = fb; fb = fa; fa = fc;
    end
    % Convergence test and possible exit
    m = 0.5*(a - b);
    tol = 2.0*eps*max(abs(b),1.0);
    if (abs(m) <= tol) | (fb == 0.0)
        break
    end
    % Choose bisection or interpolation
    if (abs(e) < tol) | (abs(fc) <= abs(fb))
    % Bisection
        d = m;
        e = m;
    else
    % Interpolation
        s = fb/fc;
        if (a == c)
        % Linear interpolation (secant)
            p = 2.0*m*s;
            q = 1.0 - s;
        else
        % Inverse quadratic interpolation
            q = fc/fa;
            r = fb/fa;
            p = s*(2.0*m*q*(q - r) - (b - c)*(r - 1.0));
            q = (q - 1.0)*(r - 1.0)*(s - 1.0);
        end;
        if p > 0, q = -q; else p = -p; end;
        % Is interpolated point acceptable
        if (2.0*p < 3.0*m*q - abs(tol*q)) & (p < abs(0.5*e*q))
            e = d;
            d = p/q;
        else
            d = m;
            e = m;
        end;
    end
    % Next point
    c = b;
    fc = fb;
    if abs(d) > tol
        b = b + d;
    else
        b = b - sign(b-a)*tol;
    end
    fb = feval(F,b,varargin{:});
end
```

Chapitre 3

Interpolation.

3.1 Interpolation polynomiale

3.1.1 Existence et unicité du polynôme d'interpolation.

Le problème de l'interpolation est le suivant : Etant donné $n+1$ points distincts (x_i, y_i), $i = 0 \dots n$, construire une courbe passant par ces points. On désire de plus que la courbe soit *lisse* et donnée par une formule simple, du type $y = f(x)$.

Ce problème intervient dans de nombreuses situations industrielles.

Exemple. génie mécanique : construction de carrosserie par machines à commande numérique. On rentre un certains nombres de points de contrôle (x_i, y_i) et la machine calcule et usine la forme passant par ces points.

Exercice. Montrer que s'il existe une fonction f dont le graphe contient les points (x_i, y_i), $i = 0 \dots n$ alors les abscisses x_i sont toutes distinctes.

Une façon simple de résoudre le problème est de chercher la fonction f sous forme d'un *polynôme* (qui est toujours $\mathcal{C}^\infty$). L'algèbre linéaire donne immédiatement le résultat d'existence et d'unicité du polynôme d'interpolation.

Théorème 4 *Etant donnés $n+1$ abscisses distinctes $x_0, x_1, \dots x_n$ et $n+1$ valeurs quelconques $y_0, y_1, \dots y_n$, il existe un unique polynôme P de degré au plus n tel que $P(x_j) = y_j$, $j = 0 \dots n$.*

Preuve. Notons $\mathbb{R}_n[X] = \left\{ \sum_{k=0}^{n} a_k X^k, a_k \in \mathbb{R} \right\}$ l'espace vectoriel des polynômes de degré inférieur ou égal à n. Soit l'application linéaire :

$$\begin{aligned} F : \mathbb{R}_n[X] &\to \mathbb{R}^{n+1} \\ P &\mapsto (P(x_0), P(x_1), \dots, P(x_n)) \end{aligned}$$

Le noyau de F se réduit au polynôme nul $P = 0$. En effet un polynôme de degré inférieur ou égal à n a au plus n racines, sauf s'il s'agit du polynôme nul. L'application F est donc injective. D'autre part la dimension de $\mathbb{R}_n[X]$ est $n+1$, donc F est un isomorphisme. ∎

On pourrait penser que ce théorème résout complètement la question de l'interpolation. En fait, le résultat précédent est *théorique*, il reste à construire le polynôme P, de façon simple, précise et flexible : c'est l'objet du reste du chapitre.

Remarque. On peut calculer les coefficients du polynôme en résolvant un système linéaire suivant dit de Vandermonde.

$$\left\{ \begin{array}{rcl} a_0 + a_1 x_0 + \dots + a_n x_0^n &=& y_0 \\ a_0 + a_1 x_1 + \dots + a_n x_1^n &=& y_1 \\ \vdots & = & \vdots \\ a_0 + a_1 x_n + \dots + a_n x_n^n &=& y_n \end{array} \right.$$

dont les inconnues sont les coefficients du polynôme, i.e. le vecteur $\mathbf{a} = (a_0, a_1, \dots, a_n)^T$ de dimension $n+1$.

La matrice de taille $(n+1)\times(n+1)$

$$A = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix}$$

est appelée matrice de Vandermonde. On constate que si l'on note $\mathbf{a} = (a_0, a_1, \dots, a_n)^T \in \mathbb{R}^{n+1}$ et $\mathbf{y} = (y_0, y_1, \dots, y_n)^T \in \mathbb{R}^{n+1}$ le système (3.1.1) s'écrit simplement

$$A\,a = \mathbf{y} \in \mathbb{R}^{n+1}.$$

En théorie on peut donc calculer les coefficients du polynôme en résolvant ce système linéaire. Cependant la matrice de Vandermonde devient très mal conditionnée lorsque n devient grand. Donc cette méthode n'est pas très fiable lorsque n dépasse 8. □

3.1.2 Interpolation et approximation

On peut aussi se poser la question de la précision de l'interpolation polynômiale pour approcher une fonction donnée f sur un segment $[a,b]$. Etant donnée une fonction f, quel erreur commet-on si on la remplace par un polynôme prenant les mêmes valeurs en des points fixés ?

Soit f une fonction définie sur le segment $[a,b]$. Soit $n+1$ points distincts $x_0, x_1, \dots x_n$ de $[a,b]$. D'après le théorème 4, il existe un unique polynôme P_n de degré $\leq n$ tel que

$$P_n(x_j) = f(x_j),\ j = 0 \dots n.$$

On l'appelle *polynôme d'interpolation* de f aux points x_i, $i = 0, \dots n$.

Théorème 5 (des accroissements finis généralisés) *Soit f une fonction de classe $\mathcal{C}^{n+1}$ sur le segment $[a,b]$. Soit $n+1$ points distincts $x_0, x_1, \dots x_n$ de $[a,b]$. Soit P_n le polynôme de degré $\leq n$ qui interpole f aux x_i. Il vérifie pour tout x de $[a,b]$:*

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}(x-x_0)(x-x_1)\dots(x-x_n) \tag{3.1}$$

où ξ_x est un point de $[a,b]$ [1].

Preuve. La preuve repose sur le théorème de Rolle appliqué « en rafale ». Considérons la fonction

$$\varphi(t) = f(t) - P_n(t) - \lambda \cdot (t-x_0)(t-x_1)\dots(t-x_n)$$

où λ est un paramètre réel qui sera fixé plus tard. Soit maintenant $x \in \mathbb{R}$ différent des x_j, $j = 0 \dots n$. La fonction φ s'annule en $x_0, x_1, \dots x_n$. Choisissons maintenant λ de sorte que φ s'annule également en $t = x$. C'est possible car $(x-x_0)(x-x_1)\dots(x-x_n) \neq 0$. Ainsi la fonction φ s'annule en $x_0, x_1, \dots x_n$ et x donc au moins $n+2$ fois sur l'intervalle $[a,b]$. Par application du théorème de Rolle sur chaque intervalle séparant deux zéros consécutifs de φ, on obtient que la fonction dérivée φ' s'annule donc au moins $n+1$ fois sur $[a,b]$. De même la dérivée seconde φ'' s'annule donc au moins n fois sur $[a,b]$. En réitérant, on obtient que $\varphi^{(n+1)}$ s'annule au moins une fois sur $[a,b]$. Notons ξ_x ce point tel que $\varphi^{(n+1)}(\xi_x) = 0$. Mais la dérivée $(n+1)$-ème de $\varphi(t)$ se calcule aisément (P_n étant de degré n disparaît !) :

$$\varphi^{(n+1)}(t) = f^{(n+1)}(t) - \lambda(n+1)!$$

D'où l'on tire que

$$\lambda = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}.$$

1. dépendant évidemment de x

Ecrivons alors le fait que φ s'annule en $t = x$.

$$0 = f(x) - P_n(x) - \lambda \cdot (x - x_0)(x - x_1) \dots (x - x_n).$$

En remplaçant λ par sa valeur on obtient le résultat.

■

Remarque. Lorsque $n = 0$, on retrouve le théorème des accroissements finis $f(x) - f(x_0) = f'(\xi)(x - x_0)$. □

Ce théorème précise l'erreur d'*approximation* entre f et P_n.

$$|f(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |(x - x_0)(x - x_1) \dots (x - x_n)|$$

où $M_{n+1} = \max_{t \in [a,b]} |f^{(n+1)}(t)|$ qui est bien défini puisque par hypothèse $f^{(n+1)}$ est continue sur $[a, b]$. Supposons que les x_i soit régulièrement espacés, i.e.

$$x_i = a + i \cdot h,\ h = (b - a)/n,\ i = 0, \dots n$$

on peut alors calculer aisément le maximum de $\Pi_{0 \leq i \leq n}(x - x_i)$.

Exercice. Montrer que

$$|f(x) - P_1(x)| \leq \frac{1}{8} M_2 h^2$$

$$|f(x) - P_2(x)| \leq \frac{\sqrt{3}}{27} M_3 h^3$$

$$|f(x) - P_3(x)| \leq \frac{3}{128} M_4 h^4$$

En conclusion, nous voyons que la qualité de l'interpolation dépend de $\varphi(x) = (x - x_0)(x - x_1) \dots (x - x_n)$. Cette fonction s'annulle aux x_i et vu les résultats particulier des exercices ci-dessus, on pourrait croire que

$$|f(x) - P_n(x)| = O(h^{n+1})$$

donc que la qualité de l'approximation augmente avec le nombre de points. Or il n'en est rien, comme le montre la figure 3.1 qui trace la fonction φ pour 9 points x_i équidistants sur $[-1, 1]$ (le pas $h = 0.25$). Le maximum de φ est environ 0.02 et non pas $h^9 \approx 4.10^{-6}$. En fait lorsque x est loin d'un noeud x_k, le terme $(x - x_k)$ peut être grand et on ne pas simultanément rendre tous les termes $(x - x_i)$ petits ... Ce phénomène est général avec des points x_i équidistants : lorsqu'on augmente le nombre de points, l'approximation est très mauvaise aux extrémités. Le polynôme d'interpolation oscille avec un grande amplitude vers les extrémités.

3.1.3 Phénomène de Runge

Soit la fonction $t \mapsto 1/(1 + t^2)$. Considérons son polynôme d'interpolation aux points $-5, -4, \dots, 4, 5$. Il se calcule aisément, par exemple avec MAPLE.

