Marseille 04 – 07 Juillet 2017

Parallélisme synchrone et asynchrone en calcul scientifique

Pierre Spiteri

(1) IRIT – ENSEEIHT, UMR CNRS 5505, Toulouse – France

Plénière JDEV 2017

Marseille 04 – 07 Juillet 2017

Parallélisme synchrone et asynchrone en calcul scientifique

Pierre Spiteri

(2) IRIT – ENSEEIHT, UMR CNRS 5505, Toulouse – France

Objectifs visés dans le parallélisme asynchrone

- **enjeu du parallélisme**: diminuer le temps de restitution des calculs et utiliser au mieux les ressources des machines en tenant compte de leurs architectures et ce en fonction du problème à résoudre.

- exemple : diminuer par un facteur deux le temps de restitution des calculs est déjà appréciable.

- **objectifs de l'asynchronisme** : minimiser les contraintes de synchronisation, les délais de communications, la portabilité des applications et la flexibilité des calculs.

- un nombre élevé de synchronisations induit une attente pénalisante des résultats calculés par les autres processeurs ce qui induit une sous utilisation des processeurs en attente.

- recenser dans quelles types de situations le parallélisme asynchrone est plus pertinent à mettre en œuvre que le parallélisme synchrone.

- remédier aux effets de déséquilibrages des charges des processeurs (load balancing).

- **danger** (fort) : possible perte de convergence des méthodes itératives séquentiellement convergentes : **rôle des mathématiques**.

- domaine d'application : mécanique des fluides (météo, océanographie,..) construction mécanique (aéronautique, constructions navale, ferroviaire, automobile …), circuits électronique, chimie, économie & finance, ….. etc ….

