

Méthodes numériques

Laurent Signac


Table des matières

Avant-Propos	vii
Notations	ix
1 Problèmes de méthodes numériques	1
1 Conditionnement d'une matrice	1
2 Erreurs d'arrondi	2
3 Erreurs de troncature	3
2 Résolution de systèmes linéaires	5
1 Méthodes directes	5
1.1 Résolution de systèmes particuliers	5
1.2 Méthode de Gauss, décomposition de Crout	6
1.3 Méthode de Gauss Jordan	6
1.4 Méthode de Choleski	6
1.5 Factorisation QR	7
1.6 Résolution effective des systèmes	7
2 Méthodes itératives	8
2.1 Méthode de Jacobi	8
2.2 Méthode de Gauss-Seidel	8
2.3 Méthode de relaxation	9
2.4 Preuves de convergence	9
3 Cas des systèmes homogènes	9
3 Calcul matriciel	11
1 Rappels sur les matrices	11
2 Multiplication	12
2.1 Algorithme naïf	12
2.2 Algorithme de Strassen	12
3 Déterminant	12
4 Inversion	13
5 Valeurs propres et vecteurs propres	13
5.1 Obtention des vecteurs propres	13

5.2	Méthode de Ruthishauser	13
5.3	Méthode QR	13
5.4	Méthode des puissances et puissances inverses	14
5.5	Méthode de Jacobi	15
6	Décomposition en valeurs singulières	15
6.1	Méthode de calcul	15
4	Formules de Taylor, expression des dérivées	17
1	Développement de Taylor	17
2	Calcul des dérivées	17
2.1	Forme analytique	18
2.2	Schémas de discrétisation	19
5	Équations différentielles	21
1	Premier ordre et conditions initiales	21
1.1	Méthode d'Euler	21
1.2	Méthode de Taylor	21
1.3	Méthodes de Runge-Kutta	22
1.4	Méthodes d'Adams (pas multiple)	23
2	Systèmes d'équations du premier ordre	25
3	Ordre 2 ou plus	26
4	Conditions aux limites pour le second ordre : méthode de tir	26
4.1	Équation linéaire	27
4.2	Équations non linéaires	27
6	Équations aux dérivées partielles	29
1	Rappel sur les coniques	29
2	Classification des équations d'ordre 2	30
3	Méthode des différences finies	30
3.1	Consistance	31
3.2	Stabilité	31
3.3	Convergence	32
4	Équations paraboliques	32
4.1	Mise sous forme canonique	32
4.2	Résolution par les différences finies	33
5	Équations hyperboliques	34
5.1	Mise sous forme canonique	34
5.2	Différences finies	35
5.3	Décomposition en équations du premier ordre	35
6	Équations Elliptiques	35
6.1	Mise sous forme canonique	35

6.2	Différences finies	36
7	Quelques mots sur la méthode des éléments finis	36
7	Optimisation	39
1	Forme standard	39
2	Exemples de problèmes	39
3	Transformation en problème linéaire	40
4	Méthode de Newton	40
5	Méthodes de descente	41
5.1	Généralités	41
5.2	Cas particulier des systèmes linéaires	41
6	Minimisation d'un critère quadratique	42
6.1	Approximation polynomiale	42
6.2	Résolutions des systèmes linéaires	43
6.3	Méthode de Gauss Newton	44
6.4	Méthode de Levenberg-Marquardt	44
7	Méthode de Lagrange*	44
8	Algorithmes génétiques*	45
9	Quelques mots sur les LMI*	45
8	Programmation linéaire*	47
1	Représentation géométrique	47
1.1	Représentation des contraintes	47
1.2	Recherche graphique d'une solution	47
1.3	Introduction à l'algorithme du simplexe	48
2	Mise sous forme standard	48
2.1	Contraintes de positivité	49
2.2	Contraintes d'inégalité	49
2.3	Exemple de mise sous forme standard	49
3	Algorithme du simplexe	49
3.1	Tableau du simplexe	49
3.2	Itérations de l'algorithme	50
3.3	Extensions à plus de 2 dimensions	51
3.4	Choix du point de départ de l'algorithme	51
A	Givens et Householder	53
1	Matrices de Givens	53
2	Matrices de Householder	54
2.1	Construction d'une matrice de Householder	54
2.2	Annulation d'une partie d'une colonne	54

Avant-Propos

E COURS constitue un approfondissement du cours d'analyse numérique de première année. Les méthodes numériques sont en effet très employées en physique, chimie ou biologie, pour résoudre des problèmes complexes, car issus du monde réel. Si la description d'une méthode particulière est parfois détaillée, l'objectif de ce cours est plutôt d'offrir un panel assez vaste des méthodes généralement employées pour résoudre numériquement des problèmes. Une bibliographie relativement complète est fournie et permettra je l'espère au lecteur de trouver rapidement une information précise et technique.

Les justifications mathématiques de la validité de telle ou telle méthode sont parfois données et parfois omises, selon leur complexité et l'intérêt qu'on trouve effectivement à les vérifier en pratique.


Enfin, si les logiciels de calcul numérique sont très utilisés dans de nombreuses disciplines scientifiques, il est faux de penser qu'ils sont suffisamment perfectionnés pour résoudre tous les types de problèmes *en choisissant une méthode de leur propre chef*. L'intervention humaine est toujours nécessaire dans le choix des méthodes et appréciable en ce qui concerne les vérifications. Le but de ce cours est aussi de permettre à un utilisateur de logiciel de calcul scientifique de sélectionner les méthodes qu'il utilise en connaissance de cause.

Notations

$\ \cdot\ $	norme d'un vecteur ou d'une matrice
$\text{cond}(A)$	conditionnement de la matrice A
A^T	matrice transposée de la matrice A
A^*	matrice transposée conjuguée de la matrice A (matrice adjointe)
v^T	vecteur transposé du vecteur v
$\det(A)$	déterminant de la matrice A
A^{-1}	inverse de la matrice A
$\text{sgn}(x)$	fonction signe, valant 1 si $x > 0$, -1 si $x < 0$ et 0 sinon
$O(f)$	notation de Landau
$\vec{\nabla} f$	gradient d'une fonction vectorielle
$[[m..n]]$	$[m, n] \cap \mathbb{N}$

Chapitre 1

Problèmes de méthodes numériques

 LE PREMIER chapitre constitue une sorte de «recueil» de quelques uns des problèmes spécifiques au caractère *numérique* et *approximatif* des méthodes présentées. En particulier, les problèmes d'arrondis liés aux codage des nombres ne seront plus abordés, mais ils sont bien sûr omniprésents par définition en calcul numérique.

1 Conditionnement d'une matrice

Ce premier problème n'est pas lié à proprement parler à la *résolution* d'un problème sur ordinateur. Néanmoins, il se rencontre dans des problèmes qu'on est le plus souvent amené à traiter numériquement. Nous en parlons donc ici. Considérons l'exemple suivant dû à R.S. Wilson :

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix} \quad b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix}$$

Nous cherchons à résoudre le système : $Ax = b$ dans \mathbb{R}^4 . La matrice A étant inversible, la solution existe. Il s'agit de :

$$x_0 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Notons que x_0 est la solution *exacte* du problème (au sens mathématique du terme).

À présent considérons le vecteur :

$$b' = \begin{pmatrix} 32.1 \\ 22.9 \\ 33.1 \\ 30.9 \end{pmatrix}$$

Ce vecteur est «presque» égal à b , chaque composante ne diffère au plus que de 0.1, c'est à dire d'au plus 0.44% (dans le cas de la deuxième composante). La nouvelle solution du système, qui est toujours la solution *exacte*, est :

$$x'_0 = \begin{pmatrix} 9.2 \\ -12.6 \\ 4.5 \\ -1.1 \end{pmatrix}$$

et n'a plus grand chose à voir avec la solution de départ.

Cet exemple illustre un fait malheureux : les données servant à construire les problèmes que nous serons amenés à résoudre en calcul scientifique sont *le plus souvent issues de mesures*. Une erreur de mesure de 0.44% est généralement considérée comme négligeable. Or, la résolution d'un système linéaire, de manière *exacte* nous montre qu'une telle erreur peut aller jusqu'à modifier l'ordre de grandeur de la solution... qui ne signifie manifestement plus rien.

Si ce problème ne relève pas réellement des méthodes numériques (l'ordinateur n'y est pour rien), on le rencontre souvent, et il est nécessaire de connaître son existence.

Fort heureusement, nous disposons d'une *mesure* de ce problème. Avec les notations qui précèdent, on peut montrer que si la matrice A est inversible, nous avons¹ :

$$\frac{\|x_0 - x'_0\|}{\|x_0\|} \leq \|A\| \|A^{-1}\| \frac{\|b - b'\|}{\|b\|}$$

Le facteur important est ici $\|A\| \|A^{-1}\|$ et s'appelle conditionnement de la matrice A (nous le noterons $\text{cond}(A)$).

On peut de même montrer qu'une légère variation sur les coefficients de la matrice A (plutôt que sur le second membre) peut entraîner de grandes variations sur la solution exacte du système linéaire, et que, à peu de choses près, c'est toujours $\text{cond}(A)$ qui permet d'obtenir une majoration de l'erreur.

Un système linéaire sera donc dit *bien conditionné*, si $\text{cond}(A)$ est petit.

Les quelques propriétés suivantes peuvent s'avérer utiles :

- $\forall A, \text{cond}(A) \geq 1$
- $\forall A$ unitaire, $\text{cond}(A) = 1$
- $\forall A$, si U est une matrice unitaire, $\text{cond}(A) = \text{cond}(U^*AU)$
- $\forall A$ normale, si on note $\lambda_i(A)$ la $i^{\text{ème}}$ valeur propre de A , on a² :

$$\text{cond}(A) = \frac{\max_i |\lambda_i(A)|}{\min_i |\lambda_i(A)|}$$

2 Erreurs d'arrondi

Les méthodes décrites dans ce cours ont pour objet la résolution pratique de problèmes en utilisant des calculateurs numériques. Au moins une spécificité de ces derniers doit être connue : l'inexactitude des calculs en virgule flottante.

Codage des nombres en virgule flottante Un nombre à virgule flottante est codé sur un ordinateur par trois nombres binaires appelés signe, mantisse et exposant, qui ont tous une taille maximum finie, en nombre de chiffres. La norme en vigueur pour le codage en virgule flottante est la norme IEEE754 que nous allons détailler.

Un nombre décimal N à stocker doit d'abord être écrit sous la forme $(-1)^S \times m \times 2^e$ où e est choisi de telle sorte que l'écriture en binaire de m soit $1.xxx$. En d'autres termes, on écrit N en binaire (avec une virgule), puis on place cette virgule juste après le premier 1, et on en déduit l'exposant :

$$\begin{aligned} 11001.111101 &= 1.100111101 \times 2^4 \\ 0.001101 &= 1.101 \times 2^{-3} \end{aligned}$$

Les nombres à coder sont donc : la partie décimale de m (la partie entière est forcément 1), la valeur de l'exposant et le signe (0 pour un nombre positif et 1 pour un nombre négatif).

Selon que les nombres sont en simple ou double précision, c'est la taille du codage de la partie décimale de m (la mantisse), et la taille du codage de l'exposant qui varient. En simple précision, on utilise 8 bits pour coder l'exposant, et 23 bits pour la mantisse. En double précision on utilise 11 bits pour l'exposant et 52 pour la mantisse. De plus, l'exposant n'est pas codé directement. On lui ajoute 127 en simple précision et 1023 en double précision avant de le coder (pour obtenir un nombre positif). Ainsi, en simple précision, l'exposant -127 sera codé par 0, l'exposant 0 par 01111111 et l'exposant 128 par 11111111.

Dans la suite, nous supposons que le codage est en simple précision. Les nombres normalisés sont ceux pour lesquels le codage de l'exposant (E dans la suite³) vérifie $0 < E < 255$. Le nombre représenté vaut alors $N = (-1)^S \times 2^{E-127} \times 1.M$ où M est le codage de la mantisse. Les nombres dénormalisés sont ceux pour lesquels E est nul alors que M non. Cette distinction permet de représenter des nombres plus petits que $2^{-126} \times 1.M$ en spécifiant qu'un nombre dénormalisé vaut : $2^{-126} \times 0.M$. Enfin, si $E = 255$ et $M \neq 0$, le codage ne représente pas un nombre et si $E = 255$ et $M = 0$, la valeur codée est $(-1)^S \infty$.

¹La norme matricielle utilisée ici est la norme spectrale (plus grande valeur singulière).

²Cette dernière propriété n'est pas vraie en général, mais elle l'est si on utilise la norme euclidienne dans le calcul de $\text{cond}(A)$.

³À ne pas confondre avec e . On a en simple précision : $E = e + 127$.

Le plus petit nombre non nul positif vaut :

$$2^{-126} \times 0.\underbrace{0 \dots 0}_{22 \text{ fois}}1 = 2^{-149} \approx 1.4013 \times 10^{-45}$$

Le plus grand nombre positif vaut :

$$2^{127} \times 1.\underbrace{1 \dots 1}_{23 \text{ fois}} = 2^{128} - 2^{104} \approx 3.4 \times 10^{38}$$

Entre deux nombres, il y a un «trou». Par exemple :

$$0 \ 11111110 \ \underbrace{0 \dots 0}_{23 \text{ fois}} = 2^{127}$$

et le nombre suivant est :

$$0 \ 11111110 \ \underbrace{0 \dots 0}_{22 \text{ fois}}1 = 2^{127} + 2^{104}$$

Par conséquent, entre ces deux nombres successifs, il y a un trou de dimension 2^{104} (!) sans aucun nombre.

Accumulation des erreurs d'arrondis Selon les méthodes employées, les erreurs d'arrondi que nous venons d'évoquer peuvent ou non s'accumuler. Il est utile de savoir pour telle ou telle méthode si c'est le cas ou non.

3 Erreurs de troncature

Les erreurs de troncature ne sont pas dues à l'emploi d'un calculateur, mais aux méthodes de calcul, et aux approximations mathématiques qu'elles mettent en jeu.

Toutes les méthodes itératives (c'est à dire celles dont le résultat exact est la limite d'une suite calculée terme à terme) présentent des erreurs de troncature. En effet, la limite de la suite n'est jamais atteinte, et le calcul est arrêté lorsqu'on suppose qu'une précision suffisante est atteinte. L'erreur de troncature est alors la distance entre la limite et le dernier terme calculé, qui est pris pour résultat.

De même, lorsqu'une fonction est remplacée par son développement de Taylor au premier ordre, on commet une erreur de troncature, qui vaut les termes négligés du développement de Taylor.

itérations) et qui contient une addition et une multiplication. Le nombre d'opérations est donc de l'ordre de :

$$\sum_{i=1}^{i=n} 1 + 2(n - i) = n^2$$

La complexité est donc de l'ordre de n^2 .

1.1.2 Système diagonal

Si $\forall i, j \in [1..n]^2, i \neq j \Rightarrow a_{ij} = 0$, le système est diagonal. Puisque de plus $\text{rang}(A) = n$, alors pour tout $i, a_{ii} \neq 0$. La fonction suivante prend un tableau de taille $n \times (n+1)$ en paramètre², et renvoie un tableau de taille n contenant les solutions du système si celui-ci est diagonal.

resol_diag(a[][] : tableau de réels, n : entier) : tableau de réels

```

┌ i : entier
├ x[1..n] : tableau de réels
├ répéter pour i variant de n à 1
│   ┌ x[i] ← a[i][n+1]/a[i][i]
└ renvoyer x[]
```

En ce qui concerne la complexité de cet algorithme, il contient uniquement n multiplications.

1.2 Méthode de Gauss, décomposition de Crout

La décomposition de Crout s'apparente à l'élimination des inconnues telle qu'on la réalise à la main sur de petites systèmes (méthode de Gauss).

Théorème : *Toute matrice carrée régulière d'ordre n peut se décomposer en $A = PLU$ où P est une matrice de permutation, L est une matrice triangulaire inférieure à diagonale unité et U est une matrice triangulaire supérieure. Démonstration dans [11].*

Remarque : *La présence de la matrice de permutation P est indispensable et correspond aux cas où, dans la méthode de Gauss, il a fallu inverser des lignes en raison d'un coefficient diagonal nul. Même dans le cas où cette inversion n'est pas nécessaire à l'exécution de l'algorithme, on a toujours intérêt à utiliser comme pivot (dans la méthode de Gauss) un nombre de grande valeur absolue (afin qu'une division par celui-ci ne donne pas un nombre trop grand). Choisir le plus grand pivot dans la colonne de travail correspond à la méthode du pivot partiel et choisir le plus grand pivot dans tout la sous matrice concernée (inversions de colonnes nécessaires) correspond à la méthode du pivot total.*

La décomposition de Crout nécessite un nombre d'opérations en $O(n^3)$.

▷ *Matlab :* [L U P]=lu(A)

1.3 Méthode de Gauss Jordan

Si la méthode de Gauss consiste à triangulariser la matrice A la méthode Gauss Jordan permet de la diagonaliser. Elle est réalisable à la main et consiste, le pivot courant étant situé sur la ligne i , à supprimer l'inconnue i de toutes les équations (sauf la $i^{\text{ème}}$).

1.4 Méthode de Choleski

La méthode suivante s'applique au cas des matrices A symétriques et définies positives³

²La colonne supplémentaire correspond au second membre.

³Ce cas semble être bien particulier, mais nous le rencontrerons en fait souvent.

Théorème : Si la matrice A d'ordre n est symétrique et définie positive, alors il existe une matrice L triangulaire inférieure telle que $A = LL^T$. Si de plus, on impose que les coefficients diagonaux de L soient positifs, alors L est unique. Démonstration dans [11].

La décomposition de Choleski nécessite un nombre d'opérations en $O(n^3)$ (avec une constante plus petite que la factorisation LU).

▷ Matlab : `L=chol(A)`

1.5 Factorisation QR

La méthode suivante s'applique à des matrices A quelconques. On obtient le produit $A = QR$ où Q est orthogonale, et R est triangulaire supérieure⁴.

Les matrices Q et R peuvent être obtenues en construisant la suite : $A^{[0]} = A$, $\forall k > 0$, $A^{[k]} = \mathcal{H}_k A^{[k-1]}$, où les matrices \mathcal{H}_k sont des matrices de Householder (voir annexe). Nous obtenons après $n - 1$ itérations :

$$A^{[n-1]} = \mathcal{H}_{n-1} A^{[n-2]} = \dots = \mathcal{H}_{n-1} \dots \mathcal{H}_1 A^{[0]}$$

Les matrices de Householder étant orthogonales et symétriques :

$$\mathcal{H}_1 \dots \mathcal{H}_{n-1} A^{[n-1]} = A$$

Les matrices \mathcal{H}_k sont choisies pour que $A^{[n-1]}$ soit diagonale supérieure. Nous obtenons :

$$QR = A$$

La factorisation QR est en $O(n^3)$. De plus, si la matrice de départ A est inversible, alors, en imposant que R ait des coefficients diagonaux strictement positifs, la décomposition est unique.

▷ Matlab : `[Q R]=qr(A)`

1.6 Résolution effective des systèmes

Nous venons de voir comment factoriser les matrices. Cette factorisation est ensuite utilisée pour résoudre des systèmes linéaires. Ainsi, si l'on doit résoudre : $Ax = b$ et que nous avons obtenu la décomposition $PLUx = b$, alors la résolution du système se fait :

- En appliquant la matrice de permutation au second membre $LUx = Pb = b'$ (nécessite environ $\frac{n^2}{2}$ opérations).
- En résolvant le système $Ly = b'$ ce qui ne coûte que n^2 opérations puisque L est triangulaire.
- Puis en résolvant le système $Ux = y$, ce qui ne coûte que n^2 opérations puisque U est triangulaire.

