
Cours Ensimag

Visualisation Scientifique 3D

Stefanie Hahmann

Laboratoire Jean Kuntzmann

Georges-Pierre Bonneau

Laboratoire Jean Kuntzmann / INRIA

<<Imagination or visualization and in particular the use of diagrams has a crucial part to play in scientific investigations.>>

René Descartes (1637)

Contenue du cours

1. La boucle de la découverte scientifique
 - 1.1 Flux d'information pour la ViSc
 - 1.2 Exemple: Simulation Numérique
 - 1.3 Imagerie médiacle
 - 1.4 Modélisation de scattered data
 - 1.5 Besoin en puissance de calcul et de réseau
2. Les données
 - 2.1 Exemple
 - 2.2 Classification
 - 2.3 Structures de données
3. Modélisation de "scattered data"
 - 3.1 Méthodes de base pour l'interpolation de SD
 - 3.2 Modélisation de SD du type "surface-sur-une-surface"
4. Visualisation de données "surface-sur-une-surface"
5. Visualisation de données volumiques
 - 5.1 Méthodes de décomposition du domaine
 - 5.2 Méthodes de slicing
 - 5.3 Iso-surfaces - Contouring
 - 5.4 Volume rendering - Ray-casting
6. Visualisation de champs de vecteurs et de tenseurs
 - 6.1 Méthodes directes
 - 6.2 Méthodes topologiques
 - 6.3 LIC
7. Visualisation multirésolution de données scientifiques
 - 7.1

URL's

Bibliographie

Qu'est-ce que la Visualisation Scientifique ?

- est l'utilisation d'images créées par ordinateur afin de comprendre les données d'origine de mesures ou de simulation.
- elle permet à un expert d'appréhender de manière synthétique et pertinente un ensemble de données sous forme d'images.
- est un important nouveau domaine de recherche qui est utilisé par différentes sciences de l'ingénieur et d'informatique.
(1.conférence en 1991, 1.revue international en 1996)
- un grand nombre de méthodes en ViSc, pratiquement une méthode différente par type de données et domaine d'application.

1. LA BOUCLE DE LA DÉCOUVERTE SCIENTIFIQUE

Comment la ViSc peut être utilisée pour améliorer le processus de la découverte scientifique ?

Pour introduire ce cours, on va d'abord parler un peu de ce qu'est la boucle de la découverte scientifique. Le processus classique du calcul scientifique (modélisation mathématique du réel physique, simulation numérique, visualisation) y est inclus.

Cela va être expliqué avec 3 exemples prototypes qui illustrent le context scientifique de la visualisation et les problèmes qui se posent. Ce sont 3 exemples dans lesquels se retrouvent la plupart des techniques et algorithmes dont on va faire connaissance dans ce cours.

L'importance particulière est portée sur des applications des modèles physiques en 3D. Cela inclut la classes de problèmes

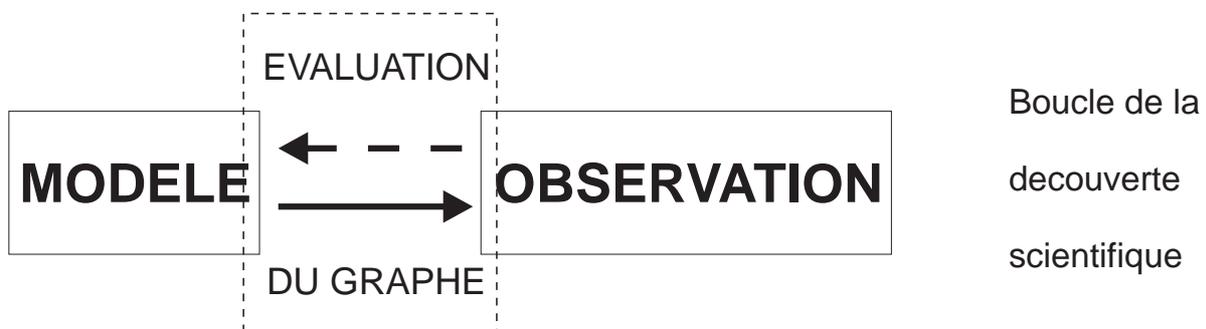
les plus intéressants, c.a.d. la modélisation et la compréhension de phénomènes de notre monde réel.

Mots clés

- **Interaction:** on va dans ce chapitre donner une introduction à la ViSc interactive. On est le plus souvent intéressé au cas où l'utilisateur est en interaction avec le modèle mathématique et dirige la simulation dans l'espoir de "gagner new insight".
- **Comprendre l'intérieure.**
- **Modification ou amélioration du modèle.**

L'interface primeur pour l'interaction consiste en

- l'image visuelle *et en conséquent*
- l'input basé sur la perception et l'interprétation des images par l'utilisateur.



Modèle: Le modèle ici peut soit être un modèle mathématique d'un phénomène physique que l'on veut simuler, donc un modèle abstrait; soit être un modèle physique sur lequel on veut faire des observations, lequel on voudrait bien construire ou améliorer.

MODELE

La boucle de base consiste donc en un modèle \rightarrow à partir duquel on obtient des données (soit un très grand nombre, soit très compliqué, ou les deux) qui eux seules nous disent rien.

OBSERVATION

Ici intervient donc la visualisation qui nous crée une image. On comprend mieux ce qui se passe. On peut comprendre l'ensemble de données numériques obtenues auparavant. On fait l'observation et \leftarrow en conséquent on fait peut-être des modifications/améliorations au modèle.



*On comprend donc la nécessité d'une interface (en 2 directions) qui est la phase intermédiaire entre le modèle et l'observation, dont on extrait des informations qui forment l'image. C'est la phase que l'on appelle "**évaluation du graphe**". Très souvent on aura à calculer le graphe d'une fonction qui ensuite va être visualisée d'une manière appropriée à l'écran d'ordinateur.*

Nous n'allons pas nous occuper, comment une image apparaît réellement sur l'écran (avec les projections 3D \rightarrow 2D, la perspective, les surfaces et lignes cachées. avec l'illumination, l'ombrage, le rendu réaliste, etc). C'est le contenu d'un cours de base sur l'infographie.

Notre problème est la visualisation d'un ensemble de données de taille et de type très différent en 3D. Il n'y a pas de technique qui s'applique dans tous les cas. Elle dépend complètement du modèle du phénomène à étudier et des informations que l'on souhaite à extraire

des ensembles de données.

P.ex.:

Comment visualiser de la pluie sur une carte? ou comment visualiser qu'il y avait 52 l/m^2 de pluie en Auvergne à midi?

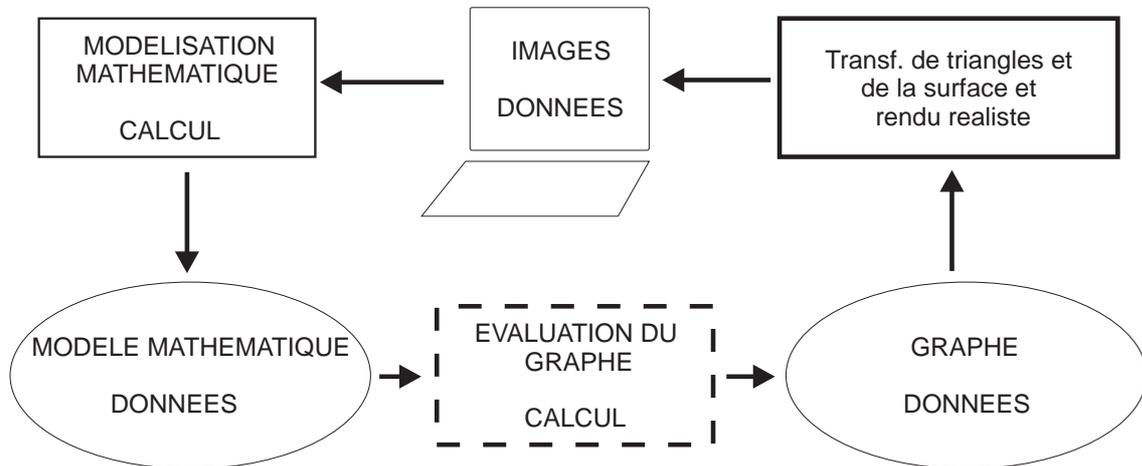
- *dessiner un nuage et ' 52 l/m^2 ' sur la carte,*
- *attribuer une couleur à cette endroit en fonction des l/m^2 tombés,*
- *tracer des histogrammes,*
- *faire une animation au cours de la journée.*

Le problème n'est donc pas comment tracer sur l'écran un nuage de pluie mais l'algorithme ou la technique qui à partir d'un fichier de données enregistrées nous finalement dit qu'il faudra tracer un nuage de pluie au Puy de Dôme sur la carte de France.

Et pour réellement faire de la ViSc il faut finalement quand même savoir calculer une image par ordinateur.

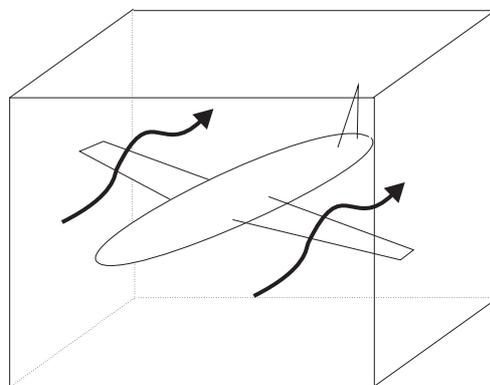
Pour la plus grande partie, les techniques de la ViSc reviennent d'une manière ou d'une autre à calculer une fonction et à évaluer son graphe à partir d'un ensemble de données souvent discrètes.

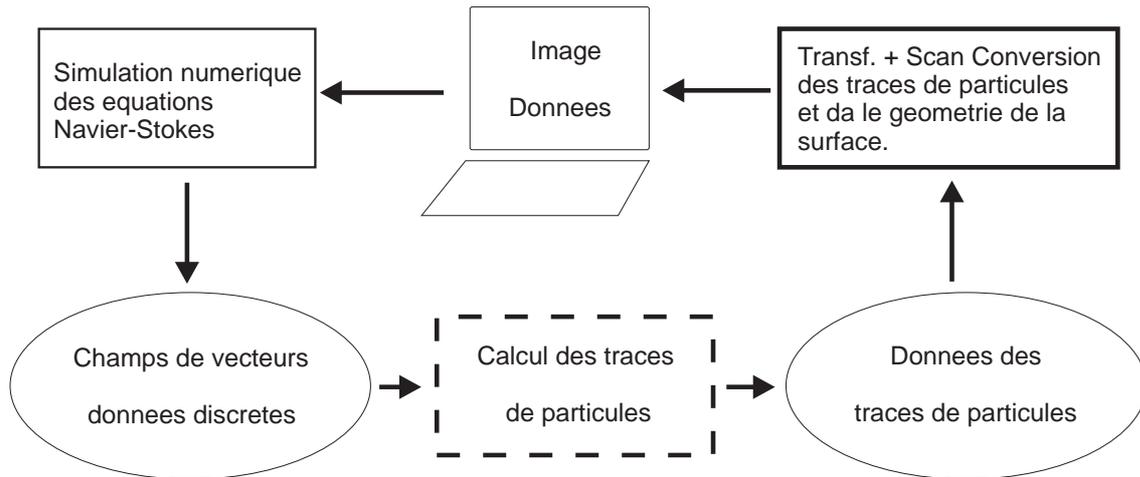
1.1 FLUX D'INFORMATION POUR LA VISC



1.2 EXEMPLE: SIMULATION NUMERIQUE

— Design d'une aile d'avion—





Modélisation mathématique d'un phénomène physique

- mettre en équation les lois de la physique: ici de la dynamique des fluides

⇒ **équations de Navier-Stokes** (EDP)

- mettre en relation le vecteur de velocity $\vec{\mathbf{V}} = (u, v, w)$, le scalaire de la pression p , une constante de densité du fluide ρ , une constante de viscosité dynamique μ et le vecteur des forces extérieures $\vec{\mathbf{F}} = (x, y, z)$

$$\rho \left[\frac{\partial \vec{\mathbf{V}}}{\partial t} + (\vec{\mathbf{V}} \cdot \nabla) \vec{\mathbf{V}} \right] = -\nabla p + \mu \nabla^2 \vec{\mathbf{V}} + \vec{\mathbf{F}}.$$

Ensemble avec une équation de continuité

$$\text{div} \vec{\mathbf{V}} = 0.$$

Il s'agit de résoudre ces équations pour $\vec{\mathbf{V}}$ et p en fonction du temps t et les coordonnées de l'espace (x, y, z) .

Il faut d'abord fixer les **conditions de bord** sur la surface S de l'aile. Elles sont très importantes. Dans ce cas elles doivent correspondre à la condition que la vitesse du fluide doit être égale à la vitesse de l'avion $\vec{\mathbf{A}}$ à sa surface et la vitesse du fluide loins de

l'avion doit approximer les condition d'un flux libre.

($\frac{d\vec{A}}{dt}(t)$ vitesse de l'avion)

$$\vec{V}(x, y, z, t) = \frac{d\vec{A}}{dt}(t) = 0 \quad \text{pour } (x, y, z) \in S.$$

⇒ Il faut une description mathématique du “bord”, qui est dans beaucoup de cas un objet géométrique.

Ici: surface de l'aile.

On utilise des surfaces de forme libre, p.ex. les surfaces B-spline trimmées:

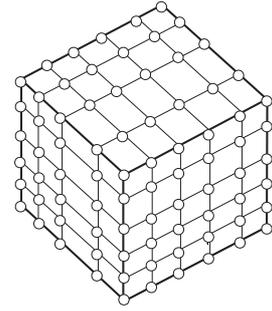
$$S(u, v) = \sum \sum \mathbf{b}_{ij} N_i^n(u) N_j^m(v).$$



Résolution de l'EDP par des méthodes numériques (en général il n'existe pas de solutions explicites). Elles fournissent les valeurs de \vec{V} et p en quelques points discrets:

(a) par *différences finies*:

utiliser une grille uniforme de l'espace. L'EDP est remplacée par une approximation où les dérivées partielles sont approximées par des différences



$$u_x \simeq \frac{1}{2}(u_{i+1,j,k} - u_{i-1,j,k})$$

$$u_y \simeq \frac{1}{2}(u_{i,j+1,k} - u_{i,j-1,k})$$

$$u_z \simeq \frac{1}{2}(u_{i,j,k+1} - u_{i,j,k-1})$$

$$\nabla^2 u = u_{xx} + u_{yy} + u_{zz} \simeq (u_{i+1,j,k} - 2u_{i,j,k} + u_{i-1,j,k}) + (\quad) + (\quad)$$

avec $u_{i,j,k} = u(x_i, y_j, z_k)$, (x_i, y_j, z_k) étant un point de la grille.

(b) par *éléments finis*

La solution est supposée d'être sous la forme d'une combinaison linéaire de fonctions b_i définies par morceaux sur une décomposition de l'espace et qui n'ont qu'un petit support (i.e. support local).

$$u(x, y, z) = \sum_{i=1}^n a_i b_i(x, y, z) \quad (*)$$

Calculer les coefficients a_i en exigeant p.ex. que (*) satisfait l'EDP en n points discrets.

⇒ système d'équations linéaires creux: en 3D 100^3 équations et 100^3 inconnues.

⇒ méthodes itératives (gradient conjugué)

⇒ **Champs de vecteurs, données discrètes**

Evaluation du graphe

préparer les données (champs de vecteurs 3D)

$$\vec{\mathbf{V}}(x, y, z) = \begin{bmatrix} u(x, y, z) \\ v(x, y, z) \\ w(x, y, z) \end{bmatrix}, \quad (x, y, z) \in D,$$

connues sur une grille (x_i, y_j, z_k) , $i = 1, \dots, N_x$, $j = 1, \dots, N_y$, $k = 1, \dots, N_z$, pour la visualisation: approximation, interpolation, analyse multi-résolution.

Choix de la méthode de visualisation !

→ **hedgehog** (hérisson): tracer une collection de flash/vecteurs en les points de la grille. C'est une méthode très limitée pour apprendre qc d'un champs en 3D.

→ **Animation** des particules suivant les lignes de courant : courbes 3D tangentes au champs $\vec{\mathbf{V}}$ (**streamlines**).

$$\mathbf{P}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}$$

i.e. $\mathbf{P}'(t) = (x'(t), y'(t), z'(t))^T = \vec{\mathbf{V}}(x(t), y(t), z(t)) = \vec{\mathbf{V}}(\mathbf{P}(t))$.

i.e. résoudre numériquement une EDO (Runge Kutta).

→ **déformation** du type FFD

Affichage à l'écran - display

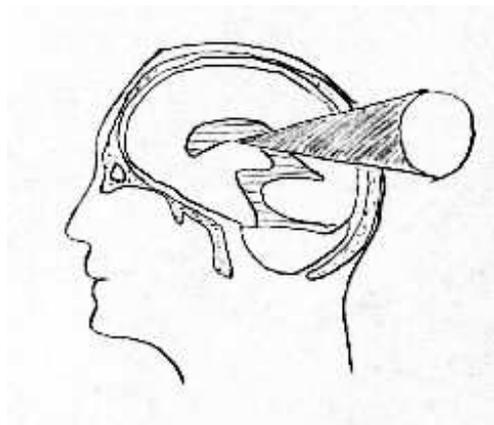
La phase d'évaluation du graphe produit en général des "primitives graphiques" qui sont ensuite affichés par un grand nombre de techniques disponibles à l'écran (voir techniques en infographie, synthèse d'images) [Folvdam??].

ici: **Animation d'une trace de particules autour de l'aile.**

- Transformation et rendu de la trace des particules et des bord polygonaux (triangles, quadrilatères) de la surface de l'aile
- les polygones passent par une "viewing pipeline":
coordonnées 3D du domaine de modélisation ⇒
coordonnées 2D de l'écran ⇒
colorage des pixels à l'intérieur du polygone d'après une méthode d'illumination et d'ombrage, calcul des surfaces cachées, clipping des primitives.

1.3 EXEMPLE 2: IMAGERIE MEDICALE

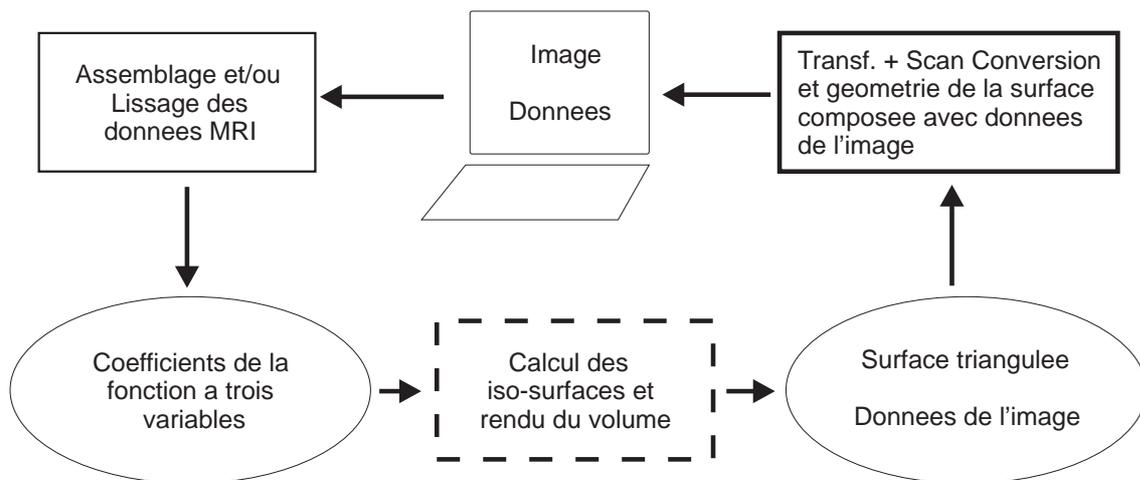
Rayonnement d'une tumeur (radiothérapie)



Le médecin doit positionner l'appareil pour qu'il cible la tumeur et ne détruise pas trop de cellules saines autour.

Le médecin doit avoir connaissance de l'endroit exacte d'une tumeur.

Les méthodes de la visualisation volumique permettent un regard à l'intérieur non-chirurgical pour trouver l'endroit à traiter. Une simulation interactive du traitement permet de fixer les paramètres du traitement (direction, focus, intensité, etc.) pour que l'exposition du patient au rayonnement soit minimisée.



Modélisation mathématique

Techniques fournissant des données pour l'imagerie médicale:

- PET positron emission tomography
- MRI magnetic resonance imaging
(IRM Imagerie par résonance magnétique)
- Ultra-son

Les données: $(x_i, y_j, z_k; F_{ijk}),$ $i = 1, \dots, N_x$
 $j = 1, \dots, N_y$
 $k = 1, \dots, N_z,$

$\mathbf{V}_{ijk} = (x_i, y_j, z_k)$ positions des cotes de données dans l'espace 3D.

Déterminer une fonction trivariate $F(x, y, z)$ telle que

$$F(\mathbf{V}_{ijk}) = F_{ijk} \quad (\text{interpolation}), \text{ ou}$$

$$F(\mathbf{V}_{ijk}) \approx F_{ijk} \quad (\text{approximation})$$

- \mathbf{V}_{ijk} se trouvent sur une **grille cube** (grille régulière)
 $\Rightarrow F$ trilinéaire par morceaux sur chaque voxel de la grille (ou produit tensoriel triple de splines cubiques \rightarrow plus lisse).
- \mathbf{V}_{ijk} **non-structurés**
 \Rightarrow Méthodes d'interpolation de "scattered data".

Evaluation du graphe

- (a) **Calcul d'iso-surfaces** (surfaces d'iso-valeurs) de la fonction modélisant $F(x, y, z)$

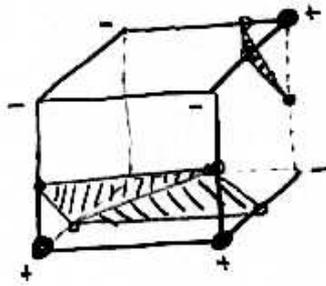
$$S_\alpha = \{(x, y, z) | F(x, y, z) = \alpha\}.$$

C'est la version 3D des lignes de contour, p.ex. les lignes de niveau de pression constante sur une carte météo.

- S_α se compose souvent en plusieurs morceaux de surface.
- S_α est calculée approximativement et représentée par un ensemble de triangles:

Algorithme "Marchign cubes"

- évaluer F en $x_i, y_j, z_k \Rightarrow F_{ijk} > \alpha$ ou $< \alpha$
 - calculer l'intersection de S_α avec les arêtes en supposant qu'elle ne varie que linéairement sur les arêtes
- \Rightarrow polygones, les trianguler
 \Rightarrow iso-surface triangulée (tri-linéaire par morceaux).

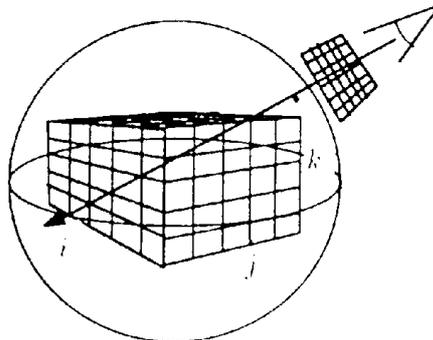


en sortie de l'algorithme: **Liste de triangles surfaciques.**

Affichage à l'écran pour visualiser l'iso-surface en 3D.

(b) **Rendu volumique** (volume rendering)

Technique permettant de regarder le volume en une seule image, ressemblant aux radiographiques, pour regarder à l'intérieur d'un objet.



Ray casting volume rendering

Algorithme:

Lancer un rayon à partir de votre point de vue à travers d'un pixel de l'écran dans le volume.

Le rayon rentre dans le volume, traverse plusieurs voxel et y calcule des valeurs le long du rayon.

Une "fonction de transfert" les compose pour attribuer une couleur à ce pixel à l'écran.

Répétition pour chaque pixel.

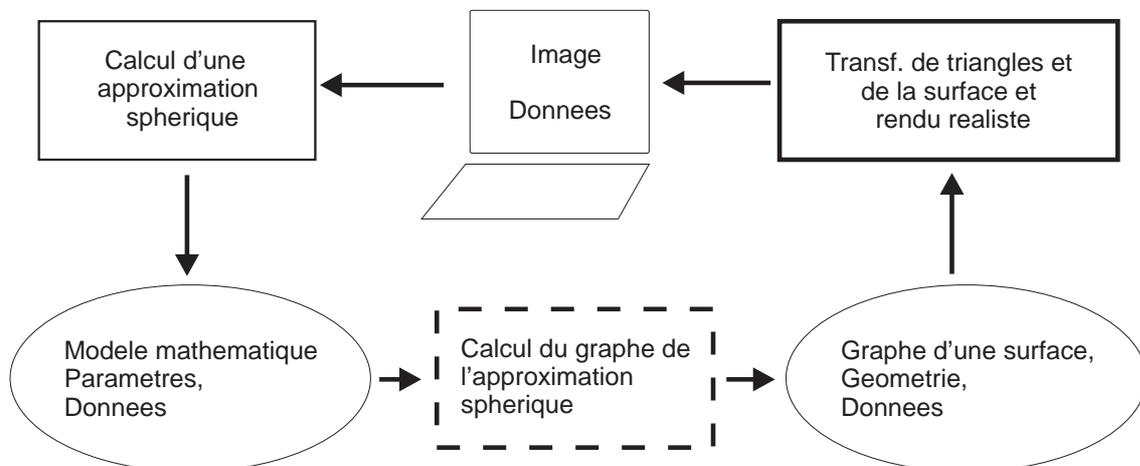
- très efficace
- très cher en temps de calcul
- calcul en temps réel souhaitable.

1.4 EXEMPLE 3: MODELISATION DE SCATTERED DATA

Surveillance de concentration de CO₂ dans l'air



- données mesurées en intervalles irréguliers dans une période de temps.
- préparer les données pour éventuellement pouvoir détecter des comportements globaux.
- visualiser des fonctions sur la terre.



Modélisation mathématique

la terre $S = \{(x, y, z) | x^2 + y^2 + z^2 = 1\}$ sphère unité.

données: $\mathbf{P}_i \in S$ et valeurs F_i en les \mathbf{P}_i , $i = 1, \dots, N$.

on cherche une fonction $F : S \rightarrow \mathbb{R}$ qui interpole ou approxime les valeurs sur la sphère S : $F(\mathbf{P}_i) = F_i$.

⇒ **surface-on-surface.**

Evaluation du graphe

- calculer le graphe tel qu'il est représentée sur l'image précédente: fonction à valeur réelle sur une spère.
prendre chaque pixel dans l'espace image, déterminer ou il se trouve sur la spère, et évaluer la fonction en ce point pour lui associer une couleur. ! trop cher en temps de calcul !
- calculer des approximations polygonales des courbes de contour

$$C_\alpha = \{(x, y, z) | F(x, y, z) = \alpha\}.$$

Représenter le domaine S par des triangles et supposer que F est très simple (ex. linéaire) sur les arêtes du triangle.

⇒ les courbes de contour sont polygonales.

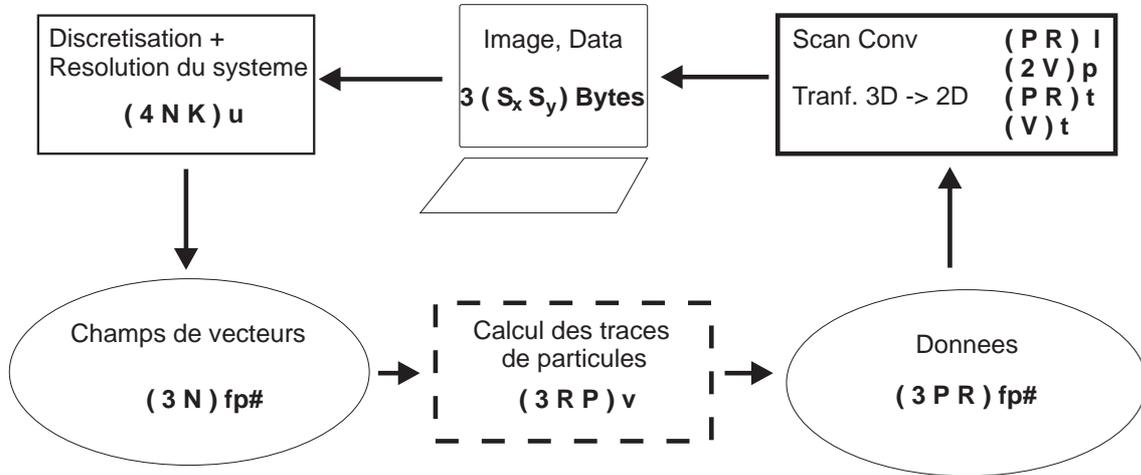
! très utile ! mais certains propriétés de la fonction F peuvent rester invisibles: cercles cocentriques, etc.

⇒ tracer la surface F semi-transparent avec différentes couleurs + lignes de contour sur la terre en arrière-plan.

Extension: surface-sur-une-surface arbitraire.