Schéma parallèle synchrone de calcul : avec temps d'attente **(hachurés)**

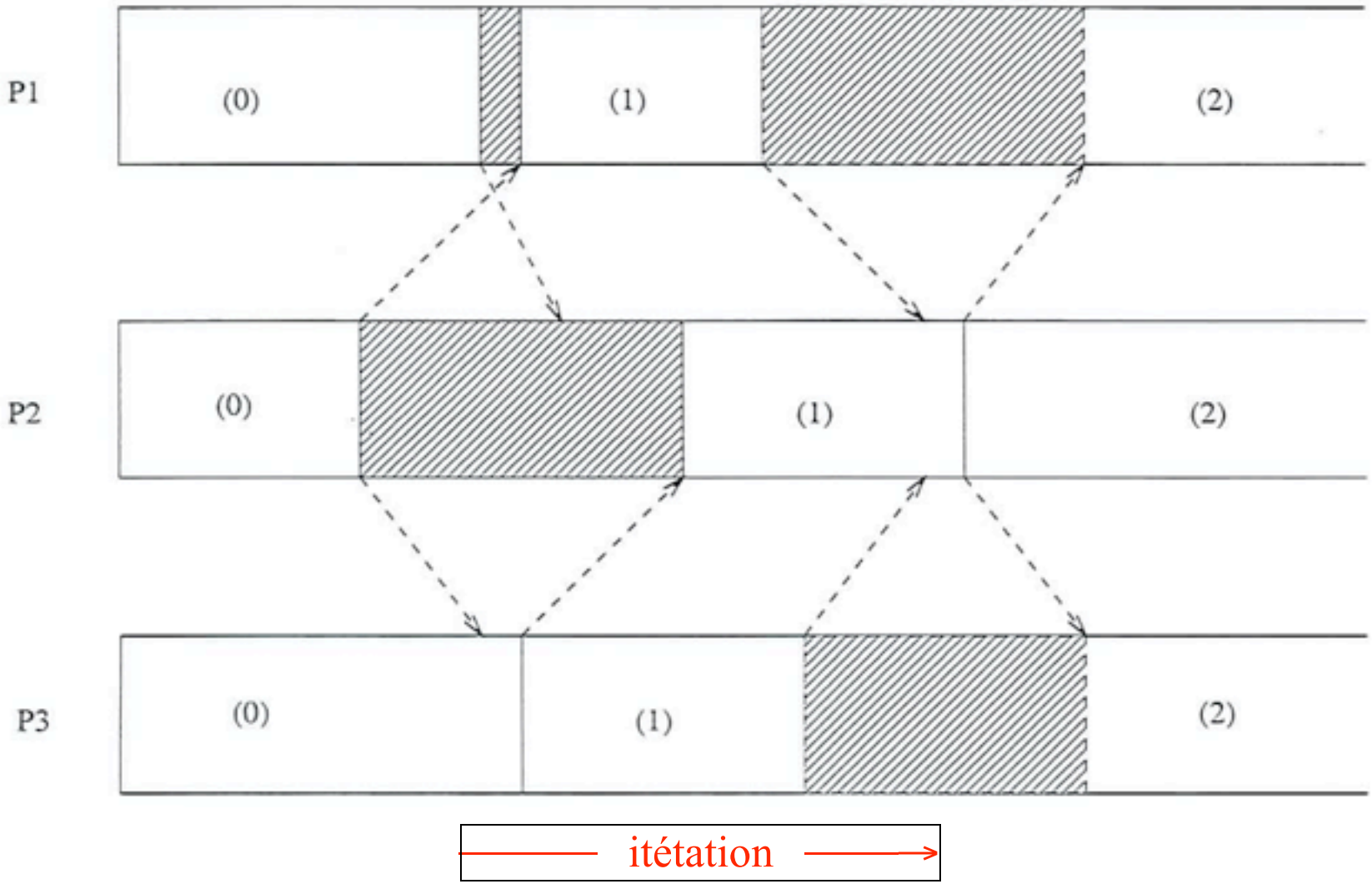


Schéma parallèle asynchrone de calcul : pas de temps d'attente

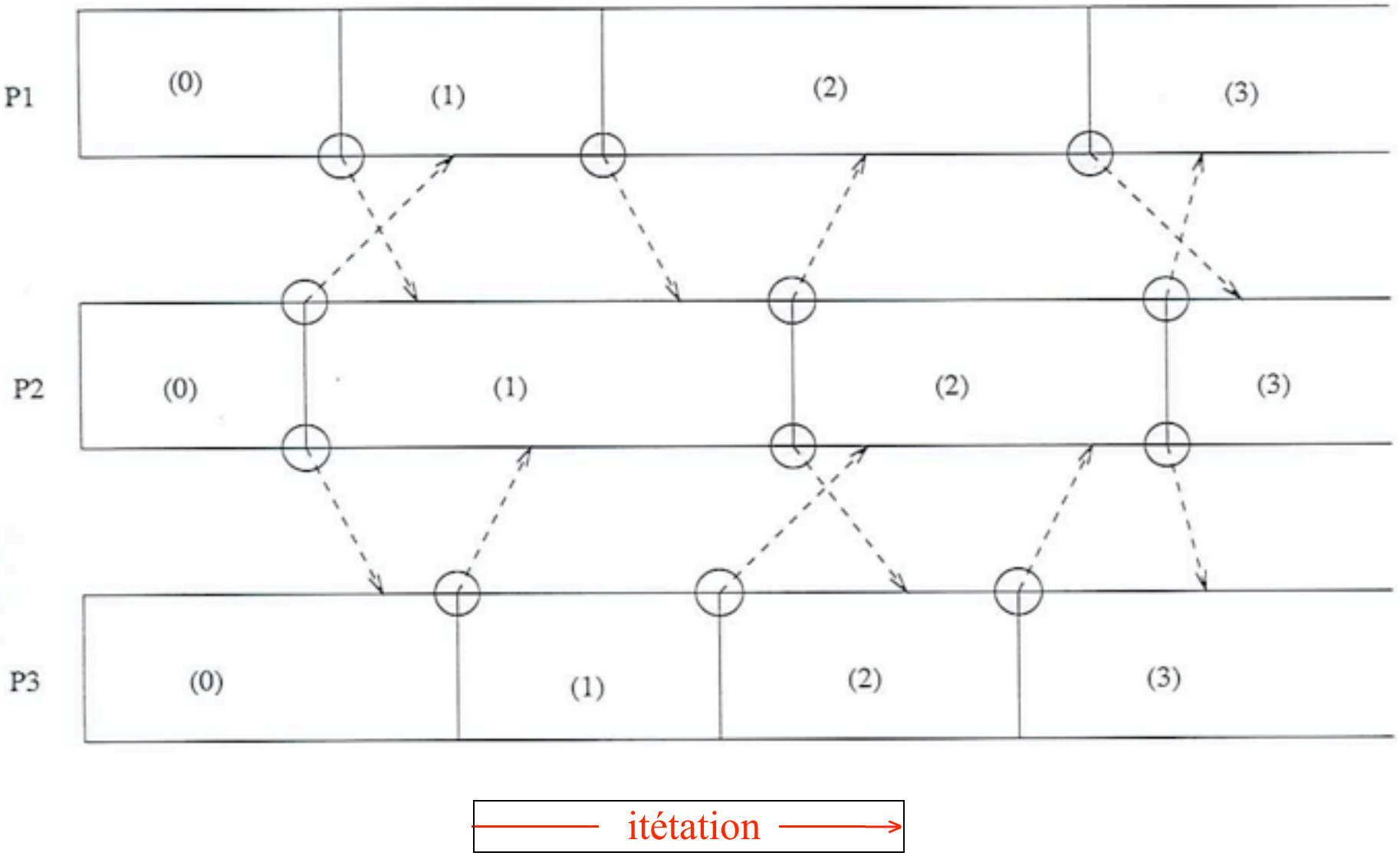
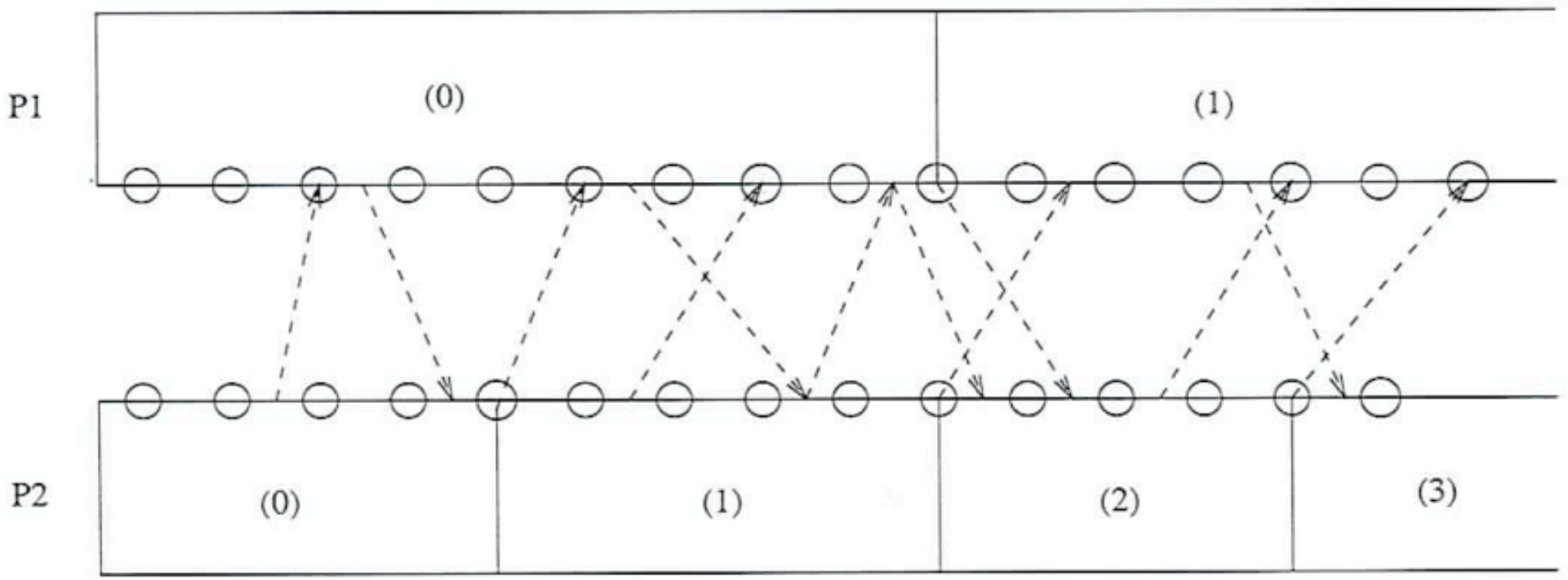


Schéma parallèle asynchrone de calcul avec communications flexibles : pas de temps d'attente



utilisation de mises à jour incomplètes de composantes itérées (relaxées) durant les calculs

itétation

Objectifs visés dans cet exposé

- situer le contexte de la simulation numérique en partant du niveau zéro

- indiquer sommairement les notions basiques de la simulation numérique (discrétisation et résolutions des systèmes algébriques linéaires)

- justifier les choix algorithmiques

- présenter de façon pédagogique les méthodes parallèles synchrones & asynchrones

- in fine présenter la modélisation mathématiques des méthodes parallèles asynchrones

- situer l'applicabilité des méthodes parallèles asynchrones par rapport aux méthodes synchrones

- des précisions viendront en complément dans le Groupe de Travail (T8.GT09) et dans l'Atelier (T8.A10)

1 – Simulation numérique : Applications visées

- électrostatique : problème de Poisson / Laplace
- thermodynamique : équation de la chaleur
- acoustique : équation des ondes
- mécanique des fluides : équation de Navier – Stokes, équation de convection - diffusion
- élasticité – calcul de structure : laplacien généralisé, double laplacien
- traitement d'images : modèles anisotropes (imagerie ultrasons & TEP), équations d'Hamilton – Jacobi – Belmann
- mathématiques financières : laplacien généralisé avec ou sans contraintes
- ……. Etc ……

⇔ Point commun entre tous ces modèles : formulation avec un laplacien dans R^3

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

Problème de convection – diffusion : gradient de u ajouté au laplacien dans R^3

$$\nabla u + \Delta u = \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} + \frac{\partial u}{\partial z} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$$

2 – Principe de résolution numérique de ces problèmes

Deux phases distinctes

- 1) discrétisation des opérateurs : remplacement des dérivées par des quotients différentiels

$$\frac{\partial u}{\partial x} \approx \begin{cases} \dfrac{u(x+\delta x,y,z)-u(x,y,z)}{\delta x}+O(\delta x) \\ \dfrac{u(x+\delta x,y,z)-u(x-\delta x,y,z)}{2.\delta x}+O(\delta x^2) \\ \dfrac{u(x,y,z)-u(x-\delta x,y,z)}{\delta x}+O(\delta x) \end{cases}$$

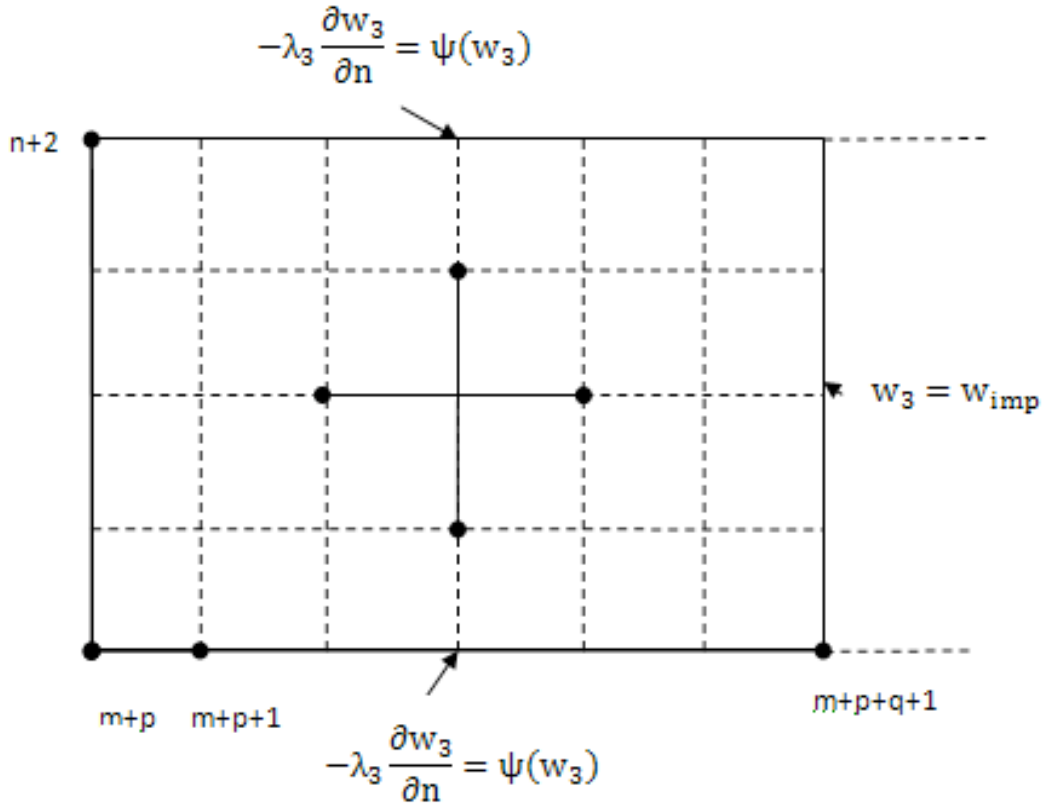
$$\frac{\partial^2 u}{\partial x^2} \approx \frac{1.0}{\delta x}\left(\frac{u(x+\delta x,y,z)-u(x,y,z)}{\delta x}-\frac{u(x,y,z)-u(x-\delta x,y,z)}{\delta x}\right)+O(\delta x^2)$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u(x+\delta x,y,z)-2.u(x,y,z)+u(x-\delta x,y,z)}{\delta x^2}+O(\delta x^2)$$