$$t \mapsto -\frac{1}{44200} t^{10} + \frac{7}{5525} t^8 - \frac{83}{3400} t^6 + \frac{2181}{11050} t^4 - \frac{149}{221} t^2 + 1$$

Si on trace les graphes des fonctions, on voit le problème des oscillations aux extrémités, voir fig. 3.2. On peut remédier à ce problème en prenant des x_i *non équidistants* ou en prenant plus de points vers les extrémités, voir l'interpolation aux noeuds de Tchebitcheff, Cf le livre de Schwarz [**?**], voir fig. 3.3.

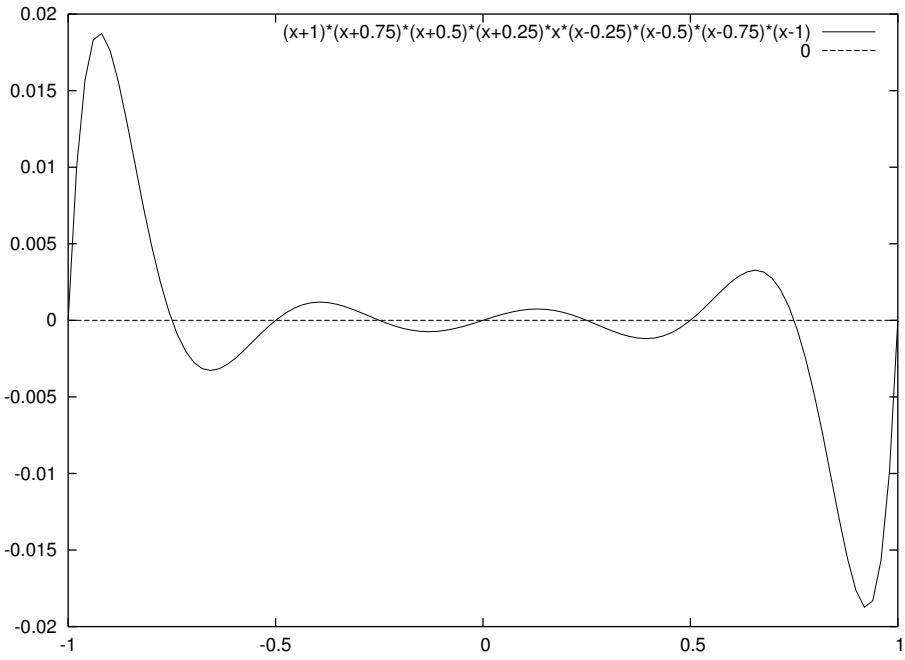


FIGURE 3.1 – Erreur interpolation

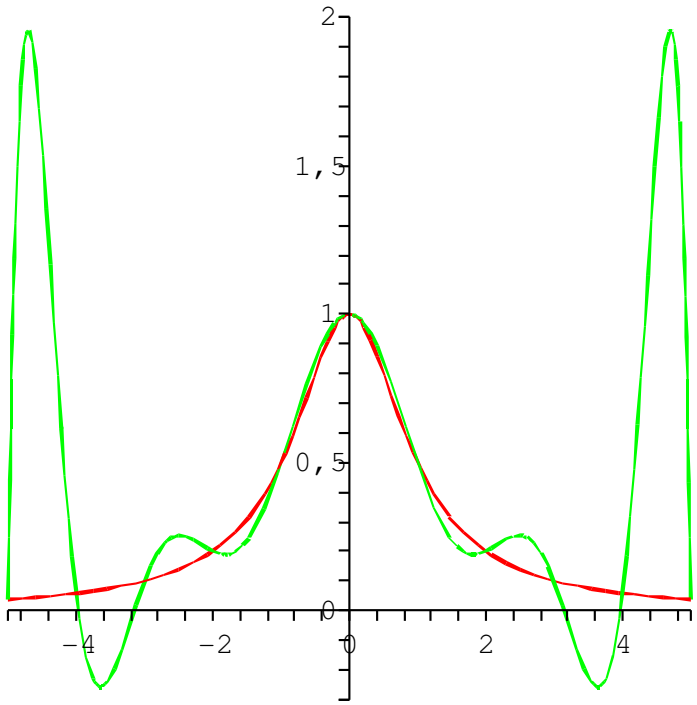


FIGURE 3.2 – Phénomène de Runge

3.2 Polynômes de Lagrange.

Soient $n+1$ points distincts $x_0, x_1, \dots x_n$. Nous allons donner une base de $\mathbb{R}_n[X]$ particulièrement adaptée à l'interpolation aux points x_i. Pour $j = 0, \dots, n$, notons

$$L_j(X) = \frac{\Pi_{i \neq j}(X - x_i)}{\Pi_{i \neq j}(x_j - x_i)}$$

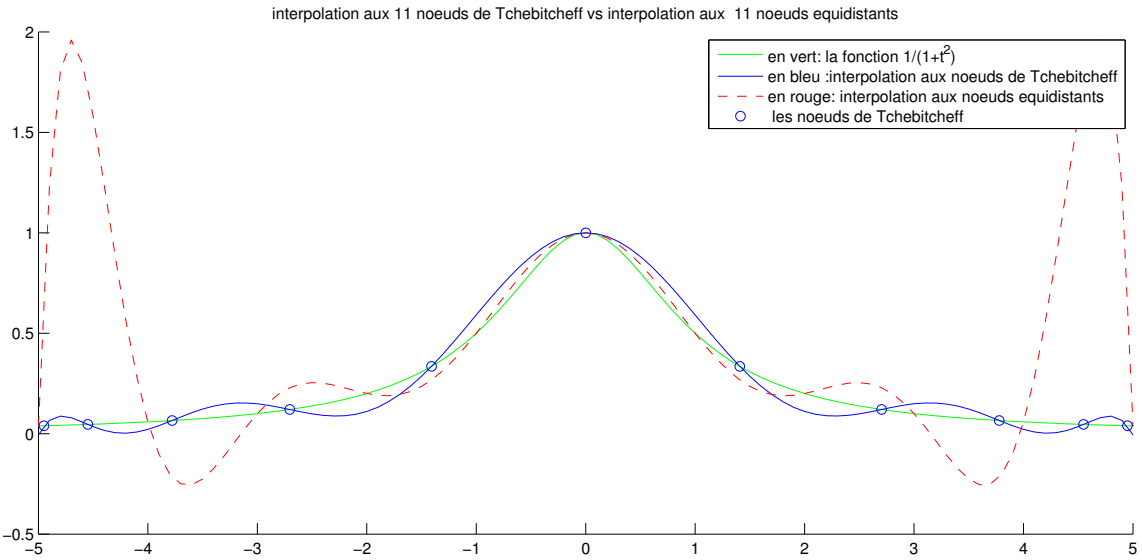


FIGURE 3.3 – Interpolation de Tchebitcheff.

Remarque. Attention, c'est bien un polynôme, et non une fraction rationnelle : l'inconnue X ne figure qu'au numérateur. Le dénominateur est une constante de *normalisation* prévue pour que $L_j(x_j) = 1$. □

Exemple. Fixons $n = 2$ et explicitons les polynômes de Lagrange.

$$L_0(X) = \frac{(X - x_1)(X - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

$$L_1(X) = \frac{(X - x_0)(X - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

$$L_2(X) = \frac{(X - x_0)(X - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

Exercice. Visualisez quelques polynômes de Lagrange avec un logiciel graphique. (voir fig 3.4).

Propriétés 1 *L_j est le polynôme de degré n tel que $L_j(x_i) = 0$ pour $i \neq j$ et $L_j(x_j) = 1$:*

$$L_j(x_i) = \delta_{i,j}$$

Preuve. Il suffit de faire $X \leftarrow x_i$ dans la définition de L_j. ■

Proposition 1 *La famille $(L_j)_{0 \leq j \leq n}$ est une base de $\mathbb{R}_n[X]$.*

Preuve. Comme la famille contient exactement $n + 1$ éléments, il suffit de prouver qu'elle est libre. Soit une combinaison linéaire

$$\sum_j \lambda_j L_j(X) = 0$$

Faisant $X \leftarrow x_i$ dans cette égalité, on obtient un seul terme non nul :

$$\lambda_i = 0.$$

En faisant cela pour chaque i, on obtient que la combinaison linéaire est triviale. ■

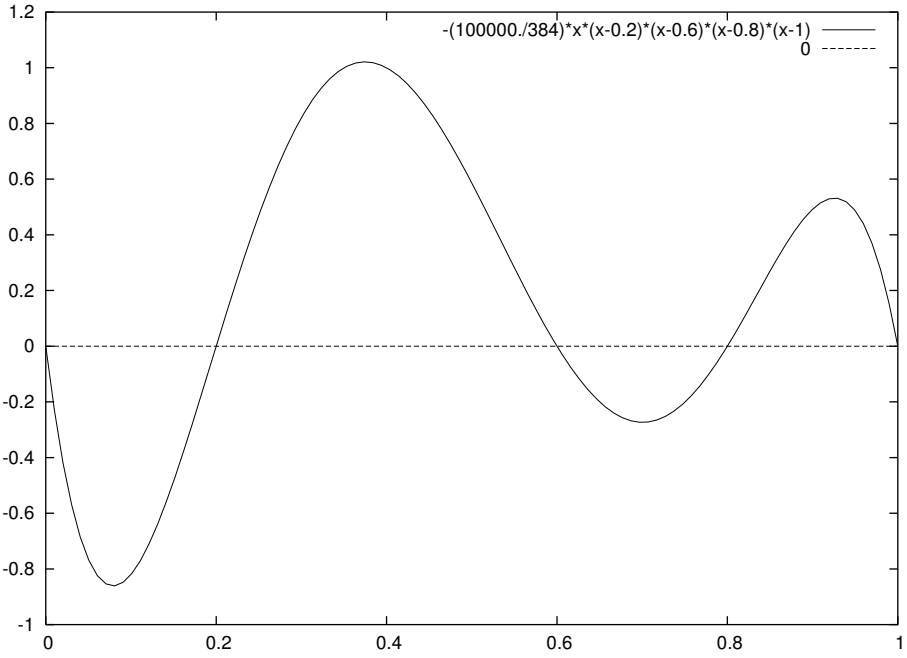


FIGURE 3.4 – Un Polynôme de Lagrange de degré 5.