1.5 BESOIN EN PUISSANCE DE CALCUL ET DE RESEAU

Exemple: Navier-Stokes (Simulation aile d'avion)



Exemple: Navier-Stokes:

N	$N_x = N_y = N_z = 100$	$N = 10^6$	} 1G flop
K	itérations pour convergence	$K = 10$	
u	opérations par itération	$u = 25 flop$	
4	valeurs non-nulles par équation	4	

(3 N) 32 bits } 96 M bits

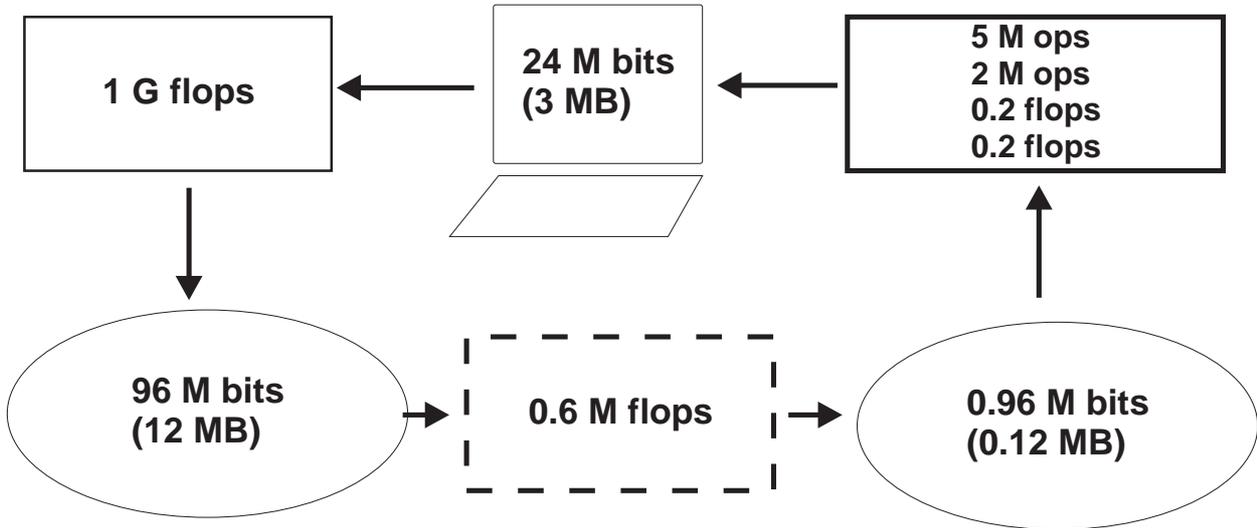
R	# pas du EDO-solver	$R = 100$	} 0.6M flop
P	# patches	$P = 100$	
v	# opérations par pas	$v = 20 flop$	

(3 P R) 32 bits } 0.96 M bits

V	# sommets de la triangulation surfacique	$V = 10.000$	} 7.4M ops
$2V$	# triangles		
l	# op: convertir triang en lignes	$l = 500 ops$	
p	# op:	$p = 1000 ops$	
t	# op: transf. + clipping	$t = 20 flop$	

S_x taille de l'image
 S_y

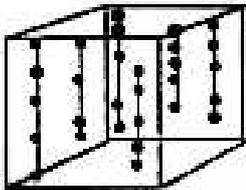
$S_x = 1000$
 $S_y = 1000$
3 RGB couleur
8 bits = 1 Byte } **24 Mbits**



2. LES DONNÉES

2.1 EXEMPLES

- **WELL LOG DATA** (Courtesy D.Lane & D.Krinsel)



	Location			Mineral
	5.50	1.00	0.00	11.0
	5.50	1.00	10.00	10.0

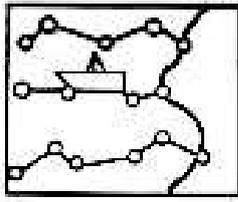
Géophysique: collectionner des mesures de différentes profondeurs à certaines positions fixes.

positions $(x_i, y_i) \in$ surface de la terre

la profondeur et le nombre de mesures peuvent varier d'une position à l'autre.

Données: $(x_i, y_i, z_{ij}; M_{ij}) \quad i = 1, \dots, N, \quad j = 1, \dots, N_i.$

- **BIG SUR** (Courtesy R.Franke)



	Location		Temperature
	23.56	37.80	27.3
	49.29	54.78	69.2
	67.24	43.42	10.2
	.	.	.
	.	.	.
	.	.	.
	.	.	.

Track Data: $(x_{ij}, y_{ij}; T_{ij}) \quad i, j = 1, \dots, N, M.$

• SCANNEUR MÉDICAL



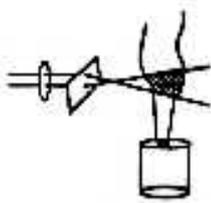
X_i	Y_j	Z_k	Density
0.000	0.000	0.000	243
0.000	0.000	0.015	175
.	.	.	.
.	.	.	.
0.000	0.000	1.000	186
0.000	0.016	0.000	187
.	.	.	.
.	.	.	.
.	.	.	.

Données: $F_{ijk} = F(x_i, y_j, z_k)$ $i, j, k = 1, \dots, N$.

Les sites des données se trouvent sur une grille cubique uniforme, i.e. équidistante dans chacune des 3 directions x, y, z .

Origine: MRI, PET, Ultra-son, ...

• FLAMME



Location			Concentration
0.00	0.00	0.02	001
0.00	0.00	0.04	007
0.00	0.00	0.06	003
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Données représentant des concentration de gaz dans un haut-fourneau. Pour chaque couche z_k la flamme est scannée par un laser qui radialement détermine les positions et permet d'effectuer les mesures de concentration

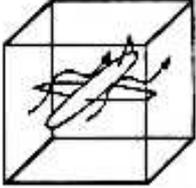
$$(r_i \cos(\Phi_j), r_i \sin(\Phi_j), z_k; C_{ijk})$$

$$r_i = r_{min} + i\Delta r$$

$$\Phi_j = \Phi_{min} + j\Delta\Phi$$

$$r_k = r_{min} + k\Delta z$$

• **AILE D'AVION** (Courtesy Nasa Ames)



	Location			Pressure
	-132.1	38.5	6.1	0.164
	-128.3	38.5	6.6	0.119
	-116.8	38.5	7.5	0.067

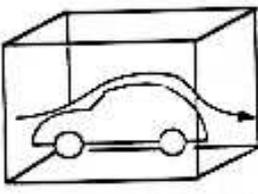
Simulation numérique de la pression sur une aile d'avion.

Données: $(x_{ij}, y_{ij}, z_{ij}; P_{ij}), \quad i = 1, \dots, N_u,$
 $j = 1, \dots, N_v,$

$$(x_{ij}, y_{ij}, z_{ij}) = W(u_i, v_j),$$

$W(u, v)$ surface paramétrique de l'aile.

• **VOITURE** (Courtesy Y.Nakajima, Nissan)

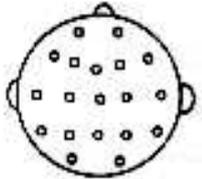


	Location			Velocity
	7.77	9.45	3.85	(1.33 2.34 0.45)
	4.14	-2.78	2.68	(1.86 3.56 1.25)

Simulation Numérique: Vitesse par Navier-Stokes

Données: $(x_{ijk}, y_{ijk}, z_{ijk}; (u_{ijk}, v_{ijk}, w_{ijk})) \quad i = 1, \dots, N_x,$
 $j = 1, \dots, N_y,$
 $k = 1, \dots, N_z.$

• CERVEAU



	Location			Voltage
	6.54	4.56	5.64	0.033
	9.14	-3.14	1.38	0.086
	9.45	2.12	1.19	0.310

Neurologie: Un électro-encéphalogramme (EEG) permet de mesurer l'activité du cerveau à plusieurs endroits en même temps en plaçant ici 6 électrodes sur le crâne.

Données: $(x_i, y_i, z_i; V_{ij}), \quad i = 1, \dots, N,$
 $j = 1, \dots, 6.$
 $(x_i, y_i, z_i) \in \text{crâne}.$

• MÉTÉO - PLUIE



	Longitude	Latitude	Rainfall
	43 19' 34"	23 36' 13"	14.6
	21 35' 59"	45 09' 36"	23.6
	43 19' 34"	23 36' 13"	14.6
	.	.	.
	.	.	.
	.	.	.
	.	.	.

Les stations de mesure sont arbitrairement distribuées dans le monde. Les sites peuvent être considérées comme points random sur la sphère de rayon 1.

Données: $(x_i, y_i, z_i; R_i), \quad i = 1, \dots, N.$
 avec la restriction $x_i^2 + y_i^2 + z_i^2 = 1.$

On peut inclure cette restriction dans les données en les décrivant par angles de latitude Φ_i et de longitude $\Psi_i.$

$$(x_i, y_i, z_i) = (\sin\Phi_i \cos\Psi_i, \sin\Phi_i \sin\Psi_i, \cos\Phi_i)$$

• **CLIMAT** (Courtesy R.Crawfis, N.Max, LLNL)



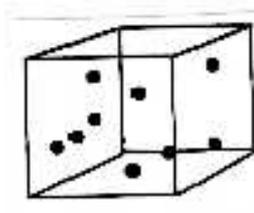
Longitude	Latitude	Wind Velocity
43 19' 34"	23 36' 13"	(1.2 2.5 6.2)
43 19' 34"	23 45' 36"	(2.6 2.9 3.7)
44 20' 57"	23 36' 13"	(2.1 5.2 2.6)
.	.	.
.	.	.
.	.	.
.	.	.

En 19 altitudes au dessus de chaque position la direction et la vitesse du vent sont mesurées.

Données: $(\Phi_{ij}, \Psi_{ij}, \varphi_{ij}^k; (u_{ijk}, v_{ijk}, w_{ijk}))$, $i = 1, \dots, N$,
 $j = 1, \dots, M$,
 $k = 1, \dots, 19$.

Grille curviligne sphérique.

• **SON**



	Location			Decibel
24.45	13.56	3.56	58.0	
10.31	50.45	5.67	49.3	
12.87	35.60	21.04	36.9	
.	.	.	.	
.	.	.	.	
.	.	.	.	
.	.	.	.	

Le niveau du son est mesuré à différents endroits dans une salle. Le but est de placer des appareils générant du son ainsi que des objets absorbant du son afin d'obtenir une distribution optimale.

Données: $(x_i, y_i, z_i; D_i)$, $i = 1, \dots, N$.

• **BOURSE** (Courtesy E-K.Koh)



	RO/4	EP	F2GRW	RTN
	24.1	38.5	62.1	37.1
	18.3	26.5	68.6	13.7

RO/4 - Juillet 1987 performance

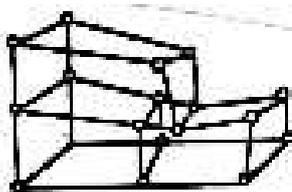
EP - Earnings pro Price Ratio

F2GRW - Groth Potential

RTN - October 1987 Performance

Données: $(x_i, y_i, z_i, w_i), \quad i = 1, \dots, N.$

• **RÉSERVOIR**

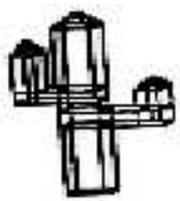


	Location			Density
	405.8	333.5	123.5	1.35
	775.6	456.9	278.2	2.59

Ensemble de cellules se composant en plusieurs faces. Il peut y avoir des failles et des discontinuités.

$F(\mathbf{P}_i)$ peut être différent pour le même point \mathbf{P}_i mais pour différentes cellules.

• ÉLÉMENTS FINIS

	Location			Temperature
	21.9	69.2	23.1	88.7
	95.3	11.9	99.7	78.9

Données: $(x_i, y_i, z_i; T_i), \quad i = 1, \dots, N,$

Faces: $F_k = P_1, \dots, P_N,$

Cellules: $C_j = F_1, \dots, F_M.$

2.2 CLASSIFICATION

Les exemples montrent bien avec quels types de données nous avons à faire en général. Ils sont choisis de manière à être représentatifs pour beaucoup d'autres applications.

Tous les ensembles de données ont 3 variables indépendantes et, soit une valeur dépendante scalaire, soit vectorielle.

$$\left. \begin{array}{l} (x_i, y_i, z_i; F_i) \\ (x_i, y_i, z_i; (u_i, v_i, w_i)) \end{array} \right\} \begin{array}{l} 3 \text{ variables indép.} \\ 1 \text{ variable dép. scalaire ou vectorielle} \end{array}$$

Les 3 variables indép. représentent soit une position dans l'espace, soit le temps et une position en 2D.

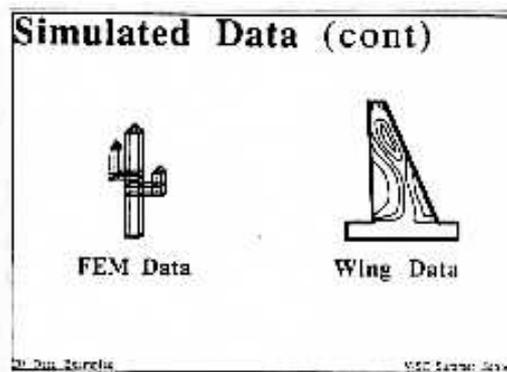
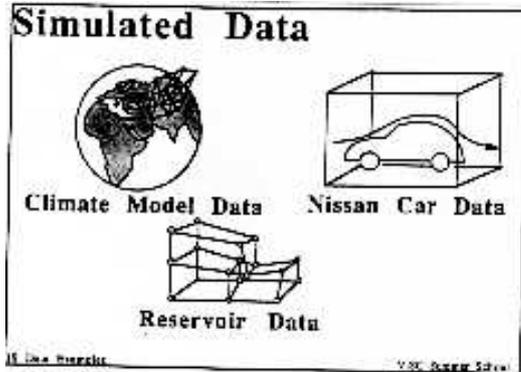
$$\left. \begin{array}{l} (x_i, y_i, z_i) \\ (t_i, x_i, y_i) \end{array} \right\} 3 \text{ variables indép.}$$

On distingue 2 catégories:

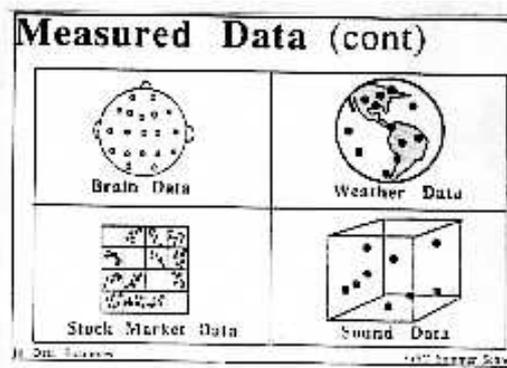
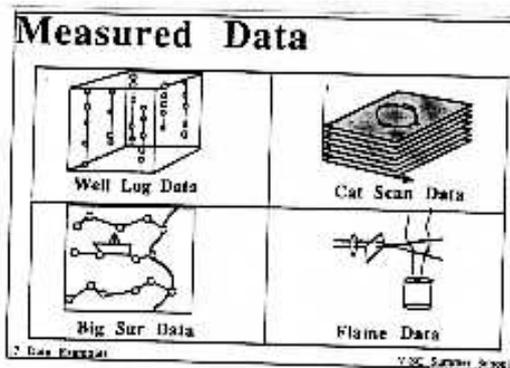
- 3 variable indép., non-contraintes:
La position n'est pas soumise à des contraintes. Elle peut se trouver n'importe où dans une région connexe d'un espace 3D.
⇒ **données volumiques.**
- 3 variables indép., contraintes:
Les données indép. sont restreintes à se trouver dans un sous-ensemble 2D de l'espace 3D.
⇒ **données "surface-on-surface".**

2.2.1 Origine

- Simulation



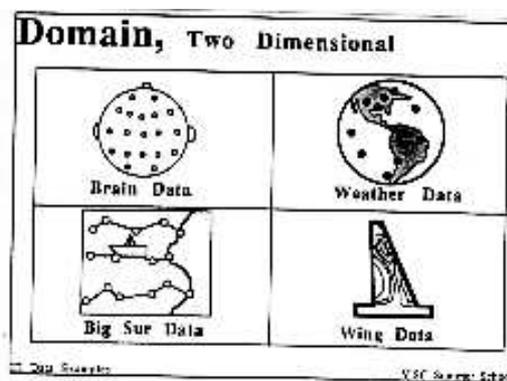
- Mesures



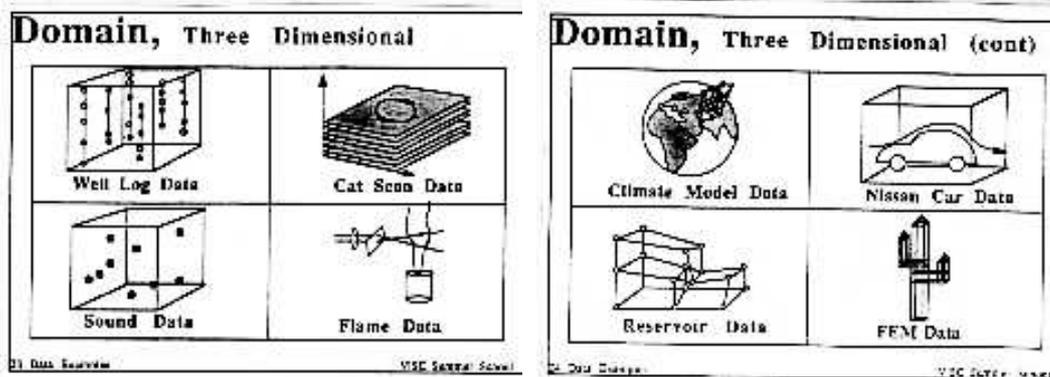
2.2.2 Dimensions

- **Domaine des sites de données**

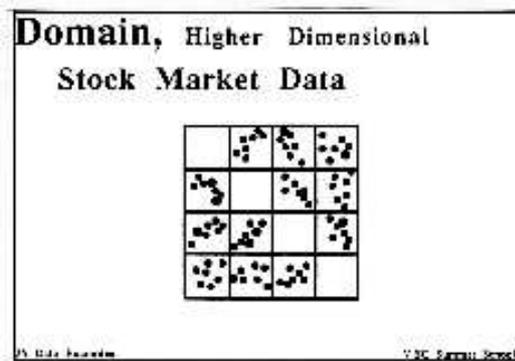
2-DIMENSIONNEL



3-DIMENSIONNEL



n-DIMENSIONNEL



- Image du site de donnée:

SCALAIRE

Densité, température, concentration, pression, potentiel, altitude, décibel, ...

VECTEUR

vélocité $(u(x, y, z), v(x, y, z), w(x, y, z))$,
gradient d'un champ scalaire $F(x, y, z)$

$$\nabla F = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z} \right),$$

force, courant, ...

TENSEUR

Gradient de vitesse

$$(\eta_{ij}) = \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \end{pmatrix}$$

vitesse de torsion $D = c(\eta_{ij} + \eta_{ji})$,

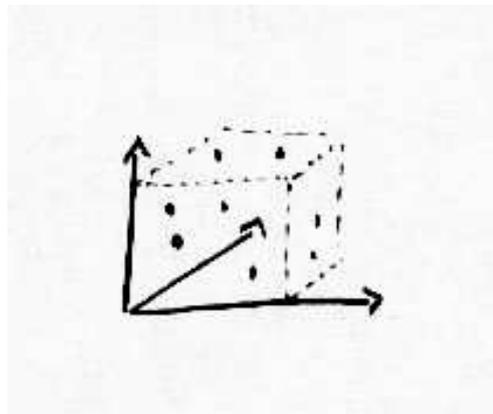
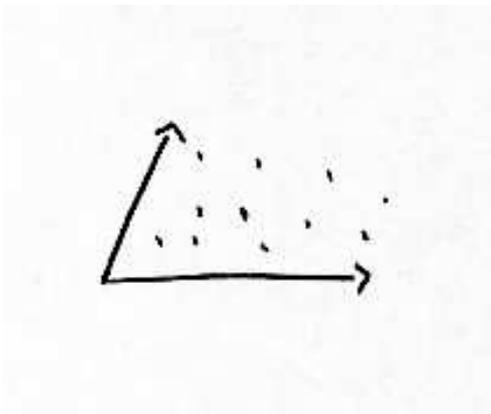
vitesse de rotation $W = k(\eta_{ij} - \eta_{ji})$,

tension, conductivité, moment d'inertie, ...

2.3 STRUCTURE DE DONNÉES

2.3.1 Vocabulaire

DONNÉES irrégulières, scattered data,
non-uniformément distribuées

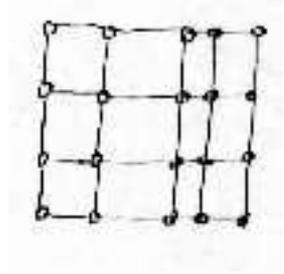
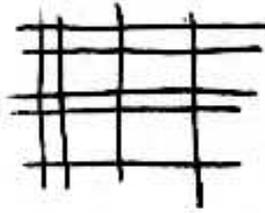
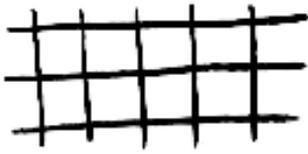


GRILLES, MAILLES (Grids & Meshes)

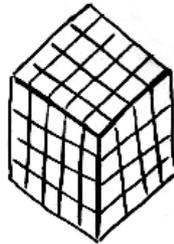
“structurés”:

uniforme
régulière

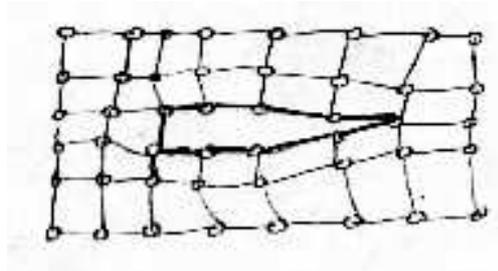
rectangulaire



grille cube

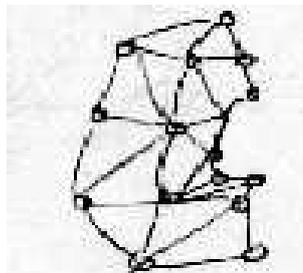


curvilinéaire



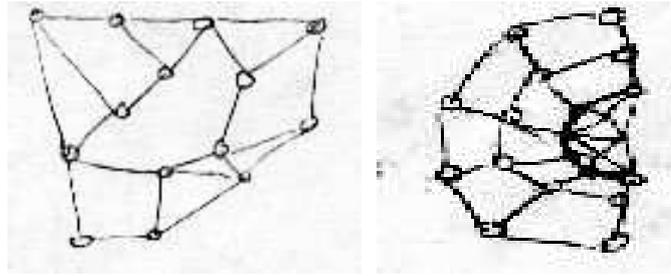
“non-structurés” :

triangulaire



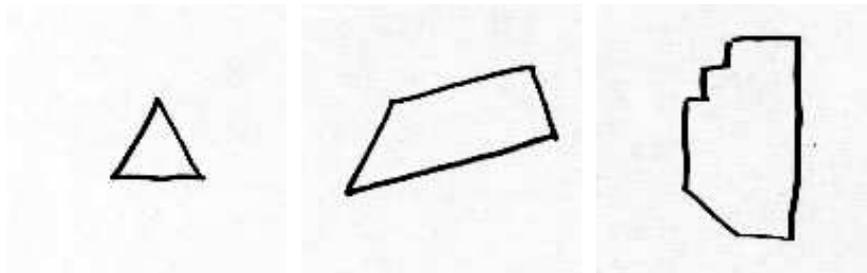
irrégulaire

hybride



CELLULES

2D

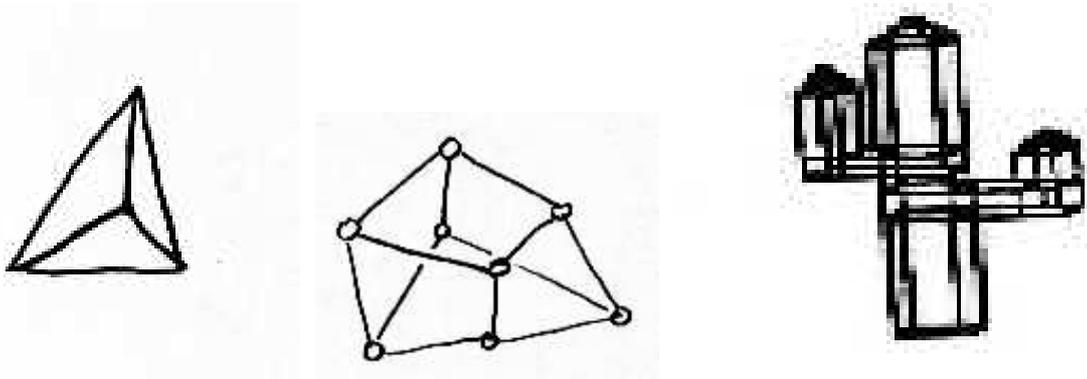


triangle

quadrilatère

polygone

3D



tetraèdre

hexaèdre

polyèdre

2.3.2 Géométrie - topologie

Géométrie: Elle est donnée par des **points** avec une structure de données (tableau, liste connectée, ...). Ils déterminent le positionnement dans l'espace et la forme de l'objet.

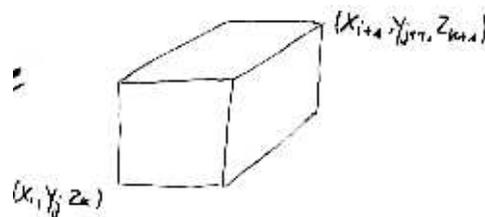
Topologie: Elle consiste en des **connections** de points, arêtes, faces, etc, pour former une décomposition cellulaire de l'objet. Elle

fournit les **relations d'adjacence** ou le voisinage entre sommets, arêtes, faces, ou cellules. La structure de données contient ces informations.

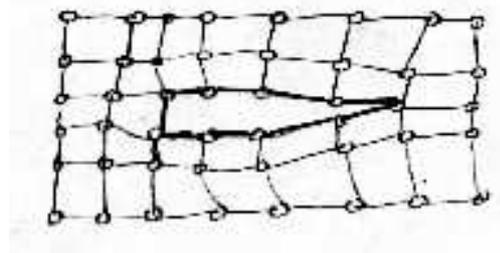
Exemple 1: Grille cartésienne

Géométrie $x_i \quad i = 1, \dots, N_x$
 $y_j \quad j = 1, \dots, N_y$
 $z_k \quad k = 1, \dots, N_z$

Topologie

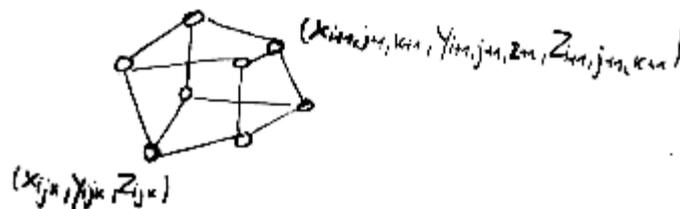


Exemple 2: Grille curviligne



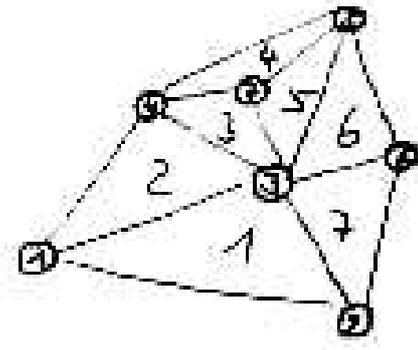
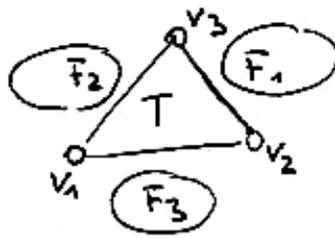
Géométrie $x_{ijk} \quad i = 1, \dots, N_x$
 $y_{ijk} \quad j = 1, \dots, N_y$
 $z_{ijk} \quad k = 1, \dots, N_z$

Topologie



Exemple 3: Grille triangulaire - triangulation

Géométrie $(x_i, y_i, z_i) \quad i = 1, \dots, N$



Topologie
Triangles

Faces voisines

V_1	V_2	V_3	F_1	F_2	F_3
1	2	3	7	2	-1
1	3	4	3	-1	1
4	3	7	5	4	2
4	7	5	5	-1	3
3	5	7	4	3	6
3	6	5	-1	5	7
2	6	3	6	1	-1

se généralise aux grilles tétraédriques.

Polyèdres

En E^3 un polyèdre est défini par un ensemble fini de polygones planes tel que chaque arête d'un polygone est partagée avec exactement un autre polygone et aucun sous-ensemble de polygones a la même propriété. P.ex.

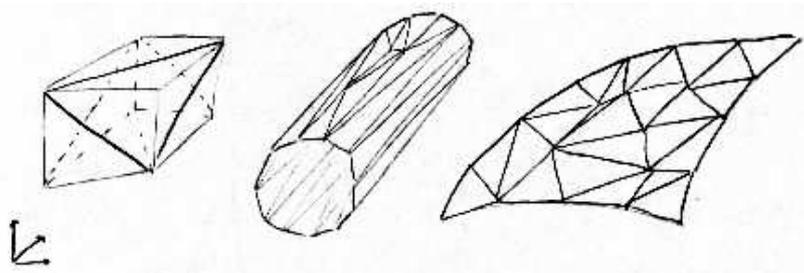
tetraèdre polygone non non

Triangulations surfaciques

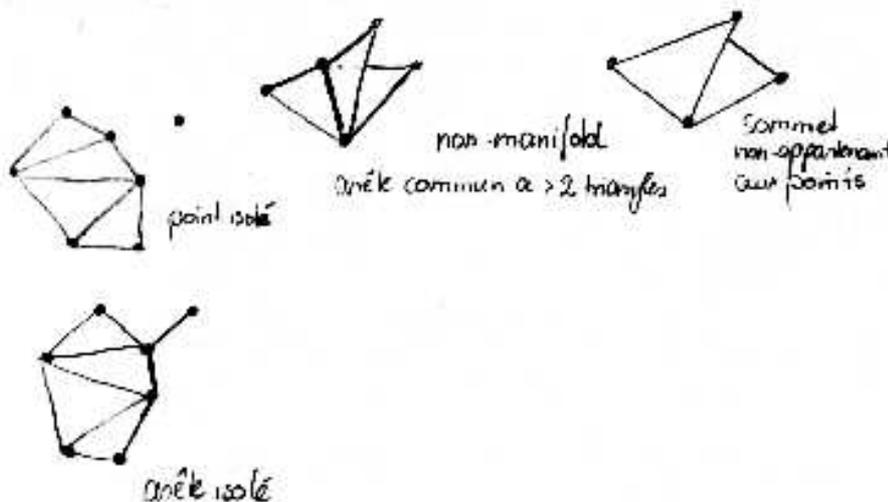
Les triangulations surfaciques sont des triangulations de points dans un espace 3-dimensionnel qui appartiennent à une surface (2-dimensionnelle).

