Mat 367, Méthodes numériques

Bernard.Parisse@ujf-grenoble.fr

2016

La version HTML de ce document comporte des champs de saisie interactifs, ceux-ci apparaissent comme des commandes “mortes” dans la version PDF. La version HTML est optimisée pour le navigateur Firefox. Vous pouvez exécuter toutes les commandes interactives en cliquant sur le bouton Exécuter (attention cela peut prendre un certain temps !), le champ suivant est la console de l’interpréteur du logiciel de calcul formel.
Les commandes sont exécutées par la version javascript de Giac/Xcas. Bien que le thème de ce cours soit numérique, on verra sur de nombreuses commandes qu’il est fort utile de disposer d’un logiciel capable de faire du calcul formel !

Table des matières

Index

1 Présentation du module

Les thèmes abordés seront :

1. calcul approché, représentation des données (flottants, vecteurs, matrices), erreurs (normes, de calcul, d'arrondi...).
2. Pivot de Gauss, factorisation LU, conditionnement, Cholesky, factorisation QR.
3. Interpolation polynômiale (évaluation, interpolation de Lagrange, Hermite, Bézier)
4. Intégration numérique
5. Méthode du point fixe, de Newton, méthodes itératives en algèbre linéaire
6. Méthode de la puissance, valeurs propres et vecteurs propres.
7. Résolution d'équations différentielles.

L'évaluation se fait sur :

- 1/2 : un DS à mi-semestre et certains compte-rendus de TP (à rédiger seul ou en binome),
- 1/2 : l'examen final

Les calculatrices et les netbooks de taille d'écran plus petits que 13 pouces sont autorisées au DS et à l'examen final (prêt possible de netbooks pour le semestre).

2 Représentation des nombres et autres données, calcul exact/approché

Mot-clefs :
Types de base : entier machine, entier long, flottant machine et multiprécision (Base 2, base 10).
Erreur relative, erreur absolue, erreur d'arrondi, +/-, */ Algorithme de Horner. Types composés : complexes, polynomes (représentation dense/creuse), symboles, listes (vecteurs, matrices), expressions, fonctions.
Les principaux ensembles de nombres en mathématiques sont les entiers positifs $\mathbb{N}$ et relatifs $\mathbb{Z}$, les rationnels $\mathbb{Q}$, les réels $\mathbb{R}$ et les complexes $\mathbb{C}$. Sur ordinateur, on peut représenter ces nombres de manière exacte dans certains cas, approchée dans d'autres.

2.1 Représentation des entiers

Proposition 1 Division euclidienne de deux entiers : *si a et b sont deux entiers, $a \geq 0, b > 0$, il existe un unique couple (q, r) tel que*

$$a = bq + r, \quad r \in [0, b[$$

Preuve : On prend pour q le plus grand entier tel que $a - bq \geq 0$.
Exemple : `iquorem(23,7)`

$$[3, 2]$$

La division euclidienne permet d'écrire un nombre entier, en utilisant une base b et des caractères pour représenter les entiers entre 0 et $b-1$. Nous écrivons les nombres entiers en **base** $b = 10$ avec comme caractères les chiffres de 0 à 9. Les ordinateurs utilisent des circuits binaires pour stocker les informations, il est donc naturel d'y travailler en base 2 en utilisant comme caractères 0 et 1 ou en base 16 en utilisant comme caractères les chiffres de 0 à 9 et les lettres de A à F. En général, pour trouver l'écriture d'un nombre en base b (par exemple $b = 2$), on effectue des divisions euclidienne successives par b du nombre puis de ses quotients successifs jusqu'à ce que le quotient soit

0 et on accolle les restes obtenus (premier reste à droite, dernier reste à gauche). Inversement, pour retrouver un entier d à partir de son écriture $d_n...d_0$, on traduit les divisions euclidiennes successives en

$$\begin{aligned} d &= (...((d_n b + d_{n-1})b + d_{n-2})... + d_1)b + d_0 \\ &= d_n b^n + d_{n-1}b^{n-1} + ... + d_0 \end{aligned}$$

Par exemple, vingt-cinq s'écrit en base 16 `0x19` car 25 divisé par 16 donne quotient 1, reste 9

```
convert(25,base,16)
```

$$[9, 1]$$

En base 2, on trouverait `0b11001` car $25 = 2^4 + 2^3 + 1$.

```
convert(25,base,2)
```

$$[1, 0, 0, 1, 1]$$

On peut effectuer les opérations arithmétiques de base (+,-,*, division) directement en base 2 (ou 16). Par exemple la table de l'addition est 0+0=0, 0+1=1+0=1 et 1+1=0 je retiens 1, donc :

```
  01001111
+ 01101011
----------
  10111010
```

Exercice : comment passe-t-on simplement de la représentation d'un nombre en base 2 à un nombre en base 16 et réciproquement ? Les microprocesseurs peuvent effectuer directement les opérations arithmétiques de base sur les entiers "machine" (déclinés en plusieurs variantes selon la taille et la possibilité d'avoir un signe). Noter que la division de deux entiers a et b n'a pas la même signification que la division de deux réels, comme elle ne tomberait pas forcément juste, on calcule le quotient et le reste de la division euclidienne. Ces entiers machines permettent de représenter de manière exacte des petits entiers relatifs par exemple un entier machine signé sur 4 octets est compris entre $[-2^{31}, 2^{31} - 1]$. Ces entiers machines permettent de faire très rapidement du calcul exact sur les entiers, mais à condition qu'il n'y ait pas de dépassement de capacité, par exemple pour des entiers 32 bits, $2^{30} + 2^{30} + 2^{30} + 2^{30}$ renverra 0. Ils sont utilisables avec tous les langages de programmation traditionnels. Les logiciels de calcul formel et certains logiciels de programmation permettent de travailler avec des entiers de taille beaucoup plus grande, ainsi qu'avec des rationnels, permettant de faire du calcul exact, mais on paie cette exactitude par un temps de calcul plus long, de plus pas mal de méthodes numériques ne gagnent rien à faire des calculs intermédiaires exacts. Néanmoins, l'utilisation d'un logiciel de calcul formel permettra dans certains cas d'illustrer certains phénomènes dus au calcul approché.

2.2 Les réels

On se ramène d'abord au cas des réels positifs, en machine on garde traditionnellement un bit pour stocker le signe du réel à représenter.

2.2.1 Virgule fixe et flottante.

La première idée qui vient naturellement serait d'utiliser un entier et de déplacer la virgule d'un nombre fixe de position, ce qui revient à mulitplier par une puissance (négative) de la base. Par exemple en base 10 avec un décalage de 4, `1234.5678` serait représenté par `12345678` et `1.2345678` par `12345` (on passe de l'entier au réel par multiplication par 10^{-4}. L'inconvénient d'une telle représentation est qu'on ne peut pas représenter des réels grands ou petits, comme par exemple le nombre d'Avogadro, la constante de Planck, etc. D'où l'idée de ne pas fixer la position de la virgule, on parle alors de représentation à virgule flottante ou de nombre flottant : on représente un nombre par deux entier, l'un appelé **mantisse** reprend les chiffres significatifs du réel sans virgule, l'autre l'exposant, donne la position de la virgule. Attention, le séparateur est un point et non une virgule dans la grande majorité des logiciels scientifiques. On sépare traditionnellement la mantisse de l'exposant par la lettre `e`. Par exemple `1234.5678` peut être représenté par `12345678e-8` (mantisse `12345678`, exposant -8) mais aussi par `1234567800e-10`. Naturellement, sur un ordinateur, il y a des limites pour les entiers représentant la mantisse m et l'exposant e. Si on écrit les nombres en base b, la mantisse m s'écrira avec un nombre n fixé de chiffres (ou de bits en base 2), donc $m \in [0, b^n[$. Soit un réel x représenté par

$$x = mb^e, \quad m \in [0, b^n[$$

Si $m \in [0, b^{n-1}[$, alors on peut aussi écrire $x = m'b^{e-1}$ avec $m' = mb \in [0, b^n[$, quelle écriture faut-il choisir ? Intuitivement, on sent qu'il vaut mieux prendre m' le plus grand possible, car cela augmente le nombre de chiffres significatifs (alors que des 0 au début de m ne sont pas significatifs). Ceci est confirmé par le calcul de l'erreur d'arrondi pour représenter un réel. En effet, si x est un réel non nul, il ne s'écrit pas forcément sous la forme mb^e, on doit l'arrondir, par exemple au plus proche réel de la forme mb^e. La distance de x à ce réel est inférieure ou égale à la moitié de la distance entre deux flottants consécutifs, mb^e et $(m+1)b^e$, donc l'erreur d'arrondi est inférieure ou égale à $b^e/2$. Si on divise par $x \geq mb^e$, on obtient une erreur relative d'arrondi majorée par $1/(2m)$. On a donc intérêt à prendre m le plus grand possible pour minimiser cette erreur. Quitte à mulitplier par b, on peut toujours se ramener (sauf exceptions, cf. ci-dessous), à $m \in [b^{n-1}, b^n[$, on a alors une erreur d'arrondi relative majorée par

$$\frac{1}{2b^{n-1}}$$

On appelle **flottant normalisé** un flottant tel que $m \in [b^{n-1}, b^n[$. Pour écrire un réel sous forme de flottant normalisé, on écrit le réel en base b, et on déplace la virgule pour avoir exactement n chiffres non nuls avant la virgule et on arrondit (par exemple au plus proche). L'exposant est égal au décalage effectué. Notez qu'en base 2, un flottant normalisé commence forcément par 1, ce qui permet d'économiser un bit dans le stockage. Ainsi, l'erreur d'**arrondi** commise lorsqu'on représente un réel (connu exactement) par un double normalisé est une erreur relative inférieure à de 2^{-53} ($b = 2$ et $n = 52 + 1$ pour les doubles). Exemples :

- en base 10 avec $n = 6$, pour représenter $\pi = 3,14159265...$, on doit décaler la virgule de 5 positions, on obtient `314159.265...` on arrondit à 314159 donc on obtient `314159e-5`.
- en base 2 avec $n = 10$, pour représenter trois cinquièmes ($3/5$ en base 10, noté $11/101$ en base 2), on pose la division en base 2 de `11` par `101`, ce qui donne

```
 11         | 101
 110        ---------
-101        | 0.1001
----        |
  010       |
   100      |
   1000     |
```

```
- 101     |
-----     |
  011     |
```

on retrouve le nombre de départ donc le développement est périodique et vaut `0.1001 1001 1001 ...`. On décale le point de 10 positions, on arrondit, donc trois cinquièmes est représenté par la mantisse `1001100110` et l'exposant `-10`. On observe aussi sur cet exemple que $3/5$ dont l'écriture en base 10 `0.6` est exacte, n'a pas d'écriture exacte en base 2 (de même que 1/3 n'a pas d'écriture exacte en base 10).

Il existe une exception à la possibilité de normaliser les flottants, lorsqu'on atteint la limite inférieure de l'exposant e. Soit en effet e_m le plus petit exposant des flottants normalisés et considérons les flottants $x = b^{e_m}(1 + 1/b)$ et $y = b^{e_m}$. Ces flottants sont distincts, mais leur différence n'est plus représentable par un flottant normalisé. Comme on ne souhaite pas représenter $x - y$ par 0, (puisque le test $x == y$ renvoie faux), on introduit les flottants **dénormalisés** , il s'agit de flottants dont l'exposant est l'exposant minimal représentable sur machine et dont la mantisse appartient à $[0, b^{n-1}[$. Par exemple 0 est représenté par un flottant dénormalisé de mantisse 0 (en fait 0 a deux reprśentation, une de signe positif et une de signe négatif). Enfin, on utilise traditionnellement une valeur de l'exposant pour représenter les nombres plus grands que le plus grand réel reprśentable sur machine (traditionnellement appelé plus ou moins infini) et les erreurs (par exemple 0./0. ou racine carrée d'un nombre réel négatif, traditionnellement appelé NaN, Not a Number). **Exercice** : quels sont les nombres réels représentables exactement en base 10 mais pas en base 2 ? Si on écrit $1/10$ en base 2 avec 53 bits de précision, puis que l'on arrondit avec 64 bits de précision, ou si on écrit $1/10$ en base 2 avec 64 bits de précision, obtient-on la même chose ? Les ordinateurs reprśentent généralement les flottants en base 2 (cf. la section suivante pour plus de précisions), mais cette représentation n'est pas utilisée habituellement par les humains, qui préfèrent compter en base 10. Les ordinateurs effectuent donc la conversion dans les routines d'entrée-sortie. Le format standard utilisé pour saisir ou afficher un nombre flottant dans un logiciel scientifique est composé d'un nombre à virgule flottante utilisant le point comme séparateur décimal (et non la virgule) suivi si nécessaire de la lettre `e` puis de l'exposant, par exemple `1.23e-5` ou `0.0000123`. Dans les logiciels de calcul formel, pour distinguer un entiers représentés par un entier d'un entier représenté par un flottant on écrit l'entier suivi de `.0` par exemple `23.0`. **Remarque** :
Les microprocesseurs ayant un mode **BCD** peuvent avoir un format de représentation des flottants en base 10, les nombres décimaux comme par exemple 0.3 peuvent être représentés exactement. Certains logiciels, notamment maple, utilisent par défaut des flottants logiciels en base 10 sur des microprocesseurs sans mode BCD, ce qui entraine une baisse de rapidité importante pour les calculs numériques (on peut partiellement améliorer les performances en utilisant `evalhf` en maple).

2.2.2 Les flottants au format `double`

Cette section développe les notions de la section précédente pour les flottants machine selon la norme IEEE-754, utilisables dans les langage de programmation usuels, elle peut être omise en première lecture. La représentation d'un double en mémoire se compose de 3 parties : le bit de signe $s = \pm 1$ sur 1 bit, la mantisse $M \in [0, 2^{52}[$ sur 52 bits, et l'exposant $e \in [0, 2^{11}[$ sur 11 bits. Pour les nombres "normaux", l'exposant est en fait compris entre 1 et $2^{11} - 2$, le nombre représenté est le rationnel

$$(1 + \frac{M}{2^{52}})2^{e+1-2^{10}}$$

Pour écrire un nombre sous cette forme, il faut d'abord chercher par quel multiple de 2 il faut le diviser pour obtenir un réel r dans $[1, 2[$, ce qui permet de déterminer l'exposant e. Ensuite on écrit la représentation en base 2 de $r - 1 \in [0, 1[$. Exemples :

– -2
  Signe négatif. Il faut diviser sa valeur absolue 2 par 2^1 pour être entre 1 et 2 dont $e+1-2^{10}=1$, l'exposant est $e=2^{10}$. On a alors $r=1$, $r-1=0$. Représentation
  `1 10000000000 00000000...0000`
– 1.5=3/2
  Signe positif, compris entre 1 et 2 dont l'exposant vérifie $e+1-2^{10}=0$ soit $e=2^{10}-1=2^9+2^8+2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0$. On a $r-1=1/2=2^{-1}$. D'où la représentation
  `0 01111111111 10000000...0000`
– 6.4=32/5
  Positif. Il faut le diviser par 2^2 pour avoir $8/5 \in [1,2[$ donc $e+1-2^{10}=2$ soit $e=2^{10}+1$. Ensuite $r=3/5$ qu'il faut écrire en base 2 (cf. section précédente), on écrit donc les 52 premiers éléments du développement avec une règle d'arrondi du dernier bit au nombre le plus proche. Ici le bit suivant le dernier `1001` est un `1`, on arrondit donc à `1010`. D'où la représentation
  `0 1000000001 100110011001...10011010`

On observe que la représentation en base 2 de 6.4 a du être arrondie (car elle est infinie en base 2) bien qu'elle soit exacte (finie) en base 10. Seuls les entiers et les rationnels dont le dénominateur est une puissance de 2 peuvent être représentés exactement. Ceci entraine des résultats qui peuvent surprendre comme par exemple le fait que `0.5 - 5*0.1` n'est pas nul. Des représentations spéciales (avec $e=0$ ou $e=2^{11}-1$) ont été introduites pour représenter $\pm\infty$ (pour les flottants plus grands en valeur absolue que le plus grand flottant représentable), et pour représenter les nombres non nuls plus petits que le plus petit flottant représentable de la manière exposée ci-dessus (on parle de flottants dénormalisés), ainsi que le nombre NaN (Not a Number) lorsqu'une opération a un résultat indéfini (par exemple 0/0). Remarque : Sur les processeurs compatibles avec les i386, le coprocesseur arithmétique i387 gère en interne des flottants avec 80 bits dont 64 bits de mantisse. Sur les architectures 64 bits (x86 ou AMD), le jeu d'instruction SSE permet de travailler avec des flottants de 128 bits. Le compilateur gcc permet d'utiliser ces flottants longs avec le type `long double` ou les types `__float80` et `__float128` en utilisant un drapeau de compilation du type `-msse`

2.2.3 Opérations sur les flottants

Les opérations arithmétiques de base sur les flottants se font de la manière suivante :
– addition et soustraction : on détecte s'il faut additionner ou soustraire en valeur absolue en analysant les signes, on détermine l'exposant le plus grand et on décale la partie mantisse du flottant dont l'exposant est le plus petit pour se ramener à additionner deux entiers (partie mantisses correspondant au même exposant), on décale à nouveau la partie mantisse en modifiant l'exposant après l'opération pour normaliser le flottant
– multiplication : on additionne les exposants et on multiplie les parties mantisses (vus comme des entiers), on arrondit et on ajuste l'exposant si nécessaire
– division : on soustrait les exposants et on divise les parties mantisses (division "à virgule"), on tronque et on ajuste l'exposant si nécessaire

2.2.4 Erreurs

La représentation des nombres réels par des doubles présente des avantages, les opérations arithmétiques sont faites au plus vite par le microprocesseur. Les coprocesseurs arithmétiques (intégrés sur les microprocesseurs de PC) proposent même le calcul des fonctions usuelles (trigonométriques, racine carrée, log et exp) sur le type double et utilisent des formats de représentation interne ayant plus de 64 bits pour les doubles, ce qui permet de limiter les

erreurs d'arrondi. Par contre, des erreurs vont être introduites, on parle de calcul approché par opposition au calcul exact sur les rationnels. En effet, la représentation doit d'abord arrondir tout réel qui n'est pas un rationnel dont le dénominateur est une puissance de 2. Ensuite chaque opération va entrainer une propagation de ces erreurs et va y ajouter une erreur d'arrondi sur le résultat. Enfin, l'utilisation du type double peut provoquer un dépassement de capacité (par exemple `100!*100!`). Pour diminuer ces erreurs et les risques de dépassement de capacité, il existe des types flottants multiple précision, qui permettent de travailler avec un nombre fixé à l'avance de décimales et une plage d'exposants plus grande. Les calculs sont plus longs mais les erreurs plus faibles. Attention, il s'agit toujours de calcul approché ! De plus, pour des quantités dont la valeur est déterminée de manière expérimentale, la source principale de propagation d'erreurs est la précision des quantités initiales, il ne sert souvent à rien d'utiliser des types flottants multiprécision car les erreurs dus à la représentation (double) sont négligeables devant les erreurs de mesure. Dans ce cas, il est pertinent lorsqu'on évalue $f(x)$ avec x mal connu de calculer aussi $f'(x)$, en effet :

$$f(x(1+h)) = f(x) + xhf'(x) + O(h^2)$$

l'erreur relative sur $f(x)$ est donc au premier ordre multipliée par

$$|\frac{xf'(x)}{f(x)}|$$

Par exemple, l'erreur relative sur e^x est au premier ordre l'erreur relative sur x multipliée par $|x|$.

```
a:=convert(100,interval); (right(a)-left(a))/evalf(a,12)
```

$$[0.999999999999991e2..0.100000000000000e3], 0.136424205265939e-13$$

```
b:=exp(a); (right(b)-left(b))/evalf(b,12)
```

$$[0.268811714181369e44..0.268811714181736e44], 0.136443784064836e-11$$

2.2.5 Erreur absolue, relative, arrondi propagation des erreurs.

On a vu précédemment que pour représenter un réel, on devait l'arrondir, ce qui introduit une erreur même si le réel est connu exactement (par exemple 1/10). Voyons comment se propagent les **erreurs** dans les opérations arithmétiques de base : on distingue l'addition, la multiplication et l'inversion. La soustraction se ramène à l'addition car le calcul de l'opposé n'introduit aucune erreur nouvelle. Pour l'addition, si $|x - x_0| \leq \varepsilon_0$ et si $|y - y_0| \leq \varepsilon_1$ alors par l'inégalité triangulaire ($|a + b| \leq |a| + |b|$), on a :

$$|(x + y) - (x_0 + y_0)| \leq |x - x_0| + |y - y_0| \leq \varepsilon_0 + \varepsilon_1$$

on dit que les erreurs *absolues* s'additionnent.

Définition 1 *L'***erreur absolue** *est définie comme un majorant de la valeur absolue de la différence entre le nombre réel et son représentant double :*

$$|x - x_0| \leq \varepsilon$$

Mais comme il faut représenter $x_0 + y_0$ en machine, on doit ajouter une erreur d'arrondi, qui est proportionnelle à la valeur absolue de $x_0 + y_0$ d'où la notion d'erreur *relative* :

Définition 2 *L'***erreur relative** *est égale à l'erreur absolue divisée par la valeur absolue du nombre*

$$|x - x_0| \leq \varepsilon |x_0|$$

Remarquons au passage que les erreurs de mesure expérimentales sont pratiquement toujours des erreurs relatives. Donc lorsqu'on effectue une addition (ou une soustraction) de deux réels sur machine, on doit additionner les deux erreurs absolues sur les opérandes et ajouter une erreur d'arrondi (relative de 2^{-53}, à titre d'exercice, on pourra vérifier que cette erreur d'arrondi est majorée par l'erreur absolue de la somme $x+y$ dès l'instant où x et y ont eux-même une erreur d'arrondi). Lorsqu'on effectue une multiplication de deux nombres x, y dont les représentants x_0, y_0 sont non nuls, on a

$$\left|\frac{xy - x_0y_0}{x_0y_0}\right| = \left|\frac{x}{x_0}\frac{y}{y_0} - 1\right| = \left|(\frac{x}{x_0} - 1)(\frac{y}{y_0} - 1) + (\frac{x}{x_0} - 1) + (\frac{y}{y_0} - 1)\right|$$

l'erreur relative est donc la somme des erreurs relatives et du produit des erreurs relatives (on peut souvent négliger le produit devant la somme). Il faut aussi y ajouter une erreur relative d'arrondi de 2^{-53} sur x_0y_0. On observe que la multiplication est une opération posant moins de problèmes que l'addition, car on manipule toujours des erreurs relatives, par exemple si l'erreur relative sur deux doubles x et y non nuls est de 2^{-53}, alors l'erreur relative sur xy sera de

$$2^{-53} + 2^{-53} + 2^{-106} + 2^{-53} \approx 3 \times 2^{-53}$$

Lorsque l'erreur relative sur les données est grande devant 2^{-53}, l'erreur relative d'arrondi final est négligeable, on peut alors dire que les erreurs relatives s'additionnent pour un produit (c'est aussi vrai pour un quotient : exercice !). Par contre, si on additionne deux nombres dont le représentant de la somme est proche de 0, la somme des erreurs absolues peut devenir non négligeable par rapport à la somme des représentants, entrainant une erreur relative très grande. Par exemple si x est représenté par $x_0 = 1 + 2^{-52}$ avec une erreur d'arrondi de 2^{-53} et y par $y_0 = -1$ avec la même erreur d'arrondi, l'addition de x et y renvoie 2^{-52} avec une erreur absolue de $2 * 2^{-53}$ (ici il n'y a pas d'arrondi lorsqu'on fait la somme). C'est une erreur relative de 1 (qui domine largement l'erreur d'arrondi) ce qui signifie que dans la mantisse, seul le premier bit sur les 52 a un sens, la perte de précision est très grande. Une autre conséquence importante est que l'addition de réels sur machine n'est pas une opération associative, par exemple

$$(2.0^{-53} + 2.0^{-53}) + 1.0 \to 1 + 2^{-52}$$

alors que

$$2.0^{-53} + (2.0^{-53} + 1.0) \to 1$$

Dans Xcas, il n'y a que 48 bits de mantisse :

```
(2.^(-48)+(2.^(-48)+1.))-1.;   ((2.^(-48)+2.^(-48))+1.)-1.
```

$$0.0, 7.1054273576e - 15$$

Si on a plusieurs termes à additionner, il faut commencer par additionner entre eux les termes les plus petits, pour que les petits termes ne soient pas absorbés un à un dans les erreurs d'arrondi (les petits ruisseaux font les grands fleuves). Exercice : pour calculer la valeur numérique d'une dérivée de fonction, il vaut mieux calculer $(f(x+h)-f(x-h))/(2h)$ que $(f(x+h)-f(x))/h$ car le terme d'erreur est en $O(h^2)$ et non en $O(h)$. Attention toutefois à ne pas prendre h trop petit, sinon $x + h = x$ en flottants et même si $x + h \neq x$, l'erreur absolue sur $f(x+h) - f(x-h)$ est (au moins) d'ordre $\varepsilon|f(x)|$, donc l'erreur relative est d'ordre $\varepsilon/h|f(x)|$. Par exemple pour `h=1e-8` le reste est en $O(h^2)$ donc de l'ordre des erreurs d'arrondi mais l'erreur relative sur $f(x+h) - f(x-h)$ est d'ordre ϵ/h largement supérieure (en flottants double-précision). On choisira plutôt h tel que ϵ/h soit proche

de h^2, donc de l'ordre de `1e-5`, qui fournira une valeur approchée avec une erreur relative de l'ordre de `1e-10`.
Exemple : calcul de la dérivée numérique de $\exp(\sin(x))$ en $x = 1$

```
f(x):=exp(sin(x)); seq(taux_accroissement(f(x),1.0-10^(-k),1.0+10^(-k))-f'(1.0),k,1,11)
```

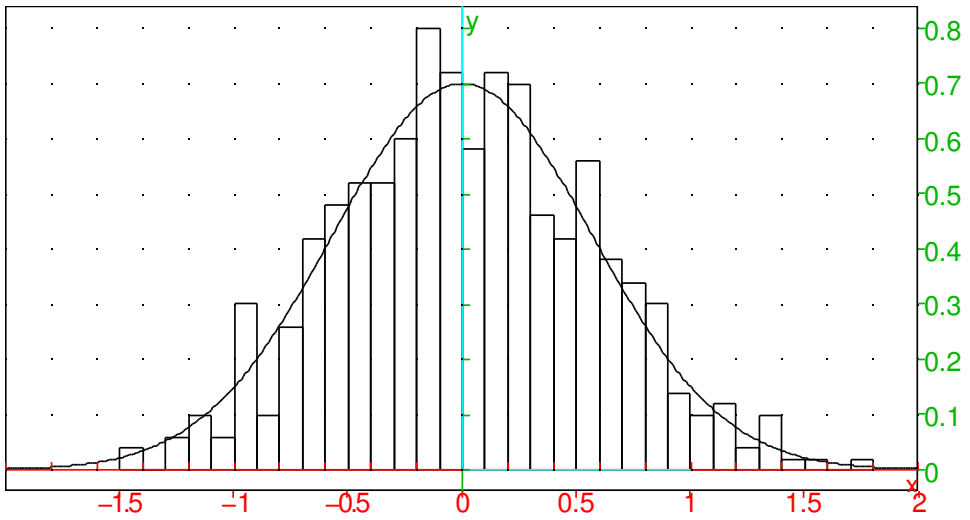
```
(x)->exp(sin(x))
```
$, [-0.00673280473483, -6.75236243595e-0$

Remarquons néanmoins que les erreurs calculées ici sont des majorations des erreurs réelles (ou si on préfère l'erreur obtenue dans le pire des cas), statistiquement les erreurs sur les résultats sont moindres, par exemple si on effectue n calculs susceptibles de provoquer des erreurs indépendantes suivant une même loi d'espérance nulle, la moyenne des erreurs divisée par l'écart-type de la loi tend vers une loi normale centrée réduite. De manière plus déterministe, on a l'inégalité de Bienaymé-Tchebyshev

$$P(|X| > \alpha) \leq \frac{n\sigma^2}{\alpha^2}$$

où X est la variable aléatoire somme des n erreurs, α l'erreur et $n\sigma^2$ la variance de la somme n erreurs supposées indépendantes, cette probabilité tend vers 0 pour n grand si α est d'ordre n, et ne tend pas vers 0 si α est de l'ordre de $\sqrt{n}$. Exemple : somme de $n = 400$ nombres répartis sur $[-1, 1]$ selon la loi uniforme (représentant des erreurs), on divise par $\sqrt{n}$ =20, on effectue plusieurs tirages (par exemple 500) on trace l'histogramme et on compare avec la loi normale de moyenne nulle (l'espérance de la somme) et d'écart-type celui de la loi uniforme.

```
m:=ranm(400,500,-1..1):;gl_x=-2..2;histogram(sum(m)/20,-1,0.1);
 plot(normald(0,0.57),-2..2)
```



Il est d'ailleurs souvent trop difficile de calculer une majoration rigoureuse de l'erreur pour des calculs sauf les plus simples. Lorsqu'on doute de la précision d'un calcul, un test peu couteux consiste à refaire ce calcul en utilisant des flottants en précision plus grande et tester si le résultat varie en fonction du nombre de chiffres significatifs utilisés, ou faire varier légèrement les données et observer la sensibilité du résultat. Si on veut travailler en toute rigueur sans pour autant calculer les erreurs à priori, il faut utiliser un logiciel utilisant des intervalles pour représenter les réels (section suivante)

2.3 L'arithmétique d'intervalle.

Certains systèmes de calcul formel peuvent manipuler directement des intervalles réels, par exemple par l'intermédiaire de la bibliothèque C MPFI. Les opérations arithmétiques sur des intervalles renvoient alors le meilleur intervalle possible contenant toutes les valeurs possibles lorsque les opérandes parcourent leurs intervalles respectifs. Exemple en Xcas (version 1.1.1 et ultérieures) : `[-1..2]*[-1..2]` renvoie `[-2..4]`. Attention ici on parcourt toutes les valeurs possibles de $xy,\ x \in [-1,2], y \in [-1,2]$. Ce qui est différent du carré d'un intervalle ou plus généralement de l'évaluation d'un polynôme en un intervalle, `horner(x^2,[-1..2])` renvoie ainsi `[0..4]`. Les fonctions disponibles sont souvent moins riches qu'en arithmétique flottante, le calcul d'une fonction non monotone sur un intervalle peut s'avérer délicat, alors que si la fonction est monotone, il suffit de calculer l'image des deux bornes de l'intervalle. Pour les polynômes, Xcas décompose les coefficients en deux parties $P = P_+ - P_-$ en fonction du signe, puis utilise la monotonie de P_+ et P_- sur $\mathbb{R}^+$ et $\mathbb{R}^-$ respectivement. L'arithmétique d'intervalle dans $\mathbb{C}$ est beaucoup plus difficile à mettre en oeuvre puisqu'il n'y a plus d'ordre ni de monotonie, on doit alors s'en remettre à des estimations sur les parties réelles et imaginaires qui ne tiendront pas compte du phénomène ci-dessus sur la différence entre $xy,\ x \in [-1,2], y \in [-1,2]$ et $x^2,\ x \in [-1,2]$.

2.4 Types composés.

Après les nombres réels, on passe aux **nombres complexes** : on utilise un couple (partie réelle, imaginaire) de fractions (exacts) ou de flottants et les règles habituelles sur les complexes. On peut représenter les **polynôme** de la facon suivante :

- les polynômes à 1 variable, représentation dense, on stocke la liste des coefficients du polynôme par ordre croissant ou décroissant
- les polynômes à 1 variable, représentation creuse, on stocke des paires coefficients, degré pour les coefficients non nuls
- les polynômes à plusieurs variables, représenté récursivement de manière dense ou creuse (i.e. $P(x_1, ..., x_n)$ vu comme polynôme en x_n à coefficients polynômes dépendant des variables $x_1, ..., x_{n-1}$), ce sont des cas particuliers des 2 cas précédents
- les polynômes à plusieurs variables distribués, on stocke des monômes, qui sont des paires coefficient, liste d'entiers, la liste représentant les exposant des variables dans le monôme.
- la représentation symbolique (par exemple $xy^2 - 5x + y^3$) beaucoup plus difficile à manipuler directement

Algorithmes de base sur les polynômes : l'évaluation en un point (Horner, cf. TD/TP), la multiplication et division euclidienne et le PGCD (même algorithme que pour les entiers mais avec la division euclidienne des polynômes, attention toutefois aux erreurs d'arrondis) Les **symboles** ou noms de variable désignent par exemple le nom d'une inconnue dans un polynôme, ils sont représentés par une chaine de caractére et peuvent être affectés à une valeur pendant une session (la valeur dépend d'un contexte d'exécution et le remplacement du symbole par sa valeur affectée est appelé évaluation). Les **expressions**, par exemple `sin(x)+2*x^2`, elles peuvent être représentées par des arbres. L'évaluation d'une expression consiste à remplacer les symboles de l'expression par leur valeur, puis à effectuer les opérations en tenant compte de la substitution. Il est parfois souhaitable de ne pas effectuer certaines opérations de substitution, on empêche l'évaluation, explicitement (`''`) ou implicitement (par exemple l'affectation n'évalue pas le symbole qu'on va affecter). Les **fonctions** ne doivent pas être confondues avec les expressions, elles associent à leurs arguments une expression. Par exemple `sin` est une fonction, alors que `sin(x)` est une expression. Les conteneurs contiennent plusieurs objets et permettent d'associer à un indice un objet. Il en existe de plusieurs types, par exemple les **listes** et les **séquences** dont l'indice est un entier compris entre 1 (ou 0) et la taille (-1), les **tables** dont l'indice est plus général, et les tableaux (utilisés pour les **vecteurs, matrices**) qui sont essentiellement des listes ou des listes de listes de même taille. Les séquences sont des listes d'objets ordonnés

"non récursifs" (ils ne peuvent contenir des séquences), alors que les listes peuvent contenir des listes, sinon il n'y a pas de différences. Dans les logiciels de calcul formel, la plupart du temps les séquences se notent en indiquant les éléments séparés par des virgules. Les listes s'en distinguent par les délimiteurs `[]`. Il faut prendre garde au fait qu'en général affecter par exemple `l[1]:=3;` à une variable libre `l` crée une table et non une liste. Remarque : certains logiciels accédent à certains types de conteneurs uniquement par référence (par exemple maple pour les vecteurs et matrices), dans ce dernier cas une seule copie des objets du conteneur existe si on copie de la manière habituelle une variable contenant un vecteur ou une matrice dans une autre variable, la modification d'un élément du conteneur modifie alors toutes les copies pointant sur ce conteneur. Cette méthode est plus efficace mais peut être surprenante.

3 Algèbre linéaire

3.1 Le pivot de Gauss

Cet algorithme permet de créer des zéros en effectuant des manipulations réversibles sur les lignes d'une matrice. Ces lignes peuvent représenter les coefficients d'un système linéaire, on obtient alors un système linéaire équivalent, ou les coordonnées d'un système de vecteur, on obtient alors les coordonnées d'un système de vecteur engendrant le même sous-espace vectoriel. On peut également représenter 2 matrices A et B reliés par une relation $Ax = B$, cette relation reste alors vraie au cours et donc après la réduction.

3.1.1 L'algorithme

L'algorithme est le suivant :

1. on initialise $c = 1$ et $l = 1$, c désigne le numéro de colonne c à réduire, et l le numéro de ligne à partir duquel on cherche un "pivot" (au début l et c valent donc 1, en général les 2 augmentent de 1 à chaque itération)
2. Si c ou l est plus grand que le nombre de colonnes ou de lignes on arrête.
3. Si la colonne c n'a que des coefficients nuls à partir de la ligne l, on incrémente le numéro de colonne c de 1 et on passe à l'étape 2. Sinon, on cherche la ligne dont le coefficient est en valeur absolue le plus grand possible (en calcul approché) ou le plus simple possible (en calcul exact), on échange cette ligne avec la ligne l. Puis on effectue pour toutes les lignes sauf l ou pour toutes les lignes à partir de $l + 1$ (selon qu'il s'agit d'une réduction de Gauss complète ou d'une réduction de Gauss sous-diagonale) la manipulation réversible

$$L_j \leftarrow L_j - \frac{a_{jc}}{a_{lc}} L_l$$

On incrémente c et l de 1 et on passe à l'étape 2.

3.1.2 Efficacité de l'algorithme

Si la matrice possède L lignes et C colonnes, le nombre maximal d'opérations pour réduire une ligne est C opérations (une opération=1 multiplication + 1 soustraction, en calculant une seule fois le quotient). Si on a déjà réduit $k-1$ colonnes, ce nombre est en fait $C-(k-1)$ puisqu'on ne calcule pas les 0 des colonnes déjà réduites. Il y a $L-1$ lignes à réduire à chaque étape et $\min(L, C)$ étapes à effectuer, on en déduit que le nombre maximal d'opérations pour réduire une matrice est $(L-1)\sum_{k=1}^{\min(L,C)}(C-(k-1))$. Pour une matrice carrée de taille n, cela fait $(n-1)\sum_{k=1}^{n}(n-(k-1)) = n^3/2 + O(n^2)$ opérations.

```
purge(n);normal((n-1)*sum((n-(k-1)),k,1,n))
```

$$\text{Nosuchvariablen}, \frac{1}{2} \cdot n^3 - \frac{1}{2} \cdot n$$

Si on effectue une réduction en-dessous de la diagonale, toujours pour une matrice carrée, on trouve $\sum_{k=1}^{n}(n-(k-1))(n-k) = n^3/3 + O(n^2)$.

```
normal(sum((n-(k-1))*(n-k),k,1,n))
```

$$\frac{1}{3} \cdot n^3 - \frac{1}{3} \cdot n$$

3.1.3 Erreurs d'arrondis du pivot de Gauss

Comme $|a_{jc}| \leq |a_{lc}|$, une étape de réduction multiplie au plus l'erreur absolue des coefficients par 2. Donc la réduction complète d'une matrice peut multiplier au pire l'erreur absolue sur les coefficients par 2^n (où n est le nombre d'étapes de réduction, inférieur au plus petit du nombre de lignes et de colonnes). Ceci signifie qu'avec la précision d'un double, on peut au pire perdre toute précision pour des matrices pas si grandes que ça ($n = 52$). Heureusement, il semble qu'en pratique, l'erreur absolue ne soit que très rarement multipliée par un facteur supérieur à 10, par exemple si on multiplie une matrice n, n à coefficients flottants aléatoires dans $[0, 1]$ selon la loi uniforme par son inverse, on trouve une erreur absolue très faible même pour des valeurs de n largement plus grandes que 100.

```
n:=300; a:=ranm(n,n,0..1):; b:=inv(a):;  maxnorm(a*b-idn(a))
```

1

Par contre, si on ne prend pas la précaution de choisir le pivot de norme maximale dans la colonne, les erreurs d'arrondis se comportent de manière bien moins bonnes, comme le montre l' :

Exemple

Soit à résoudre le système linéaire

$$\epsilon x + 1.0y = 1.0, \quad x + 2.0y = 3.0$$

avec $\epsilon = 2^{-54}$ (pour une machine utilisant des doubles pour les calculs en flottant, plus généralement on choisira ϵ tel que $(1.0 + 3\epsilon) - 1.0$ soit indistinguable de 0.0).