Propriétés 2 (formule de Lagrange) *Etant donnés $n+1$ abscisses distinctes $x_0, x_1, \dots x_n$ et $n+1$ valeurs quelconques $y_0, y_1, \dots y_n$, le polynôme d'interpolation est donné par l'expression :*

$$P(X) = \sum_{0 \leq j \leq n} y_j L_j(X)$$

Preuve. Il suffit de remarquer que $\sum_j y_j L_j(x_i) = y_i L_i(x_i) = y_i$ Un seul terme de la somme est non nul ! ■

Exercice. Formule barycentrique. Soit $w_i = \frac{1}{\Pi_{j \neq i}(x_i - x_j)}$ et $\mu_i(t) = \frac{w_i}{t - x_i}$ Montrer que

$$P_n(t) = \frac{\sum_i y_i \mu_i(t)}{\sum_i \mu_i(t)}.$$

Cette formule est utile pour le calcul pratique (Cf TP). On remarquera que les coefficients $\mu_i(t)$ ne sont pas positifs en général et sont à recalculer pour chaque point t.

Exercice. Montrer que $\sum_i w_i = 0$ (Indication : interpoler la fonction constante 1 et considérer le coefficient dominant).

Les polynômes de Lagrange conduisent à des formules élégantes. Cependant la formule de Lagrange souffre d'un handicap : l'ajout d'un point nécessite de recalculer tous les polynômes. Il faut pour cela écrire le polynôme d'interpolation sous une autre forme. C'est l'objet du paragraphe suivant.

3.3 Polynômes de Newton.

Un base de l'espace $\mathbb{R}_n[X]$ des polynômes de degré inférieur ou égal à n particulièrement adaptée à l'interpolation est formée des polynôme de Newton :

$$1, (X - x_0), (X - x_0)(X - x_1), \dots, (X - x_0)(X - x_1)(X - x_2) \dots (X - x_{n-1})$$

Les polynômes $N_k(X) = \Pi_{j \leq k-1}(X - x_j)$ sont appelés polynômes de Newton. C'est une famille étagée, i.e. $\deg N_k = k$, donc c'est une base de $\mathbb{R}_n[X]$. De plus les racines de N_k sont exactement $x_0, x_1, \dots x_{k-1}$. Si l'on cherche le polynôme d'interpolation dans cette base,

$$P_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \dots c_n(x - x_0)(x - x_1) \dots (x - x_{n-1}),$$

les coefficients s'obtiennent en résolvant un système triangulaire beaucoup plus simple que le système linéaire de Vandermonde vu à la section 3.1.1 :

$$\left\{\begin{array}{lcl} c_0 & = & f(x_0) \\ c_0 + c_1(x_1 - x_0) & = & f(x_1) \\ \vdots & = & \vdots \\ c_0 + c_1(x_n - x_0) + \ldots + c_n(x_n - x_0)(x_n - x_1)\ldots(x_n - x_{n-1}) & = & f(x_n) \end{array}\right.$$

Les coefficients c_k se calculent donc facilement. $c_0 = f(x_0)$, $c_1 = \frac{f[x_1]-f[x_0]}{x_1-x_0}$,...
Le polynôme d'interpolation s'exprime dans cette base par une formule élégante attribuée à Newton.

Proposition 2

$$P_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \ldots f[x_0, x_1, \ldots, x_n](x - x_0)(x - x_1)\ldots(x - x_{n-1}). \tag{3.2}$$

où les $f[x_0, x_1, \ldots, x_k]$ appelées *différences divisées* sont définies par récurrence :

$$f[x_0] := f(x_0), \quad f[x_0, x_1] := \frac{f[x_1] - f[x_0]}{x_1 - x_0}$$

$$f[x_0, x_1, \ldots, x_k] := \frac{f[x_1, x_2, \ldots, x_k] - f[x_0, x_1, \ldots, x_{k-1}]}{x_k - x_0}$$

Preuve. La preuve repose sur le lemme d'interpolation suivant.

Lemme 1

$$P_k(x) = \frac{(x - x_0)P_{1,2,\ldots,k}(x) - (x - x_k)P_{0,1,\ldots,k-1}(x)}{x_k - x_0},$$

où $P_{1,2,\ldots,k}(x)$ (resp. $P_{0,1,\ldots,k-1}(x)$) désignent les polynômes d'interpolation aux points $x_1, x_2, \ldots, x_k$ (resp. $x_0, x_1, \ldots, x_{k-1}$.)

Preuve du lemme. Pour $x = x_1, x_2, \ldots, x_{k-1}$, les deux polynômes $P_{1,2,\ldots,k}(x)$ et $P_{0,1,\ldots,k-1}(x)$ interpolant f prennent la valeurs $f(x_i)$, $i = 1 \ldots k-1$. Donc

$$\frac{(x - x_0)P_{1,2,\ldots,k}(x) - (x - x_k)P_{0,1,\ldots,k-1}(x)}{x_k - x_0} = f(x) = P_k(x).$$

Pour $x = x_0$,

$$\frac{(x - x_0)P_{1,2,\ldots,k}(x) - (x - x_k)P_{0,1,\ldots,k-1}(x)}{x_k - x_0} = P_{0,1,\ldots,k-1}(x_0) = P_k(x_0).$$

Pour $x = x_k$,

$$\frac{(x - x_0)P_{1,2,\ldots,k}(x) - (x - x_k)P_{0,1,\ldots,k-1}(x)}{x_k - x_0} = P_{1,2,\ldots,k}(x_k) = P_k(x_k).$$

Ainsi $\frac{(x-x_0)P_{1,2,\ldots,k}(x)-(x-x_k)P_{0,1,\ldots,k-1}(x)}{x_k-x_0}$ interpole f aux points $x_0, x_1, \ldots, x_k$, or c'est un polynôme de degré au plus k donc c'est le polynôme P_k. □

La formule de Newton s'en déduit alors par récurrence sur n. Par hypothèse de récurrence on a

$$P_{0,1,2\ldots,n-1}(x) = f[x_0] + f[x_0, x_1](x - x_0) + \ldots f[x_0, x_1, \ldots, x_{n-1}](x - x_0)(x - x_1)\ldots(x - x_{n-2}).$$

Considérons le polynôme différence $Q(x) = P_{0,1,2\ldots,n}(x) - P_{0,1,2\ldots,n-1}(x)$. Le polynôme Q s'annule en $x_0, x_1, \ldots x_{n-1}$ puisque ce sont des noeuds d'interpolation de $P_{0,1,2\ldots,n}$ et $P_{0,1,2\ldots,n-1}$. D'autre part, le degré de Q est inférieur ou égal à n. Par conséquent, le polynôme Q est multiple de $(x - x_0)(x - x_1)\ldots(x - x_{n-1})$:

$$P_{0,1,2\ldots,n}(x) - P_{0,1,2\ldots,n-1}(x) = \lambda \cdot (x - x_0)(x - x_1)\ldots(x - x_{n-1}).$$

Identifions le scalaire λ. C'est le coefficient dominant (celui de x^n) dans le polynôme Q et, comme $P_{0,1,2\ldots,n-1}(x)$ est de degré inférieur ou égal à $n-1$, c'est aussi le coefficient dominant de $P_{0,1,2\ldots,n}$. D'après le lemme,

$$P_{0,1,2\ldots,n}(x) = \frac{(x - x_0)P_{1,2,\ldots,n}(x) - (x - x_n)P_{0,1,\ldots,n-1}(x)}{x_n - x_0},$$

Or par hypothèse de récurrence

$$P_{1,2,\dots,n}(x) = f[x_1] + f[x_1,x_2](x-x_1) + \dots f[x_1,x_2,\dots,x_n](x-x_1)(x-x_2)\dots(x-x_{n-1})$$

et

$$P_{0,1,2\dots,n-1}(x) = f[x_0] + f[x_0,x_1](x-x_0) + \dots f[x_0,x_1,\dots,x_{n-1}](x-x_0)(x-x_1)\dots(x-x_{n-2}).$$

Le terme dominant de $P_{0,1,2\dots,n}(x)$ s'obtient donc par combinaison des termes dominants :

$$\lambda = \frac{f[x_1,x_2,\dots,x_n] - f[x_0,x_1,\dots,x_{n-1}]}{x_n - x_0}$$

■

Remarque. L'intérêt des polynômes de Newton est que la formule de Newton se comporte bien lorsqu'on ajoute un point supplémentaire x_{n+1}. Il suffit de corriger la formule 3.2 en ajoutant le terme $f[x_0,x_1,\dots,x_{n+1}](x-x_0)(x-x_1)\dots(x-x_n)$. Ce terme s'annulant pour $x = x_0, x_1, \dots, x_n$, il ne sert qu'à corriger le polynôme pour qu'il interpole au point rajouté $x = x_{n+1}$ tout en conservant l'interpolation aux points déjà interpolés. □

Dans la pratique, on dispose les différences divisées sous forme de tableau, et on les calcule de proche en proche, par colonne.

$$\begin{array}{c|ccccc}
x_0 & f[x_0] & & & & \\
 & & f[x_0,x_1] & & & \\
x_1 & f[x_1] & & f[x_0,x_1,x_2] & & \\
 & & f[x_1,x_2] & & f[x_0,x_1,x_2,x_3] & \\
x_2 & f[x_2] & & f[x_1,x_2,x_3] & & f[x_0,x_1,x_2,x_3,x_4] \\
 & & f[x_2,x_3] & & f[x_1,x_2,x_3,x_4] & \\
x_3 & f[x_3] & & f[x_2,x_3,x_4] & & \\
 & & f[x_3,x_4] & & & \\
x_4 & f[x_4] & & & &
\end{array}$$

Remarque. Le coefficient $f[x_0,x_1,\dots,x_k]$ est le coefficient dominant du polynôme d'interpolation aux noeuds $x_0,x_1,\dots,x_k$. Or la formule (3.2) ne suppose pas que les x_j soient rangés par ordre croissant. On peut donc les *permuter* ! Le monôme $(x-x_0)(x-x_1)\dots(x-x_{n-1})$ étant invariant par permutation des noeuds, il résulte que $f[x_0,x_1,\dots,x_k]$ est également *invariant par permutation* des x_i. Ainsi on a également, par exemple,

$$f[x_0,x_1,\dots,x_k] := \frac{f[x_0,x_1,\dots,x_{k-2},x_k] - f[x_0,x_1,\dots,x_{k-1}]}{x_k - x_{k-1}}.$$

□

Exercice. Si f est de classe $\mathcal{C}^k$ montrer qu'il existe un réel ξ compris $\min(x_i)$ et $\max(x_i)$ tel que $f[x_0,x_1,\dots,x_k] = \frac{f^{(k)}(\xi)}{k!}$.