Une triangulation surfacique est une approximation d'une surface par une surface **plane par morceaux**, chaque morceau étant un triangle.

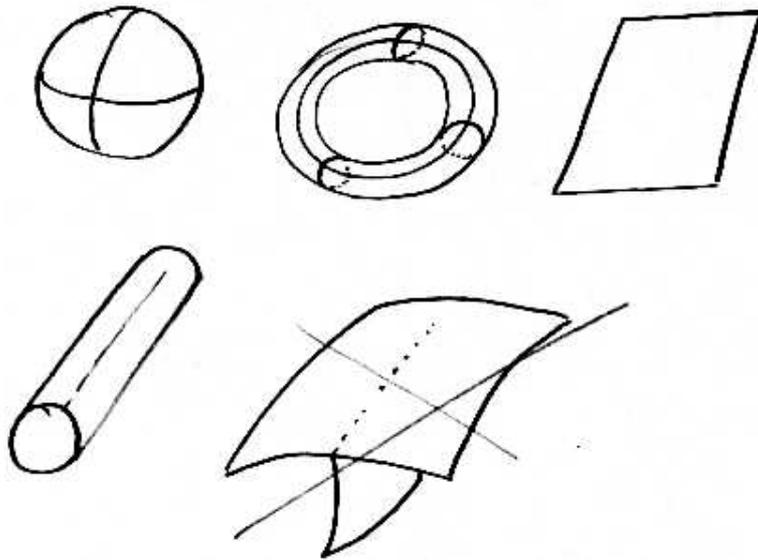
La topologie de la surface sous-jacente peut être quelconque mais doit être "**2-manifold**".



Les situations suivantes sont interdites:



point isolé non-manifold, arête commune à > 2 triangles, sommet non-appartenant, à l'ensemble de points, arête isolée.



Remarque: Ce qui est la triangulation surfacique pour la surface est la **tétrédrisation** pour les volumes.

2.3.3 Structure de données B-rep

Le B-rep permet de décrire autant les informations géométriques que topologiques définissant un objet polyédrique dans l'espace 3D.

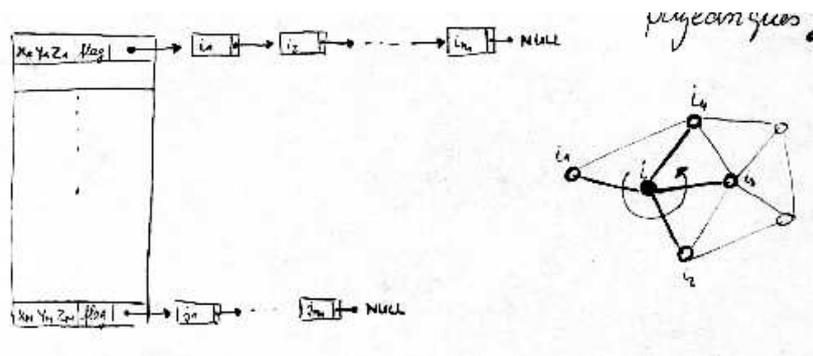
Cette structure de donnée est une représentation d'un objet par ces bord, d'où son nom: B-rep: **Boundary representation**. Elle décrit en priorité la topologie de l'objet sous form de graphe, sur lequel on mémorise les informations géométriques.

On distingue 2 classes de structures ici:

- **Edge List** (EL)
- **Winged Edge List** (Arête Ailée)

Structure de données EL

Elle marche pour tous les objets polyédriques.



```
typedef struct _successeur successeur;
```

```
struct _successeur{
    int numero;
    successeur *suiv;
}
```

```
typedef struct _sommetsommet;
```

```
struct _sommetsommet {
    double x, y, z;
    char flag;
    successeur *succ;
}
```

```
sommetsommet *tableau;
```

```
tableau = (sommetsommet*) malloc (N*sizeof (sommetsommet));
```

Exemple:

- coordonnées du 3^{ème} sommet: tableau[3].x

tableau[3].y

tableau[3].z

- indice du 1^{er} voisin du 5^{ème} sommet:

tableau[5].succ → numero

fichier de données pour des triangulations:

Fichier "classique":

$$\left. \begin{array}{ccc} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & z_n \end{array} \right\} \text{coordonnees des } n \text{ sommets}$$

$$\left. \begin{array}{ccc} \vdots & \vdots & \vdots \\ ind_{i_1} & ind_{i_2} & ind_{i_3} \\ \vdots & \vdots & \vdots \end{array} \right\} \text{indices des sommets des } m \text{ triangles}$$

les indices des sommets d'un triangles doivent être triés dans l'ordre trigonométrique.

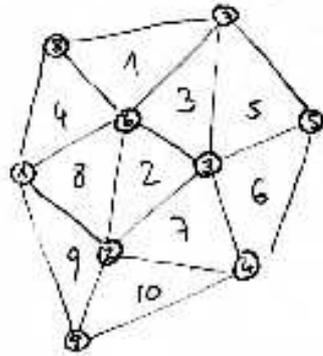
Fichier "EL":

$$N \quad \text{nombre de sommets}$$

$$\left. \begin{array}{cccc} i_1 & i_2 & \cdots & i_{n_1} \\ \vdots & & & \\ k_1 & k_2 & \cdots & k_{n_N} \end{array} \right\} \text{indices des } n_i \text{ voisins du sommet } i$$

$$\left. \begin{array}{ccc} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_N & y_N & z_N \end{array} \right\} \text{coordonnees 3D des } N \text{ sommets}$$

Exemple:



Fichier classique

```
0.1 1.5 3.5
  ⋮
  ⋮
7.8 4.3 1.8
8   6   7
2   3   6
6   3   7
  ⋮
  ⋮
9   4   2
```

liste des coord. des points

liste des triangles

Fichier EL

```
9
9  2  6  8
9  2  6  8
1  9  4  3  6
2  4  5  7  6
9  5  3  2
4  7  3
1  2  3  7  8
1  2  3  7  8
1  6  7
0.1 1.5 3.5
```

⋮

⋮

```
7.8 4.3 1.8
```

liste des sommets voisins à chaque
sommets (topologie)

liste des coord. des points (géométrie)

Opérations :

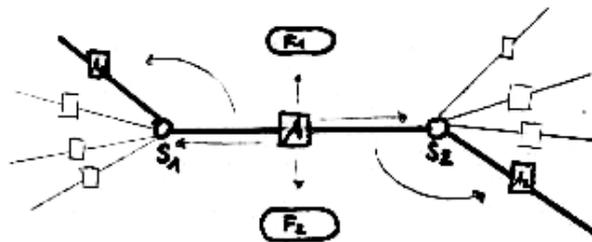
- ajouter sommet
- supprimer sommet
- marquer points du bord
- construire la liste des triangles
- tests si structure est correcte
- ...

Commentaires :

- structure simple
- pas la plus rapide pour les opérations à effectuer
- peu de place mémoire nécessaire.

Structure de données DCEL (Doubly Connected Edge List)

Structure du type "arête ailée", basée sur une représentation par arêtes, faces et sommets, axée sur la description des arêtes.



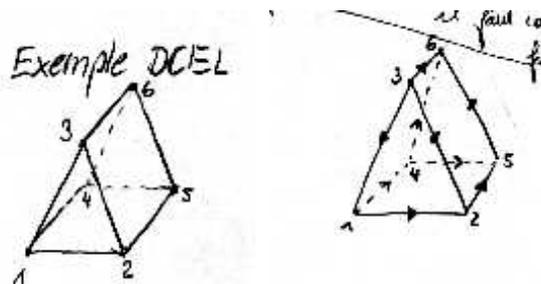
- arête A est représentée par deux sommets extrêmeaux S_1 et S_2 définissant son orientation $S_1 \vec{S}_2$.
- arête A pointe sur deux faces adjacentes: F_1 à sa gauche, F_2 à sa droite (vue de l'extérieure).
- arête A pointe sur deux arêtes obtenues l'une par rotation de A autour de S_1 sur $F_1 \Rightarrow$ arête A_1 ; et rotation (de A) autour de S_2 sur $F_2 \Rightarrow$ arête A_2 .

Pour faciliter la manipulation des objets polyédriques suivant les différents éléments qui le composent, il est nécessaire de construire

une passerelle entre une face et une arête adjacente (n'importe laquelle) et un sommet et une arête (n'importe laquelle).

On rajoute alors au niveau d'une face un lien à une de ces arêtes et au niveau d'un sommet une arête dont il est extrémité. Ces liens ne sont pas uniques.

Exemple: DCEL



Géométrie: liste des coordonnées des sommets, ou points de contrôle d'une surface nurbs pour chaque face,...

Topologie: Il faut commencer par écrire toutes les arêtes et toutes les faces avant de pouvoir remplir le tableau des arêtes.

No.	Face	Arêtes		Sommet		Face arête
	V_1, V_2, \dots	S_1	S_2	A_1	A_2	
1	1 2 3	1	2	6	2	6
2	1 4 5 2	2	5	9	5	9
3	2 5 6 3	1	4	7	1	2
4	1 3 6 4	3	6	8	6	4
5	4 6 5	4	5	8	3	2
6		3	1	3	9	1
7		4	6	6	4	5
8		5	6	2	7	2
9		3	2	4	1	2

Opérations :

- déterminer sommets adjacents à un sommet donné
- déterminer arêtes formant le contour d'une face
- déterminer la normale extérieure à une face
- déterminer l'arête précédente à une arête sur le contour d'une face donnée
- Opérations d'Euler (modifications de la structure)
- construction d'une DCEL à partir d'une EL.
- ...

Commentaires :

- DCEL est un bon compromis entre peu de place mémoire et rapidité.
- DCEL est plus complexe que EL.
- Les opérations DCEL de base sont plus rapides que les EL.

3. MODÉLISATION DE “SCATTERED DATA”

Grands ensembles de données discrètes

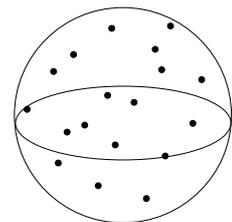
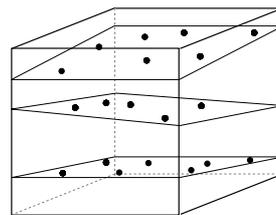
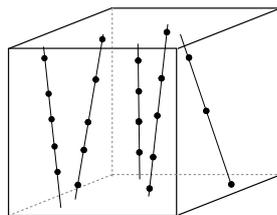
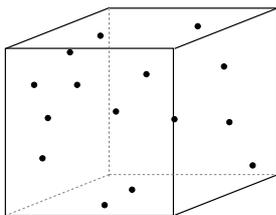
⇒ visualiser, en tirer de l'information

⇒ comprendre la relation entre les données, connaître le graphe de la fonction interpolant ou approximant les données.

Dans beaucoup de domaines techniques ou des sciences naturelles on rencontre le problème d'interpolation ou d'approximation d'un très grand ensemble de données non-structurées (dispersées, irrégulières).

Exemples:

- mesures de température à différents endroits dans un haut fourneau.
- mesures de concentrations de minéraux connues à différentes profondeurs dans des trous de forage.
- mesures de densité à différents endroits dans le corps humain.
- mesures de concentration CO_2 dans l'air à différents endroits sur la terre.
- mesures de pression sur une aile d'avion dans une soufflerie.



donné : $(\mathbf{x}_i; f_i), i = 0, \dots, N$

site de données: $\mathbf{x}_i = (x_i, y_i) \in \mathbb{R}^2$

$\mathbf{x}_i = (x_i, y_i, z_i) \in \mathbb{R}^3$

mesures (données) $f_i \in \mathbb{R}$

cherché : fonction continue $F : \mathbb{R}^d \rightarrow \mathbb{R} (d=2,3)$ t.q.

$F(\mathbf{x}_i) = f_i$ pour l'interpolation

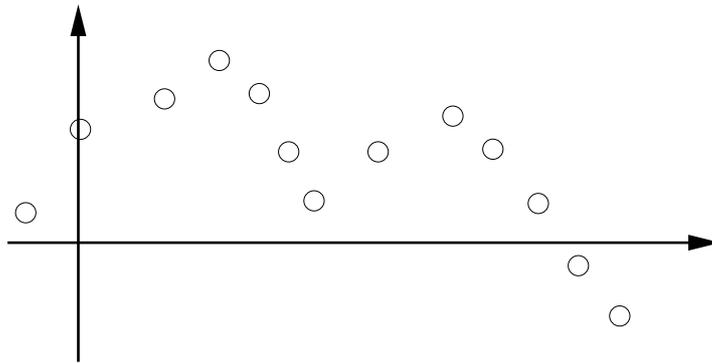
$F(\mathbf{x}_i) \sim f_i$ pour l'approximation, p.ex.

$$E = \sum_{i=1}^N \omega_i (f_i - F(\mathbf{x}))^2 \rightarrow \min.$$

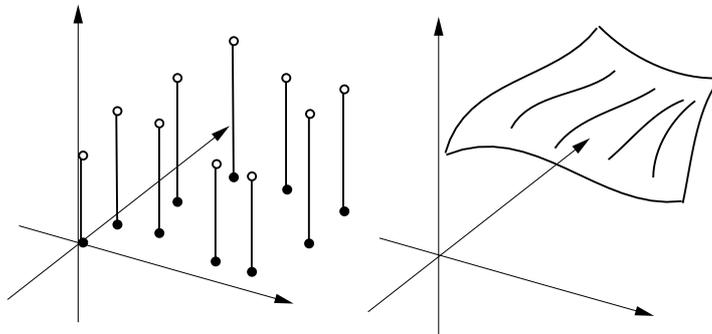
moindres carrés pondérés.

La compréhension de la relation entre les données n'est en général pas possible en les traçant simplement:

- $(x_i; f_i)$

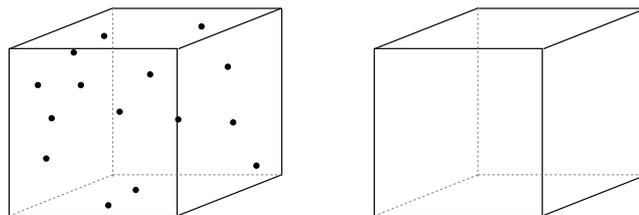


- $(x_i, y_i; f_i)$

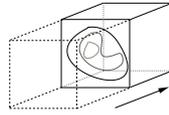


Ici, en traçant le graphe de la fonction interpolante F peut considérablement aider à comprendre les données.

- $(x_i, y_i, z_i; f_i)$



→ Pour beaucoup de méthodes de visualisation de volumes il est indispensable de connaître une fonction (continue) qui représente les données, c'est les cas pour la méthode de visualisation par coupe (slicing).



→ Une fonction continue peut même servir à l'extraction de données sur une grille cubique.

Bien que nous sommes intéressés principalement par la modélisation de scattered data volumiques ou sur une surface, il est plus facile de comprendre les différentes techniques sur des données à 2 variables indépendantes: $(x_i, y_i; f_i) \quad i = 1, \dots, N$.

3.1 MÉTHODES DE BASE POUR L'INTERPOLATION DE SCATTERED DATA DU TYPE $(\mathbf{x}_i \in \mathbb{R}^2, \mathbb{R}^3; f_i)$

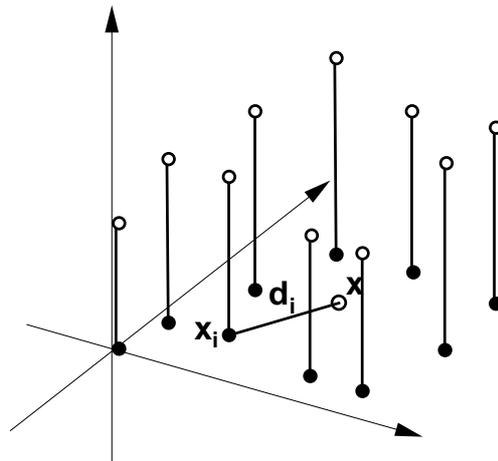
3.1.1 Méthode de Shepard (1965)

Idée principale: L'interpolant est une moyenne pondérée des ordonnées, les poids étant une puissance de l'inverse de la distance entre les sites de données.

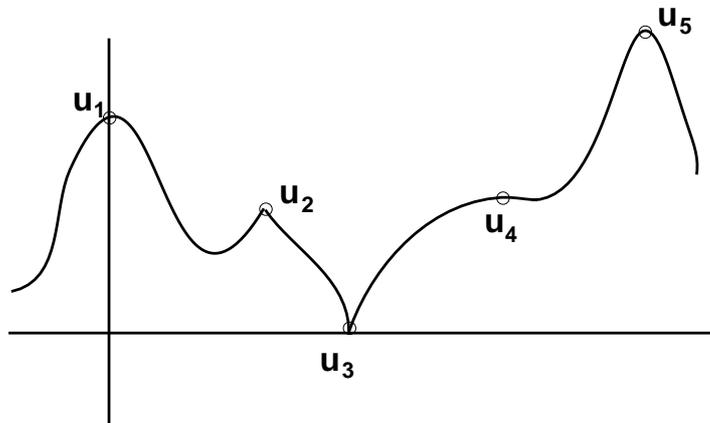
$$F(\mathbf{x}) = \sum_{i=1}^N \omega_i(\mathbf{x}) f_i$$

avec les fonctions de pondération

$$\omega_i(\mathbf{x}) = \frac{\frac{1}{[d_i(\mathbf{x})]^{\mu_i}}}{\sum_{j=1}^N \frac{1}{[d_j(\mathbf{x})]^{\mu_j}}}, \quad d_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_i\|_2, \quad \mu_i \geq 1.$$



$\mu_i = 2$ en pratique.



Influence des μ_i :

$$\mu_1 = 2, \mu_2 = 1, \mu_3 = 0.5, \mu_4 = 15, \mu_5 = 2$$

La fonction de pondération est construite de manière à ce que son influence sur un point diminue lorsque sa distance au point augmente.

Propriétés de $\omega_i(\mathbf{x})$:

- $\omega_i(\mathbf{x}) \in C^0$ continuité
- $\omega_i(\mathbf{x}) \geq 0$ positivité
- $\omega_i(\mathbf{x}_j) = \delta_{ij}$ interpolation $F(\mathbf{x}_i) = f_i$
- $\sum_{i=1}^N \omega_i(\mathbf{x}) = 1$ normalisation.

expression équivalente:

$$\omega_i(\mathbf{x}) = \frac{\prod_{j \neq i} [d_j(\mathbf{x})]^{\mu_i}}{\sum_{k=1}^N \prod_{j \neq k} [d_k(\mathbf{x})]^{\mu_i}}$$

$$\implies \omega_i(\mathbf{x}_j) = \delta_{ij}$$

$$\implies F(\mathbf{x}_j) = f_j, \quad j = 1, \dots, N.$$

Ici on supprime les $\frac{+\infty}{+\infty}$ et $\frac{const}{+\infty}$.

Méthode connue pour avoir beaucoup de désavantages:

Désavantages:

- **Méthode globale:**
 - nécessite de recalculer toutes les ω_i si un point est modifié,
 - $\omega_i(\mathbf{x}) = 1/d_i^2(\mathbf{x})$ influence trop globale
- $\mu_i = 1$: **dérivée discontinue** de l'interpolant en les \mathbf{x}_i
- $\mu_i > 1$: **“flat spots”** (plan tangent parallèle au plan (x, y)) en les (x_i, y_i) , car $\partial F/\partial x(x_i, y_i) = \partial F/\partial y(x_i, y_i) = 0$. Donc la méthode ne reproduit pas la forme locale inhérente des données.
- pas assez d'influence des points éloignés.

Avantages:

- pas de système linéaire à résoudre
- $\text{Min} f_i \leq F(\mathbf{x}) \leq \text{Max} f_i$ (propriété de l'enveloppe convexe due à la normalisation et la positivité de ω_i)
- point de départ pour le développement de beaucoup d'autres méthodes qui elles éliminent les discontinuités:
- **localisation:** $\omega_i(\mathbf{x}) = \left(\frac{(R-d_i(x))_+}{Rd_i(x)}\right)^2$ (voir MQS après)
 $d_i(x) > R \implies \omega_i(x) = 0$.
- Méthode locale avec formule de récurrence, comme Newton (HL 391)

- interpolation des premiers termes du développement de Taylor de F jusqu'à l'ordre k .

Exemple $k = 1$: Interpolation des f_i et de l'espace tangent en les \mathbf{x}_i , i.e. $(f_i = F(\mathbf{x}_i), \frac{\partial F}{\partial x}(\mathbf{x}_i), \frac{\partial F}{\partial y}(\mathbf{x}_i), \frac{\partial F}{\partial z}(\mathbf{x}_i))$

$$F(\mathbf{x}) = \frac{\sum_{i=1}^N \omega_i (f_i + \langle \text{Grad } F(\mathbf{x}_i), (\mathbf{x} - \mathbf{x}_i) \rangle)}{\sum_{i=1}^N \omega_i}$$

$$= \frac{\sum_{i=1}^N \omega_i \left(f_i + \frac{\partial F}{\partial x}(\mathbf{x}_i)(x - x_i) + \frac{\partial F}{\partial y}(\mathbf{x}_i)(y - y_i) + \frac{\partial F}{\partial z}(\mathbf{x}_i)(z - z_i) \right)}{\sum_{i=1}^N \omega_i}$$

\implies plus de "flat spots", \implies améliore l'ordre d'approximation, MAIS, dérivées en général pas disponibles.

Généralisations:

- Interpolation:

$$F(\mathbf{x}) = \frac{\sum \omega_k(\mathbf{x}) L_k f(\mathbf{x})}{\sum \omega_i(\mathbf{x})}$$

$L_k f$ étant une approximation de f t.q. $L_k f(\mathbf{x}_k) = f_k$.

- Approximation au sens des moindres carrés:

$F(\mathbf{x}) = \tilde{F}(a_0, a_1, \dots, a_n; \mathbf{x})$ où a_0, a_1, \dots, a_n sont des paramètres à déterminer t.q.

$$\sum_{k=1}^n \left[f_k - \tilde{F}(a_0, a_1, \dots, a_n; \mathbf{x}) \right]^k \omega_k(\mathbf{x}) \rightarrow \min .$$

La performance des méthodes du dernier type dépend beaucoup du choix de la fonction de pondération ω_i .

Choix de ω_i et de $L_k f$?

3.1.2 Modified Quadratic Shepard (MQS)

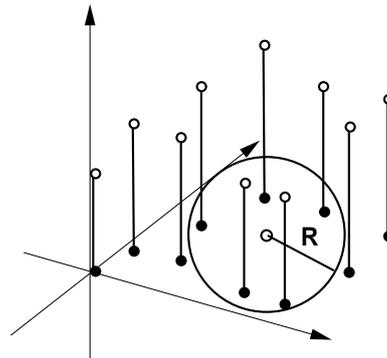
Franke, Nielson 1980

- modification des fonctions de pondération pour localiser l'approximation,
- remplacer $L_k f$ par une approximation locale "appropriée" : Q_i .

$$F(\mathbf{x}) = \frac{\sum_{i=1}^N \omega_i(\mathbf{x}) Q_i(\mathbf{x})}{\sum_{i=1}^N \omega_i(\mathbf{x})}, \quad \omega_i(\mathbf{x}) = \frac{(R_\omega - d_i)_+^2}{(R_\omega d_i)^2} \quad R_\omega = \text{const.}$$

$Q_i(\mathbf{x})$: fonction biquadratique t.q. $Q_i(\mathbf{x}_i) = f_i$ obtenue par approximation par moindres carrés pondérés (poids ϕ_i).

La fonction de pondération ϕ_i pour Q_i est égale à ω_i avec une autre constante R_q .



Algorithme:

1. choisir les paramètres N_q et N_ω afin de définir

$$\omega_i(\mathbf{x}) = \frac{(R_\omega - d_i(\mathbf{x}))_+^2}{(R_\omega d_i)^2}, \quad R_\omega = \frac{D}{2} \sqrt{\frac{N_\omega}{N}}$$

$$\phi_i(\mathbf{x}) = \frac{(R_q - d_i(\mathbf{x}))_+^2}{(R_q d_i)^2}, \quad R_q = \frac{D}{2} \sqrt{\frac{N_q}{N}}$$

avec $D = \max_{i,j} \|\mathbf{x}_i - \mathbf{x}_j\|_2$.

(valeurs par défaut: $N_q = 54$, $N_\omega = 27$).

2. déterminer Q_i $i = 1, \dots, N$, avec

$$\begin{aligned}
 Q_i(\mathbf{x}) = Q_i(x, y, z) = & f_i + a_{i2}(x - x_i) + a_{i3}(y - y_i) + a_{i4}(z - z_i) \\
 & + a_{i5}(x - x_i)^2 + a_{i6}(y - y_i)^2 + a_{i7}(z - z_i)^2 \\
 & + a_{i8}(x - x_i)(y - y_i) + a_{i9}(x - x_i)(z - z_i) \\
 & + a_{i10}(y - y_i)(z - z_i)
 \end{aligned}$$

par la méthode des moindres carrées suivante:

$((x_i, y_i, z_i)$ fixé, k varie, a_{ij} inconnues)

$$G_i = \sum_{\substack{k=1 \\ k \neq i}}^N \phi_i(x_k, y_k, z_k) \left[Q_i(x_k, y_k, z_k) - f_k \right]^2 \longrightarrow \min_{a_{ij}, j=2, \dots, 10}$$

G_i étant une fonctionnelle quadratique convexe.

$$G_i \rightarrow \min \Leftrightarrow \frac{\partial G_i}{\partial a_{ij}} = 0, \quad j = 2, \dots, 10.$$

\Rightarrow système linéaire (9x9) à résoudre.

$\Rightarrow a_{ij}$

$$\Rightarrow Q_i(\mathbf{x}) = f_i + a_{i2}(x - x_i) + \dots + a_{i10}(y - y_i)(z - z_i).$$

$\Rightarrow F$.

Q_i étant bien un interpolant : $Q_k(\mathbf{x}_k) = f_k$.

Remarques:

- L'influence de chaque point ne dépasse pas le rayon de $R_q + R_w$.

Si les données sont bien distribuées régulièrement, les constantes R_q, R_w sont bien appropriées. Si les données ne sont pas d'une densité régulière raisonnable, alors il pourrait être souhaitable de faire dépendre les rayons R_q, R_w de i .

- choix des rayons?
- expérience: bons résultats pour des rayons R_q, R_w ne contenant que 26 et 13 points respectivement.

3.1.3 Spline volumiques (Nielson 1993)

Généralisation directe des fonctions “splines cubiques naturelles” pour les données volumiques si les splines sont représentées par des fonctions de distance. Les fonctions splines cubiques sont très connues et largement acceptées grâce à leur propriété variationnelle:

**Généralisation directe des fonctions
“splines cubiques naturelles”.**

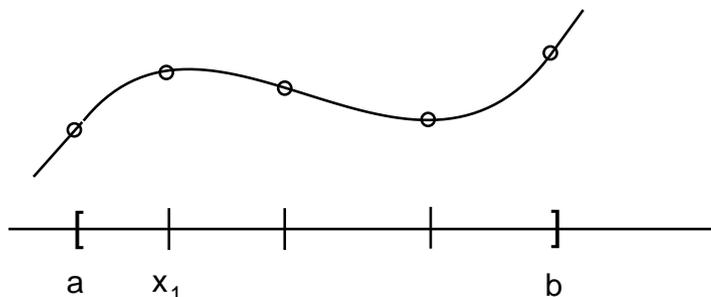
petit rappel: (splines cubiques naturelles)

Propriété:

Entre toutes les fonctions $f : [a, b] \subset \mathbb{R} \rightarrow \mathbb{R}$, 2 fois continuellement dérivables, interpolant x_1, \dots, x_N ce sont les splines cubiques qui minimisent

$$\int_a^b [F''(x)]^2 dx .$$

⇒ effet de lissage, car minimisant une fonctionnelle d'énergie de tension.



F spline cubique naturelle interpolant:

- F, F', F'' continue sur $[a, b]$
- $F(x_i) = f_i, i = 1, \dots, N$
- F cubique par morceau sur $[x_1, x_N]$
- F linéaire sur $[a, x_1]$ et $[x_N, b]$ donc $F''(x_1) = F''(x_N) = 0$ (conditions de bord).

sont les équivalents des conditions de bord $F''(\mathbf{x}_1) = 0$ et $F''(\mathbf{x}_N) = 0$ (spline naturelles) qui impliquent: (*attention: la dérivée seconde n'a pas de sens dans \mathbb{R}^3*)

$$\sum c_i |\mathbf{x}_1 - \mathbf{x}_i| = 0 \quad (1)$$

$$\sum c_i |\mathbf{x}_N - \mathbf{x}_i| = 0. \quad (2)$$

Preuve: $\sum c_i |\mathbf{x}_1 - \mathbf{x}_i| = \sum c_i |\mathbf{x}_1 - \mathbf{x}_N + \mathbf{x}_N - \mathbf{x}_i| \stackrel{(2)}{=} \sum c_i |\mathbf{x}_1 - \mathbf{x}_N| = 0$
 $\Rightarrow \sum c_i = 0.$

$$(1) : \sum c_i |\mathbf{x}_1 - \mathbf{x}_i| = \mathbf{x}_1 \sum c_i - \sum c_i \mathbf{x}_i = 0$$

$$\Rightarrow \sum c_i \mathbf{x}_i = 0. \quad \square$$

Remarques:

- méthode globale
- très mauvais conditionnement de la matrice si $N > 300, 500$ ou si les données sont entachées d'erreurs.
(si 2 points $\mathbf{x}_i, \mathbf{x}_j$ se rapprochent, alors le cond. $\rightarrow \infty$)
- précision polynomiale est 1
- très bons résultats si $N < 300, 500$, surtout à l'intérieur du domaine.

