



Outils mathématiques et numériques pour la physique

Nicolas Fressengeas

► To cite this version:

Nicolas Fressengeas. Outils mathématiques et numériques pour la physique. 3rd cycle. Université Paul Verlaine Metz, 2010. cel-00520195v1

HAL Id: cel-00520195

<https://cel.hal.science/cel-00520195v1>

Submitted on 22 Sep 2010 (v1), last revised 14 Dec 2010 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UE SPM-PHY-S07-101

Outils mathématiques et numériques pour la physique

N. Fressengeas

Laboratoire Matériaux Optiques, Photonique et Systèmes
Unité de Recherche commune à l'Université Paul Verlaine Metz et à Supélec

Document à télécharger sur <http://moodle.univ-metz.fr/>



Quelques ouvrages. . .

[GJP05, Pie01, Bis04]



Christopher M. Bishop.

Neural Networks for Pattern Recognition, chapter
7 :Parameter Optimization Algorithms.

Oxford University Press, 2004.



2005 Grivet Jean-Philippe.

Analyse numérique pour les sciences physiques.

Association *Libre Cours*, 2005.



Nougier Jean Pierre.

Méthodes de calcul numérique.

Hermes Sciences, 2001.



Partie I

Formats numériques et codage

I Formats numériques et codage

- 1 Nécessité d'un format de données
 - Fonctionnement d'un ordinateur
 - Stockage des données en mémoire
 - Un format, obligatoirement
- 2 Formats binaires
 - Formats binaires simples
 - Formats binaires complexes
- 3 Formats ASCII
 - Le code ASCII
 - Formats ASCII simples
 - Formats ASCII complexes
- 4 Formats d'image
 - Analyse d'une image numérique
 - Vectoriel ou BitMap ?
 - Formats comprimés avec ou sans perte ?

Fonctionnement élémentaire d'un ordinateur

Courte introduction pour ceux qui ne sauraient pas

Un ordinateur est un outil pour le traitement des données

- Il utilise des données d'entrée
- Effectue dessus un traitement programmé
- Fournit des données en sortie
- Et c'est tout !
- Les programmes les plus complexes se résument à ça

3 composants principaux

Le(s) microprocesseur(s)

CPU

- Effectue(nt) **toutes**¹ les tâches de traitement
- C'est un circuit électronique
- Les informations sont transmises sur des **bus** : ensembles de fils électriques

La mémoire

- Stockage des données d'entrée et de sortie
- Stockage des résultats intermédiaires
- Stockage des *programmes de traitement*

Les périphériques

Sans eux, ni entrée, ni sortie de donnée

¹Certaines tâches de traitement, comme la gestion de l'affichage, peuvent être sous-traitées à des cartes spécialisées

La mémoire

2 grands types de mémoire

La mémoire vive

RAM

- Très rapide (ns)
- Capacité limitée à quelques Giga Octets
- Sert au stockage **temporaire** des données

Le Disque Dur

HDD

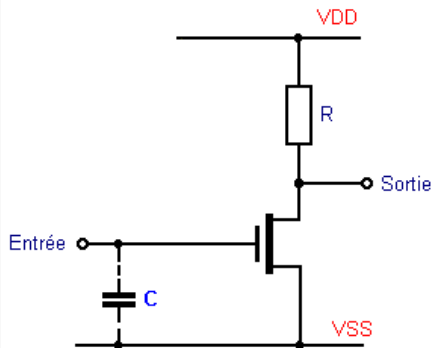
- 10^3 à 10^6 fois plus lent que la RAM
- Grandes capacités (Tera Octet)
- Stockage permanent
- Emulation de la RAM (swap) si elle vient à manquer

Cellule (BIT) de RAM

Un simple condensateur

BIT

- chargé/non chargé : 1/0
- Très volatile
- Rafrachissement régulier (ms)



Organisation et traitement de la mémoire

BITs regroupés en Octets

- 1 octet : 8 BITS à 0 ou 1
- 1 octet \longleftrightarrow entier de 0 à $2^8 - 1 = 255$

Traitement des octets par le microprocesseur

- Un par un pour les microprocesseurs 8 bits Z81,6809
- 2 par 2 pour les microprocesseurs 16 bits Motorola,68000
- Quatre à la fois pour les 32 bits Intel
- 8 à la fois pour les 64 bits AMD/Intel

Octets organisé en un tableau unique

Chacun porte un numéro : son **adresse**

Prenons un exemple

Comment stocker les entiers négatifs ou supérieurs à 255 ?

Stockage des grands entiers

- 8 bits ne suffisent pas : prenons 2 octets
- 2 octets : entiers de 0 à $2^{16} - 1 = 65535$
- `b1111 1111 1111 1110` vaut donc 65534

Nombre négatifs

- Prenons la convention du complément² à 2 :
Si le premier BIT est 1, le nombre représenté est l'opposé du complément à 2 des autres bits
- `b1111 1111 1111 1110` vaut $-b000\ 0000\ 0000\ 0001$ donc -1

²Cette convention est commode car, grâce à elle, l'addition relative est compatible avec les additionneur binaires de nombres entiers naturels

Un format pour les entiers

*b*1111 1111 1111 1110

65534 ou -1 ?

- Stocke-t-on des entiers relatif (ou non) ?
- Il faut le définir à priori
- C'est le **format** de ces 2 octets

Même les entiers naturels posent problème

Big Endian

Little Endian

- Prenons $b1000\ 0000\ 0000\ 0000$, entier naturel sur 2 octets
- Stockons l'octet de poids fort ($b1000\ 0000$) dans l'adresse la plus basse
- Puis l'octet de poids faible ($b0000\ 0000$) dans l'adresse la plus haute
- C'est le **format Big Endian**
- On aurait pu faire le contraire

Définition préalable du format

- Un entier naturel : 2 octets et un **format**
- Si on se trompe de format $2^{15} = 32768$ devient $2^7 = 128$
- Chaque processeur sa convention
(Intel : Little Endian, Motorola, SUN : Big Endian)

Comment enregistrer la mémoire sur le disque ?

C'est simple, BIT à BIT, non ?

Format binaire brut

- Ecriture de la mémoire telle quelle sur le disque (BIT à BIT)
- Rapide, simple et économe
- Exemple en C : le fonction `write`

Inconvénients majeurs

- Ne peut être relu QUE sur une machine du même type³
- Souvent uniquement par le même programme

³La convention BigEndian/LittleEndian doit par exemple rester inchangée

Exemple de format binaire simple bien connu

.EXE,.DLL sous WindowsTM

Les fichiers **exécutables**

- Ils doivent être transcrits tel quels en mémoire
- Pour y être exécutés
- Ils contiennent des codes d'instructions du microprocesseur

Corrolaire

- Ne sont lisible que sur le type de machine qui l'a écrit
- Souvent avec le même système d'exploitation uniquement

Lorsque la simple copie de mémoire ne suffit plus

Il faut complexifier les formats binaires

Stocker des documents complexes

Nécessité de stocker des informations de structure en plus des copies mémoire

Exemples de formats binaires complexes

- Fichiers Microsoft Office (avant 2007) : *.DOC...
- Format HDF (Hierarchical Data File)
- Format Origin...

Inconvénients majeurs

- Formats propriétaires, structure inconnue
- Interopérabilité nulle
- Sauf quand le format est publié (HDF)

Le code ASCII

American Standard Code for Information Interchange

Un code standard pour coder les caractères

- Standardisé dans les années 60
- Un octet = un caractères

Caractères de contrôle 0 à 31

- 7 : beep
- 10 : Line Feed
- 13 : Carriage Return

Exemples

- 48 à 57 : les chiffres
- 65 à 90 : les majuscules
- 97 à 122 : les minuscules⁴

⁴Entre A et a : modification du BIT 6

Code ASCII... encore un problème de format

Pour aller à la ligne avec une *vieille* machine à écrire

- Retour charriot CR
- Descente d'une ligne LF

Codage du retour à la ligne

- WindowsTM : CR+LF
- MacOS : CR
- Unix/Linux : LF

Conversion de format texte entre plateforme

- Soit faite par le logiciel de transfert e.g. ftp en mode texte
- Soit à faire a posteriori

Notion d'encodage

Codage des accents et autres cédilles

Le code ASCII définit 7 bits

de 0 à 127

- Les codes de 128 à 255 sont disponibles
- Il faut en définir le format
- C'est l'**encodage**

Format d'encodage

- Les formats d'encodage sont très variables
- D'une plateforme à une autre
- D'un pays à un autre
- Peu de normalisation...
- e.g. UTF-8, Latin1 (ISO 8859-1)...

Formats ASCII simples

LA solution à l'interopérabilité

Utilisation du codage ASCII de préférence sans encodage

- Interopérabilité maximum
- Lisible par la plupart des ordinateurs et logiciels
- e.g. : stockage des nombres par leur écriture décimale

Utilisation très répandue

- Tous les fichiers de configuration de Linux
- Format Comma⁵Separated Value
- Et bien d'autres...