Le même principe s'applique à la factorisation de Choleski.

En ce qui concerne la factorisation $A = QR$, le système devient $QRx = b$. La matrice Q étant orthogonale, $Q^{-1} = Q^T$ donc : $Rx = Q^T b$. Le calcul de $Q^T b$ est très facile (car on n'a plus l'inversion de matrice), et il reste à résoudre $Rx = b'$, ce qui est simple puisque R est triangulaire.

Le second intérêt des factorisations apparaît dans les cas où il faut résoudre *plusieurs* systèmes dans lesquels seul le second membre change. Dans ces cas précis, la factorisation, qui est l'étape la plus coûteuse n'est réalisée qu'une seule fois.

MATLAB utilise les trois décompositions que nous avons vu pour résoudre les systèmes linéaires :

- factorisation de Cholesky dans le cas de matrices carrées symétriques définies positives ;
- factorisation LU dans le cas de matrices carrées quelconques ;
- factorisation QR dans le cas de matrices rectangulaires.

❶ ~~~~~ ❶ ~~~~~ ❶

⁴Si A est carrée. Sinon, les coefficients de R vérifient $i > j \Rightarrow r_{ij} = 0$.

2.3 Méthode de relaxation

En règle générale, cette méthode est utilisée conjointement à la méthode de Gauss-Seidel. La matrice A est alors décomposée en :

$$A = \left(\frac{D}{\lambda} - L \right) - \left(\frac{1-\lambda}{\lambda} D + U \right)$$

On obtient ainsi les itérations :

$$\left(\frac{D}{\lambda} - L \right) x^{[k+1]} = \left(\frac{1-\lambda}{\lambda} D + U \right) x^{[k]} + b$$

En pratique, on retrouve la méthode de Gauss-Seidel en choisissant $\lambda = 1$. On a souvent des convergences plus rapides pour des valeurs de λ différentes de 1, mais il faut néanmoins conserver le facteur de relaxation dans l'intervalle $]0, 2[$ (démonstration dans [9]).

2.4 Preuves de convergence

Les méthodes de Gauss-Seidel ou Jacobi s'écrivent plus généralement :

$$x^{[k+1]} = Bx^{[k]} + C$$

La suite converge alors si et seulement si le rayon spectral de B que nous noterons⁵ $\rho(B)$ est strictement inférieur à 1. En effet :

$$x^{[k+1]} = B^{k+1}x^{[0]} + \sum_{l=0}^k B^l C$$

Si $\rho(B) < 1$, il existe $a \in \mathbb{R}$ tel que $\rho(B) < a < 1$. Or, puisque $\rho(B)$ est le rayon spectral, $\exists K \in \mathbb{N}/\forall k \geq K, \|B^k\|^{1/k} \leq a$. On en déduit :

$$\lim_{k \rightarrow +\infty} B^{k+1}x^{[0]} = 0$$

et :

$$\lim_{k \rightarrow +\infty} \sum_{l=0}^k B^l C = (I - B)^{-1}C$$

Notons aussi que si l'une des deux méthodes converge, l'autre converge aussi (le rayon spectral obtenu pour chaque méthode est lié [9]). C'est la méthode de Gauss Seidel qui converge le plus vite dans la plupart des cas.

3 Cas des systèmes homogènes

Un système homogène est un système de n équations à n inconnues qui s'écrit : $Ax = 0$ (0 désigne ici le vecteur nul). Si le déterminant est non nul, la solution mathématique est triviale ($x = 0$). Dans le cas contraire, le système admet une infinité de solutions. Dans le cas où il y a une infinité de solutions, cela signifie qu'au moins une équation est combinaison linéaire des autres, et qu'il y a au plus $n - 1$ équations indépendantes. Fixer la valeur d'une inconnue arbitrairement et supprimer une des deux équations linéairement dépendantes permettrait de résoudre le système.

En pratique, il est difficile de trouver quelles sont les équations linéairement dépendantes. On procède donc en fixant la valeur d'une inconnue, de façon à obtenir un système de n équations à $n - 1$ inconnues et à second membre non nul. Puis, on résout ce système, faussement surdéterminé par la méthode exposée en section 6.2 du chapitre 7. La solution obtenue n'est alors plus une approximation, mais la solution réelle du système, une inconnue ayant été fixée arbitrairement.

Si on souhaite résoudre le système $Ax = 0$ et que la matrice A résulte d'un calcul, il faut être conscient du fait que même si $\det(A) \neq 0$, il se peut qu'en «réalité», on ait $\det(A) = 0$ ⁶. Il ne faut donc pas donner la solution $x = 0$ mais bien un vecteur x non nul tel que $Ax \approx 0$. On choisit donc une valeur de départ $x^0 \neq 0$, puis on résout comme précédemment.

⁵Rappel : $\rho(B) = \limsup_{k \rightarrow +\infty} \|B^k\|^{1/k} = \sup_{\lambda_i \in \text{spectre}(B)} |\lambda_i|$.

⁶La matrice A étant le résultat d'un calcul, son déterminant, mathématiquement nul, peut seulement être «très petit» du point de vue de la machine

Chapitre 3

Calcul matriciel



DANS CE chapitre, nous donnons quelques méthodes de calcul classiques sur les matrices : multiplication, calcul de déterminant, inversion, calcul de valeurs et vecteurs propres. Le lecteur souhaitant de plus amples informations pourra se reporter à [11] et [10].

1 Rappels sur les matrices

Une matrice carrée A est inversible (ou non singulière), s'il existe une matrice B telle que $AB = BA = I$. Une telle matrice B est unique.

Une matrice carrée réelle A est symétrique si $A^T = A$.

Une matrice carrée réelle A est anti-symétrique si $A^T = -A$.

Une matrice carrée réelle A est orthogonale si $A^T A = AA^T = I$ c'est à dire si elle est inversible et que son inverse est sa transposée.

Une matrice carrée réelle A est normale si elle commute avec sa transposée, c'est à dire si $A^T A = AA^T$.

Une matrice carrée complexe est hermitienne si $A^* = A$.

Une matrice carrée complexe est anti-hermitienne si $A^* = -A$.

Une matrice carrée complexe A est unitaire si $A^* A = AA^* = I$ c'est à dire si elle est inversible et que son inverse est le conjugué de sa transposée.

Une matrice carrée complexe A est normale si elle commute avec le conjugué de sa transposée, c'est à dire si $A^* A = AA^*$.

La norme spectrale d'une matrice est la norme induite par la norme euclidienne sur les vecteurs : $\|A\| = \max_{\|x\|=1} (\|Ax\|)$ (où x désigne un vecteur). Elle vaut :

$$\|A\| = \max_{\lambda \in \text{Spectre}(A^* A)} \sqrt{\lambda}$$

Deux matrices carrées A et B sont semblables s'il existe une matrice P inversible telle que $B = P^{-1}AP$.

Deux matrices semblables ont les mêmes valeurs propres.

Si A et B sont inversibles, alors AB est semblable à BA .

Une matrice et sa transposée ont les mêmes valeurs propres.

Une matrice est singulière si et seulement si elle possède une valeur propre nulle.

Une matrice A est définie positive si $\forall x \neq 0, x^T Ax > 0$.

2 Multiplication

2.1 Algorithme naïf

L'algorithme naïf de multiplication de matrices découle directement du calcul manuel du produit de deux matrices. Si $A = (a_{ij}) \in \mathbb{R}^{p \times q}$ et $B = (b_{ij}) \in \mathbb{R}^{q \times r}$, alors le produit $C = AB$ existe, $C = (c_{ij}) \in \mathbb{R}^{p \times r}$, et $\forall (i, j) \in \llbracket 1..p \rrbracket \times \llbracket 1..r \rrbracket$, $c_{ij} = \sum_{k=1}^{k=q} a_{ik} b_{kj}$. L'algorithme qui calcule C de cette façon nécessite $O(pqr)$ opérations (autrement dit, pour des matrices $n \times n$, il est en $O(n^3)$).

2.2 Algorithme de Strassen

Strassen a découvert en 1969 un algorithme permettant de multiplier des matrices en $O(n^{2.81})$. Le gain semble faible... mais pour des matrices de grande taille, il est appréciable¹. Cet algorithme n'est pas optimal. On en a découvert de meilleurs depuis. Il est cité ici à titre d'exemple et pour illustrer de quelle façon un gain de complexité améliore les performances en calcul numérique.

On va supposer pour simplifier qu'on souhaite multiplier deux matrices A et B de $\mathbb{R}^{2^k \times 2^k}$. On décompose chacune des deux matrices en quatre blocs de taille $2^{k-1} \times 2^{k-1}$:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

On a alors :

$$C = A \times B = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix}$$

C'est à dire :

$$C = \begin{pmatrix} m_1 + m_2 - m_4 + m_6 & m_4 + m_5 \\ m_6 + m_7 & m_2 - m_3 + m_5 - m_7 \end{pmatrix}$$

avec :

$$\begin{cases} m_1 = (A_{12} - A_{22})(B_{21} + B_{22}) \\ m_2 = (A_{11} + A_{22})(B_{11} + B_{22}) \\ m_3 = (A_{11} - A_{21})(B_{11} + B_{12}) \\ m_4 = (A_{11} + A_{12})B_{22} \\ m_5 = (B_{12} - B_{22})A_{11} \\ m_6 = (B_{21} - B_{11})A_{22} \\ m_7 = (A_{21} + A_{22})B_{11} \end{cases}$$

Ce calcul comporte exactement 7 multiplications de matrices de taille 2^{k-1} et 18 additions de matrices de taille 2^{k-1} . Notons $P(2^k)$ le nombre d'opérations élémentaires (i.e. sur des *nombre*s) nécessaires à la multiplication de deux matrices de taille 2^k par l'algorithme de Strassen. On a : $P(2^k) = 7 \times P(2^{k-1}) + 18 \times (2^{k-1})^2$ puisque additionner deux matrices de taille n requiert n^2 additions. En résolvant la récurrence, on en déduit que le nombre d'opérations élémentaires est de l'ordre de 7^k . Par conséquent, $P(2^k) = O(7^k) = O(2^{k \ln_2 7})$ et $P(n) = O(n^{\ln_2 7})$.

3 Déterminant

Le calcul du déterminant est généralement fait *accessoirement* à la résolution d'un système linéaire. Par exemple, une fois effectuée la décomposition $PA = LU$ (cf. section 1.2 du chapitre 2), on a :

$$\det(PA) = \det(L) \det(U) = \prod_{i=1}^{i=n} l_{ii} u_{ii} = \prod_{i=1}^{i=n} u_{ii}$$

or si P résulte de k inversions de lignes, $\det(P) = (-1)^k$, et donc :

$$\det(A) = (-1)^k \prod_{i=1}^{i=n} u_{ii} \text{ avec } k \text{ le nombre d'inversions de lignes}$$

¹Si on multiplie par 100 la dimension des matrices, l'algorithme naïf met 1 000 000 fois plus de temps pour s'exécuter, alors que l'algorithme de Strassen ne mettra «que» 416 870 fois plus de temps.

4 Inversion

De même que pour le calcul du déterminant, on peut utiliser la décomposition LU pour inverser une matrice. Pour calculer la colonne k de la matrice A^{-1} , il suffit de résoudre le système $Ax = e_k$ où e_k est un vecteur dont toutes les composantes valent 0 sauf la composante k qui vaut 1. L'inversion d'une matrice $n \times n$ consiste donc à faire une factorisation LU puis n résolutions de systèmes en utilisant cette décomposition. Rappelons que la décomposition $PA = LU$ est en $O(n^3)$, et que la résolution d'un système linéaire utilisant cette décomposition est en $O(n^2)$. Le calcul de l'inverse est donc en $O(n^3)$.

Une autre solution consiste à utiliser la méthode de Gauss-Jordan pour résoudre un système linéaire (cf. chapitre 2). Puisque cette méthode permet de diagonaliser une matrice, on l'utilise sur la matrice à inverser et chaque opération est effectuée aussi sur la matrice identité. Au moment où la matrice A a été transformée en l'identité, la matrice identité modifiée vaut A^{-1} . Cette méthode, comme celle de Gauss-Jordan, est aussi en $O(n^3)$.

5 Valeurs propres et vecteurs propres

Dans la suite, les méthodes proposées ont pour but de construire une suite de matrices semblables à la matrice de départ, jusqu'à obtenir une matrice dont les valeurs propres sont facilement calculables. On obtiendra ainsi les valeurs propres de la matrice de départ.

▷ *Matlab* : `[Val Vec]=eig(A)`

5.1 Obtention des vecteurs propres

On peut, à partir d'une valeur propre donnée approximativement, construire une suite qui converge vers le vecteur propre associé. Si λ est une valeur propre de A et qu'un vecteur propre associé est e , alors, $Ae = \lambda e$, donc e est solution du système linéaire : $(A - \lambda I)e = 0$.

Pour la valeur propre exacte, on devrait avoir $\det(A - \lambda I) = 0$. Ce n'est pas souvent le cas lorsque la valeur propre a été évaluée numériquement. On se trouve donc dans le cas de la résolution d'un système homogène à déterminant petit, mais non nul.

On trouvera à la section 3 du chapitre 2 le principe de résolution d'un tel système.

5.2 Méthode de Ruthishauser

Cette méthode est exposée ici en raison de sa simplicité. Néanmoins, elle est peu utilisée en raison de son instabilité.

On décompose la matrice de départ $A_0 = A$ en L_0U_0 (cf. section 1.2 du chapitre 2). Si L et U sont inversibles, c'est à dire si U_0 n'a pas de 0 sur sa diagonale, alors la matrice U_0L_0 a les mêmes valeurs propres que A_0 .

Posons $A_1 = U_0L_0$. On cherche à présent les valeurs propres de A_1 , en réalisant sa décomposition LU : $A_1 = L_1U_1$. Puis on pose $A_2 = U_1L_1$ et ainsi de suite...

Si la méthode converge, A_n tend vers une matrice triangulaire dont les éléments diagonaux sont ses valeurs propres, donc celle de A . On n'est sûr de la convergence que si A est symétrique et définie positive.

5.3 Méthode QR

Nous avons vu au chapitre 2 section 1.5 qu'il était possible de factoriser une matrice quelconque en produit d'une matrice unitaire Q et d'une matrice triangulaire supérieure R (factorisation QR).

Considérons la suite $A_0 = A$, $A_k = Q_kR_k$, $A_{k+1} = R_kQ_k = Q_{k+1}R_{k+1}$. Les matrices A_k et A_{k+1} sont semblables, puisque $Q_k^T A_k Q_k = Q_k^T Q_k R_k Q_k = R_k Q_k = A_{k+1}$ (et Q_k est orthogonale). Donc, pour tout k , A_k et A ont les mêmes valeurs propres. Si la suite A_k converge vers une matrice diagonale, alors les valeurs propres de A seront les éléments diagonaux de A_∞ , limite de la suite A_k .

Dans le cas général, la convergence n'est pas toujours obtenue. Dans le cas particulier des matrices Hessenberg

irréductibles², la suite converge effectivement vers une matrice diagonale. De plus, si A vérifie cette propriété, alors pour tout k , $Q_k R_k$ et $R_k Q_k$ sont aussi des matrices Hessenberg supérieures. Le procédé est donc stable.

Fort heureusement, il existe un procédé permettant de transformer une matrice A carrée en matrice Hessenberg supérieure *qui lui est semblable*. En pratique la méthode QR est donc appliquée uniquement sur des matrices Hessenberg supérieures.

La mise sous forme Hessenberg d'une matrice est très similaire à la factorisation QR . On utilise de la même façon les matrices de Householder, mais sans annuler le coefficient sous la diagonale. De plus, pour obtenir des matrices semblables, à chaque itération, la matrice est multipliée à gauche *et à droite* par une matrice de Householder (rappelons que les matrices de Householder sont unitaires).

▷ *Matlab* : $H = \text{hess}(A)$

5.4 Méthode des puissances et puissances inverses

5.4.1 Calcul de la plus grande valeur propre

Cette méthode est basée sur le fait que si A a pour valeur propre de plus grand module λ_1 , alors, si $v = \sum_{i=1}^{i=n} v_i e_i$ avec $v_1 \neq 0$ et $(e_i)_{i \in [1..n]}$ la base (on suppose que la matrice A a n valeurs propres distinctes) formée des vecteurs propres de A :

$$\lim_{p \rightarrow +\infty} A^p v = \lambda_1^p v_1 e_1$$

En pratique, on ne peut pas calculer la suite $A^p v$, car le module de $\lambda_1^p v_1 e_1$ peut être très grand si $\lambda_1 > 1$. On procède donc ainsi :

$v \leftarrow (1, \dots, 1)$ répéter $v' \leftarrow v$ $v \leftarrow Av$ $m \leftarrow$ composante de v de plus grand module $v \leftarrow v/m$ jusqu'à ce que $\ v' - v\ < \epsilon$	$v \leftarrow (1, \dots, 1)$ répéter $v' \leftarrow v$ $v \leftarrow Av$ $m \leftarrow$ composante de v de plus grand module $v \leftarrow v/m$ jusqu'à ce que $\ v' - v\ < \epsilon$
--	--

En divisant ainsi v par sa composante de plus grand module, on limite sa norme. De plus, puisqu'après la division, il a une composante égale à 1, m tend vers la valeur propre λ_1 .

Cette méthode est simple mais ne converge pas très vite.

5.4.2 Calcul de la plus petite valeur propre

Si la matrice A est inversible, la plus petite valeur propre de A est aussi la plus grande valeur propre de A^{-1} . On peut donc appliquer dans ce cas la méthode précédente sur A^{-1} pour calculer la plus petite valeur propre de A .

5.4.3 Calcul des autres valeurs propres : déflation de la matrice

Cette méthode consiste à construire une autre matrice, qui a les mêmes valeurs propres que A , sauf celle de plus grand module, qui est ramenée à 0. En transformant ainsi la matrice A en une matrice A_1 , on peut à nouveau appliquer la méthode des puissance itérées pour trouver la deuxième valeur propre de plus grand module.

Construisons la matrice A_1 ainsi :

$$A_1 = A - \frac{\lambda_1 e_1 e_1^T}{e_1^T e_1}$$

²Une matrice $H = h_{i,j}$ est dite de Hessenberg si pour tout (i, j) , $i > j + 1 \Rightarrow h_{i,j} = 0$. Elle est dite plus irréductible si pour tout i , $h_{i+1,i} \neq 0$

On a alors :

$$\begin{aligned} A_1 e_j &= A e_j - \frac{\lambda_1 e_1 e_1^T e_j}{e_1^T e_1} \\ &= \lambda_j e_j - \frac{\lambda_1 e_1}{e_1^T e_1} e_1^T e_j \\ &\begin{cases} = \lambda_j e_j & \text{si } j \neq 1 \\ = 0 & \text{sinon} \end{cases} \end{aligned}$$

Le vecteur propre e_1 peut être calculé comme indiqué en début de section, connaissant la matrice A ainsi que la valeur propre λ_1 .

Cette méthode est assez peu précise. En effet, si une erreur est commise sur la valeur propre λ_1 , alors, la matrice A_1 est d'autant moins exacte. La seconde valeur propre sera donc recherchée dans une matrice qui n'est pas la matrice exacte, et de plus cette valeur propre sera obtenue de façon approchée. On ne peut donc généralement pas calculer plus de quelques valeurs propres par ce système.

5.5 Méthode de Jacobi

La méthode de Jacobi consiste aussi en la construction d'une suite de matrices semblables qui tend vers une matrice diagonale. Les valeurs propres seront alors données par les coefficients diagonaux de cette matrice. Cette méthode s'applique au cas des matrices symétriques réelles uniquement. Elle est basée sur l'utilisation de matrices de Givens (annexe) pour annuler certains coefficients.