“Splines volumiques locales”

a) Approximation par moindres carrés

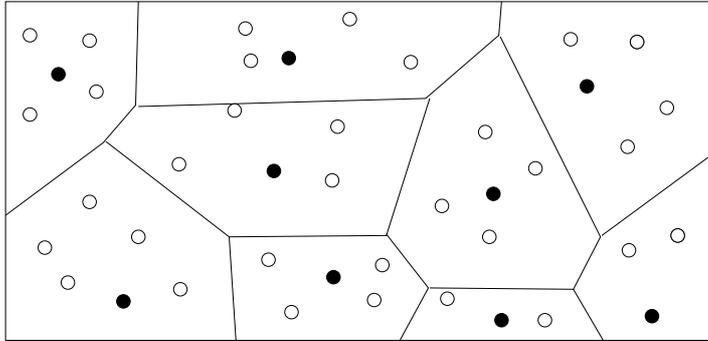
$$F(\mathbf{x}) = \sum_{j=1}^M \underbrace{c_j \|\mathbf{x} - \mathbf{q}_j\|^3 + a + bx + cy + dz}_{Q_j} \quad (*)$$

déterminer c_i, a, b, c, d en minimisant

$$\sum_{i=1}^N \sum_{j=1}^M \underbrace{(c_j \|\mathbf{x}_i - \mathbf{q}_j\|^3 + a + bx_i + cy_i + dz_i - f_i)^2}_{(Q_j(x_i) - f_i)^2}.$$

Choix des noeuds \mathbf{q}_j , $j = 1, \dots, M$ avec $M \ll N$ pour définir les fonctions de base :

- les déterminer uniformément sur une grille cubique
- les distribuer en fonction de la distribution des données:



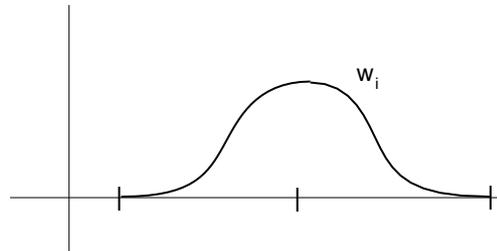
algorithme itératif qui démarre avec une configuration initiale d'une distribution des noeuds \mathbf{q}_j et qui les modifie en minimisant les distances entre les noeuds et les sites des données.

b) Interpolants locaux

cas à une variable:

$$F(x) = \sum \omega_i(x) F_i(x), \quad \sum \omega_i(x) = 1 \quad \forall x$$

$\omega_i(x)$ fonctions de localisation, $\sum \omega_i(x) = 1$, non-nuls seulement sur 2 intervalles



$F_i(x)$ interpolant local, i.e. $F_i(x_j) = f_j$, $\forall x_j \in \text{support de } \omega_i = [x_{i-1}, x_{i+1}]$ (construit par méthode classique des splines p.ex.).
 $\Rightarrow F(x_j) = f_j$ pour tous les $x_j \in \cup_i \text{ support de } \omega_i$. Ce n'est vrai que si les supports des ω_i n'ont pas d'intersection.

$$\begin{aligned}
F(x_j) &= \sum \omega_i F_i(x_j) \\
&= \sum_{x_j \in \text{Supp}(\omega_i)} \omega_i F_i(x_j) \\
&= \sum \omega_i f_j = f_j
\end{aligned}$$

cas volumique:

$$F(\mathbf{x}) = \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \sum_{k=1}^{N_z} \omega_{ijk}(\mathbf{x}) Q_{ijk}(\mathbf{x}),$$

N_x, N_y, N_z taille de la grille.

Support de $Q_{ijk} : [q_{i-1}, q_{i+1}] \times [r_{j-1}, r_{j+1}] \times [s_{k-1}, s_{k+1}]$

- Maximum 8 termes sont non-nuls dans cette somme.
- Fonctions de pondération / localisation:

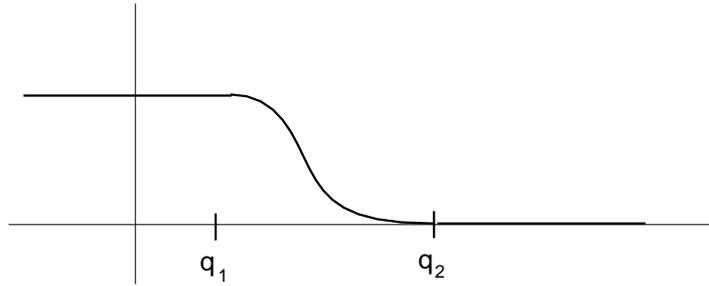
$$\omega_{ijk}(\mathbf{x}) = t_i(x) u_j(y) v_k(z),$$

- t, u, v sont polynomiales par morceaux de degré 3, basées sur les polynômes d'Hermite cubiques: $H(x) = 1 - 3x^2 + 2x^3$.
On a pour ceux-là:

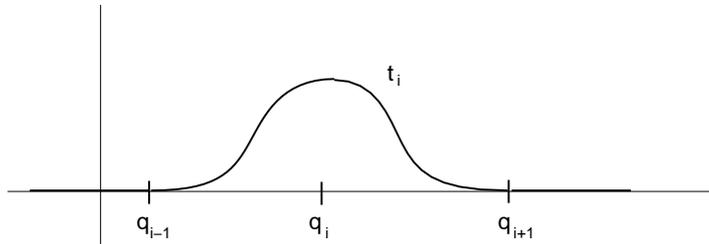
$$\begin{aligned}
H(0) &= 1 & H'(0) &= 0 \\
H(1) &= 0 & H'(1) &= 0.
\end{aligned}$$

Soit par exemple $q_1 < q_2 < \dots < q_{N_x}$ la subdivision du domaine, alors les fonction t_i sont définies comme suivant:

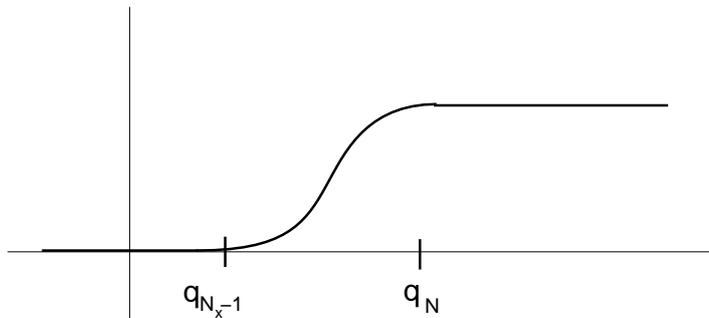
$$t_1(x) = \begin{cases} 1 & x < q_1 \\ H\left(\frac{x-q_1}{q_2-q_1}\right) & q_1 \leq x < q_2 \\ 0 & x \geq q_2 \end{cases}$$



$$t_i(x) = \begin{cases} 0 & x < q_{i-1} \\ 1 - t_{i-1}(x) & q_{i-1} \leq x < q_i \\ H\left(\frac{x - q_i}{q_{i+1} - q_i}\right) & q_i \leq x < q_{i+1} \\ 0 & x \geq q_{i+1} \end{cases}$$



$$t_{N_x}(x) = \begin{cases} 0 & x < q_{N_x-1} \\ 1 - t_{N_x-1} & q_{N_x-1} \leq x < q_{N_x} \\ 1 & x \geq q_{N_x} \end{cases}$$



- Définition analogue de $u_j(y)$ et $v_k(z)$ sur des subdivisions $r_1 < \dots < r_{N_y}$ et $s_1 < \dots < s_{N_z}$.
- $Q_{ijk}(\mathbf{x})$ spline volumique interpolant les points de la région de support R_{ijk} correspondante.
- TB résultats

- utilisable pour des très grands ensembles de données.
- entrer valeurs pour la partition, ou accepter une partition par défaut (*il doit y avoir assez de points par région pour les Q_{ijk} !!*).
- implémentation un peu plus complexe que pour des splines volumiques ou les MQ.

3.1.4 Méthode des Multi-Quadric (MQ) (Hardy 1971)

Méthode du Type *méthode des fonctions de base radiales* (\subset *méthodes globales de fonctions de base*), qui est une des plus performantes:

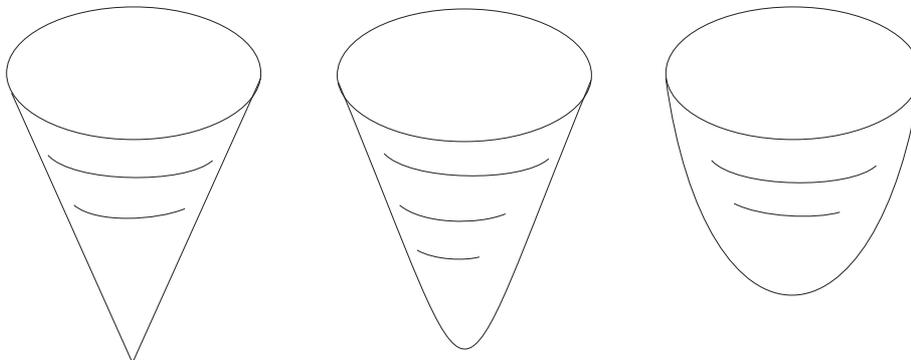
$$F(\mathbf{x}) = \sum_{k=1}^N \alpha_k h_k(\mathbf{x}), R \text{ constante}$$

avec $h_k(\mathbf{x}) = \sqrt{R + \|\mathbf{x} - \mathbf{x}_k\|^2}$ (*fct radiale*).

Les conditions d'interpolation $F(\mathbf{x}_i) = f_i$ donnent le système suivant à résoudre:

$$\begin{pmatrix} \sqrt{R + \|\mathbf{x}_1 - \mathbf{x}_1\|^2} \\ \vdots \\ \sqrt{R + \|\mathbf{x}_1 - \mathbf{x}_N\|^2} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix}.$$

La méthode s'appelle "multiquadrique", car les h_k représentent des hyperboloïde de rotation, $R = 0$: h_k est un cône.



- Existence et unicité de la solution
- méthode donne des résultats "lisses" et tient très bien compte des formes locales: a beaucoup de succès et est facile à implémenter
- mauvais conditionnement de la matrice pour $N > 200, 300, 500$ (cond $> 10^6$)
- L'utilisateur doit choisir le paramètre R en entrée
 - pas de résultats théoriques sur le choix d'un R optimal
 - Propositions: (Hardy 71): $R = 0.815d$, $d =$ dist moyenne des données à leurs voisins
(Franke 82): $R = 1.25D/\sqrt{N}$, $D =$ diamètre du cercle qui comprend tous les données
(Stead 84):

$$R = \sqrt{\frac{1}{10} \max\{\max\{x_i - x_k\}, \max\{y_i - y_k\}, \max\{z_i - z_k\}\}}$$

Précision polynomiale:

$$F(\mathbf{x}) = \sum_{k=1}^N \alpha_k h_k(\mathbf{x}) + \sum_{k=0}^M \beta_k p_k(\mathbf{x}) \quad (*)$$

$\{p_k\}$, $k = 0, \dots, M$ base polynomiale de \mathcal{P}_M .

On peut ainsi forcer les fonctions F à avoir une précision polynomiale d'ordre M .

Le système à résoudre

- interpolation

$$\sum_{k=1}^N \alpha_k h_k(\mathbf{x}_i) + \sum_{k=1}^M \beta_k p_k(\mathbf{x}_i) = f_i \quad i = 1, \dots, N \quad (1)$$

- équilibre physique

$$\sum_{k=1}^N \alpha_k p_j(\mathbf{x}_k) = 0 \quad j = 0, \dots, M \quad (2)$$

$N + M + 1$ équations et $N + M + 1$ inconnues.

Preuve: En fait, supposons que les f_i proviennent d'un polynôme de degré m , i.e. $f_i = p_m(\mathbf{x}_i)$ où $p_m \in \mathcal{P}_m$ donné.

$\alpha_k = 0, k = 1, \dots, N$ est solution de (2).

$$\sum_{k=1}^m \beta_k p_k(\mathbf{x}) = p_m(\mathbf{x})$$

β_k étant les coefficients de $p_m(\mathbf{x}) = \sum_{k=1}^m \beta_k p_k(\mathbf{x}) + 0 \stackrel{(*)}{=} F(\mathbf{x})$

(*) reproduit donc les éléments de \mathcal{P}_m . □

Méthode des fonctions de base radiales:

$$F(\mathbf{x}) = \sum_{k=1}^N \alpha_k h_k(\mathbf{x}) + \sum_{k=1}^M \beta_k p_k(\mathbf{x})$$

$h_k(\mathbf{x}) = h(d_k(\mathbf{x})), d_k(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_k\|_2$ fonctions de base radiales.

$h_k = (R + d_k^2)^{1/2}$ (Hardy)

Multiquadrique

meilleurs résultats aux tests

$h_k = (R + d_k^2)^{-1/2}$ (Hardy)

Multiquadrique réciproque,
presque aussi bon

$h_k = d_k^2 \log d_k$

(Duchon 72, Thin Plate Splines, minimisant

Franke 82) $\int \int_R \left[\left(\frac{\partial^2 f}{\partial x^2} \right)^2 + 2 \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2 + \left(\frac{\partial^2 f}{\partial y^2} \right)^2 \right] dx dy$

dans un espace de Hilbert

- très belle théorie mathématique,
- résultats proche des MQ, résultats très lisse,
- conditionnement un peu moins bon.

Si l'ensemble de données est trop grand pour la méthode globale, on procède comme pour les splines volumiques, soit LS-approximation, soit localiser avec des fonctions de pondération ayant un

support local.

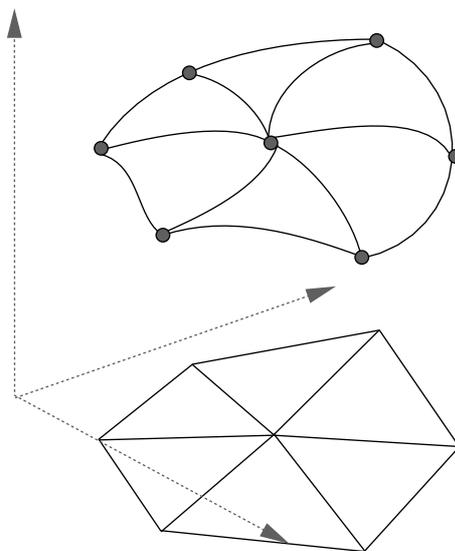
($\omega_k(\mathbf{x})$ Hermite cubique, choix des noeuds optimales + NPPR).
(*number of pts per region*)

3.1.5 Minimum Norm Network (MNN) (Nielson 1983)

Cette méthode n'est pas directement généralisable au 3D. La méthode MNN pour 2 variables indépendantes ($x_i, y_i; F_i$) doit être comprise d'abord.

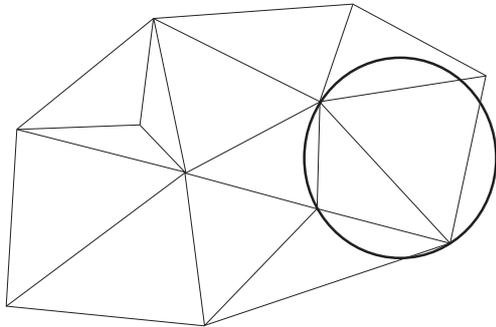
3 étapes:

- 1) **Triangulation** 2D de l'EC des points \mathbf{P}_i aux coordonnées $\mathbf{x}_i = (x_i, y_i)$ $i = 1, \dots, N$.
- 2) définir un **réseau de courbes** C^2 interpolant les données, et minimisant une fonctionnelle.
- 3) **Remplissage** du réseau de courbe par de patches C^1 .



1 TRIANGULATION

$$P = \{P_1, \dots, P_N\}$$



EC (Enveloppe Convexe):
ensemble d'arêtes pour lesquelles
tous les points de l'ensemble se
trouvent sur un des deux demi-
plans séparés par l'arête.

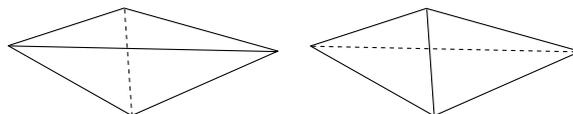
⇒ **Triangulation de Delaunay:**

entre toutes les triangulations possibles de l'EC d'un ensemble de points P , c'est la triangulation de Delaunay qui vérifie

- le critère local d'angle max-min: l'angle minimal de tous les triangulations est maximal, (ou ce qui est équivalent)
- le critère du cercle circonscrit vide: aucun cercle circonscrit d'un triangle contient un autre point de P .

Algorithmes:

- **Edge swap** $O(N^2)$: pour chaque arête, voisine à deux triangles, vérifier le critère local d'angle max-min.



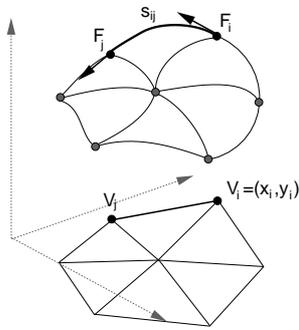
Changer l'arête diagonale (swap) si l'angle minimal de la présente configuration est plus petit que dans l'autre. Répéter cette traversée de toutes les arêtes internes de la triangulation jusqu'à ce qu'il n'y a plus de changement à faire.

- **Divide-and-Conquer** $O(N \log N)$:
- **Algorithme dynamique** $O(N^2)$

⇒ **D'autres triangulations:**

- **angle min-max:** T est mieux que \tilde{T} si $\alpha(T) < \alpha(\tilde{T})$, où $\alpha(T) = \max\{\alpha(T_j)/T_j \in T\}$.
- **Rayon max-min:** le plus petit des rayons des cercles circonscrits des triangles est maximal.
- **Aire max-min:** le minimum des aires des triangles est maximal.

2 RESEAU DE COURBES



$$s_{ij} : [0, 1] \subset \mathbb{R} \rightarrow \mathbb{R}$$

s_{ij} courbe d'Hermite cubique

$$s_{ij}(t) = F_i H_0(t) + F_j H_1(t) + \frac{\partial s}{\partial e_{ij}}(V_i) \bar{H}_0(t) + \frac{\partial s}{\partial e_{ij}}(V_j) \bar{H}_1(t)$$

H_i, \bar{H}_i polynômes d'Hermite de degré 3

$$S : D \subset \mathbb{R}^2 \rightarrow \mathbb{R} \text{ où } s_{ij} = S/e_{ij}$$

$$\text{Dérivée directionnelle: } e_{ij} = \frac{1}{\|e_{ij}\|} \begin{pmatrix} x_j - x_i \\ y_j - y_i \end{pmatrix}$$

$$\frac{\partial s}{\partial e_{ij}}(V_i) = \frac{(x_j - x_i)}{\|e_{ij}\|} S_x(V_i) + \frac{(y_j - y_i)}{\|e_{ij}\|} S_y(V_i)$$

$$\frac{\partial s}{\partial e_{ij}}(V_j) = \frac{(x_j - x_i)}{\|e_{ij}\|} S_x(V_j) + \frac{(y_j - y_i)}{\|e_{ij}\|} S_y(V_j)$$

Pour pouvoir définir les courbes d'Hermite s_{ij} , il nous manque encore les dérivées partielles premières en V_i et V_j : S_x et S_y . Ils

sont déterminer de façon à ce que le réseau de courbes résultant soit la restriction sur les arêtes e_{ij} d'une surface minimisant une fonctionnelle d'énergie (voir splines cubiques).

Rappel: Les splines cubiques naturelles interpolant $(t_i; F_i)$ $i = 1, \dots, N$, sont polynomiale de degré 3 par morceau et solution de $\min_{f \in H^2[t_1, t_n]} \int_{t_1}^{t_n} [f''(t)]^2 dt$ avec $f(t_i) = F_i$ où $H^2[t_1, t_n] = \left\{ f/f \in C^1[t_1, t_n], f' \text{ absolument continue}, f'' \in L^2[t_1, t_n] \right\}$.

La fonctionnelle adaptée à ce réseau de courbes est

$$\sigma(F) = \sum_{ij \in N_e} \int_{e_{ij}} \left[\frac{\partial^2 F}{\partial e_{ij}^2} \right]^2 ds_{ij} \quad (*)$$

- $N_e = \{ij/e_{ij} \text{ est une arête de la triangulation} \}$
- ds_{ij} élément d'arc de la courbe correspondant à l'arête e_{ij}
- F est la restriction aux arêtes e_{ij} d'une fonction de classe C^1 sur le domaine D .

Pour calculer la solution S de $\sigma(F) \rightarrow \text{Min}$ avec $S(V_i) = F_i$, il faut résoudre le système suivant:

$$\sum_{ij \in N_i} \frac{(x_j - x_i)}{\|e_{ij}\|^3} \left[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i) + \frac{1}{2}(x_j - x_i)S_x(V_j) + \frac{1}{2}(y_j - y_i)S_y(V_j) + \frac{3}{2}(F_j - F_i) \right] = 0$$

$i = 1, \dots, N$

$$\sum_{ij \in N_i} \frac{(y_j - y_i)}{\|e_{ij}\|^3} \left[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i) + \frac{1}{2}(x_j - x_i)S_x(V_j) + \frac{1}{2}(y_j - y_i)S_y(V_j) + \frac{3}{2}(F_j - F_i) \right] = 0$$

$i = 1, \dots, N$

$N_i = \{ij / e_{ij} \text{ est arête de la triang, t.q. } V_i \text{ est sommet de } e_{ij}\}$

$\left. \begin{array}{l} 2N \text{ équations} \\ 2N \text{ inconnues} \\ S_x(V_i), S_y(V_i) \end{array} \right\} \text{ système creux, méthode itérative}$

Ainsi nous avons le réseau de courbes cubiques C^1 associé à la triangulation des sommets V_i et minimisant l'énergie de tension d'une latte élastique.

Démonstration (classique):

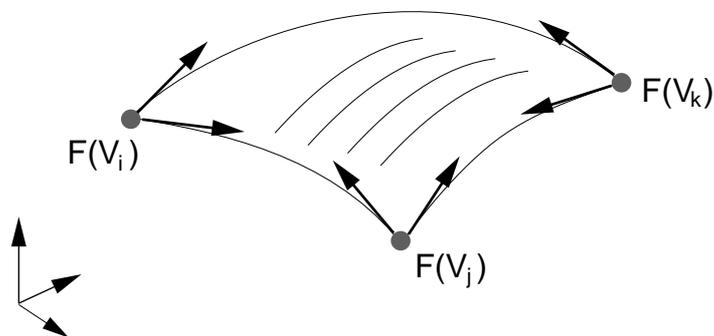
définir un produit scalaire $\langle F, G \rangle$ t.q.

$$\sigma(F) - \sigma(S) = \langle F - S, F - S \rangle$$

et montrer que $\sigma(F) - \sigma(S) \geq 0$.

3 REMPLISSAGE

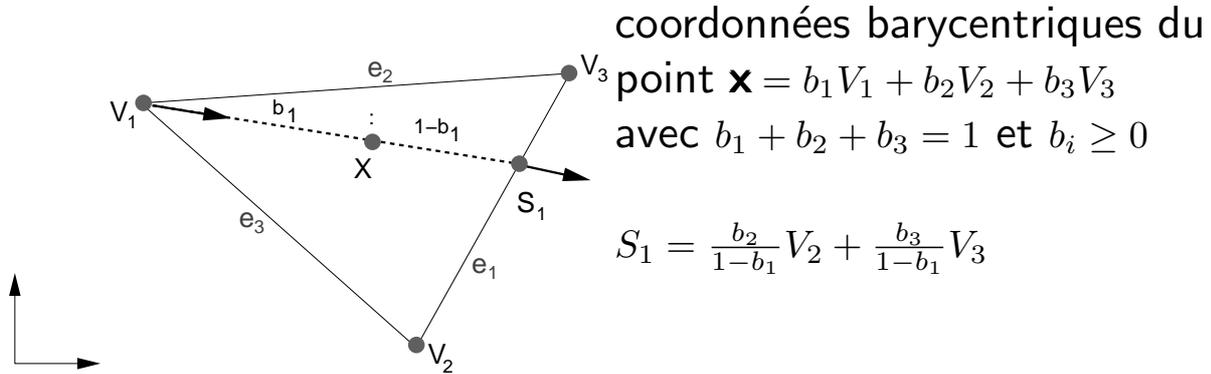
Construction d'un interpolant C^1 triangulaire: $P[F] : \mathbb{R}^2 \rightarrow \mathbb{R}$:



réseau de courbes C^1
cubiques par morceaux

Remplissage avec la méthode **d'interpolation coté - sommet** (side-vertex) (Nielson 1979)

⇒ interpolation d'une infinité de positions et de dérivées le long du bord du triangle.



$P_1[F]$ opérateur interpolant $F(V_1)$, $F(S_1)$, $F'(V_1)$, $F'(S_1)$ avec Hermite cubique

$$P_1[F] = H_0(b_1)F(V_1) + H_1(b_1)F(S_1) + \bar{H}_0(b_1)F'(V_1) + \bar{H}_1(b_1)F'(S_1)$$

$F(V_1)$ données en entrée

$F(S_1)$ MNN

$F'(V_1)$ MNN: $F_x(V_1)$, $F_y(V_1)$

$F'(S_1)$ à calculer par interpolation lin des dérivées partielles en V_2, V_3

$$F'(V_1) = \frac{(x - x_1)}{\|\mathbf{x} - \mathbf{x}_1\|} F_x(V_1) + \frac{(y - y_1)}{\|\mathbf{x} - \mathbf{x}_1\|} F_y(V_1) \quad \text{avec } \|\mathbf{x} - \mathbf{x}_1\| = 1 - b_1$$

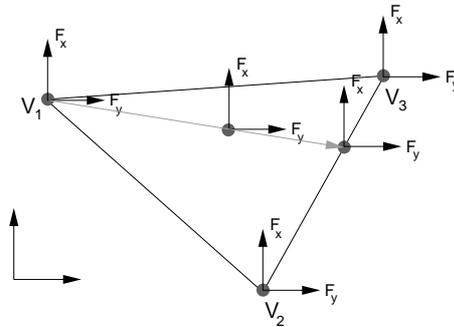
$$= \frac{(x - x_1)F_x(V_1) + (y - y_1)F_y(V_1)}{1 - b_1} \quad \text{dérivée dir. } \frac{1}{1-b_1} \begin{pmatrix} x - x_1 \\ y - y_1 \end{pmatrix}$$

$$F'(S_1) = \frac{(x - x_1)F_x(S_1) + (y - y_1)F_y(S_1)}{1 - b_1} \quad \text{dérivée dir. transversale}$$

$F_x(S_1)$, $F_y(S_1)$ obtenue par interpolation lin.:

$$F_x(S_1) = \left(\frac{b_2}{1-b_1} \right) F_x(V_2) + \left(\frac{b_3}{1-b_1} \right) F_x(V_3)$$

$$F_y(S_1) = \left(\frac{b_2}{1-b_1} \right) F_y(V_2) + \left(\frac{b_3}{1-b_1} \right) F_y(V_3)$$



Les opérateurs $P_2[F]$ et $P_3[F]$ se définissent de manière analogue.

Une combinaison pondérée (fonctions poids ω_i) de ces opérateurs résulte en un interpolant C^1 :

$$P[F](b_1, b_2, b_3) = \frac{b_2^2 b_3^2 P_1[F] + b_1^2 b_3^2 P_2[F] + b_1^2 b_2^2 P_3[F]}{b_1^2 b_2^2 + b_1^2 b_3^2 + b_2^2 b_3^2}$$

$$= \omega_1 P_1[F] + \omega_2 P_2[F] + \omega_3 P_3[F]$$

$$\text{avec } \sum_{i=1}^3 \omega_i = 1$$

$$\omega_i / \text{arête } j = \delta_{ij}$$

$$\partial \omega_i / \text{arête } j = 0$$

facile à vérifier, p.ex. $\omega_1 = \frac{b_2^2 b_3^2}{b_1^2 b_2^2 + b_1^2 b_3^2 + b_2^2 b_3^2}$:

$$\Rightarrow \omega_1 / \text{arête } 1 = 1, \text{ car } b_1 = 0 \text{ sur l'arête } 1.$$

$$\Rightarrow \omega_1 / \text{arête } 2 = 0, \text{ car } b_2 = 0 \text{ sur l'arête } 2.$$

$$\Rightarrow \omega_1 / \text{arête } 3 = 0, \text{ car } b_3 = 0 \text{ sur l'arête } 3.$$

D'où $P[F] / \text{arête } 1 = P_1[F] / \text{arête } 1$ interpole bien l'arête e_1 ,

et $P[F](V_1) = P_1[F](V_1)$ interpole bien le sommet V_1 .

Maintenant les dérivées:

$$\partial P = \partial\omega_1 P_1 + \omega_1 \partial P_1 + \partial\omega_2 P_2 + \omega_2 \partial P_2 + \dots \quad \partial\omega_1 = 0$$

$$\Rightarrow \partial P = \omega_1 \partial P_1 + \omega_2 \partial P_2 + \omega_3 \partial P_3$$

Grâce à la forme Hermite des $P_i[F]$ on sait qu'ils interpolent bien les dérivée sur les arêtes et sommets correspondants.

+++

3.1.6 MNN VOLUMIQUE (Nielson 1993)

1 TETRAÉDRISATION

“L’algorithme le pire au monde” :

- a) générer chaque combinaison de 4 points dans P $O(N^4)$
- b) pour chaque ensemble de 4 points $O(N)$
 - calculer l’équation de la sphère circonscrite unique
 - s’il n’y a pas d’autres points de P à l’intérieur de la sphère, ajouter ces 4 points dans la liste des tétraèdres.