Un inconvénient

- Taille des fichiers beaucoup plus importante qu'en binaire
- Solution : la compression

⁵La *Comma* du format est une virgule qui peut être remplacée par autre

Les formats ASCII complexes

Des formats simples qui se sont complexifiés peu à peu

Des formats polyvalents

- Interopérabilité des formats ASCII
- Flexibilité des formats complexes
- Permettent de décrire à peu près tout
- Normalisation des formats complexes : la norme XML
- Formats très répandus

Exemples de formats ASCII complexes

Formats répondant à la norme XML

- HTML Protocole WWW
- Open Document¹ OpenOffice *.odt, *.ods...
- OpenXML¹ MS Office *.docx, *.xlsx...
- ...

Autres formats ASCII complexes

- L^AT_EX
- PostScript, Portable Document Format
- ...

¹Si vous tentez d'ouvrir ces fichiers avec un éditeur de texte simple, vous aurez l'impression que c'est un format binaire. Il s'agit cependant d'une simple compression (ZIP) de fichiers ASCII.

Caractéristiques d'une image numérique

Une image analogique échantillonnée

Pas toujours

- Ensemble fini de points
- Echantillonnage 2D : un pas en x , un autre en y
- Un codage de couleur en chaque point

Caractéristiques d'une image numérique

- Sa résolution, éventuellement différente en x et y

$$\text{Resolution} = \frac{\text{Nombre de points}}{\text{Taille}}$$

- Sa profondeur de couleur

Nombre de BITS utilisés pour le codage des couleurs

Prenons un exemple

Prise de vue avec un appareil numérique conventionnel 5Mpixels

Caractéristiques de l'image

- Nombre de points : $1944 * 2592 = 5\,038\,848$
- Profondeur de couleur : $3 * 8 = 24$ BITS
256 niveaux pour chaque couleur primaire
soit $2^{24} = 16\,777\,216$ couleurs
- Mémoire totale utilisée :
 $3 * 5\,038\,848 = 15\,116\,544$ octets ≈ 15 Mo.

Sa résolution dépend de la taille physique souhaitée e.g. 30*40cm

$$\frac{1944}{30} = 64,8 \approx \frac{2592}{40} = 64,9 \text{ points/cm} \approx 165 \text{ Dot Per Inch}$$

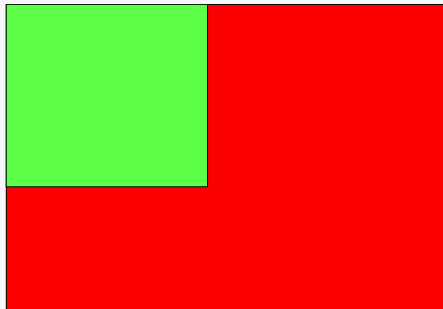
Format BitMap vs. format vectoriel

Codage BitMap

1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Codage vectoriel

- Un carré rouge (x_1, y_1)
- Un carré vert (x_2, y_2)



Les formats vectoriels

Pour la description des figures géométriques

Avantages

- Très faible encombrement mémoire
- Description indépendante de la résolution
- Résolution virtuellement infinie

Inconvénients

- IL n'y en a pas pour les objets géométriques
A part (peut être) le manque d'outils populaires pour les manipuler⁶
- Encombrement mémoire énorme pour une image échantillonnée
Sauf si celle-ci est considérée comme un objet BitMap

⁶Citons Inkscape, excellent logiciel libre qui manipule la plupart des formats vectoriels

Formats combinés

Insertion d'une BitMap dans un format vectoriel

La plupart des formats vectoriels permettent d'insérer des objets BitMap en tant qu'objets vectoriels



Fig.: Insertion d'une image BitMap dans cette présentation vectorielle

En pratique

Choix d'un format BitMap

- Photos
- Images analogiques numérisées

Taille du fichier

- Enorme
- Recours à la compression

Formats utilisés

- Graphics Interchange Format
- Joint Photographic Experts Group
- Portable Network Graphics
- Tagged Image File Format

Choix d'un format vectoriel

- Diagrammes
- Schémas
- Images géométriques

Taille du fichier

Tout petit

Formats utilisés

- PostScript
- Portable Document Format
- Scalable Vector Graphics
- Windows Meta File
- Enhanced Meta File

Formats comprimés sans perte

PNG, GIF . . .

Principe : élimination de la redondance

1	1	1	0	0	0		
1	1	1	0	0	0	3	1 3 0
1	1	1	0	0	0	3	
0	0	0	0	0	0	⇒ 6	0
0	0	0	0	0	0	3	
0	0	0	0	0	0		

Même principe que la compression ZIP standard

- Fonctionne très mal avec des photos
- A réserver aux images géométriques **que l'on ne peut avoir en format vectoriel**

Formats comprimés avec perte

Prise en compte des performances limitées de l'œil humain

Les performances de l'œil – et du cerveau – humain

- Vision Noir & Blanc détaillée et périphériques
- Gestion des couleurs centrale et peu résolue

Conséquences pour le codage des images

- La luminance doit être bien résolue
- Les couleurs – la chrominance – peuvent l'être moins

Et donc... le codage JPEG

- Décomposition Luminance-Chrominance
- Transformée de Fourier spatiale 2D
- Suppression des hautes fréquences spatiales invisibles
- Compression jusqu'à un facteur 25 sans perte apparente de qualité

JPEG : attention aux schémas et graphiques

Les codages avec perte sont conçus pour coder les images analogiques numérisées

Ça bave...

- Schémas et graphiques : hautes fréquences spatiales
- Codage JPEG adapté aux image analogiques numérisées
- Résultat peu convaincant...

Ca bave

Les formats vidéo

MPEG

Du fait de l'énorme taille des fichiers, les formats vidéo sont des formats de codage

Codage JPEG image par image

Norme MJPEG... abandonnée

Exploitation de la redondance temporelle

- On ne code que les changements
- MPEG 1/2 ou 4
- Trois normes qui diffèrent par le taux de compression

Utilisation

- MPEG 1 : abandonnée
- MPEG 2 : DVD / TNT Gratuite
- MPEG 4 : TNT payante / télévision sur IP

Partie II

Résolution numérique des systèmes linéaires

II Résolution numérique des systèmes linéaires

5 Méthodes directes

- Le pivot de Gauss
- Le pivot de Gauss : notation matricielle
- Autres méthodes directes

6 Conditionnement d'un système linéaire

- Notion de conditionnement
- Attitude à adopter face à un système mal conditionné
- Notion de pré-conditionnement

7 Méthodes itératives

- Méthodes itératives et matrices creuses
- Principes généraux
- Quelques méthodes classiques

Résoudre des systèmes linéaires, pourquoi ?

Et pourquoi faire un cours là-dessus, vous l'avez appris au lycée !

On en trouve partout

- Physique, Chimie, Mécanique...
- Généralement issus de la résolution des Équations Différentielles (ED)

Pas si faciles à résoudre

- Pour les systèmes linéaires simples : substitution, addition...
- Les ED génèrent de **grands** systèmes : 1000 inconnues ou plus
- Nécessité d'une méthode systématique programmable

Méthodes systématiques de résolution

Deux grandes classes

Les méthode directes

- Le pivot de Gauss en est le meilleur représentant
- Elles donnent un résultat exact aux erreurs d'arrondi près

Les méthode indirectes

- Ce sont des méthodes itératives
- On construit une suite convergent vers la solution
- La solution trouvée est toujours approchée

La méthode du Pivot de Gauss

C'est la méthode reine, toutes les autres en découle

Principe

- Objectif : transformer un système linéaire en un système du type $Rx = c$

- avec R triangulaire supérieure

$$R = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \end{pmatrix}$$

La solution d'un système triangulaire est directe

- Par substitution à partir de la dernière ligne
- On peut le résoudre formellement

$$\forall i : r_{ii} \neq 0 \Rightarrow x_i = \frac{1}{r_{ii}} \left(c_i - \sum_{k=i+1}^n r_{ik} x_k \right)$$

Description formelle de la méthode du pivot de Gauss

Système à résoudre

$$Ax = b$$

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

Pour mettre des 0 sur la première colonne de la ligne k

- Si L_1 est la première ligne
- Soustraire $\frac{a_{k1}}{a_{11}} L_1$ à la ligne k
- a_{11} est appelé le **pivot**

Après la première étape

Un nouveau système à résoudre

$$A'x = b'$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a'_{22} & \cdots & a'_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & a'_{n2} & \cdots & a'_{nn} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b'_2 \\ \vdots \\ b'_n \end{pmatrix}$$

Et on recommence sur la sous matrice. . .

Quelques remarques. . .

- La première ligne est inchangée
- le second membre b doit subir les même modifications
- Ceci n'est possible que si tous les a_{ji} sont non nuls

Choix du pivot

Que faire si le pivot est nul ou trop petit ?

$$Ax = b$$

Si le pivot est nul

- Une permutation de lignes ou de colonnes résout le problème
- Lignes permutées ? Éléments de b aussi !
- Colonnes permutées ? x doit l'être aussi !

Un pivot trop petit ?