Si on résoud le système exactement, on obtient $x = 1/(1 - 2\epsilon)$ (environ 1) et $y = (1 - 3\epsilon)/(1 - 2\epsilon)$ (environ 1). Supposons que l'on n'utilise pas la stratégie du pivot partiel, on prend alors comme pivot ϵ, donc on effectue la manipulation de ligne $L_2 \leftarrow L_2 - 1/\epsilon L_1$ ce qui donne comme 2ème équation $(2.0 - 1.0/\epsilon)y = 3.0 - 1.0/\epsilon$. Comme les calculs sont numériques, et à cause des erreurs d'arrondis, cette 2ème équation sera remplacée par $(-1.0/\epsilon)y = -1.0/\epsilon$ d'où $y = 1.0$, qui sera remplacé dans la 1ère équation, donnant $\epsilon x = 1.0 - 1.0y = 0.0$ donc $x = 0.0$.

Inversement, si on utilise la stratégie du pivot partiel, alors on doit échanger les 2 équations $L'_2 = L_1$ et $L'_1 = L_2$ puis on effectue $L_2 \leftarrow L'_2 - \epsilon L'_1$, ce qui donne $(1.0 - 2.0\epsilon)y = 1.0 - 3.0\epsilon$, remplacée en raison des erreurs d'arrondi par $1.0 * y = 1.0$ donc $y = 1.0$, puis on remplace y dans L'_1 ce qui donne $x = 3.0 - 2.0y = 1.0$.

On observe dans les deux cas que la valeur de y est proche de la valeur exacte, mais la valeur de x dans le premier cas est grossièrement eloignée de la valeur correcte. On peut aussi s'intéresser à la sensibilité de la solution d'un système linéaire à des variations de son second membre. Cela fait intervenir le nombre de conditionnement de la matrice A (voir plus bas) du système (qui est essentiellement la valeur absolue du rapport de la valeur propre la plus grande sur la valeur propre la plus petite), plus ce nombre est grand, plus la solution variera (donc plus on perd en précision).

3.2 Applications de Gauss

3.2.1 Base d'un sous-espace

On réduit la matrice des vecteurs écrits en ligne, puis on prend les lignes non nulles, dont les vecteurs forment une base du sous-espace vectoriel engendré par les vecteurs du départ. Exemple : base du sous-espac engendré par $(1,2,3),(4,5,6),(7,8,9)$. On réduit la matrice, la 3ème ligne est nulle donc on ne garde que les 2 premières lignes $(1,0,-1),(0,1,2)$ (remarque : une réduction sous-diagonale aurait suffi).

3.2.2 Déterminant

On réduit la matrice (carrée) en notant le nombre d'inversions de ligne, et on fait le produit des coefficients diagonaux, on change le signe si le nombre d'inversions de lignes est impair.

3.2.3 Réduction sous forme échelonnée (rref)

On réduit la matrice puis on divise chaque ligne par son premier coefficient non nul. Si la matrice représentait un système linéaire inversible on obtient la matrice identité sur les colonnes sauf la dernière et la solution en lisant la dernière colonne. La relation conservée est en effet $Ax = b$ où x est la solution de l'équation, et à la fin de la réduction $A = I$. Par exemple pour résoudre le système

$$\begin{cases} x + 2y + 3z &= 5 \\ 4x + 5y + 6z &= 0 \\ 7x + 8y &= 1 \end{cases}$$

on réduit sous forme échelonnée la matrice

```
rref([[1,2,3,5],[4,5,6,0],[7,8,0,1]])
```

$$\begin{pmatrix} 1 & 0 & 0 & -9 \\ 0 & 1 & 0 & 8 \\ 0 & 0 & 1 & \frac{-2}{3} \end{pmatrix}$$

ce qui donne `[[1,0,0,-9],[0,1,0,8],[0,0,1,-2/3]]`, d'où la solution $x = -9, y = 8, z = -2/3$.

3.2.4 Inverse

On accolle la matrice identité à droite de la matrice à inverser. On effectue la réduction sous forme échelonnée, on doit obtenir à droite l'identité si la matrice est inversible, on a alors à gauche la matrice inverse. La relation conservée est en effet $Ax = B$ où x est l'inverse de la matrice de départ, et en fin de réduction $A = I$. Par exemple, pour calculer l'inverse de `[[1,2,3],[4,5,6],[7,8,0]]`, on réduit :

```
rref([[1,2,3,1,0,0],[4,5,6,0,1,0],[7,8,0,0,0,1]])
```

$$\begin{pmatrix} 1 & 0 & 0 & \frac{-16}{9} & \frac{8}{9} & \frac{-1}{9} \\ 0 & 1 & 0 & \frac{14}{9} & \frac{-7}{9} & \frac{2}{9} \\ 0 & 0 & 1 & \frac{-1}{9} & \frac{2}{9} & \frac{-1}{9} \end{pmatrix}$$

3.2.5 Noyau

On réduit la matrice sous forme échelonnée. Puis on introduit des lignes de 0 afin que les 1 en tête de ligne se trouvent sur la diagonale de la matrice. On enlève ou on rajoute des lignes de 0 à la fin pour obtenir une matrice carrée. Une base du noyau est alors formée en prenant chaque colonne correspondant à un 0 sur la diagonale, en remplaçant ce 0 par -1. On vérifie qu'on obtient bien 0 en faisant le produit de ce vecteur par la matrice réduite. De plus les vecteurs créés sont clairement linéairement indépendants (puisqu'ils sont échelonnés), et il y en a le bon nombre (théorème noyau-image). Exemple : calcul du noyau de `[[1,2,3,4],[1,2,7,12]]`, on réduit la matrice avec rref, ce qui donne `[[1,2,0,-2],[0,0,1,2]]`, on ajoute une ligne de 0 entre ces 2 lignes pour mettre le 1 de la 2ème ligne sur la diagonale ce qui donne `[[1,2,0,-2],[0,0,0,0],[0,0,1,2]]`, puis on ajoute une ligne de 0 à la fin pour rendre la matrice carrée. On obtient ainsi le système équivalent de matrice :
`[[1,2,0,-2],[0,0,0,0],[0,0,1,2],[0,0,0,0]]`
La 2ème colonne donne le premier vecteur de la base du noyau, $(2, -1, 0, 0)$, la 4ème colonne donne le deuxième vecteur $(-2, 0, 2, -1)$, on vérifie aisément que ces 2 vecteurs forment une famille libre du noyau, donc une base car la dimension du noyau est égale à 2 (dimension de l'espace de départ moins le rang de la matrice, c'est-à-dire le nombre de lignes non nulles de la matrice réduite).

3.3 La méthode de factorisation LU

Dans sa forme la plus simple, elle permet d'écrire une matrice A comme produit de deux matrices triangulaire inférieures et supérieures, ce qui ramène la résolution de système à la résolution de deux systèmes triangulaires. Pour tenir compte d'éléments diagonaux nuls et pour optimiser les erreurs d'arrondi, il est nécessaire d'effectuer des permutations sur les lignes de la matrice.

3.3.1 Interprétation matricielle du pivot de Gauss

On notera l et c le nombre de lignes et colonnes de A (pour éviter la confusion avec le facteur L) et on supposera A non singulière pour simplifier l'exposition. Lorsqu'on réduit la colonne j d'une matrice $\tilde{A}$ (partiellement réduite) à partir de la ligne $j+1$ (en supposant $\tilde{A}_{j,j} \neq 0$), cela revient à multiplier $\tilde{A}$ à gauche par une matrice $\tilde{L}_j$ créée en partant de la matrice identité de taille l où on remplace les 0 colonne j, lignes $j+1$ à l par le coefficient de la combinaison de ligne effectuée :

$$l_i \to l_i - \frac{\tilde{A}_{i,j}}{\tilde{A}_{j,j}} l_j$$

donc :

$$\tilde{L}_j = \begin{pmatrix} 1 & ... & 0 & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... & 0 \\ 0 & ... & 1 & 0 & ... & ... & 0 \\ 0 & ... & 0 & 1 & ... & ... & 0 \\ 0 & ... & 0 & -\frac{\tilde{A}_{j+1,j}}{\tilde{A}_{j,j}} & 1 & ... & 0 \\ 0 & ... & 0 & .... & 0 & ... & 0 \\ 0 & ... & 0 & -\frac{\tilde{A}_{l,j}}{\tilde{A}_{j,j}} & 0 & ... & 1 \end{pmatrix}$$

On vérifie facilement que l'inverse de cette matrice est

$$L_j = \tilde{L}_j^{-1} = \begin{pmatrix} 1 & ... & 0 & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... & 0 \\ 0 & ... & 1 & 0 & ... & ... & 0 \\ 0 & ... & 0 & 1 & ... & ... & 0 \\ 0 & ... & 0 & \frac{\tilde{A}_{j+1,j}}{\tilde{A}_{j,j}} & 1 & ... & 0 \\ 0 & ... & 0 & .... & 0 & ... & 0 \\ 0 & ... & 0 & \frac{\tilde{A}_{l,j}}{\tilde{A}_{j,j}} & 0 & ... & 1 \end{pmatrix}$$

Donc A est le produit des matrices L_j par une matrice réduite U qui est triangulaire supérieure

$$A = L_1 ... L_{l-1} U$$

On vérifie ensuite que le produit des matrices $L_1 ... L_{l-1}$ revient à remplacer les coefficients de la colonne j sous la diagonale par ceux de L_j, ce qui donne une matrice L triangulaire inférieure (avec des 1 sur la diagonale). Pour l'obtenir il suffit au cours de l'algorithme de réduction sous-diagonale du pivot de Gauss de stocker le coefficient de la combinaison linéaire dans une matrice initialisée à la matrice identité (on peut aussi le faire en place dans la matrice á réduire). Attention, le produit $\tilde{L}_{l-1} ... \tilde{L}_1$ ne s'obtient pas en copiant la colonne j de $\tilde{L}_j$ pour j variant de 1 à $l-1$! On peut l'obtenir en faisant une réduction sous-diagonale de la matrice bloc obtenue en collant A avec la matrice identité ayant l lignes.

3.3.2 Factorisation $PA = LU$

Si on veut mettre en oeuvre la stratégie du pivot partiel (ou en calcul exact si un coefficient diagonal est nul), il est nécessaire d'intervertir une ligne de la matrice partiellement réduite avec une ligne en-dessous. Cela revient à réduire la matrice A de départ après échange de ces mêmes lignes. En conséquence ce n'est pas A qui est le produit LU mais une matrice obtenue par permutations de lignes de A, que l'on peut écrire comme produit à gauche de A par une matrice de permutation P. Remarque : si à une étape de réduction, tous les coefficients de la colonne j à partir de la ligne j sont nuls, on peut simplement ignorer cette colonne, on remplit la colonne j de la matrice L_j avec la colonne $j+1$ de A à partir de la ligne j.

3.3.3 Applications

On peut résoudre des systèmes linéaires par la factorisation LU. En effet soit à résoudre $Ax = b$. On effectue la permutation de lignes sur A et b (correspondant à la matrice de permutation P), ce qui donne $PAx = Pb = LUx$, puis on résoud $Ly = Pb$ (système triangulaire inférieur), puis on résoud $Ux = y$ (système triangulaire supérieur). Comparaison avec la réduction complète sous forme échelonnée de $(A|b)$:

- La factorisation LU peut reservir plus tard pour résoudre le même système linéaire avec un autre second membre. Avec `rref` il faut dès le départ mettre tous les vecteurs colonnes second membre à A.
- Le nombre d'opérations pour résoudre un système n, n est moindre. La réduction sous-diagonale nécessite de réduire les colonnes j de 1 à $n-1$, avec pour réduire la colonne j $n-j$ combinaisons linéaire de lignes ayant $n+1-j$ coefficients non nuls, soit $\sum_{j=1}^{(} n-1)(n-j)(n+1-j) = 1/3n^3 + O(n^2)$ opérations (1 opération = 1 multiplication et 1 soustraction). La résolution des systèmes triangulaires est en $O(n^2)$.
- Le calcul est plus favorable au cache mémoire, puisqu'on travaille sur une portion de plus en plus petite de la matrice.

```
n:=500; a:=ranm(n,n,0..1):; p,l,u:=lu(a):;b:=randvector(n):;
 time(c:=linsolve(p,l,u,b)); maxnorm(a*c-b);time(d:=linsolve(a,b));
 maxnorm(a*d-b);
```

1

On peut inverser une matrice en utilisant la décomposition LU. Supposons pour simplifier que la permutation est l'identité. On calcule d'abord L^{-1} en utilisant le fait que L est triangulaire inférieure, voici comment cela est implémenté dans Xcas (L est noté `l`) :

```
first step compute l^-1,
solve l*a=y for y a canonical basis vector
  a0=y0, a1=y1-l_{1,0}*a0, ..., ak=yk-sum_{j=0..k-1}(l_kj*aj)
if y=(0,..,0,1,0,...0) (1 at position i),
  a0=..=a_{i-1}=0, a_i=1 and we start at equation k=i+1 and sum_{j=i...}
-> n^3/6 operations
To store the result in place of l
we first compute all the a2 (there is only 1), then all the a3 (2), etc.
a0=y0, a1=y1-l_{1,0}*a0, ..., ak=yk-sum_{j=0..k-1}(l_kj*aj)
```

Puis on résoud $UA^{-1} = L^{-1}$ colonne par colonne

```
second step, solve u*inverse=l^-1 (now under the diagonal)
we compute a column of inverse by solving the system:
u*col(inverse)=corresponding row of l^-1,
and overwrite the row of l^-1 by solution
u*[x0,..,xn-1]=[a0,...,an]
x_{n-1}=a_{n-1}/u_{n-1,n-1}
x_{n-2}=(a_{n-2}-u_{n-2,n-1}*x_{n-1})/u_{n-2,n-2}
...
x_k=(a_{k}-sum_{j=k+1..n-1} u_{k,j}x_j)/u_{k,k}
-> n^3/2 operations
To store the solution in place, we first compute all the x_{n-1}
put them in the last line of m, then all the x_{n-2}, etc.
```

3.4 La factorisation de Cholesky

Dans le cas où la matrice est réelle symétrique définie positive (ou plus généralement hermitienne), on peut obtenir une écriture analogue mais où U est la transconjuguée de L

$$A = U^*U = LL^*$$

L reste triangulaire inférieure, mais n'a plus des 1 sur sa diagonale en général. Si A est définie positive, on peut rendre l'écriture unique en imposant aux coefficients diagonaux de L d'être réels positifs. L'algorithme de calcul de U est la traduction matricielle de l'algorithme de Gauss de réduction des formes quadratiques. On a en effet

$$x^*Ax = x^*U^*Ux = ||Ux||^2$$

les lignes de U (ou les colonnes de L) sont donc les coefficients des formes linéaires indépendantes qui interviennent dans l'écriture de la forme quadratique comme somme/différence de carrés de formes linéaires. Si A est définie positive, seules des sommes interviennent, et les variables s'éliminent l'une après l'autre (le coefficient de x_2 est forcément non nul lorsqu'on a éliminé x_1 et ainsi de suite), ceci explique la forme triangulaire de U et L. Le calcul de L se fait donc colonne par colonne, en calculant d'abord le coefficient diagonal comme racine carrée du coefficient diagonal $\alpha_j = \sqrt{A_{j,j}}$. Ensuite on effectue les combinaisons de ligne sous la forme

$$l_j \to \frac{1}{\alpha_j} l_j, \quad l_i \to \alpha_j l_i - \frac{A_{i,j}}{\alpha_j} l_j$$

On peut aussi tout simplement effectuer le produit de LL^* et chercher les inconnues en commencant par $l_{1,1}$ puis on calcule les $l_{i,1}$ pour $i > 1$, etc. En suivant wikipedia, pour une matrice réelle :

$$L = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix}$$

$$a_{ij} = \left(LL^T\right)_{ij} = \sum_{k=1}^{n} l_{ik} l_{jk} = \sum_{k=1}^{\min\{i,j\}} l_{ik} l_{jk}, \ 1 \leq i, j \leq n$$

La matrice A étant symétrique, il suffit que les relations ci-dessus soient vérifiées pour $i \leq j$, c'est-à-dire que les éléments $l_{i,j}$ de la matrice L doivent satisfaire

$$a_{ij} = \sum_{k=1}^{i} l_{ik} l_{jk}, \ 1 \leq i \leq j \leq n$$

Pour $i = 1$, on détermine la première colonne de L

$$a_{11} = l_{11} l_{11}, \quad a_{1j} = l_{11} l_{j1}$$

donc

$$l_{11} = \sqrt{a_{11}}, \quad l_{j1} = \frac{a_{1j}}{l_{11}} \text{ (pour } j > 1)$$

On détermine la i-ième colonne de L ($2 \leq i \leq n$) après avoir calculé les $i-1$ premières colonnes

$$a_{ii} = l_{i1} l_{i1} + \ldots + l_{ii} l_{ii}, \quad a_{ij} = l_{i1} l_{j1} + \ldots + l_{ii} l_{ji}$$

d'où

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}, \quad l_{ji} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}}{l_{ii}} \text{ (pour } j > i)$$

Pour une matrice hermitienne complexe, il suffit de remplacer $l_{ik} l_{jk}$ par $l_{ik} \overline{l_{jk}}$ et l_{ik}^2 par $|l_{ik}|^2$. Le nombre d'opérations arithmétiques à effectuer est asymptotiquement 2 fois plus faible que celui pour LU. En effet, pour la première ligne, il faut 1 racine et $n-1$ divisions, pour la deuxième ligne, 1 racine, $n-1$ additions, multiplications et $n-2$ divisions, ..., pour la i-ième ligne 1 racine, $(i-1)(n-i)$ additions, multiplications et $n-2$ divisions, au final le cout est dominé par les additions et multiplications en $n^3/6$ pour chaque, contre $n^3/3$ pour la factorisation LU. Mais le temps réel dépend de l'optimisation de l'implémentation, en particulier des accès mémoire plus couteux que les opérations arithmétiques, ainsi avec Xcas, l'algorithme de Cholesky n'est pas plus rapide que la factorisation LU. La commande Xcas correspondante est `cholesky` et renvoie la matrice L.

3.5 Conditionnement

Le conditionnement mesure la sensibilité de la solution renvoyée d'un système linéaire aux données du problème.

3.5.1 Rappel sur les normes matricielles

On utilisera les normes suivantes sur l'espace vectoriel des matrices n, n :

– la norme infinie $|A|_\infty$=max$|a_{i,j}|$
```
a:=ranm(3,3); maxnorm(a)
```

$$\begin{pmatrix} -99 & 58 & 44 \\ -10 & 1 & -63 \\ 57 & 36 & 56 \end{pmatrix}, 99$$

– les normes subordonnées à une norme de $\mathbb{R}^n$ ou $\mathbb{C}^n$ dite norme triple

$$|||A||| = \max_{x/\|x\|=1} \|Ax\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|}$$

Xcas reconnait trois normes triples, subordonnées aux normes L^1, L^2 (euclidienne) et L^∞ sur $\mathbb{R}^n$ ou $\mathbb{C}^n$
```
matrix_norm(a,1); matrix_norm(a,2); matrix_norm(a,inf)
```

$$166, 101.590207227, 201$$

– la norme de Frobénius $\sqrt{\sum_{i,j} |a_{i,j}|^2}$
```
frobenius_norm(a)
```

$$14 \cdot \sqrt{137}$$

Si on prend comme norme la norme L^2, le calcul de $|||A|||$ nécessite de maximiser $\sqrt{< Ab|Ab >}$ pour b de norme 1, ce qui revient à maximiser $\sqrt{< b|A^*Ab >}$. En diagonalisant la matrice hermitienne A^*A, on voit qu'il suffit d'en trouver la plus grande valeur propre et d'en prendre la racine carrée. Les valeurs propres de A^*A sont appelées valeurs singulières de A (ce sont des réels positifs). Le calcul de la normes triple infinie est plus simple, si $\|x\|_\infty = 1$

$$|(Ax)_i| = |\sum_j a_{i,j}x_j| \leq \sum_j |a_{i,j}|$$

avec égalité si les x_j valent ± 1 avec le bon choix de signe. On en déduit que

$$|||A|||_\infty = \max_i \sum_j |a_{i,j}|$$

aussi appelé norme de ligne (on prend le max sur les lignes de la somme des valeurs absolues des coefficients de la ligne). Pour la norme 1, on trouve la norme colonne (max sur les colonnes de la somme des valeurs absolues des coefficients d'une colonne) :

$$\sum_i |(Ax)_i| = \sum_i |\sum_j a_{i,j}x_j| \leq \sum_j |x_j| \sum_i |a_{i,j}| \leq \max_j \sum_i |a_{i,j}|$$

Pour réaliser la norme triple, on prend $x_i = 0$ si $i \neq j$ qui réalise le max, et $x_j = 1$.

3.5.2 Nombre de condition

Soit le système linéaire $Ax = b$ de solution $x = A^{-1}b$, supposons b connu avec une erreur e, alors la solution renvoyée sera $x + A^{-1}e$, on a donc une erreur relative sur la solution de

$$\frac{\|A^{-1}e\|}{\|A^{-1}b\|} = \frac{\|A^{-1}e\|}{\|e\|}\frac{\|e\|}{\|b\|}\frac{\|b\|}{\|A^{-1}b\|} \leq |||A^{-1}|||\frac{|e|}{|b|}|||A|||$$

(la dernière inégalité s'obtient en écrivant $b = A(A^{-1}b)$). On en déduit que le rapport de l'erreur relative sur la solution par l'erreur relative du second membre est majorée par le produit de la norme de A (en tant qu'application linéaire) par la norme de A^{-1}, ce produit est appelé conditionnement de la matrice A (ou parfois nombre de condition de A en adoptant la terminologie anglo-saxonne). On remarquera que le conditionnement dépend du choix de la norme sur l'espace vectoriel. Pour la norme 2, on sait que $|||A|||$ est la plus grande valeur singulière de A le même calcul pour A^{-1} (dont les valeurs singulières sont les inverses des valeurs singulières de A) nous donne alors le :

Théorème 2 *Lorsqu'on résoud un système linéaire $Ax = b$, A matrice connue précisément et inversible, b connu avec une erreur relative en norme L^2, l'erreur relative en norme L^2 sur x est au plus multipliée par*

$$K_2(A) = \frac{\lambda_n}{\lambda_1}$$

oú λ_n [resp. λ_1] est la plus grande [resp. plus petite] valeur singulière de A (racines carrées des valeurs propres de A^*A).
Ce facteur d'amplification des erreurs relatives est évidemment supérieur ou égal à 1. Il est égal à 1 si la matrice est unitaire (puisque A est une matrice d'isométrie ou car $AA^* = I$). S'il est de l'ordre de 2^c on perdra (au plus) c bits de précision sur la mantisse de x.

Avec Xcas, les valeurs singulières s'obtiennent par l'instruction `SVL(A)` , le conditionnement L^2 par `cond(A,2)` (`cond(A,1)` donne le conditionnement L^1 et `cond(A,inf)` donne le conditionnement L^∞). Attention, les valeurs singulières de A ne sont pas les valeurs absolues des valeurs propres de A (c'est le cas si A commute avec sa transconjuguée mais ce n'est pas général). On peut utiliser la méthode de la puissance (cf. infra) pour estimer rapidement la plus grande valeur singulière de A (donc sans diagonaliser complètement la matrice A^*A), et de même sur A^{-1} (en utilisant LU ou Cholesky pour trouver les itérées sans calculer A^{-1}). On remarque que si A est une matrice orthogonale, alors son nombre de conditionnement vaut 1, si on doit utiliser des matrices orthogonales dans un algorithme, ce sera stable numériquement. On peut aussi prendre la norme L^1 sur l'espace vectoriel, dans ce cas la norme de matrice correspondante est la norme de colonne (exercice !), le maximum des sommes valeurs absolues des éléments des colonnes (`colNorm(A)` en Xcas) et le conditonnement est le produit de `colNorm(A)` par `colNorm(inv(A))` qui est renvoyé par `COND(A)` en Xcas. Si la matrice du système A (de nombre de condition noté $\kappa(A)$) est elle-même connue avec une certaine incertitude, alors pour $||\Delta A||$ suffisamment petit, la solution de $(A + \Delta A)(x + \Delta x) = b + \Delta b$ vérifie

$$\frac{|\Delta x|}{|x|} \leq \frac{\kappa(A)}{1 - \kappa(A)\frac{||\Delta A||}{||A||}}\left(\frac{|\Delta b|}{|b|} + \frac{||\Delta A||}{||A||}\right)$$

En effet, on a

$$A\Delta x = \Delta b - \Delta A(x + \Delta x) \Rightarrow \Delta x = A^{-1}(\Delta b - \Delta A(x + \Delta x))$$

donc en norme

$$\|\Delta x\| \leq |||A^{-1}|||(\frac{\|\Delta b\|}{\|b\|}\|Ax\| + \frac{|||\Delta A|||}{|||A|||}|||A|||(\|x\| + \|\Delta x\|))$$

puis :

$$\frac{\|\Delta x\|}{\|x} \leq \kappa(A)(\frac{\|\Delta b\|}{\|b\|} + \frac{|||\Delta A|||}{|||A|||}(1 + \frac{\|\Delta x\|}{\|x})$$

3.6 Quelques méthodes alternatives au pivot

3.6.1 Factorisation QR

La factorisation QR consiste à écrire une matrice A comme produit d'une matrice orthogonale (ou unitaire dans $\mathbb{C}$) et d'une matrice triangulaire supérieure. Les matrices orthogonales ayant un conditionnement de 1 (en norme L^2) cette factorisation peut s'obtenir de manière numériquement stable. Il existe plusieurs algorithmes pour effectuer cette factorisation. On peut voir cette factorisation comme l'orthonormalisation de Gram-Schmidt appliqué aux vecteurs colonnes de A si A est non singulière. Mais la procédure de Gram-Schmidt n'est pas numériquement stable (car on retranche d'un vecteur une combinaison linéaire des vecteurs précdents pour rendre le nouveau vecteur orthogonal et les erreurs d'arrondi s'accumulent rendant l'orthogonalité plus aléatoire). La méthode de Householder utilise des matrices de symétrie par rapport à un hyperplan et ne souffre pas de ce problème d'instabilité. Pour annuler les coefficients de la première colonne c_1 de A, on construit le vecteur $u = c_1 \pm \|c_1\|e_1$ où e_1 est le premier vecteur de base et le signe $\pm$ est le signe de la première composante de c_1 (pour assurer la stabilité numérique). On fait alors la symétrie Q_1 par rapport à l'hyperplan H orthogonal à u, qui laisse H invariant et transforme u en $-u$. Comme u est vecteur directeur de la bissectrice intérieure ou extérieure de c_1 et e_1, la symétrie échange ces deux vecteurs, éventuellement au signe près. La matrice $A_1 = Q_1A$ a donc comme première colonne un multiple de e_1, on continue ensuite en faisant le même raisonnement sur la matrice A_1 en se limitant et lignes et colonnes d'indice ≥ 2. Après $n-1$ itérations, on a $A_{n-1} = Q_{n-1}...Q_1A$ qui est triangulaire supérieure d'où la factorisation annoncée. Matriciellement, $Q_1 = I - 2vv^*$ où $v = u/\|u\|$, pour calculer Q_1A il faut effectuer $A - 2vv^*A$, on calcule donc $w = v^*A$ en n^2 opérations (ou une opération est une addition et une multiplication) puis on soustrait $(2v)_i w_j$ de a_{ij} en n^2 opérations. En faisant de même aux étapes qui suivent, sans tenir compte de la simplification progressive du vecteur v, on effectue $2n^3$ opérations. La constante 2 peut être un peu améliorée en tenant compte des 0 initiaux de v aux étapes 2 et ultérieures, elle est toutefois supérieure à LU (et Cholesky), mais en contrepartie la méthode est très stable numériquement. On peut aussi utiliser des rotations (méthode de Givens) pour annuler les coefficients de A sous la diagonale. C'est par exemple très efficace pour des matrices tridiagonales. Lorsque la matrice A n'est pas carrée, mais possède n lignes et c colonnes, la factorisation QR est encore possible, la matrice Q est alors une matrice carrée d'ordre n, et R a les mêmes dimensions que A. En particulier si A a plus de lignes que de colonnes (matrice verticale), R aussi. Par exemple si A est de rang maximal c, alors R se décompose en un premier bloc c, c inversible R_1 et un deuxième bloc $n-c, c$ entièrement nul. **Applications :**
On peut alors écrire $Ax = b$ sous la forme $QRx = b$ donc $Rx = Q^*b$ qui est un système triangulaire supérieur, donc résoudre $Ax = b$ en $O(n^2)$ opérations une fois la factorisation QR effectuée.

```
a:=[[1.,2.,3.],[4.,5.,6.],[7.,8.,0.]]; q,r:=qr(a);
```

$$\begin{pmatrix} 1.0 & 2.0 & 3.0 \\ 4.0 & 5.0 & 6.0 \\ 7.0 & 8.0 & 0.0 \end{pmatrix}, \begin{pmatrix} -0.123091490979 & 0.904534033733 & 0.408248290464 \\ -0.492365963917 & 0.301511344578 & -0.816496580928 \\ -0.861640436855 & -0.301511344578 & 0.408248290464 \end{pmatrix}, \begin{pmatrix} -8.12403840464 & -9.601136296 \\ 0 & 0.90453403373 \\ 0 & 0 \end{pmatrix}$$

Mais c'est surtout pour résoudre **au sens des moindres carrés** un système sur-déterminé que la factorisation QR trouve tout son intérêt. Soit A la matrice d'un système sur-déterminé avec n lignes et c colonnes, $n > c$ (matrice "verticale" ou mince). Le système n'a en général pas de solution, on cherche alors à minimiser $\|Ax - b\|_2^2$. Ceci revient à chercher la projection orthogonale de b sur Im(A). Soit Ax cette projection, on a $b = Ax + p$ avec p orthogonal à Im(A) donc dans Ker$(A*)$, donc $A^*b = A^*(Ax + p) = A^*Ax$.

Proposition 3 *La solution x du problème de minimisation de $\|Ax - b\|_2$ est donnée par $A^*Ax = A^*b$*

Si $A = QR$ on a alors $R^*Rx = R^*Q^*b$. Si A est de rang maximal c, on décompose R en $\binom{R_1}{R_2}$ avec R_1 inversible de taille c et $R_2 = 0$, donc $R^* = (R_1^*, 0)$ et le système devient $R_1^*R_1x = (R_1^*, 0)Q^*b$ et comme R_1 est inversible, on a $R_1x =$ les c premières lignes de Q^*b. Il vaut mieux résoudre ce système que $A^*Ax = A^*b$ car le conditionnement de A^*A est le carré du conditionnement de A (le nombre d'opérations est un peu supérieur : QR a une constante nettement plus grande que Cholesky mais il n'y a pas de multiplication de matrice à faire, et le résultat est plus précis). Notons qu'on peut résoudre de manière assez analogue un système sous-déterminé au sens des moindres carrés. On considère toujours le système $Ax = b$ mais cette fois-ci A est une matrice "horizontale" (nombre de lignes $n < c$ le nombre de colonnes). Si $b \in$Im(A) (on peut génériquement supposer que A est de rang n), il y a une infinité de solutions, on cherche alors la solution de norme minimale. On cherche donc le projeté orthogonal de l'espace affine des solutions (dirigé selon Ker(A)) sur l'orthogonal de Ker(A) qui est ImA^*. Ainsi $x = A^*y$ vérifie $Ax = b$, donc $AA^*y = b$. Si A est de rang n alors AA^* est inversible et $y = (AA^*)^{-1}b$ donc $x = A(AA^*)^{-1}b$. En utilisant la factorisation QR de A^* (attention pas celle de A), on a $AA^* = R^*R$, donc $x = QR(R^*R)^{-1}b$. Si A est de rang n, on a alors en posant $R = \binom{R_1}{0}$ avec R_1 matrice n, n inversible :

$$x = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} R_1^{-1} {R_1^*}^{-1} b = Q \begin{pmatrix} {R_1}^{*^{-1}} \\ 0 \end{pmatrix} b = Q_1 {R_1}^{*^{-1}} b$$

où Q_1 désigne les n premières colonnes de Q.

Proposition 4 *Soit A une matrice ayant n lignes et $c > n$ colonnes. Si A est de rang maximal n, posons $A^* = QR$, R_1 les n premières lignes de R_1, Q_1 les n premières colonnes de Q. La solution de $Ax = b$ de norme minimale est donnée par Q_1c où c est la solution de $R_1^*c = b$ (système triangulaire).*

Exemple 1 : la régression linéaire :
On se donne n points de coordonnées (x_i, y_i) où les x_i sont distincts 2 à 2 et on cherche la droite qui approche le mieux ces points au sens de l'écart en y au carré. Il s'agit donc de trouver α et β qui minimisent :

$$\sum_{j=1}^{n} (\alpha + \beta x_i - y_i)^2$$

Ici on a :

$$A = \begin{pmatrix} 1 & x_1 \\ ... & ... \\ 1 & x_n \end{pmatrix}, \quad x = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \quad b = \begin{pmatrix} y_1 \\ ... \\ y_n \end{pmatrix}$$

Pour trouver α, β de manière précise, on préferera donc calculer la factorisation QR de A plutôt que calculer A^*A, R_1 est une matrice carrée d'ordre 2, Q une matrice carrée d'ordre n, la première colonne de Q est le vecteur de coordonnées $\pm(1, ..., 1)/\sqrt{n}$. Exemple : on se donne les points $(1, 2), (2, 1), (3, 5), (4, 4)$. La matrice A et b sont donc

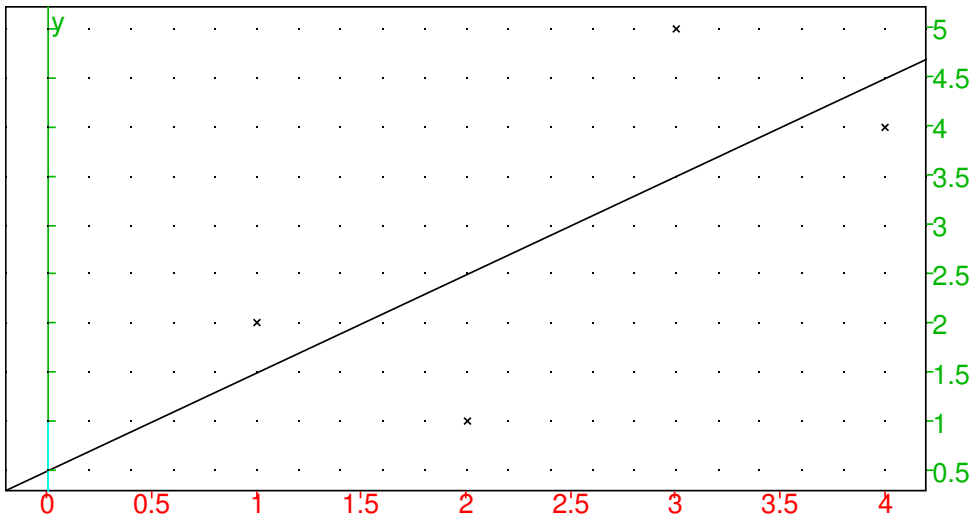
```
A:=evalf(tran([seq(1,4),seq(j,j,1,4)])); b:=[2,1,5,4];
```

$$\begin{pmatrix} [1.0,1.0] & [1.0,2.0] & [1.0,3.0] & [1.0,4.0] \\ 2 & 1 & 5 & 4 \end{pmatrix}$$

```
q,r:=qr(A); y0,m:=normal(r[0..1,0..1]^-1*trn(q)[0..1]*b);
```

$$\begin{pmatrix} -0.5 & 0.67082039325 \\ -0.5 & 0.22360679775 \\ -0.5 & -0.22360679775 \\ -0.5 & -0.67082039325 \end{pmatrix}, \begin{pmatrix} -2.0 & -5.0 \\ 0 & -2.2360679775 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, [0.5, 1.0]$$

```
purge(x);droite(y=m*x+y0); scatterplot(seq(j,j,1,4),b);linear_regression_plot(seq(j,j,1,
```

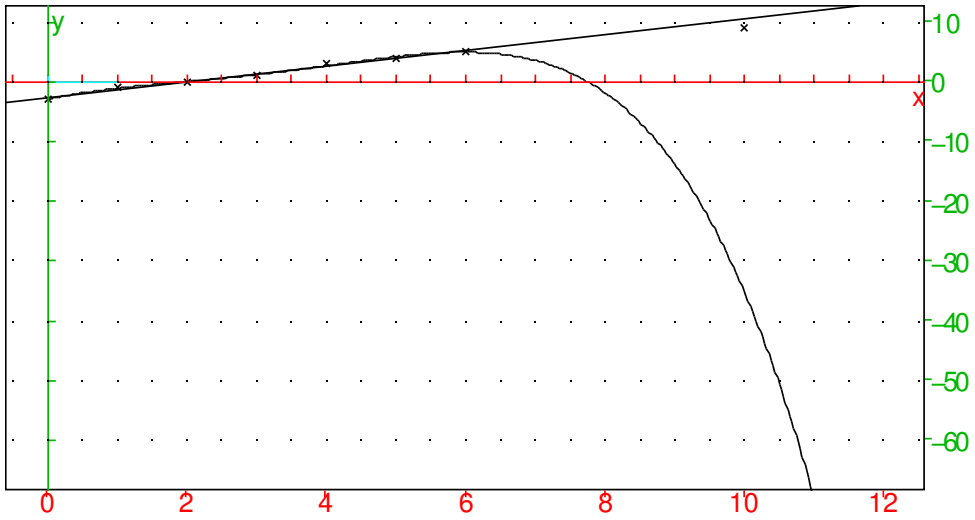


Exemple 2 : régression polynomiale :
C'est une généralisation de l'exemple précédent, on cherche un polynôme $P(x)$ de degré fixé tel que $\sum_i (y_i - P(x_i))^2$ soit minimal. Si on choisit la base canonique des polynômes, la matrice A est alors une matrice de Vandermonde et son conditionnement est en général très mauvais. De plus, plus le degré augmente, plus l'extrapolation est hasardeuse :

```
a:=[[0,-3],[1,-1],[2,0],[3,1],[4,3],[5,4],[6,5],[10,9]]:;
```

Done

```
scatterplot(a); linear_regression_plot(a[0..6]); polynomial_regression_plot(a[0..6],4);
```



On préfèrera utiliser une base de polynômes orthogonaux pour le produit scalaire $\sum_i P(x_i)Q(x_i)$ ou minimiser une autre fonctionnelle quadratique, par exemple une intégrale du carré de la différence entre une fonction à approcher et un polynôme ou un polynôme trigonométrique. **Remarque** : La factorisation QR peut s'obtenir en calculant la factorisation de Cholesky de A^*A qui donne R, mais on lui préfère une méthode de calcul direct pour des raisons de stabilité, en tout cas pour des matrices denses (pour des matrices creuses des considérations d'efficacité peuvent faire préférer la factorisation de Cholesky).

3.6.2 Jacobi, Gauss-Seidel, relaxation

Lorsqu'on a une matrice creuse (peu d'éléments non nuls), l'algorithme du pivot de Gauss a tendance à densifier rapidement la matrice réduite (surtout avec le pivot partiel où on ne controle pas le nombre de zéros de la ligne contenant le pivot). Il peut alors être intéressant d'utiliser des méthodes alternatives ne faisant intervenir que des produits de matrice, donnant éventuellement un résultat seulement approché. Par exemple pour calculer l'inverse d'une matrice $A = M - N$, avec M facile à inverser (par exemple diagonale) et N petit en norme et creuse, on peut écrire :

$$A^{-1} = (M-N)^{-1} = (M(I - M^{-1}N))^{-1} = (I + M^{-1}N + (M^{-1}N)^2 + ...)M^{-1}$$

De même pour résoudre un système linéaire $Ax = b$ avec $A = M - N$, on considère la suite $Mx_{n+1} - Nx_n = b$, donc x_{n+1} est obtenu en résolvant le système :

$$Mx_{n+1} = b + Nx_n, \quad x_0 = 0$$

pour laquelle on vérifiera les hypothèses du théorème du point fixe, il suffit par exemple de vérifier que la plus grande valeur singulière de $M^{-1}N$ est strictement plus petite que 1. Lorsque la matrice N n'est pas creuse, le procédé est intéressant pour résoudre approximativement un système si n est grand et si chaque itération est en $O(n^2)$ (ceci veut dire qu'on ne calcule pas M^{-1} sauf si c'est évident, par exemple si M est diagonale), mais le procédé n'est pas intéressant pour le calcul de l'inverse de A.

