Formalisation et automatisation de YAO, générateur de code pour l'assimilation variationnelle de données

Luigi NARDI

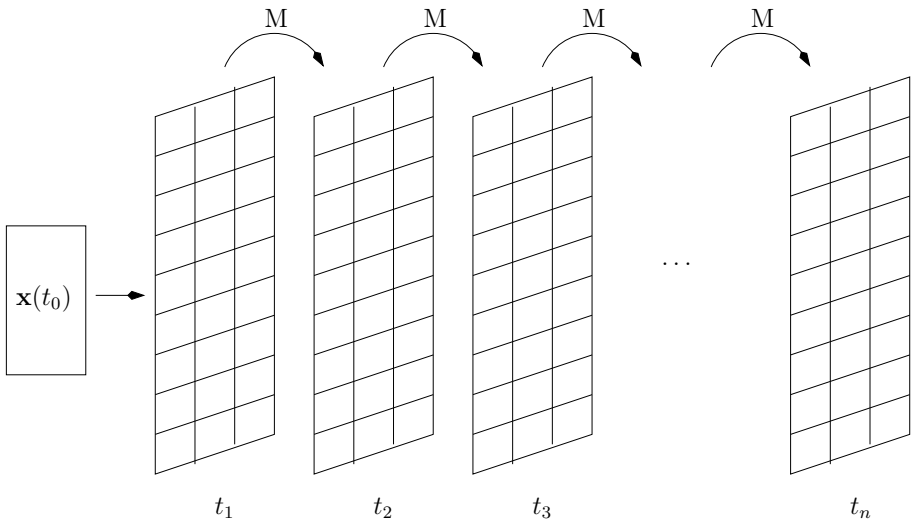
sous la direction de Fouad BADRAN et Sylvie THIRIA

LOCEAN : Laboratoire d'Océanographie et du Climat Expérimentation et Approches Numériques