Réponse. Par la formule des accroissements finis généralisés aux points $x_0,x_1,\dots,x_{n-1}$, on a :

$$f(x) = P_{n-1}(x) + \frac{f^{(n)}(\xi_x)}{(n)!}(x-x_0)(x-x_1)\dots(x-x_{n-1})$$

où ξ_x est un point [2]compris entre $\min(x_i)$ et $\max(x_i)$. D'autre part, par la formule de Newton (3.2) aux points $x_0,x_1,\dots,x_n$ on a :

$$P_n(x) = P_{n-1}(x) + f[x_0,x_1,\dots,x_n](x-x_0)(x-x_1)\dots(x-x_{n-1}).$$

2. dépendant évidemment de x

En prenant $x = x_n$, comme $f(x_n) = P_n(x_n)$ on peut égaler les expressions

$$\frac{f^{(n)}(\xi_x)}{(n)!}(x_n - x_0)(x_n - x_1)\dots(x_n - x_{n-1}) = f[x_0, x_1, \dots, x_n](x_n - x_0)(x_n - x_1)\dots(x_n - x_{n-1}).$$

Puis, en simplifiant par $(x_n - x_0)(x_n - x_1)\dots(x_n - x_{n-1})$ qui est non nul, car les noeuds sont supposés distincts, on tire $f[x_0, x_1, \dots, x_n] = \frac{f^(n)(\xi)}{n!}$. □

Dans la formule de Newton 3.2, on peut faire *coïncider* des points : Si par exemple $x_0 = x_1$, x_0 est un point "double" la différence divisée devient

$$f[x_0, x_0] = \lim_{x_1 \to x_0} \frac{f[x_1] - f[x_0]}{x_1 - x_0} = f'(x_0)$$

Le polynôme d'interpolation vérifie donc en x_0 $P(x_0) = f(x_0)$ ET $P'(x_0) = f'(x_0)$. On peut faire évidemment coïncider plusieurs points (points triples, pour faire coïncider les dérivées secondes,etc.... On procède de même par passage à la limite. Par exemple, calculons $f[x_i, x_i, x_i]$. Pour cela calculons $f[x_i, x_i + h, x_i + 2h]$ et prenons la limite quand $h \to 0$.

$$f[x_i, x_i + h, x_i + 2h] = \frac{f[x_i + h, x_i + 2h] - f[x_i, x_i + h]}{2h} = \frac{f(x_i + 2h) - 2f(x_i + h) + f(x_i)}{2h^2}.$$

Maintenant $\lim_{h\to 0} \frac{f(x_i+2h)-2f(x_i+h)+f(x_i)}{2h^2} = \frac{f''(x_i)}{2}$. (Il suffit d'effectuer une développement de Taylor de f à l'ordre 2 au point x_i.) Ainsi $f[x_i, x_i, x_i] = \frac{f''(x_i)}{2}$ et plus généralement l'exercice précédent montre que pour un noeud multiple d'ordre $k + 1$, $f[x, \dots, x] = \frac{f^{(k)}(x)}{k!}$.

Lorsqu'on veut interpoler une fonction f par un polynôme en faisant coïncider les derivées premières, seconde, ..., on parle *d'interpolation de Hermite*[3]. Plus précisément, on a le résultat.

Théorème 6 *Supposons qu'on se donne* $(n + 1)$ *noeuds éventuellement répétés* avec leur multiplicité, *c'est à dire* $(x_0, m_0), \dots (x_k, m_k)$ *de sorte que* $m_0 + m_1 + \dots m_k = n + 1$. *Il existe un unique polynôme* P *de degré inférieur ou égal à* n *tel que* $P(x_i) = f(x_i), \quad P'(x_i) = f'(x_i), \dots P^{(m_i-1)}(x_i) = f^{(m_i-1)}(x_i), \quad i = 0, \dots, k.$

Preuve. Notons $\mathbb{R}_n[X] = \left\{\sum_{k=0}^n a_k X^k, a_k \in \mathbb{R}\right\}$ l'espace vectoriel des polynômes de degré inférieur ou égal à n. Soit l'application linéaire :

$$\begin{aligned} F : \mathbb{R}_n[X] &\to \mathbb{R}^{n+1} \\ P &\mapsto (P(x_0), P'(x_0), \dots, P^{(m_0-1)}(x_0), \dots, P(x_k), P'(x_k), \dots, P^{(m_k-1)}(x_k)) \end{aligned}$$

Le noyau de F se réduit au polynôme nul $P = 0$. En effet si un polynôme de degré inférieur ou égal à n a $n + 1$ racines comptées avec leur multiplicité, c'est le polynôme nul. L'application F est donc injective. D'autre part la dimension de $\mathbb{R}_n[X]$ est $n + 1$, donc F est un isomorphisme. ■

Voyons cela sur un exemple. Supposons que nous ayons deux noeuds triples $x_0, x_0, x_0, x_1, x_1, x_1$. Les différences divisées se calculent ainsi :

$$\begin{array}{c|ccccccc}
x_0 & f[x_0] & & & & & & \\
 & & f'(x_0) & & & & & \\
x_0 & f[x_0] & & \frac{1}{2}f''(x_0) & & & & \\
 & & f'(x_0) & & f[x_0,x_0,x_0,x_1] = c_3 & & & \\
x_0 & f[x_0] & & f[x_0,x_0,x_1] & & f[x_0,x_0,x_0,x_1,x_1] = c_4 & & \\
 & & f[x_0,x_1] & & f[x_0,x_0,x_1,x_1] & & f[x_0,x_0,x_0,x_1,x_1,x_1] = c_5 & \\
x_1 & f[x_1] & & f[x_0,x_1,x_1] & & f[x_0,x_0,x_1,x_1,x_1] & & \\
 & & f'(x_1) & & f[x_0,x_1,x_1,x_1] & & & \\
x_1 & f[x_1] & & \frac{1}{2}f''(x_1) & & & & \\
 & & f'(x_1) & & & & & \\
x_1 & f[x_1] & & & & & &
\end{array}$$

3. Charles Hermite, 1822-1901, mathématicien français.

Le polynôme d'interpolation de Newton est donné par

$$P_5(x) = f[x_0] + f'(x_0)(x-x_0) + \frac{1}{2}f''(x_0)(x-x_0)^2 + c_3(x-x_0)^3 + c_4(x-x_0)^3(x-x_1) + c_5(x-x_0)^3(x-x_1)^2$$

On peut vérifier qu'il a les propriétés suivantes :

$$\begin{array}{lll} P_5(x_0) = f(x_0) & P_5'(x_0) = f'(x_0) & P_5''(x_0) = f''(x_0) \\ P_5(x_1) = f(x_1) & P_5'(x_1) = f'(x_1) & P_5''(x_1) = f''(x_1) \end{array}$$

Remarque. Un cas particulier instructif. Prenons tous les x_i confondus et égaux à x_0 et réalisons l'interpolation de Hermite suivante : on cherche un polynôme de degré n tel que

$$P^{(j)}(x_0) = f^{(j)}(x_0) \quad j = 0 \dots n.$$

On sait déjà que

$$P(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2}(x-x_0)^2 + \dots \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n$$

On peut ainsi voir la formule de Newton comme une généralisation de la formule de Taylor. □

3.4 (⋆) Splines cubiques naturelles.

3.4.1 Spline = tige souple

Pour des raisons pratiques, on aimerait interpoler une fonction avec une méthode
— qui soit flexible, en ce sens qu'elle gère facilement l'ajout d'un point supplémentaire,
— qui ne génère pas d'oscillations du type phénomène de Runge aux extrémités.

Un méthode, relativement récente (les années 1970) et maintenant universellement adoptée, est celle des splines cubiques, que nous présenterons succinctement. La figure 3.5 illustre l'efficacité de la méthode. On a interpolé la fonction $t \mapsto 1/(1+t^2)$ aux points $-5, -3, -1, 0, 1, 3, 5$ avec le polynôme d'interpolation de degré 6 et aussi avec une fonction spline cubique (calculée avec MAPLE).

$$s := \begin{cases} -\frac{31}{1040} - \frac{621}{5200}\,t - \frac{33}{1040}\,t^2 - \frac{11}{5200}\,t^3, & t < -3, \\ \frac{7567}{5200} + \frac{7101}{5200}\,t + \frac{2409}{5200}\,t^2 + \frac{11}{208}\,t^3, & -3 \le t < -1, \\ 1 - \frac{1173}{1300}\,t^2 - \frac{523}{1300}\,t^3, & -1 \le t < 0, \\ 1 - \frac{1173}{1300}\,t^2 + \frac{523}{1300}\,t^3, & 0 \le t < 1, \\ \frac{7567}{5200} - \frac{7101}{5200}\,t + \frac{2409}{5200}\,t^2 - \frac{11}{208}\,t^3, & 1 \le t < 3, \\ -\frac{31}{1040} + \frac{621}{5200}\,t - \frac{33}{1040}\,t^2 + \frac{11}{5200}\,t^3, & t \ge 3 \end{cases}$$

Les fonctions spline ne sont plus des polynômes, elles sont seulement polynomiale *par morceaux.* Elles sont caractérisées par les trois propriétés

1. polynomiales par morceaux, de degré trois (d'où le terme cubique) sur chaque (x_i, x_{i+1})
2. elles sont de classe $\mathcal{C}^2$. En chaque x_i, les dérivées premières et secondes à gauche et à droite coïncident.
3. la dérivée seconde s'annule aux deux extrémités x_0 et x_n.

Les fonctions spline sont en fait solution d'un problème mécanique simple : on imagine une tige élastique ou tringle souple [4] passant par les points $(x_i, f(x_i))$. La forme $x \mapsto s(x)$ adoptée par la tringle doit minimiser l'énergie potentielle (énergie élastique de déformation) qui s'exprime (voir cours d'élasticité) :

$$J = \frac{1}{2}\int_{x_0}^{x_n} s''(x)^2\,dx$$

On peut démontrer que la fonction spline cubique s minimise $J(s) = \frac{1}{2}\int_{x_0}^{x_n} s''(x)^2\,dx$ parmi toutes les fonctions f vérifiant les propriétés

4. c'est la traduction de l'anglais "spline"

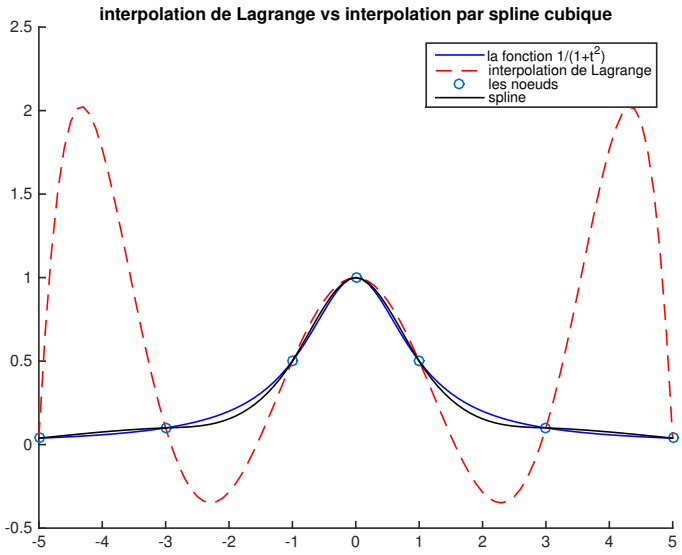


FIGURE 3.5 – Spline cubique vs interpolation de Lagrange

(i) $f(x_i) = y_i, \quad i = 0, \ldots, n$

(ii) f est de classe $\mathcal{C}^2$ sur $[x_0, x_n]$

Preuve. Donnons une preuve succincte. Tout repose sur la propriété d'« orthogonalité »

$$\int_{x_0}^{x_n} (f"(x) - s"(x)) \cdot s"(x)\, dx = 0. \tag{3.3}$$

En effet une intégration par partie donne

$$\int (f"(x) - s"(x)) \cdot s"(x)\, dx = -\int (f'(x) - s'(x)) \cdot s'''(x)\, dx$$

Il n'y a pas de terme de bord car $s"$ s'annule aux extrémités x_0 et x_n. Ensuite comme s est de degré trois, s''' est constant par morceaux.