schémas différences finies

Remarque : même type de schéma par discrétisation volumes finis ou éléments finis

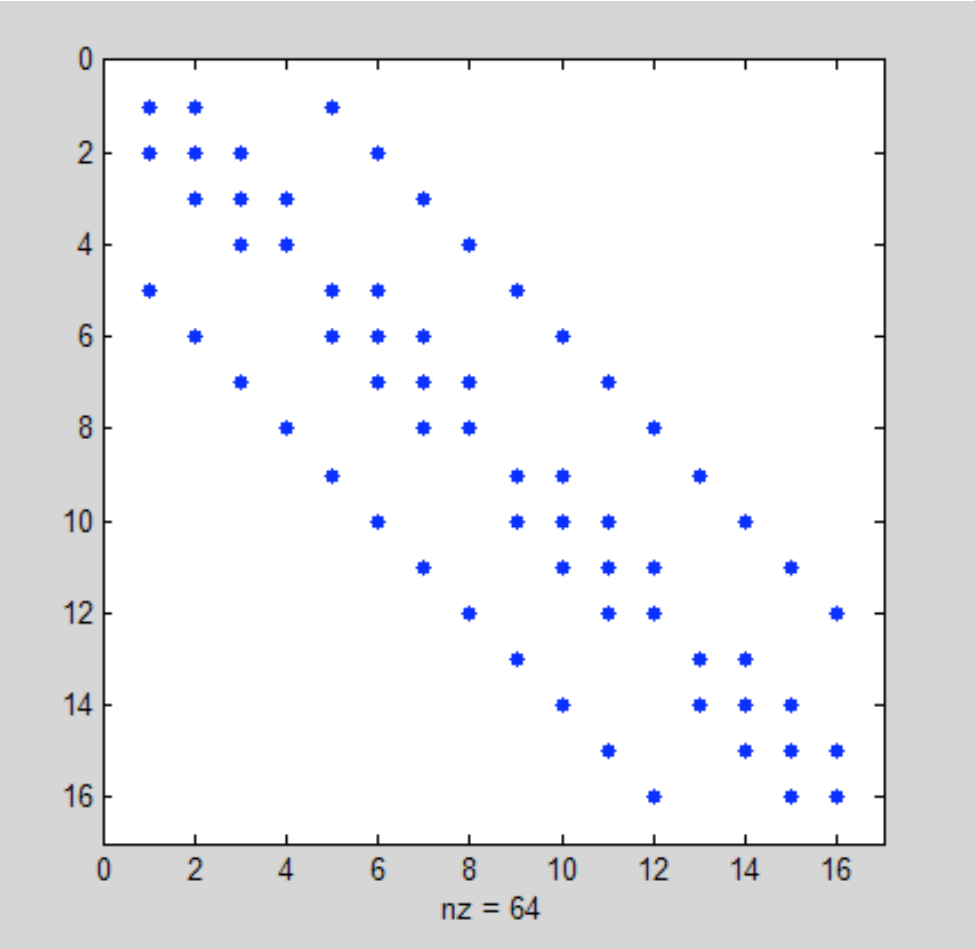
On remplace le domaine de définition de l'opérateur par une **grille de points** (équidistants ou non) et on écrit les schémas de discrétisation précédents en chaque point de grille



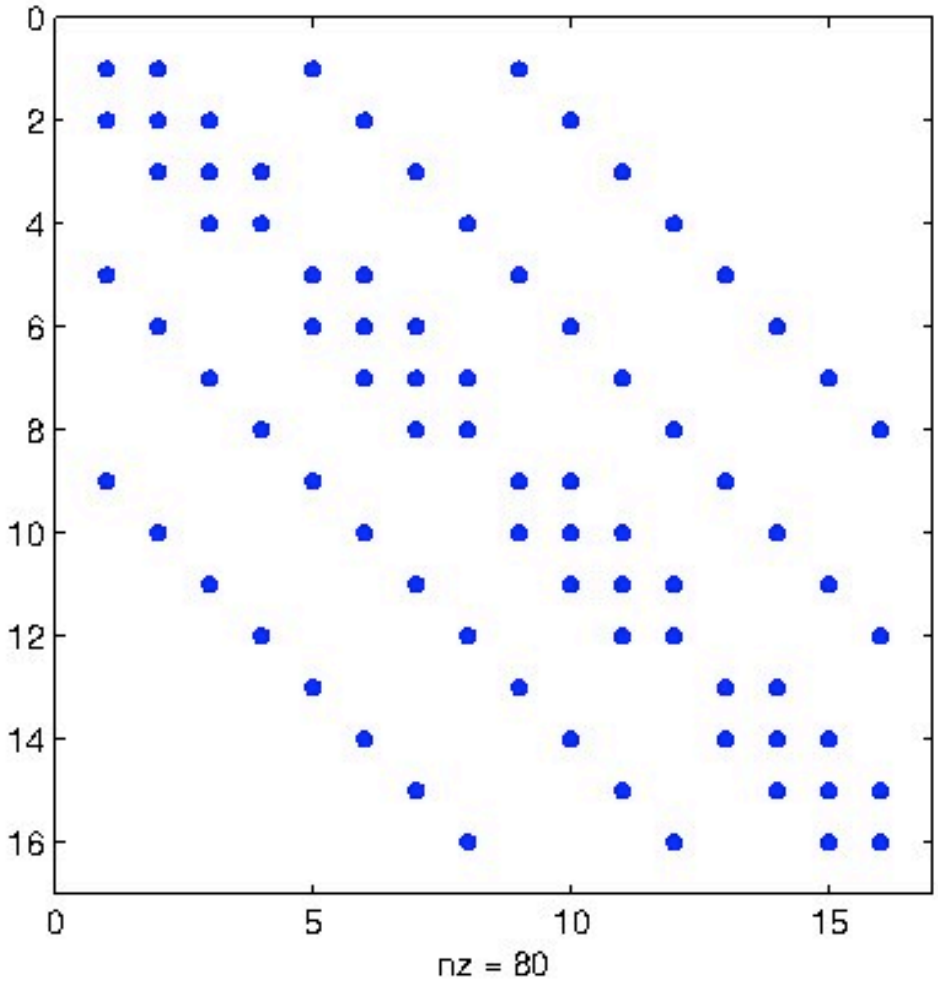
⇒ Obtention d'un **système algébrique linéaire ou pas** suivant le problème

Forme des matrices de discrétisation

En 2D : matrice penta diagonale



En 3D : matrice hepta diagonale



Caractéristiques de ces matrices :

- pour une bonne précision δx, δy et δz sont petits ⇒ **grande dimension** des matrices
  exemple : si 10^3 points de discrétisation sur chaque axe ⇒ taille matrices = 10^9

- les matrices sont **creuses** (énormément de zéros), à structure bande (pour un domaine rectangulaire) ou quasi - bande

- les coefficients **diagonaux** sont **positifs** et **hors diagonaux négatifs**

- **dominance diagonale** (avec schémas de discrétisation adaptés)

$$a_{i,i} \geq \sum_{i \neq j} |a_{i,j}| \quad \text{et même des fois} \quad a_{i,i} > \sum_{i \neq j} |a_{i,j}|$$

EXCELLENTES PROPRIETES !!!!!

