Algorithmique numérique : matière et méthodes

Ne contient pas d'explications sur le fonctionnement des méthodes, c'est pour avoir une vue d'ensemble des chapitres et des principales méthodes (pour choisir la bonne en fonction du problème)

Chapitre 1 : number representation

Pour : représenter des nombres sur un ordinateur

Matière/méthodes :

- **Sources d'erreurs**
- **Opérations sur flottants**

Méthodes :

Méthode	*Description*	*Contraintes/problèmes*	*Avantages*
Unsigned type	Binaire classique	- Seulement des entiers positifs	
Signed type	Binaire Two's complement	- Seulement des entiers - Plus petite range pour le même nombre de bits	- Possibilité de nombres négatifs
Fixed-point representation	Binaire avec partie entière et décimale	- Seulement des nombres positifs ? - Très demandeur en espace si on veut stocker à la fois des grands et des petits nombres - Partie décimale représentée par $1/2^k$ donc forte approximations	
Floating-point representation (standard IEEE 754*)*	Notation scientifique	- Si on obtient une infinité suite à un calcul, ça pourrait être un overflow - L'ordre des opérations est important et peut risquer un overflow si mal agencé *On peut scale down nos nombres durant les calculs pour réduire le risque d'overflow* - Certains nombres ne peuvent pas être représentés avec un significant fini *Il ne faut jamais comparer directement a = b mais plutôt |a – b| < epsilon*	- Possibilité de nombres négatifs (bit de signe)

Chapitre 2 : systems of linear equations

Pour : résoudre des systèmes d'équations linéaires

Matière :

- **Ill conditionning**
  - Une matrice est mal conditionnée si le déterminant de A est très petit
  - Possibilité de solutions très éloignées à cause d'erreurs numériques ; can't be trusted

Méthodes :

Méthode	*Description*	*Complexité*	*Cas intéressants*	*Contraintes/problèmes*	*Avantages*
Gauss-Jordan	Classique	$O(n^3/2)$	- Pour résoudre X vecteurs au même moment	- Si un pivot est proche de zéro, possibilité d'explosion d'erreur *Il faut idéalement sélectionner les pivots dont la valeur absolue est la plus grande, donc pivoter les lignes/colonnes ; faire attention car il faudra pivoter les vecteurs B aussi*	- Inverse de A facilement calculable en égalant une matrice identité (même si déconseillé car instable)
Doolittle	Décomposition LU	Décomposition en $O(n^3)$ Résolution en $O(n^2)$	- Pour résoudre X vecteurs à différents moments		- On peut stocker la décomposition en une seule matrice (même celle de départ)
Choleski	Décomposition LL^T	Décomposition en $O(n^3)$ Résolution en $O(n^2)$	- Pour résoudre une matrice symétrique	- A doit être symétrique et définie positive	- Deux fois plus rapide que Doolittle - On peut stocker la décomposition en une seule matrice (même celle de départ) - Possibilité de variantes pour cas particuliers (ex. matrice tridiagonale)
Jacobi	Méthode itérative	Itération en $O(n^2)$	- Pour de très grandes matrices, car complexité plus faible - Pour des matrices sparses (avec beaucoup de zéros) - Pour des matrices "diagonally dominant" : *Lorsque les éléments en diagonale sont plus grands (en valeur absolue) que la somme des éléments de leur ligne*	- Ne converge pas toujours - Plus lente de manière générale - Nécessite plus d'espace à priori - Parfois, deux itérations sont proches, mais celle d'après est plus proche de la solution ("convergence locale") *On peut comparer tous les deux résultats, mais alors ça pose problème si la fonction est oscillante ; bien choisir en fonction du problème*	- Autocorrection d'erreurs numériques (on finira par converger malgré les erreurs) - Facilement parallélisable
Gauss-Seidel	Variation de Jacobi	Itération en $O(n^2)$		- Parfois plus lent que Jacobi, en fonction de A - Non parralélisable	- Parfois plus rapide que Jacobi, en fonction de A - Un seul vecteur pour le stockage
SOR	Généralisation de Gauss-Seidel pour faire converger plus ou moins rapidement	Itération en $O(n^2)$		- Parfois plus lent que Jacobi, en fonction de A - Non parralélisable	- Parfois plus rapide que Jacobi, en fonction de A - Un seul vecteur pour le stockage