Un pivot trop petit induit des erreurs d'arrondi

L'ordinateur n'aime pas diviser par de petits nombres

- Si δ est connu à ε près, comme dans tout ordinateur
- Comparez l'erreur relative obtenue sur le calcul de $1/\delta$
 - Pour $\delta = 2\varepsilon$
 - Pour $\delta = 10^6\varepsilon$

Un bon pivot doit être grand ... voire le plus grand possible

- **Méthode du pivot partiel** : permuter les lignes pour obtenir le plus grand pivot
- **Méthode du pivot total** : permuter lignes et colonnes

Et si, malgré tout le pivot est nul

De l'inconvénient d'avoir un petit pivot

Tous les pivots possibles sont nuls

- A un pas de la méthode, impossibilité de trouver un pivot non nuls
- Cela signifie qu'une des inconnues a toujours un coefficient nul
- Le système n'est pas soluble : il est dit **singulier**
- Il possède une infinité de solution ou pas du tout

Si le pivot est trop petit

- Le système est sûrement mal conditionné (presque singulier)
- On en parle à la fin de cette partie

Algorithme du Pivot de Gauss

$$Ax = b$$

Résumé

- ❶ Déterminer (r, s) tel que $|a_{rs}| = \max_{i,j} |a_{i,j}|$
- ❷ Si $|a_{rs}| = 0$ alors STOP, le système est singulier
- ❸ Sinon échanger lignes (et colonnes) pour obtenir le système $\overline{A}\overline{x} = \overline{b}$
- ❹ Pour $i > 1$:
 - Si L_1 est la première ligne
 - Soustraire $\frac{\overline{a}_{i1}}{\overline{a}_{11}} * L1$ à la ligne i
 - Pour obtenir le nouveau système $A' * x' = b'$
- ❺ Recommencer en (1) avec la sous-matrice A' de A privée des premières ligne et colonne



Notations matricielles des permutations

Permutation des lignes i et j

$$\bar{A} = \mathcal{P}_{ij} * A$$

$$\mathcal{P}_{ij} = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 0 & \cdots & 1 \\ & & \vdots & \ddots & \vdots \\ & & 1 & \cdots & 0 \\ & & & & & 1 \end{pmatrix}$$

- Matrice identité dont les éléments (i, i) et (j, j) sont permutés avec (i, j) et (j, i)
- Multiplication à gauche : permutation des lignes
- A droite : permutation des colonnes
- Remarque : $\mathcal{P}_{ij}^{-1} = \mathcal{P}_{ij}$

Notation matricielle de l'opération Pivot

Comment mettre des 0 dans une colonnes à l'aide d'un produit matriciel

Soustraire Soustraire $\frac{\overline{a_{j1}}}{\overline{a_{11}}} * L1$ à la ligne i

$$A' = G\overline{A}$$

$$G = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ -\frac{\overline{a_{21}}}{\overline{a_{11}}} & 1 & \ddots & & & \vdots \\ -\frac{\overline{a_{31}}}{\overline{a_{11}}} & 0 & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & 1 & 0 \\ -\frac{\overline{a_{n1}}}{\overline{a_{11}}} & 0 & \dots & \dots & 0 & 1 \end{pmatrix}$$

Remarque sur l'inverse de \mathcal{G}

Il est facile à trouver⁷

$$\mathcal{G}^{-1} = \begin{pmatrix} 1 & 0 & \dots & \dots & \dots & 0 \\ +\frac{\overline{a_{21}}}{a_{11}} & 1 & \ddots & & & \vdots \\ +\frac{\overline{a_{31}}}{a_{11}} & 0 & \ddots & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & 1 & 0 \\ +\frac{\overline{a_{n1}}}{a_{11}} & 0 & \dots & \dots & 0 & 1 \end{pmatrix}$$

⁷Chercher une démonstration sans calcul

Notation matricielle de la méthode de Gauss

Chaque étape conduit à produire des 0 sur la colonne suivante

- $A \rightarrow A^{[1]} \rightarrow A^{[2]} \rightarrow \dots \rightarrow A^{[j]} \rightarrow \dots \rightarrow R$
- $b \rightarrow b^{[1]} \rightarrow b^{[2]} \rightarrow \dots \rightarrow b^{[j]} \rightarrow \dots \rightarrow c$

Forme de $A^{[j]}$

$$A^{[j]} = \begin{pmatrix} \begin{bmatrix} * & \dots & * \\ & \ddots & \vdots \\ 0 & & * \end{bmatrix} & * \\ & 0 & *_{(jj)} & * \end{pmatrix}$$

avec $A^{[j]} = g^{[j]} p^{[j]} A^{[j-1]}$ et $b^{[j]} = g^{[j]} p^{[j]} b^{[j-1]}$

Expression matricielle de la matrice triangulaire R

Rappel

- $A^{[j]} = \mathcal{G}^{[j]} \mathcal{P}^{[j]} A^{[j-1]}$
- $b^{[j]} = \mathcal{G}^{[j]} \mathcal{P}^{[j]} b^{[j-1]}$

On en déduit

- $R = \left(\prod_{k=n-1}^1 \mathcal{G}^{[k]} \mathcal{P}^{[k]} \right) A$
- $c = \left(\prod_{k=n-1}^1 \mathcal{G}^{[k]} \mathcal{P}^{[k]} \right) b$

Décomposition⁸ L.R : produit de deux matrices triangulaires

Dans le cas particulier où aucune permutation n'est requise

Si aucune permutation n'est nécessaire

- Si $\forall k, \mathcal{P}^{[k]} = I$ alors $R = \left(\prod_{k=n-1}^1 \mathcal{G}^{[k]} \right) A$

- Donc $A = \underbrace{\left(\prod_1^{k=n-1} \mathcal{G}^{[k]-1} \right)}_{\text{Triangulaire inférieure}} \times R = L.R$

⁷On parle aussi de décomposition L.U. (Lower.Upper)

Décomposition LR dans le cas général

La permutation de A est parfois nécessaire

- Pour éviter les pivots non nuls
- Mais aussi pour choisir **les meilleurs pivots**

Décomposition LR conventionnelle

- $A = \mathcal{P}LR$ où \mathcal{P} est une matrice de permutation

De l'utilité de la décomposition LR

Résolution de systèmes linéaires pour divers seconds membres

- Supposons $A = \mathcal{P}LR$
- A résoudre : $Ax = b_i$ pour divers i
- $\mathcal{P}L(Rx) = b_i$
- $\begin{cases} Rx = y \\ \mathcal{P}Ly = b_i \end{cases}$
- Ce sont deux systèmes triangulaires⁹ simples à résoudre
- Utiliser la décomposition LR formellement revient à utiliser la méthode de Gauss

Autres utilisations

- Inversion matricielle
- Calcul de déterminant

⁹Le deuxième système est triangulaire à une permutation près, ce qui ne change pas la complexité de sa résolution.

De la supériorité numérique de la méthode de Gauss

Tous les logiciels de calcul utilisent la méthode de Gauss pour le calcul de déterminant, l'inversion...

Pivot de Gauss ou calcul de l'inverse ?

- $Ax = b$ pourrait se résoudre par le calcul de l'inverse :
 $x = A^{-1}b$
- Le calcul de l'inverse pourrait être fait par une autre méthode¹⁰
- La méthode du Pivot est la plus rapide (complexité en $n^3/2$)

¹⁰e.g. en utilisant $A \times^t \text{comm} A = \det A \times I$

Élimination de Gauss-Jordan

Une autre présentation de la méthode de Gauss

Résolution d'un système

$Ax = b$

$$([A] [b]) \Rightarrow \left(\begin{bmatrix} * & \cdots & \cdots & * \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & * \end{bmatrix} [c] \right) \Rightarrow \left(\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} [\textcolor{red}{x}] \right)$$

Système triangulaire résolu à l'aide du Pivot de Gauss (inversé)

Remarque sur le Pivot Total

- Si le Pivot Total est utilisé, la permutation des colonnes permute le vecteur solution $[\textcolor{red}{x}]$
- La *remontée* ne peut faire l'objet de recherche de pivot

Élimination de Gauss-Jordan

Application au calcul de l'inverse matriciel

Déterminer l'inverse de A , c'est résoudre des systèmes

- Chaque colonne de l'inverse est solution d'un système :
 - dont la matrice est A
 - le second membre est un vecteur de base
- Proposition : les résoudre tous en même temps

Le calcul de l'inverse le plus rapide qui soit

$$\left([A] \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix} \right) \Rightarrow \left(\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 1 \end{bmatrix} [A^{-1}] \right)$$

Un système mal conditionné ?