Supposons que la matrice de départ est d'ordre n et considérons la matrice de Givens G_θ^{pq} . Cette dernière est inversible, et en conséquence, $(G_\theta^{pq})^{-1} A G_\theta^{pq}$ est semblable à A et a donc les mêmes valeurs propres. De plus, on a $(G_\theta^{pq})^{-1} = (G_\theta^{pq})^T$ d'où l'on tire que la matrice $(G_\theta^{pq})^{-1} A G_\theta^{pq}$ est symétrique si A l'était.

Le principe de la méthode de Jacobi est de choisir dans la partie inférieure de la matrice le coefficient qui a la plus grande valeur absolue, et à déterminer l'angle θ qui l'annulera. De cette façon, la méthode converge (preuve dans [10]).

On trouvera en annexe comment calculer les coefficients de $C = (G_\theta^{pq})^T A G_\theta^{pq}$.

La matrice C est ensuite considérée comme la matrice d'entrée et on réitère le processus en annulant le nouveau plus grand coefficient de la partie inférieure. Notons au passage qu'un coefficient nul peut redevenir non nul au cours du calcul, mais il restera «petit». Si a_{pj} est nul, on aura $c_{pj} = -a_{qj} \sin \theta$.

L'algorithme est stoppé lorsque les coefficients non diagonaux sont suffisamment petits.

6 Décomposition en valeurs singulières

▷ *Matlab* : Fonction `svd()`

Toute matrice M de taille (m, n) peut être décomposé en le produit :

$$M = U \Sigma V^* \text{ où}$$

- U est une matrice unitaire (m, m)
- Σ est une matrice diagonale à coefficients réels positifs ou nuls (on impose souvent qu'ils soient classés par ordre décroissant, auquel cas Σ est unique)
- V est une matrice unitaire (n, n)

La relation précédente peut être réécrite $MV = U\Sigma$ puisque V est unitaire. Une valeur singulière σ , élément de Σ vérifie $Mv = \sigma u$ et $M^*u = \sigma v$ où u et v , appelés vecteur singulier à gauche et vecteur singulier à droite sont respectivement des vecteurs colonnes des matrices U et V .

Les valeurs singulières de M sont aussi les racines carrées positives ou nulles des valeurs propres de M^*M

6.1 Méthode de calcul

Il existe plusieurs méthodes de calcul de la décomposition en valeurs singulières. Parmi elles, une est basée sur la décomposition QR :

- calculer $C = A^T A$
- itérer l'algorithme QR (cf. 5.3) pour obtenir : $V_1^T C V_1 = \text{diag}(\sigma_i^2)$
- chercher la décomposition QR de AV_1 (avec pivotage des colonnes) : $(AV_1)\Pi = UR$ où U est orthogonale (c'est le Q habituel), R est triangulaire, avec des valeurs diagonales décroissantes (d'où l'apparition de la matrice de permutation Π)

Nous obtenons ainsi :

$$A = U\Sigma V^T$$

avec :


- $V = V_1\Pi$
- $\Sigma = \text{diag}(\sigma_i)$ (car si $U^T AV = \text{diag}(\sigma_i)$, $V^T(A^T A)V = \text{diag}(\sigma_i^2)$)

Il peut être nécessaire d'ajuster le signe des coefficients de R car les valeurs singulières sont positives. Inverser le signe du $k^{\text{ème}}$ élément diagonal de R revient à modifier le signe de la $k^{\text{ème}}$ colonne de U ou de la $k^{\text{ème}}$ colonne de V .

Notons qu'il existe d'autres méthodes, notamment la méthode de Golub et Kahan [1], basée sur les matrices de Householder (pour bidiagonaliser A) et sur la décomposition QR.

Chapitre 4

Formules de Taylor, expression des dérivées

E COURT chapitre ne présente pas réellement de méthode numérique, mais plutôt des outils qui seront utilisés dans les chapitre sur la résolution d'équations différentielles ou d'équations aux dérivées partielles.

1 Développement de Taylor

Le développement de Taylor à l'ordre n d'une fonction f au point x_0 est :

$$f(x_0 + \Delta x) = f(x_0) + \sum_{i=1}^{i=n} \frac{\Delta x^i}{i!} f^{(i)}(x_0) + O(\Delta x^{n+1}) \quad (4.1)$$

où $f^{(i)}$ désigne la dérivée $i^{\text{ème}}$ de f , et f est supposée suffisamment dérivable. Dans cette égalité et dans toute la suite du chapitre, Δx^i signifiera : $(\Delta x)^i$.

Pour une fonction à deux variables, le développement de Taylor est (sous forme symbolique) :

$$f(u_0 + \Delta u, v_0 + \Delta v) = f(u_0, v_0) + \sum_{i=1}^{i=n} \frac{1}{i!} \left(\frac{\partial f}{\partial u} \Delta u + \frac{\partial f}{\partial v} \Delta v \right)^i + O((\Delta u + \Delta v)^{n+1}) \quad (4.2)$$

Le développement à un certain ordre est obtenu à partir de cette formule «symbolique» en écrivant que :

$\left(\frac{\partial f}{\partial u}\right)^n \left(\frac{\partial f}{\partial v}\right)^m = \frac{\partial^{n+m} f}{\partial u^n \partial v^m}$. On obtient ainsi à l'ordre 1 (c'est à dire en négligeant les termes d'ordre 2 ou plus) :

$$f(u_0 + \Delta u, v_0 + \Delta v) = f(u_0, v_0) + \Delta u \frac{\partial f}{\partial u}(u_0, v_0) + \Delta v \frac{\partial f}{\partial v}(u_0, v_0)$$

et à l'ordre 2 :

$$\begin{aligned} f(u_0 + \Delta u, v_0 + \Delta v) &= f(u_0, v_0) + \Delta u \frac{\partial f}{\partial u}(u_0, v_0) + \Delta v \frac{\partial f}{\partial v}(u_0, v_0) \\ &+ \frac{\Delta u^2}{2} \frac{\partial^2 f}{\partial u^2}(u_0, v_0) + \frac{\Delta v^2}{2} \frac{\partial^2 f}{\partial v^2}(u_0, v_0) \\ &+ \Delta u \Delta v \frac{\partial^2 f}{\partial u \partial v}(u_0, v_0) \end{aligned}$$

2 Calcul des dérivées

Le développement de Taylor permet d'évaluer de façon approchée les dérivées et dérivées partielles à un ordre quelconque.

2.1 Forme analytique

Écrivons le développement de Taylor à l'ordre n en remplaçant Δx par $-k\Delta x$ avec k entier positif :

$$f(x_0 - k\Delta x) = f(x_0) + \sum_{i=1}^{i=j} (-1)^i \frac{(k\Delta x)^i}{i!} f^{(i)}(x_0) + O(\Delta x^{j+1})$$

Nous allons voir comment évaluer les dérivées $j^{\text{èmes}}$ de f avec une erreur de troncature en $O(\Delta x)$ (pour une erreur plus faible, se rapporter à [14] ou [8]).

Étant donnée une « précision » (un ordre), nous pouvons calculer la valeur de $f^{(j)}(x_0)$ comme combinaison linéaire des $f(x_0 - m\Delta x)$ avec $m \in \llbracket 0..j \rrbracket$. Autrement dit, nous pouvons évaluer la dérivée $j^{\text{ème}}$ de la fonction f au point x_0 en connaissant les valeurs de f aux points $x_0, x_0 - \Delta x, \dots, x_0 - j\Delta x$.

Ainsi, pour évaluer la dérivée d'ordre j avec une précision à l'ordre 1 (égalité en $O(\Delta x)$), nous écrivons l'équation précédente pour k variant de 1 à j (on obtient j équations). Puis par combinaisons linéaires, nous éliminons les termes en $f^{(i)}$ où $0 < i < j$. On obtient ainsi une expression de $f^{(j)}(x_0)$ ne contenant que des $f(x_0 - k\Delta x)$, avec $0 \leq k \leq j$:

$$f^{(j)}(x_0) = \frac{1}{\Delta x^j} \sum_{k=0}^{k=j} (-1)^k C_j^k f(x_0 - k\Delta x) + O(\Delta x) \quad (4.3)$$

Plutôt que d'exprimer $f^{(j)}(x_0)$ en fonction des $f(x_0 - k\Delta x)$ avec $k \in \llbracket 0..j \rrbracket$, nous aurons parfois besoin d'exprimer $f^{(j)}(x_0)$ en fonction des $f(x_0 - k\Delta x)$ pour k décrivant un autre ensemble.

La méthode est calquée sur la précédente. Nous donnerons simplement un exemple, ainsi que le résultat général. De :

$$\forall k \in \mathbb{Z}, f(x_0 - k\Delta x) = f(x_0) + \sum_{i=1}^{i=n} (-1)^i \frac{(k\Delta x)^i}{i!} f^{(i)}(x_0) + O(\Delta x^{n+1})$$

nous tirons que :

$$f(x_0 - \Delta x) = f(x_0) - \Delta x f^{(1)}(x_0) + \frac{\Delta x^2}{2} f^{(2)}(x_0) - \frac{\Delta x^3}{6} f^{(3)}(x_0) + O(\Delta x^4)$$

et :

$$f(x_0 + \Delta x) = f(x_0) + \Delta x f^{(1)}(x_0) + \frac{\Delta x^2}{2} f^{(2)}(x_0) + \frac{\Delta x^3}{6} f^{(3)}(x_0) + O(\Delta x^4)$$

En retranchant la première équation à la deuxième (et en supprimant le terme d'ordre 3), nous obtenons :

$$f(x_0 + \Delta x) - f(x_0 - \Delta x) = 2\Delta x f^{(1)}(x_0) + O(\Delta x^3)$$

et donc :

$$f^{(1)}(x_0) = \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x} + O(\Delta x^2)$$

De même, en ajoutant les deux équations, nous obtenons :

$$f(x_0 - \Delta x) + f(x_0 + \Delta x) = 2f(x_0) + \Delta x^2 f^{(2)}(x_0) + O(\Delta x^4)$$

et donc :

$$f^{(2)}(x_0) = \frac{f(x_0 - \Delta x) - 2f(x_0) + f(x_0 + \Delta x)}{\Delta x^2} + O(\Delta x^2)$$

Par une méthode similaire à celle des différences à gauche, il est possible d'obtenir la formule générale ($\lfloor \cdot \rfloor$ représente l'opérateur de partie entière) pour les différences centrées :

$$f^{(n)}(x_0) = \frac{1}{2\Delta x^n} \sum_{k=0}^{k=n} (-1)^k C_n^k \left[f\left(x_0 + \left(\left\lfloor \frac{n}{2} \right\rfloor - k\right) \Delta x\right) + (-1)^n f\left(x_0 - \left(\left\lfloor \frac{n}{2} \right\rfloor - k\right) \Delta x\right) \right] + O(\Delta x^2)$$

On constate en particulier que pour l'évaluation d'une dérivée particulière, il faut évaluer autant de valeurs de f (ce fait apparaît mieux avec d'autres écritures de l'expression, moins denses, mais qui ne font apparaître qu'une fois chaque valeur de f) que dans le cas des différences à gauche, mais qu'on obtient ici une précision en $O(\Delta x^2)$ au lieu de $O(\Delta x)$, ce qui est nettement mieux.

2.2 Schémas de discrétisation

Nous atteignons enfin notre but : discrétiser une équation différentielle ou aux dérivées partielles en ne laissant dans le schéma discret que les *valeurs* de la fonction cherchée.

Pour obtenir ces schémas discrets, on peut partir des expressions analytiques données plus haut et les discrétiser, ou bien discrétiser les développements de Taylor et par combinaisons linéaires obtenir notre schéma discret.

Dans la suite, nous considérons une fonction f de une ou deux variables x et y (selon le contexte). Les valeurs Δx et Δy sont les pas de discrétisation. De plus, nous noterons $F_{m,n}$ la valeur $f(m\Delta x, n\Delta y)$.

Nous allons donc exprimer $F_{m,n}^{(i)}$ en fonction des $F_{m+k,n+l}$. Pour une variable donnée (la première par exemple), nous dirons avoir procédé par différences à gauche si les valeurs de k sont négatives, par différences à droite si les valeurs de k sont positives et par différences centrées sinon.

2.2.1 Dérivées

Nous pouvons utiliser l'expression analytique de la dérivée d'ordre n ou bien repartir du développement de Taylor donné en (4.1). Écrivons le (sous forme discrète) à l'ordre 3 :

$$F_{m+\epsilon} = F_m + \epsilon \Delta x F_m^{(1)} + \frac{(\epsilon \Delta x)^2}{2} F_m^{(2)} + \frac{(\epsilon \Delta x)^3}{6} F_m^{(3)} + O(\Delta x^4)$$

Pour $\epsilon = 1$, cette expression nous donne le schéma des différences à droite :

$$F_m^{(1)} = \frac{1}{\Delta x} (F_{m+1} - F_m) + O(\Delta x)$$

Pour $\epsilon = -1$, cette expression nous donne le schéma des différences à gauche :

$$F_m^{(1)} = \frac{1}{\Delta x} (F_m - F_{m-1}) + O(\Delta x)$$

Par combinaisons linéaire des deux expressions obtenues pour $\epsilon = \pm 1$, nous obtenons le schéma des différences centrées :

$$F_m^{(1)} = \frac{1}{2\Delta x} (F_{m+1} - F_{m-1}) + O(\Delta x^2)$$

Ce dernier schéma est plus précis.

2.2.2 Dérivées partielles

Le développement par rapport à la première variable à l'ordre 3 est :

$$F_{m+\epsilon,n} = F_{m,n} + \epsilon \Delta x \left(\frac{\partial f}{\partial x} \right)_{m,n} + \epsilon^2 \frac{\Delta x^2}{2!} \left(\frac{\partial^2 f}{\partial x^2} \right)_{m,n} + \epsilon^3 \frac{\Delta x^3}{3!} \left(\frac{\partial^3 f}{\partial x^3} \right)_{m,n} + O(\Delta x^4)$$

En combinant ces expression pour $\epsilon = \pm 1$, nous obtenons le schéma centré :

$$\left(\frac{\partial^2 f}{\partial x^2} \right)_{m,n} = \frac{1}{\Delta x^2} (F_{m+1,n} - 2F_{m,n} + F_{m-1,n}) + O(\Delta x^2)$$

Chapitre 5

Équations différentielles



OUS ABORDONS à présent un des chapitres qui sera pour nous des plus importants puisqu'il concerne en particulier la simulation de systèmes régis par des équations différentielles. Des informations complémentaires pourront être trouvées dans [14] et [9]. Le sujet est abordé d'un point de vue pratique dans [3] et [6]

1 Premier ordre et conditions initiales

On traite ici le cas d'une équation différentielle du premier ordre avec conditions initiales. Il s'agit de déterminer numériquement la fonction $y(t)$, solution de :

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

Déterminer la fonction numériquement consiste à connaître avec une bonne précision certains de ses points, que nous noterons (t_i, y_i) (y_i sera donc une approximation de la valeur réelle de y en $t_i : y(t_i)$).

1.1 Méthode d'Euler

La méthode découle de l'écriture du développement de Taylor de y à l'ordre 1 :

$$y(t+h) = y(t) + hy'(t) + O(h^2)$$

Ce qui nous donne comme approximation :

$$y(t+h) \approx y(t) + hf(t, y(t))$$

La méthode d'Euler, partant de la condition initiale (t_0, y_0) consiste à calculer dans l'ordre les points (t_i, y_i) où $t_i = t_{i-1} + h$ (avec h le plus souvent indépendant de i).

Connaissant la valeur de y_i , la valeur de y_{i+1} sera donnée par :

$$y_{i+1} \leftarrow y_i + h \times f(t_i, y_i)$$

À chaque pas, il est donc nécessaire d'évaluer f . De plus, c'est le résultat du calcul au pas i (y_i) qui est réutilisé pour calculer y_{i+1} . Les erreurs de calcul (dues à la machine et à la troncature du développement) s'accumulent donc. La quantité $|y_i - y(t_i)|$ croît généralement avec i .

▷ *Matlab* : Fonction `sim()`

1.2 Méthode de Taylor

Comme son nom l'indique, cette méthode est basée sur le développement de Taylor à un ordre supérieur ou égal à 2 (sinon, c'est tout simplement la méthode d'Euler).

Détaillons cette méthode à l'ordre 2 :

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + O(h^3)$$

Puisque $y'(t) = f(t, y(t))$:

$$y(t+h) = y(t) + hf(t, y(t)) + \frac{h^2}{2}f'(t, y(t)) + O(h^3)$$

Or :

$$f'(t, y(t)) = \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \times y'(t) = \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \times f(t, y(t))$$

On obtient :

$$y(t+h) = y(t) + hf(t, y(t)) + \frac{h^2}{2} \left\{ \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \times f(t, y(t)) \right\} + O(h^3) \quad (5.1)$$

En ce qui nous concerne, nous utiliserons donc la relation :

$$y_{i+1} \leftarrow y_i + hf(t_i, y_i) + \frac{h^2}{2} \left\{ \frac{\partial f}{\partial t}(t_i, y_i) + \frac{\partial f}{\partial y}(t_i, y_i) \times f(t_i, y_i) \right\}$$

Un nouveau calcul d'erreur (similaire à celui décrit pour la méthode d'Euler) nous donnerait ici une erreur de troncature par pas de l'ordre de h^3 et une erreur de troncature globale en h^2 . En revanche, cette méthode nécessite de calculer, puis d'évaluer les dérivées partielles de f .

En procédant de façon similaire (développement de Taylor à un ordre plus important), on peut encore réduire l'ordre de grandeur de l'erreur. En contrepartie, il faudra calculer et évaluer des dérivées partielles de f d'ordre supérieur.

1.3 Méthodes de Runge-Kutta

Les méthodes de Runge-Kutta sont inspirées de la méthode de Taylor. Elles permettent de réduire l'erreur de la même façon, mais elles ne nécessitent pas le calcul de dérivées partielles.

1.3.1 Ordre 2

Nous détaillons ici la méthode de Runge-Kutta d'ordre 2. En s'inspirant du développement de Taylor, écrivons :

$$y(t+h) = y(t) + a_1hf(t, y(t)) + a_2hf(t+a_3h, y(t) + a_4h) + O(h^3) \quad (5.2)$$

Essayons à présent de déterminer a_1, a_2, a_3 et a_4 . Le développement de Taylor à deux variables autour du point $(t, y(t))$ donne (cf. chapitre 4) :

$$f(t+a_3h, y(t) + a_4h) = f(t, y(t)) + a_3h \frac{\partial f}{\partial t}(t, y(t)) + a_4h \frac{\partial f}{\partial y}(t, y(t)) + O(h^2)$$

En injectant cette expression dans la précédente, nous trouvons :

$$y(t+h) = y(t) + (a_1 + a_2)hf(t, y(t)) + a_2a_3h^2 \frac{\partial f}{\partial t}(t, y(t)) + a_2a_4h^2 \frac{\partial f}{\partial y}(t, y(t)) + O(h^3)$$

En identifiant les termes de cette expression à l'expression (5.1), nous obtenons un système d'équations dont les inconnues sont a_1, a_2, a_3 et a_4 :

$$\begin{cases} a_1 + a_2 = 1 \\ a_2a_3 = \frac{1}{2} \\ a_2a_4 = \frac{1}{2}f(t, y(t)) \end{cases}$$

Des solutions de ce système (il y en a plusieurs), injectées dans (5.2), nous permettent d'obtenir une méthode de calcul de $y(t+h)$ dont l'erreur (de méthode) est en $O(h^3)$ et qui ne nécessite pas le calcul de dérivées partielles.