Notons D la partie diagonale de A, L sa partie triangulaire inférieure stricte, U sa partie triangulaire supérieure stricte, La méthode de Jacobi utilise pour M la diagonale D de A, alors que la méthode de Gauss-Seidel prend

pour M la partie triangulaire inférieure $D+L$ de A (diagonale comprise). Pour Jacobi, on a donc

$$x_{n+1} = D^{-1}(b + (D-A)x_n) = x_n + D^{-1}(b - Ax_n)$$

En Xcas, l'instruction préprogrammée est `jacobi_linsolve`, on peut aussi programmer la méthode par

```
jacobi(A,b,eps,N):={
  local D,x0,x1,n,j;
  n:=size(A);
  D:=diag(A).^-1;
  x0:=seq(0,n);
  pour j de 1 jusque N faire
    x1:=b-A*x0;
    si l2norm(x1)<eps alors return x0; fsi;
    x0:=x0+D .*x1;
  fpour;
  return "non convergent";
}:;
```

Un cas simple où on a convergence :

Proposition 5 *Lorsque la matrice A est à diagonale strictement dominante, c'est-à-dire que l'élément diagonal est en valeur absolue strictement supérieur à la somme des éléments non diagonaux de la même ligne :*

$$|a_{ii}| > \sum_{j\neq i} |a_{ij}|$$

la méthode de Jacobi converge.

En effet, $\|M^{-1}Nx\|_\infty < \|x\|_\infty$ car :

$$|(Nx)_i| \leq \sum_{j\neq i} |a_{ij}||x_j| \leq \sum_{j\neq i} |a_{ij}|\|x\|_\infty < |a_{ii}|\|x\|_\infty \quad \Rightarrow \quad |M^{-1}(Nx)_i| < \|x\|_\infty$$

On retrouve ce cas pour une petite perturbation d'une matrice diagonale, par exemple

```
n:=500;A:=2*idn(n)+1e-4*ranm(n,n,uniform,-1,1):;b:=seq(1,n):;
```

$$1$$

```
time(c:=linsolve(A,b));
```

$$[0.57, 0.556544931]$$

```
time(d:=jacobi(A,b,1e-12,50));
```

$$[0.075, 0.0776688295]$$

```
maxnorm(d-c)
```

$$8.27782287161e-13$$

Pour n assez grand, la méthode de Jacobi devient plus rapide. Cela se vérifie encore plus vite si A est une matrice creuse. Pour Gauss-Seidel, le calcul de M^{-1} n'est pas effectué, on résoud directement le système triangulaire $Mx_{n+1} = b + Nx_n$ soit

$$(D + L)x_{n+1} = b - Ux_n$$

Gauss-Seidel est moins adapté à la parallélisation que Jacobi. On adapte le programme précédent

```
seidel(A,b,N,eps):={
  local L,U,x0,x1,n,j;
  n:=size(A);
  L:=diag(A,left);
  U:=A-L;
  x0:=seq(0.0,n);
  pour j de 1 jusque N faire
    x1:=b-U*x0;
    x1:=linsolve(L,x1);
    si l2norm(x1-x0)<eps*l2norm(x0) alors return x0; fsi;
    x0:=x1;
  fpour;
  return "non convergent";
};
```

Dans la méthode de relaxation, on pose pour M la matrice triangulaire inférieure $M = \frac{1}{\omega}D + L$ où $\omega > 0$, donc $N = (\frac{1}{\omega} - 1)D - U$ et on utilise la récurrence $Mx_{n+1} = b + Nx_n$ donc $Mx_{n+1} = b + (M - A)x_n$ puis $M(x_{n+1} - x_n) = b - Ax_n$ puis

$$(D + \omega L)(x_{n+1} - x_n) = \omega(b - Ax_n)$$

On remarque que Gauss-Seidel correspond à $\omega = 1$. L'instruction correspondante de Xcas est `gauss_seidel_linsolve` et peut prendre en paramètre le coefficient de relaxation, dont la valeur par défaut est 1.

Proposition 6 *(Convergence) : si $A = M - N$ est une matrice symétrique définie positive et si $M^* + N$ est définie positive, alors la méthode converge.*

On utilise la norme correspondant à la forme quadratique de matrice A et on calcule la norme subordonnée de $M^{-1}N$, on a $M^{-1}Nx = x - y$ avec $y = M^{-1}Ax$ donc

$$\begin{aligned}
\|M^{-1}Nx\|_A^2 &= < x - y|A(x - y)) > \\
&= < x|Ax > + < y|Ay > - < y|Ax > - < x|Ay > \\
&= < x|Ax > + < y|Ay > - < y|My > - < My|y > \\
&= < x|Ax > - < y|(M^* + N)y > \\
&< < x|Ax >= \|x\|_A^2
\end{aligned}$$

Conséquence : si A symétrique définie positive, alors Gauss-Seidel converge, car $M^* + N = D$. Pour la relaxation, on a $M^* + N = (2/\omega - 1)D$ qui est définie positive si $\omega < 2$. **Remarque** : Jacobi et Gauss-Seidel sont implémentées dans les commandes Xcas `jacobi_linsolve` et `gauss_seidel_linsolve`.

3.6.3 Le gradient conjugué

Il s'agit de résoudre $Ax = b$, où A est définie positive. Si on a une base orthogonale pour le produit scalaire induit par A, on peut calculer la j-ième coordonnée de x dans cette base en faisant le produit scalaire de $Ax = b$ par le j-ième vecteur de la base. On construit donc petit à petit une base orthogonale pour A par un procédé à la Gram-Schmidt, mais on ne part pas de la base canonique : on construit cette famille orthogonale pour A en meme temps qu'on calcule les composantes de x. On pose $x_0 = 0$, à la i-ième itération si $Ax_i - b = 0$ on a terminé, sinon $r_i = Ax_i - b$ est linéairement indépendant des éléments de la famille orthogonale dejà construite, on complète la famille orthogonale avec un nouveau vecteur, on calcule la $i + 1$-ième composante de x sur la famille orthogonale, et on ajoute le tout à x_i pour obtenir x_{i+1}. On s'arrête en au plus la dimension itérations lorsque la famille orthogonale est devenue une base. La commande `conjugate_gradient(A,b)` de Xcas permet de faire ce calcul, on peut préciser une valeur initiale de recherche `x0` et une précision `eps` en tapant `conjugate_gradient(A,b,x0,eps)`. Voir aussi le menu `Exemple, analyse, gradconj`

4 Approximation polynomiale

On présente dans cette section quelques méthodes d'approximation de fonctions par des polynômes sur un intervalle, la section suivante présente des méthodes d'approximation près d'un point ou de l'infini.

4.1 Polynôme de Lagrange

Étant donné la facilité de manipulation qu'apportent les polynomes, on peut chercher à approcher une fonction par un polynôme. La méthode la plus naturelle consiste à chercher un polynôme de degré le plus petit possible égal à la fonction en certains points $x_0, ..., x_n$ et à trouver une majoration de la différence entre la fonction et le polynôme. Le polynome interpolateur de Lagrange répond à cette question.

4.1.1 Existence et unicité

Soit donc $x_0, ..., x_n$ des réels distincts et $y_0, ..., y_n$ les valeurs de la fonction à approcher en ces points (on posera $y_j = f(x_j)$ pour approcher la fonction f). On cherche donc P tel que $P(x_j) = y_i$ pour $j \in [0, n]$. Commencons par voir s'il y a beaucoup de solutions. Soit P et Q deux solutions distinctes du problème, alors $P - Q$ est non nul et va s'annuler en $x_0, ..., x_n$ donc possède $n + 1$ racines donc est de degré $n + 1$ au moins. Réciproquement, si on ajoute à P un multiple du polynome $A = \prod_{j=0}^{n}(X - x_j)$, on obtient une autre solution. Toutes les solutions se déduisent donc d'une solution particulière en y ajoutant un polynome de degré au moins $n+1$ multiple de A. Nous allons maintenant construire une solution particulière de degré au plus n. Si $n = 0$, on prend $P = x_0$ constant. On procède ensuite par récurrence. Pour construire le polynôme correspondant à $x_0, ..., x_{n+1}$ on part du polynoôme P_n correspondant à $x_0, ..., x_n$ et on lui ajoute un multiple réel de A

$$P_{n+1} = P_n + \alpha_{n+1} \prod_{j=0}^{n}(X - x_j)$$

Ainsi on a toujours $P_{n+1}(x_j) = y_j$ pour $j = 0, ..n$, on calcule maintenant α_{n+1} pour que $P_{n+1}(x_{n+1}) = y_{n+1}$. En remplacant avec l'expression de P_{n+1} ci-dessus, on obtient

$$P_n(x_{n+1}) + \alpha_{n+1} \prod_{j=0}^{n}(x_{n+1} - x_j) = y_{n+1}$$

Comme tous les x_j sont distincts, il existe une solution unique :

$$\alpha_{n+1} = \frac{y_{n+1} - P_n(x_{n+1})}{\prod_{j=0}^{n}(x_{n+1} - x_j)}$$

On a donc prouvé le :

Théorème 7 *Soit $n+1$ réels distincts $x_0, ..., x_n$ et $n+1$ réels quelconques $y_0, ..., y_n$. Il existe un unique polynôme P de degré inférieur ou égal à n, appelé polynome de Lagrange, tel que :*

$$P(x_i) = y_i$$

Exemple : déterminons le polynome de degré inférieur ou égal à 2 tel que $P(0) = 1, P(1) = 2, P(2) = 1$. On commence par $P_0 = 1$. Puis on pose $P_1 = P_0 + \alpha_1 X = 1 + \alpha_1 X$. Comme $P(1) = 2 = 1 + \alpha_1$ on en tire $\alpha_1 = 1$ donc $P_1 = 1 + X$. Puis on pose $P_2 = P_1 + \alpha_2 X(X-1)$, on a $P_2(2) = 3 + 2\alpha_2 = 1$ donc $\alpha_2 = -1$, finalement $P_2 = 1 + X - X(X-1)$.

```
P:=interp([0,1,2],[1,2,1]);
```

$$(-x + 1 + 1) \cdot x + 1$$

4.1.2 Majoration de l'erreur d'interpolation.

Reste à estimer l'écart entre une fonction et son polynome interpolateur, on a le :

Théorème 8 *Soit f une fonction $n+1$ fois dérivable sur un intervalle $I = [a, b]$ de $\mathbb{R}$, $x_0, ..., x_n$ des réels distincts de I. Soit P le polynome de Lagrange donné par les x_j et $y_j = f(x_j)$. Pour tout réel $x \in I$, il existe un réel $\xi_x \in [a, b]$ (qui dépend de x) tel que :*

$$f(x) - P(x) = \frac{f^{[n+1]}(\xi_x)}{(n+1)!} \prod_{j=0}^{n}(x - x_j) \tag{1}$$

Ainsi l'erreur commise dépend d'une majoration de la taille de la dérivée $n+1$-ième sur l'intervalle, mais aussi de la disposition des points x_j par rapport à x. Par exemple si les points x_j sont équidistribués, le terme $|\prod_{j=0}^{n}(x - x_j)|$ sera plus grand près du bord de I qu'au centre de I. Preuve du théorème : Si x est l'un des x_j l'égalité est vraie. Soit

$$C = (f(x) - P(x))/\prod_{j=0}^{n}(x - x_j)$$

on considère maintenant la fonction :

$$g(t) = f(t) - P(t) - C\prod_{j=0}^{n}(t - x_j)$$

elle s'annule en x_j pour j variant de 0 à n ainsi qu'en x suite au choix de la constante C, donc g s'annule au moins $n+2$ fois sur l'intervalle contenant les x_j et x, donc g' s'annule au moins $n+1$ fois sur ce même intervalle, donc g'' s'annule au moins n fois, etc. et finalement $g^{[n+1]}$ s'annule une fois au moins sur cet intervalle. Or

$$g^{[n+1]} = f^{[n+1]} - C(n+1)!$$

car P est de degré inférieur ou égal à n et $\prod_{j=0}^{n}(x-x_j)-x^{n+1}$ est de degré inférieur ou égal à n. Donc il existe bien un réel ξ_x dans l'intervalle contenant les x_j et x tel que

$$C=\frac{f^{[n+1]}(\xi_x)}{(n+1)!}$$

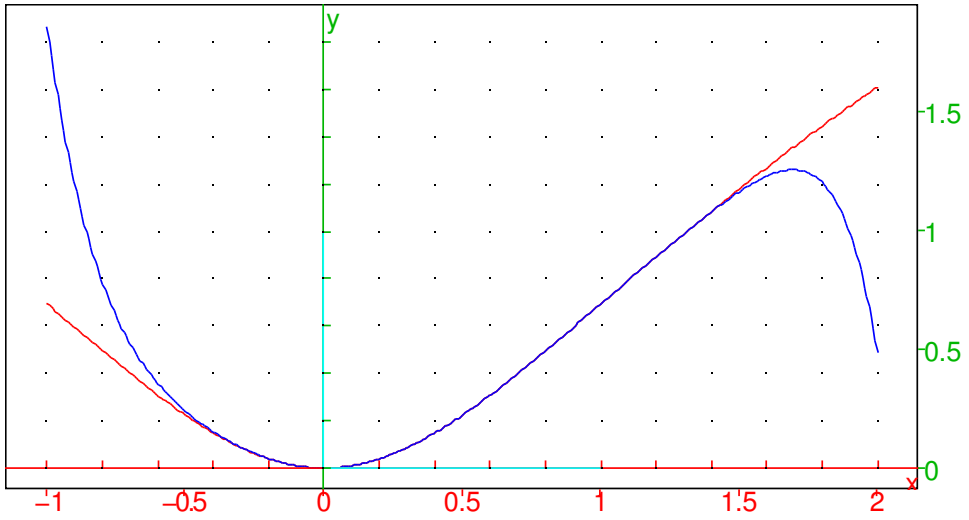
```
f(x):=ln(x^2+1); n:=10; X:=evalf(seq(j/n,j,0,n)); Y:=map(X,f);
```

$$(x)\text{->}\ln(x^2+1) \qquad ,10,[0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9$$

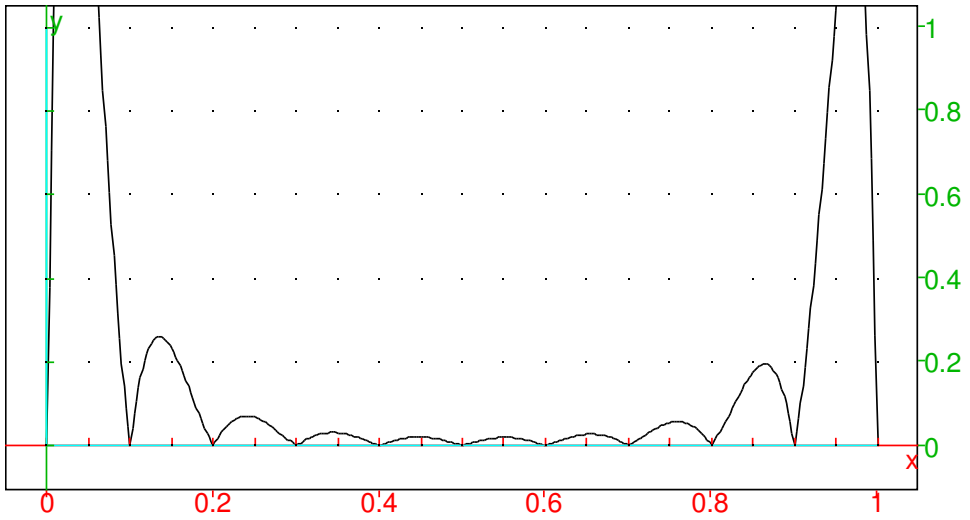
```
P:=interp(X,Y);
```

$$(((((((((-0.0178001249166\cdot(x-0.9)-0.0363274736916)\cdot(x-0.8)+0.115803003415)\cdot(x-0.7)-0.130300852032)\cdot(x-0.6)-0.0$$

```
plot([f(x),P],x=-1..2,color=[red,blue])
```




```
plot(1e7*abs(f(x)-P),x=0..1)
```



Attention, l'erreur d'interpolation peut devenir très grande lorsqu'on utilise beaucoup de points d'interpolation.

4.1.3 Calcul efficace du polynôme de Lagrange.

Avec la méthode de calcul précédent, on remarque que le polynôme de Lagrange peut s'écrire à la Horner sous la forme :

$$\begin{aligned} P(x) &= \alpha_0 + \alpha_1(x-x_0) + ... + \alpha_n(x-x_0)...(x-x_{n-1}) \\ &= \alpha_0 + (x-x_0)(\alpha_1 + (x-x_1)(\alpha_2 + ... + (x-x_{n-2})(\alpha_{n-1} + (x-x_{n-1})\alpha_n)...)) \end{aligned}$$

ce qui permet de le calculer rapidement une fois les α_i connus. On observe que

$$\alpha_0 = f(x_0), \quad \alpha_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

On va voir que les α_k peuvent aussi se mettre sous forme d'une différence. On définit les différences divisées d'ordre n par récurrence

$$f[x_i] = f(x_i), \quad f[x_i, ..., x_{k+i+1}] = \frac{f[x_{i+1}, ..., x_{k+i+1}] - f[x_i, ..., x_{k+i}]}{x_{k+i+1} - x_i}$$

On va montrer que $\alpha_k = f[x_0, ..., x_k]$. C'est vrai au rang 0, il suffit donc de le montrer au rang $k+1$ en l'admettant au rang k. Pour cela on observe qu'on peut construire le polynôme d'interpolation en $x_0, ..., x_{k+1}$ à partir des polynômes d'interpolation P_k en $x_0, ..., x_k$ et Q_k en $x_1, ..., x_{k+1}$ par la formule :

$$P_{k+1}(x) = \frac{(x_{k+1} - x)P_k + (x - x_0)Q_k}{x_{k+1} - x_0}$$

en effet on vérifie que $P_{k+1}(x_i) = f(x_i)$ pour $i \in [1, k]$ car $P_k(x_i) = f(x_i) = Q_k(x_i)$, et pour $i = 0$ et $i = k+1$, on a aussi $P_{k+1}(x_0) = f(x_0)$ et $P_{k+1}(x_{k+1}) = f(x_{k+1})$. Or α_{k+1} est le coefficient dominant de P_{k+1} donc c'est la différence du coefficient dominant de Q_k et de P_k divisée par $x_{k+1} - x_0$, c'est-à-dire la définition de

$f[x_0,...,x_{k+1}]$ en fonction de $f[x_1,...,x_{k+1}]$ et $f[x_0,...,x_k]$. Exemple : on reprend $P(0)=1, P(1)=2, P(2)=1$. On a

x_i	$f[x_i]$	$f[x_i,x_{i+1}]$	$f[x_0,x_1,x_2]$
0	$\boxed{1}$		
		$(2-1)/(1-0)=\boxed{1}$	
1	2		$(-1-1)/(2-0)=\boxed{-1}$
		$(1-2)/(2-1)=-1$	
2	1		

donc $P(x)=\boxed{1}+(x-0)(\boxed{1}+(x-1)(\boxed{-1}))=1+x(2-x)$. On peut naturellement utiliser l'ordre que l'on souhaite pour les x_i, en observant que le coefficient dominant de P ne dépend pas de cet ordre, on en déduit que $f[x_0,...,x_k]$ est indépendant de l'ordre des x_i, on peut donc à partir du tableau ci-dessus écrire P par exemple avec l'ordre 2,1,0, sous la forme

$$P(x)=1+(x-2)(-1+(x-1)(-1))=1+(x-2)(-x)$$

Le nombre d'opérations nécessaires pour faire ce calcul est proportionnel à n^2. La commande Xcas `interp` ou son synonyme `lagrange` effectue ce calcul. Pour avoir les différences divisées, on peut créer le programme suivant :

```
dd(X,Y):={ // Algorithme des différences divisées
  local k,l,n,A,old,cur;
  si size(X)!=size(Y) alors return "erreur" fsi;
  n:=size(X)-1;
  A:=[Y[0]];
  old:=Y;
  pour k de 1 jusque n faire
    // calcul de cur en fonction de old
    cur:=[];
    pour l de 0 jusque n-k faire
      cur[l]:=(old[l+1]-old[l])/(X[l+k]-X[l])
    fpour;
    A[k]:=cur[0];
    old:=cur;
  fpour;
  retourne A;
}:;
```

```
dd([0,1,2],[1,2,1])
```

$$[1,1,-1]$$

(N.B. pour rendre ce programme optimal, il faudrait utiliser l'affectation en place =< au lieu de :=)

4.1.4 Sensibilité aux erreurs sur les données.

Si les y_j sont connus avec une certaine erreur, alors le polynôme d'interpolation est connu de manière approchée. Plus précisément, si on note

$$\pi_j(x) = \prod_{k \neq j} \frac{x - x_k}{x_j - x_k}$$

le j-ième polynome de Lagrange valant 1 en x_j et 0 ailleurs, l'erreur vaut :

$$\sum_j (\tilde{y}_j - y_j)\pi_j(x)$$

Si l'erreur relative sur les y_j est majorée par ϵ, l'erreur sur le polynôme d'interpolation est majorée par :

$$\epsilon \max_j |y_j| \sum_j |\pi_j(x)|$$

il y a amplification de l'erreur par un facteur majoré par

$$\max_{x \in [a,b]} \sum_{j=0}^{n} |\pi_j(x)|$$

Ce facteur s'appelle **constante de Lebesgue** relative à la subdivision $x_0, ..., x_n$ de $[a, b]$. On peut le calculer numériquement pour une subdivision équidistribuée, et montrer qu'il croit comme $\frac{2^{n+1}}{en\ln(n)}$, par exemple pour $n = 40$, il vaut environ `5e9`. Illustration avec Xcas :

```
l(k,n):=product((x-j)/(k-j),j,0,k-1)*product((x-j)/(k-j),j,k+1,n);
```

$$Indexincorrect k-1,9 Erreur : TypeArgumentIncorrect$$

```
n:=10; f:=add(abs(l(k,n)),k,0,n); plot(f,x=0..n)
```

$$abs(l(0,n)) Erreur : Dimensioninvalide$$

puis essayer avec $n = 20$. Pour $n = 40$, en observant que le max est atteint dans $[0, 1]$, on peut remplacer les valeurs absolues par la bonne puissance de -1

```
n:=40; g:=l(0,n)+add((-1)^(k+n-1)*l(k,n),k,1,n)
```

$$sto(l(0,n) + add((-1)^(k+n-1) * l(k,n), k, 1, n), g) Erreur : Dimensioninvalide$$

on a alors un polynôme, dont on calcule l'abscisse du maximum par

```
g1:=normal(g'); l:=realroot(g1,1e-12,evalf)
```

$$0, []$$

puis `subst(g,x=l[0])`

$Indexhorslimite : 0, vectorsizeis0, syntaxcompatibilitymodexcasErreur : Dimensionincorrecte$

qui donne environ `4.7e9`.

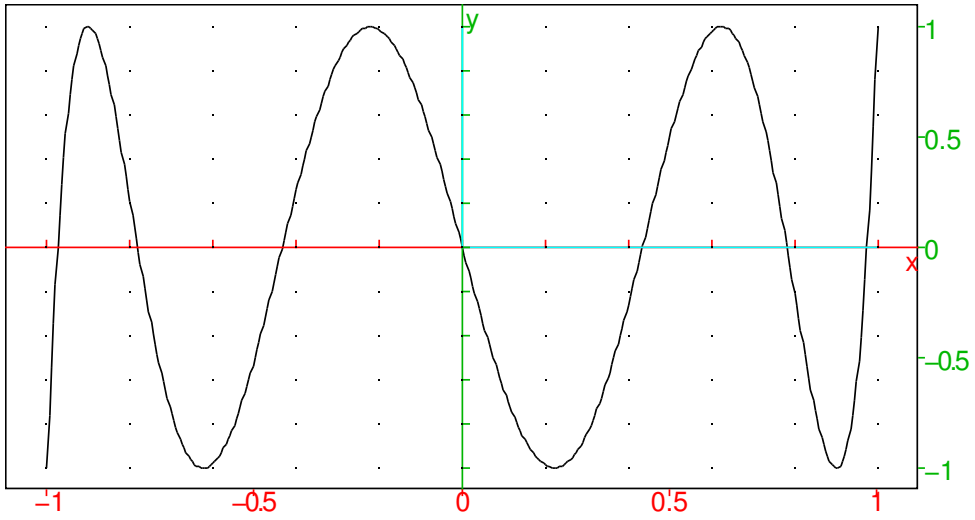
4.2 Interpolation aux points de Tchebyshev

L'idée la plus naturelle pour interpoler un polynôme en $n+1$ points d'un intervalle $[a,b]$ consiste à couper en n morceaux de même longueur. Mais ce n'est pas le plus efficace car le terme $|\prod_{j=0}^{n}(x-x_j)|$ est plus grand près des bords. Il est donc plus judicieux d'avoir plus de points près des bords et moins à l'intérieur. C'est là qu'interviennent les polynômes de Tchebyshev, ils sont définis par développement de $\cos(nx)$ en puissances de $\cos(x)$:

$$T_n(\cos(x)) = \cos(nx)$$

Sur $[-1,1]$, le polynôme T_n vaut en valeur absolue au plus 1, et atteint cette valeur exactement $n+1$ fois lorsque $x = k\pi/n$ donc $X = \cos(x) = \cos(k\pi/n)$.

```
n:=7; plot(tchebyshev1(n),x=-1..1)
```



De plus cette majoration est optimale. En effet soit U un polynôme de degré au plus n qui vérifie $|U|_\infty < 1$ sur $[-1,1]$ et tel que U ait le même coefficient dominant que T_n. Alors la différence $T_n - U$ est du signe de T_n en $X = \cos(k\pi/n), k \in [0,n]$ puisqu'en ces points T_n est extrêmal de valeur absolue 1. Donc $T_n - U$ s'annule n fois sur $[-1,1]$, mais son degré est au plus $n-1$ donc $T_n = U$ absurde. On a donc intérêt à prendre les abscisses des points d'interpolation en les racines t_n de T_n

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

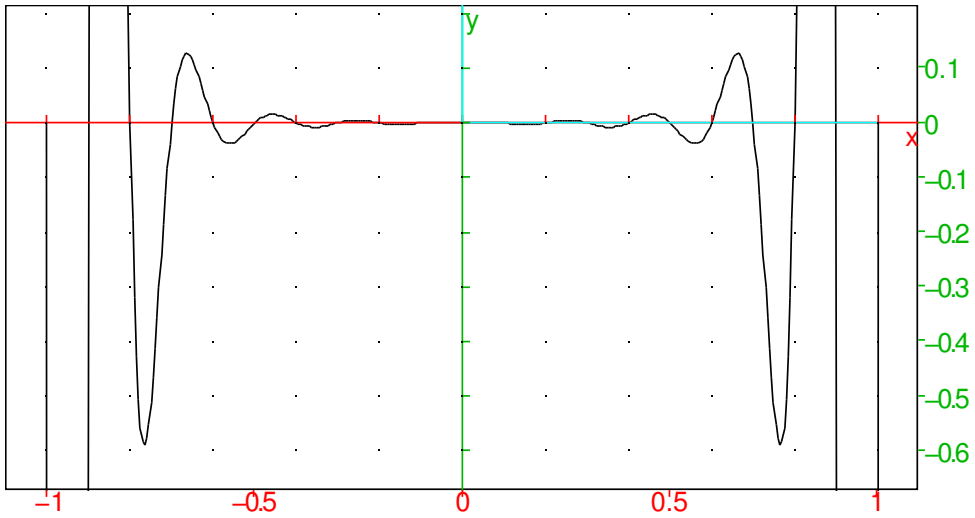
On pourra observer que le phénomène de Runge qui apparait par exemple pour $f(x) = 1/(25x^2 + 1)$ sur $[-1, 1]$ avec des points d'interpolation équidistants n'apparait plus si on prend des points de Tchebyshev.

```
f(x):=1/(1+25x^2); n:=20; X:=seq(2*j/n-1.0,j,0,n);
```

```
(x)->1/(1+25*x^2)
```

$$, 20, [-1.0, -0.9, -0.8, -0.7, -0.6, -0.5, -0.$$

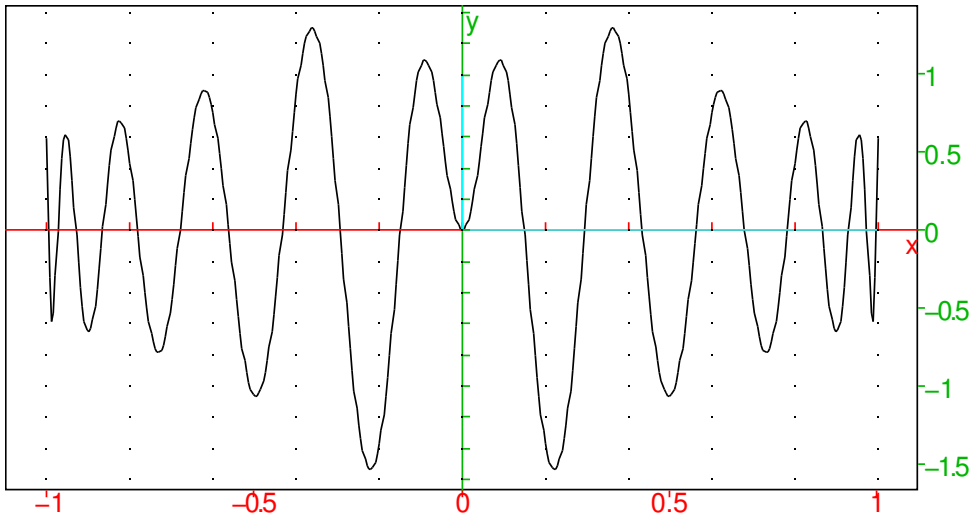
```
l:=lagrange(X,map(X,f)):; plot(l-f(x),x=-1..1)
```




```
T:=seq(cos((j+0.5)*pi/(n+1)),j,0,n);
```

$$[0.997203797181, 0.974927912182, 0.930873748644, 0.866025403784, 0.781831482468, 0.680172737771, 0.563320058064, 0.43$$

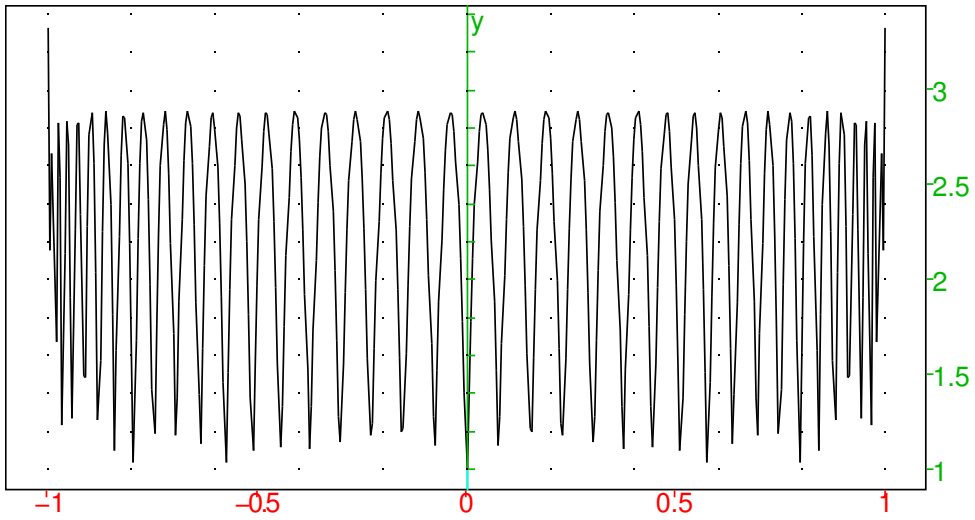
```
l:=lagrange(T,map(T,f)):; plot(100*(l-f(x)),x=-1..1)
```



Ceci est relié à la constante de Lebesgue qui pour des points de Tchebyshev vaut un peu moins de 4 pour $n < 100$ (se comporte comme $\frac{2}{\pi}\ln(n)$ pour n grand), on peut montrer que les polynômes de Lagrange aux points de Tchebyshev convergent uniformément vers $1/(25x^2+1)$ (c'est plus généralement vrai pour toute fonction C^1 sur l'intervalle). Remarque : ce n'est pas le polynôme de meilleure approximation, de f (celui qui minimise la norme L^∞ de la différence) car la dérivée $n+1$-ième varie en général sur $[a,b]$. Mais il est trop difficile de le calculer en général. Exemple de calcul explicite de constante de Lebesgue pour $n = 40$ avec Xcas

```
purge(x);
t(k,n):={
 local T;
 T:=seq(cos(pi*(k+.5)/(n+1)),k,0,n);
 return product((x-T[j])/(T[k]-T[j]),j,0,k-1)*
  product((x-T[j])/(T[k]-T[j]),j,k+1,n);
}:;

  n:=40; f:=add(abs(t(k,n)),k,0,n):;plot(f,x=-1..1)
```



4.3 Interpolation de Hermite

Si on fait tendre un des points d'interpolation vers un autre, la donnée de la valeur en ces 2 points serait redondante, elle est remplacée par la valeur de la dérivée. Dans le calcul des différences divisées ci-dessus on fera comme si les 2 points étaient distincts et successifs, disons x_i et x_{i+1}, on remplace le rapport indéterminé

$$\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = \frac{0}{0}$$

par $f'(x_i)$. On montre qu'une fois ce changement réalisé tout le reste est identique (y compris la majoration d'erreur). On peut bien sur généraliser au cas de plusieurs paires de points identiques ou des multiplicités plus grandes faisant intervenir des dérivées d'ordre supérieures, dans ce cas la différence divisée $f[x_i, ..., x_{i+m}]$ sera remplacée par $f^{[m]}(x_i)/m!$.

4.4 Polynômes de Bernstein et courbes de Bézier

Les polynômes de Bernstein de degré m sont les

$$B_k^n(x) = \begin{pmatrix} n \\ k \end{pmatrix} x^k (1-x)^{n-k}$$

On reconnait la probabilité d'avoir k succès si on effectue n tirages indépendants (avec remise) avec probabilité $x \in [0, 1]$ de succès par tirage. Ceci donne une relation de récurrence

$$B_{k+1}^{n+1} = (1-x)B_{k+1}^n + xB_k^n$$

qui peut servir à calculer les B_i^m. On en déduit aussi que l'espérance de k selon cette loi vaut nx (somme de n variables d'espérance x) et l'espérance de $(k - nx)^2$ vaut $nx(1-x)$ (variance de la somme de n variables indépendantes de variance x). On en déduit qu'on peut approcher uniformément une fonction continue sur un

intervalle $[a, b]$ par des polynômes, en se ramenant à $a = 0, b = 1$, on pose :

$$P_n(x) = \sum_{k=0}^{n} f(\frac{k}{n}) B_k^n(x)$$

En effet, par continuité uniforme de f sur $[0, 1]$, pour $\epsilon > 0$, il existe $\delta > 0$ tel que $|x - y| < \delta \Rightarrow |f(x) - f(y)| < \epsilon/2$, dans

$$P_n(x) - f(x) = \sum_{k=0}^{n} (f(\frac{k}{n}) - f(x)) B_k^n(x)$$

on décompose la somme sur k en deux parties, $|k/n - x| < \delta$ et $|k/n - x| \geq \delta$, pour la première somme, on majore $|f(\frac{k}{n}) - f(x)|$ par $\epsilon/2$ puis par $\sum_{k=0}^{n}$, pour la deuxième somme, on majore par $2|f|_\infty$ et on utilise $1 < (k/n - x)^2/\delta^2 = 1/n^2/\delta^2(k - nx)^2$ pour se ramener au calcul de la variance de k, au final

$$|P_n(x) - f(x)| \leq \frac{\epsilon}{2} + \frac{1}{n^2\delta^2} nx(1 - x)|f|_\infty$$

il suffit de choisir n assez grand pour rendre le membre de droite plus petit que ϵ. Les polynômes de Bernstein ne sont pas des polynômes interpolateurs aux points k/n, $0 < k < n$, et la convergence n'est pas forcément très rapide. On les utilise pour approcher rapidement des morceaux de courbes, si on se donne des "points de controle" $A_0, ..., A_n$ on construit la courbe paramétrée

$$A(t) = \sum_{k=0}^{n} A_k \left(\begin{array}{c} n \\ i \end{array} \right) x^i (1 - x)^{n-i}$$

appelée courbe de Bézier. En pratique on les utilise pour $n = 3$.

4.5 Polynômes orthogonaux.

Il s'agit d'une autre méthode d'approximation, particulièrement important pour l'intégration : les polynômes de meilleur approximation au sens de normes L^2 ou L^2 à poids $w(x) > 0$ sur l'intervalle de bornes α et β (finis ou infinis). On considère le produit scalaire

$$< f|g >= \int_\alpha^\beta w(x) f(x) g(x)\, dx$$

et on projette alors la fonction à approcher sur une base de polynômes orthogonaux de degrés croissants construit par la procédure de Gram-Schmidt à partir de la base canonique pour le produit scalaire ci-dessus.