$$\int (f(x) - s'(x)) \cdot s'''(x)\, dx = \sum_i K_i \int_{x_i}^{x_{i+1}} (f'(x) - s'(x))\, dx = \sum_i K_i [f(x) - s(x)]_{x_i}^{x_{i+1}} = 0$$

car f et s sont égales aux noeuds x_i. Cela permet ensuite de prouver

$$\int f"(x)^2\, dx = \int (f"(x) - s"(x))^2\, dx + \int s"(x)^2\, dx \geq \int s"(x)^2\, dx$$

En effet

$$\int (f"(x) - s"(x))^2\, dx = \int f"(x)^2\, dx - 2\int f"(x) \cdot s"(x)\, dx + \int s"(x)^2\, dx = \int f"(x)^2\, dx - \int s"(x)^2\, dx$$

car par la propriété (3.3) $\int f"(x) \cdot s"(x)\, dx = \int s"(x)^2\, dx$ ■

3.4.2 Calcul des splines cubiques

A partir des propriétés 1, 2, 3 il est facile de calculer la fonction s. Soit $[x_i, x_{i+1}]$ un sous-intervalle. Posons $h_i = x_{i+1} - x_i$ et cherchons la restriction de s à cet intervalle sous la forme :

$$s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i.$$

On obtient :

$$\begin{array}{lcccl}
s_i(x_i) & = & d_i & = & y_i \\
s_i(x_{i+1}) & = & a_i h_i^3 + b_i h_i^2 + c_i h_i + d_i & = & y_{i+1} \\
s_i'(x_i) & = & c_i & & \\
s_i'(x_{i+1}) & = & 3a_i h_i^2 + 2b_i h_i + c_i & & \\
s_i''(x_i) & = & 2b_i & = & y_i'' \\
s_i''(x_{i+1}) & = & 6a_i h_i + 2b_i & = & y_{i+1}''
\end{array}$$

d'où l'on tire (exercice élémentaire) :

$$\begin{array}{rcl} a_i & = & \frac{1}{6h_i}(y''_{i+1} - y''_i) \\ b_i & = & \frac{1}{2}y''_i \\ c_i & = & \frac{1}{h_i}(y_{i+1} - y_i) - \frac{h_i}{6}(y''_{i+1} + 2y''_i) \\ d_i & = & y_i \end{array}$$

On voit donc que les valeurs des dérivées secondes y''_k (en plus des valeurs y_k de la fonction à interpoler) déterminent complètement la spline. Pour les déterminer, nous allons utiliser la continuité de la dérivée première. On obtient

$$s'_i(x_{i+1}) = \frac{1}{h_i}(y_{i+1} - y_i) + \frac{h_i}{6}(2y''_{i+1} + y''_i)$$

En faisant $i \leftarrow i-1$, On obtient la condition de continuité $s'_{i-1}(x_i) = s'_i(x_i)$

$$\frac{1}{h_{i-1}}(y_i - y_{i-1}) + \frac{h_{i-1}}{6}(2y''_i + y''_{i-1}) = \frac{1}{h_i}(y_{i+1} - y_i) - \frac{h_i}{6}(y''_{i+1} + 2y''_i)$$

En exprimant cette condition en $x_1, x_2, \ldots x_{n-1}$ on obtient un système linéaire en les inconnues $y''_1, y''_2, \ldots y''_{n-1}$ dont la i^{e} ligne est

$$h_{i-1}y''_{i-1} + 2(h_{i-1} + h_i)y''_i + h_i y''_{i+1} = \frac{6}{h_i}(y_{i+1} - y_i) - \frac{6}{h_{i-1}}(y_i - y_{i-1})$$

Exercice. Assembler la matrice A du système obtenu et montrer qu'elle est symétrique, tridiagonale, et à diagonale dominante. En déduire l'existence et l'unicité de la spline cubique.
On peut enfin prouver que A est bien conditionnée :

Exercice. Montrer que la plus grande (resp. petite) valeurs propre de A vérifie

$$\lambda_{max} \leq \max_i\{2h_0 + 3h_1, 3(h_i + h_{i+1}), 3h_{n-2} + 2h_{n-1}\}$$

$$\lambda_{min} \geq \min_i\{2h_0 + h_1, h_i + h_{i+1}, h_{n-2} + 2h_{n-1}\}$$

On peut donc calculer le conditionnement de A :

$$\kappa(A) = \frac{\lambda_{max}}{\lambda_{min}} \leq C^{te}\frac{\max h_i}{\min h_i}$$

Le calcul numérique de la spline cubique est donc rapide, fiable et précis.

Mentionnons pour conclure que d'autres types de splines existent :

— on peut relaxer la condition de nullité des dérivées secondes aux extrémités, cela est intéressant si la fonction à interpoler présente une grande courbure aux extrémités.

— on peut monter en degré, mais on constate que les splines de degré supérier oscillent davantage. On préfère utiliser les splines cubiques pour l'interpolation.

Chapitre 4

Quadrature.

4.1 Méthodes élémentaires

4.1.1 Principe d'intégration numérique : poids et noeuds.

Soit f une fonction continue sur un intevalle $[a, b]$. Le but de ce chapitre est de donner des méthodes de calcul approché de

$$\int_a^b f(x)\,dx = I(f).$$

Bien sûr, lorsqu'on connaît une primitive F de f, la valeur exacte est

$$I(f) = F(b) - F(a).$$

Cependant, ce cas est très rare ...

Remarque. le nombre $I(f)$ s'interprète comme l'aire sous la courbe représentative de f, d'où le nom *quadrature*, de quarrer, en ancien français, calculer une aire. □

Commençons par rappeler les méthodes élémentaires.

4.1.2 Rectangle, Trapèze, Simpson

Si on interpole $y = f(x)$ par la fonction constante $y = f((a+b)/2)$ i.e. si on calcule l'aire du rectangle de hauteur $f((a+b)/2)$, on obtient l'approximation

$$I(f) \approx (b-a) \cdot f(\frac{a+b}{2}).$$

C'est la *méthode du point milieu.*

Si on interpole $y = f(x)$ par la fonction affine $y = f(a) + (x-a)\frac{f(b)-f(a)}{b-a}$ prenant les mêmes valeurs aux deux extrémités a et b, i.e. si on calcule l'aire du trapèze, on obtient l'approximation

$$I(f) \approx (b-a) \cdot \frac{f(a)+f(b)}{2}.$$

C'est la *méthode du trapèze.*

Si on interpole $y = f(x)$ par le polynôme du second degré passant par $(a, f(a))$, $(\frac{a+b}{2}, f(\frac{a+b}{2}))$, $(b, f(b))$ i.e. si on calcule l'aire sous la parabole, on obtient l'approximation

$$I(f) \approx (b-a) \cdot \{\frac{1}{6}f(a) + \frac{4}{6}f(\frac{a+b}{2}) + \frac{1}{6}f(b)\}.$$

C'est la *méthode de Simpson.*

Si on interpole $y = f(x)$ par le polynôme du troisième degré passant par $(x_j, f(x_j))$ où $x_j = a + j \cdot \frac{b-a}{3}$, $j = 0 \ldots 3$ on obtient l'approximation

$$I(f) \approx (b-a) \cdot \{\frac{1}{8}f(a) + \frac{3}{8}f(\frac{2a+b}{3}) + \frac{3}{8}f(\frac{a+2b}{3}) + \frac{1}{8}f(b)\}.$$

C'est la *Règle des 3/8 de Newton.*
Donnons pour terminer ce premier panorama une formule plus sophistiquée. Si on interpole $y = f(x)$ par le polynôme du quatrième degré passant par $(x_j, f(x_j))$ où $x_j = a+j\cdot\frac{b-a}{4},\ j = 0\dots 4$ on obtient l'approximation

$$I(f) \approx (b-a)\cdot\{\frac{7}{90}f(a) + \frac{32}{90}f(\frac{3a+b}{4}) + \frac{12}{90}f(\frac{a+b}{2}) + \frac{32}{90}f(\frac{a+3b}{4}) + \frac{7}{90}f(b)\}.$$

C'est la *méthode de Boole-Villarceau.*
Plus généralement il y a un procédé général pour calculer les coefficients des formules précédentes, appelées formule de Newton-Cotes.

Lemme 2 *Soit $a \le x_1 < x_2 < \dots x_n \le b$. Il existe un unique n-uplet $w_1, w_2, \dots w_n$ tel que pour tout f polynôme de degré $\le n-1$*

$$\int_a^b f(x)\,dx = \sum_k w_k f(x_k).$$

Le poids est donné par $w_j := \int_a^b L_j(x)\,dx$ où L_j est le j-ème polynôme de Lagrange.

Preuve. Il suffit de prendre $f := L_j$ pour obtenir que nécessairement $w_j = \int_a^b L_j(x)\,dx$ où L_j est le j-ième polynôme de Lagrange. Ensuite comme f est de degré inférieur ou égal à $n-1$, il est son propre polynôme d'interpolation $f = \sum_k f(x_k)L_k$ d'où le résultat en intégrant par rapport à x sur $[a,b]$. ■

En conclusion, pour donner une valeur approchée de $I(f)$, on utilise donc un tableau de valeurs de f, i.e. la donnée de $f(x_j)$ en certains points x_j appelés *noeuds* et on calcule ensuite une moyenne pondérée de ces valeurs, le coefficient affecté à chaque valeur étant évidemment appelé *poids*. La règle de quadrature a toujours la forme suivante :

$$Q(f) = (b-a)\cdot\sum_j \omega_j f(x_j)$$

Faisons maintenant quelques remarques importantes.