Prenons un exemple

Système A

$$\begin{cases} x_1 - x_2 = 1 \\ 1,002x_1 - x_2 = 1 \end{cases}$$

Solution

$$\begin{cases} x_1 = 0 \\ x_2 = -1 \end{cases}$$

Système B

$$\begin{cases} x_1 - x_2 = 1,001 \\ 1,002x_1 - x_2 = 1 \end{cases}$$

Solution

$$\begin{cases} x_1 = 0,5 \\ x_2 = -0,5 \end{cases}$$

Deux systèmes quasi identiques

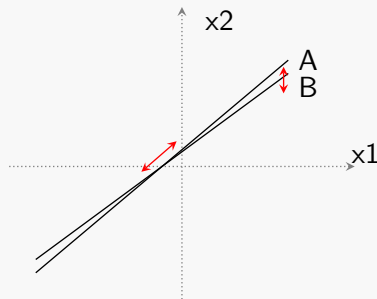
- Ils ne diffèrent que de 0,1%
- Leurs solutions sont très différentes
- Ils sont dits **mal conditionnés**

Systemes mal conditionnés

Interprétation graphique pour un système à 2 inconnues

Droites quasi-parallèles

Une variation **infime** de l'ordonnée à l'origine ou de la pente modifie le point d'intersection substantiellement



De l'importance du conditionnement d'un système linéaire

Sources d'erreur

- Analyse numérique toujours faite par ordinateur
- Le stockage en mémoire crée des **erreurs d'arrondi**

Si je résous un système mal conditionné

- Les coefficients sont entachés d'erreurs
- **Le résultat n'a aucune signification**

Définition quantitative du conditionnement

En anglais, le *Condition Number*

Conditionnement de $Ax = b$

$$\text{cond}(A) = \|A\| \cdot \|A^{-1}\|$$

Cette définition est valide quelle que soit la norme matricielle choisie

Erreur sur le second membre

$$\frac{\|\Delta x\|}{\|x\|} \leq \text{cond}(A) \frac{\|\Delta b\|}{\|b\|}$$

Erreur sur la matrice

$$\frac{\|\Delta x\|}{\|x + \Delta x\|} \leq \text{cond}(A) \frac{\|\Delta A\|}{\|A\|}$$

Un bon conditionnement est petit

Un système mal conditionné est un problème mal posé

Quand le système n'est pas inversible

- Un système non inversible est un système qui n'a pas de solution (ou une infinité)
- Chercher une solution à ce système n'a pas de sens

Quand il est mal conditionné

- Système non inversible + erreurs ?
 - Chercher une solution n'a pas de sens
- Problème mal posé
 - Ne demande qu'à devenir non inversible

Premier réflexe : reconsidérer le problème posé

Ne suis je pas en train de chercher à résoudre un problème qui n'a pas de solution ?

Méthode de résolution d'un système mal conditionné

Méthode de résolution d'un problème mal posé... **donc faillible**

$$Ax = b$$

Idee : considérer que la solution du système n'est qu'approchée

- x^* solution approchée de $Ax = b$
- x^* n'est pas solution donc $Ax^* = b^*$
- Par différence : $A(x - x^*) = (b - b^*)$
- On retrouve le même système^a : $A \Delta x = \Delta b$
- Dont la solution approchée est Δx^*

^aSeul le second membre variant, penser à la décomposition LR

Solution obtenue en itérant le processus

$$x^* + \Delta x^* + \Delta\Delta x^* + \Delta\Delta\Delta x^* + \dots$$

Le pré-conditionnement

Une attitude alternative face à un mauvais conditionnement

Définition

P est appelé *pré-conditionneur* de A si $\text{cond}(P^{-1}A) < \text{cond}(A)$

Principe

- Résoudre $(P^{-1}A)x = (P^{-1}b)$ au lieu de $Ax = b$

Détermination de P

- P n'est en général pas calculée directement
- Elle est souvent issue d'un algorithme dérivé de la méthode de Gauss

Utilisation de pré-conditionnement

- En général peu d'intérêt pour les méthodes directes
- Peut présenter un intérêt pour les méthodes itératives

Pourquoi résoudre des systèmes linéaires ?

Résolution Numérique des Équations Différentielles

- De grands systèmes : autant d'inconnues que de points de discrétisation
- Des systèmes dits **creux**¹¹ : avec beaucoup de 0

Problèmes posés :

- Grands systèmes : beaucoup de mémoire utilisée
- Temps de calcul important
- parfois impraticable même sur les ordinateurs actuels

Solution proposée : ne pas stocker les 0

- Il faut utiliser des algorithmes de calcul qui travaillent avec des matrices creuses
- Ce n'est pas le cas du Pivot de Gauss

¹¹ Creux en anglais : *sparse*

Comment conserver des systèmes creux ?

Les matrices creuses

- Elles sont issues de relations différentielles
- Qui impliquent en général les points voisins
- Les éléments non nuls sont donc proches de la diagonale
- On parle de matrices multidimensionnelles

Proposition de méthode itérative

- Construction d'une suite n'impliquant que des matrices creuses
- Définie par une **relation de récurrence**
- Son point fixe est la solution du système
- La suite converge vers la solution¹²

¹²Une suite définie par une relation de récurrence ne peut que converger vers son point fixe, **si elle converge**

Méthodes itératives : des suites vectorielles

Avantages et inconvénients

Avantages des méthodes itératives

- Limiter les besoins en mémoire vive
- Et donc limiter le temps de calcul

Inconvénients

- Convergence en général assez lente
- Solution nécessairement approchée

Quel type de méthode choisir ?

- Pour les petits systèmes (creux ou denses) : Gauss
- Pour les grands systèmes creux : méthode itératives
- Pour les grands systèmes denses : Gauss (mais ce sera difficile)

Construction de la suite vectorielle

$$Ax = b$$

Une matrice arbitraire B permet de faire apparaître la solution x comme point fixe

Introduction d'une matrice arbitraire

B

$$Ax = b \Leftrightarrow Bx + (A - B)x = b$$

Construction de la suite vectorielle

- $Bx_{i+1} + (A - B)x_i = b$
- $x_{i+1} = x_i - B^{-1}(Ax_i - b)$
- $x_{i+1} = (I - B^{-1}A)x_i + B^{-1}b$
- $x_{i+1} = Mx_i + p$

Jacobi, Gauss-Seidel...

Le choix d'un pré-conditionnement

- Le choix de B correspond au choix d'une méthode
- La vitesse de convergence en dépendra

Convergence, valeurs propres et choix de B

Convergence de la suite $x_{i+1} = Mx_i + p$

- Elle converge¹³ si $\rho(M) < 1$
- $\rho(M)$ est le **rayon spectral** de M
- $\rho(M) = \max_i |\lambda_i|$, si les λ_i sont les **valeurs propres** de M

Le choix de B est donc guidé par

- $\rho(I - B^{-1}A) < 1$ et le plus petit possible
- $Bx_{i+1} + (A - B)x_i = b$ aisément inversible

¹³Une démonstration élémentaire de cette propriété pour la suite $x_{i+1} = Mx_i + p$ peut être faite en se plaçant dans la base des vecteurs propres.

Quelques matrices auxiliaires

$$Ax = b$$

Décomposition de A en matrice diagonale (D) et deux matrices triangulaires (E et F)

$$A = D - E - F$$

$$D = \begin{pmatrix} a_{11} & & 0 \\ & \ddots & \\ 0 & & a_{nn} \end{pmatrix}$$

$$E = - \begin{pmatrix} 0 & \dots & \dots & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \dots & a_{n,n-1} & 0 \end{pmatrix}$$

$$\begin{cases} L = D^{-1}E \\ U = D^{-1}F \\ J = L + U \\ H = (I - L)^{-1}U \end{cases}$$

$$F = - \begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \dots & \dots & 0 \end{pmatrix}$$

Méthode de Jacobi

Aussi connue sous le nom de *méthode du pas total*

$$B = D$$

$$B = D$$

$$M = (I - B^{-1}A) = J$$

$$x^{[i+1]} = Mx^{[i]} + B^{-1}b$$

$$x_j^{[i+1]} = \frac{1}{a_{jj}} \left(b_j - \sum_{k \neq j} a_{jk} x_k^{[i]} \right)$$

Méthode de Gauss-Seidel

Aussi connue sous le nom de *méthode du pas unique*

$$B = D - E$$

$$M = (I - B^{-1}A) = (I - L)^{-1}U = H$$

$$x^{[i+1]} = Mx^{[i]} + B^{-1}b$$

B étant triangulaire, la détermination de M se fait par résolution d'un système triangulaire¹⁴

$$\forall i, \sum_{k < j} a_{jk} x_k^{[i+1]} + a_{jj} x_j^{[i+1]} + \sum_{k > j} a_{jk} x_k^{[i]} = b_j$$

¹⁴On rappelle qu'en analyse numérique, il est absolument proscrit de calculer un inverse autrement que par la méthode de Gauss ou via une méthode itérative. Dans ce cas ci, l'application de la formule pour i croissant à partir de 1 suffit.