À titre d'exemple, si nous prenons :

$$\begin{cases} a_1 = \frac{1}{2} \\ a_2 = \frac{1}{2} \\ a_3 = 1 \\ a_4 = f(t, y(t)) \end{cases}$$

nous aboutissons à la méthode de calcul suivante (appelée méthode d'Euler modifiée) :

$$y_{n+1} \leftarrow y_n + \frac{h}{2} \{f(t_n, y_n) + f(t_n + h, y_n + hf(t_n, y_n))\} \quad (5.3)$$

D'autres choix de paramètres conduisent à d'autres méthodes, comme celle du point milieu, avec $a_1 = 0$, $a_2 = 1$, $a_3 = \frac{1}{2}$ et $a_4 = \frac{1}{2}f(t, y(t))$.

1.3.2 Ordre 4

Les méthodes d'ordre 4 sont les plus utilisées en pratique. Les formules sont obtenues par un raisonnement similaire à celui qui donne la méthode d'ordre 2. On part de l'écriture suivante :

$$y(t+h) = y(t) + a_1 hf(t, y(t)) + a_2 hf(t + a_3 h, y(t) + a_4 h) + \dots$$

Le système obtenu est un système de 8 équations à 10 inconnues. Nous donnons ici le résultat le plus utilisé :

$$y_{i+1} \leftarrow y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

avec :

$$\begin{cases} k_1 = hf(t_i, y_i) \\ k_2 = hf\left(t_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right) \\ k_3 = hf\left(t_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right) \\ k_4 = hf(t_i + h, y_i + k_3) \end{cases}$$

▷ *Matlab* : Fonctions `ode23()` et `ode45()`

1.4 Méthodes d'Adams (pas multiple)

Les méthodes d'Adams sont elles aussi tirées du développement de Taylor. La différence essentielle entre les méthodes d'Adams et les précédentes est qu'elles nécessitent d'utiliser le résultat de plusieurs pas précédents pour calculer une nouvelle valeur.

1.4.1 Formules ouvertes : Adams-Bashforth

Réécrivons le développement de Taylor à l'ordre n de y (sous forme discrète), en remplaçant directement y' par f (f_i est mis pour $f(t_i, y_i)$) :

$$y_{i+1} = y_i + \sum_{k=1}^{k=n} \frac{h^k}{k!} f_i^{(k-1)} + O(h)$$

L'idée, ici est de remplacer les $f_i^{(j)}$ par ses valeurs en des points antérieurs à i . Nous disposons déjà de l'expression (4.3), de laquelle nous tirons :

$$f_i'' = \frac{1}{h^2}(f_i - 2f_{i-1} + f_{i-2}) + O(h)$$

Nous pouvons, par des principes similaires à ceux exposés au chapitre 4, obtenir¹ :

$$f'_i = \frac{1}{h} \left(\frac{3}{2}f_i - 2f_{i-1} + \frac{1}{2}f_{i-2} \right) + O(h^2)$$

Des deux expressions précédentes et du développement de Taylor (version discrète) à l'ordre 3 :

$$y_{i+1} = y_i + hf_i + \frac{h^2}{2!}f''_i + \frac{h^3}{3!}f'''_i + O(h^4)$$

nous tirons :

$$y_{i+1} = y_i + \frac{h}{12}(23f_i - 16f_{i-1} + 5f_{i-2}) + O(h^4)$$

Naturellement, le calcul de y_1 et y_2 ne peut pas se faire par cette formule. On utilise donc généralement la méthode de Runge-Kutta pour calculer ces deux valeurs.

D'une manière plus générale, la formule ouverte d'Adams à l'ordre $n + 1$ s'écrit sous la forme :

$$y_{i+1} = y_i + h \sum_{k=0}^{k=n} \beta_{n+1,k} f_{i-k} + O(h^{n+2})$$

Les coefficients $\beta_{n+1,k}$ (où $n + 1$ est l'ordre de la méthode) sont des nombres rationnels, et sont généralement calculés une fois pour toutes pour les ordres les plus courants. Voici une table, tirée de [14], donnant les premières valeurs de ces coefficients :

ordre	$k = 0$	$k = 1$	$k = 2$	$k = 3$	$k = 4$	$k = 5$
1	1					
2	$\frac{3}{2}$	$-\frac{1}{2}$				
3	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$			
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$		
5	$\frac{1901}{720}$	$-\frac{2774}{720}$	$\frac{2616}{720}$	$-\frac{1274}{720}$	$\frac{251}{720}$	
6	$\frac{4277}{1440}$	$-\frac{7923}{1440}$	$\frac{9982}{1440}$	$-\frac{7298}{1440}$	$\frac{2877}{1440}$	$-\frac{475}{1440}$

⑤ ~~~~~ ⑤ ~~~~~ ⑤

1.4.2 Formules fermées : Adams-Moulton

Le point de départ est le développement de Taylor de $y((t + h) - h)$:

$$y(t + h - h) = y(t + h) + \sum_{k=1}^{k=n} \frac{(-h)^k}{k!} y^{(k)}(t + h) + O(h^{n+1})$$

On remplace comme précédemment y' par f , et en discrétisant, on obtient :

$$y_i = y_{i+1} + \sum_{k=1}^{k=n} \frac{(-h)^k}{k!} f_{i+1}^{(k-1)} + O(h^{n+1})$$

Ce qui est équivalent à :

$$y_{i+1} = y_i + \sum_{k=1}^{k=n} -\frac{(-h)^k}{k!} f_{i+1}^{(k-1)} + O(h^{n+1})$$

Nous trouvons les formules d'Adams fermées par des méthode identiques à celles utilisée plus haut. À l'ordre 1 :

$$y_{i+1} = y_i + hf_{i+1}$$

¹Notons que l'expression (4.3) ne nous donnerait que : $f'_i = \frac{1}{h}(f_i - f_{i-1}) + O(h)$, ce qui ne nous conviendrait pas ici.

4.1 Équation linéaire

Dans le cas où l'équation est de type (nous renommons ici l'inconnue en x car les problèmes de conditions aux limites sont courants dans l'espace plutôt que dans le temps) :

$$y''(x) = A(x)y'(x) + B(x)y(x) + C(x) \quad A(x), B(x), C(x) \text{ sont des polynômes en } x$$

avec pour conditions aux limites :

$$y(a) = y_a \text{ et } \begin{cases} y(b) = y_b \\ \text{ou} \\ y'(b) = Y_b \end{cases}$$

il suffit de résoudre deux fois l'équation pour deux valeurs arbitraires de $y'(a)$ (on choisit généralement 0 et 1) avec une des méthodes qui précède. Supposons que les fonctions trouvées soient y_0 (pour la condition initiale $y'(a) = 0$) et y_1 (pour la condition initiale $y'(a) = 1$). En pratique, on dispose de valeurs numériques de ces fonctions en un certain nombre de points.

Dans le cas où la condition aux limites est $y(b) = y_b$, la solution mathématique de l'équation de départ est donnée par :

$$y(x) = \left(\frac{y_b - y_1(b)}{y_0(b) - y_1(b)} \right) y_0(x) + \left(\frac{y_0(b) - y_b}{y_0(b) - y_1(b)} \right) y_1(x)$$

On peut donc calculer les valeurs numériques approchées de $y(x)$ à partir des valeurs numériques approchées de y_0 et y_1 aux mêmes points.

Dans le cas où la conditions aux limites est $y'(b) = Y_b$, la solution de l'équation est donnée par :

$$y(x) = \left(\frac{Y_b - y'_1(b)}{y'_0(b) - y'_1(b)} \right) y_0(x) + \left(\frac{y'_0(b) - Y_b}{y'_0(b) - y'_1(b)} \right) y_1(x)$$

4.2 Équations non linéaires

La méthode est un peu similaire mais ne permet pas un calcul direct de la solution à partir de y_0 et y_1 . Détaillons le cas où les conditions aux limites sont : $y(a) = y_a$ et $y(b) = y_b$ (l'autre cas est similaire). On commence par résoudre deux équations différentielles ayant pour conditions initiales :

$$\begin{array}{lcl} y_{\beta_0}(a) & = & y_a \\ y'_{\beta_0}(a) & = & \beta_0 \end{array} \text{ puis } \begin{array}{lcl} y_{\beta_1}(a) & = & y_a \\ y'_{\beta_1}(a) & = & \beta_1 \end{array}$$

où β_0 et β_1 sont arbitraires (nous verrons plus loin des exemples de choix judicieux), mais de telle sorte que : $(y_{\beta_0}(b) - y_b) \times (y_{\beta_1}(b) - y_b) < 0$. Cette condition signifie que β_0 et β_1 sont de part et d'autre de la solution de l'équation $\Delta(\beta) = y_\beta(b) - y_b = 0$ On peut alors estimer une nouvelle valeur de β (β_2) plus proche de la solution (en utilisant la méthode de Newton par exemple) :

$$\beta_2 = \frac{\beta_0 + \beta_1}{2}$$

On résout alors à nouveau l'équation différentielle avec pour conditions initiales :

$$\begin{array}{lcl} y_{\beta_2}(a) & = & y_a \\ y'_{\beta_2}(a) & = & \beta_2 \end{array}$$


Selon le signe de $y_{\beta_2}(b) - y_b$, on réitère le processus en partant de β_0 et β_2 ou bien de β_1 et β_2 jusqu'à ce que la solution soit acceptable (i.e. on s'arrête lorsque $|y_{\beta_k}(b) - y_b| < \epsilon$).

Cette méthode nécessite donc de résoudre deux fois l'équation différentielle au départ, puis de la résoudre une nouvelle fois à chaque itération.

On a donc intérêt à choisir correctement les valeurs initiales β_0 et β_1 . Il est conseillé [8] de prendre : $\beta_0 = \frac{y_b - y_a}{b - a}$ (approximation linéaire de y), et β_1 de façon à avoir effectivement le changement de signe.

Chapitre 6

Équations aux dérivées partielles

ANS LA continuité du chapitre précédent, nous abordons maintenant la question de la résolution des équations aux dérivées partielles. Après avoir brièvement caractérisé ces équations et avoir en partie traité les problèmes de consistance et de stabilité, nous nous intéresserons tout particulièrement à la méthode des différences finies. Des informations complémentaires pourront être trouvées dans [9], [6]. Le sujet est traité en détail dans [15]. Nous ne ferons qu'évoquer la méthode des éléments finis mais le lecteur intéressé pourra se reporter à [7], [12] ou [17].

1 Rappel sur les coniques

Une conique de \mathbb{R}^3 (non dégénérée) est définie par :

$$\phi(x, y) = ax^2 + 2bxy + cy^2 + dx + ey + f = 0$$

Les directions asymptotiques sont obtenues en négligeant les termes de degré 1 ou moins et sont données par $m = \frac{y}{x}$. Elles sont solution de :

$$cm^2 + 2bm + a = 0$$

Posons $\Delta = b^2 - ac$. Selon le signe de Δ , on peut déterminer le nombre de directions asymptotiques :

- si $\Delta > 0$, il y a deux directions asymptotiques réelles, et la conique est une hyperbole ;
- si $\Delta = 0$, il y a une direction asymptotique réelle, et la conique est une parabole ;
- si $\Delta < 0$, il n'y a pas de direction asymptotique réelle, et la conique est une ellipse.

On peut montrer qu'après un changement de repère, si l'équation de la même conique devient :

$$\Phi(X, Y) = AX^2 + 2BXY + CY^2 + DX + EY + F = 0$$

le signe de $B^2 - AC$ est le même que celui de $b^2 - ac$. Le signe de cette valeur est donc une *caractéristique* de la conique, indépendante du repère (ce dont on aurait pu se douter).

La conique a une forme canonique, c'est à dire qu'il existe un repère dans lequel son équation est :

$$\Phi(X, Y) = AX^2 + CY^2 + F = 0 \text{ où } F \text{ ne contient que des termes de degré 1 au plus}$$

On peut montrer que les coefficients A et C sont les racines de l'équation :

$$\lambda^2 - (a + c)\lambda - b^2 + ac$$

Les coefficients A et C sont donc les valeurs propres de la matrice¹ :

$$Q = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

En conséquence, le signe de Δ est l'opposé du signe du produit des deux valeurs propres de Q . Donc, la conique $\phi(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0$ peut être classifiée en fonction du signe des valeurs propres de Q :

- Si Q a deux valeurs propres non nulles de même signe, alors la conique est une ellipse.
- Si Q a deux valeurs propres non nulles de signes opposés, alors la conique est une hyperbole.
- Si Q a une valeur propre nulle, alors la conique est une parabole.

¹Puisque les valeurs propres sont les solutions du polynôme $\det(Q - \lambda I) = 0$.

2 Classification des équations d'ordre 2

Une e.d.p. d'ordre 2 à deux variables indépendantes s'écrit :

$$a \frac{\partial^2 f}{\partial x^2} + b \frac{\partial^2 f}{\partial x \partial y} + c \frac{\partial^2 f}{\partial y^2} + d = 0$$

où a , b , c et d dépendent éventuellement de x , y , f , $\frac{\partial f}{\partial x}$ et $\frac{\partial f}{\partial y}$.

L'essentiel des résultats sur les coniques que nous venons de présenter peuvent être adaptés à une telle équation. Le changement de base des coniques devient un changement de variable, et on peut montrer que le signe de la quantité $b^2 - 4ac$ se conserve par un tel changement de variable. Il est donc caractéristique de l'e.d.p. et par analogie avec les coniques :

- Si $b^2 - 4ac > 0$, l'équation est dite hyperbolique.
- Si $b^2 - 4ac = 0$, l'équation est dite parabolique.
- si $b^2 - 4ac < 0$, l'équation est dite elliptique.

Les méthodes de résolution des e.d.p. d'ordre 2 sont plus ou moins bien adaptées à tel ou tel type d'e.d.p. et savoir les classifier est donc fondamental.

À titre d'exemple, voici quelque équations «célèbres» et leur type :

- **L'équation des ondes :**

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0$$

est hyperbolique car $\Delta = (0)^2 - 4 \times (1) \times (-c^2) > 0$ (si $c \neq 0$).

- **L'équation de Laplace :**

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

est elliptique car $\Delta = (0)^2 - 4 \times (1) \times (1) < 0$.

- **L'équation de la diffusion :**

$$\frac{\partial u}{\partial t} = \sigma \frac{\partial^2 u}{\partial x^2}$$

est parabolique, car $\Delta = (0)^2 - 4 \times (1) \times (0) = 0$

La classification des e.d.p. d'ordre 2 s'étend au cas où il y a plus de deux variables indépendantes. Soit l'e.d.p. :

$$\sum_{i=1}^{i=N} \sum_{j=1}^{j=N} a_{ij} \frac{\partial^2 f}{\partial x_i \partial x_j} + b = 0$$

où b ne contient pas de dérivées secondes ou croisées. On considère la matrice $A = [a_{ij}]$ des coefficients de plus haut degré de l'e.d.p. Le signe des valeurs propres de A indique le type d'équation :

- Si A a toutes ses v.p. de même signe et non nulles, l'équation est elliptique.
- Si A a au moins une v.p. nulle, l'équation est parabolique.
- Si A a au moins deux v.p. de signes opposés, mais pas de v.p. nulle, l'équation est hyperbolique.

Dans ce chapitre, nous présenterons la méthode des différences finies, et l'appliquerons aux trois types d'équations aux dérivées partielles d'ordre 2.

3 Méthode des différences finies

La méthode rappelle celle que nous utilisons pour les équations différentielles. Il s'agit, à partir du développement de Taylor de discrétiser les dérivées de la fonction par une combinaison de certaines de ses valeurs. Selon l'utilisation des différences à gauche, à droite ou centrée, on obtient différents schémas de discrétisation aux différences finies. Tous ne sont pas équivalents, et une étude fine est nécessaire pour choisir le schéma représentant le meilleur compromis.

La présentation générale de la méthode va nous permettre d'aborder les notions de consistante, de stabilité et de convergence.

3.1 Consistance

Une méthode numérique est consistante si l'erreur de discrétisation de l'équation tend vers 0 lorsque le (les) pas de discrétisation tend(ent) vers 0.

Voyons ce que signifie cette définition sur un exemple. Considérons l'équation aux dérivées partielles :

$$\frac{\partial u}{\partial t} - \sigma \frac{\partial^2 u}{\partial x^2} = 0$$

Nous devons choisir des schémas de discrétisation (voir chapitre 4) pour cette équation. Par exemple (notons $U_{m,n}$ la valeur au temps $n\Delta t$ et à l'abscisse $m\Delta x$) :

$$\begin{aligned} \left(\frac{\partial^2 u}{\partial x^2}\right)_{m,n} &= \frac{U_{m-1,n} - 2U_{m,n} + U_{m+1,n}}{\Delta x^2} - \frac{\Delta x^2}{12} \left(\frac{\partial^4 u}{\partial x^4}\right)_{m,n} + O(\Delta x^4) \\ \left(\frac{\partial u}{\partial t}\right)_{m,n} &= \frac{U_{m,n+1} - U_{m,n}}{\Delta t} - \frac{\Delta t}{2} \left(\frac{\partial^2 u}{\partial t^2}\right)_{m,n} + O(\Delta t^2) \end{aligned}$$

En négligeant les termes de degré 2 en Δx et 1 en Δt , nous obtenons comme schéma de calcul pour notre équation :

$$\frac{U_{m,n+1} - U_{m,n}}{\Delta t} - \sigma \frac{U_{m-1,n} - 2U_{m,n} + U_{m+1,n}}{\Delta x^2} = 0$$

L'erreur de troncature vaut les termes que nous avons négligé :

$$R(u, \Delta x, \Delta t) = \frac{\Delta t}{2} \left(\frac{\partial^2 u}{\partial t^2}\right)_{m,n} - \sigma \frac{\Delta x^2}{12} \left(\frac{\partial^4 u}{\partial x^4}\right)_{m,n} + O(\Delta t^2) + O(\Delta x^4)$$

Or, $R(u)$ tend vers 0 lorsque Δt et Δx tendent vers 0. En conséquence, le schéma est consistant, puisqu'en diminuant le pas de temps et le pas d'espace, notre équation discrète approchée tend effectivement vers l'équation que nous devons résoudre.

Ce n'est néanmoins pas toujours le cas, et il peut y avoir des conditions à la consistance. Le schéma discret suivant de la même équation (dit de Dufort et Frankel) :

$$\frac{U_{m,n+1} - U_{m,n-1}}{2\Delta t} - \sigma \frac{U_{m-1,n} - U_{m,n+1} - U_{m,n-1} + U_{m+1,n}}{\Delta x^2} = 0$$

a pour erreur de troncature :

$$R(u, \Delta x, \Delta t) = \sigma \frac{\Delta t^2}{\Delta x^2} \frac{\partial^2 u}{\partial t^2} + O(\Delta t^2) + O(\Delta x^2)$$

et $R(u, \Delta x, \Delta t)$ tend vers 0 lorsque Δx et Δt tendent vers 0, *uniquement si le rapport $\frac{\Delta t}{\Delta x}$ tend lui aussi vers 0*.

Nous avons à présent le moyen de contrôler que le schéma discret choisi est une bonne approximation de l'équation à résoudre.

3.2 Stabilité

Une fois le schéma discret choisi, il va être nécessaire de le résoudre. Le processus de résolution, à la vue des équations sera la plupart du temps itératif. On calculera les valeurs de U de proche en proche : une valeur donnée de U sera donc calculée en utilisant le résultat du calcul d'autres valeurs de U . Les erreurs d'arrondi étant inévitables sur machine, la méthode sera stable si ces erreurs ne s'amplifient pas (trop) au cours du calcul.