Chapitre 3 : interpolation et curve fitting

Pour : évaluer des données non-mesurées à partir de points mesurés

Matière :

- **Interpolation polynomiale**
  - Il existe un unique polynôme qui passe par tous les points (toutes les méthodes donneront le même polynôme)
- **Extrapolation**
  - Aller au-delà que la limite de nos mesures
  - Généralement une mauvaise idée car notre approximation continuera dans sa direction alors que ce n'est pas forcément la bonne, menant à de grandes erreurs
- **Interpolation inverse**
  - Pour calculer l'inverse d'une fonction en échangeant x et y
  - Utile pour calculer les racines d'une fonction par exemple
- **Phénomène de Runge**
  - Plus le degré est élevé, plus des oscillations auront tendance à apparaître
  - On ne sait pas nécessairement où les oscillations apparaîtront
  - *L'idéal est d'utiliser l'interpolation polynomiale avec le plus petit nombre faisable de points possibles*

- **Interpolation par partie**
  - Pour éviter le phénomène de Runge, on cherche $n-1$ polynômes de même degré qui interpolent chacun une petite partie de données

- **Curve fitting**
  - On se dit qu'en pratique, bien souvent, les données sont légèrement différentes d'une mesure à l'autre
  - Et donc ce n'est pas si important de passer absolument par nos points de données
  - C'est donc une approximation
  - Ce n'est pas une méthode par parties, on se retrouve avec un seul polynôme

- **Quantification du spread des données (déviation standard/variance)**

Méthodes :

Méthode	*Description*	*Complexité*	*Cas intéressants*	*Contraintes/problèmes*	*Avantages*
Vandermonde	Interpolation polynomiale, simple système d'équations	$O(n^3)$		- Ne marche pas pour de grands n ; le système sera mal conditionné	- Très simple
Lagrange	Interpolation polynomiale	$O(n^2)$	- Pour interpoler plusieurs valeurs de x	- Beaucoup de produits, donc forte chance d'accumulation d'erreurs - Pas très efficace - Phénomène de Runge à haut degré	- Relativement simple
Newton	Interpolation polynomiale	$O(n^2)$	- Pour interpoler plusieurs valeurs de x	- Phénomène de Runge à haut degré	- Peut aussi être calculé avec relations de récurrence
Neville	Interpolation polynomiale	$O(n^2)$	- Pour interpoler une seule valeur de x	- Phénomène de Runge à haut degré	- Plus rapide que les autres si on doit calculer une seule valeur
Cubic Splines	Interpolation par parties, en utilisant des splines de degré 3	$O(n)$?			- Connexion smooth entre les splines (on force la même pente et courbure) - On peut choisir la condition de bordure qu'on souhaite (pour les premier et dernier points de données) - Pas de phénomène de Runge
Least-squares	Curve fitting Pour un degré 2, on parle de régression linéaire, on associe une droite à des données x et y	$O(n)$?		- Ne passe pas forcément tout à fait par nos points de données initiaux	- On peut choisir relativement facilement le degré qu'on souhaite
Régression linéaire multivariée	Régression linéaire à deux variables, on associe un plan à des données x1, x2 et y			- Ne passe pas forcément tout à fait par nos points de données initiaux	

Chapitre 4 : roots of equations

Pour : trouver les racines réelles d'une fonction f

Méthodes :