Proposition 9 *Le polynôme P_n de degré n obtenu par orthogonalisation de Gram-Schmidt pour le produit scalaire $\int_\alpha^\beta w(x) f(x) g(x)\, dx$ possède n racines réelles.*

En effet, soit r le nombre de racines réelles de P_n, on pose $f = P_n$ et $g = \prod_{i=1}^{r} (x - x_i)$, wfg est de signe constant et non identiquement nul donc $< f|g >\neq 0$ donc $r = n$ sinon g serait de degré strictement plus petit que n donc orthogonal à P_n. On peut aussi construire ces polynômes en cherchant les valeurs propres de

$$T(f) = \frac{1}{w}(awf')'$$

où $a > 0$ est un polynôme de degré au plus 2 sur $]\alpha, \beta[$, tel que aw s'annule (ou tend vers 0) aux bornes de l'intervalle d'intégration : si α et β sont finis $a(x) = (x-\alpha)(\beta - x)$. On a alors

$$< T(f)|g >= \int_\alpha^\beta (awf')'g = [awf'g]_\alpha^\beta - \int_\alpha^\beta awg'f' = -\int_\alpha^\beta awg'f'$$

car le terme tout intégré s'annule (puisque aw s'annule en α et β). Donc $< T(f)|g >=< f|T(g) >$ est symétrique, les vecteurs propres de T correspondant à des valeurs propres distinctes sont donc orthogonaux entre eux. Pour trouver ces valeurs propres/polynômes vecteurs propres, on écrit $T(P_n) = \lambda_n P_n$ pour un polynôme P_n de degré n. Si aw'/w est un polynôme de degré au plus 1, le terme de degré dominant de cette équation donne la valeur de λ_n et les termes de degré plus petits permettent en général de déterminer de manière unique les coefficients de P_n en fonction du coefficient dominant. Pour certains poids $w(x)$ standards, les polynômes obtenus ont un nom :

- Legendre pour $w(x) = 1$ sur $[-1, 1]$, $a(x) = 1 - x^2$
- Hermite pour $w(x) = e^{-x^2}$ sur $\mathbb{R}$, $a(x) = 1$
- Laguerre pour $w(x) = x^a e^{-x}$ sur $\mathbb{R}^+$,
- Tchebyshev de première espèce pour $w(x) = 1/\sqrt{1-x^2}$ sur $[-1, 1]$, $a(x) = 1 - x^2$
- Tchebyshev de deuxième espèce pour $w(x) = \sqrt{1-x^2}$ sur $[-1, 1]$, $a(x) = 1 - x^2$

Ainsi, les polynômes de Legendre vérifient

$$((1-x^2)P_n')' = \lambda_n P_n$$

Le terme de degré n de cette équation donne $\lambda_n = -n(n+1)$, le terme sous-dominant est nul. Plus généralement, le terme de degré k vérifie

$$(k+2)(k+1)p_{k+2} - k(k+1)p_k = -n(n+1)p_k \Rightarrow p_{k+2} = \frac{k(k+1) - n(n+1)}{(k+2)(k+1)} p_k \tag{2}$$

Ceci permet de calculer le polynôme P_n, on normalise P_n par $P_n(1) = 1$.

```
n:=10; p:=makelist(n+1); p[0]:=1;
```

$$10, [0,0,0,0,0,0,0,0,0,0,0], [1,0,0,0,0,0,0,0,0,0,0]$$

```
for k from 0 to size(p)-3 step 2 do p[k+2]:=(k*(k+1)-n*(n+1))/(k+2)/(k+1)*p[k];
 od;
```

$$[1, 0, -55, 0, \frac{1430}{3}, 0, -1430, 0, \frac{12155}{7}, 0, \frac{-46189}{63}]$$

```
q:=p*1/sum(p); normal(poly2symb(revlist(q))); legendre(n)
```

$$[\frac{63}{-256}, 0, \frac{3465}{256}, 0, \frac{-15015}{128}, 0, \frac{45045}{128}, 0, \frac{-109395}{256}, 0, \frac{46189}{256}], \frac{46189}{256}\cdot x^{10} - \frac{109395}{256}\cdot x^8 + \frac{45045}{128}\cdot x^6 - \frac{15015}{128}\cdot x^4 + \frac{3465}{256}\cdot x^2 + \frac{-63}{256},$$

Le calcul de ces polynômes peut aussi se faire par une récurrence à 2 crans du type :

$$P_{n+1} = (a_n x - b_n)P_n - c_n P_{n-1} \tag{3}$$

La valeur de a_n est définie par la normalisation de la famille de polynômes, ensuite b_n et c_n sont déterminés respectivement par

$$< P_{n+1}|P_n >= 0 \Rightarrow b_n < P_n|P_n >= a_n < xP_n|P_n >$$

et

$$< P_{n+1}|P_{n-1} >= 0 \Rightarrow c_n < P_{n-1}|P_{n-1} >= a_n < xP_n|P_{n-1} >= a_n < P_n|xP_{n-1} >$$

donc

$$c_n < P_{n-1}|P_{n-1} >= \frac{a_n}{a_{n-1}} < P_n|P_n + b_{n-1}P_{n-1} + c_{n-1}P_{n-2} >= \frac{a_n}{a_{n-1}} < P_n|P_n >$$

Les autres relations d'orthogonalité $< P_{n+1}|P_j >= 0, j < n-1$ sont automatiquement vérifiées puisque $< xP_n|P_j >=< P_n|xP_j >= 0$ et P_n est orthogonal aux polynômes de degré $\leq n-1$. Dans l'exemple des polynômes de Legendre, les polynômes obtenus sont pairs si n est pair et impairs sinon, la relation de récurrence a donc un coefficient b_n nul. La convention de normalisation usuelle est $P_n(1) = 1$, on peut montrer qu'on a $a_n = (2n+1)/(n+1)$ et $< P_n|P_n >= 2/(2n+1)$ donc $c_n = n/(n+1)$.

$$(n+1)P_{n+1} = (2n+1)xP_n - nP_{n-1}$$

En effet la normalisation en 1 donne $1 = a_n - c_n$ donc $c_n = a_n - 1$, les termes de degré $n+1$ et $n-1$ de $P_{n+1} = a_n xP_n - c_n P_{n-1}$ donnent

$$p_{n+1,n+1} = a_n p_{n,n}, \quad p_{n+1,n-1} = a_n p_{n,n-2} - (a_n - 1)p_{n-1,n-1}$$

On applique alors (2) et on déduit de la deuxième équation ci-dessus :

$$-\frac{n(n+1)}{2(2n+1)}p_{n+1,n+1} = -\frac{n(n-1)}{2(2n+1)}a_n p_{n,n} - (a_n - 1)p_{n-1,n-1}$$

puis on applique la première :

$$\frac{n(n+1)}{2(2n+1)} - \frac{n(n-1)}{2(2n+1)} = \frac{a_n - 1}{a_n a_{n-1}}$$

d'où

$$a_n = \frac{1}{1 - \frac{n^2}{4n^2-1}a_{n-1}}$$

```
S(f,g):=int(f*g,x,-1,1);
```

```
(f,g)->int(f*g,x,-1,1)
```

```
gramschmidt([1,x,x^2,x^3,x^4],S);
```

$$[\text{inv}\left(\sqrt{2}\right), \frac{x}{\frac{\sqrt{6}}{3}}, \frac{(x^2 - \frac{1}{3})}{\frac{6\cdot\sqrt{10}}{45}}, \frac{(x^3 - \frac{3}{5}\cdot x)}{\frac{10\cdot\sqrt{14}}{175}}, \frac{(-\frac{1}{5} - \frac{6}{7}\cdot(x^2 - \frac{1}{3}) + x^4)}{\frac{840\cdot\sqrt{2}}{11025}}]$$

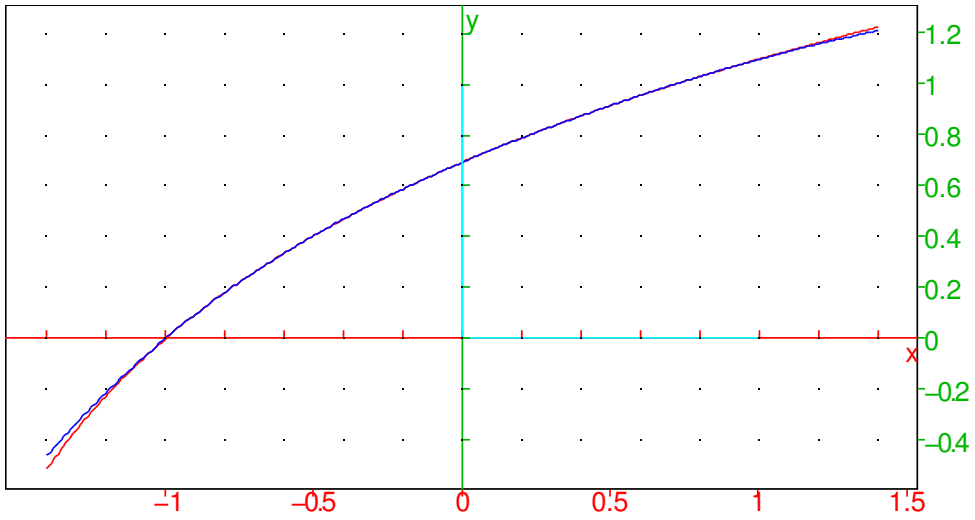
```
f:=ln(x+2);C:=seq(S(f,legendre(k))/S(legendre(k),legendre(k)),k,0,4);
```

$$\ln(x+2), [\frac{(3\ln(3)-2)}{2}, \frac{3(\frac{1}{4}\cdot(-6\ln(3)+15)+\frac{-7}{4})}{2}, \frac{5(\frac{1}{2}\cdot(6\ln(3)-15)+\frac{25}{6})}{2}, \frac{7(\frac{1}{32}\cdot(-228\ln(3)+615)+\frac{-1093}{96})}{2}, \underline{9(\frac{1}{80}}$$

```
g:=sum(C[j]*legendre(j),j,0,4);
```

$$\frac{(354375\cdot x^4\ln(3)-389340\cdot x^4-59850\cdot x^3\ln(3)+65800\cdot x^3-292950\cdot x^2\ln(3)+321720\cdot x^2+33750\cdot x\ln(3)-36600}{960}$$

```
plot([f,g],x,-1.4,1.4,color=[red,blue]);
```



Remarques

– On peut utiliser la récurrence à deux crans pour évaluer le polynôme en un réel, il n'est alors pas nécessaire de calculer P_n dans la base canonique.

– La récurrence à deux crans se généralise pour les degrés plus petits que d au produit scalaire

$$< P|Q >= \sum_{j=0}^{d} P(x_j)Q(x_j)$$

où les x_j sont des abscisses distinctes 2 à 2. Le calcul d'une base orthogonale permet de faire des calculs de régressions polynomiales mieux conditionnés qu'en utilisant la base canonique (et la matrice de Vandermonde).

– On peut voir l'interpolation trigonométrique (séries de Fourier, transformée de Fourier discrète) comme une généralisation en remplaçant une base orthonormalisée de polynômes par une base orthogonale de sinus, cosinus ou exponentielles.

4.6 Les splines

Il s'agit de fonctions définies par des polynomes de degré borné sur des intervalles, dont on fixe la valeur aux extrémités des intervalles (comme pour le polynome de Lagrange) ce qui rend la fonction continue, de plus on exige un degré de régularité plus grand, par exemple etre de classe C^2. Enfin, on fixe des conditions aux bornes de la réunion des intervalles, par exemple avoir certaines dérivées nulles. Par exemple supposons qu'on se donne n intervalles, donc $n+1$ points $x_0, ..., x_n$, on se fixe une régularité C^{d-1}. Ceci entraine $(n-1)d$ conditions de recollement, on y ajoute $n+1$ conditions de valeur en $x_0, ..., x_n$, on a donc $nd+1$ conditions, la borne sur le degré des polynomes doit donc etre d (ou plus, mais d suffit) ce qui donne $n(d+1)$ degrés de liberté, on peut donc ajouter $d-1$ conditions, par exemple pour les splines naturelles, on impose que les dérivées d'ordre $d/2$ à $d-1$ soient nulles en x_0 et x_n (si d est pair, on commence à la dérivée $d/2+1$-ième nulle en x_n). Pour trouver les polynomes, on doit donc résoudre un grand système linéaire. Une méthode permettant de diminuer la taille du système linéaire à résoudre dans le cas des splines naturelles consiste à se fixer n inconnues $z_0, .., z_{n-1}$ représentant les dérivées d-ième de la spline f en x_0 sur $[x_0, x_1]$ à x_{n-1} sur $[x_{n-1}, x_n]$, et $(d-1)/2$ inconnues f_j, représentant la valeur de la dérivée de f en x_0 pour j variant de 1 à $(d-1)/2$. On peut alors écrire le polynome sur l'intervalle $[x_0, x_1]$ car on connait son développement de Taylor en x_0. On effectue un changement d'origine (par application répétée de Horner) en x_1. On obtient alors le polynome sur $[x_1, x_2]$ en remplaçant uniquement la dérivée d-ième par z_1. On continue ainsi jusqu'en x_{n-1}. Le système s'obtient en calculant la valeur du polynome en $x_0, ..., x_n$ et la nullité des dérivées d'ordre $(d-1)/2$ à $d/2$ en x_n. On résoud le système et on remplace pour avoir les valeurs numériques des coefficients du polynome.

4.7 Autres approximations polynomiales.

On peut citer le polynôme de Taylor en un point qui donne une bonne approximation près du point (voir aussi ci-dessus le polynôme de Hermite), les approximants de Padé où on approche par le quotient de 2 polynômes (ceci donne parfois de très bons résultats comme pour la fonction exponentielle près de 0 par exemple).

```
purge(x); p:=pade(exp(x),x,10,6);
```

$$assume[[], \begin{pmatrix} -1 & 1 \end{pmatrix}, []], \frac{(-x^5 - 30 \cdot x^4 - 420 \cdot x^3 - 3360 \cdot x^2 - 15120 \cdot x - 30240)}{(x^5 - 30 \cdot x^4 + 420 \cdot x^3 - 3360 \cdot x^2 + 15120 \cdot x - 30240)}$$

On observe que le dénominateur est le numérateur pris en $-x$. Ceci permet de calculer précisément e^x par exemple sur $[-1, 1]$ en très peu d'opérations arithmétiques, calcul de $X = xx$, puis de $a = (30X + 3360)X + 30240, b = x((X + 420)X + 15120)$ puis $(a+b)/(a-b)$, soit 5 multiplications, 6 additions, 1 division.

```
x:=0.5; X:=x*x; a:=(30X+3360)*X+30240;  b:=x*((X+420)*X+15120);
 (a+b)/(a-b); exp(0.5);
```

$$0.5, 0.25, 31081.875, 7612.53125, 1.6487212707, 1.6487212707$$

On utilise aussi souvent des approximations distinctes selon l'intervalle considéré, par exemple l'interpolation linéaire par morceaux, les fonctions splines ... ou la réduction d'arguments pour se ramener à un intervalle avec une bonne apprixomation (par exemple $e^x = (e^{x/2})^2$)

5 Intégration numérique

Les fractions rationnelles admettent une primitive que l'on calcule en décomposant la fraction avec Bézout comme expliqué précédemment. Mais elles font figure d'exceptions, la plupart des fonctions n'admettent pas de

primitives qui s'expriment à l'aide des fonctions usuelles. Pour calculer une intégrale,on revient donc à la définition d'aire sous la courbe, aire que l'on approche, en utilisant par exemple un polynome de Lagrange. Le principe est donc le suivant : on découpe l'intervalle d'intégration en subdivisions $[a,b] = [a,a+h] + [a+h,a+2h] + ...[a+(n-1)h, a+nh = b$, où $h = (b-a)/n$ est le pas de la subdivision, et sur chaque subdivision, on approche l'aire sous la courbe.

5.1 Les rectangles et les trapèzes

Sur une subdivision $[\alpha,\beta]$, on approche la fonction par un segment. Pour les rectangles, il s'agit d'une horizontale : on peut prendre $f(\alpha)$, $f(\beta)$ (rectangle à droite et gauche) ou $f((\alpha+\beta)/2)$ (point milieu), pour les trapèzes on utilise le segment reliant $[\alpha, f(\alpha)]$ à $[\beta, f(\beta)]$. Exemple : calcul de la valeur approchée de $\int_0^1 t^3 dt$ (on en connait la valeur exacte $1/4 = 0.25$) par ces méthodes en subdivisant $[0,1]$ en 10 subdivisions (pas $h = 1/10$), donc $\alpha = j/10$ et $\beta = (j+1)/10$ pour j variant de 0 à 9. Pour les rectangles à gauche, on obtient sur une subdivision $f(\alpha) = (j/10)^3$ que l'on multiplie par la longueur de la subdivision soit $h = 1/10$:

$$\frac{1}{10}\sum_{j=0}^{9}(\frac{j}{10})^3 = \frac{81}{400} = 0.2025$$

Pour les rectangles à droite, on obtient

$$\frac{1}{10}\sum_{j=1}^{10}(\frac{j}{10})^3 = \frac{121}{400} = 0.3025$$

Pour le point milieu $f((\alpha+\beta)/2) = f((j/10 + (j+1)/10)/2) = f(j/10 + 1/20)$

$$\frac{1}{10}\sum_{j=0}^{9}(\frac{j}{10} + \frac{1}{20})^3 = 199/800 = 0.24875$$

Enfin pour les trapèzes, l'aire du trapèze délimité par l'axe des x, les verticales $y = \alpha$, $y = \beta$ et les points sur ces verticales d'ordonnées respectives $f(\alpha)$ et $f(\beta)$ vaut

$$h\frac{f(\alpha)+f(\beta)}{2}$$

donc

$$\frac{1}{10}\sum_{j=0}^{9}\left((\frac{j}{10})^3 + (\frac{j+1}{10})^3\right) = \frac{101}{400} = 0.2525$$

Dans la somme des trapèzes, on voit que chaque terme apparait deux fois sauf le premier et le dernier. Plus généralement, les formules sont donc les suivantes :

$$\text{rectangle gauche} = h\sum_{j=0}^{n-1} f(a+jh) \tag{4}$$

$$\text{rectangle droit} = h\sum_{j=1}^{n} f(a+jh) \tag{5}$$

$$\text{point milieu} = h\sum_{j=0}^{n-1} f(a+jh+\frac{h}{2}) \tag{6}$$

$$\text{trapezes} = h\left(\frac{f(a)+f(b)}{2}+\sum_{j=1}^{n-1} f(a+jh)\right) \tag{7}$$

où $h=(b-a)/n$ est le pas de la subdivision, n le nombre de subdivisions.

```
f(x):=ln(1+x^2); a:=0; b:=1.0; n:=100.0; h:=(b-a)/n;
 I:=int(f(x),x,a,b)
```

```
(x)->ln(1+x^2)
```

$$, 0, 1.0, 100.0, 0.01, 0.263943507355$$

```
R:=h*sum(f(a+j*h),j,0,n-1); h*sum(f(a+j*h),j,1,n); R-I;
```

$$0.260486104799, 0.267417576605, -0.00345740255559$$

```
M:=h*sum(f(a+h/2+j*h),j,0,n-1); M-I;
```

$$0.263939340676, -4.16667883663e-06$$

```
T:=h*(f(a)/2+f(b)/2+sum(f(a+j*h),j,1,n-1)); T-I;
```

$$0.263951840702, 8.33334720873e-06$$

On observe sur l'exemple que le point milieu et les trapèzes donnent une bien meilleure précision que les rectangles. Plus généralement, la précision de l'approximation n'est pas la même selon le choix de méthode. Ainsi pour les rectangles à gauche (le résultat est le même à droite), si f est continument dérivable, de dérivée majorée par une constante M_1 sur $[a,b]$, en faisant un développement de Taylor de f en α, on obtient

$$|\int_\alpha^\beta f(t)dt - \int_\alpha^\beta f(\alpha)dt| = |\int_\alpha^\beta f'(\theta_t)(t-\alpha)dt| \leq M_1\int_\alpha^\beta (t-\alpha)dt = M_1\frac{(\beta-\alpha)^2}{2}$$

Ainsi dans l'exemple, on a $M_1 = 3$, l'erreur est donc majorée par 0.015 sur une subdivision, donc par 0.15 sur les 10 subdivisions. Pour le point milieu, on fait le développement en $(\alpha+\beta)/2$ à l'ordre 2, en supposant que f est deux fois continument dérivable :

$$\begin{aligned} |\int_\alpha^\beta f(t) - \int_\alpha^\beta f(\frac{\alpha+\beta}{2})| &= |\int_\alpha^\beta f'(\frac{\alpha+\beta}{2})(t-\frac{\alpha+\beta}{2})dt \\ &\quad + \int_\alpha^\beta \frac{f''(\theta_t)}{2}(t-\frac{\alpha+\beta}{2})^2| \\ &\leq \frac{M_2}{2} 2 \int_{\frac{\alpha+\beta}{2}}^\beta (t-\frac{\alpha+\beta}{2})^2 dt \\ &\leq M_2 \frac{(\beta-\alpha)^3}{24} \end{aligned}$$

Dans l'exemple, on a $M_2 = 6$, donc l'erreur sur une subdivision est majorée par $0.25e-3$, donc sur 10 subdivisions par $0.25e-2 = 0.0025$. Pour les trapèzes, la fonction g dont le graphe est le segment reliant $[\alpha, f(\alpha)]$ à $[\beta, f(\beta)]$ est $f(\alpha) + (t-\alpha)/(\beta-\alpha)f(\beta)$, c'est en fait un polynome de Lagrange, si f est deux fois continument dérivable, on peut donc majorer la différence entre f et g en utilisant (1), on intègre la valeur absolue ce qui donne

$$|\int_\alpha^\beta f(t)dt - \int_\alpha^\beta g(t)dt| \leq \int_\alpha^\beta |\frac{f''(\xi_x)}{2}(x-\alpha)(x-\beta)| \leq M_2 \frac{(\beta-\alpha)^3}{12}$$

où M_2 est un majorant de $|f''|$ sur $[a,b]$. Lorsqu'on calcule l'intégrale sur $[a,b]$ par une de ces méthodes, on fait la somme sur $n = (b-a)/h$ subdivisions de longueur $\beta - \alpha = h$, on obtient donc une majoration de l'erreur commise sur l'intégrale :

- pour les rectangles à droite ou gauche $nM_1h^2/2 = M_1h(b-a)/2$
- pour le point milieu $M_2h^2(b-a)/24$
- pour les trapèzes $M_2h^2(b-a)/12$.

Lorsque h tend vers 0, l'erreur tend vers 0, mais pas à la même vitesse, plus rapidement pour les trapèzes et le point milieu que pour les rectangles. Plus on approche précisément la fonction sur une subdivision, plus la puissance de h va être grande, plus la convergence sera rapide lorsque h sera petit, avec toutefois une contrainte fixée par la valeur de M_k, borne sur la dérivée k-ième de f (plus k est grand, plus M_k est grand en général). Nous allons voir dans la suite comment se comporte cette puissance de h en fonction de la facon dont on approche f.

5.2 Ordre d'une méthode

On appelle méthode d'intégration l'écriture d'une approximation de l'intégrale sur une subdivision sous la forme

$$\int_\alpha^\beta f(t)dt \approx I(f) = \sum_{j=1}^k w_j f(x_j)$$

où les x_j sont dans l'intervalle $[\alpha, \beta]$, par exemple équirépartis sur $[\alpha, \beta]$. On utilise aussi la définition :

$$\int_\alpha^\beta f(t)dt \approx I(f) = (\beta-\alpha)\sum_{j=1}^k \tilde{w}_j f(x_j)$$

On prend toujours $\sum_j w_j = \beta - \alpha$ (ou $\sum_j \tilde{w}_j = 1$) pour que la méthode donne le résultat exact si la fonction est constante. On dit qu'une méthode d'intégration est d'ordre n si il y a égalité ci-dessus pour tous les polynômes de degré inférieur ou égal à n et non égalité pour un polynôme de degré $n+1$. Par exemple, les rectangles à droite et gauche sont d'ordre 0, le point milieu et les trapèzes sont d'ordre 1. Plus généralement, si on approche f par son polynôme d'interpolation de Lagrange en $n+1$ points (donc par un polynôme de degré inférieur ou égal à n), on obtient une méthode d'intégration d'ordre au moins n. Si une méthode est d'ordre n avec des $w_j \geq 0$ et si f est $n+1$ fois continument dérivable, alors sur une subdivision, on a :

$$|\int_\alpha^\beta f - I(f)| \leq M_{n+1} \frac{(\beta-\alpha)^{n+2}}{(n+1)!}(\frac{1}{n+2}+1) \tag{8}$$

En effet, on fait le développement de Taylor de f par exemple en α à l'ordre n

$$\begin{aligned} f(t) &= T_n(f) + \frac{(t-\alpha)^{n+1}}{(n+1)!} f^{[n+1]}(\theta_t), \\ T_n(f) &= f(\alpha) + (t-\alpha)f'(\alpha) + ... + \frac{(t-\alpha)^n}{n!} f^{[n]}(\alpha) \end{aligned}$$

Donc

$$|\int_\alpha^\beta f - \int_\alpha^\beta T_n(f)| \leq \int_\alpha^\beta \frac{(t-\alpha)^{n+1}}{(n+1)!} |f^{[n+1]}(\theta_t)| \leq \left[M_{n+1} \frac{(t-\alpha)^{n+2}}{(n+2)!} \right]_\alpha^\beta$$

De plus,

$$\begin{aligned} |I(f) - I(T_n(f))| = |I\left(f^{[n+1]}(\theta_t) \frac{(t-\alpha)^{n+1}}{(n+1)!} \right)| &\leq \sum_{j=1}^k |w_j| M_{n+1} \frac{(x_j-\alpha)^{n+1}}{(n+1)!} \\ &\leq \sum_{j=1}^k |w_j| M_{n+1} \frac{(\beta-\alpha)^{n+1}}{(n+1)!} \end{aligned}$$

Donc comme la méthode est exacte pour $T_n(f)$, on en déduit que

$$\begin{aligned} |\int_\alpha^\beta f - I(f)| &= |\int_\alpha^\beta f - \int_\alpha^\beta T_n(f) + I(T_n(f)) - I(f)| \\ &\leq |\int_\alpha^\beta f - \int_\alpha^\beta T_n(f)| + |I(T_n(f)) - I(f)| \\ &\leq M_{n+1} \frac{(\beta-\alpha)^{n+2}}{(n+2)!} + \sum_{j=1}^k |w_j| M_{n+1} \frac{(\beta-\alpha)^{n+1}}{(n+1)!} \end{aligned}$$

Si les $w_j \geq 0$, alors $\sum_{j=1}^k |w_j| = \sum_{j=1}^k w_j = \beta - \alpha$ et on obtient finalement (8) On remarque qu'on peut améliorer la valeur de la constante en faisant tous les développement de Taylor en $(\alpha+\beta)/2$ au lieu de α, Après sommation sur les n subdivisions, on obtient que :

Théorème 10 *Pour une méthode d'ordre n à coefficients positifs et une fonction f $n+1$ fois continument dérivable*

$$|\int_a^b f - I(f)| \leq M_{n+1} \frac{h^{n+1}}{2^{n+1}(n+1)!}(b-a)(\frac{1}{(n+2)}+1)$$

On observe que cette majoration a la bonne puissance de h sur les exemples déja traités, mais pas forcément le meilleur coefficient possible, parce que nous avons traité le cas général d'une méthode d'ordre n, et utilisé une majoration pas toujours optimale du reste. Pour obtenir la meilleure valeur possible de la constante, il faut exprimer le reste de la formule de Taylor sous forme intégrale et utiliser la forme précise de la méthode :

$$f(t) = T_n(f)(t) + \int_\alpha^t \frac{(t-x)^n}{n!} f^{[n+1]}(x)\, dx$$

donc

$$\int_\alpha^\beta f - \int_\alpha^\beta T_n(f) = \int_\alpha^\beta \int_\alpha^t \frac{(t-x)^n}{n!} f^{[n+1]}(x)\, dx\, dt = \int_\alpha^\beta \int_\alpha^\beta \frac{(t-x)_+^n}{n!} f^{[n+1]}(x)\, dx\, dt$$

où $(t-x)_+ = \max(0, t-x)$. En intervertissant les deux intégrales (Fubini), on obtient :

$$\begin{aligned}
\int_\alpha^\beta f - \int_\alpha^\beta T_n(f) &= \int_\alpha^\beta \int_\alpha^\beta \frac{(t-x)_+^n}{n!}\, dt f^{[n+1]}(x)\, dx \\
&= \int_\alpha^\beta \int_x^\beta \frac{(t-x)_+^n}{n!}\, dt f^{[n+1]}(x)\, dx \\
&= \int_\alpha^\beta \frac{(\beta-x)^{n+1}}{(n+1)!} f^{[n+1]}(x)\, dx
\end{aligned}$$

D'autre part :

$$\begin{aligned}
I(f) - I(T_n(f)) &= I\left(\int_\alpha^\beta \frac{(x-t)_+^n}{n!} f^{[n+1]}(t)\, dt\right) \\
&= \sum_{j=1}^k w_j \int_\alpha^\beta \frac{(x_j-t)_+^n}{n!} f^{[n+1]}(t)\, dt
\end{aligned}$$

Donc :

$$\int_\alpha^\beta f - I(f) = \int_\alpha^\beta \left(\frac{(\beta-x)^{n+1}}{(n+1)!} - \sum_{j=1}^k w_j \frac{(x_j-x)_+^n}{n!}\right) f^{[n+1]}(x)\, dx$$

On appelle noyau de Péano l'expression

$$N(x) = \frac{(\beta-x)^{n+1}}{(n+1)!} - \sum_{j=1}^k w_j \frac{(x_j-x)_+^n}{n!}$$

On a alors :

$$\left|\int_\alpha^\beta f - I(f)\right| \le \int_\alpha^\beta |N(x)||f^{[n+1]}(x)|\, dx \le M_{n+1} \int_\alpha^\beta |N(x)|\, dx$$

5.3 Simpson

Il s'agit de la méthode obtenue en approchant la fonction sur la subdivision $[\alpha, \beta]$ par son polynome de Lagrange aux points $\alpha, (\alpha+\beta)/2, \beta$. On calcule l'intégrale par exemple avec un logiciel de calcul formel, avec Xcas :

```
factor(int(lagrange([a,(a+b)/2,b],[fa,fm,fb]),x=a..b))
```

$$\frac{(\mathrm{fa}+\mathrm{fb}+4\cdot\mathrm{fm})}{6}$$

qui donne la formule sur une subdivision

$$I(f)=\frac{h}{6}(f(\alpha)+4f(\frac{\alpha+\beta}{2})+f(\beta))$$

et sur $[a,b]$:

$$I(f)=\frac{h}{6}\left(f(a)+f(b)+4\sum_{j=0}^{n-1}f(a+jh+\frac{h}{2})+2\sum_{j=1}^{n-1}f(a+jh)\right) \tag{9}$$

```
f(x):=ln(1+x^2); a:=0; b:=1.0; n:=100.0; h:=(b-a)/n;
 int(f(x),x,a,b)
```

```
(x)->ln(1+x^2)
```

$$,0,1.0,100.0,0.01,0.263943507355$$

```
h/6*(f(a)+f(b)+4*sum(f(a+j*h+h/2),j,0,n-1)+2*sum(f(a+j*h),j,1,n-1))
```

$$0.263943507351$$

Si on intègre t^3 sur $[0,1]$ en 1 subdivision par cette méthode, on obtient

$$\frac{1}{6}(0+4\frac{1}{2^3}+1)=\frac{1}{4}$$

c'est-à-dire le résultat exact, ceci est aussi vérifié pour f polynome de degré inférieur ou égal à 2 puisque l'approximation de Lagrange de f est alors égale à f. On en déduit que la méthode de Simpson est d'ordre 3 (pas plus car la méthode de Simpson appliquée à l'intégrale de t^4 sur $[0,1]$ n'est pas exacte). On peut améliorer la constante générale de la section précédente pour la majoration de l'erreur en :

$$|\int_a^b f-I(f)|\leq\frac{h^4}{2880}(b-a)M_4$$

En effet sur une subdivision élémentaire $[\alpha,\beta]$, le noyau de Péano vaut :

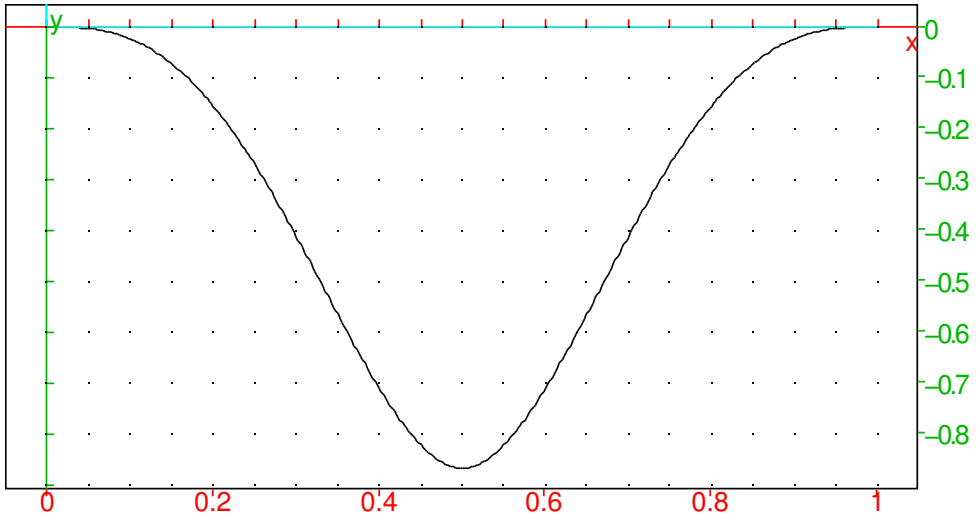
$$\begin{aligned}N(x) &= \frac{(\beta-x)^4}{4!}-\frac{1}{6}\frac{(\beta-x)_+^3}{3!}-\frac{2}{3}\frac{(\frac{\alpha+\beta}{2}-x)_+^3}{3!}-\frac{1}{6}\frac{(\alpha-x)_+^3}{3!}\\ &= \frac{(\beta-x)^4}{4!}-\frac{1}{6}\frac{(\beta-x)^3}{3!}-\frac{2}{3}\frac{(\frac{\alpha+\beta}{2}-x)_+^3}{3!}\end{aligned}$$

on observe que $N(x)\leq 0$ sur $[\alpha,\beta]$

```
N(x):=(1-x)^4/4!-1/6*(1-x)^3/3!-2/3/3!*max(0,1/2-x)^3
```

```
(x)->(1-x)^4/(4!)-1/6*(1-x)^3/(3!)-2/3/(3!)*max(0,1/2-x)^3
```

```
plot(1000*N(x),x=0..1)
```



et son intégrale vaut $-1/2880(\beta-\alpha)^4$:

```
int(N(x),x=0..1/2)+int(N(x),x=1/2..1)
```

$$\frac{-1}{2880}$$

La méthode de Simpson nécessite $2n+1$ évaluations de f (le calcul de f est un point étant presque toujours l'opération la plus couteuse en temps d'une méthode de quadrature), au lieu de n pour les rectangles et le point milieu et $n+1$ pour les trapèzes. Mais on a une majoration en h^4 au lieu de h^2 donc le "rapport qualité-prix" de la méthode de Simpson est meilleur, on l'utilise donc plutot que les méthodes précédentes sauf si f n'a pas la régularité suffisante (ou si M_4 est trop grand).

5.4 Newton-Cotes

On peut généraliser l'idée précédente, découper la subdivision $[\alpha,\beta]$ en n parts égales et utiliser le polynôme d'interpolation en ces $n+1$ points $x_0=\alpha, x_1, ..., x_n=\beta$. Ce sont les méthodes de Newton-Cotes, qui sont d'ordre n au moins. Comme le polynôme d'interpolation dépend linéairement des ordonnées, cette méthode est bien de la forme :

$$I(f)=(\beta-\alpha)\sum_{j=0}^{n}\tilde{w}_j f(x_j)$$

De plus les $\tilde{w}_j$ sont universels (ils ne dépendent pas de la subdivision), parce qu'on peut faire le changement de variables $x = \alpha + t(\beta - \alpha)$ dans l'intégrale et le polynôme d'interpolation et donc se ramener à $[0, 1]$. Exemple : on prend le polynôme d'interpolation en 5 points équidistribués sur une subdivision $[a, b]$ (méthode de Boole). Pour calculer les $\tilde{w}_j$, on se ramène à $[0, 1]$, puis on tape

```
int(lagrange(seq(j/4,j,0,4),[f0,f1,f2,f3,f4]),x=0..1)
```

$$\frac{(7 \cdot \mathrm{f0} + 32 \cdot \mathrm{f1} + 12 \cdot \mathrm{f2} + 32 \cdot \mathrm{f3} + 7 \cdot \mathrm{f4})}{90}$$

et on lit les coefficients de `f0` à `f4` qui sont les $\tilde{w}_0$ à $\tilde{w}_4$: 7/90, 32/90, 12/90, 32/90, 7/90. Voir aussi la section 5.5 La méthode est d'ordre au moins 4 par construction, mais on vérifie qu'elle est en fait d'ordre 5 (exercice), la majoration de l'erreur d'une méthode d'ordre 5 est

$$|\int_a^b f - I(f)| \leq \frac{M_6}{2^6 6!}(1 + \frac{1}{7})h^6(b - a)$$

elle peut être améliorée pour cette méthode précise en

$$|\int_a^b f - I(f)| \leq \frac{M_6}{1935360}h^6(b - a)$$

En pratique, on ne les utilise pas très souvent, car d'une part pour $n \geq 8$, les w_j ne sont pas tous positifs, et d'autre part, parce que la constante M_n devient trop grande. On préfère utiliser la méthode de Simpson en utilisant un pas plus petit. Il existe aussi d'autres méthodes, par exemple les quadratures de Gauss (on choisit d'interpoler en utilisant des points non équirépartis tels que l'ordre de la méthode soit le plus grand possible, cf. infra) ou la méthode de Romberg qui est une méthode d'accélération de convergence basée sur la méthode des trapèzes (on prend la méthode des trapèzes en 1 subdivision de $[a, b]$, puis 2, puis 2^2, ..., et on élimine les puissances de h du reste $\int f - I(f)$ en utilisant un théorème d'Euler-Mac Laurin qui montre que le développement asymptotique de l'erreur en fonction de h ne contient que des puissances paires de h). De plus, on peut être amené à faire varier le pas h en fonction de la plus ou moins grande régularité de la fonction.