Complexité: $O(N^5)$!!!

Algorithme:

On utilise le fait que chaque face d’un tétraèdre appartient soit à un autre tétraèdre, soit à l’enveloppe convexe de P .

- a) trouver le premier tétraèdre, le mettre dans la liste
- b) pour chaque tétraèdre de la liste
 - {
 - pour chaque face de ce tétraèdre
 - {
 - IF elle n’est pas la face d’un autre tétraèdre de la liste
 - THEN

{

pour chaque point de P (sauf les 3 de la face actuelle)

- calculer la sphère circonscrite
- s'il n'y a pas d'autres points à l'intérieur, ajouter les 4 points comme tétraèdre à la liste.

}

}

}

2 RESEAU DE COURBES

Minimisation
$$\sigma(F) = \sum_{ij \in N_e} \int_{e_{ij}} \left(\frac{\partial^2 F}{\partial e_{ij}^2} \right)^2 ds_{ij}$$

$N_e = \{ij : e_{ij} \text{ arête d'un tétraèdre} \}$.

Solution: réseau de courbes C^1 , cubiques par morceau, définie sur les arêtes de la tétraédrisation, t.q.

$$\sum_{ij \in N_i} \frac{(x_j - x_i)}{\|e_{ij}\|^3} \left[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i) + (z_j - z_i)S_z(V_i) + \right. \\ \left. \frac{1}{2}(x_j - x_i)S_x(V_j) + \frac{1}{2}(y_j - y_i)S_y(V_j) + \frac{1}{2}(z_j - z_i)S_z(V_j) + \right. \\ \left. \frac{3}{2}(F_j - F_i) \right] = 0, \quad i = 1, \dots, N$$

$$\sum_{ij \in N_i} \frac{(y_j - y_i)}{\|e_{ij}\|^3} \left[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i) + (z_j - z_i)S_z(V_i) + \right. \\ \left. \frac{1}{2}(x_j - x_i)S_x(V_j) + \frac{1}{2}(y_j - y_i)S_y(V_j) + \frac{1}{2}(z_j - z_i)S_z(V_j) + \right. \\ \left. \frac{3}{2}(F_j - F_i) \right] = 0, \quad i = 1, \dots, N$$

$$\sum_{ij \in N_i} \frac{(z_j - z_i)}{\|e_{ij}\|^3} \left[(x_j - x_i)S_x(V_i) + (y_j - y_i)S_y(V_i) + (z_j - z_i)S_z(V_i) + \right. \\ \left. \frac{1}{2}(x_j - x_i)S_x(V_j) + \frac{1}{2}(y_j - y_i)S_y(V_j) + \frac{1}{2}(z_j - z_i)S_z(V_j) + \right. \\ \left. \frac{3}{2}(F_j - F_i) \right] = 0, \quad i = 1, \dots, N$$

$\left. \begin{array}{l} 3N \text{ équations} \\ 3N \text{ inconnues} \\ S_x(V_i), S_y(V_i), S_z(V_i) \end{array} \right\} \text{ système linéaire, creux, méthode itérative}$

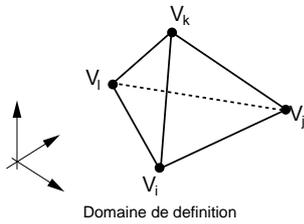
⇒ En chaque point V_i de la tétraédrisation

- valeur de la fonction F
- les dérivées (partielles) premières.

⇒ courbes cubiques d'Hermite, C^1 en les points.

3 REMPLISSAGE

Construction d'un interpolant C^1 tétraédrique:



$$F : \mathbb{R}^3 \rightarrow \mathbb{R} \text{ avec } F(V_i) = F_i, i = 1, \dots, N$$

$$V_i = (x_i, y_i, z_i) \text{ sommet } i$$

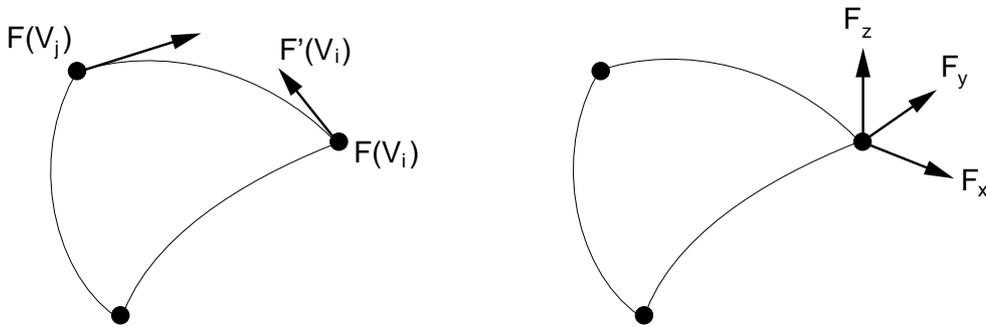
$$e_{ij} = \text{arête } (V_i, V_j)$$

Le MNN fournit en plus les valeurs de dérivée en chaque sommet:

$$\nabla F_i = \left(\frac{\partial F}{\partial x}(V_i), \frac{\partial F}{\partial y}(V_i), \frac{\partial F}{\partial z}(V_i) \right) = (F_x(V_i), F_y(V_i), F_z(V_i))$$

telles que les dérivées directionnelles $\partial F / \partial e_{ij}$ en V_i et V_j , qui servent à déterminer l'interpolant d'Hermite à une variable sur une arête e_{ij} , donnent des courbes minimisant $\sigma(F)$.

On pourrait bien sur se fixer n'importe quelles valeurs pour ∇F_i en les sommets, mais on n'interpolerait pas un MNN.



$$F'(V_i) = \frac{\partial F}{\partial e_{ij}} = \frac{(x_j - x_i)F_x(V_i) + (y_j - y_i)F_y(V_i) + (z_j - z_i)F_z(V_i)}{\|e_{ij}\|}$$

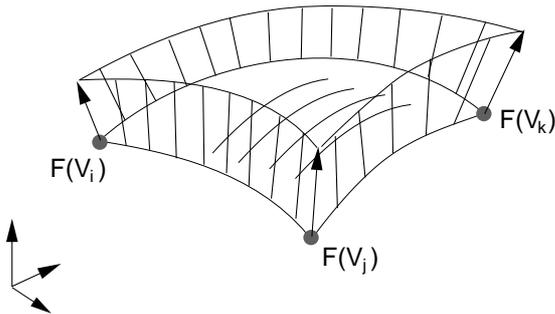
Le but est d'obtenir des informations C^1 partout sur le bord du tétraèdre (i.e. sur les arêtes et sur les faces) pour pouvoir appliquer la méthode d'interpolation "face-sommet" (face-vertex).

(Informations C^1 : positions et dérivées partielles premier ordre)

a) supposons info C^1 est connue sur toutes les arêtes:
Comment l'obtenir sur les faces ?

b) supposons info C^1 est connue dans les sommets:
Comment l'obtenir sur les arêtes ?

a) La méthode *side-vertex- C^1* nous donne les positions et les dérivées directionnelles parallèles à la face du triangle (dans le plan des triangles).



Pour déterminer toutes les dérivées (info C^1 complète) à l'intérieur des faces, nous avons aussi besoin de définir les dérivées orthogonales aux faces.

Il suffit pour cela d'interpoler avec la méthode "coté-sommet- C^0 " les **dérivées normales** connues sur les arêtes (voir b)).

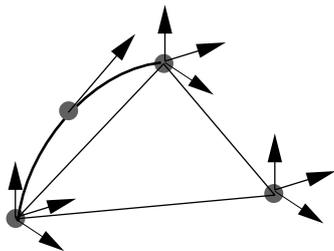
Interpolant *side-vertex- C^0* :

$$A[F] = (1 - b_i)f(e_{jk}) + (1 - b_j)f(e_{ik}) + (1 - b_k)f(e_{ij}) - b_i f(V_i) - b_j f(V_j) - b_k f(V_k)$$

ici: $f(e_{ij}) = \frac{\partial F}{\partial n}(e_{ij})$ dérivée normale à la face sur l'arête e_{ij} .

Cette méthode est l'analogue de la méthode précédente, avec des fonctions poids ω_i linéaires).

b) • position: cubique Hermite avec



$$V_i, V_j, \frac{\partial F}{\partial e_{ij}}(V_i), \frac{\partial F}{\partial e_{ij}}(V_j)$$

cela nous détermine déjà une dérivée directionnelle, celle de direction e_{ij} (*variant comme un polynôme quadratique le long l'arête e_{ij}*).

Pour obtenir un raccord C^1 avec les autres tétraèdres voisins, les autres dérivées directionnelles doivent varier (au moins) linéairement le long cette arête.

D'où la définition du gradient de manière suivante: (*pour $\mathbf{x} \in$ arête e_{ij}*)

$$\nabla F_{ij}(\mathbf{x}) = (1-t)\nabla F_i + t\nabla F_j + \left[\frac{\partial F}{\partial e_{ij}}(\mathbf{x}) - \langle (1-t)\nabla F_i + t\nabla F_j, e_{ij} \rangle \right] e_{ij}$$

où $\nabla F_i = (F_x(\mathbf{x}_i), F_y(\mathbf{x}_i), F_z(\mathbf{x}_i))$, $t = \frac{\|\mathbf{x} - \mathbf{x}_i\|}{\|\mathbf{x}_j - \mathbf{x}_i\|}$, $\|e_{ij}\| = 1$.

Attention: ce gradient définit les 3 dérivées, il faut vérifier que l'on retrouve bien $\frac{\partial F}{\partial e_{ij}}$. Avec le terme correctif [...] ceci est vérifié.

Remarques:

- Ce gradient est bien cohérent avec la valeur $\frac{\partial F}{\partial e_{ij}}$ qui existe déjà:

la composante en direction e_{ij} du gradient:

$$\begin{aligned} \langle \nabla F_{ij}(\mathbf{x}), e_{ij} \rangle &= \langle (1-t)\nabla F_i + t\nabla F_j, e_{ij} \rangle + \langle [\quad] e_{ij}, e_{ij} \rangle \\ &= \langle (1-t)\nabla F_i + t\nabla F_j, e_{ij} \rangle + [\quad] \\ &= \frac{\partial F}{\partial e_{ij}}(\mathbf{x}) \end{aligned}$$

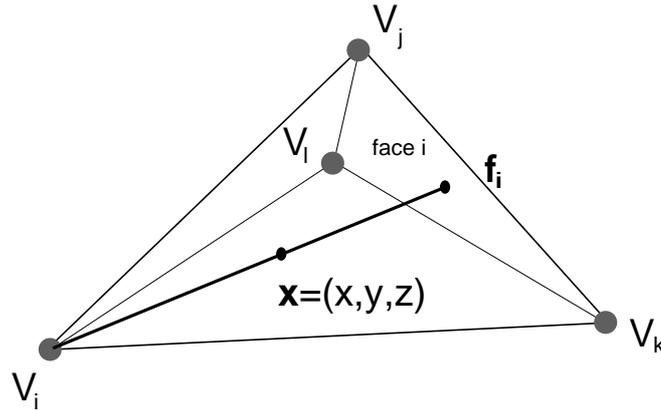
- Ce gradient vérifie bien pour une direction \mathbf{n} , avec $\langle \mathbf{n}, e_{ij} \rangle = 0$, la propriété de variation linéaire dans sa composante normale:

$$\langle \nabla F_{ij}(\mathbf{x}), \mathbf{n} \rangle = (1-t) \langle \nabla F_i, \mathbf{n} \rangle + t \langle \nabla F_j, \mathbf{n} \rangle$$

Nous avons donc une interpolation linéaire de chaque dérivée en direction normale à l'arête e_{ij} .

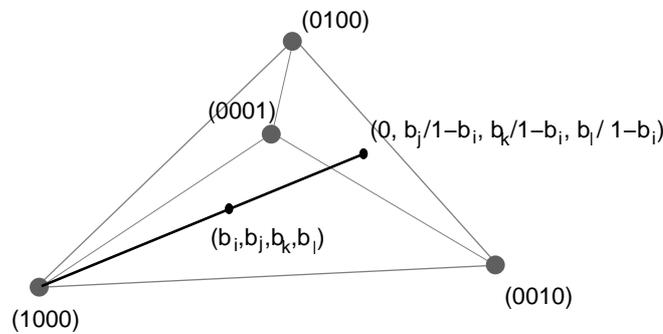
On possède les infos C^1 sur toutes les faces et les arêtes (a) et b)). Donc on peut faire:

Interpolation C^1 tétraédrique avec la méthode **face-vertex**:



Opérateur de base:

$$P_i[F](\mathbf{x}) = F(V_i)H_0(b_i) + F(\mathbf{f}_i)H_1(b_i) + F'(V_i)\bar{H}_0(b_i) + F'(\mathbf{f}_i)\bar{H}_1(b_i)$$



$$\mathbf{x} = b_i\mathbf{x}_i + b_j\mathbf{x}_j + b_k\mathbf{x}_k + b_l\mathbf{x}_l$$

$$b_i + b_j + b_k + b_l = 1 \quad \Leftrightarrow \quad \frac{b_j}{1-b_i} + \frac{b_k}{1-b_i} + \frac{b_l}{1-b_i} = 1$$

$$\frac{\mathbf{x} - b_i\mathbf{x}_i}{1-b_i} = \frac{b_j}{1-b_i}\mathbf{x}_j + \frac{b_k}{1-b_i}\mathbf{x}_k + \frac{b_l}{1-b_i}\mathbf{x}_l = \mathbf{f}_i$$

\mathbf{f}_i face opposée à V_i

D'où

$$\begin{aligned} F'(V_i) &= \frac{x - x_i}{\|\mathbf{x} - \mathbf{x}_i\|} F_x(V_i) + \frac{y - y_i}{\|\mathbf{x} - \mathbf{x}_i\|} F_y(V_i) + \frac{z - z_i}{\|\mathbf{x} - \mathbf{x}_i\|} F_z(V_i) \\ &= \frac{(x - x_i)F_x(V_i) + (y - y_i)F_y(V_i) + (z - z_i)F_z(V_i)}{1 - b_i} \end{aligned}$$

$$F'(\mathbf{f}_i) = \frac{x - x_i}{1 - b_i} F_x(V_i) + \frac{y - y_i}{1 - b_i} F_y(V_i) + \frac{z - z_i}{1 - b_i} F_z(V_i)$$

L'interpolant C^1 sur un tétraèdre est finalement la combinaison des 4 interpolants face-vertex $P_i[F]$:

$$P[F] = \omega_i P_i[F] + \omega_j P_j[F] + \omega_k P_k[F] + \omega_l P_l[F]$$

avec

$$\omega_i = \frac{b_j^2 b_k^2 b_l^2}{b_j^2 b_k^2 b_l^2 + b_i^2 b_k^2 b_l^2 + b_i^2 b_j^2 b_l^2 + b_i^2 b_j^2 b_k^2}$$

$$\text{et } \omega_i + \omega_j + \omega_k + \omega_l = 1$$

$P[F]$ interpole bien les infos C^1 du bord, car les propriétés suivantes des fonctions poids ω_i sont vérifiées:

$$\omega_\alpha / \text{face}_\beta = \delta_{ij}$$

$$\partial \omega_\alpha / \text{face}_\alpha = 0 \quad \alpha, \beta = i, j, k, l$$

$$\text{car : sur la face } j : b_j = 0$$

$\partial \omega_i$: désigne toutes les dérivées premières.

$\partial P[F] / \text{face}_i$ doit être égal à $\partial P_i[F] / \text{face}_i$ ce qui est égale à

$$\partial P_i[F] / \text{face}_i = \partial \omega_i / \text{face}_i P_i[F] / \text{face}_i + \omega_i / \text{face}_i \partial P_i[F] / \text{face}_i$$

$$\partial \omega_i / \text{face}_i = 0 \quad \text{et} \quad \omega_i / \text{face}_i = 1$$

$$= \partial P_i[F] / \text{face}_i \stackrel{\text{Hermite}}{=} F'(\mathbf{f}_i)$$

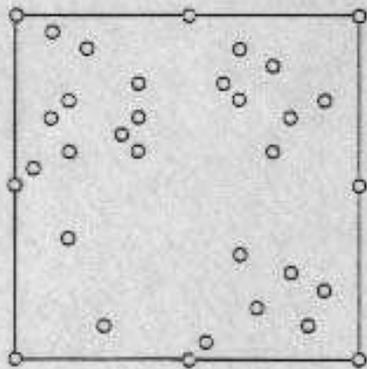


Fig. 9.12a. Data points.

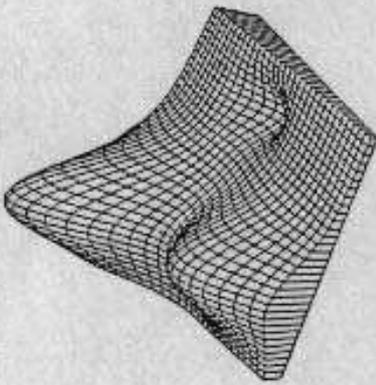


Fig. 9.12b. Test function.

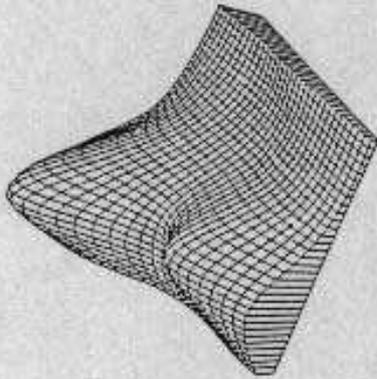


Fig. 9.12g. Duchon's TPS.

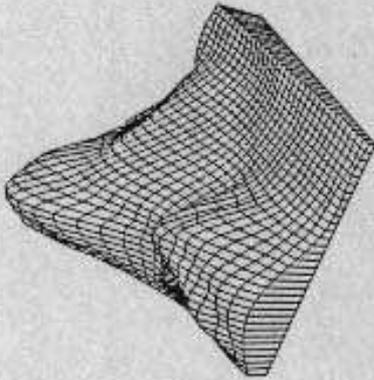


Fig. 9.12h. Akima's method.

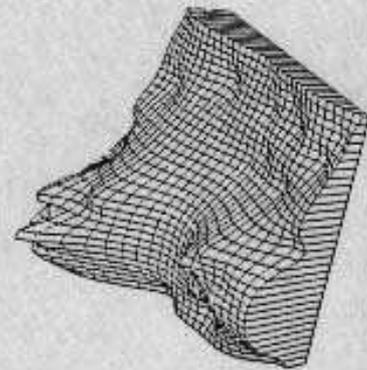


Fig. 9.12c. Shepard's method.

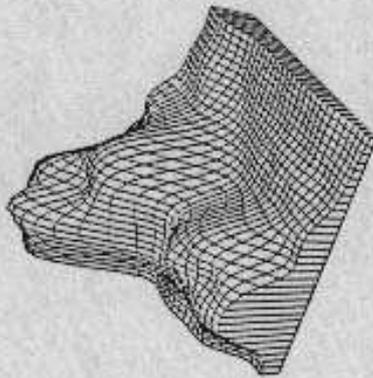


Fig. 9.12d. Modified Shepard's method of (9.7).

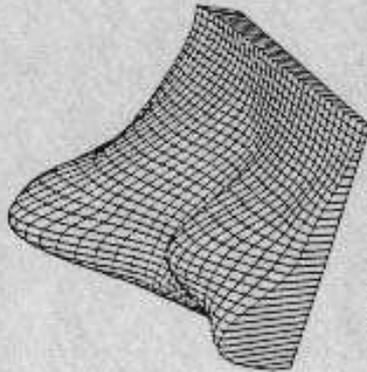


Fig. 9.12i. Lawson's method.

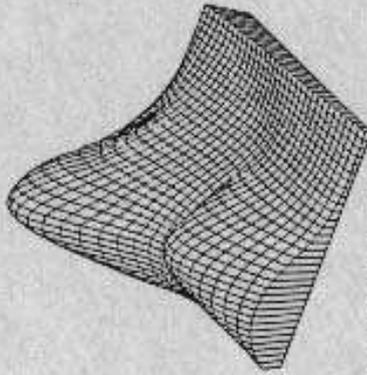


Fig. 9.12j. Nielson's MINN method.

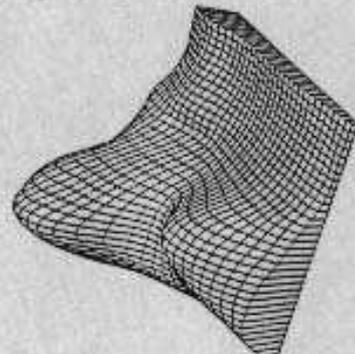


Fig. 9.12e. Modified quadratic Shepard's method [Fra 80].

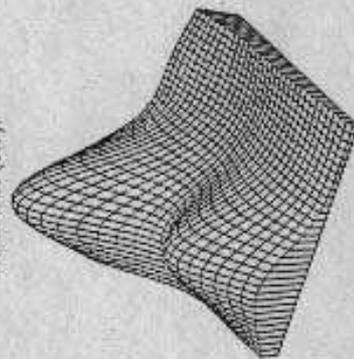
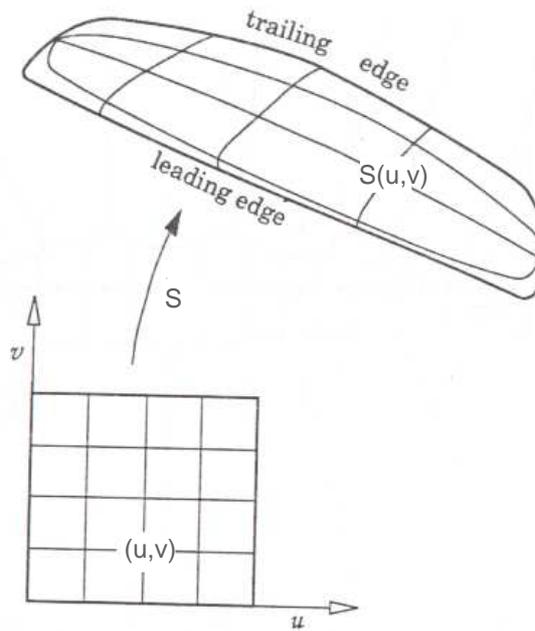


Fig. 9.12f. Hardy's MQ method.

3.2 MODÉLISATION DE SCATTERED DATA DU TYPE “SURFACE-SUR-UNE-SURFACE”

Problème : Etant donné $(x_i, y_i, z_i; F_i)$, $i = 1, \dots, N$ avec la restriction que $\mathbf{x}_i = (x_i, y_i, z_i)$ appartiennent à une surface.

Trouver une fonction $F : S \subset \mathbb{R}^3 \rightarrow \mathbb{R}$, t.q. $F(\mathbf{x}_i) = F_i$.



3.2.1 Surface générale

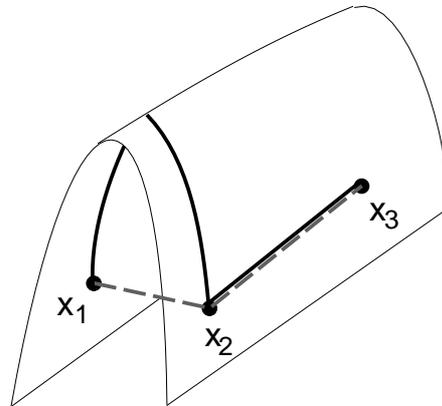
Application: déterminer la distribution de chaleur / pression sur une aile d'avion.

- **Multiquadriques de Hardy 3D:** (Barnhill, Piper 87)

$$F(\mathbf{x}) = \sum_{i=1}^N c_i (d_i^2(\mathbf{x}) + R^2)^{1/2}$$

où $d_i(\mathbf{x}) = \|\mathbf{x} - \mathbf{x}_i\|$, et $\mathbf{x}, \mathbf{x}_i \in S$.

Attention, il faut veiller à ce que l'on mesure les distances *correctes* !



3.2.2 Sphère (MNN sur la sphère)

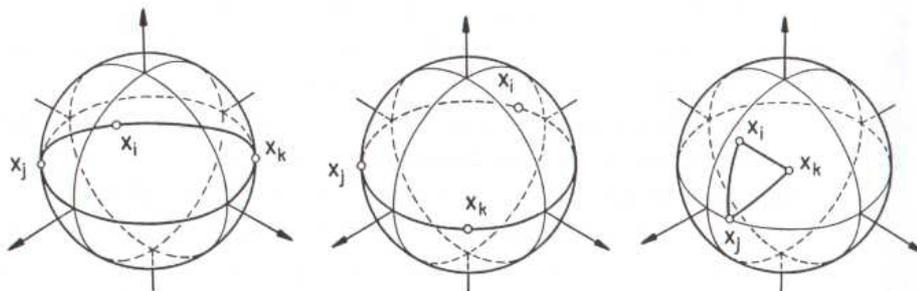
$$S = \{(x, y, z) / x^2 + y^2 + z^2 = 1\} \quad \text{sphère unité}$$

1 Triangulation d'une sphère:

On parle de triangles sphériques propres T_{ijk} , formant une triangulation sphérique, si

- aucun couple de $\{\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k\}$ est anti-podal,
- $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$ ne se trouvent pas sur un grand cercle,
- pas d'intersection de triangles sphériques.

Les arêtes de la triangulation sont des courbes géodésiques



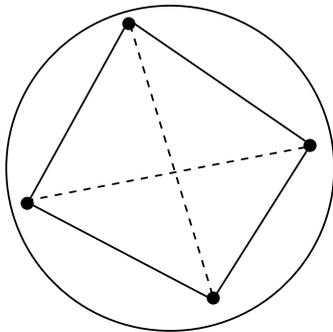
non-admissible,
antipodal

non-admissible,
même hémisphère

admissible

N sommets $\Rightarrow 2(N - 2)$ triangles
 $3(N - 2)$ arêtes.

Algorithme:

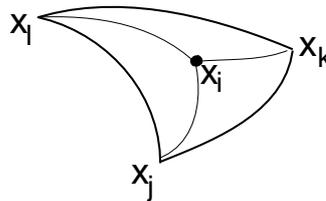


1) choisir 4 points des $\mathbf{x}_i, i = 1, \dots, N$ qui ne se trouvent pas tous sur la même hémisphère (Halbkugel).
 Ces 4 points forment la triangulation initiale de 4 triangles sphériques propres (tetraèdre sphérique).

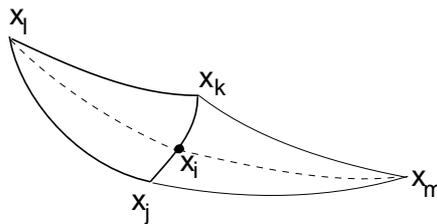
2) Insérer les $N - 4$ points restant itérativement dans la triangulation actuelle:

- si \mathbf{x}_i est à insérer, considérer 2 possibilités:

a) $\mathbf{x}_i \in T_{jkl} \setminus \partial T_{jkl}$



b) $\mathbf{x}_i \in \partial T_{jkl} \cup T_{jmk}$



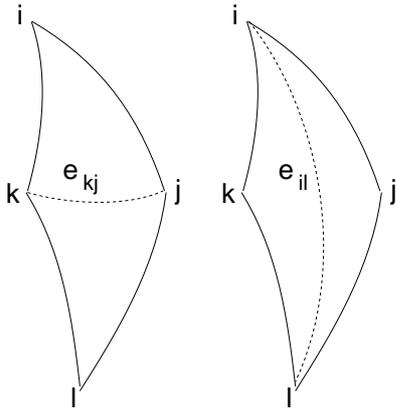
3) Optimiser la triangulation pour éviter des triangles allongées (avec l'algorithme de swapping):

Pour chaque quadrilatère sphérique T_{ijk}, T_{lkj}

IF $\min(\Theta_{kij}, \Theta_{ijk}, \Theta_{jki}, \Theta_{kjl}, \Theta_{jlk}, \Theta_{lkj})$
 $< \min(\Theta_{ijl}, \Theta_{jli}, \Theta_{lij}, \Theta_{ilk}, \Theta_{lki}, \Theta_{kil})$

THEN choisir l'arête diagonale e_{il}

ELSE choisir l'arête diagonale e_{jk}



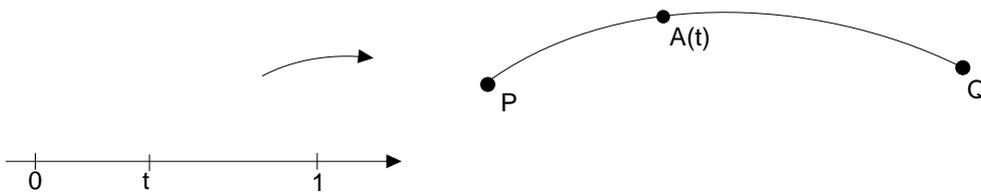
où Θ_{abc} est l'angle entre les deux plans contenant les arêtes e_{ab} et e_{bc} an passant par l'origine.

Distance géodésique entre $\mathbf{x}_i, \mathbf{x}_j$: $d(\mathbf{x}_i, \mathbf{x}_j) = \arccos(\langle \mathbf{x}_i, \mathbf{x}_j \rangle)$.

Paramétrisation de la sphère:

$$\begin{aligned} x &= \cos\Theta \cos\Phi & -\frac{\pi}{2} &\leq \Theta \leq \frac{\pi}{2} \\ y &= \cos\Theta \sin\Phi & 0 &\leq \Phi \leq 2\pi \\ z &= \sin\Theta \end{aligned}$$

Tracer les arcs géodésiques: $A : [0, 1] \rightarrow S \subset \mathbb{R}^3$



$$A(t) = \frac{\mathbf{P} \sin(\phi(1-t)) + \mathbf{Q} \sin(\phi t)}{\sin\phi}, \quad \phi = \arccos(\langle \mathbf{P}, \mathbf{Q} \rangle)$$

2 Réseau de courbes:

Sur chaque arête géodésique de la triangulation sphérique un polynôme cubique est construit, le paramètre est la distance géodésique.