Nous donnons seulement ici une «recette» (justifiée dans [15]) pour vérifier dans une certaine mesure la stabilité du processus de calcul. Soit un équation discrétisée :

$$\sum_{l=-l_1, k=-k_1}^{l=l_2, k=k_2} a_{lk} U_{m+l, n+k} = 0$$

avec k_1, k_2, l_1 et l_2 positifs. En remplaçant $U_{m+l, n+k}$ par $\xi_m^k e^{i\omega_m l \Delta x}$ dans le schéma de résolution, on obtient un polynôme de degré $N - 1$ en ξ_m si N pas de temps sont mis en jeu. On admettra que le schéma est stable si

$\max_m |\xi_m| \leq 1.$

④ ~~~~~ ④

Voyons comment étudier la stabilité sur un exemple. Si nous reprenons le premier schéma de discrétisation de la section précédente, et effectuons le remplacement ci-dessus, nous obtenons :

$$\frac{\xi_m}{\Delta t} - \frac{1}{\Delta t} - \frac{\sigma}{\Delta x^2} [e^{-i\omega_m \Delta x} + e^{i\omega_m \Delta x} - 2] = 0$$

Cette expression nous donne :

$$\xi_m = 1 + 2 \frac{\sigma \Delta t}{\Delta x^2} (\cos(\omega_m \Delta x) - 1)$$

On a :

$$\xi_m \geq 1 + 2 \frac{\sigma \Delta t}{\Delta x^2} (-1 - 1)$$

$$\xi_m \leq 1 + 2 \frac{\sigma \Delta t}{\Delta x^2} (1 - 1)$$

C'est à dire :

$$\forall m, 1 - 4 \frac{\sigma \Delta t}{\Delta x^2} \leq \xi_m \leq 1$$

Par conséquent, le calcul sera stable si : $\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2\sigma}$.

Il est tout à fait possible que la ou les conditions de stabilité soient incompatibles avec les conditions de consistance.

Notons au passage que le schéma de Dufort et Frankel, cité plus haut est par exemple stable sans condition, alors qu'il n'est pas toujours consistant.

3.3 Convergence

Après s'être assuré que le schéma discret tend vers l'équation, que ce schéma conduit à un calcul stable, on dispose d'une solution. Le schéma utilisé est dit convergent si sa solution ainsi obtenue tend vers la solution de l'équation de départ, lorsque les pas de discrétisation tendent vers 0.

Nous ne citerons ici qu'un résultat (dû à Lax), permettant de vérifier la convergence dans certains cas : *Si le problème est linéaire² et bien posé³, consistance et stabilité sont nécessaires et suffisantes pour assurer la convergence.*

4 Équations paraboliques

Rappelons la forme d'une équation différentielle parabolique :

$$a \frac{\partial^2 u}{\partial x^2} + 2b \frac{\partial^2 u}{\partial x \partial t} + c \frac{\partial^2 u}{\partial t^2} = g \text{ avec } \Delta = b^2 - ac = 0$$

où a , b , c et g dépendent éventuellement de x , t , u et de ses dérivées du premier ordre.

4.1 Mise sous forme canonique

Il existe un changement de variables tel que l'équation s'écrive :

$$\frac{\partial^2 u}{\partial X^2} = G$$

où G dépend éventuellement de x , t , u et de ses dérivées du premier ordre. Nous allons voir comment trouver un tel changement de variables. Soient $X(x, t)$ et $T(x, t)$ deux fonctions de x et t . Calculons $\frac{\partial^2 u}{\partial x^2}$, $\frac{\partial^2 u}{\partial t^2}$ et $\frac{\partial^2 u}{\partial x \partial t}$ en faisant apparaître les nouvelles variables X et T :

$$\frac{\partial u}{\partial x} = \frac{\partial u}{\partial X} \frac{\partial X}{\partial x} + \frac{\partial u}{\partial T} \frac{\partial T}{\partial x}$$

²Ce sera le cas dans ce chapitre.

³Un problème bien posé est un problème qui, mathématiquement, admet une unique solution et dépend de façon continue des conditions aux limites comme c'est généralement le cas pour un problème issu de la physique.

$$\frac{\partial^2 u}{\partial x^2} = \left(\frac{\partial^2 u}{\partial X^2} \frac{\partial X}{\partial x} + \frac{\partial^2 u}{\partial X \partial T} \frac{\partial T}{\partial x} \right) \frac{\partial X}{\partial x} + \frac{\partial u}{\partial X} \frac{\partial^2 X}{\partial x^2} + \left(\frac{\partial^2 u}{\partial T^2} \frac{\partial T}{\partial x} + \frac{\partial^2 u}{\partial X \partial T} \frac{\partial X}{\partial x} \right) \frac{\partial T}{\partial x} + \frac{\partial u}{\partial T} \frac{\partial^2 T}{\partial x^2}$$

Posons :

$$\alpha = \frac{\partial X}{\partial x}, \beta = \frac{\partial X}{\partial t}, \gamma = \frac{\partial T}{\partial x}, \delta = \frac{\partial T}{\partial t}$$

On obtient :

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} &= \alpha^2 \frac{\partial^2 u}{\partial X^2} + 2\alpha\gamma \frac{\partial^2 u}{\partial X \partial T} + \gamma^2 \frac{\partial^2 u}{\partial T^2} + \frac{\partial u}{\partial X} \frac{\partial^2 X}{\partial x^2} + \frac{\partial u}{\partial T} \frac{\partial^2 T}{\partial x^2} \\ \frac{\partial^2 u}{\partial x^2} &= \alpha^2 \frac{\partial^2 u}{\partial X^2} + 2\alpha\gamma \frac{\partial^2 u}{\partial X \partial T} + \gamma^2 \frac{\partial^2 u}{\partial T^2} + a' \end{aligned}$$

où a' ne contient que des dérivées de u d'ordre 1 au plus. En procédant de même avec les autres dérivées, on obtient :

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= \beta^2 \frac{\partial^2 u}{\partial X^2} + 2\beta\delta \frac{\partial^2 u}{\partial X \partial T} + \delta^2 \frac{\partial^2 u}{\partial T^2} + c' \\ \frac{\partial^2 u}{\partial x \partial t} &= \alpha\beta \frac{\partial^2 u}{\partial X^2} + \gamma\delta \frac{\partial^2 u}{\partial T^2} + (\alpha\delta + \beta\gamma) \frac{\partial^2 u}{\partial X \partial T} + b' \end{aligned}$$

L'équation de départ devient donc :

$$\overbrace{(a\alpha^2 + 2b\alpha\beta + c\beta^2)}^A \frac{\partial^2 u}{\partial X^2} + 2 \overbrace{(a\alpha\gamma + c\beta\delta + b(\alpha\delta + \beta\gamma))}^B \frac{\partial^2 u}{\partial X \partial T} + \overbrace{(a\gamma^2 + c\delta^2 + 2b\gamma\delta)}^C \frac{\partial^2 u}{\partial T^2} = G$$

où G ne contient que des dérivées de u d'ordre inférieur ou égal à 1. Jusqu'à présent, ces calculs sont valables pour les trois types d'équation (parabolique, hyperbolique et elliptique). À présent, nous voulons annuler B et C et savons que $b^2 - ac = 0$. Ceci nous permet d'écrire :

$$\begin{aligned} a\alpha\gamma + c\beta\delta + b(\alpha\delta + \beta\gamma) &= 0 \\ a\gamma^2 + c\delta^2 + 2b\gamma\delta &= 0 \\ b^2 - ac &= 0 \end{aligned}$$

De la deuxième équation, un polynôme de degré 2 en γ , nous tirons (en utilisant que $b^2 = ac$) :

$$\gamma = \frac{-b\delta}{a}$$

Pour que le changement de base ne soit pas dégénéré, il faut de plus que $\alpha\delta - \beta\gamma \neq 0$. Nous avons cependant une certaine latitude dans le choix des paramètres. Par exemple :

$$\begin{cases} \alpha = 1 \\ \beta = 0 \\ \gamma = -b \\ \delta = a \end{cases}$$

qui nous donne bien la forme canonique.

4.2 Résolution par les différences finies

Nous travaillerons sur l'équation de la diffusion, déjà rencontrée, et qui est généralement donnée sous forme canonique :

$$\frac{\partial u}{\partial t} = \sigma \frac{\partial^2 u}{\partial x^2}$$

Notons au passage que les équations de la physique sont le plus souvent données sous forme canonique, qu'elles soient paraboliques, hyperboliques ou elliptiques.

Comme c'est souvent le cas pour les équations de ce type, le temps intervient. Le vocabulaire employé au chapitre 4 est donc légèrement particularisé. Nous parlerons de niveaux de temps pour indiquer combien de pas de temps apparaissent dans l'équation discrète, ainsi que de schéma implicite, explicite ou mixte pour indiquer que les valeurs de la fonction à la date t sont obtenues par des valeurs de la fonction à des dates postérieures, antérieures, ou les deux.

Les valeurs de la fonction à la date $n\Delta t$ et à l'abscisse $m\Delta x$ sont notées $U_{m,n}$.

Nous avons jusqu'à présent proposé deux schémas de discrétisation que nous rappelons ici :

$$\begin{aligned} \text{schéma 1} \quad & \frac{U_{m,n+1} - U_{m,n}}{\Delta t} - \sigma \frac{U_{m-1,n} - 2U_{m,n} + U_{m+1,n}}{\Delta x^2} = 0 \\ \text{schéma 2} \quad & \frac{U_{m,n+1} - U_{m,n-1}}{2\Delta t} - \sigma \frac{U_{m-1,n} - U_{m,n+1} - U_{m,n-1} + U_{m+1,n}}{\Delta x^2} = 0 \end{aligned}$$

Nous donnons à présent un troisième schéma, obtenu par une première façon d'écrire la dérivée partielle d'ordre 2 en x , que nous avons déjà vue, mais qui est ici exprimée à la date $(n+1)\Delta t$:

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{m,n+1} \approx \frac{1}{\Delta x^2} (U_{m-1,n+1} - 2U_{m,n+1} + U_{m+1,n+1})$$

Puis, nous utilisons les différences à gauche à l'instant $t + \Delta t$ pour la dérivée d'ordre 1 en t :

$$U_{m,n+1-1} \approx U_{m,n+1} - \Delta t \left(\frac{\partial u}{\partial t}\right)_{m,n+1}$$

C'est à dire :

$$\left(\frac{\partial u}{\partial t}\right)_{m,n+1} \approx \frac{U_{m,n+1} - U_{m,n}}{\Delta t}$$

Nous avons deux expressions au point m , à l'instant $n+1$ que l'on peut donc utiliser comme schéma pour notre équation :

$$\text{schéma 3} \quad \frac{U_{m,n+1} - U_{m,n}}{\Delta t} - \sigma \frac{U_{m-1,n+1} - 2U_{m,n+1} + U_{m+1,n+1}}{\Delta x^2} = 0$$

Ces trois schémas peuvent être réécrits, en posant $r = \frac{\sigma\Delta t}{\Delta x^2}$:

$$\begin{aligned} \text{schéma 1} \quad & U_{m,n+1} = U_{m,n} + r(U_{m-1,n} - 2U_{m,n} + U_{m+1,n}) \\ \text{schéma 2} \quad & U_{m,n+1} = \frac{2r}{1+2r} (U_{m-1,n} + U_{m+1,n}) + \frac{1-2r}{1+2r} U_{m,n-1} \\ \text{schéma 3} \quad & U_{m,n+1} = U_{m,n} + r(U_{m-1,n+1} - 2U_{m,n+1} + U_{m+1,n+1}) \end{aligned}$$

De ces trois schémas, nous voyons que les deux premiers sont explicites (le calcul au temps $n+1$ ne fait intervenir que des valeurs *antérieures*) et le troisième implicite (le calcul au temps $n+1$ ne peut pas se faire directement).

Les schémas 1 et 3 sont à deux niveaux de temps. En conséquence, la connaissance de la fonction au temps 0 suffit pour lancer le processus de résolution (c'est généralement la condition initiale qui est donnée). En revanche, le schéma numéro 2 est à trois niveaux de temps, et la donnée de la fonction à la date 0 ne suffit plus. Ce schéma présentant l'intérêt d'être stable sans condition *et* explicite, il reste assez utilisé et est initialisé par l'utilisation d'un autre schéma à deux niveaux de temps uniquement.

5 Équations hyperboliques

Rappelons la forme d'une équation différentielle hyperbolique :

$$a \frac{\partial^2 u}{\partial x^2} + 2b \frac{\partial^2 u}{\partial x \partial t} + c \frac{\partial^2 u}{\partial t^2} = g \text{ avec } \Delta = b^2 - ac > 0$$

où a , b , c et g dépendent éventuellement de x , t , u et ses dérivées du premier ordre.

5.1 Mise sous forme canonique

Nous ne référons pas ici les calculs, tout à fait similaires à ceux réalisés pour les équations paraboliques. Nous donnerons seulement les *deux* formes canoniques :

$$A \frac{\partial^2 u}{\partial X^2} + C \frac{\partial^2 u}{\partial T^2} + D = 0 \quad (AC < 0)$$

et

$$B \frac{\partial^2 u}{\partial X \partial T} + D' = 0 \quad (B \neq 0)$$

où A, B, C, D et D' dépendent éventuellement de X, T, u et des dérivées d'ordre un de u .

⑥ ~~~~~ ⑥

5.2 Différences finies

Nous travaillerons sur l'exemple de l'équation des ondes, qui est déjà sous forme canonique :

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0$$

Proposons un premier schéma de discrétisation :

$$\frac{1}{\Delta t^2} (U_{m,n-1} - 2U_{m,n} + U_{m,n+1}) - \frac{c^2}{\Delta x^2} (U_{m-1,n} - 2U_{m,n} + U_{m+1,n}) = 0$$

En posant $r = \frac{c^2 \Delta t^2}{\Delta x^2}$, nous obtenons le schéma explicite suivant :

$$U_{m,n+1} = 2(1-r)U_{m,n} + r(U_{m-1,n} + U_{m+1,n}) - U_{m,n-1}$$

En utilisant la méthode présentée plus haut, nous pouvons montrer que ce schéma est stable si : $c \frac{\Delta t}{\Delta x} \leq 1$.

5.3 Décomposition en équations du premier ordre

Notons que les équations hyperboliques peuvent être décomposées en équations du premier ordre. L'équation des ondes :

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0$$

peut ainsi s'écrire :

$$\begin{aligned} \frac{\partial y}{\partial t} + c \frac{\partial y}{\partial x} &= 0 \\ \frac{\partial u}{\partial t} - c \frac{\partial u}{\partial x} &= y \end{aligned}$$

Chacune de ces deux équations peut être résolue plus simplement que l'équation de départ.

6 Équations Elliptiques

Rappelons la forme d'une équation différentielle elliptique :

$$a \frac{\partial^2 u}{\partial x^2} + 2b \frac{\partial^2 u}{\partial x \partial y} + c \frac{\partial^2 u}{\partial y^2} = g \text{ avec } \Delta = b^2 - ac < 0$$

où a, b, c et g dépendent éventuellement de x, y, u et de ses dérivées du premier ordre.

6.1 Mise sous forme canonique

Un équation elliptique peut être mise sous forme canonique :

$$A \frac{\partial^2 u}{\partial X^2} + C \frac{\partial^2 u}{\partial Y^2} + D = 0 \quad (AC > 0)$$

où A, C et D dépendent éventuellement de X, Y, u et des dérivées d'ordre un de u . Les variables x et t on ici été renommées en x et y car, physiquement, les équations elliptiques correspondent généralement à des problèmes stationnaires.

6.2 Différences finies

Les conditions aux limites d'une équation elliptique sont généralement données sur une frontière fermée du domaine (de par la nature des phénomènes physiques qui correspondent). Ce fait facilite la résolution matricielle des équations, comme nous allons le voir dans le cas de l'équation de Poisson monodimensionnelle :

$$\frac{\partial^2 f}{\partial x^2} = g(x)$$

Discrétisée, l'équation devient :

$$F_{m-1} - 2F_m + F_{m+1} = \Delta x^2 g(m\Delta x)$$

La fonction g étant connue ainsi que les conditions aux limites donnant $f(0) = F_0$ et $f(N\Delta x) = F_N$ nous obtenons le système (en posant $g_m = g(m\Delta x)$) :

$$\begin{aligned} -2F_1 + F_2 &= \Delta x^2 g_1 - F_0 \\ F_1 - 2F_2 + F_3 &= \Delta x^2 g_2 \\ F_2 - 2F_3 + F_4 &= \Delta x^2 g_3 \\ &\dots = \dots \\ F_{N-2} - 2F_{N-1} &= \Delta x^2 g_{N-1} - F_N \end{aligned}$$

Si nous notons F le vecteur (F_1, \dots, F_{N-1}) ,

G le vecteur $(\Delta x^2 g_1 - F_0, \Delta x^2 g_2, \dots, \Delta x^2 g_{N-2}, \Delta x^2 g_{N-1} - F_N)$

et A la matrice :

$$\begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \dots & 0 & 1 & -2 \end{pmatrix}$$

la solution F est obtenue en résolvant le système :

$$AF = G$$

ce qui peut être réalisé avec toutes les méthodes du chapitre 2. Dans le cas d'une équation à deux dimensions, comme l'équation de Laplace :

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$

Si nous choisissons comme schéma discret (nous notons $F_{m,n}$ la valeur calculée de $f(m\Delta x, n\Delta y)$) :

$$\frac{1}{\Delta x^2} (F_{m-1,n} - 2F_{m,n} + F_{m+1,n}) + \frac{1}{\Delta y^2} (F_{m,n-1} - 2F_{m,n} + F_{m,n+1}) = 0$$

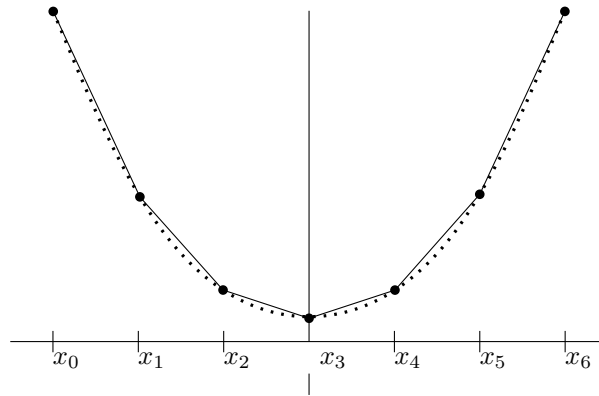
Nous obtenons aussi un système linéaire en construisant un vecteur F à partir des valeurs $F_{m,n}$ en les ordonnant ainsi par exemple : $(F_{1,1}, F_{1,2}, \dots, F_{1,N-1}, F_{2,1}, \dots, F_{2,N-1}, \dots, F_{M-1,1}, \dots, F_{M-1,N-1})$. La matrice obtenue ne sera plus tridiagonale, mais la résolution du système linéaire restera possible.

7 Quelques mots sur la méthode des éléments finis

Nous ne ferons qu'évoquer ici le principe général de la méthode. Comme nous venons de le voir, la méthode des différences finies consiste, dans le principe, à trouver la fonction recherchée sous forme de valeurs discrètes en certains points d'un maillage. La méthode des éléments finis consiste de son côté à trouver une approximation de la fonction recherchée sous forme analytique en chacune des mailles (qu'on appellera alors éléments) de l'espace des variables.