Remarque : pour des **problèmes d'évolution** en temps on se ramène par des discrétisations ad hoc à résoudre aussi des systèmes algébriques creux de grande taille à diagonale dominance stricte à **chaque pas de temps**. **ENCORE MIEUX !!**

- 2**) résolution de systèmes algébriques** linéaires ou non linéaires selon le modèle

- si le système algébrique est **non linéaire** ⇒ on se ramène à un séquence de systèmes linéaires par un procédé itératif adéquat (méthode de Newton avec linéarisation locale …..)
⇒ donc dans la **suite** on ne considère que des **systèmes linéaires**

- systèmes linéaires : en gros **deux** grands **types** de méthodes

 les méthodes directes : type méthode d'élimination de Gauss, de Jordan, Choleski, Houssolder :

- en un nombre fini d'opérations arithmétiques on **transforme** la matrice en une matrice diagonale (non recommandée), triangulaire ou triple – diagonale

exemple : méthode de Gauss nécessite $\frac{2}{3}(\dim(A))^3$ opérations arithmétiques si dim(A) = 10^9 ⇒ en gros 10^{27} opérations arithmétiques !!!!

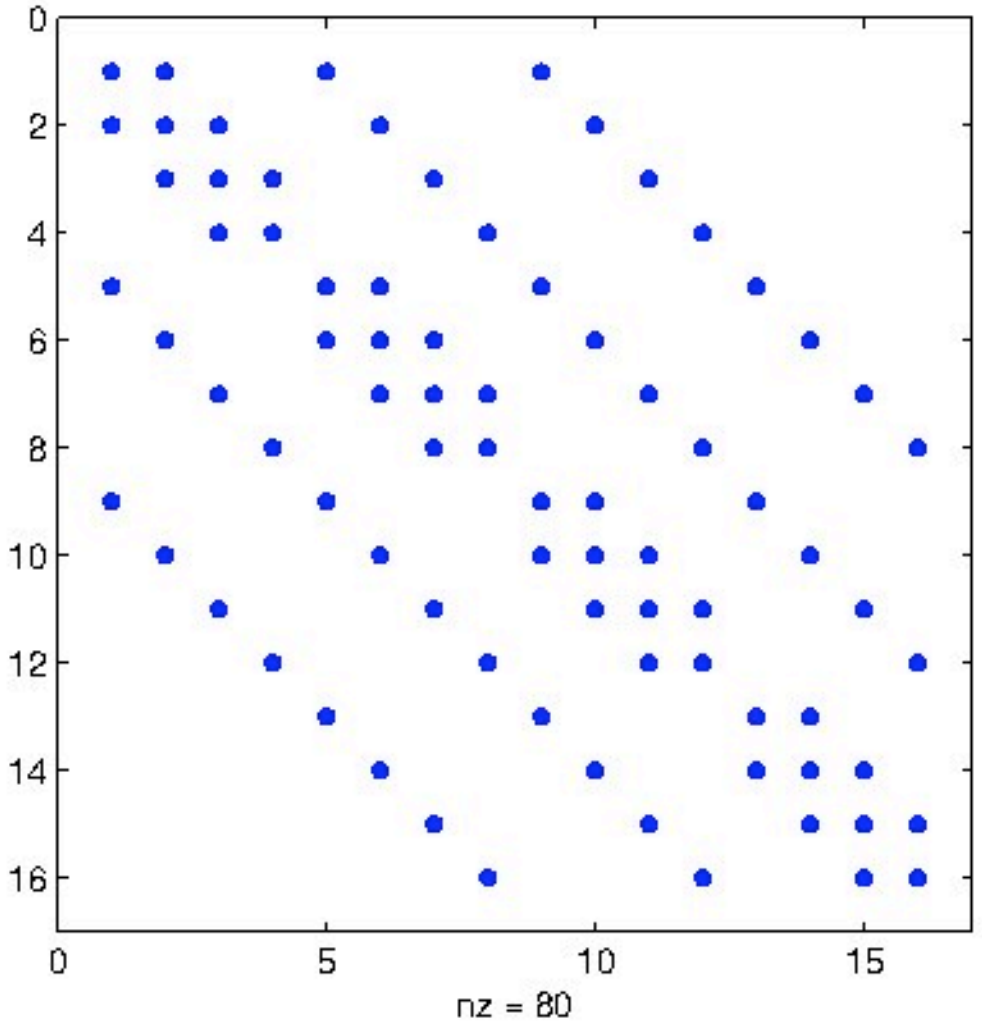
Autre inconvénients :

- phénomène de **fill-in** (remplissage durant la triangularisation), i.e. les « zéros » situés dans la bande sont transformés en valeurs « non nulles »

- s'il y a « pivotage » ⇒ **perte** la **structure bande** (pourtant intéressante)

- mais SURTOUT : matrices **mal conditionnées**, i.e. la propagation d'erreurs de chute peut **dénaturer** les calculs lorsque dim(A) → + ∞ , i.e. pas de discrétisation en espace diminue:

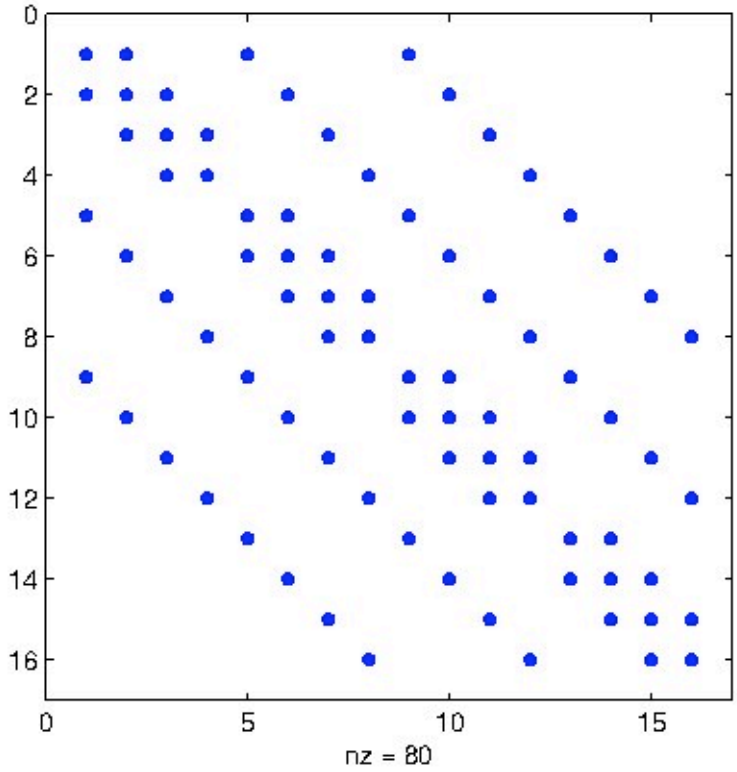
PERTE DE PRECISION !!!



Phénomène de fill – in

 les méthodes itératives : type **méthode de relaxation**, voir méthode gradient conjugué. On se focalise ici surtout sur les méthodes de relaxation

$$-\Delta u \approx \frac{-u(x-h,y,z)-u(x,y-h,z)-u(x,y,z-h)+6u(x,y,z)-u(x,y,z+h)-u(x,y+h,z)-u(x+h,y,z)}{h^2}$$



- Lors de l'utilisation de ces méthodes la matrice n'est pas modifiée

- Compte tenu du caractère creux : peu d'opérations arithmétiques ⇒ influence minimisée de la propagation d'erreur de chute

- Méthode itérative : nécessité d'étudier des critères de convergence (il en existe !!)