5.5 Calcul des poids w_i

Si la méthode d'intégration consiste à interpoler f en n points $x_0, ..., x_n$, alors la méthode est exacte pour tout polynôme de degré n. Si on prend $P_j(x) = \prod_{k \neq j}(x - x_k)$, on en déduit :

$$\int_\alpha^\beta P_j(x)\, dx = w_j P_j(x_j)$$

Par exemple en interpolant en $0, 1/2, 1$ sur $[0, 1]$, on obtient

$$w_0 P_0(0) = \int_0^1 P_0(x)\, dx, \quad P_0(x) = (x - \frac{1}{2})(x - 1)$$

```
purge(x);P0:=(x-1)*(x-1/2); int(P0,x,0,1)/P0(x=0);
```

$$0.5, (x - 1) \cdot (x - \frac{1}{2}), \frac{1}{6}$$

On peut aussi résoudre un système linéaire en prenant pour f les polynômes de la base canonique, la matrice du système est la transposée de la matrice de Vandermonde en les x_j et le second membre a pour j-ième composante $\int_\alpha^\beta x^j\,dx$.

```
inv(tran(vandermonde(0,1/2,1)))*[1,1/2,1/3]
```

$$[\frac{1}{6},\frac{2}{3},\frac{1}{6}]$$

5.6 En résumé

Intégration sur $[a,b]$, h pas d'une subdivision, M_k majorant de la dérivée k-ième de la fonction sur $[a,b]$

	formule	Lagrange degré	ordre	erreur
rectangles	(4), (5)	0	0	$M_1h(b-a)/2$
point milieu	(6)	0	1	$M_2h^2(b-a)/24$
trapèzes	(7)	1	1	$M_2h^2(b-a)/12$
Simpson	(9)	2	3	$M_4h^4(b-a)/2880$

5.7 Accélération de Richardson-Romberg

Proposition 11 *Soit g une fonction de classe C^{2k} sur $[a,b]$, $T_h(g)$ la valeur de la méthode des trapèzes sur $[a,b]$ de pas $h=(b-a)/N$ (N entier). Alors $T_h(g)$ admet un développement en puissances paires de h à l'ordre $2k$.*

Pour montrer ce résultat, il faut établir la formule d'Euler-Mac Laurin. On commence par se placer sur une subdivision de l'intervalle $[0,1]$, on intègre par parties $\int_0^1 f(t)\,dt$ en faisant apparaitre la formule des trapèzes, on intègre donc 1 en $t-\frac{1}{2}$

$$\int_0^1 f(t)\,dt=[(t-\frac{1}{2})f(t)]_0^1-\int_0^1(t-\frac{1}{2})f'(t)\,dt=\frac{f(0)+f(1)}{2}-\int_0^1(t-\frac{1}{2})f'(t)\,dt$$

où encore

$$T_1(f)=\int_0^1 f(t)\,dt+\int_0^1(t-\frac{1}{2})f'(t)\,dt$$

Pour poursuivre, on pose $B_1(t)=t-\frac{1}{2}$, qu'on va intégrer en $\frac{1}{2}((t-\frac{1}{2})^2+c)$, où on choisit c pour que l'intégrale soit nulle, donc $c=-1/6$. On pose $B_2=(t-\frac{1}{2})^2-1/6$, on a :

$$T_1(f)=\int_0^1 f(t)\,dt+[\frac{B_2}{2}f']_0^1-\int_0^1\frac{B_2}{2}f''(t)\,dt$$

Plus généralement, on pose

$$B'_{p+1}=pB_p,\quad \int_0^1 B_{p+1}(t)\,dt=0$$

ce qui définit de manière unique les B_p. La nullité de l'intégrale montre que $B_{p+1}(1)=B_{p+1}(0)$ ce qui simplifiera l'expression des termes tout intégrés. De plus, on montre par récurrence que les B_p ont une symétrie paire ou

impaire selon la parité de p par rapport à $t = 1/2$. Après p intégrations par parties, on obtient :

$$\begin{aligned} T_1(f) &= \int_0^1 f(t)\,dt + \frac{B_2(0)}{2}(f'(1) - f'(0)) + ... \\ &\quad + \frac{B_{2k}(0)}{(2k)!}(f^{[2k-1]}(1) - f^{[2k-1]}(0)) - \int_0^1 \frac{B_{2k}}{(2k)!} f^{[2k]}(t)\,dt \end{aligned}$$

En faisant le même raisonnement sur $[k, k+1]$ pour $k = 1, ..., N-1$ et en sommant, on obtient la formule d'**Euler-Mac Laurin** :

$$\begin{aligned} T_1^{[0,N]}(f) &= \int_0^N f(t)\,dt + \frac{B_2(0)}{2}(f'(N) - f'(0)) + ... \\ &\quad + \frac{B_{2k}(0)}{(2k)!}(f^{[2k-1]}(N) - f^{[2k-1]}(0)) - \int_0^N \frac{B_{2k}}{(2k)!} f^{[2k]}(t)\,dt \end{aligned}$$

On pose alors $x = a + ht$ (donc $dt = dx/h$) et $f(t) = g(x)$ (donc $f'(t) = df/dt = hdg/dx$), on obtient

$$\begin{aligned} \frac{1}{h} T_h^{[a,b]}(g) &= \frac{1}{h}\int_0^N g(x)\,dx + \frac{B_2(0)}{2} h(g'(b) - g'(a)) + ... \\ &\quad + \frac{B_{2k}(0)}{(2k)!} h^{2k-1}(g^{[2k-1]}(N) - g^{[2k-1]}(0)) - \int_0^N \frac{B_{2k}}{(2k)!} h^{2k} g^{[2k]}(x)\,\frac{1}{h}dx \end{aligned}$$

donc

$$\begin{aligned} T_h^{[a,b]}(g) &= \int_0^N g(x)\,dx + h^2\frac{B_2(0)}{2}(g'(b) - g'(a)) + ... \\ &\quad + h^{2k}\frac{B_{2k}(0)}{(2k)!}(g^{[2k-1]}(N) - g^{[2k-1]}(0)) - h^{2k}\int_0^N \frac{B_{2k}}{(2k)!} g^{[2k]}(x)\,dx \end{aligned}$$

L'accélération consiste à éliminer les puissances de h^2 en commençant par h^2 avec des subdivisions deux fois plus fines à chaque itération. Ainsi $T_h^1(f) = (4T_{h/2}(f) - T_h(f))/(4-1)$ n'a plus de termes en h^2 et tend vers l'intégrale à approcher lorsque h tend vers 0. On peut d'ailleurs vérifier qu'il s'agit de la méthode de Simpson. On élimine ensuite le terme en h^4 en posant $T_h^2(f) = (4^2 T_{h/2}^1(f) - T_h^1(f))/(4^2 - 1)$ et ainsi de suite. On construit un tableau triangulaire `T` dont chaque ligne l contient $T_{h/2^l}(f), T_{h/2^l}^1(f), ...$ (avec des indices qui commencent à 0). Pour calculer le terme d'indice 0 de la ligne courante on fait une méthode des trapèzes sur 2 fois plus de subdivisions que la précédente, puis pour le j-ième terme `T[l,j]` on effectue `(4^j*T[l-1,j-1]-T[l,j-1])/(4^j-1)` (on n'a donc besoin que de la ligne précédente pour calculer la ligne courante). On s'arrête par exemple lorsque la valeur absolue de la différence entre les derniers termes de deux lignes consécutives est inférieur à la précision souhaitée (erreur empirique).

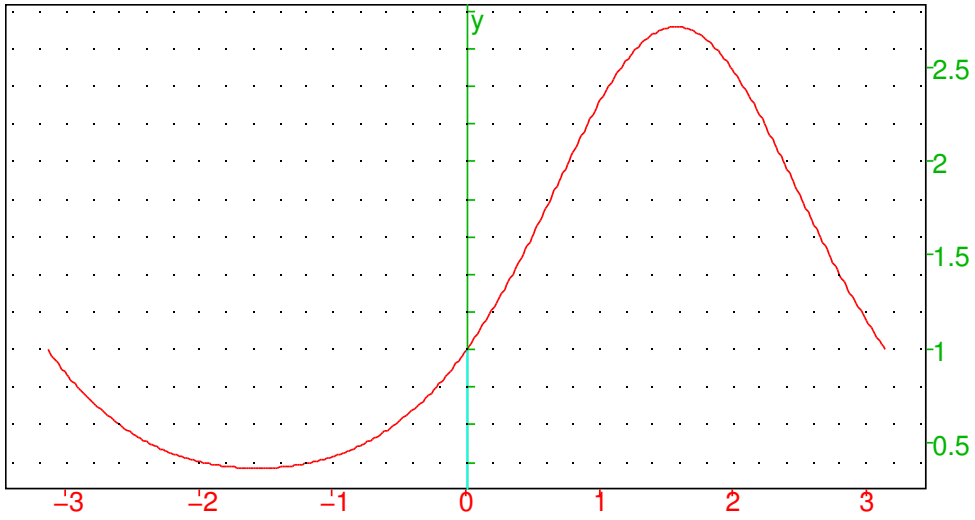
5.8 Cas des fonctions périodiques.

Si f est une fonction périodique régulière (C^∞), alors la méthode des trapèzes sur une période est d'ordre arbitrairement grand. En effet, pour une série de Fourier tronquée à l'ordre m, la formule des trapèzes avec N subdivisions donne le résultat exact de $\int_0^T f(t)dt$ dès que $N > m$. Il suffit ensuite d'utiliser que le reste de la série de Fourier ($m > N$) a des coefficients à décroissance rapide. La méthode des trapèzes donne donc de bons

résultats pour une fonction périodique, on peut d'ailleurs aussi l'appliquer pour calculer une valeur approchée des coefficients de Fourier de la fonction. La liste des valeurs approchées obtenue est alors la transformée de Fourier discrète des valeurs de la fonction f aux N points de la subdivision, elle se calcule donc rapidement avec la transformée de Fourier rapide. Par exemple, pour approcher $f(x) = e^{\sin(x)}$, on peut utiliser les commandes suivantes en Xcas :

```
f(x):=exp(sin(x));
N:=16; F:=seq(f(k/N*2.*pi),k,0,N-1); G:=fft(F);
k:=4;
g:=G[0]+sum(G[j]*exp(i*j*x),j,1,k)+sum(G[N-j]*exp(-i*j*x),j,1,k);
h:=normal(re(exp2trig(g)/N));

  plot(h,x=-pi..pi,color=red); //plot(f(x),x=-pi..pi)
```



Ou directement `2*re(G[j]/N)` est une valeur approchée du j-ième coefficient de Fourier a_j de f, et `-2*im(G[j]/N)` de b_j, par exemple :

```
1/pi*int(f(x)*cos(4x),x,0,2.*pi); 2*re(G[4]/N);
```

$$0.00547424044207, 0.00547424044313$$

```
1/pi*int(f(x)*sin(5x),x,0,2.*pi); -2*im(G[5]/N);
```

$$0.000542926311 91, 0.000542926336897$$

On observe en effet une très bonne concordance (11 décimales). Bien entendu, cela n'est pas très utile pour approcher $e^{\sin(x)}$ (il vaut mieux composer exponentielle et sinus), mais cela pourrait le devenir pour une fonction périodique plus compliquée ou pour une fonction périodique dont on ne connait qu'un échantillonage régulier (par exemple un fichier numérique audio).

5.9 Quadratures gaussiennes.

5.9.1 Description

On a vu que l'interpolation polynomiale était de meilleure qualité en prenant les points de Tchebyshev plutot que des points équidistants, il est donc naturel de calculer des approximations d'intégrale de cette manière ou encore d'optimiser le choix des abscisses pour avoir une méthode d'intégration d'ordre maximal. Si on se fixe n abscisses x_1 à x_n, on peut obtenir l'ordre $2n-1$. En effet, considérons le polynôme $P_n = \prod_{i=1}^{n}(x-x_i)$, qui est de degré n, si la méthode est d'ordre $2n-1$ alors il sera orthogonal à tous les polynômes de degré inférieur strict à n pour le produit scalaire

$$< f|g >= \int_a^b f(x)g(x)\,dx$$

puisque $< P_n|x^j >= \int_a^b P_n x^j$ sera combinaison linéaire des $P_n x^j$ en $x_k, k = 1..n$ (car la formule d'intégration est exacte puisque le degré du polynôme $P_n x^j$ est au plus $2n-1$). Donc P_n est à une constante multiplicative près le n-ième polynôme orthogonal pour l'integrale sur $[a,b]$, si $[a,b] = [-1,1]$ c'est `legendre(n)`. Réciproquement, si les x_k sont les racines de ce polynôme, alors la formule d'intégration est exacte, on effectue la division euclidienne du polynôme P de degré au plus $2n-1$ à intégrer par P_n

$$P = P_n Q + R, \quad \deg(Q) \leq n-1$$

On a $\int_a^b P_n Q = 0$ par orthogonalité et la combinaison linéaire correspondante en les x_k est nulle, et on a exactiture pour R, car de degré au plus $n-1$. Exemple :

```
l:=proot(legendre(10));
```

$[-0.973906528517, -0.865063366689, -0.679409568299, -0.433395394129, -0.148874338982, 0.148874338982, 0.433395394$

```
f(x):=ln(1+x^2); int(f(x),x,-1.0,1.0);
```

```
(x)->ln(1+x^2)
```
$, 0.527887014709$

```
p:=interp(l,f); int(p,x,-1.0,1.0)
```

$(((((((-0.0581540739601 \cdot (x-0.679409568299)+0.106943591296)\cdot(x-0.433395394129)-0.0359778773977)\cdot(x-0.148874338$

5.9.2 Calcul des poids

On peut calculer les poids en appliquant la section 5.5. On peut ainsi montrer que les poids sont positifs en appliquant la formule d'intégration au polynôme $\prod_{j\neq k}(x-x_j)$ (la formule est exacte à cause du degré du polynome). On peut d'ailleurs montrer que le poids w_i en une racine du polynôme de Legendre sur $[-1,1]$ est donné par :

$$w_i = \frac{2}{(1-x_i^2)P_n'(x_i)^2} = \frac{2(1-x_i^2)}{n^2 P_{n-1}(x_i)^2} = \frac{2}{nP_{n-1}(x_i)P_n'(x_i)}$$

```
n:=10; P:=legendre(n); Q:=legendre(n-1); l:=proot(P);
```

$$10, \frac{46189}{256}\cdot x^{10}+\frac{-109395}{256}\cdot x^{8}+\frac{45045}{128}\cdot x^{6}+\frac{-15015}{128}\cdot x^{4}+\frac{3465}{256}\cdot x^{2}+\frac{-63}{256}, \frac{12155}{128}\cdot x^{9}+\frac{-6435}{32}\cdot x^{7}+\frac{9009}{64}\cdot x^{5}+\frac{-1155}{32}\cdot x^{3}+\frac{315}{128}\cdot x, [$$

```
a:=inv(trn(vandermonde(l))):; a*seq((1+(-1)^j)/(j+1),j,0,n-1);
```

Done, $[0.0666713443087, 0.149451349151, 0.219086362516, 0.26926671931, 0.295524224715, 0.295524224715, 0.26926671931,$

```
seq(2/(1-l[j]^2)/P'(x=l[j])^2,j,0,n-1)
```

$[0.0666713443087, 0.149451349151, 0.219086362516, 0.26926671931, 0.295524224715, 0.295524224715, 0.26926671931, 0.219$

```
seq(2*(1-l[j]^2)/n^2/Q(x=l[j])^2,j,0,n-1)
```

$[0.0666713443081, 0.149451349151, 0.219086362516, 0.26926671931, 0.295524224715, 0.295524224715, 0.26926671931, 0.219$

```
seq(2/n/Q(x=l[j])/P'(x=l[j]),j,0,n-1)
```

$[0.0666713443084, 0.149451349151, 0.219086362516, 0.26926671931, 0.295524224715, 0.295524224715, 0.26926671931, 0.219$

Preuve de la dernière formule :
On a $I(P_n/(x-x_i)) = w_i P_n'(x_i)$ la valeur de $(P_n/(x-x_i))$ en x_i est par définition de $P_n'(x_i)$ la limite $(P_n(x) - P_n(x_i))/(x-x_i)$ lorsque x tend vers x_i (rappelons que $P_n(x_i) = 0$). Par exactitude de la formule d'intégration

$$w_i P_n'(x_i) = \int_{-1}^{1} \frac{P_n}{x-x_i} = \frac{1}{P_{n-1}(x_i)} \int_{-1}^{1} P_n \frac{P_{n-1}(x_i)}{x-x_i} = \frac{1}{P_{n-1}(x_i)} \int_{-1}^{1} P_n \frac{P_{n-1}(x)}{x-x_i}$$

la dernière égalité résulte du fait que $(P_{n-1}(x) - P_{n-1}(x_i))/(x-x_i)$ est un polynôme de degré au plus $n-2$ donc orthogonal à P_n. Donc

$$w_i P_n'(x_i) == \frac{1}{P_{n-1}(x_i)} \int_{-1}^{1} P_{n-1} \frac{P_n(x)}{x-x_i}$$

Or $P_n(x)/(x-x_i)$ est un polynôme de degré $n-1$, si on fait son quotient par P_{n-1} on obtient une constante qui vaut a_{n-1} en utilisant la relation de récurrence à 2 crans $P_n = (a_{n-1}x - b_{n-1})P_{n-1} - c_{n-1}P_{n-2}$. Le reste est de degré $n-2$ donc orthogonal à P_{n-1}, d'où

$$w_i P_n'(x_i) == \frac{1}{P_{n-1}(x_i)} \int_{-1}^{1} P_{n-1}^2 a_{n-1}$$

On conclut en utilisant la valeur de $a_{n-1} = (2n-1)/n$ et de $\|P_{n-1}\|^2 = 2/(2(n-1)+1)$.

5.9.3 Erreur d'une quadrature gaussienne

On considère une quadrature gaussienne d'ordre $2n+1$ obtenue par interpolation aux $n+1$ racines du $n+1$-ième polynome de Legendre sur $[-1,1]$. Rappelons que le polynôme d'interpolation de f aux points $x_0, x_1, ..., x_n$ s'écrit

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

Si on ajoute x on obtient un polynôme d'interpolation de degré $n+1$

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

qui coincide avec f en $t=x$ donc

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

Le même calcul fait avec un point x_{n+1} en plus donne

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

On a donc montré la formule :

$$f[x_0,...,x_n,x] = f[x_0,...,x_n,x_{n+1}] + f[x_0,...,x_n,x_{n+1},x](x-x_{n+1}) \tag{10}$$

mais en fait on va ajouter plus qu'un point, on va en ajouter $n+1$

$$\begin{aligned} & f(x) - (f[x_0] + f[x_0,x_1](x-x_0) + ... + f[x_0,...,x_n](x-x_0)...(x-x_{n-1})) \\ = \quad & f[x_0,...,x_n,x_{n+1}](x-x_0)...(x-x_{n-1})(x-x_n) + ... + f[x_0,...,x_{2n+1}](x-x_0)...(x-x_{2n}) + f[x_0,...,x_{2n+1},x](x-x \end{aligned}$$

puis on intègre sur $[-1,1]$, on obtient à gauche l'erreur et à droite, seul le dernier terme est non nul, car le polynôme de Legendre proportionnel à $(x-x_0)...(x-x_{n-1})(x-x_n)$, de degré $n+1$, est orthogonal à tous les polynôme de degré $\leq n$

$$\int_{-1}^{1} (f(x) - P_n(x))\, dx = \int_{-1}^{1} f[x_0,...,x_{2n+1},x](x-x_0)...(x-x_{2n+1})\, dx$$

D'autre part, le résultat sur l'erreur d'interpolation 1 donne

$$f[x_0,...,x_n,x] = \frac{f^{[n+1]}(\xi_x)}{(n+1)!}$$

en particulier $|f[x_0,...,x_n,x]| \leq M_{n+1}/(n+1)!$ D'où la majoration

$$|\int_{-1}^{1} (f(x) - P_n(x))\, dx| \leq \frac{M_{2n+2}}{(2n+2)!} \int_{-1}^{1} |(x-x_0)...(x-x_{2n+1})|\, dx$$

Il suffit ensuite de faire tendre les $x_{n+1}, ..., x_{2n+1}$ vers $x_0, ..., x_n$ pour enlever la valeur absolue et obtenir

$$|\int_{-1}^{1} (f(x) - P_n(x))\, dx| \leq \frac{M_{2n+2}}{(2n+2)!} \int_{-1}^{1} L_{n+1}^2(x)\, dx$$

où L_{n+1} est proportionnel au $n+1$-ième polynôme de Legendre de coefficient dominant 1. Par exemple, pour $n=1$, on a 2 points d'interpolation en les racines de $L_2 = x^2 - 1/3$ et l'erreur d'interpolation est majorée par $M_4/24 \int_{-1}^{1} (x^2 - 1/3)\, dx = M_4/24 \times 8/45 = M_4/135$.

5.10 Méthode adaptative.

On calcule une valeur approchée de l'intégrale sur $[a, b]$ par deux quadratures gaussiennes emboitées, on estime l'erreur, si elle est supérieure à la tolérance on divise en 2. On recommence en subdivisant en 2 l'intervalle où l'erreur est maximale. On s'arrête lorsque l'erreur estimée est inférieure à la tolérance. L'estimation de l'erreur se fait par exemple avec deux quadratures gaussiennes emboitées (c'est-à-dire que les points d'interpolation de la moins fine sont contenues dans les points d'interpolation de la plus fine, pour éviter de devoir calculer la fonction en de nouveaux points, on considére alors l'erreur sur la quadrature la moins fine comme la valeur absolue de la différence des deux valeurs). Ou avec trois quadratures emboitées, Hairer propose de prendre comme quadrature la plus fine en h^{30} (15 points), intermédiaire en h^{14} (avec les mêmes points sauf le point central), moins fine en h^6 (avec les points 1, 3, 5, 9, 11, 13), et d'estimer l'erreur par

$$\epsilon_1 = |I_{30} - I_{14}|, \epsilon_2 = |I_{30} - I_6|; \epsilon = \epsilon_1 \left(\frac{\epsilon_1}{\epsilon_2}\right)^2$$

On observe en effet que ϵ est en h^{30}, comme l'ordre de la méthode.

5.11 Accélération de Richardson-Romberg

Proposition 12 *Soit g une fonction de classe C^{2k} sur $[a, b]$, $T_h(g)$ la valeur de la méthode des trapèzes sur $[a, b]$ de pas $h = (b - a)/N$ (N entier). Alors $T_h(g)$ admet un développement en puissances paires de h à l'ordre $2k$.*

Pour montrer ce résultat, il faut établir la formule d'Euler-Mac Laurin. On commence par se placer sur une subdivision de l'intervalle $[0, 1]$, on intègre par parties $\int_0^1 f(t)\ dt$ en faisant apparaitre la formule des trapèzes, on intègre donc 1 en $t - \frac{1}{2}$

$$\int_0^1 f(t)\ dt = [(t - \frac{1}{2})f(t)]_0^1 - \int_0^1 (t - \frac{1}{2})f'(t)\ dt = \frac{f(0) + f(1)}{2} - \int_0^1 (t - \frac{1}{2})f'(t)\ dt$$

où encore

$$T_1(f) = \int_0^1 f(t)\ dt + \int_0^1 (t - \frac{1}{2})f'(t)\ dt$$

Pour poursuivre, on pose $B_1(t) = t - \frac{1}{2}$, qu'on va intégrer en $\frac{1}{2}((t - \frac{1}{2})^2 + c)$, où on choisit c pour que l'intégrale soit nulle, donc $c = -1/6$. On pose $B_2 = (t - \frac{1}{2})^2 - 1/6$, on a :

$$T_1(f) = \int_0^1 f(t)\ dt + [\frac{B_2}{2}f']_0^1 - \int_0^1 \frac{B_2}{2}f''(t)\ dt$$

Plus généralement, on pose

$$B'_{p+1} = pB_p, \quad \int_0^1 B_{p+1}(t)\ dt = 0$$

ce qui définit de manière unique les B_p. La nullité de l'intégrale montre que $B_{p+1}(1) = B_{p+1}(0)$ ce qui simplifiera l'expression des termes tout intégrés. De plus, on montre par récurrence que les B_p ont une symétrie paire ou impaire selon la parité de p par rapport à $t = 1/2$. Après p intégrations par parties, on obtient :

$$\begin{aligned} T_1(f) &= \int_0^1 f(t)\ dt + \frac{B_2(0)}{2}(f'(1) - f'(0)) + ... \\ &\quad + \frac{B_{2k}(0)}{(2k)!}(f^{[2k-1]}(1) - f^{[2k-1]}(0)) - \int_0^1 \frac{B_{2k}}{(2k)!}f^{[2k]}(t)\ dt \end{aligned}$$

En faisant le même raisonnement sur $[k, k+1]$ pour $k = 1, ..., N-1$ et en sommant, on obtient la formule d'**Euler-Mac Laurin** :

$$\begin{aligned} T_1^{[0,N]}(f) &= \int_0^N f(t)\,dt + \frac{B_2(0)}{2}(f'(N) - f'(0)) + ... \\ &\quad + \frac{B_{2k}(0)}{(2k)!}(f^{[2k-1]}(N) - f^{[2k-1]}(0)) - \int_0^N \frac{B_{2k}}{(2k)!} f^{[2k]}(t)\,dt \end{aligned}$$

On pose alors $x = a + ht$ (donc $dt = dx/h$) et $f(t) = g(x)$ (donc $f'(t) = df/dt = hdg/dx$), on obtient

$$\begin{aligned} \frac{1}{h} T_h^{[a,b]}(g) &= \frac{1}{h}\int_0^N g(x)\,dx + \frac{B_2(0)}{2} h(g'(b) - g'(a)) + ... \\ &\quad + \frac{B_{2k}(0)}{(2k)!} h^{2k-1}(g^{[2k-1]}(N) - g^{[2k-1]}(0)) - \int_0^N \frac{B_{2k}}{(2k)!} h^{2k} g^{[2k]}(x)\,\frac{1}{h}dx \end{aligned}$$

donc

$$\begin{aligned} T_h^{[a,b]}(g) &= \int_0^N g(x)\,dx + h^2\frac{B_2(0)}{2}(g'(b) - g'(a)) + ... \\ &\quad + h^{2k}\frac{B_{2k}(0)}{(2k)!}(g^{[2k-1]}(N) - g^{[2k-1]}(0)) - h^{2k}\int_0^N \frac{B_{2k}}{(2k)!} g^{[2k]}(x)\,dx \end{aligned}$$

L'accélération consiste à éliminer les puissances de h^2 en commençant par h^2 avec des subdivisions deux fois plus fines à chaque itération. Ainsi $T_h^1(f) = (4T_{h/2}(f) - T_h(f))/(4-1)$ n'a plus de termes en h^2 et tend vers l'intégrale à approcher lorsque h tend vers 0. On peut d'ailleurs vérifier qu'il s'agit de la méthode de Simpson. On élimine ensuite le terme en h^4 en posant $T_h^2(f) = (4^2 T_{h/2}^1(f) - T_h^1(f))/(4^2-1)$ et ainsi de suite. On construit un tableau triangulaire `T` dont chaque ligne l contient $T_{h/2^l}(f), T_{h/2^l}^1(f), ...$ (avec des indices qui commencent à 0). Pour calculer le terme d'indice 0 de la ligne courante on fait une méthode des trapèzes sur 2 fois plus de subdivisions que la précédente, puis pour le j-ième terme `T[l,j]` on effectue `(4^j*T[l-1,j-1]-T[l,j-1])/(4^j-1)` (on n'a donc besoin que de la ligne précédente pour calculer la ligne courante). On s'arrête par exemple lorsque la valeur absolue de la différence entre les derniers termes de deux lignes consécutives est inférieur à la précision souhaitée (erreur empirique).

5.12 Méthodes probabilistes.

Pour déterminer $\int_a^b f(t)\,dt$, on l'interprète comme une espérance, plus précisément comme $(b-a)E(f(X))$ où X est une variable aléatoire qui suit la loi uniforme sur $[a, b]$, et on approche cette valeur par

$$\frac{b-a}{n}\sum_{k=1}^{n} f(x_k)$$

où x_k est obtenu par un générateur pseudo-aléatoire (selon la loi uniforme). Par exemple

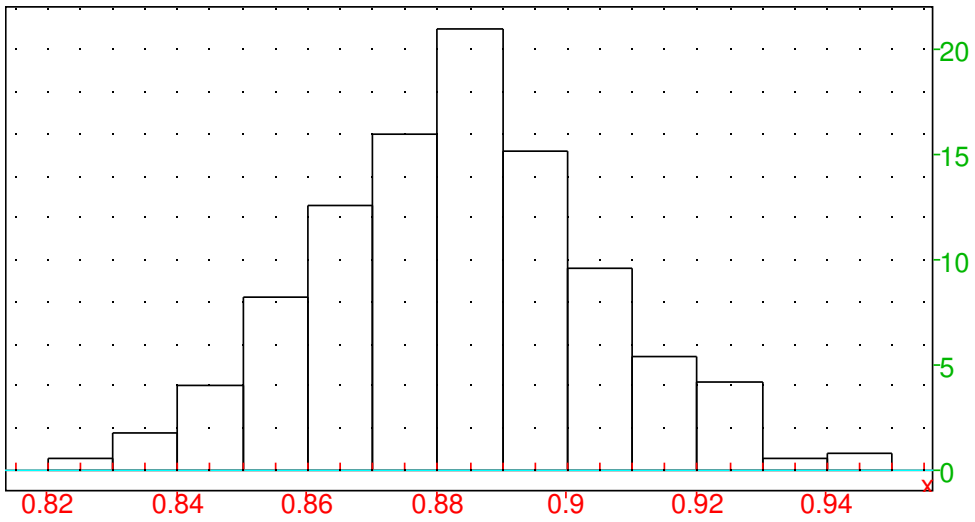
```
f(t):=exp(-t^2); n:=1000; a:=0; b:=2.0;l:=ranv(n,uniform,a,b);;
```

```
(t)->exp(-t^2)
```
$$, 1000, 0, 2.0, \text{Done}$$

```
(b-a)*sum(apply(f,l))/n;  int(f(t),t,a,b);
```
$$0.883406466146, 0.882081390762$$

La convergence en fonction de n est assez lente, on peut l'observer en faisant plusieurs estimations :

```
m:=ranm(500,n,uniform,a,b):;I:=seq(2*sum(apply(f,m[k]))/n,k,0,size(m)-1);
```



En effet, les tirages sont équidistribués selon la même loi, la loi des grands nombres s'applique donc : on fait $b-a$ fois une moyenne de n tirages, si n est grand, on converge vers une loi normale dont l'écart-type est en $(b-a)\sigma/\sqrt{n}$. La valeur de la constante σ peut se calculer à partir de f

$$\sigma^2 = \frac{1}{b-a}\int_a^b f^2(t)\,dt - \left(\frac{1}{b-a}\int_a^b f(t)\,dt\right)^2$$

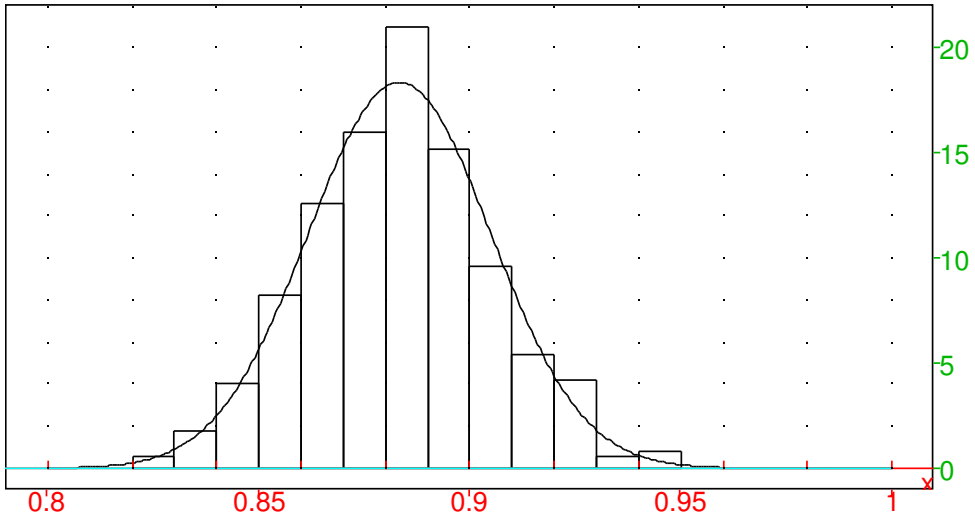
par exemple ici

```
2*sqrt(int(f(t)^2,t,0,2.)/2-(1/2*int(f(t),t,0,2.))^2)/sqrt(n);
 stddevp(I);
```

$$0.0217983295084, 0.0217106474709$$

mais on ne fait pas ce calcul en pratique (puisqu'il faudrait calculer une intégrale), on estime l'écart-type $\sigma/\sqrt{n}$ de la loi normale par l'écart-type de l'échantillon des estimations `stddevp(I)`.

```
histogram(I,0,0.01); plot(normald(mean(I),stddevp(I),x),x=0.8..1)
```



On peut donc obtenir rapidement une estimation de σ en prenant l'écart-type d'une séquence de valeurs de f

```
f(t):=exp(-t^2); n:=1000; a:=0;  b:=2.0;l:=ranv(n,uniform,a,b):;
```

```
   (t)->exp(-t^2)                                                    , 1000, 0, 2.0, Done
```

```
fl:=apply(f,l):;m:=(b-a)*mean(fl);s:=(b-a)*stddevp(fl)/sqrt(n);[m-2s,m+2s]
```

$$\text{Done}, 0.875737912285, 0.0222019063389, [0.831334099607, 0.920141724963]$$

Cette méthode converge donc beaucoup moins vite que les quadratures, en dimension 1. Mais elle se généralise très facilement en dimension plus grande en conservant la même vitesse de convergence alors que le travail nécessaire pour une méthode de quadrature croit comme une puissance de la dimension, et ne nécessite pas de paramétrer des domaines d'intégration compliqués (il suffit par exemple d'utiliser la méthode du rejet pour avoir un générateur uniforme dans un domaine inclus dans un cube).

6 Suites itératives et applications

Résumé :
Théorème du point fixe, méthode de Newton,convexité. Exemple : calcul de valeur approchée de racines carrées, Résolution d'équations.

6.1 Le point fixe dans $\mathbb{R}$

Soit f une fonction continue sur un intervalle $I = [a, b]$ de $\mathbb{R}$, et à valeurs dans I (attention à bien choisir I pour que l'image de I par f reste dans I). On s'intéresse à la suite

$$u_{n+1} = f(u_n), \quad u_0 \in I \tag{11}$$

Supposons que u_n converge vers une limite $l \in I$ lorsque $n \to +\infty$, alors la limite doit vérifier

$$f(l) = l$$

puisque f est continue. On dit que l est un point fixe de f. Ceci amène à l'idée d'utiliser ces suites pour résoudre numériquement l'équation $f(x) = x$. Nous allons donner un théorème permettant d'assurer que la suite (11) converge, et que la limite est l'unique solution de $f(l) = l$ sur I.

Définition 3 *On dit que f est contractante de rapport $k < 1$ sur I si*

$$\forall x, y \in I, \quad |f(y) - f(x)| \leq k|y - x|$$

En pratique, les fonctions f que l'on considèrera seront continument dérivables, donc d'après le théorème des accroissements finis

$$f(y) - f(x) = f'(\theta)(y - x), \quad \theta \in [x, y]$$

ainsi pour vérifier que f est contractante, on étudie la valeur absolue de f' sur I, il suffit de montrer que cette valeur absolue est strictement inférieure à un réel $k < 1$ pour conclure (il faut donc chercher le maximum de $|f'|$ sur I. Attention, il s'agit du maximum de $|f'|$ et pas du maximum de f', ce qui revient à chercher le maximum de f' et de $-f'$). On a alors le

Théorème 13 *(du point fixe)*
si f est contractante de $I = [a, b]$ dans I de rapport k alors la suite (11) converge vers l'unique solution de $f(l) = l$ dans I. On a de plus les encadrements :

$$|u_n - l| \leq k^n |b - a|, \quad |u_n - l| \leq \frac{|u_{n+1} - u_n|}{1 - k} \tag{12}$$

Démonstration : Tout d'abord si f est contractante, on montre à partir de la définition de la continuité que f est continue. Soit $g(x) = f(x) - x$, alors g est continue, positive en a et négative en b, il existe donc $l \in [a, b]$ tel que $g(l) = 0$ (théorème des valeurs intermédiaires). Soit u_n une suite définie par (11). On a alors pour tout n

$$|u_{n+1} - l| = |f(u_n) - f(l)| \leq k|u_n - l|$$

Donc par une récurrence évidente :

$$|u_n - l| \leq k^n |u_0 - l|$$

ce qui entraine d'ailleurs que $|u_n - l| \leq k^n|a-b|$. Comme $k \in [0,1[$, la suite géométrique k^n converge vers 0 lorsque n tend vers l'infini, donc u_n tend vers l. Notons que l est unique car si l' est une autre solution alors $|l-l'| = |f(l) - f(l')| \leq k|l-l'|$ donc $(1-k)|l-l'| \leq 0$, or $1-k>0$ et $|l-l'| \geq 0$ donc $|l-l'|$ doit être nul. Passons à la preuve de la majoration (12) qui est importante en pratique car elle donne un test d'arrêt de calcul des termes de la suite récurrente, on écrit pour $m>0$:

$$u_n - l = u_n - u_{n+1} + u_{n+1} - u_{n+2} + ... + u_{n+m-1} - u_{n+m} + u_m - l$$

puis on majore avec l'inégalité triangulaire

$$|u_n - l| \leq \sum_{j=0}^{m-1} |u_{n+j} - u_{n+j+1}| + |u_m - l|$$

puis on applique le fait que f est contractante de rapport k

$$|u_n - l| \leq \sum_{j=0}^{m-1} k^j |u_n - u_{n+1}| + |u_m - l|$$

soit

$$|u_n - l| \leq \frac{1-k^m}{1-k}|u_n - u_{n+1}| + |u_m - l|$$

On fait alors tendre m vers l'infini d'où le résultat. Exemples : Cherchons une valeur approchée de $\sqrt{2}$ par cette méthode. Il faut d'abord trouver une fonction f dont $\sqrt{2}$ est un point fixe, par exemple

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

On vérifie que $f(\sqrt{2}) = \sqrt{2})$, puis que $f' = -1/(x+1)^2$ donc f décroit. On va voir si les hypothèses du théorème du point fixe s'appliquent sur par exemple $[1,2]$. Comme f est décroissante $f([1,2]) = [f(2), f(1)] = [4/3, 3/2]$ qui est bien inclus dans $[1,2]$. De plus f' est comprise entre $-1/(1+1)^2 = -1/4$ et $-1/(2+1)^2 = -1/9$ donc $|f'| < 1/4$, f est contractante de rapport $1/4$. On peut donc itérer la suite à partir par exemple de $u_0 = 1$ et on va converger vers $\sqrt{2}$ (en s'en rapprochant à chaque cran d'un rapport inférieur à $1/4$). Considérons l'équation en x

$$x - e\sin(x) = t, \quad e \in [0,1[$$

c'est l'équation du temps utilisée en astronomie pour trouver la position d'une planète sur son orbite elliptique (e étant l'excentricité de l'ellipse). Il n'y a pas de formule exacte permettant de calculer x en fonction de t. Si on a une valeur numérique pour t, on peut trouver une valeur numérique approchée de x par la méthode du point fixe, en réécrivant l'équation sous la forme

$$f(x) = t + e\sin(x) = x$$

On observe que f envoie $\mathbb{R}$ dans $[t-e, t+e]$ donc on peut prendre $I = [t-e, t+e]$, de plus $|f'| \leq e < 1$, f est contractante de rapport $e \in [0,1[$, le théorème s'applique, il suffit de prendre une valeur initiale dans $[t-e, t+e]$ et d'itérer la suite jusqu'à obtenir la précision désirée. Par exemple si on veut une valeur approchée de x à 10^{-6} près, il suffira que la différence entre deux termes successifs de la suite u_n vérifie

$$|u_{n+1} - u_n| \leq 10^{-6}(1-e)$$

on aura alors bien :

$$|u_n - x| \leq \frac{|u_{n+1} - u_n|}{1 - e} \leq 10^{-6}$$

Cette méthode n'est pas toujours optimale, car la vitesse de convergence vers la limite l est dite "linéaire", c'est-à-dire que le temps de calcul pour avoir n décimales est proportionnel à n (ou encore il faut effectuer un nombre d'itérations proportionnel à n, chaque itération faisant gagner en précision de l'ordre du rapport k de contractance). En effet, supposons que f' est continue en l et que $0 < L = |f'(l)| < 1$. Il existe alors un intervalle $I = [l-\eta, l+\eta]$ tel que

$$x \in I \Rightarrow \frac{L}{2} \leq |f'(x)| \leq \frac{1+L}{2}$$

Le théorème des accroissements finis donne alors

$$|u_{n+1} - l| = |f(u_n) - f(l)| = |f'(\theta)||u_n - l|, \quad \theta \in [u_n, l]$$

Si $u_0 \in I$, alors $\theta \in I$ donc $|u_1 - l| \leq |u_0 - l|$ et $u_1 \in I$, par récurrence on a pour tout n, $u_n \in I$

$$\frac{L}{2}|u_n - l| \leq |u_{n+1} - l| \leq \frac{1+L}{2}|u_n - l|$$

on a donc par récurrence

$$\left(\frac{L}{2}\right)^n |u_0 - l| \leq |u_n - l| \leq \left(\frac{1+L}{2}\right)^n |u_0 - l|$$

Donc pour avoir $|u_n - l| \leq \epsilon$ il suffit que

$$\left(\frac{1+L}{2}\right)^n |u_0 - l| \leq \epsilon \Rightarrow n \geq \frac{\ln(\frac{\epsilon}{|u_0 - l|})}{\ln(\frac{1+L}{2})}$$

et il faut que

$$\left(\frac{L}{2}\right)^n |u_0 - l| \leq \epsilon \Rightarrow n \geq \frac{\ln(\frac{\epsilon}{|u_0 - l|})}{\ln(\frac{L}{2})}$$

Si f est suffisamment régulière, il existe une méthode plus rapide lorsqu'on est proche de la racine ou lorsque la fonction a des propriétés de convexité, c'est la méthode de Newton. Et même si Newton n'est pas applicable, une simple dichotomie peut être plus efficace si la constante de contractance est supérieure à $1/2$ (y compris prés de la solution de $f(x) = x$). Toutefois la méthode du point fixe reste intéressante si la constante de contractance est suffisamment petite (par exemple $k = 0.1$ garantit 15 décimales en 15 itérations) et présente l'avantage de se généraliser en dimension plus grande, cf. la section suivante.

6.2 Le point fixe dans $\mathbb{R}^n$

Le théorème précédent se généralise.