Remarque. La quantité

$$\frac{\int_a^b f(x)\,dx}{b-a} = \frac{\int_a^b f(x)\,dx}{\int_a^b dx}$$

correspond à la moyenne [1] de f sur (a,b). Or toutes les formules précédentes sont du type

$$\frac{\int_a^b f(x)\,dx}{b-a} \approx \sum_j \omega_j f(x_j)$$

avec les poids ω_j positifs et tels que $\sum_j \omega_j = 1$. Ainsi on a remplacé la moyenne *continue* $\frac{\int_a^b f(x)\,dx}{b-a}$ par une moyenne *discrète* $\sum_j \omega_j f(x_j)$. [2] □

Remarque. L'expression $f \mapsto Q(f) = (b-a)\cdot\sum_j \omega_j f(x_j)$ est une forme linéaire, comme l'était $f \mapsto I(f) = \int_a^b f(x)\,dx$. □

Remarque. L'expression $f \mapsto I(f) = \int_a^b f(x)\,dx$ étant positive il est capital que $f \mapsto Q(f) = (b-a)\cdot\sum_j \omega_j f(x_j)$ le soit également. [3] Cela impose que les ω_j soient *positifs*. Nous verrons que l'interpolation par des polynômes de degré supérieur ou égal à huit conduit à certains poids négatifs, ce qui fait que les formules de quadrature correspondantes sont dangereuses et inutilisées. □

Remarque. Les formules précédentes possèdent une symétrie intéressante : elles sont invariantes par le changement $x \leftarrow a+b-x$, i.e. la symétrie par rapport au point milieu du segment, qui échange a en b. les poids ω_j respectent cette symétrie. Ainsi $\check{f} : x \mapsto f(a+b-x)$ et $f : x \mapsto f(x)$ donnent la même intégrale approchée $Q(f) = Q(\check{f})$. Evidemment on a également $I(f) = I(\check{f})$(calcul par changement de variable.) □

1. ou aussi à l'espérance de $f(X)$ lorsque X suit la loi uniforme sur (a,b)
2. qui correspond à l'espérance de $f(X)$ lorsque X suit la loi discrète $\sum_j \omega_j \delta_{x_j}$, autrement dit $P(X = x_j) = \omega_j$.
3. sans quoi on pourrait obtenir une intégrale approchée négative pour certaines fonctions positives !

On voit ainsi que $Q(f)$ conserve les principales propriétés de $I(f)$. C'est une qualité extrêmement agréable de la quadrature numérique.

Remarque. Une propriété de l'intégrale sera cependant perdue irrémédiablement : la stricte positivité. Prenons par exemple la fonction $f : x \mapsto \Pi_j (x - x_j)^2$. On a évidemment $I(f) > 0$ alors que $Q(f) = 0$ □

4.1.3 Ordre et Formules d'erreur

Proposition 3 *La méthode du point milieu est exacte pour f polynôme de degré inférieur ou égal à un. La méthode du trapèze est exacte pour f polynôme de degré inférieur ou égal à un. La méthode de Simpson est exacte pour f polynôme de degré inférieur ou égal à trois. La méthode de Newton est exacte pour f polynôme de degré inférieur ou égal à trois. La méthode de Villarceau est exacte pour f polynôme de degré inférieur ou égal à cinq.*

Preuve. la preuve repose sur la linéarité. Pour alléger les calculs on se ramène aussi à l'intervalle $[-1, 1]$ par changement de variable $t \mapsto x = \frac{a+b}{2} + t\frac{(b-a)}{2}$ qui applique $[-1, 1]$ sur $[a, b]$. L'intégrale se transforme par changement de variable :

$$\int_a^b f(x)\,dx = \frac{(b-a)}{2} \int_{-1}^1 \varphi(t)\,dt$$

où

$$\varphi(t) := f(x) = f(\frac{a+b}{2} + t\frac{(b-a)}{2})$$

Il suffit alors de prouver que

$$\int_{-1}^1 \varphi(t)\,dt = 2 \cdot \sum_j \omega_j \varphi(x_j)$$

pour $\varphi(t) = 1, t, t^2, \ldots$. Pour la constante $\varphi(t) = 1$, l'égalité est équivalente à la somme des poids valant 1.

$$\sum_j \omega_j) = 1.$$

Les autres calculs sont facilités par l'(im)parité. Pour la méthode du point milieu, par imparité $I(f) = Q(f) = 0$ pour $\varphi(t) = t$. Pour Simpson, par imparité $I(f) = Q(f) = 0$ pour $\varphi(t) = t^3$. Pour Villarceau, lorsque $\varphi(t) = t^5$ on trouve $I(f) = Q(f) = 0$, ainsi il suffit de vérifier l'exactitude pour t^2, t^4. On voit aussi que l'imparité permet de gagner un degré, on a intérêt à utiliser des formules avec un nombre *impair* de noeuds. ■

Remarque. On voit aussi que l'imparité permet de gagner un degré, on a intérêt à utiliser des formules avec un nombre *impair* de noeuds. □

Cela motive la définition suivante.

Définition 1 *On dit que le degré de précision d'une quadrature est m si elle est exacte pour tous les polynômes de degré inférieur ou égal à m, mais pas pour tous les polynômes de degré m + 1. (On dit aussi que la quadrature est* d'ordre $m + 1$. [4]*)*

4. ce terme et ce décalage se justifieront dans le chapitre équations différentielles.

Propriétés 3 *On a les formules d'erreur suivantes :*

$$\int_a^b f(x)\,dx = (b-a)\cdot f(\frac{a+b}{2}) + \frac{(b-a)^3}{24}f''(\xi), \tag{4.1}$$

$$\int_a^b f(x)\,dx = (b-a)\cdot \frac{f(a)+f(b)}{2} - \frac{(b-a)^3}{12}f''(\xi), \tag{4.2}$$

$$\int_a^b f(x)\,dx = (b-a)\cdot \{\frac{1}{6}f(a) + \frac{4}{6}f(\frac{a+b}{2}) + \frac{1}{6}f(b)\} + \frac{(b-a)^5}{2880}f^{(4)}(\xi), \tag{4.3}$$

$$\int_a^b f(x)\,dx = (b-a)\cdot \{\frac{1}{8}f(a) + \frac{3}{8}f(\frac{2a+b}{3}) + \frac{3}{8}f(\frac{a+2b}{3}) + \frac{1}{8}f(b)\} - \frac{(b-a)^5}{6480}f^{(4)}(\xi), \tag{4.4}$$

$$\int_a^b f(x)\,dx = (b-a)\cdot \{\frac{7}{90}f(a) + \frac{32}{90}f(\frac{3a+b}{4}) + \frac{12}{90}f(\frac{a+b}{2}) + \frac{32}{90}f(\frac{a+3b}{4}) + \frac{7}{90}f(b)\} + \frac{(b-a)^7}{1935360}f^{(6)}(\xi), \tag{4.5}$$

pour f de classe $\mathcal{C}^2$ (resp. $\mathcal{C}^4, \mathcal{C}^6$) sur $[a,b]$ et $\xi \in [a,b]$ (évidemment dépendant de la formule utilisée).

Preuve. Les preuves s'appuient par exemple sur la formule de Taylor avec reste intégral (qui n'est rien d'autre que la formule d'intégration par partie appliquée en rafale) et le lemme de la moyenne que nous rappelons.

Lemme 3 (de la moyenne) *Soit $g(x)$ une fonction intégrable* positive *sur $[a,b]$. Soit f continue sur $[a,b]$. Alors il existe $\xi \in [a,b]$ tel que*

$$\int_a^b f(x)\cdot g(x)dx = f(\xi)\int_a^b g(x)dx$$

Démontrons par exemple la formule des trapèzes 4.2. Par une intégration par parties,

$$\int_a^b f(x)\,dx = -\int_a^b (x-c)f'(x)\,dx + [(x-c)f(x)]_a^b.$$

Pour raison de symétrie, il est judicieux de choisir $c := \frac{a+b}{2}$

$$\int_a^b f(x)\,dx = -\int_a^b (x-\frac{a+b}{2})f'(x)\,dx + \frac{f(a)+f(b)}{2}\,(b-a).$$

En intégrant à nouveau par parties :

$$\int_a^b f(x)\,dx = \int_a^b \frac{(x-a)(x-b)}{2}f''(x)\,dx + \frac{f(a)+f(b)}{2}\,(b-a).$$

L'erreur $I(f) - Q(f)$ est donc exactement

$$\int_a^b \frac{(x-a)(x-b)}{2}f''(x)\,dx = -\int_a^b \frac{(x-a)(b-x)}{2}f''(x)\,dx$$

Le lemme de la moyenne s'applique car $g(x) := \frac{(x-a)(b-x)}{2} \geq 0$. On obtient alors

$$I(f) - Q(f) = -f''(\xi)\int_a^b g(x)dx$$

Or $g(x)$ est un polynôme de degré 2, on peut utiliser Simpson qui est exacte pour calculer $\int_a^b g(x)dx = (b-a)\{1/6\times 0 + 4/6\times(b-a)^2/8 + 1/6\times 0\} = \frac{(b-a)^3}{12}$. Les autres formules peuvent se démontrer de manière

analogue. On peut aussi utiliser le théorème des noyaux de Peano, pour une preuve systématique, voir devoir maison.

■

On voit dans ce résultat que la précision des formules de quadrature croît avec les nombre de noeuds, et qu'on a intérêt à prendre des nombres de noeuds impairs. Cependant, les termes d'erreurs font intervenir les dérivées d'ordre élevé de la fonction f à intégrer.

— Si la fonction est peu régulière, on ne doit utiliser que les formules les plus simples.
— Si la fonction oscille beaucoup, il y a un risque que les dérivées d'ordre élevé soient très grandes, penser par exemple à $\sin(nx)$. Là aussi il faut éviter d'utiliser les formules d'ordre élevé.

Les formules d'ordre élevé ne présentent un intérêt que si l'intégrand f est très régulier. De plus, les problèmes liés à l'interpolation d'ordre élevé en des points équidistants constatés au chapitre précédent (oscillation de grande amplitude aux extrémités) se traduisent ici par l'apparition de poids négatifs, à partir de $n = 8$. On n'utilise donc ces méthodes que jusqu'au degré 7.

Dans tous les cas, il faut faire attention à la largeur de l'intervalle $(b-a)$ qui doit être toujours inférieure à 1, à cause du terme $(b-a)^n$, c'est pour cette raison qu'on préfère utiliser les méthodes composées qui font l'objet de la section suivante.