CEDRIC : Centre d'Etude et De Recherche en Informatique du CNAM

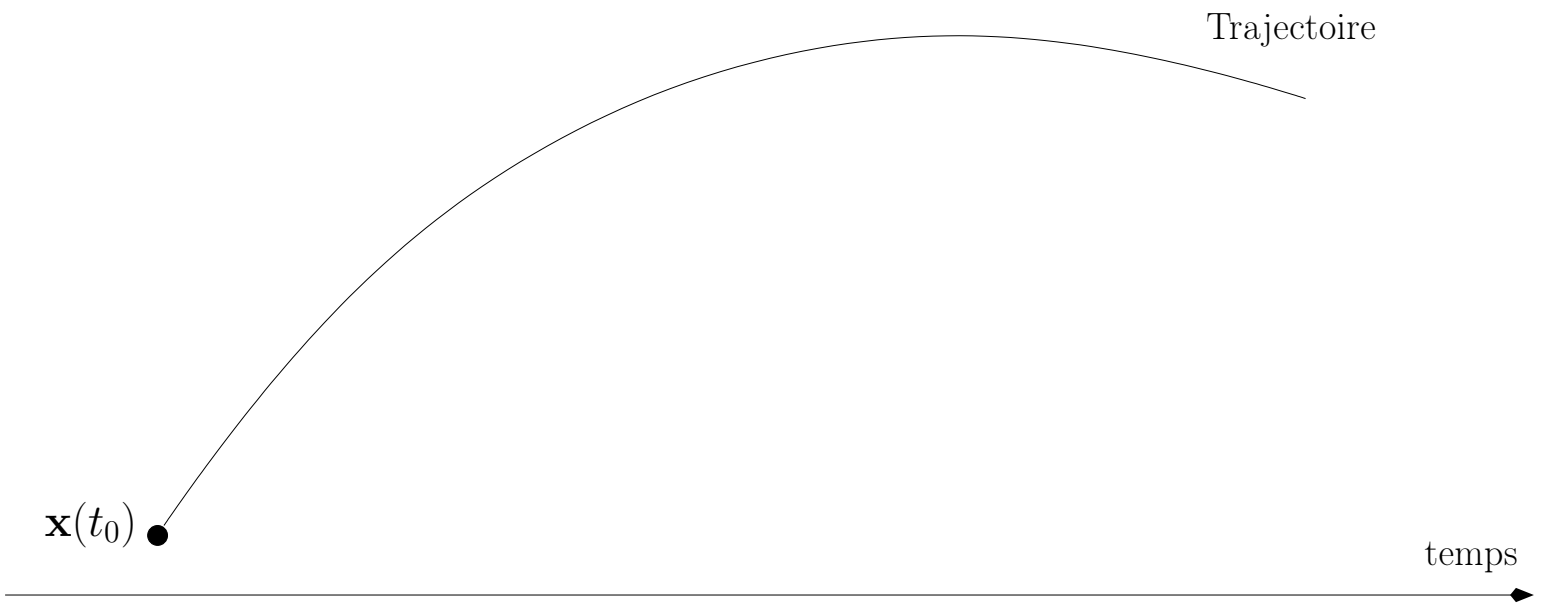
Modèle numérique direct

- M : modèle direct entre deux pas de temps t_i, t_{i+1} ;
- $\mathbf{x}(t_0)$: paramètres de contrôle ;
- $M_i(\mathbf{x}(t_0))$: état au temps t_i à partir de l'état à t_0.



Objectif : prédire ou analyser l'évolution d'un phénomène mais . . . les modèles ne sont pas parfaits !

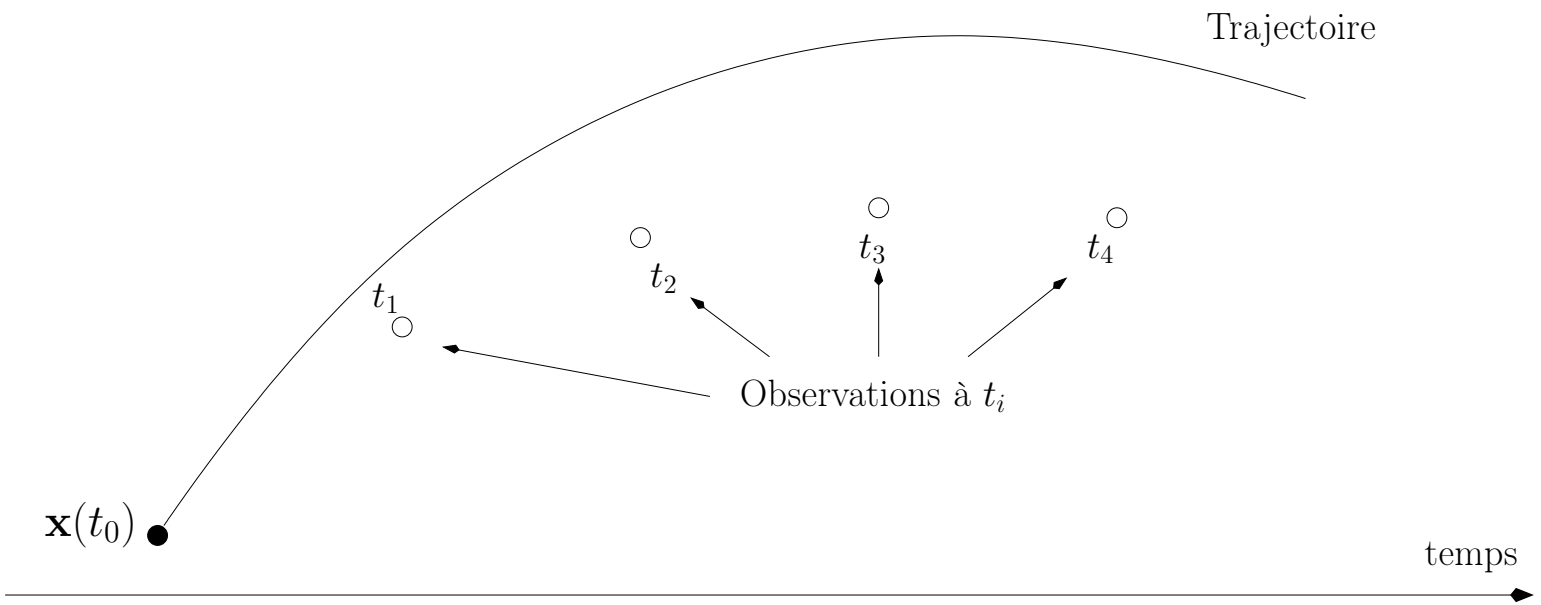
Assimilation variationnnelle de données 4D-Var