Prenons un exemple en dimension 1 et supposons que la fonction recherchée, comme solution d'une équation différentielle, soit $f(x) = x^2 + 1$ dans l'intervalle $[x_m, x_M]$. Naturellement, dans ce cas précis, la solution pourrait sans doute être trouvée analytiquement. La méthode des éléments finis consistera à découper l'intervalle $[x_m, x_M]$ en N intervalles (les éléments) : $[x_i, x_{i+1}]$, avec $i \in \llbracket 0 \dots N-1 \rrbracket$, $x_0 = x_m$ et $x_N = x_M$. Sur chaque élément $[x_i, x_{i+1}]$,

on pourra rechercher la fonction f sous forme d'une fonction affine. Le résultat obtenu sera une approximation de la fonction recherchée comme l'indique la figure suivante :



Dans l'exemple précédent, pour chaque élément, nous recherchons une approximation de la fonction sous la forme : $y = a_1x + a_0$ (polynôme de degré 1), puis nous ajoutons des contraintes de continuité entre chaque élément.

Chapitre 7

Optimisation

A PRÈS la résolution d'équations différentielles ou aux dérivées partielles, le second thème qui sera pour nous le plus important est abordé dans ce chapitre. Les problèmes d'optimisation sont aussi abordés dans [14] et [9], ou dans des ouvrages de recherche opérationnelle. Nous nous intéresserons ici aux problèmes suivants :

- trouver l'argument $x^* \in \mathbb{R}^n$ qui minimise une fonction quelconque J sur un certain intervalle (optimisation sans contrainte);
- trouver l'argument $x^* \in \mathbb{R}^n$ qui minimise une fonction quelconque J sur un certain intervalle, x^* devant par ailleurs remplir un certain nombre de conditions, comme annuler une certaine fonction g (optimisation sous contraintes).

Selon que la fonction J ou les contraintes sont des fonctions linéaires ou non, nous dirons avoir affaire à un problème de programmation linéaire ou à un problème de programmation non linéaire.

La programmation linéaire (J et les contraintes sont toutes linéaires) est abordée dans le chapitre suivant. Dans ce chapitre, nous nous intéresserons essentiellement aux cas où une des fonctions du problème n'est pas linéaire.

1 Forme standard

Un problème d'optimisation est dit sous forme standard s'il s'écrit :

$$\min_{x \in \mathbb{R}^n} J(x) \text{ avec } g(x) \leq 0 \text{ et } h(x) = 0$$

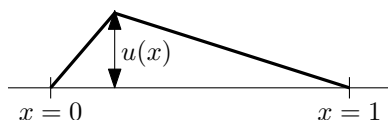
2 Exemples de problèmes

Identification On dispose d'un ensemble de mesures $\{(t_i, y_i), i \in [1..m]\}$.

Le modèle proposé étant $f(x) = a \exp(-bt) \cos(ct + d)$, on désire minimiser :

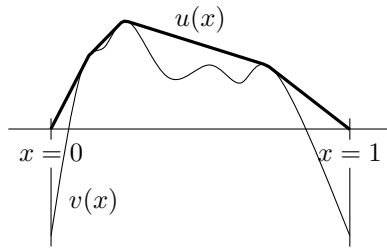
$$J(a, b, c, d) = \sum_{i=1}^{i=m} (y_i - f(t_i))^2$$

Déformation d'une corde $u(0) = u(1) = 0$, et au repos $\forall x \in [0, 1], u(x) = 0$:



En présence d'un obstacle, u minimise :

$$\frac{1}{2} \int_0^1 \tau \left[\frac{du}{dx} \right]^2 dx \text{ avec } \forall x \in]0, 1[, u(x) \geq v(x)$$



L'inconnue étant une fonction, on n'est pas sous forme standard. On s'y ramène en discrétisant le problème : Trouver $U \in \mathbb{R}^{N+1}$ qui minimise :

$$\frac{1}{2}U^T A U$$

$$\text{avec } A = \tau N \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix} \text{ et } \forall k \in \llbracket 1..N-1 \rrbracket U_k \geq V_k$$



3 Transformation en problème linéaire

Si les contraintes sont linéaires (elles définissent un polyèdre), et que seule la fonction J est non-linéaire, il peut être intéressant de «linéariser» le problème, en remplaçant J par son développement de Taylor à l'ordre 1 par exemple. Partant d'un point x_0 , la minimisation de J est donc remplacée par la minimisation de $J(x_0) + \vec{\nabla} J(x_0)(x - x_0)$ c'est à dire la minimisation de $\vec{\nabla} J(x_0)x$.

Ce genre de problème entre dans le cadre de la programmation linéaire détaillée au chapitre suivant. Nous pouvons ainsi obtenir le point x_1 . Puis, on cherche $0 < \alpha < 1$ tel que $J(x_0 + \alpha(x_1 - x_0))$ soit minimum, ce qui est encore un problème de programmation linéaire (et consiste à trouver le point $x_2 = x_0 + \alpha(x_1 - x_0)$ qui appartient au segment $[x_0, x_1]$ et qui minimise J sur ce segment).

On recommence le processus en partant du point x_2 au lieu du point x_0 . La résolution est donc un processus itératif, dans lequel chaque itération revient à résoudre deux problème de programmation linéaire.

4 Méthode de Newton

Le cours de première année donnait la méthode de Newton pour résoudre une équation de type $f(x) = 0$. Cette même méthode est adaptée pour trouver l'argument qui minimise une fonction J .

Dans le cas scalaire (J est une fonction de \mathbb{R} dans \mathbb{R}), nous allons construire une suite x_k qui converge vers x^* , argument qui minimise J . Le développement limité de J autour du point x_k est donné par :

$$J(x) = J(x_k) + (x - x_k)J'(x_k) + \frac{1}{2}(x - x_k)^2 J''(x_k) + O((x - x_k)^3)$$

En négligeant le terme d'ordre 3 et en affirmant que la dérivée de l'approximation polynomiale de J est nulle en un extremum de J , on obtient :

$$J'(x_k) + (x_{k+1} - x_k)J''(x_k) = 0$$

C'est à dire :

$$x_{k+1} = x_k - \frac{J'(x_k)}{J''(x_k)}$$

Nous avons le même résultat que pour la méthode de Newton appliquée à la résolution de $J(x) = 0$, mais J a été remplacé par J' . Les inconvénients restent les mêmes... et la méthode converge effectivement vers la solution uniquement si le point de départ x_0 est bien choisi, suffisamment proche de x^* .

La méthode se généralise pour une fonction à valeurs dans \mathbb{R}^n . L'approximation J_a de J au degré 2 autour de x_k est :

$$J_a(x) = J(x_k) + \vec{\nabla} J(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k)$$

avec H le hessien de J :

$$H = [H_{ij}]_{(i,j) \in \llbracket 1..n \rrbracket^2}, \quad H_{ij} = \frac{\partial^2 J}{\partial x_i \partial x_j}$$

Notons que H est symétrique réelle. Si elle est définie positive (resp. négative), on est en présence d'un minimum (resp. maximum) de J . La suite est donc itérée ainsi :

$$x_{k+1} = x_k - H(x_k)^{-1} \vec{\nabla} J(x_k)$$

On arrêtera les itérations lorsque le gradient de J sera suffisamment petit, ou lorsqu'il variera trop peu.

Il est souvent difficile de calculer H^{-1} . Les méthodes de quasi-Newton consistent à remplacer H^{-1} par une approximation. Nous verrons une de ces approximations dans le cadre de la minimisation d'un critère quadratique.

5 Méthodes de descente

5.1 Généralités

Dans cette section, nous essayons de construire une suite x_k qui converge vers le minimum de J . La suite est calculée par :

$$x_{k+1} = x_k + \alpha_k p_k$$

où p_k est la direction de descente et α_k le pas de descente.

Il existe plusieurs méthodes de descente, selon la direction (gradient, gradient conjugué) ou le pas (pas fixe, pas optimal).

Pour une descente avec un pas optimal, le principe est de trouver :

$$\alpha_{k(opt)} = \min_{\alpha_k} f(x_{k+1})$$

En particulier, le pas optimal varie avec la suite... Un bon pas est un pas qui fait suffisamment varier J , mais qui néanmoins est suffisamment petit pour assurer la convergence. Un pas qui décroît lorsque k croît semble donc être un bon choix. Nous reviendrons sur un calcul de pas optimal dans le cadre de l'optimisation de la fonction $J(x) = \frac{1}{2} \langle Ax | x \rangle - \langle b | x \rangle$.

Quelle que soit la direction de descente, elle va généralement dans le sens de la minimisation de J . La descente de gradient consiste à prendre $p_k = -\vec{\nabla} J(x_k)$ (avec $\alpha_k > 0$), qui est la direction de plus grande pente. La descente de gradient avec pas fixe s'appelle méthode de Richardson.

En règle générale, toute direction de descente p_k qui vérifie $\langle p_k | \vec{\nabla} J(x_k) \rangle < 0$ fonctionne (plus ou moins bien).

5.2 Cas particulier des systèmes linéaires

Cette sous-section ne s'apparente pas vraiment à de la programmation non-linéaire, mais figure ici comme cas particulier des méthodes de descente.

On souhaite résoudre le système $Ax = b$ où A est symétrique et définie positive. Soit $J(x) = \frac{1}{2} \langle Ax | x \rangle - \langle b | x \rangle$. La solution \bar{x} de $Ax = b$ est aussi la valeur qui minimise $J(x)$.

Dans ce cas particulier, nous pouvons calculer le α_k optimal pour un p_k donné. C'est celui qui donnera la valeur minimum de $J(x_{k+1})$:

$$J(x_{k+1}) = J(x_k + \alpha_k p_k) = J(x_k) + \frac{\alpha_k^2}{2} \langle A p_k | p_k \rangle + \alpha_k \langle Ax_k - b | p_k \rangle$$

Nous avons un trinôme de degré 2 en α_k à coefficient dominant positif (puisque A est définie positive). Son minimum est atteint pour :

$$\alpha_k = \frac{\langle b - Ax_k | p_k \rangle}{\langle Ap_k | p_k \rangle}$$

De même, la direction de plus grande pente est simplement ici : $p_k = -\vec{\nabla} J(x) = b - Ax$ Voici l'algorithme de descente du gradient à paramètre local optimal (rappelons que A doit être symétrique et définie positive). On suppose que les opérations de calcul matriciel ont été implantées :

gradient(A : tableau de réels $n \times n$, b : tableau de réels de taille n , n : entier , ϵ : réel)

```

     $x_k, p_k$  : tableaux de réels
     $\alpha_k$  : réel
     $x_k \leftarrow$  choix «pas trop éloigné» de  $\bar{x}$ 
     $p_k \leftarrow b - A \times x_k$ 
    répéter tant que  $\|p_k\|^2 > \epsilon$ 
    |    $\alpha_k \leftarrow \|p_k\|^2$ 
    |    $\alpha_k \leftarrow \alpha_k / \langle Ap_k | p_k \rangle$ 
    |    $p_k \leftarrow b - A \times x_k$ 

```

On peut montrer [17] que la méthode du gradient avec α_k optimal converge toujours vers la solution du système linéaire. En revanche, la méthode de Richardson (pas fixe) converge si le paramètre α choisi vérifie :

$$0 < \alpha < 2 \frac{\mathcal{I}}{\mathcal{S}}$$

avec :

$$\mathcal{I} = \inf_{y \in \mathbb{R}_*^n} \frac{\langle y | Ay \rangle}{\langle y | y \rangle}$$

$$\mathcal{S} = \sup_{y \in \mathbb{R}_*^n} \frac{\langle y | Ay \rangle}{\langle y | y \rangle}$$

6 Minimisation d'un critère quadratique

6.1 Approximation polynomiale

Disposant d'un ensemble de $n + 1$ mesures $(x_i, y_i)_{i \in \llbracket 0..n \rrbracket}$, on recherche une fonction g qui minimise l'erreur $J = \sum_{i=0}^{i=n} (y_i - g(x_i))^2$.

Soit parce qu'on souhaite retrouver un modèle particulier, soit parce qu'on observe une régularité dans les mesures (x_i, y_i) , on fait généralement des suppositions sur la forme de g : fonction affine (on fait alors de la régression linéaire), polynôme de degré k ,...

Dans tous les cas, une fois la «forme» de g choisie, on laisse naturellement certains paramètres libres (des nombres réels), et on cherche donc à déterminer la fonction $g(x; a_0, a_1, \dots, a_k)$ où a_0, a_1, \dots, a_k sont ces paramètres.

Il faut donc minimiser l'expression suivante :

$$S = \sum_{i=0}^{i=n} (g(x_i; a_0, \dots, a_k) - y_i)^2$$

C'est à dire qu'il faut avoir :

$$\left\{ \begin{array}{l} \frac{\partial S}{\partial a_0} = 0 \\ \frac{\partial S}{\partial a_1} = 0 \\ \dots \dots \dots \\ \frac{\partial S}{\partial a_k} = 0 \end{array} \right. \tag{7.1}$$

L'optimisation de l'erreur quadratique consiste à trouver le vecteur x tel que $\|Ax - y\|^2$ soit minimal. On peut montrer qu'un tel vecteur x est solution de l'équation $A^T Ax = A^T y$, qui est un système linéaire de n équations à n inconnues. Nous pouvons donc appliquer les méthodes du chapitre 2 pour résoudre ce système. Notons de plus que la matrice $A^T A$ étant symétrique, on pourra choisir parmi ces méthodes celles qui sont adaptées aux matrices symétriques.

6.3 Méthode de Gauss Newton

Nous nous intéressons à la minimisation de $J(x) = \frac{1}{2} \sum_{i=1}^{i=n} f_i(x)^2$. Si nous reprenons la méthode de Newton dans ce cas particulier, nous obtenons :

$$\vec{\nabla} J(x) = \sum_{i=1}^{i=n} \vec{\nabla} f_i(x) f_i(x)$$

et

$$H_J(x) = \sum_{i=1}^{i=n} \vec{\nabla} f_i(x) \vec{\nabla} f_i(x)^T + \sum_{i=1}^{i=n} f_i(x) H_{f_i}(x)$$

Nous faisons alors l'approximation suivante, à proximité de l'optimum :

$$\widetilde{H}_J(x) = \sum_{i=1}^{i=n} \vec{\nabla} f_i(x) f_i(x)^T \approx H_J(x)$$

La méthode de Gauss Newton consiste à itérer la suite :

$$x_{k+1} = x_k - \left[\widetilde{H}_J(x_k) \right]^{-1} \vec{\nabla} J(x_k)$$

6.4 Méthode de Levenberg-Marquardt

La méthode de Levenberg Marquardt combine les avantages de la méthode de Gauss Newton et de la descente de gradient. Les itérations sont :

$$x_{k+1} = x_k - \left[\widetilde{H}_J(x_k) + \lambda I \right]^{-1} \vec{\nabla} J(x_k)$$

On remarque que lorsque λ devient très grand, la méthode converge vers la méthode de descente du gradient avec un pas de $\frac{1}{\lambda}$. Au contraire, lorsque λ est petit, la méthode converge vers celle de Gauss Newton.

Typiquement, on initialise λ à une grande valeur, puis on itère l'algorithme. À chaque itération, on regarde si l'approximation est meilleure. Si c'est le cas, la valeur de λ est diminuée (divisée par 10 par exemple), et on continue les itérations. Si ce n'est pas le cas, la valeur de λ est augmentée (multipliée par 10 par exemple), et on reprend la même itération. Ainsi, lorsqu'on s'approche de la solution, on tend vers la méthode de Gauss Newton.

7 Méthode de Lagrange*

Nous traitons ici les problèmes d'optimisation sous contraintes : trouver $x \in \omega \subset \mathbb{R}^n$ tel que $f(x) \in \mathbb{R}$ soit minimum, l'argument x devant vérifier la contrainte $\phi(x) = \vec{0}$ où ϕ est une fonction de \mathbb{R}^n dans \mathbb{R}^m . On notera $\phi_i(x)$ la composante i du vecteur $\phi(x)$.

La fonction de Lagrange, ou Lagrangien, associé au problème précédent est :

$$L(x, \lambda) = f(x) + \lambda^T \phi(x)$$

avec $\lambda \in \mathbb{R}^m$. Les composantes du vecteur λ , que nous noterons λ_i sont appelés les multiplicateurs de Lagrange.

On peut montrer que si x^* est un optimum de la fonction f vérifiant les contraintes, alors, si f et ϕ sont différentiables, x^* vérifie aussi :

$$\vec{\nabla} L = \vec{0}$$

C'est à dire que les dérivées de L par rapport à chaque composante sont nulles. La méthode des multiplicateurs de Lagrange consiste à rechercher les solutions de ce système et à vérifier *a posteriori* si l'on a affaire à des maxima ou

des minima. Une autre méthode pour trouver les minima consiste à calculer le Hessien H de L , en fonction de λ . Si la matrice H est définie positive, pour un λ donné, l'extremum correspondant est un minimum, sinon c'est un maximum.

Notons au passage que si la donnée d'un problème incite parfois à utiliser la méthode des multiplicateurs de Lagrange, on peut parfois supprimer les contraintes en en déduisant des expressions que l'on substituera à certaines variables, comme par exemple dans : minimiser $f(x, y, z) = xyz$ sous contrainte que $x + y + z = 1$.

L'extension de la méthode des multiplicateurs de Lagrange au cas des contraintes d'inégalité est due à Kuhn et Tucker et ne sera pas abordée ici.

8 Algorithmes génétiques*

Les algorithmes génétiques sont issus du monde informatique plutôt que mathématique et résultent des principes de sélection naturelle qui régissent *a priori* notre écosystème. Ils sont généralement utilisés lorsqu'aucune autre méthode ne marche.

Le problème ici est d'optimiser un critère quelconque, dépendant de paramètres quelconques. Le critère doit être facile (au sens de la complexité informatique) à calculer en fonction des paramètres. Nous supposons dans la suite que nous cherchons à maximiser le critère.

Le principe est d'entretenir une population d'individus, chacun étant défini par un génotype (ensemble de paramètres internes). Connaissant le génotype d'un individu, on doit pouvoir en déduire son phénotype, qui est la manifestation de son génotype. Souvent, en pratique, phénotype et génotype ne font qu'un. C'est le phénotype qui donne les paramètres qui permettent de calculer le critère à optimiser. Cette fonction est appelée *fonction d'adaptation*.

Les individus ont une certaine durée de vie, peuvent se reproduire, et des mutations peuvent intervenir lors de la reproduction. La reproduction sexuée consiste à créer un nouvel individu à partir de deux individus préexistants. Si un seul individu peut donner un nouvel individu, la reproduction est dite asexuée. Le génotype de la progéniture est un mélange (cross-over) des génotypes de parents, auquel on a apporté des modifications aléatoires et peu probables (mutations).

Le principe des algorithmes génétiques en optimisation est de favoriser la reproduction des individus dont la fonction d'adaptation est la plus élevée (en vertu des principes de sélection naturelle). À chaque génération, une partie de la population est sélectionnée. Un individu particulier est sélectionné de façon d'autant plus probable que sa fonction d'adaptation est élevée. Deux à deux, les individus sélectionnés se reproduisent, donnant éventuellement des mutants. Le but est ainsi de faire converger la population vers une population uniforme, dont le phénotype donnera les paramètres qui maximisent le critère.

Il est à noter qu'un algorithme génétique fonctionne bien si, en particulier, la fonction de croisement entre deux individus *a un sens*. C'est à dire si le mélange de deux individus ayant une forte adaptation donne le plus souvent un nouvel individu qui a aussi une forte adaptation.

Le réglage des nombreux paramètres (durée de vie, taux de croissance, taux de mutation, importance des mutations) et le choix des nombreuses fonctions (croisement, mutation) sont laissés à la discrétion du programmeur car ils dépendent fortement du problème particulier posé.