4.2 Méthodes à pas multiples

4.2.1 Principe des méthodes composées.

On subdivise l'intervalle $[a, b]$ en n morceaux : $a = x_0 < x_1 < \ldots < x_n = b$. Lorsque le pas de la subdivision est constant [5] : $x_j = a + j \cdot (b-a)/n \quad j = 0, \ldots, n$. La relation de Chasles donne :

$$\int_a^b f(x)dx = \sum_j \int_{x_j}^{x_{j+1}} f(x)dx.$$

Puis on applique une quadrature sur chaque intervalle de la subdivision. Pour la méthode des trapèzes, on obtient ainsi :

$$\int_a^b f(x)dx = \sum_j \left\{ \frac{(b-a)}{n} \cdot \frac{f(x_j) + f(x_{j+1})}{2} - \frac{(b-a)^3}{12n^3} f''(\xi_j) \right\}$$

En remarquant que, hormis les extrémités a et b, chaque point x_j apparaît dans deux termes

$$\int_a^b f(x)dx = \frac{(b-a)}{n} \left\{ \frac{f(a)}{2} + \sum_{a<xj<b} f(x_j) + \frac{f(b)}{2} \right\} - \frac{(b-a)^3}{12n^2} \left\{ \frac{\sum_j f''(\xi_j)}{n} \right\}$$

Le terme $\frac{\sum_j f''(\xi_j)}{n}$ peut s'exprimer comme une moyenne de valeurs de f''. Si f est supposée de classe $\mathcal{C}^2$, on peut utiliser une forme discrète du lemme de la moyenne, très facile à prouver $\frac{\sum_j f''(\xi_j)}{n} = f''(\xi)$ avec $\xi \in (a, b)$. On obtient ainsi :

$$\int_a^b f(x)dx = \frac{(b-a)}{n} \left\{ \frac{f(a)}{2} + \sum_{a<xj<b} f(x_j) + \frac{f(b)}{2} \right\} - \frac{(b-a)^3}{12n^2} f''(\xi).$$

Notons

$$T_n(f) = \frac{(b-a)}{n} \left\{ \frac{f(a)}{2} + \sum_{a<xj<b} f(x_j) + \frac{f(b)}{2} \right\}$$

$$T_n(f) = (b-a) \left\{ \frac{f(a)}{2n} + \sum_{a<xj<b} \frac{f(x_j)}{n} + \frac{f(b)}{2n} \right\}$$

5. On peut aussi prendre un pas adapté à la fonction, plus fin là où elle oscille beaucoup et plus grossier là où elle varie peu. Ce sont les méthodes adaptatives dont l'étude dépasse la portée de ce modeste fascicule.

On remarque que $T_n(f)$ est (encore) une moyenne des valeurs de f aux points x_j, où les extrémités a et b sont affectées d'un poids deux fois moindre que les points intérieurs. Finalement

$$\int_a^b f(x)dx = T_n(f) - \frac{(b-a)^3}{12n^2} f''(\xi).$$

Plus simplement on retiendra

$$\int_a^b f(x)dx = T_n(f) + 0(\frac{1}{n^2})$$

Le même principe, appliqué à la méthode des rectangles donne :

$$R_n(f) = \frac{b-a}{n} \left\{ \sum_j f(x_{j+1/2}) \right\}$$

Les points $x_{j+1/2} = \frac{x_j + x_{j+1}}{2}$ correspondent aux milieux des segments (x_j, x_{j+1}). Remarquer que $R_n(f)$ est la moyenne arithmétique des $f(x_{j+1/2})$. On a également l'approximation du même ordre :

$$\int_a^b f(x)dx = R_n(f) + 0(\frac{1}{n^2})$$

Pour la méthode de Simpson :

$$S_n(f) = \frac{(b-a)}{n} \left\{ \sum_j \frac{1}{6} f(x_j) + \frac{4}{6} f(x_{j+1/2}) + \frac{1}{6} f(x_{j+1}) \right\}$$

En regroupant les poids par noeud de chaque type :

$$S_n(f) = (b-a) \left\{ \frac{1}{6n} f(a) + \sum_{0<j<n} \frac{2}{6n} f(x_j) + \sum_j \frac{4}{6n} f(x_{j+1/2}) + \frac{1}{6n} f(b) \right\}$$

Il s'agit évidemment toujours d'une formule de moyenne car la somme des coefficients est : $1/6n + 2(n-1)/6n + 4n/6n + 1/6n = 1$. En utilisant à nouveau le lemme de la moyenne, on prouve la formule d'erreur :

$$\int_a^b f(x)dx = S_n(f) - \frac{(b-a)^5}{2880\, n^4} f^{(4)}(\xi).$$

Plus simplement on retiendra

$$\int_a^b f(x)dx = S_n(f) + 0(\frac{1}{n^4}).$$

4.2.2 Accélération de la convergence. Extrapolation de Romberg-Richardson.

Le principe de l'accéleration de la convergence est simple et peut s'appliquer àtoute suite dont la vitesse de convergence est contrôlée explicitement. Voyons cela sur un exemple, celui du calcul approché d'intégrales par la méthodes des trapèzes. Notons

$$I = \int_a^b f(x)\, dx$$

On peut démontrer (formule d'Euler-Maclaurin voir TP 4) que

$$T_n = I + \frac{\alpha}{n^2} + 0(\frac{1}{n^4})$$

Si on double le nombre de noeuds la précision est alors

$$T_{2n} = I + \frac{\alpha}{4n^2} + 0(\frac{1}{n^4})$$

En combinant judicieusement ces deux approximations, on peut en construire une *beaucoup plus précise* :

$$T'_{2n} := \frac{4T_{2n} - T_n}{3} = I + 0(\frac{1}{n^4})$$

Supposons (c'est encore une conséquence d'Euler-Maclaurin) que

$$T'_{2n} = I + \frac{\beta}{n^4} + 0(\frac{1}{n^6})$$

On réitère alors le procédé :

$$T'_{4n} = I + \frac{\beta}{16n^4} + 0(\frac{1}{n^6})$$

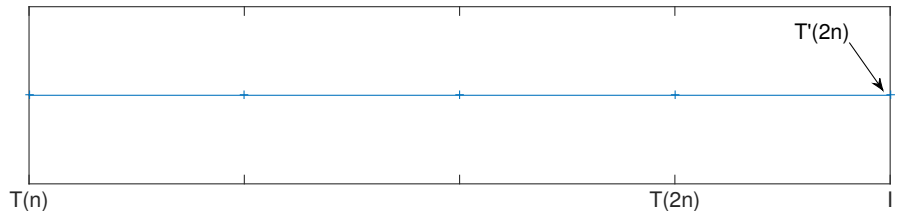
$$T''_{4n} := \frac{16T'_{4n} - T'_{2n}}{15} = I + 0(\frac{1}{n^6})$$

etc...En pratique on dispose les calculs ainsi

	T_n		
	T_{2n}	T'_{2n}	
	T_{4n}	T'_{4n}	T''_{4n}
précision	$0(\frac{1}{n^2})$	$0(\frac{1}{n^4})$	$0(\frac{1}{n^6})$

Evidemment on peut continuer, mais au delà d'un certain nombre d'accélération, les erreurs d'arrondis gâtent l'amélioration escomptée.

Remarque. Le procédé de base consiste à *extrapoler* à la limite. En effet T'_{2n} est un barycentre de T_n et T_{2n} avec des coefficients 4 et -1. Cela revient à placer sur une droite T'_{2n} à l'extérieur du segment (T_nT_{2n}) dans une proportion bien choisie. Voir la figure.



□

Remarque. on voit aisément que

$$T_{2n} = \frac{T_n + R_n}{2}$$

ainsi on fait la moyenne arithmétique des trapèzes et des rectangles. De plus

$$T'_{2n} = \frac{(b-a)}{n}\left\{\sum_j \frac{1}{6}f(x_j) + \frac{4}{6}f(x_{j+1/2}) + \frac{1}{6}f(x_{j+1})\right\} = S_n$$

qui redonne Simpson. [6] □

4.2.3 Methode de Montecarlo

Une méthode particulièrement simple est la méthode de Montecarlo. On tire au hasard les noeuds suivant une loi uniforme sur $[a, b]$, en utilisant par exemple un générateur de suite aléatoire (`rand` en scilab). On approche l'intégrale par la moyenne arithmétique des valeurs au noeuds

$$I(f) \approx (b-a) \cdot \{\frac{\sum_{1\leq j\leq n} f(\xi_j)}{n}\}$$

6. comparer T''_{4n} et Villarceau.

Evidemment le résultat est un variable aléatoire qui dépend de l'échantillon des noeuds tirés. On montre en cours de probabilités que

$$(b-a)\{\frac{\sum_{1\leq j\leq n} f(\xi_j)}{n}\}$$

est un estimateur sans biais de $I(f)$ et on peut donner un intervalle de confiance de largeur $O(1/\sqrt{n})$, grace au théorème central limite. Certes la convergence est bien plus lente que les méthodes déterministes, mais on ne fait aucune hypothèse de régularité sur f. D'autre part la vitesse en $O(1/\sqrt{n})$ est conservée lorsque on calcule des intégrales multiples par cette méthode, alors que la convergence des méthodes déterministes se dégrade lorsque la dimension augmente et devient plus lente que la méthode de Monte Carlo dès que la dimension est supérieure ou égale à 4. (Voir TP 3).

4.3 Méthode de Gauss et polynômes orthogonaux.

4.3.1 Introduction.

Nous avons construit des formules du type de sorte qu'elles soient exactes pour des polynômes de degré le plus élevé possible. Jusqu'à présent, on a placé les noeuds de façon équidistante. On choisissait ensuite les poids de façon à avoir la précision maximale. Pour une formule à n points, on avait un degré de précision n (cas n impair) ou $n-1$ (cas n pair).