Fonction de coût à minimiser : $J(\mathbf{x}(t_0))$

Gradient : $\nabla_{\mathbf{x}(t_0)} J$

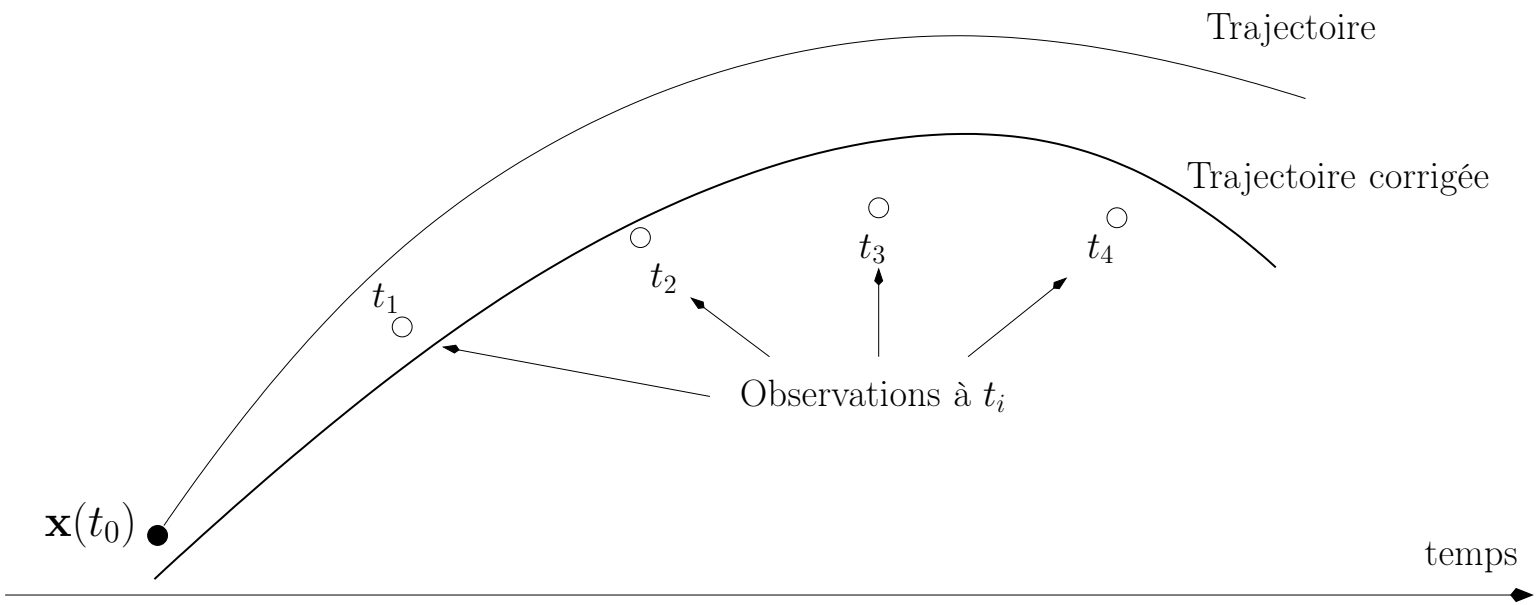
Assimilation variationnnelle de données 4D-Var



Fonction de coût à minimiser : $J(\mathbf{x}(t_0))$

Gradient : $\nabla_{\mathbf{x}(t_0)} J$

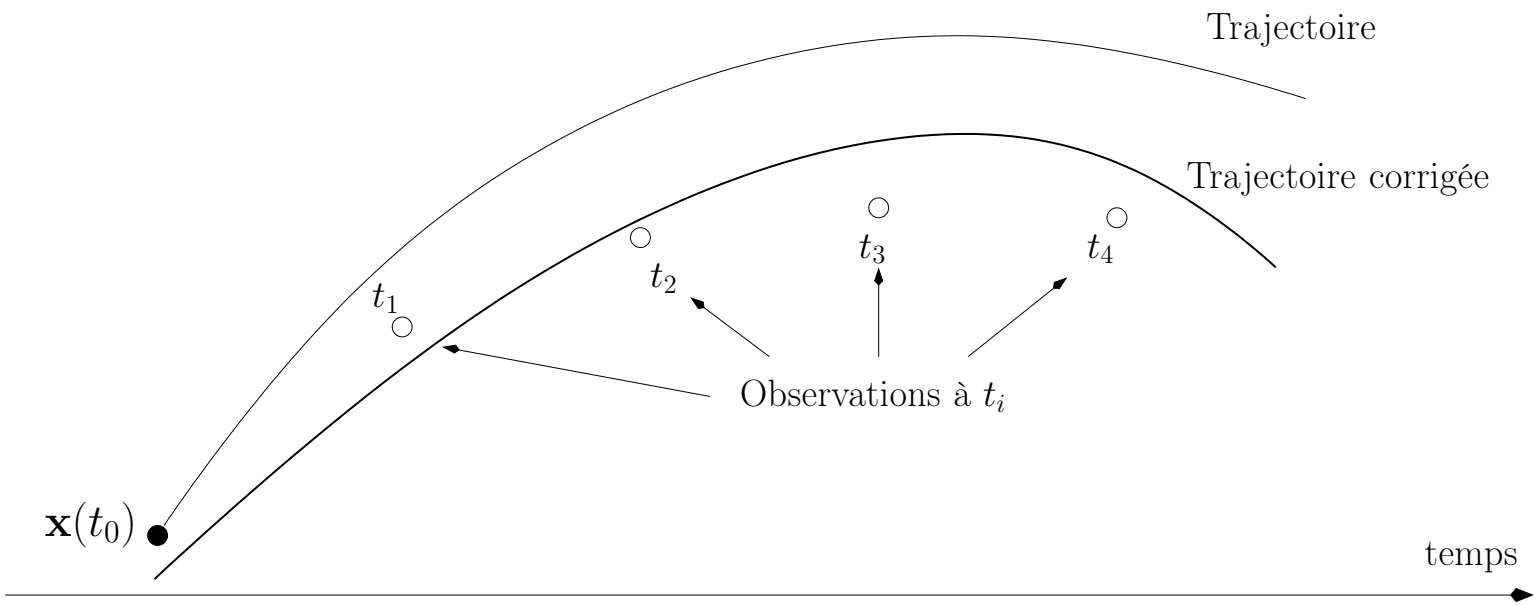
Assimilation variationnnelle de données 4D-Var