9 Quelques mots sur les LMI*

LMI est l'acronyme de *Linear Matrix Inequality*. C'est une méthode relativement récente utilisée en optimisation. Nous ne ferons qu'en évoquer le principe.

Nous noterons dans la suite > 0 au sujet d'une matrice A pour signifier «définie positive», c'est à dire telle que $\forall x \in \mathbb{R}_*^n, x^T A x > 0$

Une LMI est une «inégalité» de la forme :

$$F(x) = F_0 + x_1 F_1 + \dots + x_m F_m > 0$$

où les F_i sont des matrices de $\mathbb{R}^{n \times n}$ *symétriques* et $x = (x_1, \dots, x_m)$ est un vecteur de \mathbb{R}^m inconnu.

Une LMI permet de contraindre le vecteur x à un ensemble convexe. De nombreux problèmes de restriction à un ensemble convexe peuvent effectivement se mettre sous la forme d'une LMI.

Il existe essentiellement deux techniques de résolution des LMI, que nous n'aborderons pas ici, la méthode des ellipsoïdes, et la méthode des points intérieurs.

Chapitre 8

Programmation linéaire*



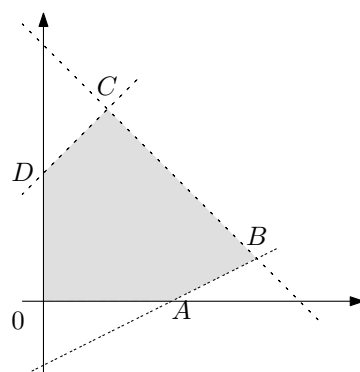
LA PROGRAMMATION linéaire est le problème de la recherche d'un extremum d'une fonction linéaire de plusieurs variables, ces variables devant en outre satisfaire un système d'équations ou d'inéquations linéaires. Par exemple, le problème qui consiste à maximiser $f(x_1, x_2) = 3x_1 + x_2$ sous contrainte que $x_1 + x_2 \geq 0$, $x_1 > 0$ et $2x_1 + x_2 - 10 < 0$ est un problème de programmation linéaire.

1 Représentation géométrique

Dans un premier temps, on se propose de résoudre le système suivant, de façon géométrique : Maximiser $f(x_1, x_2) = 2x_1 + 3x_2$ sous contrainte que $x_2 - x_1 \leq 2$, $x_1 + x_2 \leq 4$, $2x_2 - x_1 \geq -2$, avec $x_1 \geq 0$ et $x_2 \geq 0$.

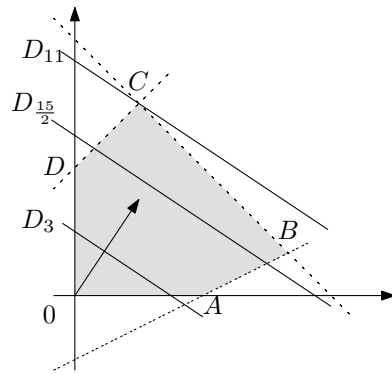
1.1 Représentation des contraintes

Les contraintes sont ici toutes des inégalités. Chaque contrainte correspond à un demi-plan. Voici la représentation graphique de l'ensemble des contraintes. La partie du plan grisée correspond au polygone qui contient les *solutions possibles* (c'est l'intersection des cinq demi-plan définis par les contraintes) :



1.2 Recherche graphique d'une solution

Il reste donc à maximiser f , les valeurs x_1 et x_2 étant contraintes à la zone hachurée. La famille de droites D_a d'équation $f(x_1, x_2) = a$ sont des droites parallèles. Trois de ces droites, D_3 , $D_{\frac{15}{2}}$ et D_{11} sont représentées sur le schéma suivant.

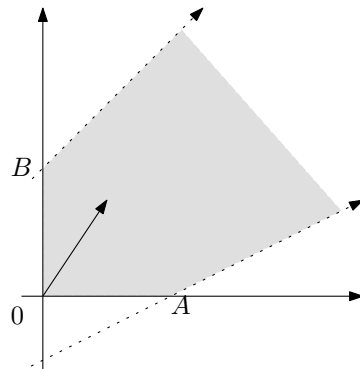


On se rend compte que parmi les droites D_a qui ont au moins un point commun avec la zone hachurée, celle pour laquelle a est maximum est D_{11} . La valeur maximum atteinte par f est donc 11, la solution de notre problème est le point $\{1, 3\}$, qui maximise f tout en respectant les contraintes.

1.3 Introduction à l'algorithme du simplexe

Si la zone des solutions possibles est bornée, le polygone formé par l'intersection des contraintes est convexe, comme intersection de demi-plans. En conséquence, la partie du polygone qui maximise f est soit un sommet, soit une arête, dans le cas où une des arêtes est orthogonale aux droites représentatives de f .

Si la zone des solutions possibles n'est pas bornée. Il se peut qu'il n'y ait pas de solution au problème dans le sens où f n'a pas de valeur maximale comme dans l'exemple qui suit (le vecteur normal aux droites représentatives de f est représenté), identique au précédent sans la contrainte $x_2 + x_1 \leq 4$:



Enfin, les contraintes peuvent être contradictoires, de telle sorte qu'il n'y ait pas de solution du tout, comme dans l'exemple suivant : Maximiser $f(x_1, x_2) = 2x_1 + 3x_2$ avec $x_1 \geq 0$, $x_2 \geq 0$, $x_1 + x_2 \leq 4$, $x_2 - x_1 \leq 5$.

Nous nous intéressons aux cas d'une région bornée non vide. Et nous avons vu que dans ce cas, il existe au moins un sommet du polygone qui est extrémal. L'algorithme du simplexe recherche donc la solution sur les sommets. Le principe, partant d'un sommet de départ (par exemple O) et de passer de sommets en sommets, en améliorant toujours la valeur de f jusqu'à ce qu'on ne puisse plus changer de sommet sans détériorer la valeur de f (ceci fonctionne parce que le polygone est convexe). Sur l'exemple précédent, l'algorithme du simplexe aurait parcouru les sommets O , puis D , puis C . Étant en C , passer en B détériore la valeur de f . Le point C est donc la solution.

2 Mise sous forme standard

Un problème de programmation linéaire est *sous forme standard* s'il consiste à maximiser une fonction, si toutes les variables doivent être positives ou nulles, et si les autres contraintes sont des contraintes d'égalité.

L'algorithme du simplexe s'applique à des problèmes mis sous forme standard. La première étape est donc de les mettre sous cette forme.

Soit f la fonction à optimiser. Chercher un minimum de f est équivalent à chercher un maximum de $-f$. Il suffit donc de changer le signe de f pour se retrouver dans le cas de la recherche d'un maximum.

2.1 Contraintes de positivité

Si une variable n'est pas nécessairement positive dans l'énoncé de départ, on peut la remplacer par la différence entre deux nouvelles variables, celles-ci devant nécessairement être positives ou nulles.

2.2 Contraintes d'inégalité

Il reste à transformer les contraintes d'inégalité (par exemple $x_1 - 3x_2 \leq 4$) en contraintes d'égalité. Pour cela, on introduit des *variables d'écart*. De façon générale, la contrainte $c_1x_1 + \dots + c_kx_k \geq c_0$ est transformée en $c_1x_1 + \dots + c_kx_k - x_{k+1} = c_0$ et $x_{k+1} \geq 0$. De même, la contrainte $c_1x_1 + \dots + c_kx_k \leq c_0$ est transformée en $c_1x_1 + \dots + c_kx_k + x_{k+1} = c_0$ et $x_{k+1} \geq 0$. Dans notre exemple précédent, nous obtiendrions donc : $x_1 - 3x_2 + x_3 = 4$ et $x_3 \geq 0$.

2.3 Exemple de mise sous forme standard

Reprenons le système du début de chapitre : Maximiser $f(x_1, x_2) = 2x_1 + 3x_2$ sous contrainte que $x_2 - x_1 \leq 2$, $x_1 + x_2 \leq 4$, $2x_2 - x_1 \geq -2$, avec $x_1 \geq 0$ et $x_2 \geq 0$. Après mise sous forme standard, nous obtenons :

- fonction à maximiser : $f(x_1, x_2) = 2x_1 + 3x_2$
- contraintes d'égalité

$$\begin{cases} -x_1 + x_2 + x_3 = 2 \\ x_1 + x_2 + x_4 = 4 \\ -x_1 + 2x_2 - x_5 = -2 \end{cases}$$

- (x_1, x_2, x_3, x_4 et x_5) positifs ou nuls

3 Algorithme du simplexe

Dans cette section, nous allons voir une première méthode pour résoudre un problème de programmation linéaire.

En premier lieu, le problème doit être mis sous *forme standard*. Il ne doit alors plus contenir comme contrainte que des égalités, ainsi que le caractère positif ou nul de toutes les variables. Les équations de départ doivent donc être modifiées pour être mises sous cette forme.

Dans un second temps, remarquons qu'aux sommets du polygone représentant les contraintes, certaines variables sont nulles et d'autres non. En effet, sur chaque segment correspondant à une inégalité, la variable d'écart est nulle. À l'intersection de deux segments, il y a donc deux variables nulles. Dans la suite, les variables nulles sur un sommet seront appelées les variables *hors base* de ce sommet. Les autres seront les variables *de base* pour ce sommet.

L'idée de l'algorithme du simplexe est d'exprimer le système d'équations de façon à obtenir une expression des variables de base en fonction des variables hors base. La valeur des variables de base est donc très simple à calculer puisque les variables hors base sont nulles. De la même façon, en chaque sommet, f sera exprimé en fonction des variables hors-base.

3.1 Tableau du simplexe

Le tableau du simplexe est un tableau qui récapitule le système d'équation en un certain sommet. Au point O (nous verrons par la suite quoi faire si O n'est pas un sommet du polygone), les variables x_1 et x_2 sont nulles. Les variables de base sont donc les variables d'écart x_3, x_4 et x_5 . Si nous recopions les équations ainsi :

x_1	x_2	x_3	x_4	x_5	b	base
-1	1	1	0	0	2	x_3
1	1	0	1	0	4	x_4
-1	2	0	0	-1	-2	x_5

Dans la colonne de droite figurent les variables de base. Chaque ligne permet d'exprimer une variable de base en fonction des variables hors-base. Par exemple la ligne 1 se lit :

$$-1 \times x_1 + 1 \times x_2 + 1 \times x_3 = 2$$

Ce qui est équivalent à :

$$x_3 = 2 + x_1 - x_2$$

Nous voyons que dans la colonne qui correspond à une variable de base, il ne peut y avoir qu'un terme non nul, qui est situé sur la même ligne que le nom de la variable dans la colonne «base».

À présent, ajoutons l'expression de f dans le tableau, que nous appellerons tableau du simplexe :

x_1	x_2	x_3	x_4	x_5	f	b	base
-1	1	1	0	0	0	2	x_3
1	1	0	1	0	0	4	x_4
-1	2	0	0	-1	0	-2	x_5
-2	-3	0	0	0	1	0	

La dernière ligne se lit :

$$-2x_1 - 3x_2 + f = 0$$

ce qui donne bien l'expression de f .

3.2 Itérations de l'algorithme

Le tableau précédent résume la situation au point O . Nous voyons que ce point n'est pas optimal car il y a des valeurs négatives sur la dernière ligne. Par conséquent, augmenter une de ces variables permettra d'augmenter f . Nous avons le choix entre augmenter la valeur de x_1 ou x_2 (toutes les deux nulles pour l'instant). Nous choisissons d'augmenter x_2 car son coefficient est plus élevé (une augmentation de 1 sur x_2 augmentera plus f qu'une augmentation de 1 sur x_1). En augmentant la valeur de x_2 , celle-ci ne sera plus nulle. Par conséquent x_2 deviendra une variable de base. Si nous augmentons x_2 jusqu'à un sommet du polygone, nous savons qu'une variable deviendra nulle (et deviendra hors-base). Cette variable est la première variable qui s'annulera si on augmente la valeur de x_2 .

Une variable de base x_i s'exprime à partir du tableau du simplexe par :

$$c_{i1}x_1 + c_{i2}x_2 + c_{ii}x_i + b_i = 0$$

Par conséquent :

$$x_i = \frac{-1}{c_{ii}}(c_{i1}x_1 + c_{i2}x_2 + b_i)$$

Or, la variable x_1 est nulle :

$$x_i = \frac{-1}{c_{ii}}(c_{i2}x_2 + b_i)$$

La variable x_i s'annule donc en :

$$x_2 = \frac{-b_i}{c_{i2}}$$

En divisant la colonne b par la colonne x_2 du tableau, nous obtenons :

2
4
-1

La variable x_3 s'annule donc pour $x_2 = 2$, la variable x_4 pour $x_2 = 4$ et la variable x_5 pour $x_2 = -1$. En conséquence, c'est la variable x_3 qui s'annule la première. Et finalement, nous devons recalculer le tableau du simplexe pour un nouveau sommet pour lequel x_2 sera une variable de base et x_3 sera hors-base.

Cette opération est assez simple et s'apparente à la méthode du pivot de Gauss. Dans la ligne correspondant à x_3 , nous divisons tous les nombres par un coefficient de façon à avoir un 1 dans la colonne correspondant à x_2 , puis par combinaison linéaire de cette nouvelle ligne avec les autres, nous annulons tous les coefficients de la colonne x_2 :

x_1	x_2	x_3	x_4	x_5	f	b	base
-1	1	1	0	0	0	2	x_2
2	0	-1	1	0	0	2	x_4
1	0	-2	0	-1	0	-6	x_5
-5	0	3	0	0	1	6	

Cette opération revient tout simplement à exprimer de nouveau les variables hors-base et f en fonction des variables de base.

Nous devons à présent nous trouver sur un nouveau sommet du polygone. En effet les variables nulles sont x_1 et x_3 . En conséquence : $x_2 = 2$, $x_4 = 2$, $x_5 = 6$. Nous nous trouvons au point $(0, 2)$, c'est à dire au point D .

De nouveau, une augmentation de f est possible, en augmentant la valeur de x_1 . La variable x_1 devient donc une variable de base. Divisons la colonne b par la colonne x_1 :

-2
1
-6

La variable hors base qui s'annule pour la plus petite valeur positive de x_1 est donc x_4 , qui devient hors base. On calcule le nouveau tableau du simplexe :

x_1	x_2	x_3	x_4	x_5	f	b	base
0	1	$-\frac{1}{2}$	$\frac{1}{2}$	0	0	3	x_2
1	0	$-\frac{1}{2}$	$\frac{1}{2}$	0	0	1	x_1
0	0	$-\frac{3}{2}$	$-\frac{1}{2}$	-1	0	-7	x_5
0	0	$\frac{1}{2}$	$\frac{5}{2}$	0	1	11	

Dans le nouveau tableau, nous avons une expression de f en fonction de x_3 et x_4 , et nous constatons que f est maximale pour $x_3 = 0$ et $x_4 = 0$ puisque tous les coefficients de la dernière ligne sont positifs. Par conséquent, l'algorithme est terminé. Le tableau nous permet d'obtenir la valeur de f et des autres variables en ce point : x_3 et x_4 sont des variables de base donc sont nulles, et par suite, $x_1 = 1$, $x_2 = 3$, $x_5 = 7$ et $f = 11$.

Le maximum de la fonction f est 11, et il est atteint en $x_1 = 1$ et $x_2 = 3$ c'est à dire au point C .

3.3 Extensions à plus de 2 dimensions

Naturellement, l'algorithme du simplexe n'est utilisé en pratique que si le système est composé de milliers de variables. Dans ce cas, on ne dispose pas de la représentation graphique et les itérations du tableau du simplexe sont purement mécaniques, et ne présentent aucune difficulté supplémentaire par rapport à l'exemple précédent.

3.4 Choix du point de départ de l'algorithme

Nous venons de voir l'algorithme du simplexe dans le cas où l'origine est un point réalisable (i.e. satisfait les contraintes). Déterminer un point réalisable si l'origine n'en est pas un nécessite la résolution d'un autre problème linéaire appelé problème auxiliaire.

Un tel cas survient, par exemple, avec la contrainte (qui exclut l'origine du polygone des solutions possibles) :

$$3x_1 + 2x_2 \geq 2$$

L'introduction de la variable d'écart x_3 donne :

$$3x_1 + 2x_2 - x_3 = 2$$

Dans ce cas, le point $x_1 = x_2 = 0$ n'est pas réalisable (nous pouvions le voir avant d'avoir introduit la variable d'écart) puisque x_3 doit être positif¹. Dans un tel cas, il faut ajouter une variable auxiliaire à la contrainte, qui devient :

$$3x_1 + 2x_2 - x_3 + w_1 = 2$$

Si k contraintes sont concernées par ce problème, alors, il faudra ajouter k variables auxiliaires $w_1 \dots w_k$.

Le nouveau problème comporte k variables supplémentaires et on lui trouve facilement une solution réalisable : $x_1 = x_2 = x_3 = 0$ dans notre exemple. Il reste donc à résoudre ce nouveau problème, en maximisant un nouvel objectif : $g = -\sum_{i=1}^k w_i$. Les w_i étant positifs, g est négatif. On maximise donc l'objectif dans l'espoir d'obtenir un point tel que $g = 0$. Si un tel point existe, les variables hors base donneront les coordonnées d'un point réalisable pour le problème de départ qu'on pourra résoudre normalement. Si on n'obtient pas $g = 0$, c'est que le problème de départ n'a pas de solution réalisable, et, en quelque sorte, le problème est aussi résolu.

¹De façon systématique, s'il y a une seule variable d'écart et que son signe est opposé à celui du second membre, alors la contrainte n'est pas réalisable.

Annexe A

Givens et Householder

1 Matrices de Givens

La matrice de Givens G_θ^{pq} est la matrice de rotation d'angle θ dans le plan des vecteurs de base (e_p, e_q) :

$$G_\theta^{pq} = \begin{pmatrix} 1 & 0 & \cdots & 0 & \cdots & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & & \vdots & & \vdots & & & \vdots \\ \vdots & & & \vdots & & \vdots & & & \vdots \\ 0 & \cdots & \cdots & \cos \theta & \cdots & \sin \theta & \cdots & \cdots & 0 \\ \vdots & & & \vdots & & \vdots & & & \vdots \\ 0 & \cdots & \cdots & -\sin \theta & \cdots & \cos \theta & \cdots & \cdots & 0 \\ \vdots & & & \vdots & & \vdots & & & \vdots \\ \vdots & & & \vdots & & \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 & \cdots & 0 & \cdots & 0 & 1 \end{pmatrix} \begin{matrix} p \\ q \end{matrix}$$

Cette matrice est utilisée pour annuler, par produit, certains coefficients d'une matrice symétrique. Considérons le produit $G_\theta^{pqT} A G_\theta^{pq}$: la multiplication par G ou sa transposée ne modifie que les colonnes et les lignes p et q .