On associe au système une équation de point fixe par points

$$\sum_{j=1}^{\dim(A)} a_{i,j}.v_j = b_i \Leftrightarrow \mathrm{v_i} = \frac{b_i - \sum_{j\neq i} a_{i,j}.v_j}{a_{i,i}} = F_i(V),\ \mathrm{i} = 1,..,\dim(\mathrm{A})$$

méthodes de Jacobi et de Gauss – Seidel par points et par blocs

Jacobi

$$\mathrm{v_i^{(p+1)}} = \frac{b_i - \sum_{j\neq i} a_{i,j}.v_j^{(p)}}{a_{i,i}} = F_{i,JACOBI}, i = 1,..,\dim(A)$$

Gauss – Seidel

$$\mathrm{v_i^{(p+1)}} = \frac{b_i - \sum_{j<i} a_{i,j}.v_j^{(p+1)} - \sum_{j>i} a_{i,j}.v_j^{(p)}}{a_{i,i}} = F_{i,G.S.}, i = 1,..,\dim(A)$$

Vocabulaire :

$$\sum_{j=1}^{\dim(A)} a_{i,j}.v_j = b_i \Leftrightarrow \; v_i = \frac{b_i - \sum_{j \neq i} a_{i,j}.v_j}{a_{i,i}} = F_i(V),\; i = 1,..,\dim(A)$$

- on appelle **relaxation** la mise à jour **d'une ou plusieurs** composantes (en mode parallèle plusieurs composantes)

- on appelle **itération** la mise à jour **de toutes** les composantes

- en mode séquentiel et synchrone la notion d'itération garde son sens.

- mais en mode asynchrone on introduit la notion de **macro – itération** qui correspond au fait que l'on a relaxé au moins une fois chaque composante.

Méthodes par blocs : Prise en compte de la structure bloc de la matrice et **fort lien avec le parallélisme**

$$A_{i,i}V_i + \sum_{j\neq i}^{\alpha} A_{i,j}V_j = B_i,\ \mathrm{i}=1,..,\alpha \ \Leftrightarrow\ \mathrm{V_i} = A_{i,i}^{-1}(B_i - \sum_{j\neq i}^{\alpha} A_{i,j}V_j)$$

α nombre de blocs

méthode de Jacobi par blocs

$$\mathrm{V_i^{(p+1)}} = A_{i,i}^{-1}(B_i - \sum_{j\neq i}^{\alpha} A_{i,j}V_j^{(p)}) = F_{i,BJ}(V), \mathrm{i}=1,..,\alpha$$

méthode de Gauss – Seidel par blocs

$$\mathrm{V_i^{(p+1)}} = A_{i,i}^{-1}(B_i - \sum_{j=1}^{i-1} A_{i,j}V_j^{(p+1)} - \sum_{j=i+1}^{\alpha} A_{i,j}V_j^{(p)}) = F_{i,GS}(V), \mathrm{i}=1,..,\alpha$$

Remarque : on n'inverse pas les matrices blocs diagonales $A_{i,i}$; on résout plutôt les sous – systèmes linéaires $A_{i,i}.V_i = G_i$ en utilisant des algorithmes efficaces (par exemple gradient conjugué et ses variantes ou bien des méthodes de relaxation ou encore des méthodes directes lorsque les sous – matrices sont creuses (triple diagonales – méthode TDMA))

Résolution parallèle asybchrone du système algébrique linéaire

Principe

- Système(s) de grande dimension ⇒ Méthodes itératives parallèles **recommandées**
- **Décomposition du problème en α sous-problème**
- En calcul **parallèle asynchrone** les processeurs utilisent les valeurs des composantes **disponibles** du vecteur itéré calculées par les autres processeurs, i.e. les dernières valeurs calculées (principe de Gauss)
- Chaque processeur **va** à son **propre rythme sans synchronisation** avec les autres processeurs ni équilibrage de charges des processeurs
- **Intérêt** : à cause des non synchronisations, **réduction** du **temps** de **restitution** des calculs lorsque de **nombreux processeurs** sont **utilisés**

  ⇒ expérimentations avec 128 processeurs: gain de 33% du temps de restitution en asynchrone par rapport au synchrone sur clusteur avec réseau rapide

– Méthode de sous-domaines asynchrones sans recouvrement : **répartition des calculs par processeur**

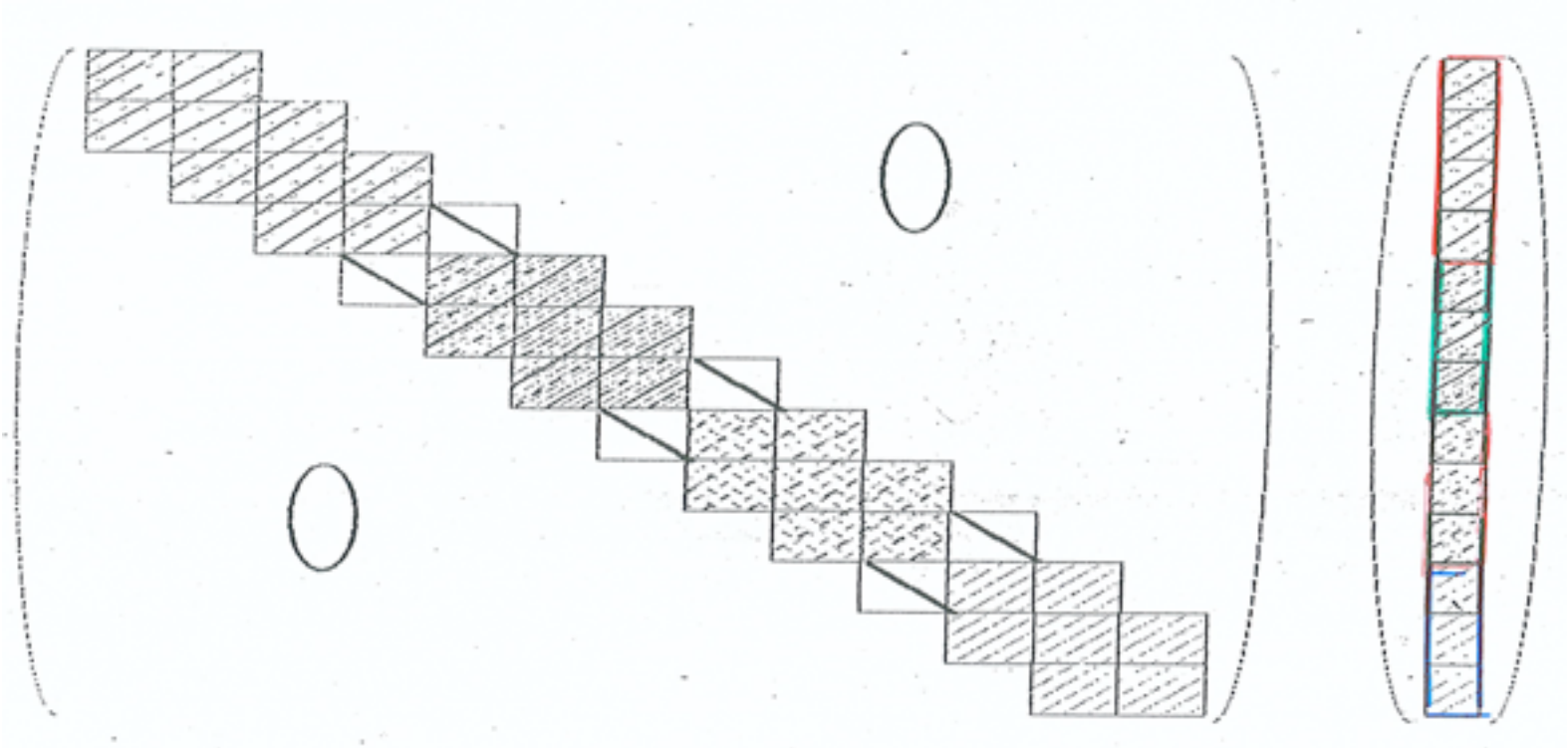
$$\Omega_i \cap \Omega_j = \varnothing, \quad i \neq j$$

décomposition en deux sous – domaines Ω_1 et Ω_2

Regroupement de blocs adjacents de la matrice pour définir un sous-domaine

Les sous-domaines peuvent être numérotés de façon lexicographique ou rouge - noire

Chaque processeur va donc effectuer les calculs nécessaires sur chaque sous – domaine Ω_I , i=1,2



A la décomposition par sous – domaine est associée naturellement une décomposition par GRANDS blocs de la matrice

<u>Exemple</u> de calcul réparti sur 4 processeurs: 4 processeurs ⇔ 4 sous domaines

Premier sous-domaine → **rouge**

Second sous-domaine → **vert** …

Dernier sous-domaine → **bleu ….** Etc ….