Fonction de coût à minimiser : $J(\mathbf{x}(t_0))$

Gradient : $\nabla_{\mathbf{x}(t_0)} J$

Assimilation variationnnelle de données 4D-Var



Fonction de coût à minimiser : $J(\mathbf{x}(t_0))$

Gradient : $\nabla_{\mathbf{x}(t_0)} J$

Formalisme de l'assimilation variationnelle de données

Fonction de coût J :

$$J(\mathbf{x}(t_0)) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o) + \frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b)^T \mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b).$$

- $\mathbf{y}_i$: sorties du modèle ;
- $\mathbf{y}_i^o$: observations au temps t_i ;
- $\mathbf{R}_i$ et $\mathbf{B}$: matrices de variance-covariance ;
- $\mathbf{x}^b$: vecteur d'ébauche.

$$\nabla_{\mathbf{x}(t_0)} J = \sum_{i=1}^{n} \underbrace{\mathbf{M}_i^T(\mathbf{x}(t_0))}_{adjoint} \mathbf{H}_i^T [\mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o)] + \mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b).$$

- H_i : opérateur d'observation ;
- $\mathbf{M}_i^T(\mathbf{x}(t_0))$: modèle adjoint.

Formalisme de l'assimilation variationnelle de données

Fonction de coût J :

$$J(\mathbf{x}(t_0)) = \underbrace{\frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o)}_{\textit{attache aux données}} + \frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b)^T \mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b).$$

- $\mathbf{y}_i$: sorties du modèle ;
- $\mathbf{y}_i^o$: observations au temps t_i ;
- $\mathbf{R}_i$ et $\mathbf{B}$: matrices de variance-covariance ;
- $\mathbf{x}^b$: vecteur d'ébauche.

$$\nabla_{\mathbf{x}(t_0)} J = \sum_{i=1}^{n} \underbrace{\mathbf{M}_i^T(\mathbf{x}(t_0))}_{\textit{adjoint}} \mathbf{H}_i^T [\mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o)] + \mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b).$$

- H_i : opérateur d'observation ;
- $\mathbf{M}_i^T(\mathbf{x}(t_0))$: modèle adjoint.

Formalisme de l'assimilation variationnelle de données

Fonction de coût J :

$$J(\mathbf{x}(t_0)) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o) + \underbrace{\frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b)^T \mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b)}_{\textit{régularisation}}.$$

- $\mathbf{y}_i$: sorties du modèle ;
- $\mathbf{y}_i^o$: observations au temps t_i ;
- $\mathbf{R}_i$ et $\mathbf{B}$: matrices de variance-covariance ;
- $\mathbf{x}^b$: vecteur d'ébauche.

$$\nabla_{\mathbf{x}(t_0)} J = \sum_{i=1}^{n} \underbrace{\mathbf{M}_i^T(\mathbf{x}(t_0))}_{\textit{adjoint}} \mathbf{H}_i^T [\mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o)] + \mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b).$$

- H_i : opérateur d'observation ;
- $\mathbf{M}_i^T(\mathbf{x}(t_0))$: modèle adjoint.