Théorème 14 *Soit I un ensemble fermé de $\mathbb{R}^n$ (ou d'un espace métrique complet) tel que f envoie I dans I et tel que f soit contractante sur I*

$$\exists k < 1, \forall x, y \in I, \quad |f(x) - f(y)| \leq k|x - y|$$

Alors pour tout $u_0 \in I$, la suite (u_n) définie par $u_{n+1} = f(u_n)$ converge vers l'unique solution dans I de $f(l) = l$.

La démonstration de la convergence est un peu différente de celle donnée en dimension 1, on montre que (u_n) est une suite de Cauchy, car pour $n > m$

$$|u_n - u_m| \leq \sum_{j=m}^{n-1} |u_{j+1} - u_j| \leq k^m \frac{1}{1-k} |u_1 - u_0|$$

donc (u_n) est convergente puisque nous sommes dans un fermé d'un espace complet. (Cela permet d'ailleurs de généraliser l'énoncé donné en dimension 1 au cas où a ou b est infini) La vitesse de convergence est linéaire, la démonstration est identique à celle de la dimension 1. Remarque :

- L'existence d'un point fixe sans hypothèse de contractance se généralise si I est un convexe compact préservé par f (théorème de Brouwer ou de Schauder).
- Pour vérifier les hypothèses du théorème dans $\mathbb{R}^n$, il suffit de montrer que dans I la norme triple de f' subordonnée à la norme choisie dans $\mathbb{R}^n$ est inférieure à $k < 1$. Pour f linéaire, cela revient à calculer une norme subordonnée de matrice, et donne lieu à des méthodes itératives alternatives à l'inversion de matrice, cf. la section 3.6.2.
- l'algorithme de recherche PageRank de google utilise le point fixe, en très grande dimension : n est le nombre de pages Web, I est l'ensemble des vecteurs de $\mathbb{R}^n$ dont toutes les coordonnées sont positives ou nulles et dont la somme des coordonnées vaut 1, f est la somme d'un vecteur constant et du produit du vecteur x par une matrice A transposée d'une matrice stochastique.

6.3 La méthode de Newton dans $\mathbb{R}$.

La méthode de Newton est une méthode de résolution de l'équation $f(x) = 0$, attention à la différence avec le théorème du point fixe qui permet de résoudre numériquement $f(x) = x$. Si x_0 est proche de la racine r on peut faire un développement de Taylor à l'ordre 1 de la fonction f en x_0 :

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + O((x - x_0)^2)$$

Pour trouver une valeur approchée de r, on ne garde que la partie linéaire du développement, on résout :

$$f(r) = 0 \approx f(x_0) + (r - x_0)f'(x_0)$$

donc (si $f'(x_0) \neq 0$) :

$$r \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

Graphiquement, cela revient à tracer la tangente à la courbe représentative de f et à chercher où elle coupe l'axe des x. On considère donc la suite récurrente définie par une valeur u_0 proche de la racine et par la relation :

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}$$

Il y a deux théorèmes importants, l'un d'eux prouve que si u_0 est "assez proche" de r alors la suite u_n converge vers r, malheureusement il est difficile de savoir en pratique si on est "assez proche" de u_0 pour que ce théorème s'applique. Le second théorème donne un critère pratique facile à vérifier qui assure la convergence, il utilise les propriétés de convexité de la fonction.

Théorème 15 *Soit f une fonction de classe C^2 (2 fois continument dérivable) sur un intervalle fermé I. Soit r une racine simple de f située à l'intérieur de I (telle que $f(r) = 0$ et $f'(r) \neq 0$). Alors il existe $\varepsilon > 0$ tel que la suite définie par*

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)}, \quad |u_0 - r| \leq \varepsilon$$

converge vers r. Si on a $|f''| \leq M$ et $|1/f'| \leq m$ sur un intervalle $[r - \eta, r + \eta]$ contenu dans I, alors on peut prendre tout réel $\varepsilon > 0$ tel que $\varepsilon < 2/(mM)$ et $\varepsilon \leq \eta$.

Démonstration : on a

$$u_{n+1} - r = u_n - r - \frac{f(u_n)}{f'(u_n)} = \frac{(u_n - r)f'(u_n) - f(u_n)}{f'(u_n)}$$

En appliquant un développement de Taylor de f en u_n à l'ordre 2, on obtient pour un réel θ situé entre r et u_n :

$$0 = f(r) = f(u_n) + (r - u_n)f'(u_n) + (r - u_n)^2 \frac{f''(\theta)}{2}$$

donc :

$$(u_n - r)f'(u_n) - f(u_n) = (u_n - r)^2 \frac{f''(\theta)}{2}$$

d'où :

$$|u_{n+1} - r| \leq |u_n - r|^2 \frac{1}{|f'(u_n)|} \frac{|f''(\theta)|}{2}$$

On commence par choisir un intervalle $[r - \varepsilon, r + \varepsilon]$ contenant strictement r et tel que $|f''| < M$ et $|1/f'| < m$ sur $[r - \varepsilon, r + \varepsilon]$ (c'est toujours possible car f'' et $1/f'$ sont continues au voisinage de r puisque $f'(r) \neq 0$). Si u_n est dans cet intervalle, alors θ aussi donc

$$|u_{n+1} - r| \leq |u_n - r|^2 \frac{Mm}{2} \leq \frac{|u_n - r|Mm}{2} |u_n - r|, \tag{13}$$

On a $|u_n - r| \leq \varepsilon$, on diminue si nécessaire ε pour avoir $\varepsilon < 2/(Mm)$, on a alors :

$$|u_{n+1} - r| \leq k|u_n - r|, \quad k = \frac{\varepsilon Mm}{2} < 1$$

donc d'une part u_{n+1} est encore dans l'intervalle $[r - \varepsilon, r + \varepsilon]$ ce qui permettra de refaire le même raisonnement au rang suivant, et d'autre part on a une convergence au moins géométrique vers r. En fait la convergence est bien meilleure lorsqu'on est proche de r grace au carré dans $|u_n - r|^2$, plus précisément, on montre par récurrence que

$$|u_n - r| \leq |u_0 - r|^{2^n} \left(\frac{Mm}{2}\right)^{2^n - 1}$$

il faut donc un nombre d'itérations proportionnel à $\ln(n)$ pour atteindre une précision donnée. **Remarque :** ce théorème se généralise sur $\mathbb{C}$ et même sur $\mathbb{R}^n$ (cf. la section suivante). **Exemple :** pour calculer $\sqrt{2}$, on écrit l'équation $x^2 - 2 = 0$ qui a $\sqrt{2}$ comme racine simple sur $I = [1/2, 2]$, on obtient la suite récurrente

$$u_{n+1} = u_n - \frac{u_n^2 - 2}{2u_n}$$

Si on prend $\eta = 1/2$, on a $f' = 2x$ et $f'' = 2$ donc on peut prendre $M = 2$ et $m = 1$ car $|1/f'| \leq 1$ sur $[\sqrt{2} - 1/2, \sqrt{2} + 1/2]$. On a $2/(mM) = 1$, on peut donc prendre $\varepsilon = 1/2$, la suite convergera pour tout $u_0 \in [\sqrt{2} - 1/2, \sqrt{2} + 1/2]$. Plus généralement, on peut calculer une racine k-ième d'un réel a en résolvant $f(x) = x^k - a$ par la méthode de Newton. L'inconvénient de ce théorème est qu'il est difficile de savoir si la valeur de départ qu'on a choisie se trouve suffisamment près d'une racine pour que la suite converge. Pour illustrer le phénomène, on peut par exemple colorer les points du plan complexe en $n + 1$ couleurs selon que la suite définie par la méthode de Newton converge vers l'une des n racines d'un polynôme de degré n fixé au bout de par exemple 50 itérations (la $n+1$-ième couleur servant aux origines de suite qui ne semblent pas converger). Passons maintenant à un critère très utile en pratique :

Définition 4 *(convexité)*
Une fonction f continument dérivable sur un intervalle I de $\mathbb{R}$ est dite convexe si son graphe est au-dessus de la tangente en tout point de I.

Il existe un critère simple permettant de savoir si une fonction de classe C^2 est convexe :

Théorème 16 *Si f est C^2 et $f'' \geq 0$ sur I alors f est convexe.*

Démonstration :
L'équation de la tangente au graphe en x_0 est

$$y = f(x_0) + f'(x_0)(x - x_0)$$

Soit

$$g(x) = f(x) - (f(x_0) + f'(x_0)(x - x_0))$$

on a :

$$g(x_0) = 0, \quad g'(x) = f'(x) - f'(x_0), \quad g'(x_0) = 0, \quad g'' = f'' \geq 0$$

donc g' est croissante, comme $g'(x_0) = 0$, g' est négative pour $x < x_0$ et positive pour $x > x_0$, donc g est décroissante pour $x < x_0$ et croissante pour $x > x_0$. On conclut alors que $g \geq 0$ puisque $g(x_0) = 0$. Donc f est bien au-dessus de sa tangente. On arrive au deuxième théorème sur la méthode de Newton

Théorème 17 *Si $f(r) = 0, f'(r) > 0$ et si $f'' \geq 0$ sur $[r, b]$ alors pour tout $u_0 \in [r, b]$ la suite de la méthode de Newton*

$$u_{n+1} = u_n - \frac{f(u_n)}{f'(u_n)},$$

est définie, décroissante, minorée par r et converge vers r. De plus

$$0 \leq u_n - r \leq \frac{f(u_n)}{f'(r)}$$

Démonstration :
On a $f'' \geq 0$ donc si $f'(r) > 0$ alors $f' > 0$ sur $[r, b]$, f est donc strictement croissante sur $[r, b]$ on en déduit que $f > 0$ sur $]r, b]$ donc $u_{n+1} \leq u_n$. Comme la courbe représentative de f est au-dessus de la tangente, on a $u_{n+1} \geq r$ (car u_{n+1} est l'abscisse du point d'intersection de la tangente avec l'axe des x). La suite u_n est donc décroissante minorée par r, donc convergente vers une limite $l \geq r$. À la limite, on a

$$l = l - \frac{f(l)}{f'(l)} \Rightarrow f(l) = 0$$

donc $l = r$ car $f > 0$ sur $]r, b]$. Comme (u_n) est décroissante, on a bien $0 \leq u_n - r$, pour montrer l'autre inégalité, on applique le théorème des accroissements finis, il existe $\theta \in [r, u_n]$ tel que

$$f(u_n) - f(r) = (u_n - r)f'(\theta)$$

comme $f(r) = 0$, on a

$$u_n - r = \frac{f(u_n)}{f'(\theta)}$$

et la deuxième inégalité du théorème en découle parce que f' est croissante. **Variantes** :
Il existe des variantes, par exemple si $f'(r) < 0$ et $f'' \geq 0$ sur $[a, r]$. Si $f'' \leq 0$, on considère $g = -f$.
Application :
On peut calculer la valeur approchée de la racine k-ième d'un réel $a > 0$ en appliquant ce deuxième théorème. En effet si $a > 0$, alors $x^k - a$ est 2 fois continument dérivable et de dérivée première kx^{k-1} et seconde $k(k-1)x^{k-2}$ strictement positives sur $\mathbb{R}^{+*}$ (car $k \geq 2$). Il suffit donc de prendre une valeur de départ u_0 plus grande que la racine k-ième, par exemple $1 + a/k$ (en effet $(1 + a/k)^k \geq 1 + ka/k = 1 + a$). En appliquant l'inégalité du théorème, on a :

$$0 \leq u_n - \sqrt[k]{a} \leq \frac{u_n^k - a}{k\sqrt[k]{a}^{k-1}} \leq \frac{u_n^k - a}{ka}\sqrt[k]{a} \leq \frac{u_n^k - a}{ka}(1 + \frac{a}{k})$$

Pour avoir une valeur approchée de $\sqrt[k]{a}$ à ε près, on peut donc choisir comme test d'arrêt

$$u_n^k - a \leq \frac{ka}{1 + \frac{a}{k}}\varepsilon$$

Par exemple pour $\sqrt{2}$, le test d'arrêt serait $u_n^2 - 2 \leq 2\varepsilon$.

6.4 La méthode de Newton dans $\mathbb{R}^n$.

Le premier énoncé du cas de la dimension 1 se généralise en :

Théorème 18 *Soit f une fonction de classe C^2 (2 fois continument dérivable) sur un fermé I de $\mathbb{R}^n$. Soit r une racine simple de f située à l'intérieur de I (telle que $f(r) = 0$ et $f'(r) = (\partial_j f_i)(r)$ inversible). Alors il existe $\varepsilon > 0$ tel que la suite définie par*

$$u_{n+1} = u_n - (f'(u_n))^{-1} f(u_n), \quad |u_0 - r| \leq \varepsilon$$

converge vers r. Si on a $|f''| \leq M$ et $|(f')^{-1}| \leq m$ sur une boule centré en r de rayon $\eta > 0$ contenue dans I, alors on peut prendre tout réel $\varepsilon > 0$ tel que $\varepsilon < 2/(mM)$ et $\varepsilon \leq \eta$.

La démonstration est calquée sur la dimension 1, mais il faut prendre le reste intégral dans la formule de Taylor

$$u_{n+1} - r = u_n - r - f'(u_n)^{-1} f(u_n) = f'(u_n)^{-1}(f'(u_n)(u_n - r) - f(u_n))$$

puis on applique Taylor le long du segment $[r, u_n]$:

$$0 = f(r) = f(u_n) + f'(u_n)(r - u_n) + \int_0^1 (1 - \theta)(r - u_n)f''(r + \theta(u_n - r))(r - u_n)\, d\theta$$

donc :

$$u_{n+1} - r = -f'(u_n)^{-1}(r - u_n)\left(\int_0^1 (1-\theta)f''(r+\theta(u_n - r))\,d\theta\right)(r - u_n)$$

et on en déduit (13) et on conclut de même en remplaçant intervalle centré en r de rayon ε par boule de rayon ε. Remarque : la convergence "numérique" (au sens du calcul en flottant) de la suite u_n ne suffit pas à montrer l'existence d'une racine proche de u_n. Une méthode de preuve alternative au calcul des constantes m et M consiste à trouver un rectangle ou une boule autour de u_n préservée par l'application $x \to x - f'(x)^{-1}f(x)$.

6.5 Calcul approché des racines complexes simples

La section précédente nous a montré qu'on pouvait se ramener à la recherche de racines simples, ce qui donne envie d'essayer la méthode de Newton. On a malheureusement rarement la possibilité de pouvoir démontrer qu'à partir d'une valeur initiale donnée, la méthode de Newton converge, parce que les racines peuvent être complexes, et même si elles sont réelles, on n'a pas forcément de résultat sur la convexité du polynôme (cf. cependant une application des suites de Sturm qui permet de connaitre le signe de P'' sur un intervalle sans le factoriser). On effectue donc souvent des itérations de Newton, en partant de 0.0, en espérant s'approcher suffisamment d'une racine pour que le théorème de convergence théorique s'applique. On se fixe un nombre maximal d'itérations, si on le dépasse on prend alors une valeur initiale aléatoire complexe et on recommence. Une fois une racine déterminée, on l'élimine en calculant le quotient euclidien Q de P par $X - r$ (par l'algorithme de Horner), puis on calcule les racines du quotient Q (qui sont des racines de P). Un problème pratique apparait alors, c'est que r n'est pas exact donc le quotient Q non plus, au fur et à mesure du calcul des racines de P, on perd de plus en plus de précision. Il existe une amélioration simple, si r' est une racine approchée de Q, alors elle est racine approchée de P et on a toutes les chances qu'elle soit suffisamment proche d'une racine de P pour que le théorème s'applique, on effectue alors 1 ou 2 itérations de Newton avec r' mais pour P (et non Q) afin d'améliorer sa précision comme racine de P. Une méthode de calcul plus stable utilise la recherche des valeurs propres de la matrice companion en double précision, puis affine par la méthode de Newton pour obtenir des valeurs approchées multi-précision, c'est ce que fait `proot`, par exemple `proot(x^3+x+1,50)`. Enfin, on peut appliquer directement la méthode de Newton pour trouver dans $\mathbb{C}^n$ toutes les racines simultanément, c'est la méthode de **Durand-Kerner, Weierstrass**. On pose $g_x(z) = \prod_{i=1}^n (x - z_i)$, il s'agit de résoudre en z $g_x(z) = P(x)$. On a a l'ordre 1 en z

$$g_x(z+w) = g_x(z) - \sum_{i=1}^n w_i \prod_{j\neq i}(x - z_j) + O(w^2) = P(x)$$

pour trouver w_i, on pose $x = z_i$, on obtient

$$-\prod_{j\neq i}(z_i - z_j)w_i = P(z_i)$$

donc

$$w_i = -\frac{P(z_i)}{\prod_{j\neq i}(z_i - z_j)}$$

On peut aussi calculer le produit du dénominateur en effectuant $g'_x(z_i)$ (la dérivée porte sur x). On retrouve la méthode de Newton à une variable où la dérivée du polynôme au dénominateur est remplacée par la valeur approchée du polynôme. D'où le programme

```
dw(P,N,eps):={ // Weierstrass, Durand-Kerner polynomial rooter
  local l,v,w,n,j,k,q,q1;
  P:=P/lcoeff(P);
  n:=degree(P);
  assume(l,symbol);
  v:=seq(exp(i*l/n*2.0*pi),l,0,n-1); w:=v;
  for k from 1 to N do
    q:=pcoeff(v);
    q1:=q';
    for j from 0 to n-1 do
      w[j]:=v[j]-horner(P,v[j])/horner(q1,v[j]);
    od;
    if (l2norm(w-v)<eps*l2norm(v))
      return w;
    v:=w;
  od;
  retourne "max iter reached";
}:;
```

Par exemple `dw(x^3+x+1,100,1e-10)` renvoie des valeurs approchées des racines de x^3+x+1. Si on s'intéresse seulement à la racine de module maximal d'un polynôme, on peut en trouver une estimation assez simplement en appliquant la méthode de la puissance à la matrice companion du polynôme. On peut améliorer la précision d'une racine par des itérations inverses ou par la méthode de Newton en une variable.

7 Réduction approchée des endomorphismes

On pourrait trouver des valeurs propres approchées d'une matrice en calculant le polynome caractéristique ou minimal puis en le factorisant numériquement. Mais cette méthode n'est pas idéale relativement aux erreurs d'arrondis (calcul du polynome caractéristiaue, de ses racines, et nouvelle approximation en calculant le noyau de $A-\lambda I$), lorsqu'on veut calculer quelques valeurs propres on préfère utiliser des méthodes itératives directement sur A ce qui évite la propagation des erreurs d'arrondi.

7.1 Méthode de la puissance

Elle permet de déterminer la plus grande valeur propre en valeur absolue d'une matrice diagonalisable lorsque celle-ci est unique. Supposons en effet que les valeurs propres de A soient $x_1,...,x_n$ avec $|x_1| \leq |x_2| \leq ... \leq |x_{n-1}| < |x_n|$ et soient $e_1,...,e_n$ une base de vecteurs propres correspondants. On choisit un vecteur aléatoire v et on calcule la suite $v_k = Av_{k-1} = A^k v$. Si v a pour coordonnées $V_1,...,V_n)$ dans la base propre, alors

$$v_k = \sum_{j=1}^{n} V_j x_j^k e_j = x_n^k w_k, \quad w_k = \sum_{j=1}^{n} V_j \left(\frac{x_j}{x_n}\right)^k e_j$$

L'hypothèse que x_n est l'unique valeur propre de module maximal entraine alors que $\lim_{k\to+\infty} w_k = V_n e_n$ puisque la suite géométrique de raison x_j/x_n converge vers 0. Autrement dit, si $V_n \neq 0$ (ce qui a une probabilité 1 d'être vrai pour un vecteur aléatoire), v_k est équivalent à $V_n x_n^k e_n$. Lorsque n est grand, v_k est presque colinéaire

au vecteur propre e_n (que l'on peut estimer par v_k divisé par sa norme), ce que l'on détecte en testant si v_{k+1} et v_k sont presques colinéaires. De plus le facteur de colinéarité entre v_{k+1} et v_k est presque x_n, la valeur propre de module maximal. Exercice : tester la convergence de $v_k = A^k v$ vers l'espace propre associé à $\lambda = 3$ pour la matrice `[[1,-1],[2,4]]` et le vecteur $v = (1, 0)$. Attention à ne pas calculer A^k pour déterminer v_k, utiliser la relation de récurrence ! Si on n'observe pas de convergence ou si elle est trop lente, alors $|x_{n-1}|$ est proche de $|x_n|$ ou égal, il est judicieux de faire subir à la matrice un shift, on remplace A par $A - \lambda I$. On peut prendre λ aléatoirement, ou bien mieux faire des itérations inverses sur $A - \lambda I$ si λ est une estimation d'une valeur propre (voir les itérations inverses ci-dessous). Lorsqu'on applique cette méthode a une matrice réelle, il peut arriver quíl y ait deux valeurs propres conjuguées de module maximal. On peut appliquer la méthode ci-dessus avec un shift complexe non réel, mais on doit alors travailler en arithmétique complexe ce qui est plus couteux. Le même type de raisonnement que ci-dessus montre que pour k grand, v_{k+2} est presque colinéaire à l'espace engendré par v_k et v_{k+1}, la recherche d'une relation $av_{k+2} + bv_{k+1} + v_k = 0$ permet alors de calculer les valeurs propres qui sont les deux racines de $ax^2 + bx + 1 = 0$. La convergence est de type série géométrique, on gagne le même nombre de décimales à chaque itération. Applications :

- la méthode de la puissance peut donner une estimation du nombre de condition L^2 d'une matrice A. On calcule $B = A^*A$ puis on effectue cette méthode sur B pour avoir une estimation de la plus grande valeur propre, puis "sur B^{-1}" par itérations inverses et on fait le rapport des racines carrées. C'est une méthode intéressante si la matrice est creuse et symétrique (pour pouvoir faire Cholesky creux pour les itérations inverses).
- la méthode de la puissance peut donner une estimation rapide de la taille de la plus grande racine d'un polynôme (en module), en itérant sur la matrice companion du polynôme, matrice qui contient beaucoup de 0, donc le produit avec un vecteur se fait en temps $O(n)$, où n est le degré du polynôme.

```
f(P,eps,N):={
  local k,l,n,v,old,new,oldratio,tmp;
  l:=coeffs(P);
  n:=degree(P);
  l:=revlist(l[1..n]/l[0]);
  v:=randvector(n,uniform,-1,1);
  oldratio:=-1;
  for k from 1 to N do
    old:=maxnorm(v);
    tmp := -l[0]*v[n-1];
    for j from 1 to n-1 do
      v[j] =< v[j-1]-l[j]*v[n-1];
    od;
    v[0] =< tmp;
    new:=maxnorm(v);
    if (abs(new/old-oldratio)<eps) return new/old;
    oldratio:=new/old;
  od;
  retourne undef;
}:;
```

Ceci peut par exemple servir à déterminer pour un polynôme P donné squarefree (de degré n et coefficient dominant p_n) l'écart minimal entre 2 racines, on calcule

`R:=normal(resultant(P,subst(P,x=x+y),x)/x^degree(P))`

c'est un polynôme bicarré dont on cherche la plus petite racine en calculant le carré de la plus grande racine en

module de `numer(subst(R,y=1/sqrt(x)))`. On peut obtenir un minorant à priori de cette plus petit racine en calculant

$$\text{resultant}(P, P') = \pm p_n^{2n-1} \prod_{1 \leq i < j \leq n} (r_i - r_j)^2$$

on isole l'écart minimal au carré, on majore les autres carrés en majorant les racines, et on peut minorer le résultant à priori par 1 si P est à coefficients entiers.

7.2 Itérations inverses

La méthode précédente permet de calculer la valeur propre de module maximal d'une matrice. Pour trouver une valeur propre proche d'une quantité donnée x, on peut appliquer la méthode précédente à la matrice $(A - xI)^{-1}$ (en pratique on effectue LU sur $A - xI$ et on résoud $(A - xI)u_{n+1} = u_n$). En effet, les valeurs propres de cette matrice sont les $(x_i - x)^{-1}$ dont la norme est maximale lorsqu'on se rapproche de x_i. Attention à ne pas prendre x trop proche d'une valeur propre, car le calcul de $(A - xI)u_{n+1} = u_n$ est alors peu précis (la matrice étant mal conditionnée).

7.3 Elimination des valeurs propres trouvées

Si la matrice A est symétrique, et si e_n est un vecteur propre normé écrit en colonne, on peut considérer la matrice $B = A - x_n e_n e_n^t$ qui possède les mêmes valeurs propres et mêmes vecteurs propres que A avec même multiplicité, sauf x_n qui est remplacé par 0. En effet les espaces propres de A sont orthogonaux entre eux, donc

$$Be_n = x_n e_n - x_n e_n e_n^t e_n = 0, Be_k = x_k e_k - x_n e_n e_n^t e_k = x_k e_k$$

On peut donc calculer la 2ème valeur propre (en valeur absolue), l'éliminer et ainsi de suite. Si la matrice A n'est pas symétrique, il faut considérer $B = A - x_n e_n f_n^t / e_n.f_n$ où f_n est vecteur propre de A^t associé à x_n. En effet $f_k.e_j = 0$ si $i \neq j$ car $Ae_k.f_j = x_k e_k.f_j = e_k.A^t f_j = x_j e_k f_j$ et donc $f_k.e_k \neq 0$ (sinon e_k est dans l'orthogonal de $\mathbb{R}^n$ =Vect$(f_1, ..., f_n)$.

7.4 Décomposition de Schur

Il s'agit d'une factorisation de matrice sous la forme

$$A = PSP^{-1}$$

où P est unitaire et S diagonale supérieure. Existence (théorique) : on prend une valeur propre et un vecteur propre correspondant, puis on projette sur l'orthogonal de ce vecteur propre et on s'y restreint, on prend à nouveau une valeur propre et un vecteur propre correspondant, etc. On peut approcher cette factorisation par un algorithme itératif qui utilise la factorisation QR d'une matrice quelconque comme produit d'une matrice unitaire par une matrice triangulaire supérieure à coefficients positifs sur la diagonale. On fait l'hypothèse que les valeurs propres de S sur la diagonale sont classées par ordre de module strictement décroissant $|\lambda_1| > |\lambda_2| > ... > |\lambda_n|$ (développement inspiré par Peter J. Olver dans le cas symétrique `http://www.math.umn.edu/~olver/aims_/qr.pdf`). On peut toujours s'y ramener quitte à remplacer A par $A - \alpha I$. Posons $A_1 = A$, et par récurrence $A_n = Q_n R_n$

(avec Q_n unitaire et R triangulaire supérieure à coefficients diagonaux positifs), $A_{n+1} = R_n Q_n$. On a alors

$$\begin{aligned} A^k &= (Q_1R_1)(Q_1R_1)(Q_1R_1)...(Q_1R_1)(Q_1R_1) \\ &= Q_1(R_1Q_1)(R_1Q_1)(R_1...Q_1)(R_1Q_1)R_1 \\ &= Q_1(Q_2R_2)(Q_2R_2)..(Q_2R_2)R_1 \\ &= Q_1Q_2(R_2Q_2)R_2..Q_2R_2R_1 \\ &= Q_1Q_2(Q_3R_3)...Q_3R_3R_2R_1 \\ &= Q_1...Q_kR_k...R_1 \end{aligned}$$

D'autre part $A = PSP^{-1}$ donc $A^k = PS^kP^{-1}$. Soit D la forme diagonale de S et U la matrice de passage $S = UDU^{-1}$, où U est triangulaire supérieure et où on choisit la normalisation des coefficients sur la diagonale de U valant 1. On a donc

$$A^k = PUD^kU^{-1}P^{-1}$$

Ensuite, on suppose qu'on peut factoriser $U^{-1}P^{-1} = L\tilde{U}$ sans permutations, donc qu'on ne rencontre pas de pivot nul, et quitte à multiplier les vecteurs unitaires de P^{-1} par une constante complexe de module 1 on peut supposer que les pivots sont positifs donc que $\tilde{U}$ a des coefficients positifs sur la diagonale, on a alors

$$A^k = PUD^kL\tilde{U} = Q_1...Q_kR_k...R_1$$

puis en multipliant par $\tilde{U}^{-1}|D|^{-k}$

$$PUD^kL|D|^{-k} = Q_1...Q_kR_k...R_1\tilde{U}^{-1}|D|^{-k}$$

où $R_k...R_1\tilde{U}^{-1}|D|^{-k}$ est triangulaire supérieure à coefficients positifs sur la diagonale et $Q_1...Q_k$ est unitaire. On regarde ensuite les entrées de la matrice $D^kL|D|^{-k}$, sous la diagonale elles convergent (géométriquement) vers 0, donc $UD^kL|D|^{-k}$ tend vers une matrice triangulaire supérieure dont les coefficients diagonaux valent $e^{ik\arg(\lambda_j)}$. On montre que cela entraine que $Q_1...Q_k$ est équivalent à $P(D/|D|)^k$

$$Q_1...Q_k \approx P(D/|D|)^k, \quad R_k...R_1\tilde{U}^{-1}|D|^{-k} \approx (D/|D|)^{-k}UD^kL|D|^{-k}$$

Donc, Q_k tend à devenir diagonale, et $R_kQ_k = A_{k+1}$ triangulaire supérieure. De plus

$$A = Q_1A_2Q_1^{-1} = ... = Q_1...Q_kA_{k+1}(Q_1...Q_k)^{-1}$$

la matrice A_{k+1} est donc semblable à A. En pratique, on n'impose pas la positivité des coefficients diagonaux de R dans la factorisation QR, ce qui ne change évidemment pas le fait que Q_k s'approche d'une matrice diagonale et A_k d'une matrice triangulaire supérieure (avec convergence à vitesse géométrique). On commence aussi par mettre la matrice A sous forme de Hessenberg (par conjugaison par des matrices de Householder), c'est-à-dire presque triangulaire supérieure (on autorise des coefficients non nuls dans la partie inférieure seulement sur la sous-diagonale, $a_{ij} = 0$ si $i > j+1$). Cela réduit considérablement le temps de calcul de la décomposition QR, le produit RQ ayant encore cette propriété, une itération se fait en temps $O(n^2)$ au lieu de $O(n^3)$. Le calcul de RQ à partir de A est d'ailleurs fait directement, on parle d'itération QR implicite. On utilise aussi des "shifts" pour accélerer la convergence, c'est-à-dire qu'au lieu de faire QR et RQ sur la matrice A_k on le fait sur $A_k - \alpha_k I$ où λ_k est choisi pour accélerer la convergence vers 0 du coefficient d'indice ligne n colonne $n-1$ (idéalement il faut prendre α_k proche de λ_n la valeur propre de module minimal, afin de minimiser $|\lambda_n - \alpha_k|/|\lambda_{n-1} - \alpha_k|$). En effet,

si $A_k - \lambda_k I = Q_k R_k$ et $A_{k+1} = R_k Q_k + \lambda_k I$ alors :

$$\begin{aligned}
(A-\alpha_1 I)...(A-\alpha_k I) &= Q_1 R_1 (Q_1 R_1 - (\alpha_2-\alpha_1)I)...(Q_1 R_1 - (\alpha_k-\alpha_1)I) \\
&= Q_1 (R_1 Q_1 - (\alpha_2-\alpha_1)I) R_1 (Q_1 R_1 - (\alpha_3-\alpha_1)I)...(Q_1 R_1 - (\alpha_k-\alpha_1)I) \\
&= Q_1 (A_2 - \alpha_1 I - (\alpha_2-\alpha_1)I) R_1 Q_1 (R_1 Q_1 - (\alpha_3-\alpha_1)I) R_1 ... (Q_1 R_1 - (\alpha_k-\alpha_1)I) \\
&= Q_1 (A_2-\alpha_2 I)(A_2-\alpha_3 I)...(A_2-\alpha_k I) R_1 \\
&= ... \\
&= Q_1 ... Q_k R_k ... R_1
\end{aligned}$$

On peut aussi éliminer la dernière ligne et la dernière colonne de la matrice pour accélerer les calculs dès que le coefficient en ligne n colonne $n-1$ est suffisamment petit. On remarque que pour une matrice réelle si on choisit des shifts conjugués, alors $Q_1...Q_k R_k...R_1$ est réel. Or si $QR = \overline{QR}$ et si R est inversible

$$\overline{Q}^{-1} Q = \overline{R} R^{-1}$$

On a donc une matrice symétrique (car $\overline{Q}^{-1} = Q^t$) et triangulaire supérieure. On en déduit que $\overline{Q}^{-1} Q = D$ est diagonale, donc $Q = \overline{Q} D$. On peut donc rendre Q réelle en divisant chaque colonne par un $e^{i\theta}$, et rendre R réelle en conjuguant par la matrice D. Mais ce procédé de retour au réel après élimination de 2 valeurs propres complexes conjuguées d'une matrice réelle se heurte à un problème de conditionnement parce que le choix d'un shift intéressant pour la convergence va rendre la matrice R proche d'une matrice non inversible (les deux derniers coefficients diagonaux de R sont proches de 0). On a alors seulement

$$\overline{Q}^{-1} Q R = \overline{R}$$

Si on décompose $\overline{Q}^{-1} Q$, R, $\overline{R}$ par blocs $n-2, n-2$, $n-2, 2, 2, n-2$ et $2, 2$, on a

$$\begin{aligned}
\begin{pmatrix} QQ_{11} & QQ_{12} \\ QQ_{21} & QQ_{22} \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} &= \begin{pmatrix} QQ_{11}R_{11} & QQ_{21}R_{11} \\ QQ_{11}R_{12} + QQ_{12}R_{22} & QQ_{21}R_{12} + QQ_{22}R_{22} \end{pmatrix} \\
&= \begin{pmatrix} \overline{R}_{11} & \overline{R}_{12} \\ 0 & \overline{R}_{22} \end{pmatrix}
\end{aligned}$$

Donc on a $QQ_{11} = \overline{R}_{11} R_{11}^{-1}$. Comme Q est unitaire, $QQ = \overline{Q}^{-1} Q = Q^t Q$ est symétrique, donc QQ_{11} est diagonale puisque symétrique et triangulaire supérieure. On peut donc ramener Q_{11} et R_{11} en des matrices réelles. L'algorihtme des itérations QR implicites traite de manière efficace le cas des couples de valeurs propres complexes conjuguées ou plus généralement de clusters de valeurs propres, c'est l'algorithme de Francis (aussi appelé *bulge chasing* en anglais, qu'on pourrait traduire par "à la poursuite du bourrelet", cela vient de la forme que prend la matrice après application d'un shift, elle a des entrées non nulles en première colonne plus bas que la sous-diagonale qui forment un bourrelet non nul, l'annulation de ces entrées par des transformations de Householder déplace le bourrelet sur la colonne suivante). Revenons à la localisation des valeurs propres (une autre approche consiste à rechercher un rectangle du plan complexe stable par itérée de la méthode de Newton). On suppose qu'on a maintenant une matrice unitaire P et une matrice triangulaire supérieure S (aux erreurs d'arrondi près) telles que

$$P^{-1} A P = S$$

Que peut-on en déduire ? On va d'abord arrondir P en une matrice exacte à coefficients rationnels, dont les dénominateurs sont une puissance de 2 (en fait c'est exactement ce que donne l'écriture d'un flottant en base 2, une fois

ramené tous les exposants à la même valeur). On a donc une matrice P_e presque unitaire exacte et telle que

$$S_e = P_e^{-1} A P_e$$

est semblable à A, et presque triangulaire supérieure. (comme P_e est presque unitaire, sa norme et la norme de son inverse sont proches de 1 donc S_e est proche de S, les coefficients de S_e sont de la même taille que les coefficients de A : le changement de base est bien conditionné et c'est la raison pour laquelle on a choisi d'effectuer des transformations unitaires). Notons $\mu_1, ..., \mu_n$ les coefficients diagonaux de S_e, soit ε un majorant de la norme des coefficients sous-diagonaux de S_e, et soit δ un minorant de l'écart entre 2 μ_j distincts. On a donc $S_e = U + E$ où U est triangulaire supérieure, E est triangulaire inférieure avec des 0 sous la diagonale et des coefficients de module majorés par ε. Si ε est suffisamment petit devant δ, on va montrer qu'on peut localiser les valeurs propres de S_e (qui sont celles de A) au moyen des μ_j. En effet, fixons j et soit C un cercle de centre $\mu = \mu_j$ et de rayon $\alpha \leq \delta/2$. Si A est une matrice diagonalisable, on sait que

$$\text{nombre de valeurs propres} \in C = \frac{1}{2i\pi} \text{trace} \int_C (A - zI)^{-1}$$

En prenant $A = S_e$, et en écrivant

$$(S_e - zI)^{-1} = (U - zI + E)^{-1} = (I + (U - zI)^{-1}E)^{-1}(U - zI)^{-1}$$

on développe le second terme si la norme de $(U - zI)^{-1}E$ est strictement inférieure à 1

$$(S_e - zI)^{-1} = (U - zI)^{-1} - (U - zI)^{-1}E(U - zI)^{-1} + (U - zI)^{-1}E(U - zI)^{-1}E(U - zI)^{-1} + ...$$

puis on calcule la trace

$$\text{trace}(S_e - zI)^{-1} = \sum_j (\mu_j - z)^{-1} + \eta$$

avec

$$|\eta| \leq 2\pi\alpha ||(U - zI)^{-1}|| \frac{||(U - zI)^{-1}E||}{1 - ||(U - zI)^{-1}E||}$$

Au final, le nombre de valeurs propres dans C est donné par

$$1 + \tilde{\eta}, \quad |\tilde{\eta}| \leq \alpha \text{max}_{z \in C} ||(U - zI)^{-1}|| \frac{||(U - zI)^{-1}E||}{1 - ||(U - zI)^{-1}E||}$$

Il suffit donc que le max soit plus petit que 1 pour avoir l'existence d'une valeur propre et une seule de S_e dans le cercle C (à distance au plus α de μ). Ce sera le cas si

$$\varepsilon \leq \frac{1}{2} \left(\frac{\delta}{2||S_e||} \right)^{n-1} \frac{\alpha}{\sqrt{n-1}}$$

on choisit donc α pour réaliser l'égalité ci-dessus, sous réserve que δ ne soit pas trop petit, rappelons que α doit être plus petit ou égal à $\delta/2$. Si δ est petit, il peut être nécessaire d'utiliser une précision plus grande pour les calculs de la décomposition de Schur en arithmétique flottante. Typiquement, on peut espérer (pour un écart δ pas trop petit) pouvoir localiser les racines d'un polynôme de degré n par cette méthode avec précision b bits en $O(n^3b^2 + n^2b^3)$ opérations pour le calcul de la décomposition de Schur en flottant (n^3b^2 pour Hessenberg initial puis n^2b^2 par itération et un nombre d'itérations proportionnel à b). Pour le calcul exact de S_e, il faut inverser une matrice de

taille n avec des coefficients de taille proportionnelle à b donc $O(n^4 b \ln(n))$ opérations (en modulaire, la taille des coefficients de l'inverse est $O(nb \ln(n))$) puis calculer un produit avec une matrice n, n de coefficients de taille proportionnelle à b, soit $O(n^4 b^2 \ln(nb))$ opérations. Asymptotiquement, on peut faire mieux avec des méthodes de multiplication et d'opérations matricielles par blocs. Pour éviter la perte d'un facteur n, on peut aussi ne pas faire de calculs en mode exact et controler les erreurs sur la matrice S. On peut regrouper les valeurs propres par "clusters" si elles sont trop proches à la précision de b bits. Pour la recherche des racines d'un polynôme P, on peut montrer, en calculant le résultant de P et de P' qui est en module plus grand ou égal à 1, et en l'écrivant comme produit des carrés de différences des racines, et en majorant toutes les différences de racine sauf une à l'aide de la norme infinie de P, qu'il faut au pire $b = O(n)$ bits pour séparer les racines).