Schéma parallèle synchrone de calcul : avec temps d'attente **hachure en noir**

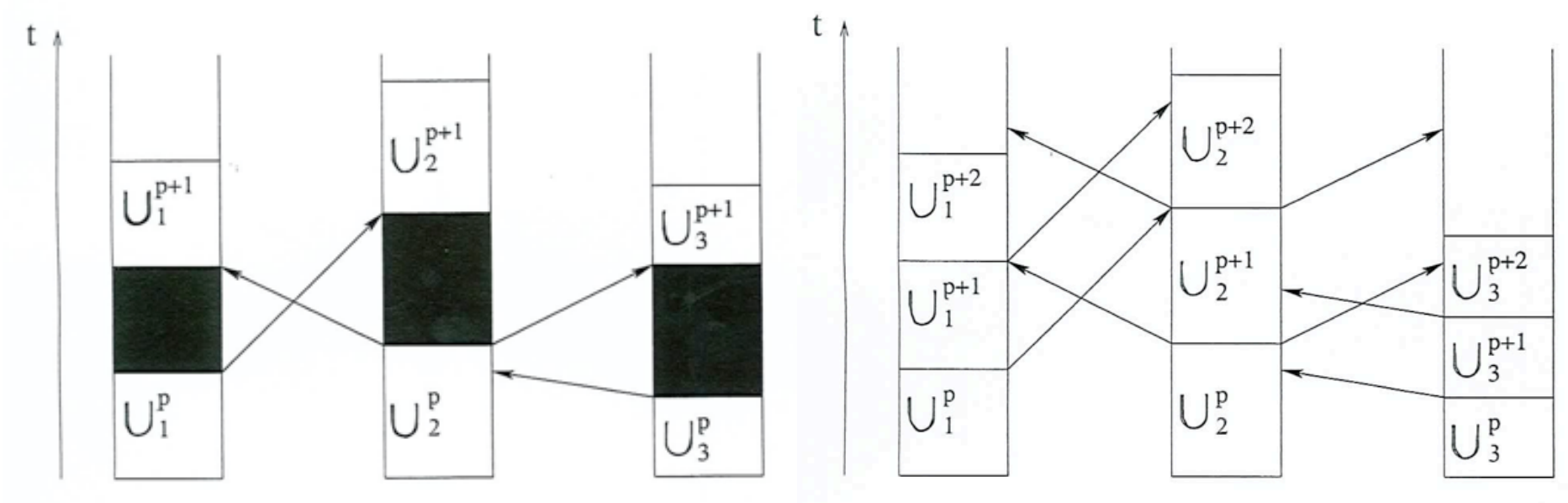


Schéma parallèle asynchrone de calcul (à gauche): pas de temps d'attente

– Méthode de sous-domaines asynchrones avec recouvrement : méthode alternée de Schwarz

⇒ Méthode alternée de Schwarz parallèle asynchrone

Cas 1D séquentiel

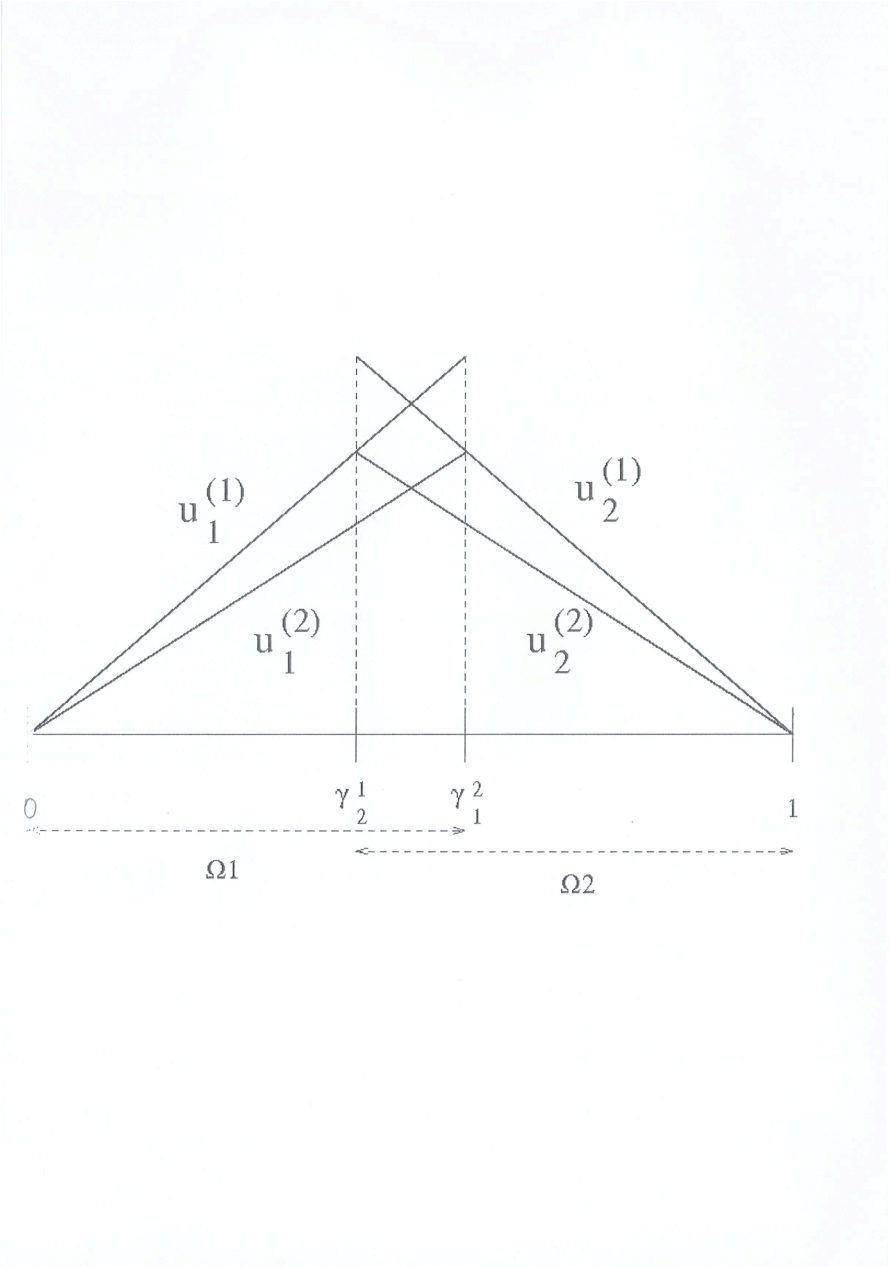
$$\begin{cases} -\dfrac{d^2u(x)}{dx^2} = f(x),\ 0 < x < 1, \\ u(0) = u(1) = 0 \end{cases}$$