Méthode	*Description*	*Complexité*	*Cas intéressants*	*Contraintes/problèmes*	*Avantages*
Root search	Permet de bracketer la zone de recherche			- Si deux racines sont très proches de l'autre, on a peu de chance de les détecter toutes les deux, voir de les détecter tout court - Si la fonction ne change pas de signe mais ne fais que toucher l'axe, ça ne sera pas détecté - L'algorithme détectera aussi les pôles (monter vers ∞ et reprendre depuis $-\infty$) - De nombreuses itérations	
Bisection	Similaire à la recherche dichotomique				- Peut être combiné à l'algorithme précédent
RegulaFalsi	Basé sur l'interpolation linéaire			- Nécessité que les bounds soient déjà bracketées	- Converge toujours - Convergence un peu meilleure que linéaire
Secant	Basé sur l'interpolation linéaire			- Ne converge pas toujours	- Pas de contrainte sur les bounds - Convergence superlinéaire
Ridder	Amélioration de RegulaFalsi en mixant avec la bisection	- Chaque itération nécessite deux évaluations de la fonction	- Si on ne sait pas calculer $f'(x)$	- Nécessité que les bounds soient déjà bracketées	- Convergence quadratique, donc meilleur que les deux précédents
Newton-Raphson	Basé sur Taylor		- Si on sait calculer $f'(x)$	- Ne converge parfois pas à cause du fait que la tangente n'est pas du tout une bonne approximation de la fonction en un certain point (par exemple au point supérieur d'une courbe en cloche) *Bien choisir les bornes initiales*	- Convergence quadratique
Newton-Raphson à n dimensions	Pour plusieurs dimensions		- Si on sait calculer $\Delta \boldsymbol{x}$ (matrice jacobienne qui contient toutes les dérivées partielles)		

Chapitre 5 : numerical differentiation

Pour : calculer la dérivée nième d'une fonction f pour un point x

Méthodes :

Méthode	*Description*	*Erreur*	*Cas intéressants*	*Contraintes/problèmes*	*Avantages*
Différence finie centrale	Basé sur Taylor avec les points environnants	$O(h^2)$	- Si on doit évaluer en un point avec suffisamment de points devant et derrière	- L'intervalle h entre les points doit être fixe	
Différence finie non-centrale	Basé sur Taylor avec uniquement les points suivants ou précédents	$O(h)$ pour la première (impopulaire) $O(h^2)$ pour la deuxième	- Si on doit évaluer en un point limite (par exemple au début ou à la fin)	- L'intervalle h entre les points doit être fixe - Si h est plutôt petit (ex. 0.00125), fortes erreurs d'arrondis *L'idéal est d'utiliser un float64* - Si h est plutôt grand (ex. 0.64), fortes erreurs de troncature *On peut régler ça avec Richardson*	
Extrapolation de Richardson	Réduit une erreur de troncature	$O(h^4)$ si la différence finie est en $O(h^2)$			- Combinée aux algorithmes précédents, plus grande précision

Dérivation par interpolation	Basé sur l'interpolation		- Si l'intervalle entre les points n'est pas fixe	- Ce n'est pas très précis avec une interpolation polynomiale unique à cause du phénomène de Runge *Idéalement utiliser les cubic splines (si not noisy) ou least-squares (si noisy)*	- L'intervalle ne peut pas être fixe

Chapitre 6 : numerical integration

Pour : calculer l'intégrale d'une fonction f

Méthodes :

Méthode	*Description*	*Erreur*	*Cas intéressants*	*Contraintes/problèmes*	*Avantages*
Méthodes naïves	Approximer par un unique rectangle (ordre 0, méthode du rectangle) Approximer par deux rectangles (ordre 1, méthode midpoint)			- Très peu précis	
Newton-Cotes	Approximer par un polynôme de degré k et avec n panneaux On distingue les règles "tout court" ($n = 1$, peu efficaces) des règles "composites" ($n > 1$, donc par parties)		- **Pour toutes les variations ci-dessous :** ○ Si $f(x)$ peut être facilement évalué ○ Si l'intervalle h est fixe	- **Pour toutes les variations ci-dessous :** ○ L'intervalle h entre les points doit être fixe ○ La fonction doit être continue	
Règle trapézoïdale composite	Newton-Cotes avec $k = 1$	$O(h^2)$			- Pas de contrainte sur le nombre de panneaux
Règle trapézoïdale récursive	Variation du trapèze composite				
Règle Simpson 1/3 composite	Newton-Cotes avec $k = 2$	$O(h^4)$		- n doit être pair	- Plus précis que le trapèze composite
Règle Simpson 3/8 composite	Newton-Cotes avec $k = 2$			- n doit être impair	
Intégration de Romberg	Variation du trapèze composite avec l'extrapolation de Richardson pour réduire l'erreur	$O(h^{2i})$ avec i le niveau			
Quadrature Gaussienne	Autre méthode, on intègre de a à b en utilisant une fonction de poids w(x)		- **Pour toutes les variations ci-dessous :** ○ Si l'évaluation de $f(x)$ est coûteuse ○ Si l'intervalle h n'est pas fixe	- Cette méthode en particulier est un peu laborieuse en terme de calcul, c'est pour ça qu'on a vu des variantes pour des cas particuliers avec des formules de poids et des nodes précalculés	- **Pour toutes les variations ci-dessous :** ○ Moins d'évaluations de $f(x)$ ○ L'intervalle h ne doit pas être fixe ○ Fonctionne si la fonction a des singularités comme des pôles ○ Fonctionne aussi si les bornes d'intégrations sont infinies
Gauss-Legendre	Quadrature gaussienne avec $a = -1, b = 1$ $w(x) = 1$ On peut passer de a à b random avec un changement de variable pour les ramener entre -1 et 1				
Gauss-Chebyshev	Quadrature gaussienne avec $a = -1, b = 1$ $w(x) = \frac{1}{\sqrt{1-x^2}} = (1-x^2)^{-1/2}$				
Gauss-Laguerre	Quadrature gaussienne avec $a = 0, b = \infty$ $w(x) = e^{-x}$				
Gauss-Hermite	Quadrature gaussienne avec $a = -\infty, b = \infty$ $w(x) = e^{-x^2}$				