Un MNN-système analogue au MNN classique donne les dérivées directionnelles de “base” en les sommets de la triangulation:

$$F_{\Theta} = \frac{\partial F}{\partial \Theta}, \quad F_{\Phi} = \frac{\partial F}{\partial \Phi}$$

avec Θ, Φ latitude, longitude de la sphère.

Une dérivée directionnelle en direction $(\mathbf{x}_i - \mathbf{x}_j)$ est calculée avec

$$\frac{\partial F}{\partial[\mathbf{x}_i - \mathbf{x}_j]} = \frac{\partial \Theta}{\partial[\mathbf{x}_i - \mathbf{x}_j]} F_{\Theta} + \frac{\partial \Phi}{\partial[\mathbf{x}_i - \mathbf{x}_j]} F_{\Phi}$$

les coefficients sont un peu plus compliqué à calculer que $\frac{(x_i - x_j)}{\|\mathbf{x}_i - \mathbf{x}_j\|}$ dans le MNN classique.

Le système à résoudre: (Nielson, Ram. Focus)

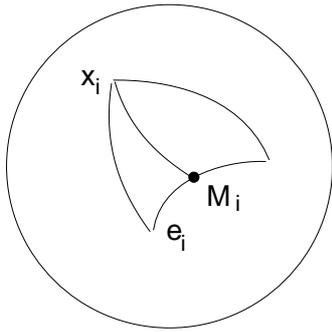
$$\sum_{i,j \in N_i} \frac{1}{\|x_j - x_i\|} \frac{\partial \Theta}{\partial[x_j - x_i]}(x_i) \left[\frac{\partial \Theta}{\partial[x_j - x_i]}(x_i) S_{\Theta}(x_i) + \frac{\partial \Phi}{\partial[x_j - x_i]}(x_i) S_{\Phi}(x_i) + \frac{1}{2} \frac{\partial \Theta}{\partial[x_j - x_i]}(x_i) S_{\Theta}(x_j) + \frac{1}{2} \frac{\partial \Phi}{\partial[x_j - x_i]}(x_i) S_{\Phi}(x_j) + \frac{3}{2} \frac{F_i - F_j}{\|x_j - x_i\|} \right] = 0$$

caractérisant un réseau de courbes minimisant

$$\sigma(F) = \int_{e_{ij}} \left(\frac{\partial^2 F}{\partial[\mathbf{x}_i - \mathbf{x}_j]^2} \right) ds_{ij}$$

3 Remplissages:

$$\begin{aligned} P_i[F] &= F(\mathbf{x}_i) H_0(b_i) + F(\mathbf{M}_i) H_1(b_i) \\ &+ \|\mathbf{x}_i - \mathbf{M}_i\| \frac{\partial F}{\partial[\mathbf{x}_i - \mathbf{M}_i]}(\mathbf{x}_i) \bar{H}_0(b_i) \\ &+ \|\mathbf{x}_i - \mathbf{M}_i\| \frac{\partial F}{\partial[\mathbf{x}_i - \mathbf{M}_i]}(\mathbf{M}_i) \bar{H}_0(b_i) \quad i = 1, 2, 3 \end{aligned}$$



\mathbf{M}_i intersection du plan $(0, \mathbf{x}_i, \mathbf{x})$

avec l'arête e_i

$$t = \frac{\|\mathbf{x} - \mathbf{M}_i\|}{\|\mathbf{x}_i - \mathbf{M}_i\|} \text{ où } \|\cdot\| \text{ distance géodésique.}$$

$P[F]$ combinaison convexe de $P_1[F]$, $P_2[F]$, et $P_3[F]$ comme dans la méthode "coté-sommet" (side-vertex) classique. Les coordonnées barycentriques b_1, b_2, b_3 de \mathbf{x} étant les coordonnées barycentriques du triangle plan $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ de \mathbf{x}' (\mathbf{x}' projection centrale de \mathbf{x} sur le triangle plan).

4. VISUALISATION DE DONNÉES

“SURFACE-SUR-UNE-SURFACE”

*Une des méthodes les plus puissantes pour la visualisation de données sur des surfaces est la **méthode de contour**. Nous présentons ici l’algorithme pour les sphères:*

4.1 CONTOUR D’UNE FONCTION SUR UNE SPHÈRE

La méthode consiste à tracer des courbes sur la sphère sur lesquelles la fonction admet une valeur constante, i.e.

$$F(\mathbf{x}) = \alpha, \quad \alpha = \text{constant}.$$

La fonction peut être connue continuellement, ou simplement en un nombre fini de points.

Il est très difficile de calculer ces courbes directement, d’où un algorithme qui donne des courbes de contour approximantes.

4.1.1 Fonction connue discrètement

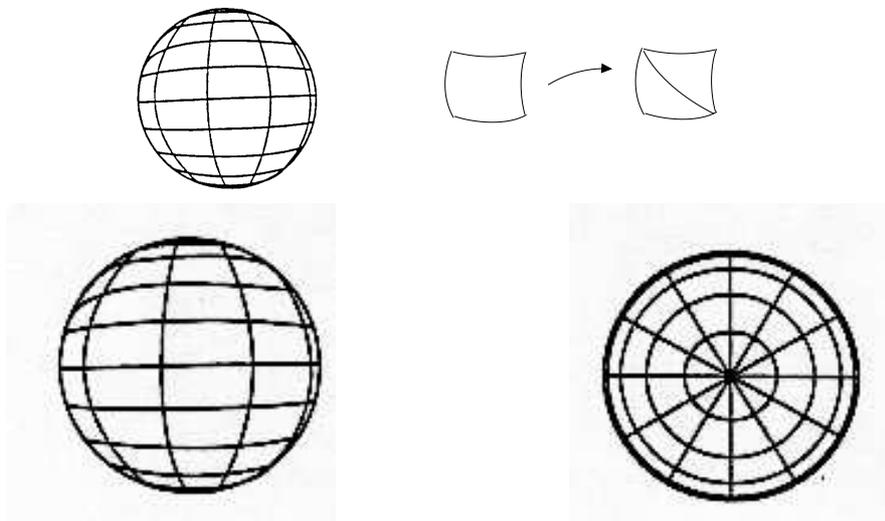
F n’est connue qu’en un nombre fini de points. Si la densité des points n’est pas suffisante pour en pouvoir extraire des informations directement (triangulation sphérique, puis contouring), une méthode de scattered data sur une sphère (ex. MNN sphérique, basée sur une triangulation de Delaunay sphérique, voir 3.2.2) est à utiliser pour interpoler les données et pour fournir une description continue. Ensuite il est à procéder comme sous 4.1.2.

4.1.2 Fonction connue continuellement

Si la fonction est connue de manière continue sur tout le domaine, il faut d'abord l'évaluer sur une grille triangulaire du domaine, ici la sphère. Pour un meilleur rendu il est important que la grille soit la plus régulière possible (triangles tous de la même taille, équiangulaires, équilatéraux).

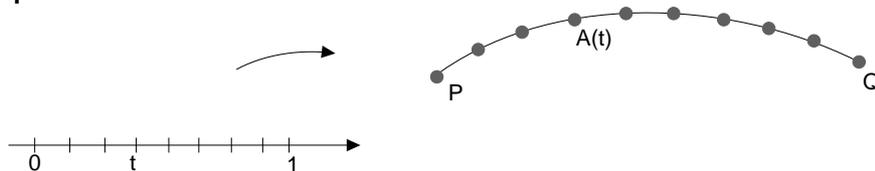
Décomposition régulière du domaine sphérique en triangles

- **algo 1:** grille latitude-longitude, puis décomposition des quadrilatères en triangles: donne des triangles plus petits proche des pôles !!!



- **algo 2:**

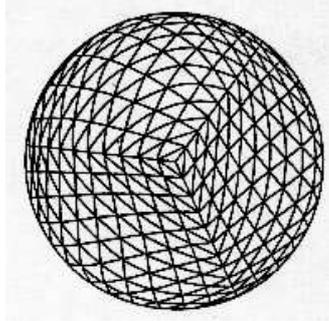
- 4 arêtes du tétraèdre initial: (4 points, 4 triangles)
 $(0, 0, 1)$, $(\frac{\sqrt{2}}{3}, -\frac{\sqrt{6}}{3}, -\frac{1}{3})$, $(\frac{\sqrt{2}}{3}, \frac{\sqrt{6}}{3}, -\frac{1}{3})$, $(-\frac{2\sqrt{2}}{3}, 0, -\frac{1}{3})$
- subdiviser uniformément les arcs géodésiques joignant ces 4 points:



$$A(t) = \frac{P \sin(\Theta(1-t)) + Q \sin(\Theta t)}{\sin(\Theta)}, \quad 0 \leq t \leq 1$$

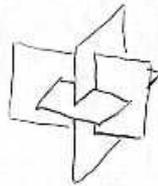
$\Theta = \arccos(\langle P, Q \rangle)$ est la distance géodésique entre P, Q .
 Pour t équidistant, on obtient des points équidistant sur
 l'arc géodésique entre P et Q .

C'est la fonction que l'on utilise pour tracer les arcs géodésiques sur une sphère.



• **algo 3:**

- Triangulation initiale avec **l'icosaèdre** sphérique:
 12 points, 30 arêtes, 20 triangles.



- **L'icosaèdre** se construit par triangulation de l'enveloppe convexe de l'intersection orthogonale de 3 plans dont les longueurs des 2 arêtes ont le rapport suivant:

$$(\sqrt{5} - 1) : 2 = 2 : (\sqrt{5} + 1)$$

qui est connu sous le nom *nombre d'or*.

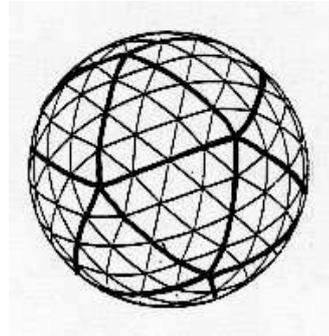
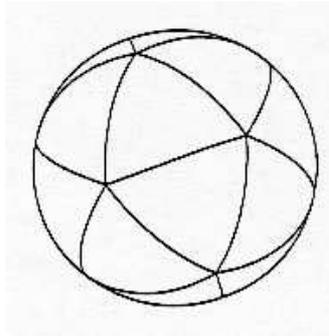
⇒ Triangulation équiangulaire.

- Ensuite subdivision régulière de chaque triangle obtenu ainsi. Après k subdivisions, on obtient

$$n^{(k)} = 2 + 4^k(n_a - n_t) \text{ points}$$

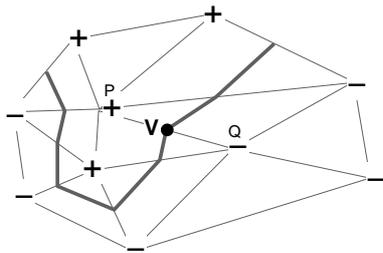
$$n_a^{(k)} = 4^k n_a \text{ arêtes}$$

$$n_t^{(k)} = 4^k n_t \text{ triangles}$$



Calcul des lignes de contour approximantes:

- En supposant que F ne varie que linéairement sur les arcs (arêtes), calculer les points d'intersection du contour sur les arcs:



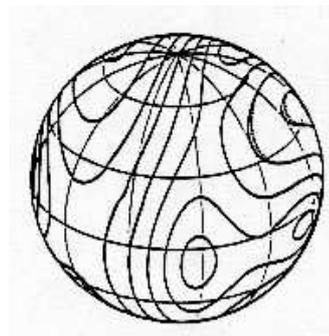
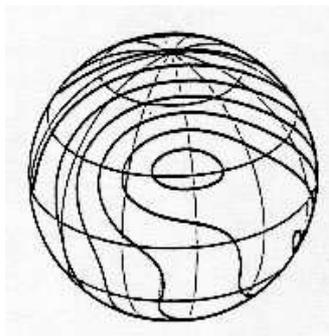
$$F(P) > \alpha$$

$$F(Q) < \alpha$$

$\Rightarrow V$ point du contour

$$V = \frac{P \cdot \sin\left(\Theta \left(\frac{\alpha - F(Q)}{F(P) - F(Q)}\right)\right) + Q \cdot \sin\left(\Theta \left(\frac{F(P) - \alpha}{F(P) - F(Q)}\right)\right)}{\sin(\Theta)}$$

- Joindre les points du contour à travers les triangles par des arcs géodésiques.



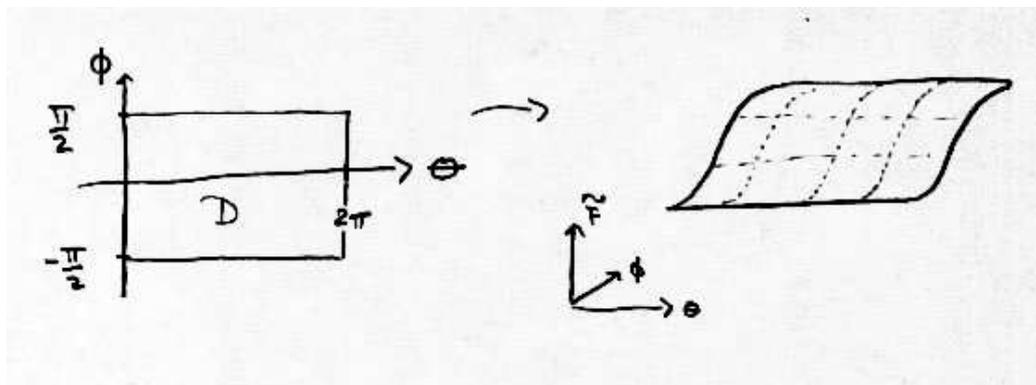
Remarques:

- \Rightarrow Les lignes de contour montrent bien la distribution des valeurs de la fonction sur la surface.

- ⇒ montre bien les positions des maxi et mini
- ⇒ peut facilement être généralisé sur des surfaces quelconques.
- ⇒ ne montre pas bien la forme géométrique de la fonction, ni si elle est lisse.

4.2 AUTRES MÉTHODES

- Visualisation de la fonction définie sur la sphère comme graphe de fonction $\tilde{F} : (\Theta, \Phi) \in D \subset \mathbb{R}^2 \rightarrow \mathbb{R}$ les 2 variables Θ, Φ étant la latitude et longitude de la paramétrisation sphérique.



- les courbes correspondantes aux lignes devant-derrrière (les 2 pôles) dans D sont des droites à hauteur constant
 - les courbes correspondantes aux lignes gauche-droite (même méridien) dans D sont des courbes identiques dans le graphe.
- Projection normale de la distance $F(\mathbf{x})$ en \mathbf{x} .

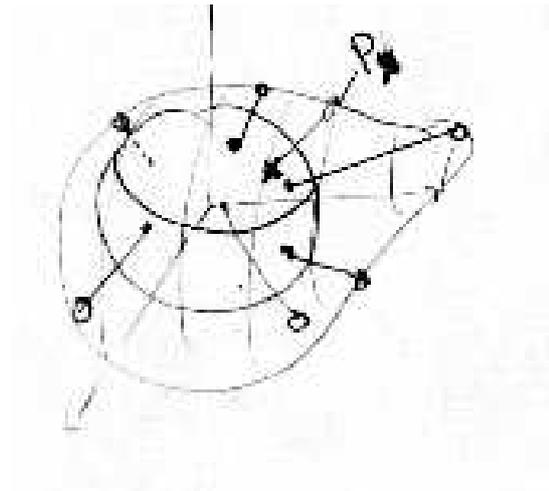
$$P_{\mathbf{x}} = \mathbf{x} + F(\mathbf{x}) \frac{\mathbf{N}}{\|\mathbf{N}\|}$$

où \mathbf{N} est la normale de la sphère. $\mathbf{N} = \mathbf{x}$ car le centre de la sphère se trouve à l'origine.

⇒ Problème d'auto-intersection de la surface ainsi obtenue si $F(\mathbf{x}) < -2$ (pour la sphère unité).

⇒ adaptation de l'échelle des valeurs de F : $\text{Min}_{\mathbf{x}} F(\mathbf{x}) = 1$,
 $\text{Max}_{\mathbf{x}} F(\mathbf{x}) = 2$

⇒ Généralisable aux surfaces fermées quelconques: $\mathbf{x} + F(\mathbf{x}) \cdot \mathcal{N}$, mais problèmes d'auto-intersection si la surface n'est pas convexe.



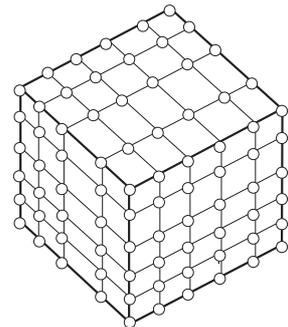
5. VISUALISATION DE DONNÉES VOLUMIQUES

La plupart des méthodes de visualisation volumique supposent que les données soient disponibles sur une grille uniforme. Comme nous l'avons vu dans les exemples, ce n'est en général pas le cas.

*Si les données ne se trouvent pas sur une grille uniforme, les méthodes de modélisation de "scattered data" interviennent pour modéliser une fonction définie sur le domaine entier qui interpole ou approxime les données. Ce modèle est ensuite utilisé pour en extraire des données sur une grille uniforme (**échantillonnage, sampling**), qui sont à l'entrée (**input**) des algorithmes de visualisation volumique.*

Nous supposons donc ici que les données volumiques $(x_i, y_j, z_k; F_{ijk})$ sont disponibles sur une grille cube uniforme.

Données volumiques uniformes: $(x_i, y_j, z_k; F_{ijk}),$
 $i = 1, \dots, N_x$
 $j = 1, \dots, N_y$
 $k = 1, \dots, N_z.$



à l'entrée de la plupart des algorithmes.

Origine: - mesure / simulation sur une grille cubique
- modélisation de "scattered data" & échantillonnage sur une grille cube

5.1 METHODES DE DECOMPOSITION DU DOMAINE

5.1.1 Cubes minces

L'idée de base consiste à placer des petits cubes dans le domaine (volume) dont un attribut (couleur, vibration, vitesse de rotation) est déterminé en fonction de la variable dépendente F à l'endroit où se trouve le cube "mince".

- Placer des cubes "minces" dans le domaine en (x_i, y_j, z_k) .
- Colorage des cubes en fonction de F_{ijk} en (x_i, y_j, z_k) .
- Paramètres à choisir:

$$N_x, N_y, N_z$$

M paramètre contrôlant l'espace entre les cubes (i.e. leur taille)

Soit $(\Delta x, \Delta y, \Delta z)$: largeur, longueur, hauteur du cube "mince".

Coordonnées du coin en bas-gauche-avant de chaque cube:

$$x_i = x_0 + (i - 1)\Delta x(M + 1) \quad i = 1, \dots, N_x$$

$$y_j = y_0 + (j - 1)\Delta y(M + 1) \quad j = 1, \dots, N_y$$

$$z_k = z_0 + (k - 1)\Delta z(M + 1) \quad k = 1, \dots, N_z$$

$(x_0, y_0, z_0) = (x_{min}, y_{min}, z_{min})$ coord. du coin en bas-gauche-avant du domaine.

$$\Delta x = \frac{x_{max} - x_{min}}{N_x(M + 1) - M} \quad \Delta y = \frac{y_{max} - y_{min}}{N_y(M + 1) - M}$$

$$\Delta z = \frac{z_{max} - z_{min}}{N_z(M + 1) - M} .$$

Exemple:

$$N_x = N_y = N_z = 5, \quad M = 1$$

domaine

$$x_{min} = y_{min} = z_{min} = 0$$

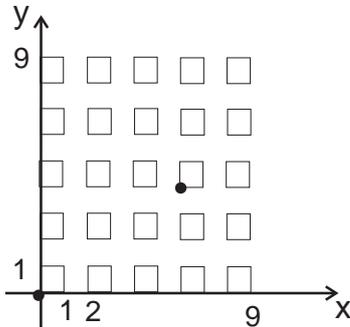
$$x_{max} = y_{max} = z_{max} = 9$$

$$\Delta x = \Delta y = \Delta z = 1$$

$$x_i = 2i - 2$$

$$y_i = 2j - 2$$

$$z_i = 2k - 2$$



A chaque sommet des cubes est attribué une couleur en fonction de la valeur F_{ijk} d'après une table de couleurs.

Exemple:

index	R	G	B
i=0,255	i	255 - i	0
i=256,511	255	i - 255	0
i=512,767	255	255	i - 512

Les faces des cubes sont ensuite colorées avec **l'algorithme de Gouraud**.

Il est important d'avoir cette méthode dans un **programme interactif** permettant de faire des **rotations du volume en temps réel** pour mieux percevoir les locations des différents cubes "minces". Cela nécessite une station de travail capable de visualiser $6 \cdot N_x \cdot N_y \cdot N_z$ polygones en temps réel. En pratique, une visualisation avec 15^{-3} cubes rend l'image déjà presque trop complexe.

- Coloriage des faces des cubes avec l'algorithme de Gouraud.

- Utilisation d'un programme interactif: rotations du volume en temps réel.

5.1.2 Cubes transparents

L'idée est d'attribuer une couleur (RGB) à chaque sommet (x_i, y_j, z_k) de la grille en fonction de la valeur F_{ijk} . L'algorithme de Gouraud ou une simple interpolation bilinéaire fait ensuite un coloriage correspondant des faces de chaque cube de la grille.

*Si on affichait ensuite toutes les faces, seul les faces extérieures seraient visibles. Pour pouvoir regarder "à l'intérieur", on calcule une image avec un simple **modèle de transparence** qui s'appelle **α -buffer**.*

- attribuer une couleur RGB à chaque sommet (x_i, y_j, z_k) :

$$C_{ijk} = (R_{ijk}, G_{ijk}, B_{ijk}) = (R(F_{ijk}), G(F_{ijk}), B(F_{ijk})) , \\ i, j, k = 1, \dots, N_x, N_y, N_z$$

$R(), G(), B() =$ table de couleurs.

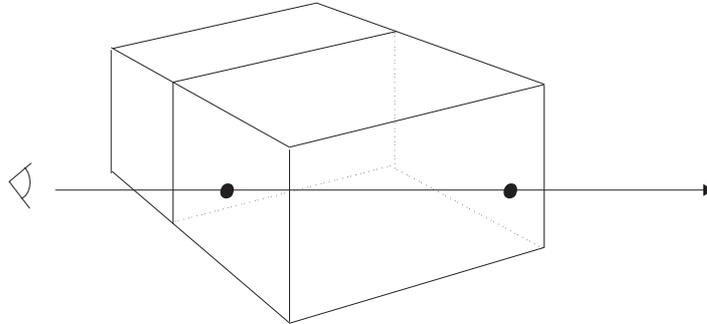
- Colorage des faces parallèles aux axes (des cubes): $3N_x N_y N_z$ rectangles (Gouraud).
- Trier les rectangles en fonction de leur distance par rapport à un point de vue.
- Afficher les rectangles dans l'ordre du fond au front en utilisant un α -buffer.

Remarque: interaction

varier la transparence α et la rotation du graphe en temps réel

- ⇒ pour chaque rotation les rectangles doivent être triés !!
- ⇒ diminution de la possibilité d'interaction.

α -buffer



Si $\alpha = \alpha_i$ pour tout i :

$$I = \alpha I_n + (1 - \alpha)\alpha I_{n-1} + (1 - \alpha)^2 \alpha I_{n-2} = \dots + (1 - \alpha)^n I_0.$$

plus α est grand, plus l'objet est transparent !

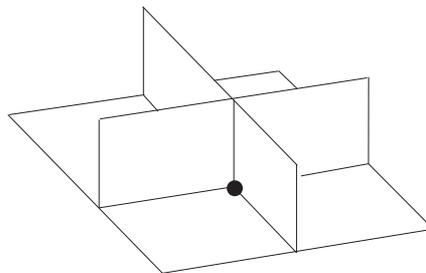
$$\alpha = 1 \quad \Rightarrow \quad I = I_n$$

$$\alpha = 0 \quad \Rightarrow \quad I = I_0$$

Ombrage de Gouraud

5.2 SLICING

L'idée principale est d'afficher simultanément les 3 grilles rectangulaires 2D parallèles aux axes, chacune obtenue en prenant une tranche en fixant une des 3 variables indépendantes à une valeur constante.



- trancher le volume parallèle aux axes
- visualiser (contouring couleur) la fonction F comme une fonction de deux variables sur ses plans en fixant une des 3 variables:

p.ex. $\bar{x} = \text{const}$

$$\text{visualiser } F_{\bar{x}}(y, z) = F(\bar{x}, y, z) \quad y_{\min} \leq y \leq y_{\max}, \\ z_{\min} \leq z \leq z_{\max}.$$

Interaction : faire varier avec la souris le point fixé $(\bar{x}, \bar{y}, \bar{z})$.

5.3 ISO-SURFACES - CONTOURING

$S_\alpha = \{(x, y, z) | F(x, y, z) = \alpha\}$ iso-surface (surface isovaleur, surface de niveau, surface de contour).

La méthode d'extraction d'iso-surfaces dépend du modèle et du type des données.

Application importante: imagerie médicale.

visualisation d'un objet composé d'un certain type de matériau: Os, cerveau, tissue doux,.....

5.3.1 Iso-surface de scattered data

données irrégulières, scattered data $(x_i, y_i, z_i; F_i) \quad i = 0, \dots, N$

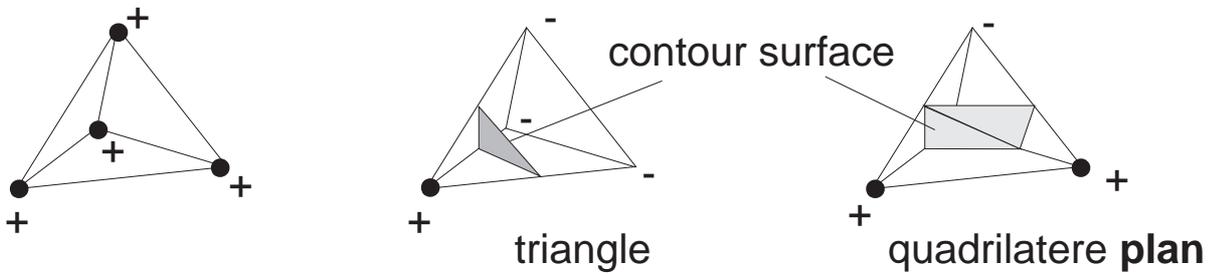
- a) Tetraédrisation de l'EC: décomposer l'EC en un ensemble de tétraèdres T_{ijkl} ayant p_i, p_j, p_k, p_l comme sommets
- b) supposer que F est linéaire par morceau sur chaque tétraèdre, la surface de contour sera donc un polyèdre avec des facettes triangulaires ou quadrilatères planes.

F linéaire est acceptable: si F ne varie pas beaucoup, ou si les tétraèdres sont suffisamment petits. Sinon, il vaut mieux faire Scattered data volumique + échantillonnage sur une grille plus fine.

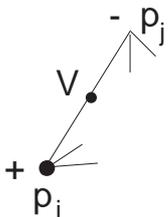
c) les F_{ijk} sont connues en les sommets des tétraèdres. Déterminer les sommets où

$$F_{ijk} < \alpha \quad \text{ou} \quad F_{ijk} > \alpha .$$

3 cas possibles :



Variation linéaire de F le long des arêtes



$$V = \left(\frac{\alpha - F_j}{F_i - F_j} \right) p_i + \left(\frac{F_i - \alpha}{F_i - F_j} \right) p_j$$

Deux algorithmes de tétraédrisation d'une EC de points

“L’algorithme le pire au monde”:

- a) générer chaque combinaison de 4 points dans P (en $O(N^4)$)
- b) pour chaque ensemble de 4 points (en $O(N)$)
 - calculer l'équation de la sphère circonscrite unique
 - s'il n'y a pas d'autres points de P à l'intérieur de la sphère, ajouter ces 4 points dans la liste des tétraèdres.

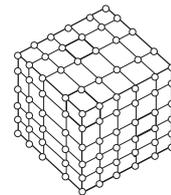
Complexité: $O(N^5)$!!!

Algorithme:

On utilise le fait que chaque face d'un tétraèdre appartient soit à un autre tétraèdre, soit à l'enveloppe convexe de P .

- a) trouver le premier tétraèdre, le mettre dans la liste
- b) pour chaque tétraèdre de la liste
 - {
 - pour chaque face de ce tétraèdre
 - {
 - IF elle n'est pas la face d'un autre tétraèdre de la liste
 - THEN
 - {
 - pour chaque point de P (sauf les 3 de la face actuelle)
 - calculer la sphère circonscrite
 - s'il n'y a pas d'autres points à l'intérieur, ajouter les 4 points comme tétraèdre à la liste.

5.3.2 Algorithme 'MARCHING CUBE'



(Lorensen, Cline 1987)

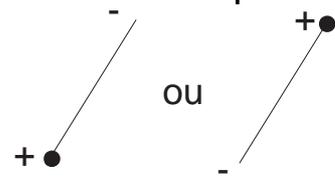
- données sur une grille cubique.
- l'algorithme "marching cubes" est appliqué à un cube après l'autre.

marcher à travers chaque cube.

- un cube est un sous-ensemble du domaine de sommets (x_a, y_b, z_c) , $a = i, i + 1$, $b = j, j + 1$, $c = k, k + 1$.

- dans chaque cube un ensemble de triangles est calculé qui contribue à approximer S_α .
- F est supposée être linéaire sur les arêtes du cube.
- évaluation en chaque sommet (x_i, y_j, z_k) de la grille, et tester si $F_{ijk} > \alpha$ ou $F_{ijk} < \alpha$.