$\Omega = \Omega_1 \cup \Omega_2$ et $\Omega_1 \cap \Omega_2 \neq \Phi$ avec $\Omega_1 = [0, \gamma_1^2]$ et $\Omega_2 = [\gamma_2^1, 1]$, $\gamma_2^1 < \gamma_1^2$

les deux sous domaines Ω_1 et Ω_2 se recouvrent

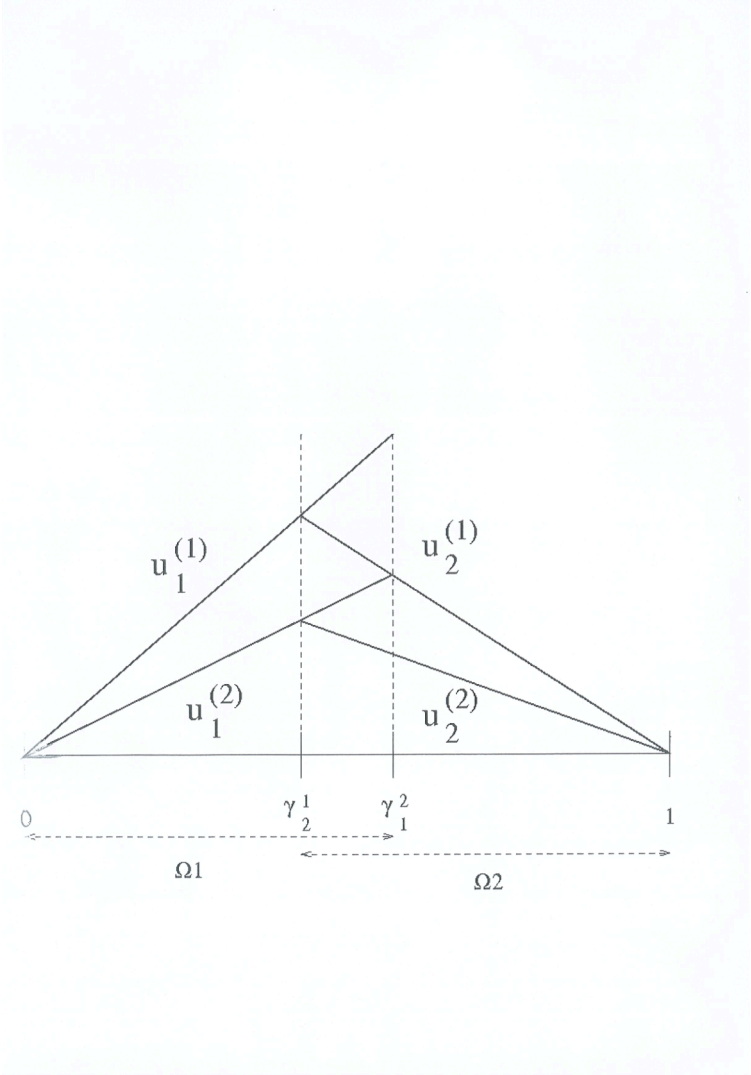
$$\begin{cases} -\dfrac{d^2u_1(x)}{dx^2} = f_1(x),\ 0 < x < \gamma_1^2, \\ u_1(0) = 0 \\ u_1(\gamma_1^2) = u_2(\gamma_1^2) \end{cases} \quad \text{et} \quad \begin{cases} -\dfrac{d^2u_2(x)}{dx^2} = f_2(x),\ \gamma_2^1 < x < 1, \\ u_2(1) = 0 \\ u_2(\gamma_2^1) = u_1(\gamma_2^1) \end{cases}$$

⇒ plusieurs stratégies d'échanges des valeurs frontières : Jacobi (Schwarz additif), Gauss – Seidel (Schwarz multiplicatif), Schwarz synchrone, Schwarz asynchrone avec ou sans communication flexible



Schwartz additif (Jacobi)

Schwartz multiplicatif (Gauss Seidel)



Cas 2D séquentiel

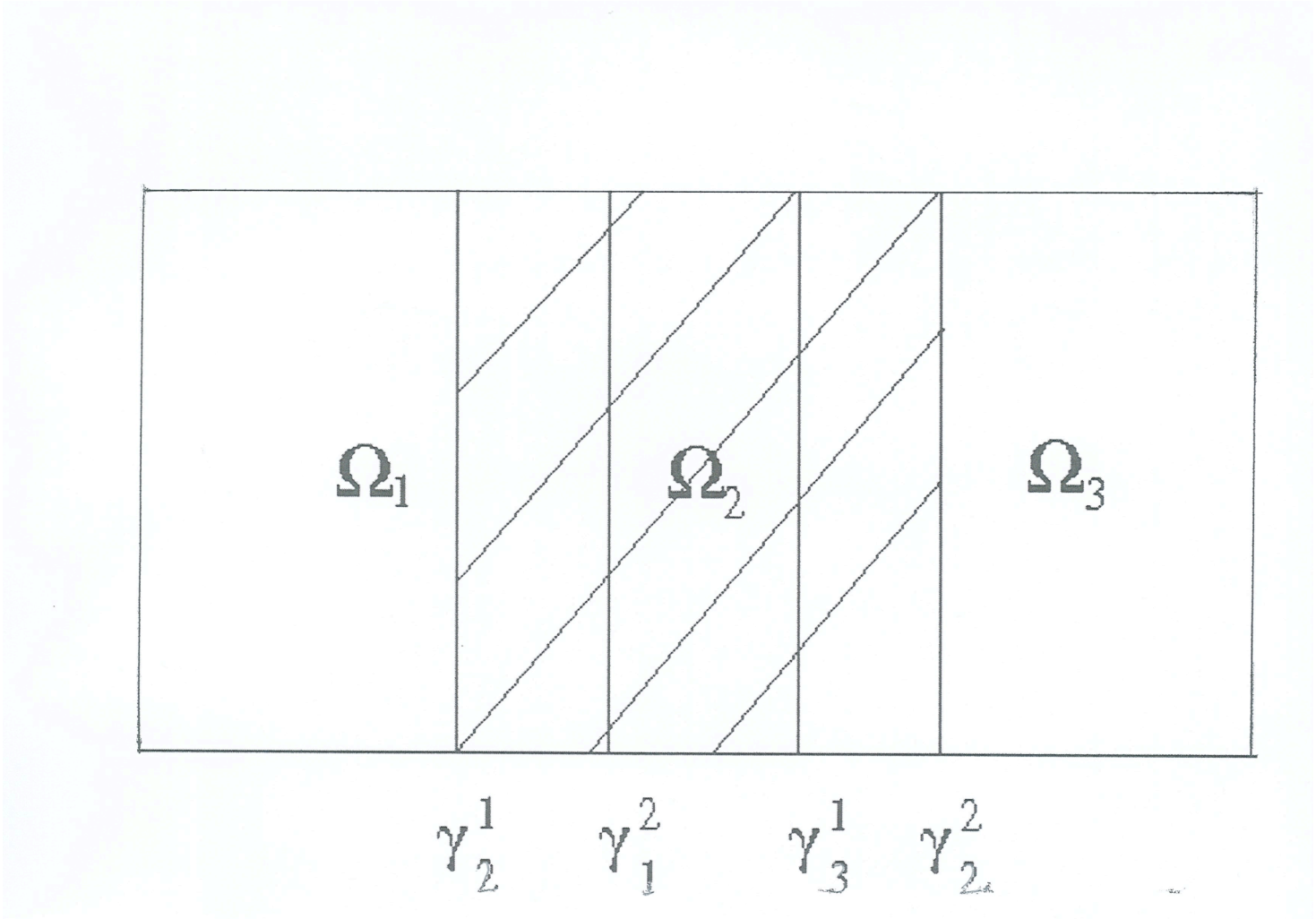
$$\begin{cases} -\Delta u = f, \text{ sur } \Omega \\ \mathrm{u}_{\partial\Omega} = 0 \end{cases}$$

$$\begin{cases} -\Delta u_i = f_i, \text{ sur } \Omega_i, \ \mathrm{i} = 1,2,.... \\ \mathrm{u}_i / \partial\Omega_i = 0 \\ \mathrm{u}_i / \gamma_i^j = \mathrm{u}_j / \gamma_i^j, \ \forall \mathrm{j} \in \mathrm{adj(i)}, \ \mathrm{j} = \{1,2\}, \ \mathrm{i} = 1,2,.... \end{cases}$$

γ_i^2 frontière de recouvrement à droite, γ_i^1 frontière de recouvrement à gauche

les nouvelles conditions aux limites pour Ωi sont données par les valeurs obtenues sur les frontières de recouvrement et calculées dans le sous domaine voisin

Plus le recouvrement est grand et plus la méthode converge vite ….. mais on calcule plusieurs fois les valeurs communes à plusieurs sous-domaines