Remarque. La précision maximum théorique est $2n-1$. En effet, soit le polynôme $P(x) := \Pi_j(x-x_j)^2$ où les noeuds sont les x_j. Le degré de P est $2n$. Le polynôme P est positif non identiquement nul donc on a évidemment $\int_a^b f(x)dx > 0$ alors que $Q(f) = 0$. □

La précision maximale théorique est donc $(2n-1)$. Les méthodes que nous avons construites jusqu'à présent sont de précision maximum n. Nous allons maintenant construire effectivement une méthode de précision $2n-1$ en *optimisant le placement des noeuds.*

4.3.2 Méthode de Gauss.

Théorème 7 *Il existe une et une seule formule de quadrature*

$$Q(f) := (b-a)\left(\sum_k \omega_k f(x_k)\right)$$

exacte pour les polynômes de degré $\leq 2n-1$. *Sur l'intervalle* $[-1,1]$ *elle s'exprime ainsi :*

$$\int_{-1}^{1} f(x)dx \approx \sum_k w_k f(x_k)$$

où les x_k *sont les racines du* n^e *polynôme de Legendre*

$$P_n(x) := \frac{1}{2^n n!}\frac{d^n}{dx^n}\{(x^2-1)^n\}$$

et

$$w_k = 2\omega_k = \int_{-1}^{1} \Pi_{j\neq k}\left(\frac{x-x_j}{x_k-x_j}\right)^2 dx.$$

Remarque. La formule sur $[a,b]$ se déduit de celle sur $[-1,1]$ par changement de variable affine. Voir proposition 3. En particulier, comme la longueur de l'intervalle $[-1,1]$ est 2, on a $\sum_k w_k = \sum_k 2\omega_k = 2$. □

Avant de démontrer le théorème, donnons quelques cas particuliers.

$$P_0 = 1, P_1(x) = x, P_2(x) = \frac{1}{2}(3x^2-1), P_3(x) = \frac{1}{2}(5x^3-3x), P_4(x) = \frac{1}{8}(35x^4-30x^2+3)$$

Cela donne pour $n = 2$, $x_k = \pm\frac{1}{\sqrt{3}}, \quad w_k = 1$

$$\int_{-1}^{1} f(x)dx \approx f(-\frac{1}{\sqrt{3}}) + f(\frac{1}{\sqrt{3}}).$$

Pour $n = 3$

$$\int_{-1}^{1} f(x)dx \approx \frac{5}{9}f(-\sqrt{\frac{3}{5}}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{\frac{3}{5}})$$

Testons la précision de ces méthodes sur une exemple simple. Prenons

$$f(x) = \frac{1}{1+x^2}$$

,

$$\int_{-1}^{1} f(x)dx = \frac{\pi}{2} \approx 1.57$$

Simpson donne

$$\int_{-1}^{1} f(x)dx \approx 2\{\frac{1}{6}\cdot\frac{1}{2} + \frac{4}{6}1 + \frac{1}{6}\frac{1}{2}\} = \frac{5}{3} = 1.66$$

Gauss (2 points) donne

$$\int_{-1}^{1} f(x)dx \approx \frac{1}{1+\frac{1}{3}} + \frac{1}{1+\frac{1}{3}} = \frac{3}{2} = 1.5$$

Gauss (3 points) donne

$$\int_{-1}^{1} f(x)dx \approx \frac{5}{9}\frac{1}{1+\frac{3}{5}} + \frac{8}{9} + \frac{5}{9}\frac{1}{1+\frac{3}{5}} = \frac{19}{12} = 1.56$$

Sur cet exemple, avec un nombre de point réduit la méthode de Gauss à 3 points est précise à 10^{-2} près alors que Simpson est précise à 10^{-1} près. Pour $n \geq 5$ les noeuds x_k ne s'expriment pas à l'aide de radicaux, ils sont cependant tabulés et disponibles dans les logiciels évolués.

Preuve.

Existence.

Soit

$$P_n(x) := \frac{1}{2^n n!}\frac{d^n}{dx^n}\{(x^2-1)^n\}.$$

P_n est un polynôme de degré n exactement. En effet, il est obtenu en dérivant n fois le polynôme $(x^2-1)^n$ qui est de degré $2n$. Montrons maintenant que $P_n(x)$ a exactement n racines simples sur $]-1,1[$. Nous allons appliquer le théorème de Rolle en rafale. Le polynôme $P(x) = (x^2-1)^n$ prend la même valeur en $x = -1$ et $x = +1$, donc il existe $c \in]-1,1[$ tel que $P' = \frac{d}{dx}(x^2-1)^n$ s'annule en c. Comme -1 et 1 sont racines de multiplicité n de P, -1 et 1 sont encore racine de P' mais de multiplicité $n-1$. Le polynôme P' prend donc la même valeur en $-1, c, 1$. Par le théorème de Rolle, P'' s'annule donc entre -1 et c et entre c et 1, cela fait donc deux racines en dehors des extrémités. Comme -1 et 1 sont encore racine de P'' de multiplicité $n-2$, on peut réitérer : à l'étape n

$$\frac{d^n}{dx^n}\{(x^2-1)^n\}$$

s'annule donc n fois sur $]-1,1[$. Comme $P_n(x)$ est de degré n cela fait donc exactement n racines simples (de multiplicité un) sur $]-1,1[$.

Montrons maintenant que la famille P_n est une famille orthogonale de $L^2(]-1,1[)$ muni du produit scalaire

$$\langle f, g\rangle := \int_{-1}^{1} f(x)g(x)dx.$$

Il s'agit de prouver que $\langle P_n, P_m\rangle = 0$ si $n \neq m$. Cela se montre très simplement en intégrant par parties de manière répétée :

Lemme 4 *Soit $n > m$. On a $\int_{-1}^{1} P_n(x)\,P_m(x)\,dx = 0$.*

Preuve. Intégrons par parties.

$$\int_{-1}^{1} \frac{d^n}{dx^n}\{(x^2-1)^n\}\, P_m(x)\, dx = -\int_{-1}^{1} \frac{d^{n-1}}{dx^{n-1}}\{(x^2-1)^n\}\, P'_m(x)\, dx + [\frac{d^{n-1}}{dx^{n-1}}\{(x^2-1)^n\}\, P_m(x)]_{-1}^{1}.$$

Le terme de bord est nul car -1 et 1 sont racines de multiplicité n de $(x^2-1)^n$! Ainsi on obtient :

$$\int_{-1}^{1} \frac{d^n}{dx^n}\{(x^2-1)^n\}\, P_m(x)\, dx = -\int_{-1}^{1} \frac{d^{n-1}}{dx^{n-1}}\{(x^2-1)^n\}\, P'_m(x)\, dx.$$

On réitère l'intégration par parties $(m+1)$ fois (les termes de bords disparaissent toujours car on a $n \geq m+1$) :

$$\int_{-1}^{1} \frac{d^n}{dx^n}\{(x^2-1)^n\}\, P_m(x)\, dx = (-1)^{(m+1)} \int_{-1}^{1} \frac{d^{n-(m+1)}}{dx^{n-(m+1)}}\{(x^2-1)^n\}\, P_m^{(m+1)}(x)\, dx.$$

et $P_m^{(m+1)}(x) = 0$ car P_m est de degré m.

■

Soit maintenant un polynôme $P(x)$ quelconque de degré $\leq 2n-1$. Effectuons la division euclidienne de P par le n-ième polynôme de Legendre P_n.

$$P = P_n \cdot Q + R. \tag{4.6}$$

Comme $degR < degP_n = n$, le degré de Q est $\leq n-1$. La famille $P_0, P_1, \dots P_{n-1}$ est étagée, donc elle est libre et c'est une base de l'espace des polynômes de degré $\leq n-1$. Or P_n est orthogonal à $P_0, P_1, \dots P_{n-1}$, donc par linéarité $P_n \perp vect\langle P_0, P_1, \dots P_{n-1}\rangle = \mathbb{R}_n[X]$ et $P_n \perp Q$:

$$\int_{-1}^{1} P_n(x)Q(x)\, dx = 0.$$

Avec Eq. (4.6) il vient

$$\int_{-1}^{1} P(x)\, dx = \int_{-1}^{1} R(x)\, dx.$$

D'autre part la formule de quadrature donne :

$$\sum_k w_k P(x_k) = \sum_k w_k (P_n(x_k)Q(x_k) + R(x_k)) = \sum_k w_k R(x_k)$$

car les x_k sont justement les racines de P_n.

Il reste à montrer que

$$\int_{-1}^{1} R(x)\, dx = \sum_k w_k R(x_k)$$

pour tout polynôme R de degré $\leq n-1$. Ceci ne pose aucune difficulté, il suffit de bien choisir les poids w_k et c'est possible avec le lemme 2 en prenant $w_k := \int_{-1}^{1} L_k(x)\, dx$. Avec ce choix de w_k, la formule de quadrature est exacte jusqu'au degré $2n-1$ puisque

$$\int_{-1}^{1} P(x)\, dx = \int_{-1}^{1} R(x)\, dx = \sum_k w_k R(x_k) = \sum_k w_k P(x_k)$$

Le problème dans le lemme c'est que rien ne garantit la positivité des poids : $w_k > 0$.[7] Mais ici, un miracle se produit. Comme la formule de quadrature est exacte pour P de degré $\leq 2n-1$, elle l'est pour $P := L_k^2$ qui est de degré $2n-2$. Rappelons en effet que

$$L_j(X) = \frac{\Pi_{i\neq j}(X - x_i)}{\Pi_{i\neq j}(x_j - x_i)}.$$

7. En fait, pour $n \geq 8$ avec des noeuds x_k équidistants, certains poids peuvent être négatifs.

Calculons

$$\int_{-1}^{1} L_k(x)^2\,dx = \sum_j w_j L_k^2(x_j) = w_k$$

car dans la somme un seul terme est non nul. On obtient finalement

$$w_k = \int_{-1}^{1} L_k(x)^2\,dx > 0$$

Unicité.
Soit une autre formule de quadrature à n noeuds définie par les noeuds x'_k et les poids w'_k, $k = 1, \ldots, n$. Les poids sont évidemment supposés non nuls, sinon le noeud x'_k ne compte pas dans la quadrature

$$Q'(f) := \sum_k w'_k f(x'_k).$$

Soit L_k le polynôme de Lagrange *construit sur les noeuds* x'_j.[8] Il est de degré $n-1$ donc $P_n \perp L_k$:

$$\int_{-1}^{1} P_n(x) L_k(x)\,dx = 0$$

Or en utilisant la formule de quadrature Q' qui est exacte car $deg\,(P_n \cdot L_k) \leq 2n-1$, on obtient

$$\int_{-1}^{1} P_n(x) L_k(x)\,dx = \sum_j w'_j P_n(x'_j) L_k(x'_j) = w'_k P_n(x'_k).$$

D'où l'on tire que

$$P_n(x'_k) = 0$$

donc que les noeuds x'_k sont les racines de P_n donc, à permutation éventuelle près, ils sont égaux aux x_k. Les deux quadratures ont donc les mêmes noeuds, elles ont donc aussi les mêmes poids correspondants (en vertu du lemme 2).

■

8. on devrait le noter L'_k mais il y a un risque de confusion avec la dérivée...