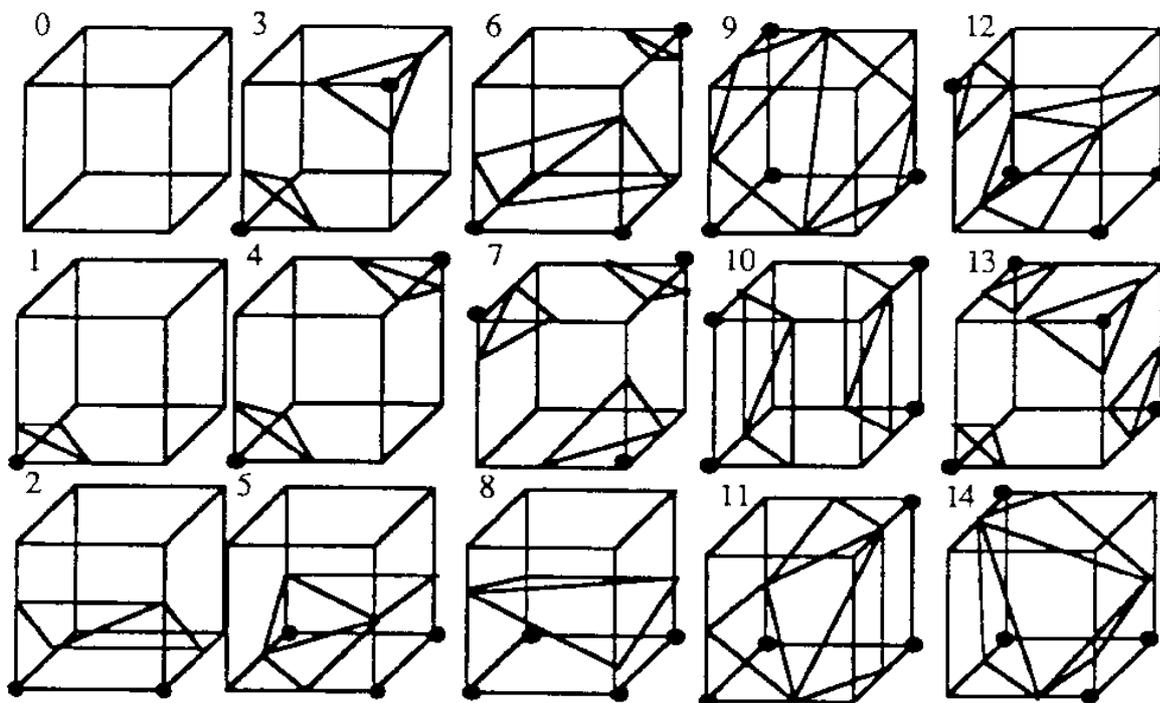
⇒ 8 sommets par voxel ⇒ $256 = 2^8$ possibilités

⇒  cas symétriques ⇒ 128 possibilités

⇒ seul les configurations avec au plus 4 valeurs supérieures à l'isovaleur α sont à considérer.

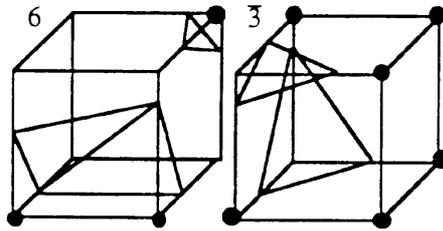
⇒ considérons de plus les équivalences par rotation

⇒ ... finalement: restent **15 configurations** à tester et à trianguler.



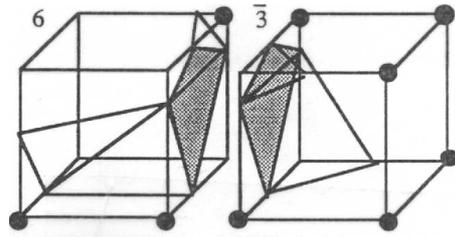
Il y a des configurations où une autre triangulation peut changer la topologie de la surface, p.ex. 14.

pb: certaines constellations peuvent produire des iso-surfaces avec des trous.

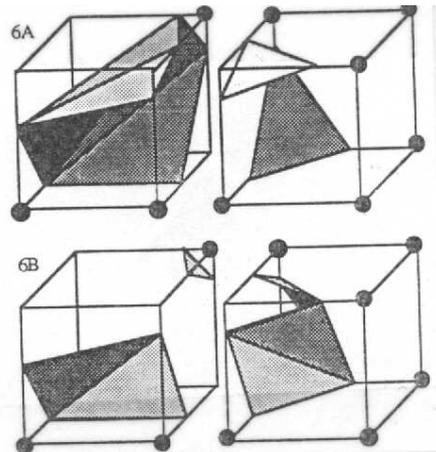


Il y a 2 possibilités pour traiter ce cas spécial:

a) ajouter les 2 triangles à la liste des triangles du contour:



b) trianguler autrement les sommets sur les arêtes (2 possib.):



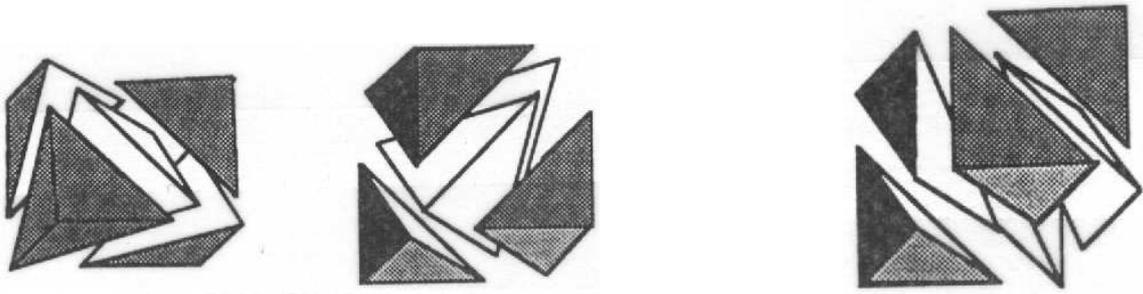
Visc p.37, unten links

voir (Nielson, Hamann 1990)

5.3.2 Algorithme 'MARCHING TETRAÈDRE'

pour éviter la création de trous avec le MC, décomposer chaque voxel en un ensemble de tétraèdres.

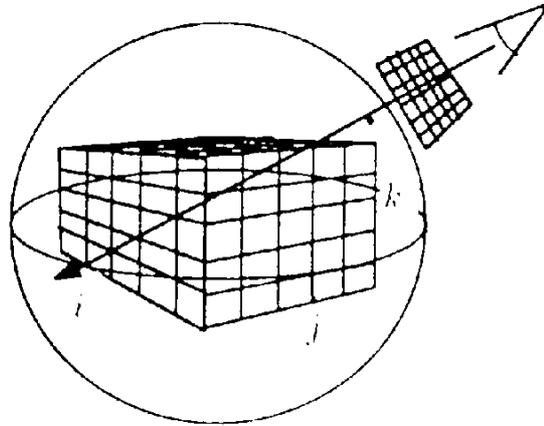
2 possibilités:



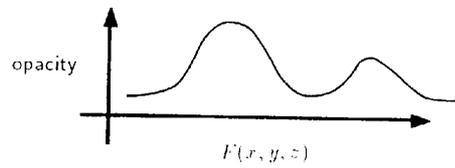
! Décomp. en 5 tétraèdres: pour maintenir la continuité C^0 il faut alterner les 2 décompositions (gauche, droite) d'un voxel à l'autre !

- Le nombre de sous-domaines à traiter par MT est multiplié par 5 ou 6 par rapport au MC classique.
- Le nombre de triangles est plus petit dans le MT que dans le MC, donc le nombre totale n'est pas multipliée par 5 ou 6. MT produit en moyenne 150 % à 250 % plus de triangles que MC.

5.4 VOLUME RENDERING - RAY CASTING



Ray casting volume rendering



mapping from density to opacity

— under construction —

Visualisation volumique

1

Equation fondamentale:

Considérons un objet translucide émettant une intensité I_e , et ayant une transparence T . Un rayon lumineux d'intensité I_a pénétrant dans cet objet, en ressortant avec une intensité $I_m = T \cdot I_a + I_e$

Suite d'objets translucides

Considérons une suite d'objets translucides, émettant une intensité I_i et ayant une transparence t_i . $i = 1, \dots, N$.

Soit F_0 l'intensité d'un rayon lumineux pénétrant cette suite d'objets translucides. On note F_i l'intensité de ce rayon à la sortie de l'objet i . On pose $I_0 = F_0$.

$$\text{On a : } F_i = t_i F_{i-1} + I_i$$

$$\text{de sorte que : } \begin{aligned} F_1 &= t_1 I_0 + I_1 \\ F_2 &= t_2 F_1 + I_2 = (t_2 t_1) I_0 + t_2 I_1 + I_2 \\ &\vdots \end{aligned}$$

$$\begin{aligned} F_N &= (t_N \dots t_1) I_0 + (t_N \dots t_2) I_2 + \dots + t_N I_{N-1} + I_N \\ &= \sum_{j=1}^N \left(\prod_{k=j+1}^N t_k \right) I_j \end{aligned}$$

$$\left(\text{avec pour convention } \prod_{j=N+1}^N t_j = 1 \right)$$

Il existe deux manières d'évaluer F_N .

→ de l'arrière-plan vers le premier-plan:

$$\begin{aligned} F &= F_0 \\ i &= 1, \dots, N. \\ F &\leftarrow t_i \cdot F + I_i. \end{aligned}$$

→ du premier-plan vers l'arrière-plan:

$$\begin{aligned} F &= I_N \quad (I_0 = F_0) \\ T &= 1 \\ i &= N, \dots, 1. \\ T &\leftarrow t_i \cdot T \\ F &\leftarrow T \cdot I_{i-1} + F \end{aligned}$$



Passage à la forme continue

On pose $\alpha_i = (1 - t_i)$: opacité de l'objet i
 $I_i = \alpha_i C_i$: C_i : valeur de l'objet i

$$\text{alors : } F_i = t_i F_{i-1} + I_i$$

$$\Leftrightarrow F_i = (1 - \alpha_i) F_{i-1} + \alpha_i C_i$$

$$\Leftrightarrow (F_i - F_{i-1}) = -\alpha_i (F_{i-1} - C_i)$$

Pour passer au continu : $F_i = F(x + \Delta x)$

$$F_{i-1} = F(x)$$

$$C_i = C(x + \Delta x)$$

$\alpha_i = \delta(x) \Delta x$ ou δ : fonction de densité par unité de longueur.

$$F(x + \Delta x) - F(x) = -\delta(x) \Delta x (F(x) - C(x + \Delta x))$$

si $\Delta x \rightarrow 0$

$$F'(x) = -\delta(x) [F(x) - C(x)].$$

Pour intégrer cette équation différentielle, on résout l'équation homogène associée : $F' = -\delta F$, solution $F(x) = F(x_0) e^{-\int_{x_0}^x \delta(s) ds}$

et on remplace donc $F(x)$ par $f(x) \times e^{-\int_{x_0}^x \delta(s) ds}$, en imposant la condition $f(x_0) = F(x_0)$.

f doit alors être solution de : $f'(x) = \delta(x) C(x) e^{-\int_{x_0}^x \delta(s) ds}$
donc $f(x) = \int_{x_0}^x \delta(u) C(u) e^{-\int_{x_0}^u \delta(s) ds} du + F(x_0)$

d'où finalement : $F(x) = \left(\int_{x_0}^x \delta(u) C(u) e^{-\int_{x_0}^u \delta(s) ds} du \right) \times e^{-\int_{x_0}^x \delta(s) ds} + F(x_0) \times e^{-\int_{x_0}^x \delta(s) ds}$

En entrée : des coupes scanners CT $C_{i,j}^k$

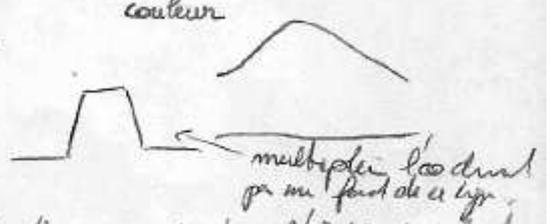
Volume Formation

réplication, interpolation $\Rightarrow V_{i,j,k}$ - densité

Volume Classification

densité \Rightarrow n matériaux \Rightarrow opacité couleur

$V_{i,j,k}$ | - densité
- n matériaux
- opacité
- couleur



Surface detection

- * densité \Rightarrow indice de refraction \Rightarrow gradient de l'indice de refraction \Rightarrow modification des matériaux et de l'opacité.
- * choix au départ de l'opacité et de la couleur au moment des matériaux qui doivent être visualisés.

Rotation

- * rotations 2D anti-aliasing cubes des axes orthogonaux \Rightarrow l'information reste vectorielle \Rightarrow toutes les données sont "autant que possible" conservées.

Shading

- * produit scalaire (gradient refraction x direction lumière) \Rightarrow modification des matériaux et de l'opacité.

Projection // axe

front-to-back $\left\{ \begin{array}{l} * t_{i,m}^a = t_i^a \times t_{i,m} \text{ (transparences accumulées)} \\ * \text{entente } (r,g,b) \neq \text{entente } (r,g,b) \times t_{i,m}^a \end{array} \right. \text{ avec } i \text{ tel } t_i^a < \epsilon$
 \Rightarrow image pix 103 r, g, b.

Ombre ^{moyen}: modification des couleurs et de l'opacité de chaque voxel.

but: simuler la réflexion de la lumière sur les surfaces de fort gradient. i.e., là où les données varient le plus.

en 2D:

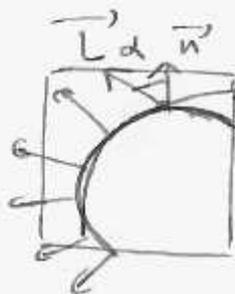
①



gradient: $\frac{\partial F}{\partial x}$ $\frac{\partial F}{\partial y}$

②

lumière: direction, \vec{L} normée en sens inverse



$\langle \vec{L}, \vec{n} \rangle$: grand si //
= 0 si \perp .

\Rightarrow couleur, opacité deviennent nuls si la surface n'est pas éclairée par les rayons lumineux.

Programmation: en chaque voxel, calculer le gradient discretisé F_{ijk}

$$\begin{matrix} G_{ijk}^x \\ G_{ijk}^y \\ G_{ijk}^z \end{matrix} \begin{pmatrix} F_{i+1jk} - F_{i-1jk} \\ F_{ij+1k} - F_{ij-1k} \\ F_{ijk+1} - F_{ijk-1} \end{pmatrix}$$

104

↑ aux bord.

Calculer $\vec{S} = \left(\begin{matrix} G_x \\ G_y \\ G_z \end{matrix} \right), \left(\begin{matrix} L_x \\ L_y \\ L_z \end{matrix} \right)$ multiplier la transparence et la couleur par ce produit scalaire

Opacité
~~Opacité~~

→ Rendu volumique

(1)

Ideé: Le rendu volumique produit une visualisation en ^{globale} profondeur des données. Les autres méthodes présentées ^{pour l'instant} donnent soit des informations locales (ex: surfaces iso-valeurs), soit des informations globales avec le défaut que les éléments du devant de la scène cachent ceux qui se trouvent à l'arrière-plan. (ex: cubes rouges).
→ en assimilant l'ensemble de données à (transparence)

un milieu semi-transparent traversé par des rayons lumineux. Ainsi les éléments du premier plan ne cachent pas ceux de l'arrière-plan qui in leur opacité est grande.

1) Propagation des rayons lumineux

~~→ forme d'onde~~

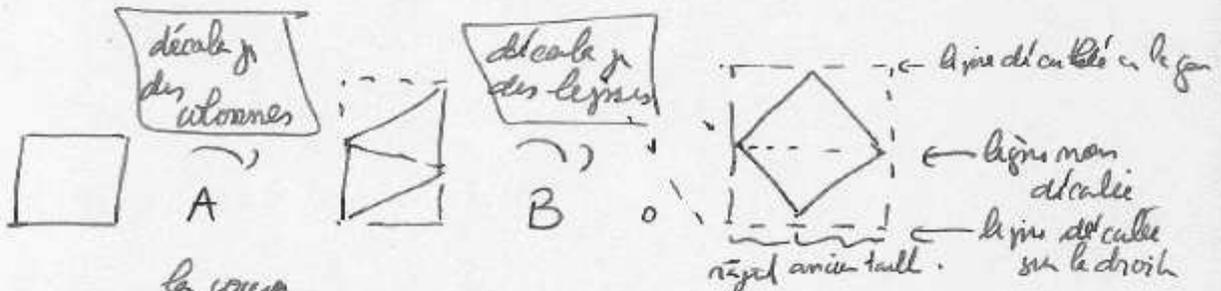
a) Traversée d'un objet translucide

On parle - - -

b) Traversée d'une suite d'objets

c) Parcours de la fonction continue.

Implémenter la rotation : composition de 2 transformations qui n'agissent que sur les lignes / colonnes :



la taille de l'image (du volume ~~plane~~ donc) grandit.

R : rotation θ , A : débale y des colonnes
B : débale x des lignes

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = R \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ +\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{SO}(\theta)$$

A doit laisser ~~le~~ 'mécanisme'

$$\Rightarrow A = \begin{pmatrix} 1 & 0 \\ \alpha & \beta \end{pmatrix}$$

$$\text{de m} : B = \begin{pmatrix} \gamma & \delta \\ 0 & 1 \end{pmatrix}$$

$$\text{donc } BA = \begin{pmatrix} \gamma & \delta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \alpha & \beta \end{pmatrix} = \begin{pmatrix} \gamma + \delta\alpha & \delta\beta \\ \alpha & \beta \end{pmatrix}$$

On compare avec R

$$\Rightarrow \begin{cases} \alpha = \sin \theta \\ \beta = \cos \theta \\ \delta\beta = -\sin \theta \\ \gamma + \delta\alpha = \cos \theta \end{cases} \Rightarrow \begin{cases} \delta = -\tan \theta \\ \gamma = \tan \theta \sin \theta + \cos \theta = \cos \theta + \frac{\sin^2 \theta}{\cos \theta} = \frac{1}{\cos \theta} \end{cases}$$

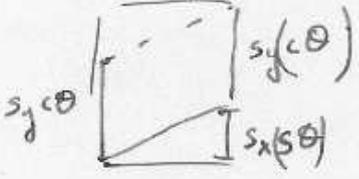
Solution $A = \begin{pmatrix} 1 & 0 \\ s\theta & c\theta \end{pmatrix}$ $B = \begin{pmatrix} \frac{1}{c\theta} & -t\theta \\ 0 & 1 \end{pmatrix}$ (4)

Changements de taille

1

taille s_y 

$$\begin{pmatrix} 1 & 0 \\ s\theta & c\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y\cos\theta + xs\theta \end{pmatrix}$$



nouvelle taille $s'_y = s_x \sin\theta + s_y \cos\theta$.

2

$$\begin{pmatrix} \frac{1}{c\theta} & -t\theta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \frac{1}{c\theta}x - t\theta y \\ y \end{pmatrix}$$



$$s'_x = s_x / c\theta - (t\theta) \times s'_y$$

Problème: $c\theta = 0$ (à $\theta = 90^\circ$) non défini.

=> autre décomposition en 3 matrices cette fois:

$$\begin{pmatrix} 1 & -t(\theta/2) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ s\theta & 1 \end{pmatrix} \begin{pmatrix} 1 & -t(\theta/2) \\ 0 & 1 \end{pmatrix} : OK \text{ pour } \theta \in [-90^\circ, 90^\circ]$$

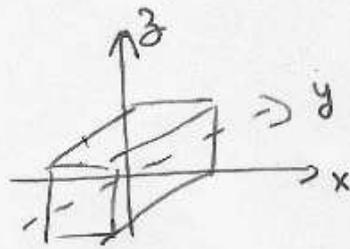
$[-180^\circ, 180^\circ]$

2) Rotation

(2)

On ~~peut~~ deux méthodes sont possibles pour -
Le ~~bon~~ des équations de propagation du rayon lumineux
possibles plus le fait précédent permettait de
programmation d'implémentation peut être implémentée directement
pour des rayons lumineux arbitraires à un des axes
du cube. Pour les directions obliques, deux méthodes
sont possibles; soit on trace des rayons obliques,
auquel cas la ~~dis~~ ~~it~~ ~~fa~~ ~~it~~ ~~est~~ ~~tr~~ ~~ab~~ ~~ar~~ ~~de~~ ~~le~~ ~~p~~ le problème
de la détermination des voxels traversés et de la ~~log~~ ~~ic~~ ~~ité~~
traverse des axes voxels ~~est~~ ~~tr~~ ~~ivial~~, soit on
effectue préalablement une rotation de l'espace de données.
Nous donnons ces lettres deux dernières solutions.

~~Rotation des axes 3D?~~



Compositions de rotations 2D:



3) Schema complet du rendu velleiniqu.

Voir feuille de je faite.

4) Formatia du volume

Different, coupes

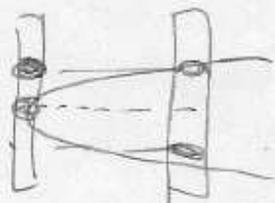


Plan: même précision mesurés sur les 3 axes

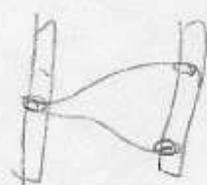
① \Rightarrow interpolater (linéar, cubiq - ...). : introduit des artefacts.

② on replicate -

③ heuristiques



interpolé



autr méthode.

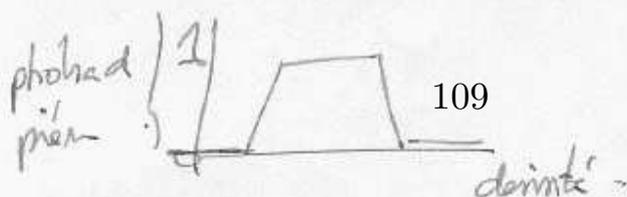
5) Détection de ~~surface~~ Classification

Plan: assigner à chaque voxel une contenue

émise et une opacité.

~~Class~~ Procédi : ~~une~~

voxel \rightarrow probabilité de présence de un matériau \rightarrow combinaison des intensité et opacité des mat



Pour nous: "à la main, avec x0" \Rightarrow voir x0.

6) Détection de Surface et Ombraje

(6)

(moyen: ombraje ~~type~~ ~~gouraud~~, par modification de l'intensité et de ~~la~~ ~~couleur~~ de l'opacité.

Voir feuille déjà écut.

→ Détection de surface: Calcul du gradient.

→ Ombraje: multiplication de l'opacité et de l'intensité par le gradient calculé et de la direction de la lumière.

produit scalaire
↓

Attention: il ne faut pas calculer le produit scalaire avant d'avoir effectué

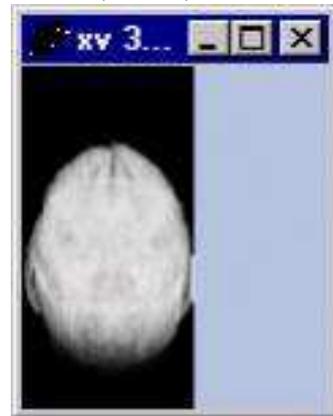
la rotation.



$L=(1\ 0\ 0)$ trans 0.9 coul 250 $L=(0\ 1\ 0)$ trans 0.9 coul 250 $L=(0\ 0\ 1)$ trans 0.9 coul 250



$L=(-1\ 0\ 0)$ trans 0.9 coul 250 $L=(0\ -1\ 0)$ trans 0.7 coul 200 $L=(1\ 1\ 1)$ trans 0.9 coul 250



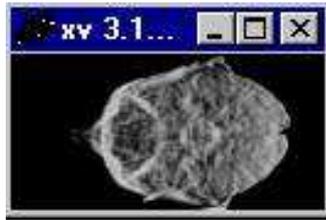
$L=(1\ 0\ 0)$ trans 0.7 coul 250 $L=(0\ 0\ 1)$ trans 0.7 coul 200 $L=(1\ 0\ 0)$ trans 0.7 coul 200



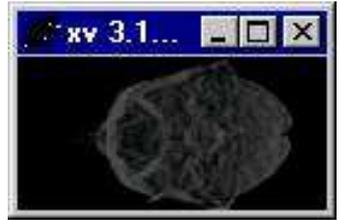
0.7/200



100/0.9/250



100/0.3/100



100/0.3/250



100/0.9/200



-100/0.9/100



010/0.9/100



0-10/0.9/100



100/0.9/100 (50)



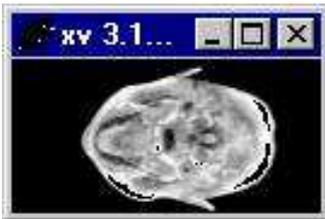
100/0.9/250 (50)



100/0.9/250 (90)



-100/0.9/250 (90)



010/0.9/250 (90)



0-10/0.9/250 (90)



010/0.9/250 (70)



0-10/0.9/250 (70)



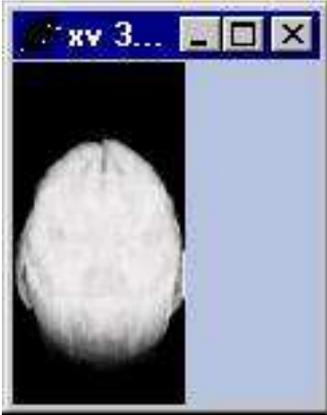
010/1/250 (70)



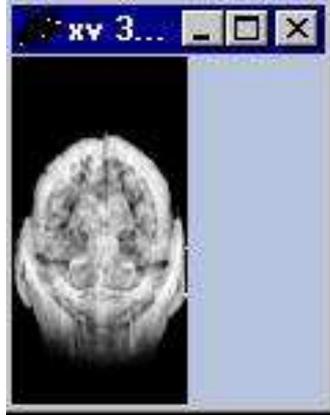
100/1/250 (70)



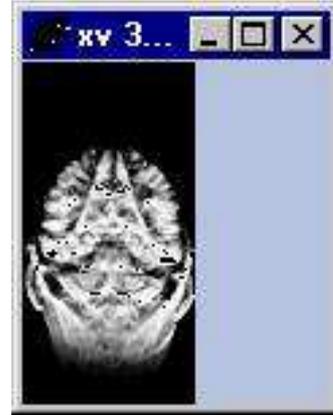
010/1/250



100/0.9/250



100/0.9/250 (100-70)



010/0.9/250 (100-70)

6. VISUALISATION DE CHAMPS DE VECTEURS

Des équations différentielles sont souvent utilisées pour la description de phénomènes naturels. En dynamique des fluides, certains peuvent être interprétés géométriquement comme champs de vecteurs ou de tenseurs. Des grands domaines d'applications comme la simulation numérique, l'analyse de données d'origine de mesures et la visualisation graphique traitent ces champs de vecteurs.

Equa. diffs. pour la description de phénomènes naturels

⇒ **Champs de vecteurs / tenseurs**

Domaines d'application:

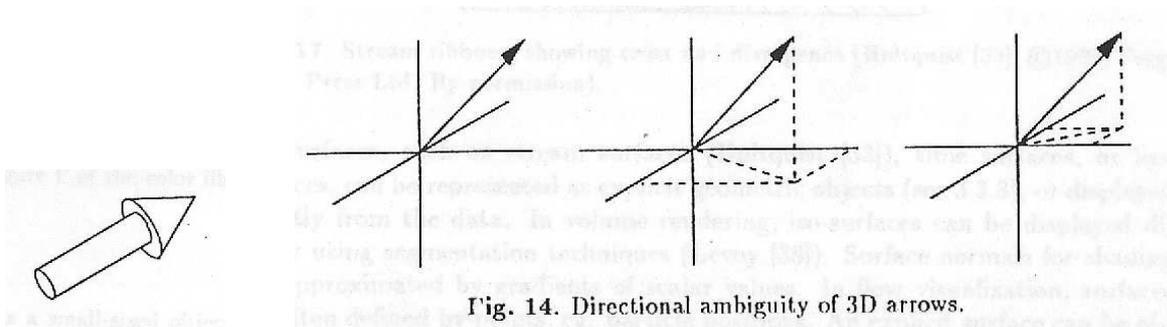
- industrie automobile, aéronautique, spatiale, navale
- prévisions/simulations en météo
- analyse du climat
- construction de turbine, centrales électriques

⇒ **mieux comprendre** les données d'origine mesure/simulations.

6.1 METHODES DIRECTES

6.1.0 Visualisation "Hedgehog"

FH Post, T van Walsum: Fluid flow visualization, 1er Dagstuhl



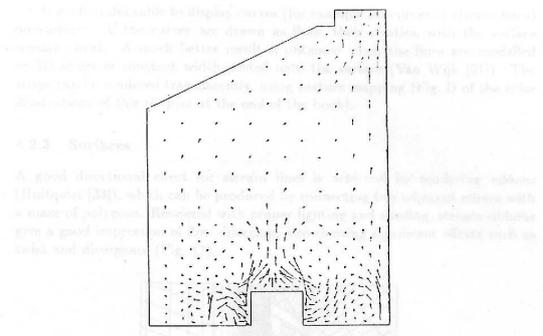
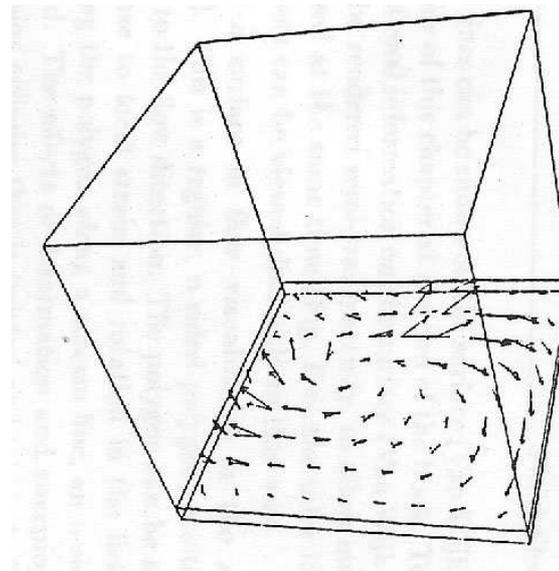


Fig. 13. 2D arrow plot (Bertrand and Tanguy [2], ©1988, John Wiley & Sons, Ltd. By permission).