Relaxation de Gauss Seidel

Généralisation de la méthode de Gauss-Seidel

Paramètre de relaxation : $\omega \in \mathbb{R}$

$$B(\omega) = \frac{1}{\omega} D(I - \omega L)$$

- $\omega = 1$: Gauss-Seidel
- $\omega < 1$: sous-relaxation
- $\omega > 1$: sur-relaxation
- Le choix de ω est un point difficile hors du périmètre de ce cours
- Permet cependant d'ajuster le rayon spectral

Partie III

Optimisation et systèmes non linéaires

III Optimisation et systèmes non linéaires

8 Équivalence Résolution – Optimisation

- Par le truchement d'une dérivée
- Dimensions de l'espace d'arrivée

9 Méthodes itérative de résolution

- Résolution dans \mathbb{R}
- Résolution dans \mathbb{R}^n

10 Application à l'optimisation

- Optimisation dans \mathbb{R}
- Optimisation dans \mathbb{R}^n

Équivalence entre Résolution et Optimisation

Optimisation

- L'optimisation est la détermination du paramètre qui permet de maximiser ou de minimiser une fonction
- Se traduit par l'annulation de dérivée : $f(x)_{\min}^{\max} \Leftrightarrow f'(x) = 0$
- Ou de gradient : $f \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}_{\min}^{\max} \Rightarrow \forall i, \frac{\partial f}{\partial x_i} = 0$

Équivalence

- Équivalence entre résolution et optimisation
- Moyennant une dérivée

Problèmes des dimensions de l'espace d'arrivée

$f : \mathcal{E} \mapsto \mathcal{F}$ avec $\mathcal{E} = \mathbb{R}^n$ et $\mathcal{F} = \mathbb{R}^m$

Fonctions à valeurs réelles

$\mathcal{F} = \mathbb{R}$

- Peut être optimisée
- Peut servir dans une résolution
- Quelle que soit la dimension n de l'espace de départ

Fonctions à valeurs vectorielles

$m > 1$

- Ne peut pas être optimisée : \mathbb{R}^m n'est pas muni d'une relation d'ordre
- Peut servir dans une résolution : correspond alors à m problèmes réels

Cas particulier $m=1$

$f : \mathcal{E} \mapsto \mathbb{R}$

Sans perte de généralité, on s'intéressera donc uniquement au cas
 $m = 1$

Méthodes itératives : construction

$$f(x) = 0$$

Principe

- Construction d'une suite convergent vers la solution
- Définie par une relation de récurrence $\Phi : \mathcal{E} \mapsto \mathcal{E}$
- Point fixe ξ solution : $\Phi(\xi) = \xi \Leftrightarrow f(\xi) = 0$

Construction de Φ

- Elle peut être évidente
 - A résoudre $x = \cos(x) \Rightarrow f(x) = x - \cos(x)$
 - $\Phi = \cos$
- Si elle n'est pas évidente
 - Un développement limité peut aider

La méthode de Newton

Une construction de Φ sur la base d'un développement limité

Développement de $f(x) = 0$

- Soit $x : f(x) = 0$
- ξ au voisinage de x
- $$f(x) = \sum_{n=0}^{+\infty} \frac{(x - \xi)^n}{n!} f^{(n)}(\xi)$$

$f(x) = 0$

Troncature ordre 1 ou 2

- $x^* = \xi - \frac{f(\xi)}{f'(\xi)}$ $x^* = \xi - \frac{f'(\xi) \pm \sqrt{f'(\xi)^2 - 2f(\xi)f''(\xi)}}{f''(\xi)}$
- ξ donné, x^* approche x à l'ordre idoine
- x^* peut servir de nouveau ξ : Φ naturellement définie

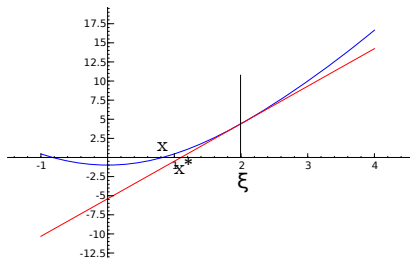
La méthode de Newton

Formulation formelle et interprétation graphique

Relation de récurrence

ordre 1

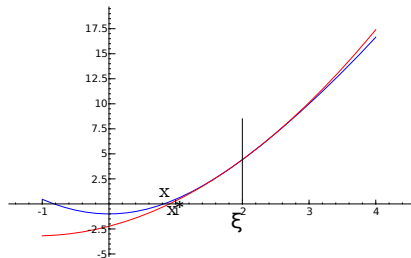
$$\Phi(x) = x - \frac{f(x)}{f'(x)}$$



Relation de récurrence

ordre 2

$$\Phi(x) = x - \frac{f'(x) \pm \sqrt{f'(x)^2 - 2f(x)f''(x)}}{f''(x)}$$



Newton \Leftrightarrow Approximation par un polynôme

- f est approchée par un polynôme d'ordre N
- Une des racines est prise comme solution approchée

Méthode de la fausse position

Aussi connue sous le nom de *Regula falsi* ou méthode de la sécante

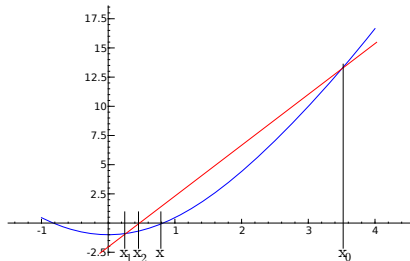
Principe

- Dérivée de la méthode Newton
- Quand la dérivée n'est pas calculable
- Approximation de la dérivée par une sécante

$$\bullet \quad f'(x) = \lim_{x \rightarrow x_0} \frac{f(x) - f(x_0)}{x - x_0} \Rightarrow f'(x) \approx \frac{f(x) - f(x_0)}{x - x_0}$$

Récurrence

$$x_{n+1} = x_n - f(x_n) \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$



Méthode de Newton multidimensionnelle

Équation dans \mathbb{R}^n $f : \mathbb{R}^n \mapsto \mathbb{R}^n$

$$f(x) = 0 \Leftrightarrow \begin{pmatrix} f_1(x_1, \dots, x_n) \\ \vdots \\ f_n(x_1, \dots, x_n) \end{pmatrix} = 0$$

Développement limité à l'ordre 1

$$f(x) \approx f(\xi) + D_\xi^f \cdot (x - \xi)$$

La Jacobienne

$$D_x^f = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \dots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \dots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

La relation de récurrence

- Si D est inversible
- $\Phi(x) = x - (D_x^f)^{-1} f(x)$

Optimisation dans \mathbb{R}

Recherche d'extrema dans \mathbb{R}

- Annulation de la dérivée
- Recherche de racine de la dérivée
- Méthodes de résolution

Exemples avec la méthode Newton

- ordre 1 : $\Phi(x) = x - \frac{f'(x)}{f''(x)}$
- ordre 2 : $\Phi(x) = x - \frac{f'''(x) \pm \sqrt{f'''(x)^2 - 2f''(x)f''''(x)}}{f''''(x)}$

Exemple dans \mathbb{R}^2

$$f : \mathbb{R}^2 \mapsto \mathbb{R}$$

Le skieur minimise son altitude f

- Méthode de la plus grande pente
- $\Phi(x) = x - \text{grad}_x(f)$

Plusieurs stratégies sont possibles

- Facteur correctif matriciel η
- $\Phi(x) = x - \eta \cdot \text{grad}_x(f)$
- Choix de η : choix de la méthode¹⁵



¹⁵Méthode de la plus grande pente : η constant

Choix de la méthode de descente

Inconvénient de la plus grande pente

La convergence peut être lente



Comment y remédier ?

- Par le choix correct du facteur correctif η
- Il existe beaucoup de propositions pour η
- Présentons quelques unes d'entre elles :
 - La méthode de Newton et ses dérivées

Méthode de Newton

Méthode dérivée du développement limité multidimensionnel à l'ordre 2

$$\eta = H^{-1}$$

Relation de récurrence

$$\Phi(x) = x - (H_x^f)^{-1} \cdot \text{grad}_x(f)$$

La Hessienne

$$H_x^f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

Méthodes dérivées

- N'est pas sans rappeler la méthode dans \mathbb{R}
- D'autres méthodes consistent à approcher H
 - Remplacer H par sa diagonale pour mieux l'inverser
 - Remplacer H par $H + \lambda I$
 - intermédiaire entre Newton et plus grande pente
 - réglable par λ

Partie IV

Interpolation, dérivation et intégration

IV Interpolation, dérivation et intégration

11 Interpolation

- Polynômes de Lagrange
- Autre formes d'interpolations polynômiales
- Interpolation par splines

12 Dérivation numérique

- Dérivation numérique d'une fonction analytique
- Dérivation numérique d'une fonction numérique

13 Intégration numérique

- Les méthodes de Newton-Cotes
- Intégration de Gauss

L'interpolation

Pourquoi interpoler ?

- Obtenir une valeur approchée là où il n'y a pas de valeur exacte
- Dériver une fonction numérique
- Intégrer numériquement avec d'avantage de précision

Interpolation par les polynômes de Lagrange

Faire passer un polynôme de degré $n - 1$ par n points

n points à interpoler

$$\forall i \in [1, n], (x_i, y_i)$$

Polynôme de degré $n - 1$

$$p(x) = \sum_{i=0}^n y_i L_i(x)$$

Polynôme de Lagrange associé aux n points (x_i)

(L_i)

- Degré $n - 1$
- $L_i(x_k) = \delta_{ik} : 1$ pour $i = k$, 0 sinon.