8 Equations différentielles (résolution numérique)

8.1 Méthodes à un pas

On considère l'équation différentielle

$$y' = f(t, y), \quad t \in \mathbb{R}, \quad y(t) \in \mathbb{R}^d, y(0) = y_0$$

où $y(t)$ est la fonction inconnue cherchée et où f est une fonction régulière de t et y (par exemple C^1 sur un domaine pour avoir existence et non recoupement des courbes intégrales dans ce domaine). On cherche à approcher numériquement $y(t)$ pour $t > 0$. On présente ici des méthodes de résolution numérique à un pas, dont le principe consiste à discrétiser l'intervalle $[0, t]$ en des subdivisions en temps de petite taille $[0, t_1], [t_1, t_2], ..., [t_{n-1}, t_n = t]$. Si y_i est une valeur approchée de $y(t_i)$ la méthode à un pas se traduit par une relation de récurrence entre y_i et y_{i+1} qui reflète une méthode d'intégration approchée de

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))\, dt$$

Par exemple, la méthode d'Euler explicite utilise la méthode des rectangles à gauche

$$y_{i+1} = y_i + (t_{i+1} - t_i) f(t_i, y_i) = y_i + h f(t_i, y_i)$$

où $h = t_{i+1} - t_i$ (pour une méthode à pas variable, le pas h peut dépendre de i) alors que la méthode d'Euler implicite utilise la méthode des rectangles à droite

$$y_{i+1} = y_i + (t_{i+1} - t_i) f(t_{i+1}, y_{i+1}) = y_i + h f(t_i + h, y_{i+1})$$

cette dernière relation nécéssite de résoudre une équation pour déterminer y_{i+1} d'où son nom de méthode implicite. Plus généralement, la méthode de résolution revient à se donner une fonction $\Phi(t, y, h)$ et à poser :

$$y_{i+1} = y_i + h\Phi(t_i, y_i, h)$$

pour la méthode d'Euler explicite, $\Phi(t, y, h) = f(t, y)$, pour la méthode d'Euler implicite, Φ s'obtient en résolvant une équation (par exemple avec la méthode du point fixe, pour h suffisamment petit). Lorsqu'on compare la solution de l'équation et une valeur approchée obtenue par une méthode à un pas, il faut distinguer

- l'erreur locale (ou erreur de consistance) de la méthode qui est une majoration de $|y_1 - y(t_1)|$ en fonction du pas $h = t_1 - t_0$, on dit qu'une méthode est d'ordre au moins n si $|y_1 - y(t_1)| \leq C_n h^{n+1}$ (cette notion est reliée à l'ordre de la méthode numérique d'intégration approchée utilisée).
- l'erreur globale de la méthode, qui accumule deux phénomènes, l'erreur locale à chaque pas et l'erreur sur la condition initiale pour les subdivisions $[t_i, t_{i+1}], i > 0$, conséquence des erreurs précédentes (en pratique il faudrait aussi ajouter les erreurs d'arrondis et l'erreur éventuelle sur la condition initiale). Pour majorer cette erreur, il est nécessaire de supposer que la fonction f est lipschitzienne par rapport à la variable y, l'erreur globale fera alors intervenir un terme en e^{Ct} multiplié par l'erreur locale (accumulation exponentielle des erreurs au cours du temps).

Plus précisément, on a le résultat suivant :

Théorème 19 *Soit $y(t)$ la solution de $y' = f(t, y), y(t_0) = y_0$ sur $[t_0, T]$. On considére une méthode de résolution à un pas :*

$$y_{i+1} = y_i + h_i \Phi(t_i, y_i, h_i)$$

Si la méthode est d'ordre p, i.e. si pour $h = max(h_i)$ l'erreur locale satisfait

$$|y(t) + h\Phi(t, y(t), h) - y(t + h)| \leq C_p h^{p+1},\ quad \forall t \in [t_0, T], h \leq H$$

et si la fonction Φ est lipschitzienne en y de constante Λ pour $h \leq H$ et y dans un voisinage de la solution $y(t)$, i.e. si

$$|\Phi(t, z, h) - \Phi(t, y, h)| \leq \Lambda |z - y|$$

alors l'erreur globale vérifie

$$|y(t_n) - y_n| \leq h^p \frac{C_p}{\Lambda}(e^{\Lambda(t_n - t_0)} - 1)$$

Par exemple, pour Euler explicite, $\Phi(t, y, h) = f(t, y)$, la constante Λ est la constante de Lipschitz de f, et on prendra pour C_1 un majorant de $\frac{1}{2}|\partial_y f(t, y)|$ dans un voisinage de la solution $y(t)$ pour $t \in [t_0, t_n]$. Pour prouver ce résultat, il faut déterminer comment se propagent les erreurs locales introduites à chaque pas. Par exemple, on a une erreur locale au pas 1 $y(t_1) - y_1$ donc une condition initiale modifiée pour le pas 2 y_1 au lieu de $y(t_1)$. Cette erreur se propage au pas 2 en une erreur

$$|y_1 - y(t_1) + h_1(\Phi(t_1, y_1, h_1) - \Phi(t_1, y(t_1), h_1)| \leq (1 + h_1\Lambda)|y_1 - y(t_1)| \leq e^{h_1\Lambda}|y_1 - y(t_1)|$$

De même aux pas suivants, donc au pas n l'erreur locale au pas 1 s'est propagée en une erreur inférieure ou égale à

$$e^{h_{n-1}\Lambda}...e^{h_2\Lambda}e^{h_1\Lambda}|y_1 - y(t_1)| = e^{\Lambda(t_n - t_0)}|y_1 - y(t_1)| \leq C_p h_0^{p+1} e^{\Lambda(t_n - t_0)}$$

Il faut ensuite sommer les erreurs locales propagées de chaque pas

$$\sum_{i=0}^{n-1} C_p h_i^{p+1} e^{\Lambda(t_n - t_i)} \leq C_p h^p \sum_{i=0}^{n-1} h_i e^{\Lambda(t_n - t_i)}$$

Comme $e^{\Lambda(t_n - t)}$ est positive décroissante sur $[t_0, t_n]$, on peut majorer la somme par l'intégrale

$$C_p h^p \int_{t_0}^{t_n} e^{\Lambda(t_n - t)}\, dt$$

d'où le résultat. (Voir aussi Demailly ou Hairer)

8.2 Méthodes de Runge-Kutta (explicites)

Ce sont des méthodes explicites qui utilisent une méthode de Newton-Cotes pour approcher $\int f(t, y(t))\, dt$ sur $[t_i, t_{i+1}]$. Pour simplifier les notations, notons $t_i = \alpha, t_{i+1} = \beta$, on a alors

$$\int_\alpha^\beta f(t, y(t)) \equiv \sum_{k=0}^{N} \omega_k f(\alpha_k, y(\alpha_k)))$$

Pour estimer la valeur de $f(\alpha_k, y(\alpha_k))$, il est nécessaire d'approcher $y(\alpha_k)$ ce qui se fait par une méthode de Newton-Cotes, en utilisant les estimations des valeurs des $y(\alpha_j), j < k$. On a donc des méthodes de Newton-Cotes avec un sous-ensemble croissant de points d'interpolation, donc pour chaque valeur de k une suite de coefficients

$\omega_{j,k}, j<k$ correspondant à la méthode de Newton-Cotes utilisée. Il faut aussi indiquer la valeur de α_k en donnant un coefficient $c_k \in [0,1]$ tel que

$$\alpha_k = t_i + c_k(t_{i+1} - t_i) = t_i + c_k h$$

En pratique on stocke un tableau dont les lignes donnent c_k et les $c_k\omega_{j,k}, j<k$, et le calcul de $y(\alpha_k)$ se fait ligne par ligne

$$y(\alpha_k) \approx Y_k = y(\alpha_0) + h\sum_{j=0}^{k-1} c_k\omega_{j,k} f(\alpha_j, y(\alpha_j))$$

. Par exemple pour la méthode d'Euler explicite, il y a deux lignes contenant 0 et un seul coefficient :

$$\begin{array}{lll} 0 & : & \\ 1 & : & 1 \end{array}$$

. Pour la méthode du point milieu, il y a trois lignes, la deuxière ligne exprime comment on estime $y(t_i + h/2)$, la troisième $y(t_{i+1}) = y(t_i + h)$:

$$\begin{array}{llll} 0 & : & & \\ \frac{1}{2} & : & \frac{1}{2} & \\ 1 & : & 0 & 1 \end{array}$$

on a donc

$$y(t_i + \frac{1}{2}h) \approx Y_1 = y(t_i) + \frac{1}{2}hf(t_i, y(t_i))$$

$$y(t_i + h) \approx Y_2 = y(t_i) + hf(t_i + \frac{h}{2}, Y_1) = y(t_i) + hf(t_i + \frac{h}{2}, y(t_i) + \frac{h}{2}f(t_i, y(t_i)))$$

La suite des temps α_k est croissante, mais pas forcément de manière stricte, on peut avoir $\alpha_k = \alpha_{k+1}$, la valeur de $y(\alpha_k)$ n'étant pas estimée par la même méthode de Newton-Cotes que $y(\alpha_{k+1})$. La valeur des coefficients est ensuite déterminée pour obtenir un ordre le plus grand possible pour l'erreur locale (ce qui peut nécessiter la résolution de systèmes avec pas mal d'inconnues). Ainsi, la méthode RK4 utilise le tableau suivant

$$\begin{array}{llllll} 0 & : & & & & \\ \frac{1}{2} & : & \frac{1}{2} & & & \\ \frac{1}{2} & : & 0 & \frac{1}{2} & & \\ 1 & : & 0 & 0 & 1 & \\ 1 & : & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

Ce qui se traduit par

$$\begin{aligned} Y_1 &= y(t_0) + \frac{h}{2}f(t_0, y_0) \\ Y_2 &= y(t_0) + \frac{h}{2}f(t_0 + \frac{h}{2}, Y_1) \\ Y_3 &= y(t_0) + hf(t_0 + h, Y_2) \\ Y_4 &= y(t_0) + \frac{h}{6}\left(f(t_0, y(t_0)) + 2f(t_0 + \frac{h}{2}, Y_1) + 2f(t_0 + \frac{h}{2}, Y_2) + f(t_0 + h, Y_3)\right) \end{aligned}$$

Les méthodes de Newton-Cotes utilisées sont les rectangles à gauche puis à droite pour estimer le point milieu, et la méthode de Simpson (en prenant la moyenne des deux estimations pour le point milieu). On peut montrer qu'elle est d'ordre 4 (erreur locale en $O(h^5)$) Les méthodes de résolution numériques implémentées dans Xcas sont des méthodes explicites de Runge-Kutta emboitées avec pas adaptatif, (le pas adaptatif est calculé en estimant l'erreur avec 2 méthodes emboitées RK4 et Prince-Dormand, cf. Hairer).

9 Quelques références

- Les polycopiés du cours d'Ernst Hairer, `http://www.unige.ch/%7Ehairer/polycop.html`
- Analyse numérique et équations différentielles, Demailly J.-P., Presses Universitaires de Grenoble, 1996
- Introduction à l'analyse numérique matricielle et à l'optimisation, Ciarlet P.
- The Art of Computer Programming, Vol. 2 : Seminumerical algorithms, Knuth D., Addison-Wesley, 1998
- Handbook of Mathematical Functions, Abramowitz and Stegun, disponible en ligne sur `http://www.math.sfu.ca/~cbm/aands/toc.htm`
- Arithmétique flottante, Rapport de l'INRIA de V. Lefèvre et P. Zimmermann, téléchargeable sur `http://www.inria.fr/rrrt/rr-5105.html`
- Matrix computations, Golub and Loa, Hopkins University Press, 1989

Index

A Développement de Taylor, séries entières, fonctions usuelles

Résumé : Séries entières. Calcul des fonctions transcendantes usuelles. Soit f une fonction indéfiniment dérivable sur un intervalle I de $\mathbb{R}$ et $x_0 \in I$. On peut alors effectuer le développement de Taylor de f en x_0 à l'ordre n

$$T_n(f)(x) = f(x_0) + (x - x_0)f'(x_0) + ... + (x - x_0)^n \frac{f^{[n]}(x_0)}{n!}$$

et se demander si $T_n(f)$ converge lorsque n tend vers l'infini, si la limite est égale à $f(x)$ et si on peut facilement majorer la différence entre $f(x)$ et $T_n(f)(x)$. Si c'est le cas, on pourra utiliser $T_n(f)(x)$ comme valeur approchée de $f(x)$. On peut parfois répondre à ces questions simultanément en regardant le développement de Taylor de f avec reste : il existe θ compris entre x_0 et x tel que

$$R_n(x) := f(x) - T_n(f)(x) = (x - x_0)^{n+1} \frac{f^{[n+1]}(\theta)}{(n+1)!}$$

C'est le cas pour la fonction exponentielle que nous allons détailler, ainsi que les fonctions sinus et cosinus.

A.1 La fonction exponentielle

Soit $f(x) = \exp(x)$ et $x_0 = 0$, la dérivée n-ième de f est $\exp(x)$, donc $R_n(x) = \exp(\theta)x^{n+1}/(n+1)!$ avec θ compris entre 0 et x, ainsi si x est positif $|R_n(x)| \leq e^x x^{n+1}/(n+1)!$ et si x est négatif, $|R_n(x)| \leq x^{n+1}/(n+1)!$. Dans les deux cas, la limite de R_n est 0 lorsque n tend vers l'infini, car pour $n \geq 2x$, on a

$$\frac{x^{n+1}}{(n+1)!} = \frac{x^n}{n!}\frac{x}{n+1} \leq \frac{1}{2}\frac{x^n}{n!}$$

on a donc pour tout x réel

$$e^x = \lim_{n\to+\infty} T_n(f)(x) = \lim_{n\to+\infty} \sum_{k=0}^{n} \frac{x^k}{k!} = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Comment en déduire une valeur approchée de e^x ? Il suffira d'arrêter la sommation lorsque $R := x^{n+1}/(n+1)!$ si $x < 0$ ou lorsque $R := e^x x^{n+1}/(n+1)!$ si $x > 0$ est inférieur à l'erreur absolue souhaitée, le plus tôt étant le mieux pour des raisons d'efficacité et pour éviter l'accumulation d'erreurs d'arrondi. Si on veut connaitre e^x à une erreur relative ε donnée (par exemple $\varepsilon = 2^{-53}$ pour stocker le résultat dans un double) il suffit que $R/e^x < \varepsilon$, donc si x est positif, il suffit que $x^{n+1}/(n+1)! < \varepsilon$, on peut donc arrêter la sommation lorsque le terme suivant est plus petit que ε. On observe que plus x est grand, plus n devra être grand pour réaliser le test d'arrêt, ce qui est facheux pour le temps de calcul. De plus, le résultat final peut être petit alors que les termes intermédiaires calculés dans la somme peuvent être grands, ce qui provoque une perte de précision relative, par exemple si on veut calculer e^{-10} ou plus généralement l'exponentielle d'un nombre négatif de grande valeur absolue. Exercice : combien de termes faut-il calculer dans le développement de l'exponentielle de -10 pour que le reste soit plus petit que 2^{-53} ? Quel est la valeur du plus grand terme rencontré dans la suite ? Quelle est la perte de précision relative occasionné par cette méthode de calcul ? On peut utiliser les propriétés de la fonction exponentielle pour éviter ce problème. Pour les nombres négatifs, on peut utiliser l'équation $e^{-x} = 1/e^x$ (ne change pas l'erreur relative). Pour les grands réels, on peut utiliser $e^{2x} = (e^x)^2$ (multiplie par 2 l'erreur relative). On peut aussi, si on connait une valeur approchée de $\ln(2)$, effectuer la division euclidienne de x par $\ln(2)$ avec reste symétrique :

$$x = a\ln(2) + r, \quad a \in \mathbb{Z}, |r| \leq \frac{ln(2)}{2}$$

puis si r est positif, on somme la série de $T(f)(r)$, si r est négatif, on calcule $T(f)(-r)$ et on inverse, on applique alors :

$$e^x = 2^a e^r$$

Il faut toutefois noter que $\ln(2)$ n'étant pas connu exactement, on commet une erreur d'arrondi absolu sur r d'ordre $a\eta$, où η est l'erreur relative sur $\ln(2)$, il faut donc ajouter une erreur d'arrondi relative de $x/\ln(2)\eta$ qui peut devenir grande si x est grand. Puis il faut ajouter la somme des erreurs d'arrondi due au calcul de e^r, que l'on peut minimiser en utilisant la méthode de Horner pour évaluer $T_n(f)(r)$ (car elle commence par sommer les termes de plus haut degré qui sont justement les plus petits termes de la somme). Les coprocesseurs arithmétiques qui implémentent la fonction exponentielle ont un format de représentation interne des double avec une mantisse plus grande que celle des double (par exemple 64 bits au lieu de 53), et une table contenant des constantes dont $\ln(2)$ avec cette précision, le calcul de e^x par cette méthode entraine donc seulement une erreur relative d'arrondi au plus proche sur le résultat converti en double (donc de 2^{-53}). Notons que en général x lui-même a déjà été arrondi ou n'est connu qu'avec une précision relative. Or si $x > 0$ est connu avec une erreur relative de ε (donc une erreur absolue de $\varepsilon|x|$, alors

$$e^{x+\varepsilon|x|} = e^x e^{\varepsilon|x|}$$

donc on ne peut pas espérer mieux qu'une erreur relative de $e^{\varepsilon|x|} - 1$ sur l'exponentielle de x. Si εx est petit cette erreur relative (impossible à éviter, quel que soit l'algorithme utilisé pour calculer l'exponentielle) est d'ordre $\varepsilon|x|$. Si εx est grand alors l'erreur relative devient de l'ordre de 1, et la valeur de l'exponentielle calculée peut être très éloignée de la valeur réelle ! Notons que pour les double, il y aura dans ce cas débordement soit vers l'infini soit vers 0 (par exemple si x est supérieur à 709, l'exponentielle renvoie infini). Exercice : refaire les mêmes calculs pour les fonction sinus ou cosinus. On utilise par exemple $\sin(x + \pi) = -sin(x)$, $\sin(-x) = -\sin(x)$, $\sin(x) = \cos(\pi/2 - x)$ pour se ramener au calcul de $\sin(x)$ ou de $\cos(x)$ sur $[0, \pi/4]$.

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!}, \quad \cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!}$$

Cette méthode a toutefois ces limites, car il peut devenir impraticable de calculer la dérivée n-ième d'une fonction (par exemple avec $\tan(x)$), et encore plus de la majorer. D'où l'intérêt de développer une théorie des fonctions qui sont égales à leur développement de Taylor à l'infini d'une part, et d'avoir d'autres méthodes pour majorer le reste, nous présentons ici le cas des séries alternées.

A.2 Séries entières.

Les séries de type prendre la limite lorsque n tend vers l'infini du développement de Taylor en x=0 sont de la forme

$$\sum_{n=0}^{\infty} a_n x^n := \lim_{k \to +\infty} \sum_{n=0}^{k} a_n x^n, a_n = \frac{f^{[n]}(0)}{n!}$$

On peut s'intéresser plus généralement à $\sum_{n=0}^{\infty} a_n x^n$ lorsque a_n est un complexe quelconque, c'est ce qu'on appelle une série entière, on peut aussi les voir comme des polynômes généralisés. S'il existe un point x_0 tel que $|a_n x_0^n|$ est borné (ce sera le cas en particulier si la série converge en x_0), alors

$$|a_n x^n| = |a_n x_0^n||\frac{x}{x_0}|^n \leq M|\frac{x}{x_0}|^n$$

la série converge donc en x si $|x| < |x_0|$ et on peut majorer le reste de la série au rang n par

$$|R_n| \leq M \frac{|\frac{x}{x_0}|^{n+1}}{1 - |\frac{x}{x_0}|}$$

la vitesse de convergence est donc du même type que pour le théorème du point fixe (le nombre de termes à calculer pour trouver une valeur approchée avec k décimales dépend linéairement k, les constantes sont d'autant plus grandes que $|x|$ est grand).

Théorème 20 *S'il existe un rang n_0, un réel $M > 0$ et un complexe x_0 tels que pour $n > n_0$, on ait :*

$$|a_n x_0|^n \leq M$$

alors la série converge pour $|x| < |x_0|$ et pour $n \geq n_0$, on a :

$$|R_n| \leq M \frac{|\frac{x}{x_0}|^{n+1}}{1 - |\frac{x}{x_0}|} \tag{14}$$

On en déduit qu'il existe un réel positif $R \geq 0$ éventuellement égal à $+\infty$ tel que la série converge (la limite de la somme jusqu'à l'infini existe) lorsque $|x| < R$ et n'existe pas lorsque $|x| > R$, ce réel est appelé **rayon de convergence** de la série. Par exemple ce rayon vaut $+\infty$ pour l'exponentielle, le sinus ou le cosinus. Il est égal à 1 pour la série géométrique $\sum x^n$ (car elle diverge si $|x| > 1$ et converge si $|x| < 1$). On ne peut pas dire ce qui se passe génériquement lorsqu'on est à la limite, c'est-à-dire lorsque $|x| = R$ (si $R \neq +\infty$). Mais cela n'a en fait pas trop d'importance en pratique car même si la série converge, elle converge souvent trop lentement pour donner de bonnes approximations. En fait, la vitesse de convergence d'une série entière de rayon $R \neq +\infty$ est en gros la même que celle d'une série géométrique de raison $|x|/R$. Lorsque 2 séries ont un rayon de convergence non nul, alors on peut effectuer leur somme, leur produit comme des polynômes et la série somme/produit a un rayon de convergence au moins égal au plus petit des 2 rayons de convergence des arguments. On peut inverser une série entière non nulle en 0 en appliquant

$$(1 + x)^{-1} = 1 - x + x^2 - x^3 + ...$$

et on obtient une série entière de rayon de convergence non nul. On peut aussi composer deux séries entières g et f en $g \circ f$ (avec les règles de calcul de composition des polynômes) si $f(0) = 0$. On peut enfin dériver et intégrer une série entière terme à terme dans son rayon de convergence. On dit qu'une fonction est développable en série entière en 0 si elle est égale à son développement de Taylor en 0 sommé jusqu'en l'infini dans un disque de centre 0 et de rayon non nul. Les fonctions exponentielle, sinus, cosinus sont donc développables en série entière en 0. La fonction tangente également car le dénominateur cosinus est non nul en 0, mais son rayon de convergence n'est pas l'infini et le calcul des a_n est assez complexe. La fonction $(1 + x)^\alpha$ est développable en séries entières pour tout $\alpha \in \mathbb{R}$ avec un rayon de convergence 1 (ou l'infini pour α entier positif).

$$(1 + x)^\alpha = 1 + \alpha x + \frac{\alpha(\alpha - 1)}{2!} x^2 + ... + \frac{\alpha(\alpha - 1)...(\alpha - n + 1)}{n!} x^n + ...$$

Pour $\alpha = -1$, c'est la série géométrique de raison $-x$, en effet si $|x| < 1$:

$$\sum_{n=0}^{k} (-x)^n = \frac{1 - (-x)^{k+1}}{1 + x} \to_{k \to \infty} \frac{1}{1 + x}$$

En intégrant par rapport à x, on obtient que $\ln(1+x)$ est développable en série entière en 0 de rayon de convergence 1 et

$$\ln(1+x) = \sum_{n=0}^{\infty} \frac{(-x)^{n+1}}{n+1}$$

On peut calculer de manière analogue le développement en série entière de $\arctan(x)$ en iintégrant celui de $1/(1+x^2)$, de même pour $\arccos(x)$ et $\arcsin(x)$ en intégrant celui de $(1-x^2)^{-1/2}$.

$$\arctan(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{2n+1},$$

On peut donc calculer $\ln$, $\arctan$, ... par ces formules, mais il faut répondre à la question où arrête-t-on la somme pour obtenir une précision donnée ? Dans le cas de $\ln(1+x)$, on pourrait répondre comme avec l'exponentielle en majorant la dérivée $n+1$-ième, mais ce n'est plus faisable pour $\arctan, \arcsin, \arccos$. On va donner un autre critère qui ne nécessite pas de calculer cette dérivée mais utilise l'alternance des signes dans la somme.

A.3 Série alternée

Théorème 21 *Soit $S_n = \sum_{k=0}^{n} (-1)^k u_k$ la somme jusqu'au rang n d'une série de réels tels que la suite des u_k décroit à partir d'un rang n_0 et tend vers 0 lorsque $k \to +\infty$. Alors S_n converge vers une limite S. Si $n \geq n_0$, la limite est comprise entre deux sommes partielles succesives S_n et S_{n+1} et le reste est majoré par la valeur absolue du premier terme non sommé :*

$$|R_n| \leq |u_{n+1}|$$

Démonstration :
on montre que les suites $v_n = S_{2n}$ et $w_n = S_{2n+1}$ sont adjacentes. On a

$$v_{n+1} - v_n = S_{2n+2} - S_{2n} = (-1)^{2n+2} u_{2n+2} + (-1)^{2n+1} u_{2n+1} = u_{2n+2} - u_{2n+1} \leq 0$$

donc v_n est décroissante, de même w_n est croissante, et $v_n - w_n = u_{2n+1}$ est positif et tend vers 0. On en déduit que v_n et w_n convergent vers la même limite S telle que $v_n > S > w_n$ et les inégalités du théorème s'en déduisent.
Remarque
lorsqu'on utilise une suite alternée pour trouver une valeur approchée, il faut que u_n tende assez vite vers 0, sinon il y aura perte de précision sur la mantisse lorsqu'on effectuera $u_{2n} - u_{2n+1}$. On sommera aussi les termes par ordre décroissant pour diminuer les erreurs d'arrondi.

A.4 La fonction logarithme

Si nous voulons calculer $\ln(1+x)$ pour $x \in [0,1[$ avec une précision ε, il suffit de calculer

$$\sum_{k=0}^{n} (-1)^k \frac{x^{k+1}}{k+1}$$

pour n tel que la valeur absolue du terme suivant soit plus petit que ε :

$$n \text{ tel que } \frac{x^{n+1}}{n+1} < \varepsilon$$

en effet, les signes sont alternés et la suite $\frac{x^{k+1}}{k+1}$ décroit vers 0. Si la suite décroit lentement vers 0, cette méthode est mauvaise numériquement et en temps de calcul car il y a presque compensation entre termes successifs donc perte de précision sur la mantisse et il y a beaucoup de termes à calculer. C'est le cas pour le logarithme, si x est voisin de 1, il faut calculer n termes pour avoir une précision en $1/n$, par exemple 1 million de termes pour avoir une précision de $1e-6$ (sans tenir compte des erreurs d'arrondi). Si x est proche de $1/2$ il faut de l'ordre de $-\ln(\varepsilon)/\ln(2)$ termes ce qui est mieux, mais encore relativement grand (par exemple 50 termes environ pour une précision en $1e-16$, 13 termes pour $1e-4$). On a donc intérêt à se ramener si possible à calculer la fonction en un x où la convergence est plus rapide (donc $|x|$ le plus petit possible). Par exemple pour le calcul de $\ln(1+x)$ on peut :

- utiliser la racine carrée

  $$ln(1+x) = 2ln(\sqrt{1+x})$$

  on observe que :

  $$X = \sqrt{1+x} - 1 = \frac{x}{1+\sqrt{1+x}} \leq \frac{x}{2}$$

  il faut toutefois faire attention à la perte de précision sur X par rapport à x lorsque x est petit.
- utiliser l'inverse

  $$\ln(1+x) = -\ln(1/(1+x)) = -\ln(1+\frac{-x}{1+x})$$

  lorsque x est proche de 1, $-x/(1+x)$ est proche de $-x/2$, on a presque divisé par 2. Attention toutefois, on se retrouve alors avec une série non alternée, mais on peut utiliser (14) pour majorer le reste dans ce cas.
- trouver une valeur approchée y_0 de $\ln(1+x)$ à une précision faible, par exemple $1e-4$, et utiliser la méthode de Newton pour améliorer la précision. Soit en effet $y = \ln(1+x)$, alors $e^y = 1+x$, on pose $f(y) = e^y - (1+x)$, on utilise la suite itérative

  $$y_{n+1} = y_n - \frac{e^{y_n} - (1+x)}{e^{y_n}}$$

  Comme y_0 est proche à $1e-4$ de y, on peut espérer avoir une valeur approchée de y à $1e-16$ en 2 itérations. Notez que y est proche de 0, on est dans un domaine où le calcul de e^y est rapide et précis et de plus la méthode de Newton "corrige" les erreurs intermédiaires.

Nous sommes donc en mesure de calculer précisément le logarithme $\ln(1+x)$ pour disons $|x| < 1/2$. Pour calculer ln sur $\mathbb{R}^+$, on se ramène à $[1,2]$ en utilisant l'écriture mantisse-exposant, puis si $x \in [3/2, 2]$ on peut en prendre la racine carrée pour se retrouver dans l'intervalle souhaité. On peut aussi effectuer une division par $\sqrt{2}$. Remarquons que si x est connu à une erreur relative ε près, comme

$$\ln(x(1 \pm \varepsilon)) = \ln(x) + \ln(1 \pm \varepsilon)$$

$\ln(x)$ est connu à une erreur absolue de $|\ln(1 \pm \varepsilon)| \approx \varepsilon$. Si $\ln(x)$ est proche de 0, on a une grande perte de précision relative. Finalement, nous savons calculer $\ln$ et $\exp$ sous réserve d'avoir dans une table la valeur de $\ln(2)$. Pour calculer $\ln(2)$ précisément, on peut utiliser

$$\ln(2) = -ln(1/2) = -ln(1-1/2)$$

et le développement en série calculé en mode exact avec des fractions à un ordre suffisant, on majore le reste en utilisant que le terme général de la série $\ln(1+x)$ est borné par $M = 1$ en $x = 1$, donc d'après (14) :

$$|R_n| \leq \frac{1}{2^n}$$

(on peut même obtenir $1/(n2^n)$ car on a besoin de M uniquement pour les termes d'ordre plus grand que n, on peut donc prendre $M = 1/n$). Par exemple, pour avoir $\ln(2)$ avec une mantisse de 80 bits, on effectue une fois pour toutes avec un logiciel de calcul formel :
`a:=sum((1/2)^k/k,k=1..80)|`
puis la division en base 2 avec 81 bits de précision `iquo(numer(a)*2^81,denom(a))` Exercice : pour les fonctions trigonométriques, il faut une méthode de calcul de π. On peut par exemple faire le calcul de $16\arctan(1/5) - 4\arctan(1/239)$ en utilisant le développement de la fonction $\arctan$ à un ordre suffisant.

A.5 Autres applications

On peut calculer certaines intégrales de la même manière, par exemple

$$\int_0^{1/2} \frac{1}{\sqrt{1+x^3}}$$

mais aussi des fonctions définies par des intégrales (cas de nombreuses fonctions spéciales).

A.5.1 Exemple : la fonction d'erreur (error fonction, `erf`)

Cette fonction est définie à une constante multiplicative près par :

$$f(x) = \int_0^x e^{-t^2}\, dt$$

On peut développer en séries entières l'intégrand (rayon de convergence $+\infty$), puis intégrer terme à terme, on obtient

$$f(x) = \sum_{n=0}^{+\infty} (-1)^n \frac{x^{2n}}{n!(2n+1)}$$

Ce développement converge très rapidement pour $|x| \leq 1$. Par contre, pour $|x|$ grand, il faut calculer beaucoup de termes avant que le reste soit suffisamment petit pour être négligeable, et certains termes intermédiaires sont grands, ce qui provoque une perte de précision qui peut rendre le résultat calculé complètement faux. Contrairement à la fonction exponentielle, il n'y a pas de possibilité de réduire l'argument à une plage où la série converge vite. Il faut donc

- soit utiliser des flottants multiprécision, avec une précision augmentée de la quantité nécessaire pour avoir un résultat fiable
- soit, pour les grandes valeurs de x, utiliser un développement asymptotique (en puissances de $1/x$) de

$$\int_x^{+\infty} e^{-t^2}\, dt$$

ainsi que

$$\int_0^{+\infty} e^{-t^2}\, dt = \frac{\sqrt{\pi}}{2}$$

Le développement asymptotique s'obtient par exemple en changeant de variable $u = t^2$ et en effectuant des intégrations par parties répétées en intégrant e^{-u} et en dérivant $u^{-1/2}$ et ses dérivées successives. Ce type de développement asymptotique a la propriété inverse du développement en 0 : les termes successifs

commencent par décroitre avant de croitre et de tendre vers l'infini. Il faut donc arrêter le développement à un rang donné (dépendant de x) et il est impossible d'obtenir une précision meilleure pour cette valeur de x par un développement asymptotique (on parle parfois de développement des astronomes).

Exercice : donner une valeur approchée de $f(1)$ à $1e-16$ près. Combien de termes faut-il calculer dans la somme pour trouver une valeur approchée de $f(7)$ à $1e-16$ près ? Comparer la valeur de $f(7)$ et la valeur absolue du plus grand terme de la série, quelle est la perte de précision relative si on effectue les calculs en virgule flottante ? Combien de chiffres significatifs faut-il utiliser pour assurer une précision finale de 16 chiffres en base 10 ? Calculer le développement asymptotique en l'infini et déterminer un encadrement de $f(7)$ par ce développement. Combien de termes faut-il calculer pour déterminer $f(10)$ à $1e-16$ près par le développement asymptotique et par le développement en séries ? Quelle est la meilleure méthode pour calculer $f(10)$?

A.5.2 Recherche de solutions d'équations différentielles

On peut aussi appliquer les techniques ci-dessus pour calculer des solutions de certaines équations différentielles dont les solutions ne s'expriment pas à l'aide des fonctions usuelles, on remplace dans l'équation la fonction inconnue par son développement en séries et on cherche une relation de récurrence entre a_{n+1} et a_n. Si on arrive à montrer par exemple qu'il y a une solution ayant un développement alternée, ou plus généralement, si on a une majoration $|a_{n+1}/a_n| < C$, alors le reste de la série entière est majoré par $|a_n x^n|/(1-|Cx|)$ lorsque $|x| < 1/C$, on peut alors calculer des valeurs approchées de la fonction solution à la précision souhaitée en utilisant le développement en séries entières.

A.5.3 Exemple : fonctions de Bessel d'ordre entier

Soit m un entier positif fixé, on considère l'équation différentielle

$$x^2y'' + xy' + (x^2 - m^2)y = 0$$

dont on cherche une solution série entière $y = \sum_{k=0}^{\infty} a_k x^k$. En remplacant dans l'équation, si x est dans le rayon de convergence de la série (rayon supposé non nul), on obtient

$$\sum_{k=0}^{\infty} k(k-1)a_k x^k + \sum_{k=0}^{\infty} k a_k x^k + \sum_{k=0}^{\infty} (x^2 - m^2)a_k x^k = 0$$

soit encore

$$\begin{aligned} 0 &= \sum_{k=0}^{\infty} (k^2 - m^2 + x^2)a_k x^k \\ &= -m^2 a_0 + (1-m^2)a_1 x + \sum_{k=2}^{\infty} [(k^2 - m^2)a_k + a_{k-2}]x^k \end{aligned}$$

Par exemple, prenons le cas $m = 0$. On a alors a_0 quelconque, a_1 nul et pour $k \geq 2$

$$a_k = -\frac{a_{k-2}}{k^2}$$

Donc tous les a d'indice impair sont nuls. Les pairs sont non nuls si $a_0 \neq 0$, et ils sont de signe alterné. Soit x fixé, on observe que pour $2k > |x|$,

$$|a_{2k} x^{2k}| < |a_{2k-2} x^{2k-2}|$$

donc la série $\sum_{k=0}^{\infty} a_k x^k$ est alternée à partir du rang partie entière de $|x|$ plus un. Donc elle converge pour tout x (le rayon de convergence de y est $+\infty$) et le reste de la somme jusqu'à l'ordre $2n$ est inférieur en valeur absolue à :

$$|R_{2n}(x)| \leq |a_{2n+2}x^{2n+2}|$$

Par exemple, pour avoir une valeur approchée à $1e-10$ près de $y(x)$ pour $a_0 = 1$ et $|x| \leq 1$, on calcule $y = \sum_{k=0}^{2n} a_k x^k$, on s'arrête au rang n tel que

$$|a_{2n+2}x^{2n+2}| \leq |a_{2n+2}| \leq 10^{-10}$$

On remarque que :

$$a_{2n} = \frac{(-1)^n}{2^2 4^2 ... (2n)^2} = \frac{(-1)^n}{2^{2n} n!^2}$$

donc $n = 7$ convient. Pour $m \neq 0$, on peut faire un raisonnement analogue (les calculs sont un peu plus compliqués). On a ainsi trouvé une solution y_0 de l'équation différentielle de départ dont on peut facilement calculer une valeur approchée (aussi facilement que par exemple la fonction sinus pour $|x| \leq 1$), on peut alors trouver toutes les solutions de l'équation différentielle (en posant $y = y_0 z$ et en cherchant z). **Exercice :** faire de même pour les solutions de $y'' - xy = 0$ (fonctions de Airy).