KLassen, Harrington: Shadowed hedgehogs, Proc. Vis'91

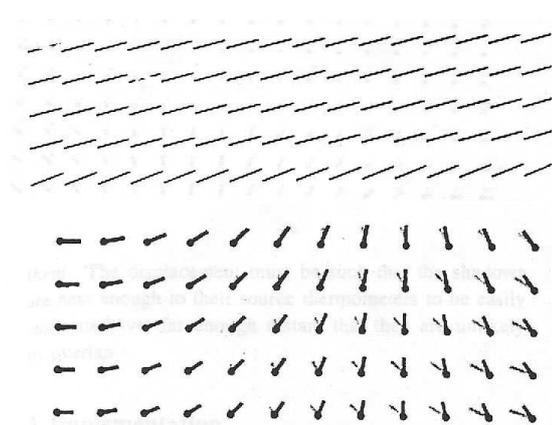
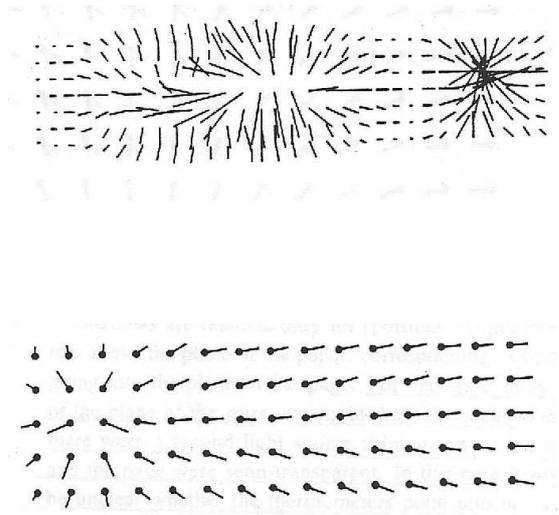


Figure 5 Bulb-shaped vectors with shadows. Bulbs are in the plane of the page, at grid points. Some of the thermometers completely obscure their own shadows.

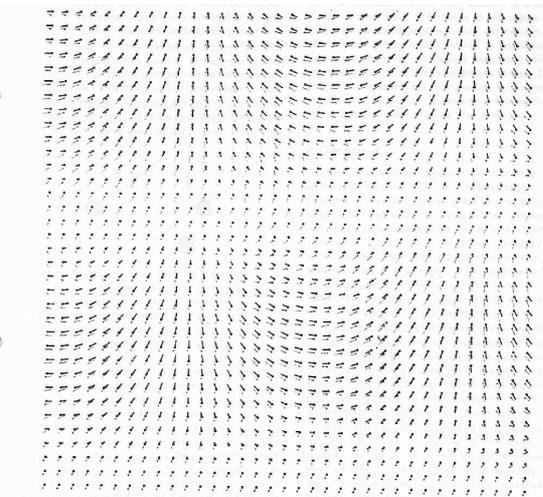


Figure 6 A test image showing thermometers displaced above the plane. The field orientation varies smoothly through all possible angles. Along the top and bottom it is coming out of the plane of the page, in between it rotates into the plane of the page and back.

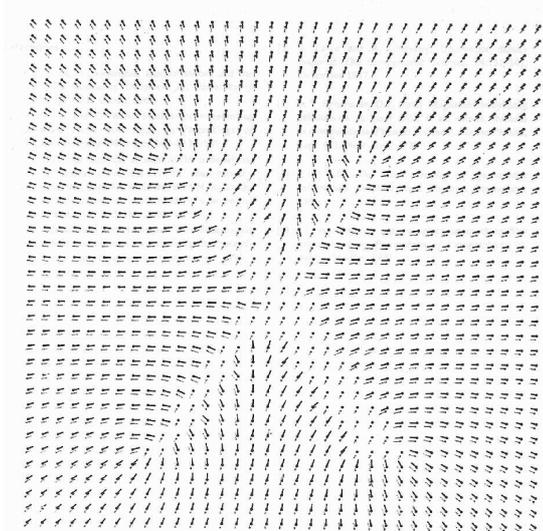


Figure 8 The field slightly above an idealized photoreceptor carrying an image of an X. The field was modeled using a point charge at each pixel of a 144 spot per inch image of a 12 point X, along with an infinite ground plane biasing the field in the direction perpendicular to the page.

6.1.1 Visualisation de champs de vecteurs stationnaires

Champs de vecteurs dans \mathbb{R}^n

$$\mathbf{v} : \mathbb{R}^n \rightarrow T\mathbb{R}^n \simeq \mathbb{R}^n$$

$T\mathbb{R}^n$ espace tangentiel. Analogue sur une variété $S \subset \mathbb{R}^n$

$$\mathbf{v} : S \subset \mathbb{R}^n \rightarrow TS$$

où TS désigne l'espace tangentiel à S .

En pratique: $n = 3$ et $S \subset \mathbb{R}^3$ est une surface.

Ceci décrit un système autonome d'équations différentielles ordinaires

$$\begin{aligned} \text{grad } f &= \mathbf{v}(\mathbf{x}) && \Leftrightarrow \\ \frac{\partial f}{\partial x_1} &= v_1(x_1, \dots, x_n) \\ &\vdots \\ \frac{\partial f}{\partial x_n} &= v_n(x_1, \dots, x_n) \end{aligned}$$

(Ce que l'on cherche est l'application $f : \mathbb{R}^n \rightarrow \mathbb{R}$).

Pour pouvoir déterminer la trace d'une particule dans un courant, il faut d'abord la position initiale. Ensuite, il faut résoudre un système d'EDO's.

Déterminer la trace d'une particule dans un courant

- choisir position initiale
- résoudre système EDO.

Soit $I \subset \mathbb{R}$ un intervalle fermé de \mathbb{R} , la trace de la particule peut être décrite par une courbe paramétrique

$$\begin{aligned} \mathbf{c} : I &\rightarrow \mathbb{R}^n \\ t &\mapsto (c_1, \dots, c_n) \end{aligned}$$

avec

$$\begin{aligned}\frac{\partial c_1}{\partial t} &= v_1(c_1, \dots, c_n) \\ &\vdots \\ \frac{\partial c_n}{\partial t} &= v_n(c_1, \dots, c_n)\end{aligned}$$

c.à.d. chaque point de la courbe (pour chaque paramètre t) doit être tangent au champs \mathbf{v} .

Exemple $n = 2$:

Soit donné un champs de vecteurs 2D $\mathbf{v} = \mathbf{v}(x_1, x_2)$. On cherche $\mathbf{c} : I \rightarrow \mathbb{R}^2, t \mapsto (c_1, c_2)$ avec

$$\begin{aligned}\frac{\partial c_1}{\partial t} &= v_1(c_1(t), c_2(t)) \\ \frac{\partial c_2}{\partial t} &= v_2(c_1(t), c_2(t))\end{aligned}$$

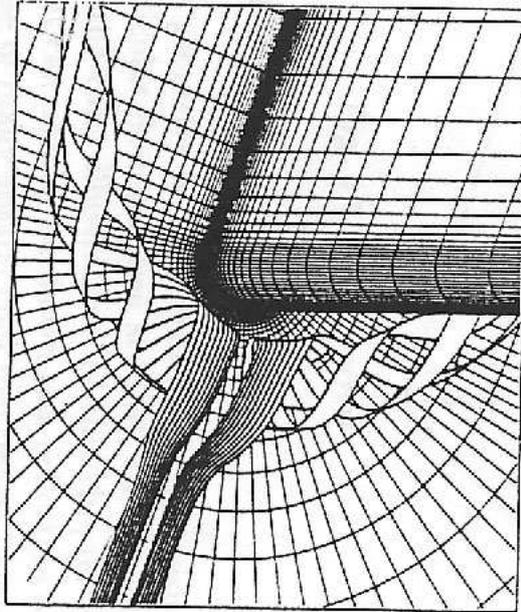
— figure —

Tracer ces courbes d'intégration est à la base des méthodes de lignes de courant.

En général, les données sont disponibles sous forme de champs de vecteurs, mesurés ou calculés, **discrets**. Ils sont donnés sur des grilles régulières ou irrégulières.

Donc la fonction \mathbf{v} (i.e champs de vecteurs) n'est pas connue continuellement. Pour pouvoir utiliser les méthodes numériques de résolution d'EDO's, on a besoin de connaître \mathbf{v} de manière continue.

— under construction —



6.1.2 Visualisation de champs de vecteurs dans le temps

— under construction —

6.2 METHODES TOPOLOGIQUES

— under construction —

Helman, Hesselink, Repres. and display of vector field topology in fluid flow data sets, IEEE Computer 1989:

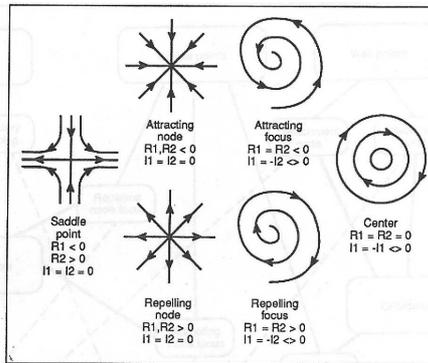


Figure 5. Classification criteria for critical points. R_1 and R_2 denote the real parts of the eigenvalues of the Jacobian; I_1 and I_2 denote the imaginary parts.

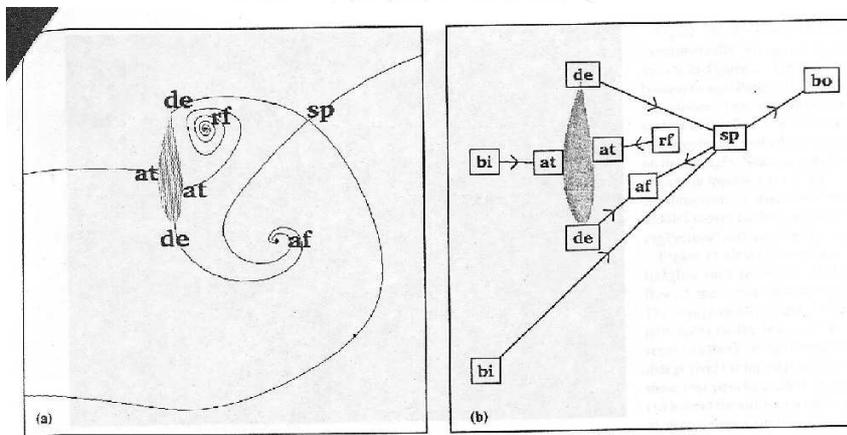


Figure 7. Tangent curves (instantaneous streamlines) and points in the computed flow around an airfoil at $t=2$ (a). Corresponding graph of flow topology (b).

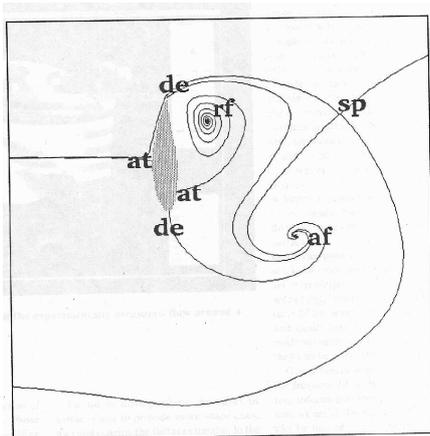


Figure 8. Tangent curves (instantaneous streamlines) and points in the computed flow around an airfoil at $t=3$.

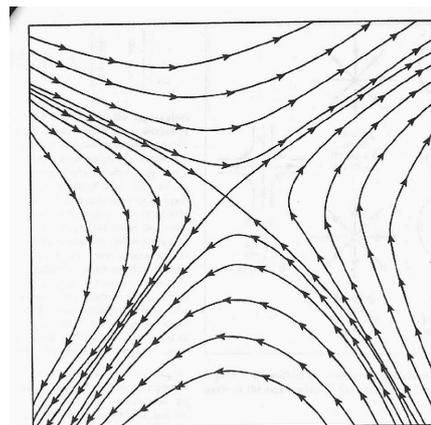


Figure 4. Vector field and tangent curves near a saddle point.

Helman, Hesselink, Visualizing vector field topology in fluid flows, IEEE CG&A

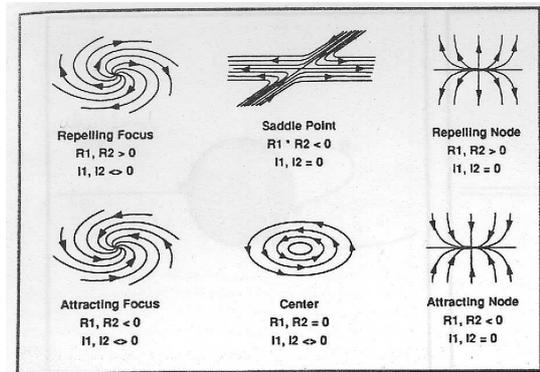


Figure 1. Classification criteria for critical points. R1 and R2 denote the real parts of the eigenvalues of the Jacobian, I1 and I2 the imaginary parts.

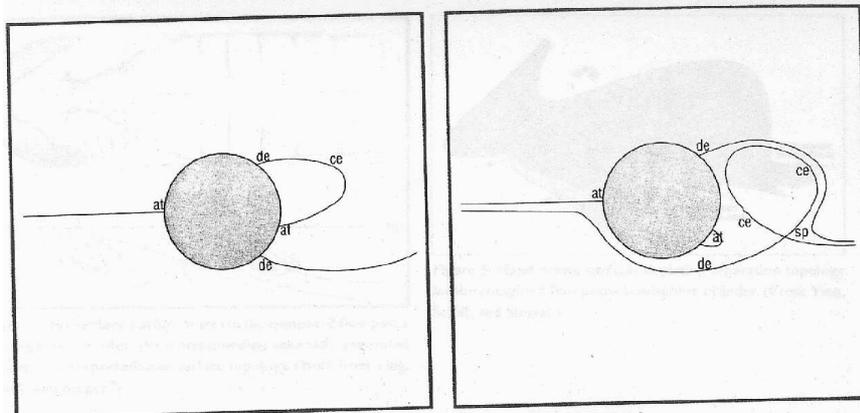


Figure 2. Topology schematics for two time steps in the computed flow around a circular cylinder.

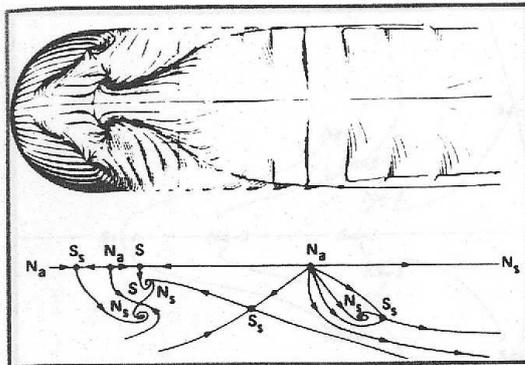


Figure 4. (a) Surface particle traces in the computed flow past a hemisphere cylinder. (b) Corresponding manually generated schematic interpretation of surface topology. (Both from Ying, Schiff, and Steger.⁸)

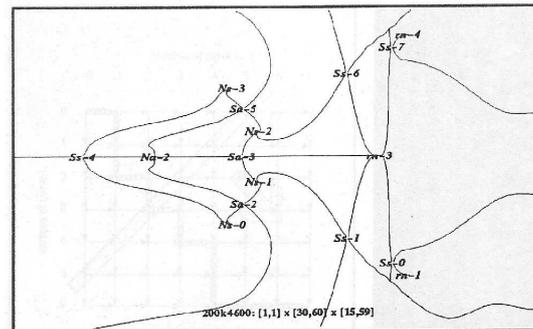


Figure 6. Computer-generated skeleton of surface topology corresponding to Figure 4.

Globus, Levit, Lsinski, A Tool for the topology of three-dimensional vector fields, Proc. Vis'91

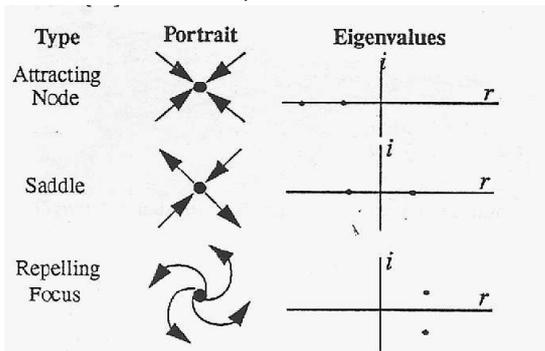


Figure 1: Classification of two dimensional critical points.

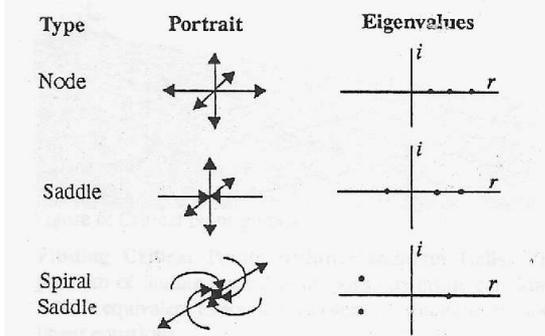


Figure 2: Classification of three dimensional critical points.

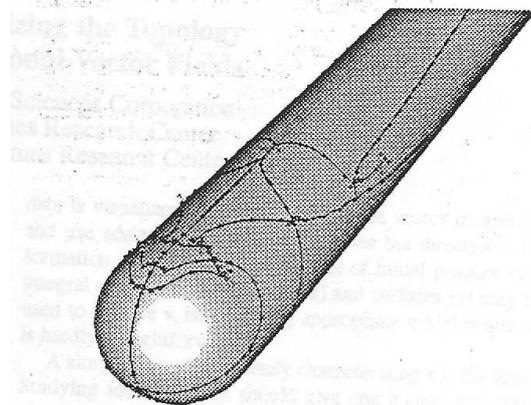


Plate 1b: TOPO's hemisphere cylinder skin friction surface topology. Data [34].

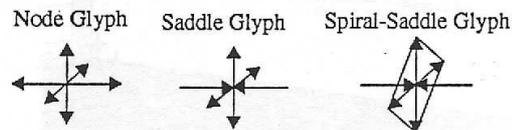


Figure 3: Critical point glyphs.

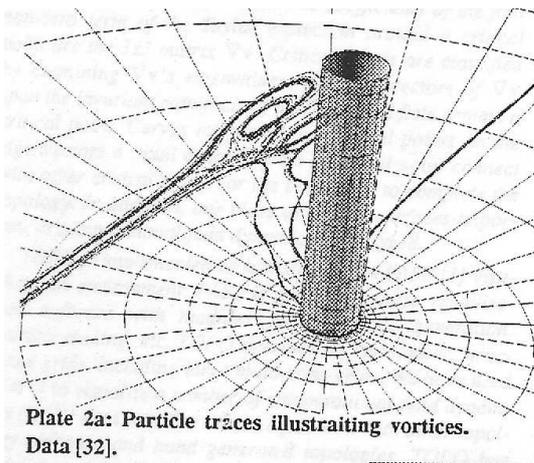


Plate 2a: Particle traces illustrating vortices. Data [32].

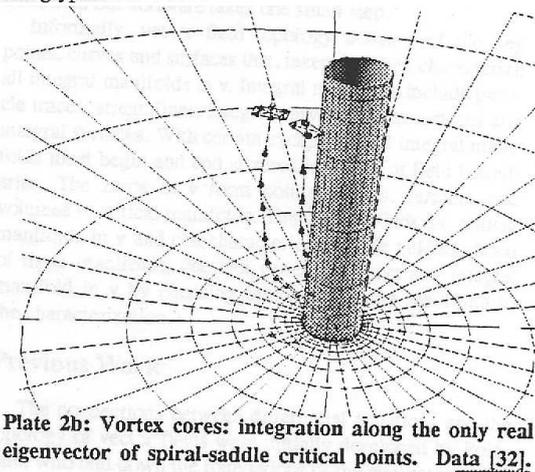


Plate 2b: Vortex cores: integration along the only real eigenvector of spiral-saddle critical points. Data [32].

6.3 LINE INTEGRAL CONVOLUTION

B. Cabral, Imaging vector fields using line integral convolution,

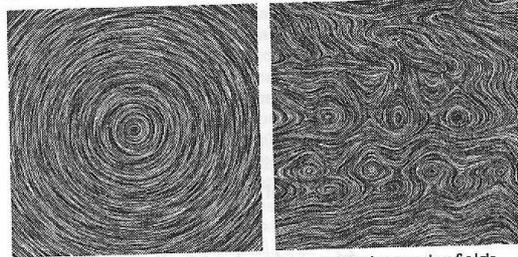


Figure 4: Circular and turbulent fluid dynamics vector fields imaged using LIC over white noise.

— under construction —

7. VISUALISATION MULTIRESOLUTION DE DONNÉES SCIENTIFIQUES

— under construction —

Sites WWW

SciVi Research

<http://www.llnl.gov/graphics/>

<http://davinci.informatik.uni-kl.de:8000/>

<http://www.cs.sunysb.edu/vislab/>

<http://www.sm.go.dlr.de/schorsch/Professional/>

http://wwwcg.twi.tudelft.nl/scientific/SciVi_e.html

Volume Rendering:

<http://www-graphics.stanford.edu/software/>

<http://www-graphics.stanford.edu/projects/volume/>

Geophysical Data

<http://web.ngdc.noaa.gov/seg/segd.html>

Bibliographie

1. Hagen, Müller, Nielson (eds): *Focus on Scientific Visualization*, Springer, (1993).
2. Nielson G., H. Hagen, and H. Müller (eds.): *Scientific Visualization - Overview, Methodologies and Techniques*, IEEE Computer Society Press, (1997).
3. Nielson G.: Course Notes for “Algorithms and Techniques for Scientific Data Visualization”, ViSC Summer School CEA, INRIA, EDF, Le Breau (1993).
4. Domik G.: A Tutorial on Scientific Visualization, Technical Report University of Colorado, Department of Computer Science, (1993).
5. IEEE Transactions on Visualization and Computer Graphics (TVCG), Revue internationale, depuis 1996.
6. Proceedings Visualization, IEEE, depuis 1989, annuel.
7. Foley, Van Dam, van Dam, Feiner, Hughes: *Computer Graphics: Principles and practice*, second edition, (1990).

Chapitre 3

8. Franke R., Nielson G.: Scattered data interpolation and Applications: A tutorial and Survey, in *Geometric Modeling*, H. Hagen, D. Roller (eds.), (1990). *Survey on SD methods*
9. Nielson G.: Minimum Norm Interpolation in triangles, *SIAM J. Numer. Anal.* 17(1), 44–62, (1980).
10. Nielson G.: A method for interpolating scattered data upon a minimum norm network, *Mathematics and Computaion* 40, 253–271, (1983).

11. Nielson G.,: The side-vertex method for interpolation in triangles, *Journal of Approx. Theory* 25, 318–336, (1979).
12. Nielson G., Foley T., Hamann B., Lane D.: Visualization and modeling of scattered multivariate data, *IEEE CG& Appl.* 11, 47–55, (1991).
13. Pottmann H., Eck M.: Modified multiquadric methods for scattered data interpolation over a sphere, *CAGD* 7, 313–321 (1990). *construction de l'icosaèdre*

Chapitre 4

14. Nielson G.M.: Modeling and Visualization volumetric and surface-on-surface data, in *Focus on Scientific Visualization*, Hagen, Müller, Nielson (eds): Springer, 191–242, (1993).
15. Nielson G.M., Ramaraj R.: Interpolation over a sphere, *CAGD* 4, 41–57, (1987).
16. Nielson G., Ramaraj R.: Interpolation over a sphere based upon a Minimum Norm network, *CAGD* 4, 41–57, (1987). *MNN on sphere*
17. Foley T., Nielson G.M., Lane D., Ramaraj R.: Visualizing functions over a sphere, *IEEE CG& Appl.* 10(1), 32–40, (1990).
18. Foley T., Franke R, hagen H., Nielson G.M.: Interpolation of data on closed surfaces, *CAGD* 7, 303–312, (1990).
19. Pottmann H.: Interpolation on surfaces using minimum norm networks, manuscript.

Chapitre 5

20. Lorensen W.E., Cline H.E.: Marching Cubes: a high-resolution 3D surface construction algorithm, *Computer Graphics*, 21(4), 163–169, (1987).
21. Nielson G.M., Hamann B.: The asymptotic decider: resolving the ambiguity in marching cubes, *IEEE Proc. Vis'91*, 83–91, (1991).
22. Nielson G.M.: Modeling and Visualization volumetric and surface-on-surface data, in *Focus on Scientific Visualization*, Hagen, Müller, Nielson (eds): Springer, 191–242, (1993).
23. Nielson G.M., Hamann B.: Interactive techniques for visualizing volumetric data, *IEEE Proc. Vis'90*, 45–50, (1990).

Chapitre 6

24. Helman J.L., Hesselink L.: Visualizing vector field topology in fluid flows, *IEEE 11 (3)*, 36–46, (1991).
25. Helman J.L., Hesselink L.: Surface representations of two- and three dimensional fluid flow topology, *IEEE Proc. Visualization'90*, (1990).
26. Post F., Walsum T.: Fluid flow visualization, in *Focus on Scientific Visualization*, Hagen, Müller, Nielson (eds): Springer, (1993).
27. Van Wijk J.J., Hin A., de Leeuw W.C., Post F.H.: Three ways to show 3D fluid flow, *IEEE sept.94*, 33–39, (1994).
28. Van Wijk J.J.: Flow visualization with surface particles, *IEEE CG&Appl.*, July'93, 18–24, (1993).
29. Delmarcelle T., Hesselink L.: Visualisation of second order tensor fields and matrix data, *IEEE Proc. Vis'92*, 316–323, (1992).

30. Delmarcelle T., Hesselink L.: The topology of symmetric, second order tensor fields, *IEEE Proc. Vis'94*, 140–147, (1994).
31. Globus A., Levit C., Lasinski T.: A tool for visualizing the topology of three-dimensional vector fields, *IEEE Proc. Vis'91*, 33–40, (1991).
32. Cabral B.: Imaging vector fields using line integral convolution, *SIGGRAPH'93*, 263–270, (1993).
33. Van Wijk J.J.: Spot noise: Texture synthesis for data visualization, *Computer Graphics* 25 (4), 309–318, (1991).
34. Crawfis R., Max N., Becker B.: Vector field visualization, *IEEE CG& Appl. Setp'94*, 50–56, (1994).

Chapitre 7

35. Stollnitz E.J., DeRose T.D., Salesin D.H.: *Wavelets for Computer Graphics, Theory and Applications*, Morgan Kaufmann, (1996).
36. Schröder P., Sweldens W.: Spherical Wavelets: Efficiently Representing Functions on the Sphere, *Siggraph'95*, 161-172, (1995).
37. Mallat S.G.: A Theory for Multiresolution Signal Decomposition: The Wavelet Representation, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11 (7), 674-693, (1989).
38. Gross M.H., Lippert L., Dittrich R., Häring S.: Two Methods for Wavelet-Based Volume Rendering, *Computers & Graphics* 21 (2), 237-252, (1997).
39. de Berg M., Dobrindt K.D.G.: On Levels of Detail in Terrains, Technical report UU-CS-1995-12, Utrecht Univ. (1995).

40. Eck M., DeRose T., Duchamp T., Hoppe H., Lounsbery M., Stuetzle W.: Multiresolution Analysis of Arbitrary Meshes, Siggraph'95, 173-182, (1995).
41. Finkelstein A., Salesin D.H.: Multiresolution Curves, Siggraph 94, 261-268, (1994).
42. Gieng T., Hamann B., Joy K., Schussman G., Trotts I.: Constructing Hierarchies for Triangle Meshes, IEEE Trans. on Visualization and Computer Graphics 4 (2), 145-161, (1998).
43. Gortler S.J., Schröder P., Cohen M.F., Hanrahan P.: Wavelet Radiosity, Siggraph'93, 221-230, (1993).
44. Zhu Z., Machiraju R., Fry B., Moorhead R.: Wavelet-based Multiresolution Representation of Computational Field Simulation Datasets, IEEE Visualization '97, 151-158, (1997).
45. Bonneau, G.-P.: Multiresolution analysis on Irregular Surface Meshes, IEEE Transactions on Visualization and Computer Graphics 4 (4), 365-378, (1998).
46. Bonneau, G.-P.: Multiresolution analysis with non-nested spaces, Computing Suppl. 13, 51-66 (1998).
47. Bonneau G.-P., Gerussi A.: Analysis of Scalar Datasets on Multi-Resolution Geometric Models, in "*Curve and Surface Fitting: Saint-Malo 1999*", Albert Cohen, Christophe Rabut, and Larry L. Schumaker (eds.), Vanderbilt University Press, Nashville, 209–218 (2000).
48. Bonneau, G.-P.: Optimal Triangular Haar Bases for Spherical Data, IEEE Proceedings Visualization'99, 279-284, (1999).
49. Bonneau, G.-P., Gerussi, A.: Level of detail visualization of scalar data sets on irregular surface meshes, IEEE Proceedings Visualization'98, 73-77, (1998).

50. Bonneau, G.-P., Hahmann, St.; Nielson G.M.: BLaC-Wavelets: a multiresolution analysis with non-nested spaces, IEEE Proceedings Visualization'96, 43–48, (1996).