Expression du polynôme de Lagrange

- Annulation pour $i \neq k$
- Normalisation à 1

$$L_i(x) = \prod_{k=0, k \neq i}^n \frac{x - x_k}{x_i - x_k}$$

Autre formes d'interpolation polynômiale

Polynôme de Newton

- Même degré que le polynôme d'interpolation de Lagrange
- Passe par les même points
- C'est le même
- Juste une autre façon de calculer

Interpolation de Hermite

$$p(x) = \sum_{i=1}^n y_i H_i(x) + \sum_{i=1}^n y'_i \bar{H}_i(x)$$

- Les H_i interpollent y , les \bar{H}_i interpolent y'
- n contraintes supplémentaires : degré $2n - 1$
- $H_i(x) = (1 - 2(x - x_i)L'_i(x_i)) [L_i(x)]^2$
- $\bar{H}_i(x) = [L_i(x)]^2 (x - x_i)$

Principe de de l'interpolation par *splines*

L'interpolation par des règles de caoutchouc ou de contreplaqué

Principe

- Fonctions d'interpolation définie par morceaux
- Continument dérivable 2 fois

A interpoler

$\forall i \in [1, n], (x_i, y_i)$

- n points, $n - 1$ *splines* $s(x)$ à calculer
- $s(x) = a_i + b_i x + c_i x^2 + d_i x^3$
- $4(n - 1)$ inconnues
- Interpolation par f , n contraintes : $\forall i \in [1, n], f(x_i) = y_i$
- Continuité de f, f' et f'' aux frontières des intervalles :
 $3 * (n - 2)$ contraintes
- $3 * (n - 2) + n = 4n - 6$: il en manque 2 , à inventer

Fonction analytique ou fonction numérique ?

Deux grands classes de fonctions à distinguer

- Celles qui sont définies analytiquement
- Celles qui sont connues uniquement en certains points

Deux grandes classes de méthodes

Les méthodes de dérivations et d'intégration numériques peuvent être différentes dans les deux cas.

Différences finies via le théorème de Taylor

L'application du développement limité au calcul de la dérivée

Théorème de Taylor ordre 1

$$f'(x) = \frac{f(x+h) - f(x)}{h} + o(1)$$

Quelle valeur pour le pas h ?

- Pas trop petit
 - Division par zéro
 - Troncature numérique
- Pas trop grand
 - Troncature du DL

Estimation de l'erreur de troncature

- Deuxième terme du développement
- $\frac{h}{2} f''(x + \theta h) : 0 \leq \theta \leq 1$

Différences finies : les trois formules

Réduction de l'erreur de troncature

- $f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + o(h^2)$
- $f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) + o(h^2)$
- $f'(x) = \frac{f(x+h)-f(x-h)}{2h} + o(h)$

Les trois formules¹⁶

- A droite : $f'(x) = \frac{f(x+h)-f(x)}{h} + o(1)$
- A gauche : $f'(x) = \frac{f(x)-f(x-h)}{h} + o(1)$
- Centrée : $f'(x) = \frac{f(x+h)-f(x-h)}{2h} + o(h)$

¹⁶Il y en a d'autres : imaginez une formule en cinq points pour prendre en compte un terme de plus du DL.

Différences finies : pour aller plus loin

Dérivée première : un ordre de plus dans le DL

- Dérivée centrée avec deux pas différents (e.g. doubles)
- Différenciation pondérée
- ...

Dérivée seconde : une formule centrée ?

- Dérivée à droite
- Dérivée à gauche
- $f''(x) = \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} + o(1)$

Méthode des coefficients indéterminés

Ou comment produire des formules aux différences finies pour une approximation à un ordre arbitraire

Exemple au premier ordre coefficients inconnus : a_0 , a_+ et a_-

- Supposons une approximation de la dérivée
- $f'(x) \approx a_+ f(x+h) + a_0 f(x) + a_- f(x-h)$
- Supposons la relation vérifiée exactement lorsque f est un polynôme
- Trois inconnues, trois polynômes : $1, x$ et x^2
 - $a_0 = 0$
 - $a_+ = -a_- = \frac{1}{2h}$

Aller plus loin ?

- Plus d'inconnues
- Plus de polynômes

Cas d'une fonction connue seulement par ses valeurs en certains points

Si c'est une fonction issue d'un calcul

- Lorsque les points d'échantillonnage sont **équidistants** : méthodes précédentes applicables
- Sinon : il faut avoir recours à l'interpolation

Si ce sont des données expérimentales

- Nécessairement irrégulièrement espacées
- Incertitude sur l'abscisse
- Grosses erreurs sur la dérivée
- Seule méthode possible : dérivation d'une fonction construite
 - Par interpolation
 - Par approximation

Deux grandes classes de méthodes

On cherche à intégrer numériquement f sur l'intervalle $[a, b]$

Les méthodes de Newton-Cotes

- Choix à priori d'un pas fixe h : $nh = (b - a)$
- Interpolation polynomiale
- Intégration du polynôme

L'intégration de Gauss

- Échantillonnage variable
- Algorithme plus performant mais plus complexe

Newton-Cotes : interpolation polynômiale à pas fixe

Pas $h = (b - a)/n$, $x_i = a + ih$

Intégration et interpolation polynômiale

- Interpolation polynomiale par les $n + 1$ points
- Intégration du polynôme

Calcul faisable une fois pour toute : *les nombres de Cotes*

$$\int_a^b f \approx \sum_{i=0}^n h A W_i f(x_i)$$

n	A	W_0	W_1	W_2	W_3	W_4
0	1	1				
1	1/2	1	1			
2	1/3	1	4	1		
3	3/8	1	3	3	1	
4	2/45	7	32	21	32	7

Inconvénient quand n croît

Comportement oscillatoire d'un polynôme de degré élevé

Méthodes composées

Pour augmenter la précision de Newton-Cotes

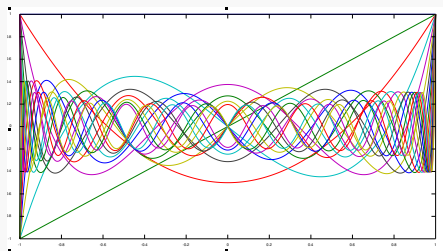
Découpage du domaine en sous-intervalles

- n intervalles
- n interpolation d'ordre 0 : méthode des rectangles
- $n - 1$ interpolations d'ordre 1 : méthode des trapèzes
- $2/n$ interpolations d'ordre 2 : méthode de Simpson
- ... Simpson (ordre 3) et Villarceau (ordre 4)

Prélude : polynômes de Legendre

Polynômes de Legendre

- $P_0(x) = 1$
- $P_1(x) = x$
- $(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)$



Orthogonalité sur $[-1, 1]$

$$\forall m \neq n, \int_{-1}^1 P_n(x) P_m(x) dx = 0 \Leftrightarrow \forall m \neq n, P_n \perp P_m$$

Intégration de Gauss et interpolation de Hermite

A intégrer : f sur $[a, b]$

- Interpolation de Hermite

$$f(x) \approx \sum_{i=1}^n f(a_i) H_i(x) + \sum_{i=1}^n f'(a_i) \bar{H}_i(x)$$

- Intégration terme à terme :

$$\int_a^b f(x) dx \approx \sum_{i=1}^n f(a_i) \mathcal{H}_i(x) + \sum_{i=1}^n f'(a_i) \bar{\mathcal{H}}_i(x)$$

Il reste à annuler le terme contenant la dérivée inconnue

$$\text{Interpolation de Hermite : } \bar{\mathcal{H}}_i(x) = \int_a^b (x - a_i) L_i^2(x) dx$$

Intégration de Gauss : choix des points d'échantillonnage

Des racines...

$$\pi(x) = \prod_{k=0}^n (x - x_k)$$

- A annuler $\tilde{\mathcal{H}}_i(x) = \int_a^b (x - a_i) L_i^2(x) = \int_a^b \pi(x) \frac{L_i(x)}{\pi'(a_i)} dx$
- Si les L_i et π ont les (a_i) comme racines communes, π a une racine (donc un degré) de plus
- Même racines que les polynômes de Legendre : identiques (à un facteur près)
- Si les (a_i) sont racines des polynômes de Legendre : $\pi \perp L_i$

Intégration de Gauss : bilan

Les polynômes de Legendre ne sont définis que sur $[-1, 1]$

Changement de variable $\xi = \frac{1}{2}(a+b) - \frac{1}{2}(a-b)x$

Formule obtenue

- $\int_{-1}^1 f(\xi) d\xi \approx \sum_{i=1}^n f(a_i) \mathcal{H}_i(\xi)$
- $\mathcal{H}_i = \int_{-1}^1 L_i$

Intégration de Gauss : variantes

Méthode précédente inapplicable pour les intervalles ouverts

- On prendra les racines d'autres polynômes orthogonaux
- sur $] -1, 1[$: Chebychev
- sur \mathbb{R}^+ : Laguerre
- sur \mathbb{R}^- : Hermite
- **Modification de la notion d'orthogonalité**

Partie V

Equations Différentielles Ordinaires

V Equations Différentielles Ordinaires

14 Résolution numérique des problèmes différentiels

- Classification
- Forme canonique

15 Problèmes aux conditions initiales

$$y' = f(t, y), y(0) = y_0$$

- Méthode d'Euler
- Méthodes Runge & Kutta
- Méthodes d'Adams

16 Problèmes aux conditions limites

- Méthode de Tir
- Méthode matricielle

Un problème différentiel ordinaire

Équation Différentielle Ordinaire (EDO)

$y(t) \in \mathbb{R}^m$

$$f(t, y, y', \dots, y^{(n)}) = 0$$

Conditions initiales ou conditions aux limites ?