A.6 Développements asymptotiques et séries divergentes

Un développement asymptotique est une généralisation d'un développement de Taylor, par exemple lorsque le point de développement est en l'infini. De nombreuses fonctions ayant une limite en l'infini admettent un développement asymptotique en l'infini, mais ces développements sont souvent des séries qui semblent commencer par converger mais sont divergentes. Ce type de développement s'avère néanmoins très utile lorsqu'on n'a pas besoin d'une trop grande précision sur la valeur de la fonction. Nous allons illustrer ce type de développement sur un exemple, la fonction exponentielle intégrale, définie à une constante près par

$$f(x) = \int_x^{+\infty} \frac{e^{-t}}{t}\, dt$$

On peut montrer que l'intégrale existe bien, car l'intégrand est positif et inférieur à e^{-t} (qui admet $-e^{-t}$ comme primitive, cette primitive ayant une limite en $+\infty$). Pour trouver le développement asymptotique de f en $+\infty$, on effectue des intégrations par parties répétées, en intégrant l'exponentielle et en dérivant la fraction rationnelle

$$\begin{aligned}
f(x) &= [\frac{-e^{-t}}{t}]_x^{+\infty} - \int_x^{+\infty} \frac{-e^{-t}}{-t^2}\, dt \\
&= \frac{e^{-x}}{x} - \int_x^{+\infty} \frac{e^{-t}}{t^2}\, dt \\
&= \frac{e^{-x}}{x} - ([\frac{-e^{-t}}{t^2}]_x^{+\infty} - \int_x^{+\infty} \frac{-2e^{-t}}{-t^3}) \\
&= \frac{e^{-x}}{x} - \frac{e^{-x}}{x^2} + \int_x^{+\infty} \frac{2e^{-t}}{t^3}\, dt \\
&= ... \\
&= e^{-x}\left(\frac{1}{x} - \frac{1}{x^2} + \frac{2}{x^3} + ... + \frac{(-1)^n n!}{x^{n+1}}\right) - \int_x^{+\infty} \frac{(-1)^n (n+1)! e^{-t}}{t^{n+2}}\, dt \\
&= S(x) + R(x)
\end{aligned}$$

où

$$S(x) = e^{-x}\left(\frac{1}{x} - \frac{1}{x^2} + \frac{2}{x^3} + ... + \frac{(-1)^n n!}{x^{n+1}}\right), \quad R(x) = -\int_x^{+\infty} \frac{(-1)^n (n+1)! e^{-t}}{t^{n+2}}\, dt \tag{15}$$

Le développement en séries est divergent puisque pour $x > 0$ fixé et n tendant vers l'infini

$$\lim_{n\to+\infty} \frac{n!}{x^{n+1}} = +\infty$$

mais si x est grand, au début la série semble converger, de manière très rapide :

$$\frac{1}{x} >> \frac{1}{x^2} >> \frac{2}{x^3}$$

On peut utiliser $S(x)$ comme valeur approchée de $f(x)$ pour x grand si on sait majorer $R(x)$ par un nombre suffisamment petit. On a

$$|R(x)| \leq \int_x^{+\infty} \frac{(n+1)! e^{-t}}{x^{n+2}} = \frac{(n+1)! e^{-x}}{x^{n+2}}$$

On retrouve une majoration du type de celle des séries alternées, l'erreur relative est inférieure à la valeur absolue du dernier terme sommé divisé par e^{-x}/x. Pour x fixé assez grand, il faut donc trouver un rang n, s'il en existe un, tel que $(n+1)!/x^{n+1} < \epsilon$ où ϵ est la précision relative que l'on s'est fixée. Par exemple, si $x \geq 100$, $n = 11$ convient pour $\epsilon = 12!/100^{12} = 5e-16$ (à peu près la précision relative d'un "double"). Ceci permet d'avoir une approximation de la fonction avec une bonne précision et peu de calculs, mais contrairement aux séries entières, il n'est pas possible d'améliorer cette précision de manière arbitraire en poussant le développement plus loin, il y a une précision maximale possible (qui dépend de x). Ce type de développement asymptotique peut être effectué pour d'autres fonctions du même type, par exemple

$$\int_x^{+\infty} e^{-t^2}\, dt, \quad \int_x^{+\infty} \frac{\sin(t)}{t}\, dt, \quad ...$$

Digression : calcul approché de la constante d'Euler γ
On peut montrer que

$$\lim_{n\to+\infty} u_n, \quad u_n = \sum_{k=1}^{n} \frac{1}{k} - \ln(n) \tag{16}$$

existe (par exemple en cherchant un équivalent de $u_{n+1} - u_n$ qui vaut $\frac{-1}{2n^2}$) et on définit γ comme sa limite. Malheureusement, la convergence est très lente et cette définition n'est pas applicable pour obtenir la valeur de γ avec une très grande précision. Il y a un lien entre γ et la fonction exponentielle intégrale, plus précisément lorsque $x \to 0$, $f(x)$ admet une singularité en $-\ln(x)$, plus précisément $f(x) + \ln(x)$ admet un développement en séries (de rayon de convergence $+\infty$), car :

$$\begin{aligned} f(x) + \ln(x) &= \int_x^1 \frac{e^{-t}-1}{t}\, dt + \int_1^{+\infty} \frac{e^{-t}}{t}\, dt \\ &= \int_0^1 \frac{e^{-t}-1}{t}\, dt + \int_1^{+\infty} \frac{e^{-t}}{t}\, dt - \int_0^x \frac{e^{-t}-1}{t}\, dt \end{aligned}$$

Que vaut la constante du membre de droite :

$$C = \int_0^1 (e^{-t}-1)\frac{1}{t}\, dt + \int_1^{+\infty} e^{-t}\frac{1}{t}\, dt$$

Il se trouve que $C = -\gamma$ (voir plus bas une démonstration condensée) et donc :

$$\gamma = \int_0^x \frac{1-e^{-t}}{t}\,dt - f(x) - \ln(x) \tag{17}$$

Pour obtenir une valeur approchée de γ, il suffit donc de prendre un x assez grand pour pouvoir calculer $f(x)$ par son développement asymptotique à la précision requise, puis de calculer l'intégrale du membre de droite par le développement en séries en $x = 0$ (en utilisant une précision intermédiaire plus grande puisque ce développement en séries va sembler diverger au début avant de converger pour n suffisamment grand). Par exemple, on pose $x = 13$, on calcule $f(13)$ par (15) avec $n = 13$ (qui correspond au moment où le terme général de la série est minimum puisque le rapport de deux termes successifs est en n/x) et une erreur absolue inférieure à $e^{-13}13!/13^{14} = 4e-12$

$$f(13) \approx \texttt{exp(-13)*sum((-1)^n*n!/13.^(n+1),n=0..13)}$$

puis on remplace dans (17), avec

$$\int_0^x \frac{1-e^{-t}}{t}\,dt = \sum_{n=0}^{\infty}(-1)^n\frac{x^{n+1}}{(n+1)(n+1)!}$$

dont on obtient une valeur approchée, en faisant la somme jusqu'au rang 49 (pour lequel le terme général est de l'ordre de `1e-12`), le reste de cette somme R_{50} est positif et est inférieur à `(-1)^50*13.^51/51/51!)` qui est de l'ordre de `8e-12`

```
evalf(sum((-1)^n*13^(n+1)/(n+1)/(n+1)!,n=0..49))
```

La somme argument de `evalf` étant exacte, il n'y a pas de problèmes de perte de précision, on peut aussi faire les calculs intermédiaires en arithmétique approchée, on doit alors prendre 4 chiffres significatifs de plus pour tenir compte de la valeur du plus grand terme sommé dans la série, terme que l'on détermine par exemple par

```
seq(13.^(n+1)/(n+1)/(n+1)!,n=0..20)
```

ce terme vaut `13^11/11/11!` soit 4000 environ)

```
Digits:=16; sum((-1)^n*13.^(n+1)/(n+1)/(n+1)!,n=0..49)
```

On obtient finalement comme valeur approchée de γ

```
-exp(-13)*sum((-1)^n*n!/13.^(n+1),n=0..13)-ln(13)+
    sum((-1)^n*13^(n+1)/(n+1)/(n+1)!,n=0..49)
```

soit `0.577215664897` avec une erreur inférieure à `1.2e-11`. Bien entendu, cette méthode est surtout intéressante si on veut calculer un grand nombre de décimales de la constante d'Euler, sinon on peut par exemple appliquer la méthode d'accélération de Richardson à la suite convergente (16) qui définit γ ou d'autres méthodes d'accélération (en transformant par exemple la série en série alternée). On calcule alors de deux manières différentes $f(x)$ pour x plus grand (déterminé par la précision qu'on peut obtenir par le développement aymptotique de f). On peut calculer π de la même manière avec le développement en séries et asymptotique de la fonction sinus intégral (on remplace exponentielle par sinus dans la définition de f) et l'égalité (dont un schéma de preuve est aussi donné plus bas)

$$\int_0^{+\infty} \frac{\sin(t)}{t}\,dt = \frac{\pi}{2} \tag{18}$$

Calcul de C (et preuve de (18)) :
Pour cela on effectue une intégration par parties, cette fois en intégrant $1/t$ et en dérivant l'exponentielle (moins 1 dans la première intégrale).

$$\begin{aligned} C &= \int_0^1 (e^{-t}-1)\frac{1}{t}\,dt + \int_1^{+\infty} e^{-t}\frac{1}{t}\,dt \\ &= [(e^{-t}-1)\ln(t)]_0^1 + \int_0^1 \ln(t)e^{-t}\,dt + [e^{-t}\ln(t)]_1^{+\infty} + \int_1^{+\infty}\ln(t)e^{-t}\,dt \\ &= \int_0^{+\infty}\ln(t)e^{-t}\,dt \end{aligned}$$

Pour calculer cette intégrale, on utilise l'égalité (qui se démontre par récurrence en faisant une intégration par parties) :

$$n! = \int_0^{+\infty} t^n e^{-t}\,dt$$

On va à nouveau intégrer par parties, on intègre un facteur multiplicatif 1 et on dérive l'intégrand, on simplifie, puis on intègre t et on dérive l'autre terme, puis $t^2/2$, etc.

$$\begin{aligned} C &= [te^{-t}\ln(t)]_0^{+\infty} - \int_0^{+\infty} te^{-t}(\frac{1}{t}-\ln(t))\,dt \\ &= 0 - \int_0^{+\infty} e^{-t}\,dt + \int_0^{+\infty} te^{-t}\ln(t)\,dt \\ &= -1 + [\frac{t^2}{2}e^{-t}\ln(t)]_0^{+\infty} - \int_0^{+\infty}\frac{t^2}{2}e^{-t}(\frac{1}{t}-\ln(t))\,dt \\ &= -1 - \int_0^{+\infty}\frac{t}{2}e^{-t} + \int_0^{+\infty}\frac{t^2}{2}e^{-t}\ln(t)\,dt \\ &= -1 - \frac{1}{2} + \int_0^{+\infty}\frac{t^2}{2}e^{-t}\ln(t)\,dt \\ &= \ldots \\ &= -1 - \frac{1}{2} - \ldots - \frac{1}{n} + \int_0^{+\infty}\frac{t^n}{n!}e^{-t}\ln(t)\,dt \\ &= -1 - \frac{1}{2} - \ldots - \frac{1}{n} + \ln(n) + I_n \end{aligned}$$

où

$$I_n = \int_0^{+\infty}\frac{t^n}{n!}e^{-t}(\ln(t)-\ln(n))\,dt$$

Pour déterminer I_n on fait le changement de variables $t = nu$

$$\begin{aligned} I_n &= \int_0^{+\infty}\frac{(nu)^n}{n!}e^{-nu}\ln(u)n\,du \\ &= \frac{n^{n+1}}{n!}\int_0^{+\infty} e^{n(ln(u)-u)}\ln(u)\,du \end{aligned}$$

Or en faisant le même changement de variables $t = nu$:

$$n! = \int_0^{+\infty} t^n e^{-t}\, dt = n^{n+1} \int_0^{+\infty} e^{n(ln(u)-u)}\, du$$

Donc

$$I_n = \frac{\int_0^{+\infty} e^{n(ln(u)-u)} \ln(u)\, du}{\int_0^{+\infty} e^{n(ln(u)-u)}\, du}$$

Lorsque n tend vers l'infini, on peut montrer que $I_n \to 0$, en effet les intégrales sont équivalentes à leur valeur sur un petit intervalle autour de $u = 1$, point où l'argument de l'exponentielle est maximal, et comme l'intégrand du numérateur a une amplitude $\ln(u)$ qui s'annule en $u = 1$, il devient négligeable devant le dénominateur. Finalement on a bien $C = -\gamma$. On peut remarquer qu'en faisant le même calcul que C mais en remplacant e^{-t} par $e^{-\alpha t}$ pour $\Re(\alpha) > 0$, donne $\lim I_n = -\ln(\alpha)$ (car le point critique où la dérivée de la phase s'annule est alors $1/\alpha$). Ceci peut aussi se vérifier pour α réel en faisant le changement de variables $\alpha t = u$

$$\int_0^1 (e^{-\alpha t} - 1)\frac{1}{t}\, dt + \int_1^{+\infty} e^{-\alpha t}\frac{1}{t}\, dt = -\gamma - \ln(\alpha)$$

En faisant tendre α vers $-i$, $-\ln(\alpha)$ tend vers $\ln(i) = i\frac{\pi}{2}$ et on obtient

$$\int_0^1 (e^{it} - 1)\frac{1}{t}\, dt + \int_1^{+\infty} e^{it}\frac{1}{t}\, dt = -\gamma + i\frac{\pi}{2}$$

dont la partie imaginaire nous donne (18), et la partie réelle une autre identité sur γ faisant intervenir la fonction cosinus intégral.

B La moyenne arithmético-géométrique.

B.1 Définition et convergence

Soient a et b deux réels positifs, on définit les 2 suites

$$u_0 = a, v_0 = b, \quad u_{n+1} = \frac{u_n + v_n}{2}, v_{n+1} = \sqrt{u_n v_n} \tag{19}$$

On va montrer que ces 2 suites sont adjacentes et convergent donc vers une limite commune notée $M(a, b)$ et il se trouve que la convergence est très rapide, en raison de l'identité :

$$u_{n+1} - v_{n+1} = \frac{1}{2}(\sqrt{u_n} - \sqrt{v_n})^2 = \frac{1}{2(\sqrt{u_n} + \sqrt{v_n})^2}(u_n - v_n)^2 \tag{20}$$

la convergence est quadratique. On suppose dans la suite que $a \geq b$ sans changer la généralité puisque échanger a et b ne change pas la valeur de u_n et v_n pour $n > 0$. On a alors $u_n \geq v_n$ (d'après (20) pour $n > 0$) et $u_{n+1} \leq u_n$ car

$$u_{n+1} - u_n = \frac{1}{2}(v_n - u_n) \leq 0$$

et $v_{n+1} = \sqrt{u_n v_n} \geq \sqrt{v_n v_n} = v_n$. Donc (u_n) est décroissante minorée (par v_0), (v_n) est croissante majorée (par u_0), ces 2 suites sont convergentes et comme $u_{n+1} = \frac{u_n+v_n}{2}$, elles convergent vers la même limite l qui dépend

de a et b et que l'on note $M(a,b)$. On remarque aussi que $M(a,b) = bM(a/b,1) = aM(1,b/a)$. Précisons maintenant la vitesse de convergence lorsque $a \geq b > 0$. On va commencer par estimer le nombre d'itérations nécessaires pour que u_n et v_n soient du même ordre de grandeur. Pour cela, on utilise la majoration

$$\ln(u_{n+1}) - \ln(v_{n+1}) \leq \ln(u_n) - \ln(v_{n+1}) = \frac{1}{2}(\ln(u_n) - \ln(v_n))$$

donc

$$\ln \frac{u_n}{v_n} = \ln(u_n) - \ln(v_n) \leq \frac{1}{2^n}(\ln(a) - \ln(b)) = \frac{1}{2^n} \ln \frac{a}{b}$$

Donc si $n \geq \frac{\ln(\ln(a/b)/m)}{\ln(2)}$ alors $\ln \frac{u_n}{v_n} \leq m$ (par exemple, on peut prendre $m = 0.1$ pour avoir $u_n/v_n \in [1, e^{0.1}]$). Le nombre minimum d'itérations n_0 est proportionnel au log du log du rapport a/b. Ensuite on est ramené à étudier la convergence de la suite arithmético-géométrique de premiers termes $a = u_{n_0}$ et $b = v_{n_0}$ et même en tenant compte de $M(a,b) = aM(1,b/a)$ à $a = 1$ et $b = v_n/u_n$ donc $0 \leq a - b \leq 1 - e^{-0.1}$. Alors l'équation (20) entraine

$$u_{n+1} - v_{n+1} \leq \frac{1}{8}(u_n - v_n)^2$$

puis (par récurrence)

$$0 \leq u_n - v_n \leq \frac{1}{8^{2^n-1}}(a-b)^{2^n}$$

Donc comme $M(a,b)$ est compris entre v_n et u_n, l'erreur relative sur la limite commune est inférieure à une précision donnée ϵ au bout d'un nombre d'itérations proportionnel au $\ln(\ln(1/\epsilon))$. Typiquement dans la suite, on souhaitera calculer $M(1,b)$ avec b de l'ordre de 2^{-n} en déterminant n chiffres significatifs, il faudra alors $O(\ln(n))$ itérations pour se ramener à $M(1,b)$ avec $b \in [e^{-0.1}, 1]$ puis $O(\ln(n))$ itérations pour avoir la limite avec n chiffres significatifs. **Le cas complexe**

On suppose maintenant que $a, b \in \mathbb{C}$ avec $\Re(a) > 0, \Re(b) > 0$. On va voir que la suite arithmético-géométrique converge encore.

Étude de l'argument

On voit aisément (par récurrence) que $\Re(u_n) > 0$; de plus $\Re(v_n) > 0$ car par définition de la racine carrée $\Re(v_n) \geq 0$ et est de plus non nul car le produit de deux complexes d'arguments dans $]-\pi/2, \pi/2[$ ne peut pas être un réel négatif. On en déduit que $\arg(u_{n+1}) = \arg(u_n + v_n)$ se trouve dans l'intervalle de bornes $\arg(u_n)$ et $\arg(v_n)$ et que $\arg(v_{n+1}) = \frac{1}{2}(\arg(u_n) + \arg(v_n))$ donc

$$|\arg(u_{n+1} - \arg(v_{n+1})| \leq \frac{1}{2}|\arg(u_n) - \arg(v_n)|$$

Après n itérations, on a

$$|\arg(u_n) - \arg(v_n)| \leq \frac{\pi}{2^n}$$

Après quelques itérations, u_n et v_n seront donc presque alignés. Faisons 4 itérations. On peut factoriser par exemple v_n et on est ramené à l'étude de la suite de termes initiaux $a = u_n/v_n$ d'argument $\arg(u_n) - \arg(v_n)$ petit (inférieur en valeur absolue à $\pi/16$) et $b = 1$. On suppose donc dans la suite que

$$|\arg(\frac{u_n}{v_n})| \leq \frac{\pi/16}{2^n}$$

Étude du module
On a :

$$\frac{u_{n+1}}{v_{n+1}} = \frac{1}{2}\left(\sqrt{\frac{u_n}{v_n}} + \frac{1}{\sqrt{\frac{u_n}{v_n}}}\right)$$

Posons $\frac{u_n}{v_n} = \rho_n e^{i\theta_n}$, on a :

$$\begin{aligned}
|\frac{u_{n+1}}{v_{n+1}}| &= \frac{1}{2}\left|\sqrt{\rho_n}e^{i\theta_n/2} + \frac{1}{\sqrt{\rho_n}}e^{-i\theta_n/2}\right| \\
&= \frac{1}{2}\left|(\sqrt{\rho_n} + \frac{1}{\sqrt{\rho_n}})\cos\frac{\theta_n}{2} + i(\sqrt{\rho_n} - \frac{1}{\sqrt{\rho_n}})\sin\frac{\theta_n}{2}\right| \\
&= \frac{1}{2}\sqrt{(\sqrt{\rho_n} + \frac{1}{\sqrt{\rho_n}})^2\cos^2\frac{\theta_n}{2} + (\sqrt{\rho_n} - \frac{1}{\sqrt{\rho_n}})^2\sin^2\frac{\theta_n}{2}} \\
&= \frac{1}{2}\sqrt{\rho_n + \frac{1}{\rho_n} + 2\cos\theta_n}
\end{aligned}$$

Si ρ désigne le max de ρ_n et $1/\rho_n$, on a alors la majoration

$$|\frac{u_{n+1}}{v_{n+1}}| \leq \frac{1}{2}\sqrt{\rho + \rho + 2\rho} = \sqrt{\rho}$$

donc en prenant les logarithmes

$$\ln\rho_{n+1} \leq \frac{1}{2}\ln\rho = \frac{1}{2}|\ln\rho_n| \tag{21}$$

On rappelle qu'on a la majoration

$$|\arg(\frac{u_n}{v_n})| = |\theta_n| \leq \frac{\pi/16}{2^n} \leq \frac{1}{2^{n+1}}$$

qui va nous donner la minoration de ρ_{n+1}

$$\begin{aligned}
\rho_{n+1} = |\frac{u_{n+1}}{v_{n+1}}| &= \frac{1}{2}\sqrt{\rho_n + \frac{1}{\rho_n} + 2 - 2(1-\cos\theta_n)} \\
&= \frac{1}{2}\sqrt{\rho_n + \frac{1}{\rho_n} + 2 - 4\sin^2(\frac{\theta_n}{2})} \\
&\geq \frac{1}{2}\sqrt{\rho_n + \frac{1}{\rho_n} + 2 - \theta_n^2} \\
&\geq \frac{1}{2}\sqrt{\rho_n + \frac{1}{\rho_n} + 2} \times \sqrt{1 - \frac{\theta_n^2}{\rho_n + \frac{1}{\rho_n} + 2}} \\
&\geq \frac{1}{2}\sqrt{\frac{1}{\rho} + \frac{1}{\rho} + 2\frac{1}{\rho}} \times \sqrt{1 - \frac{\theta_n^2}{4}} \\
&\geq \frac{1}{\sqrt{\rho}}\sqrt{1 - \frac{\theta_n^2}{4}} \\
&\geq \frac{1}{\sqrt{\rho}}\sqrt{1 - \frac{1}{4\times 2^{2n+2}}}
\end{aligned}$$

en prenant les log et en minorant $\ln(1-x)$ par $-2x$

$$\ln \rho_{n+1} \geq \frac{1}{2}(-|\ln \rho_n| + \ln(1 - \frac{1}{4 \times 2^{2n+2}})) \geq -\frac{1}{2}(|\ln \rho_n| + \frac{1}{2^{2n+3}})$$

Finalement avec (21)

$$|\ln \rho_{n+1}| \leq \frac{1}{2}(|\ln \rho_n| + \frac{1}{2^{2n+3}})$$

On en déduit

$$|\ln \rho_n| \leq \frac{1}{2^n} \ln \rho_0 + \frac{1}{2^{n+3}} + ... + \frac{1}{2^{2n+1}} + \frac{1}{2^{2n+2}} = \frac{1}{2^n} \ln \rho_0 + \frac{1}{2^{n+2}}$$

La convergence du $\ln(u_n/v_n)$ vers 0 est donc géométrique, donc u_n et v_n convergent quadratiquement.

B.2 Lien avec les intégrales elliptiques

Le calcul de la limite commune des suites u_n et v_n en fonction de a et b n'est pas trivial au premier abord. Il est relié aux intégrales elliptiques, plus précisément on peut construire une intégrale dépendant de deux paramètres a et b et qui est invariante par la transformation $u_n, v_n \to u_{n+1}, v_{n+1}$ (19)

$$I(a,b) = \int_{-\infty}^{+\infty} \frac{dt}{\sqrt{(a^2+t^2)(b^2+t^2)}}$$

On a en effet

$$I(\frac{a+b}{2}, \sqrt{ab}) = \int_{-\infty}^{+\infty} \frac{du}{\sqrt{((\frac{a+b}{2})^2+u^2)(ab+u^2)}}$$

On pose alors

$$u = \frac{1}{2}(t - \frac{ab}{t}), \quad t > 0$$

où $t \to u$ est une bijection croissante de $t \in]0, +\infty[$ vers $u \in]-\infty, +\infty[$, donc

$$\begin{aligned} I(\frac{a+b}{2}, \sqrt{ab}) &= \int_0^{+\infty} \frac{dt/2(1+ab/t^2)}{\sqrt{((\frac{a+b}{2})^2 + 1/4(t-ab/t)^2)(ab+1/4(t-ab/t)^2)}} \\ &= 2\int_0^{+\infty} \frac{dt}{\sqrt{(a^2+t^2)(b^2+t^2)}} = I(a,b) \end{aligned}$$

On note au passage que I est définie si $a, b \in \mathbb{C}$ vérifient $\Re(a) > 0, \Re(b) > 0$, on peut montrer que la relation ci-dessus s'étend (par holomorphie). Lorsque $a = b = l$ (par exemple lorsqu'on est à la limite), le calcul de $I(l,l)$ est explicite

$$I(l,l) = \int_{-\infty}^{+\infty} \frac{dt}{(l^2+t^2)} = \frac{\pi}{l}$$

donc

$$I(a,b) = I(M(a,b), M(a,b)) = \frac{\pi}{M(a,b)}$$

On peut transformer $I(a,b)$ en posant $t = bu$

$$I(a,b) = 2\int_0^{+\infty} \frac{du}{\sqrt{(a^2+b^2u^2)(1+u^2)}} = \frac{2}{a}\int_0^{+\infty} \frac{du}{\sqrt{(1+(b/a)^2u^2)(1+u^2)}}$$

Puis en posant $u = \tan(x)$ ($du = (1+u^2)dx$)

$$I(a,b) = \frac{2}{a}\int_0^{\frac{\pi}{2}} \sqrt{\frac{1+\tan(x)^2}{1+(b/a)^2\tan(x)^2}}\, dx$$

et enfin en posant $\tan^2(x) = \frac{\sin(x)^2}{1-\sin(x)^2}$

$$I(a,b) = \frac{2}{a}\int_0^{\frac{\pi}{2}} \sqrt{\frac{1}{1-(1-\frac{b^2}{a^2})\sin(x)^2}}\, dx$$

Si on définit pour $m < 1$

$$K(m) = \int_0^{\frac{\pi}{2}} \frac{dx}{\sqrt{1-m\sin(x)^2}}$$

alors on peut calculer K en fonction de I, en posant $m = 1 - b^2/a^2$ soit $b^2/a^2 = 1-m$

$$K(m) = \frac{a}{2}I(a, a\sqrt{1-m}) = \frac{a}{2}\frac{\pi}{M(a, a\sqrt{1-m})} = \frac{\pi}{2M(1,\sqrt{1-m})}$$

d'où l'on déduit la valeur de l'intégrale elliptique en fonction de la moyenne arithmético-géométrique :

$$K(m) = \int_0^{\frac{\pi}{2}} \frac{dx}{\sqrt{1-m\sin(x)^2}} = \frac{\pi}{2M(1,\sqrt{1-m})} \tag{22}$$

Dans l'autre sens, pour x et y positifs

$$K((\frac{x-y}{x+y})^2) = \frac{\pi}{2M(1,\sqrt{1-(\frac{x-y}{x+y})^2})} = \frac{\pi}{2M(1,\frac{2}{x+y}\sqrt{xy})} = \frac{\pi}{2\frac{2}{x+y}M(\frac{x+y}{2},\sqrt{xy})} = \frac{\pi}{4}\frac{x+y}{M(x,y)}$$

et finalement

$$M(x,y) = \frac{\pi}{4}\frac{x+y}{K\left((\frac{x-y}{x+y}\right)^2)}$$

B.3 Application : calcul efficace du logarithme.

On peut utiliser la moyenne arithmético-géométrique pour calculer le logarithme efficacement, pour cela on cherche le développement asymptotique de $K(m)$ lorsque m tend vers 1. Plus précisément, on va poser $1-m = k^2$ avec $k \in]0,1]$, donc

$$K(m) = \int_0^{\frac{\pi}{2}} \frac{dx}{\sqrt{1-(1-k^2)\sin(x)^2}} = \int_0^{\frac{\pi}{2}} \frac{dy}{\sqrt{1-(1-k^2)\cos(y)^2}}$$

en posant $y = \pi/2 - x$, et

$$K(m) = \int_0^{\frac{\pi}{2}} \frac{dy}{\sqrt{\sin(y)^2 + k^2\cos(y)^2}}$$

la singularité de l'intégrale pour k proche de 0 apparait lorsque y est proche de 0. Si on effectue un développement de Taylor en $y = 0$, on trouve

$$\sin(y)^2 + k^2\cos(y)^2 = k^2 + (1-k^2)y^2 + O(y^4)$$

Il est donc naturel de comparer $K(m)$ à l'intégrale

$$J = \int_0^{\frac{\pi}{2}} \frac{dy}{\sqrt{k^2 + (1-k^2)y^2}}$$

qui se calcule en faisant par exemple le changement de variables

$$y = \frac{k}{\sqrt{1-k^2}}\sinh(t)$$

ou directement avec Xcas,

```
supposons(k>0 && k<1);
J:=int(1/sqrt(k^2+(1-k^2)*y^2),y,0,pi/2)
```

qui donne après réécriture :

$$J = \frac{1}{\sqrt{1-k^2}}\left(\ln\left(\frac{\pi}{k}\right) + \ln\left(\frac{1}{2}\left(\sqrt{1-k^2+4\frac{k^2}{\pi^2}} + \sqrt{1-k^2}\right)\right)\right) \tag{23}$$

et on peut calculer le développement asymptotique de J en 0

```
series(J,k=0,5,1)
```

qui renvoie :

$$J = \ln\left(\frac{\pi}{k}\right) + O(\left(\frac{-1}{\ln(k)}\right)^5)$$

on peut alors préciser ce développement par

```
series(J+ln(k)-ln(pi),k=0,5,1)
```

qui renvoie (après simplifications et où la notation $\tilde{O}$ peut contenir des logarithmes)

$$\left(\frac{1}{\pi^2} + \frac{\ln(\pi) - \ln(k) - 1}{2}\right)k^2 + \tilde{O}(k^4)$$

donc

$$J = -\ln(k) + \ln(\pi) + \left(\frac{1}{\pi^2} + \frac{\ln(\pi) - \ln(k) - 1}{2}\right)k^2 + \tilde{O}(k^4) \tag{24}$$

Examinons maintenant $K-J$, il n'a plus de singularité en $y=0$, et il admet une limite lorsque $k\to 0$, obtenue en remplacant k par 0

$$(K-J)_{|k=0}=\int_0^{\frac{\pi}{2}}\left(\frac{1}{\sin(y)}-\frac{1}{y}\right)\,dy=\left[\ln\left(\tan\left(\frac{y}{2}\right)\right)-\ln(y)\right]_0^{\frac{\pi}{2}}=\ln(\frac{4}{\pi})$$

D'où pour K

$$K_{k\to 0}=\ln\left(\frac{4}{k}\right)+O(\left(\frac{-1}{\ln(k)}\right)^5)$$

Pour préciser la partie du développement de K en puissances de k, nous allons majorer $K-J-\ln(4/\pi)$, puis $J-\ln(\pi/k)$. Posons

$$A=\sin(y)^2+k^2\cos(y)^2,\quad B=y^2+(1-y^2)k^2$$

Majoration de $K-J-\ln(4/\pi)$
L'intégrand de la différence $K-J-\ln(\frac{4}{\pi})$ est

$$\begin{aligned}\frac{1}{\sqrt{A}}-\frac{1}{\sqrt{B}}-\left(\frac{1}{\sin(y)}-\frac{1}{y}\right) &= \frac{\sqrt{B}-\sqrt{A}}{\sqrt{A}\sqrt{B}}-\frac{y-\sin(y)}{y\sin(y)} &(25)\\ &= \frac{B-A}{\sqrt{A}\sqrt{B}(\sqrt{A}+\sqrt{B})}-\frac{y-\sin(y)}{y\sin(y)} &(26)\\ &= \frac{(y^2-\sin(y)^2)(1-k^2)}{\sqrt{A}\sqrt{B}(\sqrt{A}+\sqrt{B})}-\frac{y-\sin(y)}{y\sin(y)} &(27)\end{aligned}$$

Soit

$$K-J-\ln(\frac{4}{\pi})=\int_0^{\frac{\pi}{2}}\frac{(y-\sin(y))[(1-k^2)y\sin(y)(y+\sin(y))-\sqrt{AB}(\sqrt{A}+\sqrt{B})]}{\sqrt{A}\sqrt{B}(\sqrt{A}+\sqrt{B})y\sin(y)} \quad (28)$$

On décompose l'intégrale en 2 parties $[0,k]$ et $[k,\pi/2]$. Sur $[0,k]$ on utilise (26), on majore chaque terme séparément et on minore A et B par

$$A=k^2+(1-k^2)\sin(y)^2\geq k^2,\quad B=k^2+(1-k^2)y^2\geq k^2$$

Donc

$$\begin{aligned}|\int_0^k| &\leq \int_0^k\frac{|B-A|}{2k^3}\,dy+\int_0^k(\frac{1}{\sin(y)}-\frac{1}{y})\,dy\\ &\leq \int_0^k\frac{y^2-\sin(y)^2}{2k^3}\,dy+\ln(\tan(\frac{k}{2}))-\ln(\frac{k}{2})\\ &\leq \frac{\frac{1}{3}k^3+\frac{-1}{2}k+\frac{1}{4}\sin(2k)}{2k^3}+\ln(\sin(\frac{k}{2}))-\ln(\frac{k}{2})-\ln(\cos(\frac{k}{2}))\\ &\leq \frac{\frac{1}{3}k^3+\frac{-1}{2}k+\frac{1}{4}(2k-\frac{8k^3}{6}+\frac{32k^5}{5!}}{2k^3}-\ln(\cos(\frac{k}{2}))\\ &\leq \frac{k^2}{30}-\ln(1-\frac{1}{2!}\left(\frac{k}{2}\right)^2)\\ &\leq \frac{k^2}{30}+\frac{k^2}{4}\end{aligned}$$

Sur $[k, \pi/2]$, on utilise (28) et on minore A et B par

$$A = \sin(y)^2 + k^2\cos(y)^2 \geq \sin(y)^2, \quad B = y^2 + (1-y^2)k^2 \geq y^2$$

on obtient

$$|\int_k^{\frac{\pi}{2}}| \leq \int_k^{\frac{\pi}{2}} \frac{(y-\sin(y))|C|}{y\sin(y)(y+\sin(y))},$$

où :

$$\begin{aligned} C &= (1-k^2)y\sin(y)(y+\sin(y)) - A\sqrt{B} + B\sqrt{A} \\ &= -A(\sqrt{B}-y) - B(\sqrt{A}-\sin(y)) - Ay - B\sin(y) + (1-k^2)y\sin(y)(y+\sin(y)) \\ &= -A(\sqrt{B}-y) - B(\sqrt{A}-\sin(y)) - k^2(y+\sin(y)) \end{aligned}$$

Donc

$$\begin{aligned} |C| &\leq A(\sqrt{B}-y) + B(\sqrt{A}-\sin(y)) + k^2(y+\sin(y)) \\ &\leq A\frac{B-y^2}{\sqrt{B}+y} + B\frac{A-\sin(y)^2}{\sqrt{A}+\sin(y)} + k^2(y+\sin(y)) \\ &\leq A\frac{k^2}{2y} + B\frac{k^2}{2\sin(y)} + k^2(y+\sin(y)) \end{aligned}$$

et

$$|\int_k^{\frac{\pi}{2}}| \leq \int_k^{\frac{\pi}{2}} \frac{(y-\sin(y))k^2(\frac{A}{2y} + \frac{B}{2\sin(y)} + (y+\sin(y)))}{y\sin(y)(y+\sin(y))}$$

On peut majorer $y - \sin(y) \leq y^3/6$, donc

$$|\int_k^{\frac{\pi}{2}}| \leq \frac{k^2}{6}\int_k^{\frac{\pi}{2}} \frac{Ay}{2\sin(y)(\sin(y)+y)} + \frac{By^2}{\sin(y)^2(\sin(y)+y)} + \frac{y^2}{\sin(y)}$$

On majore enfin A et B par 1,

$$|\int_k^{\frac{\pi}{2}}| \leq \frac{k^2}{6}\int_k^{\frac{\pi}{2}} \frac{y}{2\sin(y)^2} + \frac{y^2}{\sin(y)}$$

Le premier morceau se calcule par intégration par parties

$$\begin{aligned} \frac{k^2}{6}\int_k^{\frac{\pi}{2}} \frac{y}{2\sin(y)^2} &= \frac{k^2}{6}\left([-\frac{y}{\tan(y)}]_k^{\pi/2} + \int_k^{\frac{\pi}{2}} \frac{1}{\tan(y)}\right) \\ &= \frac{k^2}{6}\left(\frac{k}{\tan(k)} + [\ln(\sin(y))]_k^{\frac{\pi}{2}}\right) \\ &= \frac{k^2}{6}\left(\frac{k}{\tan(k)} - \ln(\sin(k))\right) \\ &\leq \frac{k^2}{6}(1-\ln(k)) \end{aligned}$$

Le deuxième morceau se majore en minorant $\sin(y) \geq (2y)/\pi$

$$\frac{k^2}{6}\int_k^{\frac{\pi}{2}} \frac{y^2}{\sin(y)} \leq \frac{k^2}{6}\int_0^{\frac{\pi}{2}} \frac{\pi}{2} y = \frac{k^2\pi^3}{96}$$

Finalement

$$|K - J - \ln(\frac{4}{\pi})| \leq k^2\left(-\frac{1}{6}\ln(k) + \frac{\pi^3}{96} + \frac{1}{6} + \frac{1}{30} + \frac{1}{4}\right)$$

où J est donné en (23). **Majoration de** $J - ln(\pi/k)$
On a

$$|J - \ln\left(\frac{\pi}{k}\right)| = \left|(\frac{1}{\sqrt{1-k^2}} - 1)\ln\left(\frac{\pi}{k}\right) + \frac{1}{\sqrt{1-k^2}}\ln\left(\frac{1}{2}\left(\sqrt{1-k^2+4\frac{k^2}{\pi^2}} + \sqrt{1-k^2}\right)\right)\right|$$

et on va majorer la valeur absolue de chaque terme de la somme. Pour $k \leq 1/2$, on a

$$\frac{1}{\sqrt{1-k^2}} - 1 = \frac{k^2}{\sqrt{1-k^2}+1-k^2} \leq \frac{k^2}{3/4+\sqrt{3}/2}$$

Pour le second terme, on majore le facteur $\frac{1}{\sqrt{1-k^2}}$ par $\frac{2}{\sqrt{3}}$, l'argument du logarithme est inférieur à 1 et supérieur à

$$\frac{1}{2}(1 - \frac{k^2}{2} + 1 - \frac{k^2(1-\frac{4}{\pi^2})}{2}) = 1 - k^2(1 - \frac{1}{\pi^2}) > 1 - k^2$$

donc le logarithme en valeur absolue est inférieur à

$$2k^2$$

donc, pour $k \leq 1/2$,

$$|J - \ln\left(\frac{\pi}{k}\right)| \leq \frac{k^2}{3/4+\sqrt{3}/2}\ln\left(\frac{\pi}{k}\right) + k^2\frac{4}{\sqrt{3}}$$

Finalement, pour $k < 1/2$

$$|K - \ln\left(\frac{4}{k}\right)| \leq k^2\left(\frac{\ln\pi}{3/4+\sqrt{3}/2} + \frac{4}{\sqrt{3}} + \frac{\pi^3}{96} + \frac{9}{20} - (\frac{1}{3/4+\sqrt{3}/2} + \frac{1}{6})\ln(k)\right) \tag{29}$$

que l'on peut réécrire

$$|\frac{\pi}{2M(1,k)} - \ln\left(\frac{4}{k}\right)| \leq k^2(3.8 - 0.8\ln(k)) \tag{30}$$

La formule (30) permet de calculer le logarithme d'un réel positif avec (presque) n bits lorsque $k \leq 2^{-n/2}$ (ce à quoi on peut toujours se ramener en calculant le logarithme d'une puissance 2^m-ième de x ou le logarithme de $2^m x$, en calculant au préalable $\ln(2)$). Par exemple, prenons $k = 2^{-27}$, on trouve (en 8 itérations) $M(1, 2^-27) = M_1 = 0.0781441403763$. On a, avec une erreur inférieure à $19 \times 2^{-54} = 1.1 \times 10^{-15}$

$$M(1, 2^-27) = M_1 = \frac{\pi}{2\ln(2^{29})} = \frac{\pi}{58\ln(2)},$$

On peut donc déduire une valeur approchée de π si on connait la valeur approchée de $\ln(2)$ et réciproquement. Si on veut calculer les deux simultanément, comme les relations entre $\ln$ et π seront des équations homogènes, on est obligé d'introduire une autre relation. Par exemple pour calculer une valeur approchée de π on calcule la différence $\ln(2^{29}+1)-\ln(2^{29})$ dont on connait le développement au premier ordre, et on applique la formule de la moyenne arithmético-géométrique. Il faut faire attention à la perte de précision lorsqu'on fait la différence des deux logarithmes qui sont très proches, ainsi on va perdre une trentaine de bits, il faut grosso modo calculer les moyennes arithmético-géométrique avec 2 fois plus de chiffres significatifs. L'intérêt de cet algorithme apparait lorsqu'on veut calculer le logarithme avec beaucoup de précision, en raison de la convergence quadratique de la moyenne arithmético-géométrique (qui est nettement meilleure que la convergence linéaire pour les développements en série, ou logarithmiquement meilleure pour l'exponentielle), par contre elle n'est pas performante si on ne veut qu'une dizaine de chiffres significatifs. On peut alors calculer les autres fonctions transcendantes usuelles, telle l'exponentielle, à partir du logarithme, ou les fonctions trigonométriques inverses (en utilisant des complexes) et directes. On trouvera dans Brent-Zimmermann quelques considérations permettant d'améliorer les constantes dans les temps de calcul par rapport à cette méthode (cela nécessite d'introduire des fonctions spéciales θ) et d'autres formules pour calculer π.