Si nous notons $C = (c_{ij}) = G^T A G$, on obtient :

$$\begin{aligned} \text{si } i \notin \{p, q\} \text{ et } j \notin \{p, q\}, & \quad c_{ij} = a_{ij} \\ \text{si } j \notin \{p, q\} & \quad c_{pj} = \cos \theta a_{pj} - \sin \theta a_{qj} \\ & \quad c_{qj} = \sin \theta a_{pj} + \cos \theta a_{qj} \\ \text{si } j \in \{p, q\} & \quad c_{pq} = \cos \theta \sin \theta (a_{pp} - a_{qq}) + (\cos^2 \theta - \sin^2 \theta) a_{pq} \\ & \quad c_{pp} = \cos^2 \theta a_{pp} + \sin^2 \theta a_{qq} - 2 \cos \theta \sin \theta a_{pq} \\ & \quad c_{qq} = \sin^2 \theta a_{pp} + \cos^2 \theta a_{qq} + 2 \cos \theta \sin \theta a_{pq} \end{aligned}$$

Comment choisir l'angle θ tel que c_{pq} (et incidemment c_{qp}), avec $p < q$, soit nul ?

$$\begin{aligned} \cos \theta \sin \theta (a_{pp} - a_{qq}) + (\cos^2 \theta - \sin^2 \theta) a_{pq} &= 0 \\ -\sin 2\theta (a_{pp} - a_{qq}) &= 2 \cos 2\theta a_{pq} \end{aligned}$$

En conséquence, si $a_{pp} = a_{qq}$, $\theta = \frac{\pi}{4}$ et sinon, $\tan 2\theta = \frac{2a_{pq}}{a_{qq} - a_{pp}}$. Il est possible de calculer tous les coefficients de C à partir de la valeur de $\tan 2\theta$ ainsi obtenue.

L'algorithme de calcul devient alors :

- calculer $\mu = \frac{a_{qq} - a_{pp}}{2a_{pq}}$, ce qui est possible si a_{pq} est non nul (fait par ailleurs probable puisqu'on souhaite annuler a_{pq}) ;
- si $\mu = 0$, $t \leftarrow 1$, sinon, $t \leftarrow -\mu + \operatorname{sgn}(\mu) \sqrt{1 + \mu^2}$ (t vaut maintenant $\tan \theta$)¹ ;
- $c \leftarrow \frac{1}{\sqrt{1+t^2}}$ (c vaut maintenant $\cos \theta$)
- $s \leftarrow ct$ (s vaut maintenant $\sin \theta$)

On peut réécrire les relations donnant les coefficients de C à partir de ceux de A uniquement en utilisant c , s et t .

¹Penser que $\tan 2\theta = \frac{2 \tan \theta}{1 - \tan^2 \theta}$.

2 Matrices de Householder

2.1 Construction d’une matrice de Householder

Nous définirons la matrice carrée de Householder d’ordre n , $H_n(\beta, v)$ (avec $\beta \in \mathbb{R}$ et $v \in \mathbb{R}^n$), par :

$$H_n(\beta, v) = I_n - \frac{1}{\beta}vv^T$$

Nous devons déterminer la matrice H_n , de telle sorte que $H_n(\beta, v).a = \alpha e$ où a est un vecteur de taille n (a_1, \dots, a_n) non nul et e est le vecteur de taille n ($1, 0, \dots, 0$). La matrice $H_n(\beta, v)$ qui convient est définie par :

$$\begin{aligned} \alpha &= -\operatorname{sgn}(a_1)\sqrt{\sum_{i=1}^{i=n} a_i^2} \\ \beta &= \alpha^2 - \alpha a_1 \\ v &= (a_1 - \alpha, a_2, a_3, \dots, a_n) \end{aligned}$$

Le calcul de $H_n(\beta, v).b$, où b est un vecteur quelconque, s’effectue alors rapidement de la façon suivante : $\gamma = \frac{1}{\beta}v^T b$, puis $Hb = b - \gamma v$

Déterminer la matrice H se fait donc en $O(n)$, et effectuer le produit de matrice H par un vecteur quelconque est aussi en $O(n)$.

Vérifions à présent que nous avons bien $H_n(\beta, v).a = \alpha e$ avec les paramètres ci-dessus :

$$\frac{v^T a}{\beta} = \frac{[a_1(a_1 - \alpha)] + [\sum_2^n a_i^2]}{\alpha^2 - \alpha a_1} = \frac{[a_1^2 - \alpha a_1] + [\alpha^2 - a_1^2]}{\alpha^2 - \alpha a_1} = 1$$

Donc :

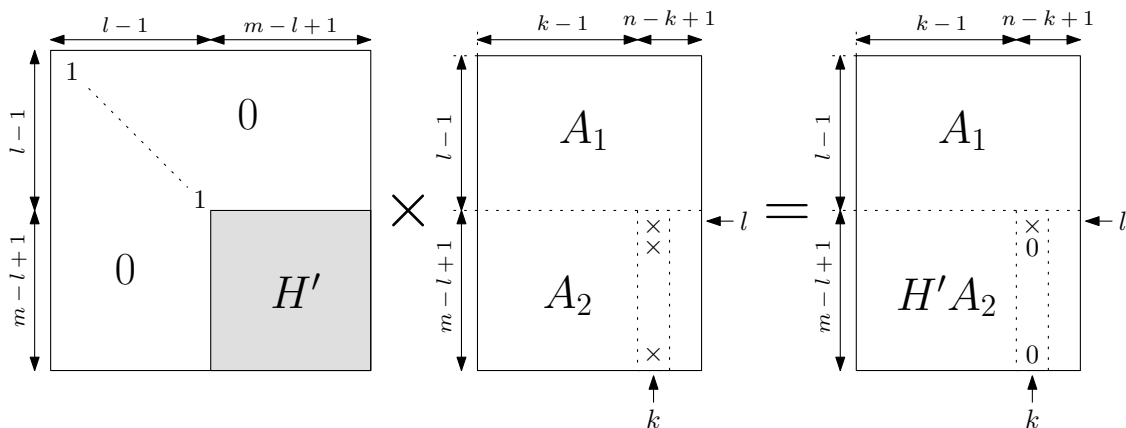
$$H.a = a - \frac{1}{\beta}vv^T a = a - v \left(\frac{1}{\beta}v^T a \right) = a - v = \alpha e$$

Notons qu’en plus d’être clairement symétriques, les matrices de Householder sont orthogonales.

2.2 Annulation d’une partie d’une colonne

Dans la suite, nous supposons que A est une matrice de $\mathbb{R}^{m \times n}$ ($m \geq n$), $k \leq n$ et $l \leq m$. Nous cherchons $H \in \mathbb{R}^m$ telle que $B = HA$ vérifie : $\forall j > l, b_{jk} = 0$

Soit $H_m(\beta, v)$ une matrice de Householder telle que v ait ses $l - 1$ premières composantes nulles. Le produit HA peut alors s’effectuer par blocs, comme indiqué sur la figure suivante. De plus, la matrice H' est encore une matrice de Householder. Le problème se ramène donc à celui de l’annulation de tous les coefficients, sauf le premier, d’une colonne de la matrice A_2 .



Ce sous problème est immédiat avec les définitions que nous avons données plus avant. Il suffit de choisir :

$$\begin{cases} \alpha = -\operatorname{sgn}(a_{lk})\sqrt{\sum_{i=l}^{i=m} a_{ik}^2} \\ \beta = \alpha^2 - \alpha a_{lk} \\ v = (a_{lk} - \alpha, a_{l+1,k}, \dots, a_{mk}) \end{cases}$$

pour obtenir la matrice H' et donc la matrice H .

Cette technique, qui permet de placer des 0 dans une colonne sera utilisée pour la factorisation QR (triangulation) et pour l'obtention d'une matrice de Hessenberg dans le cadre de la recherche de valeurs propres.

Bibliographie

- [1] Gene H. Golub and Charles F. Van Loan. *Matrix Computation*. Johns Hopkins University Press, 1996.
- [2] Grégoire Allaire and Sidi Mahmoud Kaber. *Algèbre linéaire numérique*. Ellipses, 2002. 512.64(ALL).
- [3] Adrian Biran and Mohe Breiner. *Matlab pour l'ingénieur*. Pearson Education, 2004.
- [4] Claude Brezinski. *Initiation à la programmation linéaire et à l'algorithme du simplexe*. Ellipses, 2002. ISBN 2-7298-1013, 519.8(BRE).
- [5] François Cottet-Emard and Pierre Goetgheluck. *Mathématiques sur ordinateur*. De Boeck Université, 1993. ISBN 280411757X, 681.3 :51(COT).
- [6] Ionut Danaila, Pascal Joly, Sidi Mahmoud Kaber, and Marie Postel. *Introduction au calcul scientifique par la pratique*. Dunod, 2005.
- [7] Daniel Euvrard. *Résolution numérique des équations aux dérivées partielles de la physique, de la mécanique et des sciences de l'ingénieur : Différences finies, éléments finis, problèmes en domaines non bornés*. Masson, 1994. ISBN : 2225845093 519.6(EUV) **Éléments finis**.
- [8] André Fortin. *Analyse numérique pour ingénieurs*. Presses internationales polytechnique, 2001. 519.6(FOR).
- [9] A. El Jai. *Éléments d'analyse numérique*. Presses Universitaires de Perpignan, 2003. ISBN 2914518455, 519.6 ELJ.
- [10] Patrick Lascaux and Raymond Théodor. *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, volume 2. Dunod, 2000. ISBN 2100053353, 519.6 LAS.
- [11] Patrick Lascaux and Raymond Théodor. *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, volume 1. Dunod, 2000. ISBN 2100053345, 519.6 LAS.
- [12] Brigitte Lucquin and Olivier Pironneau. *Introduction au calcul scientifique*. Masson, 1996. 517.9(LUC) **Éléments finis**.
- [13] Serge Nicaise. *Analyse numérique et équations aux dérivées partielles*. Dunod, 2000. 519.6(NIC) **Équations aux dérivées partielles, éléments finis**.
- [14] Jean-Pierre Nougier. *Méthodes de calcul numérique*. Masson, 1987. 519.6(NOU).
- [15] A. Le Pourhiet. *Résolution numérique des équations aux dérivées partielles : une première approche*. Cepadues, 1988. ISBN : 2854281756 517.7(LEP) **Équations aux dérivées partielles**.
- [16] Jean-Etienne Rombaldi. *Algorithmique numérique et Ada*. Masson, 1994. ISBN 2225843848, 681.3.06(AD) ROM.
- [17] Lionel Sainsaulieu. *Calcul scientifique : cours et exercices corrigés*. Dunod, 1999. 510(SAI) **Éléments finis**.
- [18] Jacques Teghem. *Programmation linéaire*. Ellipses, 1996. 519.852(TEG) **Programmation linéaire**.
- [19] Éric Jacquet-Lagrèze. *Programmation linéaire*. Economica, 1998. 681.3.01(JAC) **Programmation linéaire**.

Index

Adams, 23
Adams-Bashforth, 23
Adams-Moulton, 24

Choleski, 6

Dufort et Frankel, 31, 32

factorisation
 LU, 6
 QR, 7

Gauss-Jordan, 13
Gauss-Seidel, 7

Hessien, 45

Jacobi, 7, 8

Kuhn, 45

Lagrange, 44
Lagrangien, 44
LMI, 45

matrice
 déterminant, 12
 multiplication, 11

Newton, 40

Richardson, 41
Rutishauser, 13

Strassen, 12
système
 homogène, 9
 linéaire, 5
 surdéterminé, 43

Tucker, 45

TD IV : Équations différentielles

Question 1 – Méthodes d'Euler

On s'intéresse à un problème de la forme :

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

1-1 – Écrivez une fonction MATLAB qui résout cette équation différentielle en utilisant la méthode d'Euler (explicite). La fonction prendra en paramètres f , t_0 , y_0 , t_{max} et le pas de calcul. On supposera que t_0 est une des deux bornes de l'intervalle de calcul (l'autre étant t_{max}). On gèrera les deux cas : $t_0 < t_{max}$ et $t_{max} < t_0$.

1-2 – La méthode d'Euler implicite est obtenue à partir du développement de Taylor de $y((t+h) - h)$. Donnez l'expression de y_i obtenue par cette méthode. Écrivez une fonction MATLAB qui l'implante.

Question 2 – Méthode de Runge-Kutta d'ordre 2

2-1 – Écrivez une fonction MATLAB qui résout une équation différentielle en utilisant la méthode de Runge-Kutta d'ordre 2 (variante du point milieu). Le problème aura la même forme que précédemment. La fonction prendra en paramètres f , t_0 , y_0 , t_{max} et le pas de calcul.

2-2 – Proposez une implantation du même algorithme en C.

Question 3 – Méthode d'Adams

On se propose ici d'écrire une fonction MATLAB qui résout une équation différentielle du même type que les précédentes en utilisant les formules d'Adams ouvertes à l'ordre 3. Comment initialiser le processus ?

Question 4 – Conditions aux limites

Nous traitons le cas suivant :

$$y''(x) = A(x)y'(x) + B(x)y(x) + C(x) \quad A(x), B(x), C(x) \text{ sont des polynômes en } x$$

avec pour conditions aux limites :

$$\begin{cases} y(a) = y_a \\ y(b) = y_b \end{cases}$$

Nous supposons avoir résolu deux fois cette équation (numériquement) avec pour conditions aux limites :

$$\begin{cases} y(a) = y_a \\ y'(a) = 0 \end{cases}$$

puis

$$\begin{cases} y(a) = y_a \\ y'(a) = 1 \end{cases}$$

Les solutions respectives sont $y_0(x)$ et $y_1(x)$

Déduisez-en la solution numérique de l'équation de départ.

Question 5 – Erreur de troncature

Évaluez l'erreur de troncature par pas de la méthode d'Euler explicite.

TD V : Équations aux dérivées partielles

Question 1 – Équation de la diffusion

L'équation de la diffusion est :

$$\frac{\partial u}{\partial t} = \sigma \frac{\partial^2 u}{\partial x^2}$$

1-1 – Retrouvez le schéma de Dufort et Frankel :

$$\frac{U_{m,n+1} - U_{m,n-1}}{2\Delta t} - \sigma \frac{U_{m-1,n} - U_{m,n+1} - U_{m,n-1} + U_{m+1,n}}{\Delta x^2} = 0$$

Étudiez-en la consistance et la stabilité.

1-2 – Étudiez la consistance et la stabilité du schéma implicite suivant :

$$U_{m,n+1} = U_{m,n} + r(U_{m-1,n+1} - 2U_{m,n+1} + U_{m+1,n+1})$$

Question 2 – Équation des ondes

L'équation des ondes est de la forme :

$$\frac{\partial^2 u}{\partial t^2} - c^2 \frac{\partial^2 u}{\partial x^2} = 0$$

2-1 – Étudiez la consistance et la stabilité du schéma (avec $r = \frac{c^2 \Delta t^2}{\Delta x^2}$) :

$$U_{m,n+1} = 2(1-r)U_{m,n} + r(U_{m-1,n} + U_{m+1,n}) - U_{m,n-1}$$

2-2 – Proposez un schéma de discrétisation implicite faisant intervenir trois niveaux de temps.

2-3 – Étudiez-en la consistance et la stabilité.

Question 3 – Équation de Laplace

On se propose ici de résoudre l'équation de Laplace sur une grille $N \times N$. Les valeurs de u sont imposées au centre de la grille (u_0) et sur son pourtour (u_e).

3-1 – Donnez le schéma discret de résolution.

3-2 – Proposez un algorithme de résolution.

TD VI : Méthode des moindres carrés*

Disposant d'un ensemble de $n+1$ mesures $(x_i, y_i)_{i \in \llbracket 0..n \rrbracket}$, on recherche une fonction g qui minimise l'erreur $f(x) = \sum_{i=0}^n (y_i - g(x_i))^2$.

Nous allons exprimer g comme combinaison linéaire de polynômes orthogonaux $T_l(x)$ définis sur $[-1, 1]$.

Question 1 – Erreur au sens des moindres carrés

En supposant que T_l est de degré l , écrivez l'erreur (au sens des moindres carrés) que l'on commet en approchant f par un polynôme de degré l exprimé dans la base des T_j .

Question 2 – Système linéaire à résoudre

2-1 – À quelles conditions un minimum de S est-il atteint ?

2-2 – Donnez ces relations sous forme matricielle.

Question 3 – Polynômes de Tchebychev

Les polynômes de Tchebychev vérifient :

$$\forall (n, \theta) \in \mathbb{N} \times [0, \pi], T_n(\cos(\theta)) = \cos(n\theta)$$

3-1 – Vérifiez que les T_l forment une famille de polynômes orthogonaux pour la fonction poids : $p(x) = (1-x^2)^{\frac{1}{2}}$. On rappelle qu'une famille de polynômes T_l définis sur $[-1, 1]$ est orthogonale pour le poids $p(x)$ si :

$$m \neq n \Rightarrow \int_{-1}^1 T_m(x) T_n(x) p(x) dx = 0$$

3-2 – Supposons que les points de mesure soient :

$$\forall i \in \llbracket 0..n \rrbracket, x_i = \cos\left(\frac{2i+1}{2n+2}\pi\right)$$

Montrez que dans ce cas :

$$\sum_{i=0}^{i=n} T_j(x_i) T_k(x_i) = \frac{n+1}{2} \delta_{jk} (1 + \delta_{j0})$$

où $\delta_{jk} = 1$ si $k = j$ et $\delta_{jk} = 0$ sinon.

3-3 – Utilisez ce résultat pour calculer les coordonnées de g dans la base formée des polynômes de Tchebychev.

3-4 – Pour calculer les valeurs du polynôme, on peut utiliser la relation de récurrence :

$$\begin{cases} T_0(x) = 1 \\ T_1(x) = x \\ \forall l > 1, T_l(x) = 2xT_{l-1}(x) - T_{l-2}(x) \end{cases} \quad \text{avec } -1 \leq x \leq 1$$

Démontrez cette relation.

TD VII : Méthode du gradient conjugué

On souhaite résoudre le système $Ax = b$ où A est symétrique et définie positive.

Question 1 – Problème de minimisation

1-1 – Montrer que la valeur x^* qui vérifie $Ax^* = b$ est aussi la valeur qui minimise :

$$f(x) = \frac{1}{2} \langle Ax | x \rangle - \langle b | x \rangle$$

1-2 – Ce minimum est approché en itérant la suite : $x_{k+1} = x_k + \alpha_k p_k$ où α_k est un scalaire et p_k est un vecteur (c'est la direction de descente). Calculez la valeur de α_k qui minimise x_{k+1} .

Question 2 – Direction de descente

La méthode du gradient conjugué consiste à choisir une direction de descente p_k qui vérifie : $\langle p_{k-1} | Ap_k \rangle = 0$. On cherche une telle direction de la forme : $\forall k > 0, p_k = -r_k + \beta_{k-1} p_{k-1}$ avec $r_k = Ax_k - b$ et $p_0 = -r_0$.

2-1 – Calculez β_{k-1} .

2-2 – En utilisant la valeur de α_k calculée plus haut, montrez que : $\langle p_k | r_{k+1} \rangle = 0$.

2-3 – Montrez que $\langle p_k | r_k \rangle = -\langle r_k | r_k \rangle$. Déduisez en que : $p_k = 0 \Rightarrow \|r_k\| = 0$. Quand peut-on stopper les itérations ?

2-4 – Montrez que $\langle r_{k-1} | r_k \rangle = 0$.

2-5 – Montrez que β_{k-1} peut aussi s'écrire :

$$\beta_{k-1} = \frac{\|r_k\|^2}{\|r_{k-1}\|^2}$$

Question 3 – Convergence*

Nous admettrons ici la propriété suivante :

$$\forall (i, j) / 0 \leq j < i \leq N, \langle p_i | Ap_j \rangle = 0$$

Cette propriété signifie que les directions de descente sont A -orthogonales. Nous admettrons de plus que si A est de dimension N , il est impossible d'avoir plus de N vecteurs non-nuls A -orthogonaux.

3-1 – Montrez qu'il existe $l \leq N$ tel que $p_l = 0$.

3-2 – Montrez que si $p_l = 0$, alors $x_l = x^*$.

3-3 – Que pouvez-vous en déduire sur l'algorithme du gradient conjugué ? Qu'en est-il *a priori* de la méthode numérique associée ?

Question 4 – Algorithme*

Proposez un algorithme implantant la méthode du gradient conjugué.