- Un tel problème différentiel n'est pas complètement défini
Il dispose de n degrés de liberté
- Une solution unique est obtenue par ajout de n contraintes

Unicité de la solution indispensable

La résolution numérique est possible seulement si les contraintes
sont posées

Définition des contraintes

Problèmes aux conditions initiales

Problème de Cauchy

- Les contraintes sont imposées en un point unique
- Elles concernent l'inconnue et ses $n - 1$ premières dérivées
- Ce sont les **conditions initiales**

Contraintes imposées en divers points

- Elles peuvent concerner l'inconnue ou ses dérivées
- Elles peuvent être exprimées en plusieurs points
- Elles doivent être au nombre de n

Forme canonique

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

EDO d'ordre 1

- Le problème que nous allons étudier dans cette partie
- Est-il si restrictif ?

Ordre supérieur

$$w^{(n)} = g(t, w, w', \dots, w^{(n-1)})$$

Changement de variable

$$y_i = w^{(i)}$$

Équation d'ordre 1 équivalente

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{bmatrix}' = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-1} \\ g(t, y_0, \dots, y_{n-1}) \end{bmatrix}$$

Dans la suite, nous étudierons et prendrons des exemples pour des inconnues à valeurs réelles, mais l'on pourra généraliser aisément

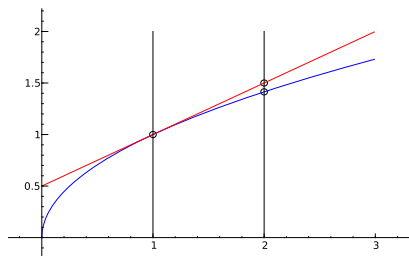
La méthode d'Euler

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

Une méthode simple mais très peu utilisée car imprécise et instable

DL ordre 1 $y(t + \Delta t) = \dots$

- $y(t) + \Delta t y'(t) + o(\Delta t)$
- $y(t) + \Delta t f(t, y) + o(\Delta t)$
- $y^{[i+1]} = y^{[i]} + \Delta t f(t, y^{[i]})$



Erreur de troncature du DL

- $\frac{\Delta t^2}{2!} y'' + \frac{\Delta t^3}{3!} y''' + \dots$
- $\frac{\Delta t^2}{2!} f'(t, y) + \frac{\Delta t^3}{3!} f''(t, y) + \dots$
- **Cumulée à chaque pas**

Méthode d'Euler implicite

Les méthodes implicites sont plus stables que les méthodes explicites

Explicite vs. Implicite ?

- Méthode **exp**licite : $y^{[i]}$ ne dépend que des $y^{[k]}$ pour $k < i$
- Méthode **im**plicite : $y^{[i]}$ ne dépend que des $y^{[k]}$ pour $k \leq i$

Euler implicite

- Explicite : $y'^{[i]} = \frac{y^{[i+1]} - y^{[i]}}{\Delta t}$ dérivée à droite
- Implicite : $y'^{[i+1]} = \frac{y^{[i+1]} - y^{[i]}}{\Delta t}$ dérivée à gauche
- $y^{[i+1]} = y^{[i]} + \Delta t \times f(t, y^{[i+1]})$
- Équation à résoudre à chaque pas
 - Formellement, une fois pour toute si possible
 - Numériquement, valeur de départ fournie par méthode explicite

Les Méthodes Runge & Kutta (RK)

Méthodes dites *A pas unique*

De nombreux avantages

- Faciles à programmer
- Très stables
- Largeur du pas aisément modifiable pour une précision souhaitée
- La condition initiale suffit

Quelques inconvénients

- Pour une même précision, il y a plus économe en temps de calcul
- Instables dans certains cas

Très utilisées en pratique

- Surtout à l'ordre 4

RK2 : Runge & Kutta à l'ordre 2

Centrons les différences finies sur le milieu

Différences finies centrées au point milieu

$$y' = f(t, y) \Rightarrow \frac{1}{\Delta t} (y(t + \Delta t) - y(t)) + o(1) = f\left(t + \frac{\Delta t}{2}, y\left(t + \frac{\Delta t}{2}\right)\right)$$

Allégeons les notations

- $t^{[i]} + \Delta t = t^{[i+1]}$
- $t^{[i]} + \frac{\Delta t}{2} = t^{[i+\frac{1}{2}]}$
- $y(t^{[k]}) = y^{[k]}$

Différences finies centrées au point milieu

$$y' = f(t, y) \Rightarrow \frac{y^{[i+1]} - y^{[i]}}{\Delta t} = f\left(t^{[i+\frac{1}{2}]}, y^{[i+\frac{1}{2}]}\right)$$

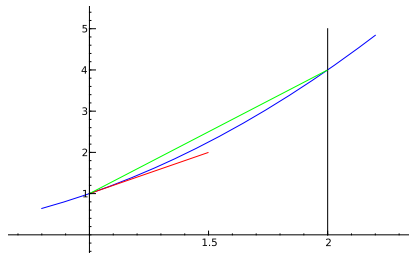
RK2 : Runge & Kutta à l'ordre 2

Différences centrées au point milieu

$$f\left(t^{[i+\frac{1}{2}]}, y^{[i+\frac{1}{2}]}\right) = \frac{y^{[i+1]} - y^{[i]}}{\Delta t}$$

Estimation de $y^{[i+1/2]}$ par Euler

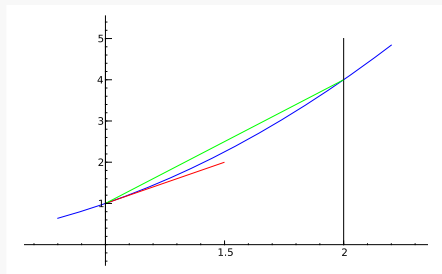
- $y^{[i+1/2]} = y^{[i]} + \frac{\Delta t}{2} f(t^{[i]}, y^{[i]})$
- $y^{[i+1]} = y^{[i]} + \Delta t \times f\left(t^{[i+\frac{1}{2}]}, y^{[i+1/2]}\right)$



Formule à l'ordre 2

RK2 : formulation standardisée

- $r_1 = f(t^{[i]}, y^{[i]})$
- $r_2 = f\left(t^{[i+\frac{1}{2}]}, y^{[i]} + \frac{\Delta t}{2} r_1\right)$
- $y^{[i+1]} = y^{[i]} + \Delta t \times r_2$



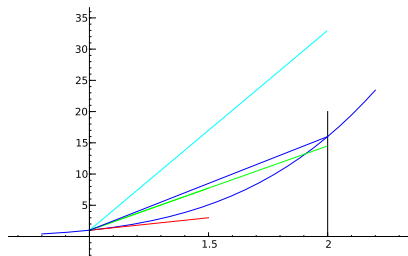
Runge & Kutta à l'ordre 4

LA méthode reine

RK4

RK4 est obtenue par pondération successive des dérivées
à droite, à gauche et centrée

- $r_1 = f(t^{[i]}, y^{[i]})$
- $r_2 = f\left(t^{[i+\frac{1}{2}]}, y^{[i]} + \frac{\Delta t}{2} r_1\right)$
- $r_3 = f\left(t^{[i+\frac{1}{2}]}, y^{[i]} + \frac{\Delta t}{2} r_2\right)$
- $r_4 = f(t^{[i+1]}, y^{[i]} + \Delta t \times r_3)$
- $y^{[i+1]} = y^{[i]} + \frac{\Delta t}{6} (r_1 + 2r_2 + 2r_3 + r_4)$



Les méthodes d'Adams ouvertes

Des méthodes à pas multiples

Adams-Bashforth

(nécessitant plusieurs points précédents)

Principe : encore une fois le Développement Limité

- Développement limité
- Différences finies
- Ordre augmenté en augmentant l'ordre du DL
- Le calcul de $y^{[i]}$ nécessite $y^{[i-1]}, y^{[i-2]} \dots$

Développement limité à droite

Différences finies à gauche

- $y(t + \Delta t) = y(t) + \Delta t \times y'(t) + \frac{\Delta t^2}{2!} y''(t) + \frac{\Delta t^3}{3!} y'''(t) + \dots$
- $y^{[i+1]} = y^{[i]} + \Delta t \times f^{[i]} + \frac{\Delta t^2}{2!} f'^{[i]} + \frac{\Delta t^3}{3!} f''^{[i]} + \dots$

En notant $f(t_i, y^{[i]}) = f^{[i]}$

- Dérivées évaluées par différences finies à gauche
Calcul simple mais fastidieux