Chapitre 7 : initial value problem / cauchy problem

Pour : approximer une fonction f si on a des informations sur ses dérivées ainsi qu'un point de départ (équations différentielle)

Matière :

- **Stabilité numérique**

Méthodes :

Méthode	*Description*	*Erreur*	*Cas intéressants*	*Contraintes/problèmes*	*Avantages*
Méthode d'Euler	Basé sur Taylor d'ordre 1	$O(h^2)$		- Peu précis, et peut mener à de grandes erreurs d'arrondis	
Méthode Runge-Kutta 2	Basé sur Taylor d'ordre 2	$O(h^3)$			
Méthode Runge-Kutta 4	Basé sur Taylor d'ordre 4	$O(h^5)$			

Chapitre 8 : introduction to optimization

Pour : trouver un x qui maximise $F(x)$

Matière :

- **Avec et sans contraintes**
  - On trouve avec contraintes des minimas là où les contraintes (d'égalité ou d'inégalité) sont respectées
  - On trouve sans contraintes des minimas là où les dérivées sont nulles
- **Transformation d'un problème avec contraintes vers un problème sans contraintes**
  - Il est plus simple de résoudre des problèmes sans contraintes
  - On va donc modifier $F(x)$ pour faire en sorte qu'elle soit pénalisée si la contrainte n'est pas respectée ; avec une force de λ

Méthodes :

Méthode	*Description*	*Erreur*	*Cas intéressants*	*Contraintes/problèmes*	*Avantages*
Bracketing	Pour bracketer le minimum, et on augmente l'étape progressivement (pas grave si le bracket est large)				
Golden section search	Pour réduire l'intervalle bracketée, similaire à la bisection				
Powell d'ordre zéro	Méthode d'ordre zéro, avec un point de départ, our plusieurs variables On choisit à chaque étape la direction idéale dans laquelle progresser, avec un point de départ		- Si on ne peut pas calculer les dérivées partielles facilement	- Les valeurs d'entrées doivent être indépendantes les unes des autres - A tendance à ne pas marcher si la fonction n'est pas quadratique *On peut faire une approximation de Taylor pour la rendre quadratique mais on préférera souvent utiliser le downhill simplex*	
Downhill simplex	Alternative à powell d'ordre zéro, qui essaie de bracketer/se déplace		- Si on ne peut pas calculer les dérivées partielles facilement	- Les valeurs d'entrées doivent être indépendantes les unes des autres - Souvent plus lent	- Beaucoup plus stable, marche la plupart du temps lorsque Powell ne marche pas

	dans l’espace avec des simplexes, avec un point de départ		- Si Powell ne marche pas		
Gradient descent	Méthode d’ordre 1, qui utilise le gradient		- Si on peut calculer les dérivées partielles	- Les valeurs d’entrées doivent être indépendantes les unes des autres	- Fonctionne très bien