Ω_1
Ω_2
Ω_3
γ_2^1
γ_1^2
γ_3^1
γ_2^2

Problème à résoudre : Déterminer $V \in R^n$ t. q. $AV=B$

Equation de point fixe associée : $V = F(V),\ F : R^n \rightarrow R^n$

Modélisation des itérations parallèles asynchrones

parallélisation nécessite la décomposition du problème en sous - problèmes interconnectés

$$E = \prod_{i=1}^{\alpha} E_i,\ E \equiv R^n \text{ (E espace produit - } \alpha \text{ nombre processeurs)}$$

$$F(V) = \{...., F_i(V), ...\},\ V \in E$$

Itérations parallèles asynchrones définies par

$$V_i^{(p+1)} = \begin{cases} V_i^{(p)} \text{ si } i \notin J(p) \\ F_i(..., V_j^{(s_j(p))}, ...) \text{ si } i \in J(p) \end{cases}$$

$\mathbf{J} = \{J(p)\}$, $p \in N$, séquence de sous ensembles non vides de $\{1,\ldots, \alpha\}$

$\Rightarrow$ **J** modélise le **parallélisme**

$\mathbf{S} = \{s_1(p),\ldots, s_\alpha(p)\}$, séquence d'éléments of N^α

$\Rightarrow$ **S** modélise l'**asynchronisme** entre les processeurs

Où J et S satisfont

$$\forall\, p \in N\ ,\ J(p) \neq \varnothing,\ J(p) \subset \{1,\ldots,\alpha\}$$

$\Leftrightarrow$ séquence de sous ensembles non vides et modélisation du parallélisme

$$\forall\, i \in \{1,\ldots,\alpha\},\ \mathrm{Card}(\{p \in N \mid i \in J(p)\}) = +\infty$$

$\Leftrightarrow$ on doit « théoriquement » relaxer une infinité de fois sur chaque composante

$$\forall\, j \in \{1,\ldots,\alpha\},\ \forall\, p \in N\ ,\ 0 \leq s_j(p) \leq p,$$

et $s_i(p) = p$ si $i \in J(p)$ (pas de retards locaux)

$\Leftrightarrow$ les nombres $s_j(p)$ sont isomorphes à des numéros de relaxation bornés par p

$$\forall\, j \in \{1,\ldots,\alpha\},\ \lim_{p \to \infty}(s_j(p)) = +\infty$$

$\Leftrightarrow$ normal qu'un numéro de relaxation puisse devenir infini dans une méthode itérative

Remarque : les retards $r_i(p)$ entre les processeurs tels que $s_i(p)=p - r_i(p)$ peuvent devenir infinis (situation de panne)

Cas particuliers : $s(p) \equiv p$, $\forall\, p \in N$ $\Leftrightarrow$ algorithme **parallèle synchrones**

- si de plus $J(p) = \{1,\ldots,\alpha\}$ et $s(p) \equiv p$, $\forall\, p \in N$, méthode de **Jacobi séquentielle**
- si de plus $J(p) = p.\mathrm{mod}(\alpha) + 1$ et $s(p) \equiv p$, $\forall\, p \in N$, méthode de **Gauss – Seidel séquentielle**

Analyse de la convergence : d'où la nécessité de la modélisation des méthodes parallèles asynchrones

- par technique de contraction

$$\|F(Y) - F(X)\| \leq \nu \|Y - X\|, \ \forall Y$$

vitesse de convergence peur être estimée par la valeur de la constante de contraction ν

(**ν petit ⇒ convergence rapide** & **ν proche de 1 ⇒ convergence lente**)

- par technique d'ordre partiel

$$Z^{(0)} \leq Z^{(1)} \leq .. \leq Z^{(q)} \leq .. \leq X \leq .. \leq Y^{(p)} \leq .. \leq Y^{(1)} \leq Y^{(0)}$$

- par appartenance à des ensembles emboîtés centrés en la solution

$$E^{(\infty)} \subseteq ..... \subseteq E^{(p)} \subseteq .... \subseteq E^{(0)}$$

MAIS CONDITION SUFFISANTE DE CONVERGENCE ASSUREE SI

- les coefficients **diagonaux** sont **positifs** et **hors diagonaux négatifs**
- **dominance diagonale** (avec schémas de discrétisation adaptés)

$$a_{i,i} \geq \sum_{i \neq j} |a_{i,j}| \quad \text{et même des fois} \quad a_{i,i} > \sum_{i \neq j} |a_{i,j}|$$

EXCELLENTES PROPRIETES !!!!!

Tests d'arrêt des itérations : **pas simple car pas de synchronisation !!**

Principe : sur chaque processus évaluation d'un critère de convergence local et la convergence globale est atteinte lorsque il y a convergence sur tous les processus

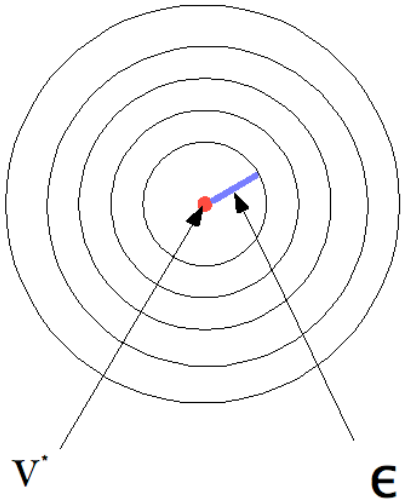
- soit **centralisé** contrôlé par un processeur maître qui arrêtent l'activité de tous les processus lorsqu'ils ont tous convergé

- soit **décentralisé** avec l'utilisation d'un jeton qui est un message spécifique qui ne contient pas les valeurs calculées localement ; ce jeton est émis initialement par n'importe quel processus de façon non bloquante puis retransmis à un autre processus qui lui même le retransmettra) à un autre processus ….. etc ……. . Ce message indique si tous les processus ont localement convergé (utilisation d'un booléen). Le processus qui a émis le jeton a la charge d'envoyer un message d'arrêt aux autres processus.

Remarque : les calculs ne sont jamais arrêtés sur un processeur, même sil y a convergence locale

Remarque : mathématiquement la propriété d'ensembles emboîtés permet d'exhiber des tests d'arrêts des itérations par **estimation** des diamètres des ensembles emboités

Remarque : on s'est restreint à des problèmes linéaires mais ce qui précède est aussi valable pour certains problèmes non linéaires


V*
ϵ

Conclusion partielle

IBM avait lancé un chalenge de machines avec 100 000 processeurs en 2010. Pari difficilement tenu jusqu'à présent. L'asynchronisme est peut être une voie possible de solution.

Domaines d'application : chaque fois que l'on a un **problème** de **point fixe** de **grande dimension** à résoudre

- Résolution de grands systèmes algébriques linéaires et non linéaires
- Résolution d'équations aux dérivées partiellers
- Résolution d'équations algébro différentielles
- Optimisation (problème de transport)
- Dissimulation d'informations, sureté informatique, tatouage (itérations chaotiques)
- Chaînes de Markov

Etc etc ……

Itérations asynchrones étudiées dans de nombreux centres

Continent	Centre - Université
U.S.A.	IBM Centre Watson
	MIT Harvard, Brown university
	Southern California University
	Pennsylvania University
	Temple University
	Washington University
	Livermore National Laboratory
	Maryland University
	North Carolina University
	Université du Texas - Arlington
	Université du Texas - Austin
	Université de Mayaguez – Porto Rico
	Université de Connecticut
Brésil	Rio de Janeiro University
Asie - Chine	Wuhan University
	Fudan University
	Académie des Sciences Beyung
	Universté de Taïguan
	Université de Shanxi
	Université de Shangai

Continent	Centre - Université
Asie - Corée	**Chungbuk University**
Asie – Hong Kong	**Hong Kong University**
Asie - Japon	**Université de Kyoto**
Asie – Inde	**Bangalore University**
Europe – United Kingdown	**Loughborough University**
	Edimbourgh University
Europe - Allemagne	**Wuppertal University**
	Karlshrue University
	Berlin University
	Université de Potsdam
Europe - Espagne	**Valence University**
	Alicante University
Europe – Russie	**Académie des Sciences - Moscou**
Europe - Norvège	**Bergen University**
Europe - Italie	**Université de Modène**
Europe - Grèce	**Université de Paras**
Europe - France	**Université de Franche Comté**
	Université Belfort Montbeliard
	Université de Grenoble
	Université de Lyon

Continent	Centre - Université
	Université de Lille
	Université de Toulouse
Moyen orient - Iran	**Université de Rasht**
Etc etc	**Maroc, Algérie, Koweït …..**

Merci de votre attention