Les méthodes d'Adam ouvertes : exemples

Ordres 1 à 2

Ordre 1

Euler

$$y^{[i+1]} = y^{[i]} + \Delta t \times f^{[i]} + o(\Delta t)$$

Ordre 2

- $y^{[i+1]} = y^{[i]} + \Delta t \times f^{[i]} + \frac{\Delta t^2}{2!} f'^{[i]} + o(\Delta t^2)$
- $f'^{[i]} = \frac{f^{[i]} - f^{[i-1]}}{\Delta t} + o(\Delta t)$
- $y^{[i+1]} = y^{[i]} + \frac{\Delta t}{2} (3f^{[i]} - f^{[i-1]}) + o(\Delta t^2)$

Démarrage de la méthode

- Nécessité de trouver un premier point par une autre méthode
d'ordre au moins égal
- Par exemple : RK2

Méthode ouverte d'Adams à l'ordre 3

Encore une fois pour être sûr de bien comprendre

Expression du DL à droite, compte tenu de l'équation différentielle

$$y^{[i+1]} = y^{[i]} + \Delta t \times f^{[i]} + \frac{\Delta t^2}{2!} f'^{[i]} + \frac{\Delta t^3}{3!} f''^{[i]} + o(\Delta t^3)$$

$f'^{[i]}$

Formule en $o(\Delta t)$

$$\frac{3f^{[i]} - 4f^{[i-1]} + f^{[i-2]}}{2\Delta t} + o(\Delta t)$$

$f''^{[i]}$

$$\frac{f^{[i]} - 2f^{[i-1]} + f^{[i-2]}}{\Delta t^2} + o(1)$$

Formule d'Adams ouverte à l'ordre 3

$$y^{[i+1]} = y^{[i]} + \frac{\Delta t}{12} [23f^{[i]} - 16f^{[i-1]} + 5f^{[i-2]}] + o(\Delta t^3)$$

Méthodes d'Adams fermées

Des méthodes implicites

Adams-Moulton

Développement limité à gauche

Différences finies à gauche

- $y(t) = y(t + \Delta t) - \Delta t \times y'(t + \Delta t) + \frac{\Delta t^2}{2!} y''(t + \Delta t) - \frac{\Delta t^3}{3!} y'''(t + \Delta t) + \dots$
- $y^{[i]} = y^{[i+1]} - \Delta t \times f^{[i+1]} + \frac{\Delta t^2}{2!} f'^{[i+1]} - \frac{\Delta t^3}{3!} f''^{[i+1]} + \dots$
- $y^{[i+1]} = y^{[i]} + \Delta t \times f^{[i+1]} - \frac{\Delta t^2}{2!} f'^{[i+1]} + \frac{\Delta t^3}{3!} f''^{[i+1]} + \dots$
- Dérivées évaluées par différences finies à gauche

On a besoin du résultat pour avoir le résultat ...

?

- Estimation du résultat par une méthode ouverte (prédiction)
- Correction par une méthode fermée d'ordre au moins égal
- Itération jusqu'à obtention de la précision voulue

Adams-Moulton : deux exemples

Ordres 1 & 4

Ordre 1

Euler implicite

$$y^{[i+1]} = y^{[i]} + \Delta t \times f^{[i+1]} + o(\Delta t)$$

Ordre 4

La plus utilisée en pratique

$$y^{[i+1]} = y^{[i]} + \frac{\Delta t}{24} [9f^{[i+1]} + 19f^{[i]} - 5f^{[i-1]} + f^{[i-2]}] + o(\Delta t^4)$$

Méthodes de prédiction correction

En pratique : combinaison des méthodes d'Adams ouvertes et fermées

Avantages

- Stabilité des méthodes implicites
- Quantité de calcul proche des méthodes explicites

Principe

- | | |
|--|-----------------|
| • Estimation par une méthode ouverte | Adams-Bashforth |
| • Correction par une méthode fermée | Adams-Moulton |
| • d'ordre au moins égal | |
| • en pratique : même ordre ou un de plus | |

Méthode de Tir

Résolution itérative des problèmes à conditions limites

Principe

- Transformer en un problème à conditions initiales
- Une condition inconnue
- Trouver l'inconnue



Fig.: Source de la Photo : Ray Bilcliff,
<http://www.trueportraits.com>

Méthode de Tir pour une équation du second ordre

Problème à résoudre

- Équation du second ordre : deux degrés de liberté
$$A(x, y) y'' + B(x, y) y' + C(x, y) y = D(x, y)$$
- Deux contraintes sous forme de conditions aux limites
$$y(0) = y_0 \text{ et } y(L) = y_L$$

Transformation en un problème aux conditions initiales

- $y'(0) = u$ inconnue : conditions initiales fixées
- Pour u donné, une estimation de y est calculée : y_u
- Au point L , sa valeur est $y_u(L)$
- C'est une fonction de u , nommons là $y_L : u \mapsto y_L(u)$
- Le problème différentiel devient une recherche de racine
$$y_L(u) = y_L$$

Recherche de la racine

Trouver u , c'est résoudre le problème

$$y_L(u) = y_L$$

Méthode de l'artilleur

- Je tire un coup au dessus
- Je tire un coup au dessous
- J'ajuste le tir
- Je recommence si nécessaire
- C'est la **Méthode de la sécante** [► Voir chapitre précédent](#)

Application dans notre cas

- Résoudre pour deux valeurs arbitraires $u^{[1]}$ et $u^{[2]}$
- $u^{[3]}$ sera la valeur trouvée pour u en interpolant y_L linéairement entre $u^{[1]}$ et $u^{[2]}$
- $u^{[n+1]} = \frac{[y_L(u^{[n]}) - y_L]u^{[n-1]} - [y_L(u^{[n-1]}) - y_L]u^{[n]}}{[y_L(u^{[n]}) - y_L(u^{[n-1]})]}$

Et si les conditions aux limites impliquent la dérivée ?

Supposons des conditions aux limites linéaires

- $\alpha_0 y(0) + \alpha_1 y'(0) = c_0$
- $\beta_0 y(L) + \beta_1 y'(L) = c_L$

Remplaçons par un problème à conditions initiales

- $\alpha_0 y(0) + \alpha_1 y'(0) = c_0$
- $\gamma_0 y(0) + \gamma_1 y'(0) = u$
- En imposant $\alpha_0 \gamma_1 - \alpha_1 \gamma_0 = 1$:
 - $y(0) = -\alpha_1 u + c_0 \gamma_1$
 - $y'(0) = \alpha_0 u - c_0 \gamma_0$

Méthode de la sécante

- Évaluation de y_u pour $u^{[1]}$ et $u^{[2]}$ arbitraires
- Méthode de la sécante pour résoudre $\beta_0 y_u(L) + \beta_1 y'_u(L) = c_L$

Équation du second ordre linéaire

Équation différentielle linéaire du second ordre

- Les coefficients ne dépendent que de x
- $A(x)y'' + B(x)y' + C(x)y = D(x)$
- La combinaison linéaire de deux solutions est solution

Cas particulier de la méthode de la sécante

- L'interpolation linéaire est **exacte**
- Un pas suffit
- $u^{[3]}$ est la solution

Équations d'ordres supérieurs

ordre n

Équivalence équation premier ordre

Équation **vectorielle**, ordre 1, n composantes

Les n conditions limites

- n_0 d'entre elles sont en x_0 : $c_0 = (c_{0,1}, \dots, c_{0,n_0})$
- $n_L = n - n_0$ d'entre elles sont en x_L : $c_L = (c_{L,1}, \dots, c_{L,n_L})$

Transformation en problème à conditions initiales en x_0

- n_0 premières conditions connues, n_L suivantes inconnues
- Posons $\mathbf{u} = (c_0, c_{\mathbf{u}})$ et $e(\mathbf{u}) = c_L(\mathbf{u}) - c_L$

Méthode de Newton discrétisée

► Voir chapitre précédent

$$e(\mathbf{u}) = 0$$

$$e(\mathbf{u} + \delta \mathbf{u}) \approx e(\mathbf{u}) + D_e^u \delta \mathbf{u} = 0 \Rightarrow \mathbf{u}^{[n+1]} = \mathbf{u}^{[n]} - (D_e^u)^{-1} \delta \mathbf{u}$$

Estimation numérique de D_e^u car dérivation impossible

EDO linéaires à conditions limites : méthode matricielle

Aussi connue sous les noms de *méthode de relaxation* ou *méthode aux différences finies*

Une EDO linéaire discrétisée est un système linéaire

- Différences finies en chaque point : n équations
- La valeur de y en chaque point : n inconnues

Résolution du système

- Les différences finies ne font référence qu'aux voisins
- C'est un système multidiagonal, donc creux
- Méthodes itératives et matrices creuses pour n grand

Mais attention à la troncature du Développement Limité

- Il est prudent de comparer la solution obtenue en doublant n
- Et de le doubler encore si nécessaire

Méthode matricielle pour les EDO non linéaires

Le système à n équations et n inconnues n'est plus linéaire

Résolution d'un système non linéaire

- Un air de déjà vu
- Méthode de Newton multidimensionnelle vue précédemment

► Ici