Formalisme de l'assimilation variationnelle de données

Fonction de coût J :

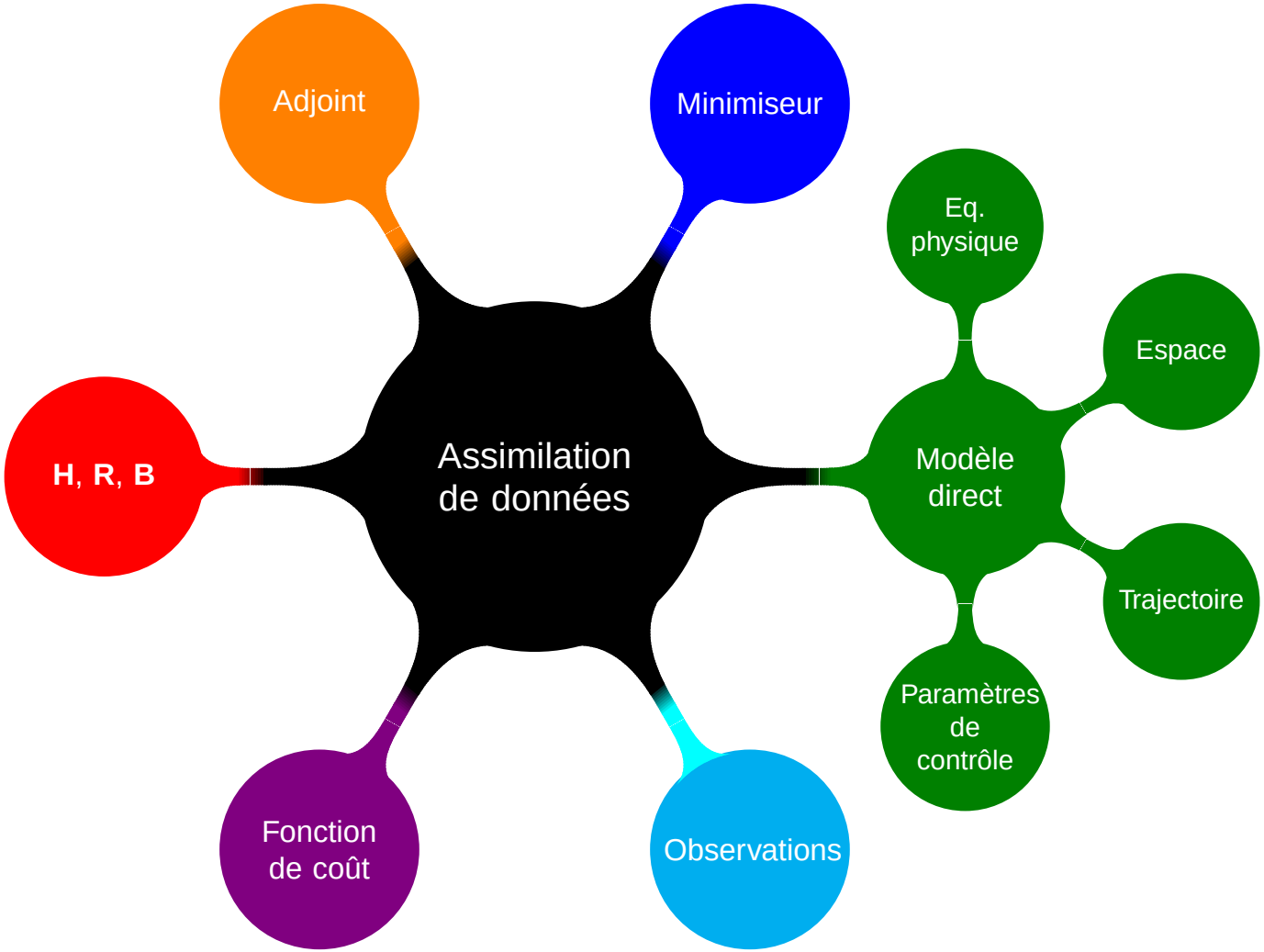
$$J(\mathbf{x}(t_0)) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{y}_i^o)^T \mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o) + \frac{1}{2}(\mathbf{x}(t_0) - \mathbf{x}^b)^T \mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b).$$

- $\mathbf{y}_i$: sorties du modèle ;
- $\mathbf{y}_i^o$: observations au temps t_i ;
- $\mathbf{R}_i$ et $\mathbf{B}$: matrices de variance-covariance ;
- $\mathbf{x}^b$: vecteur d'ébauche.

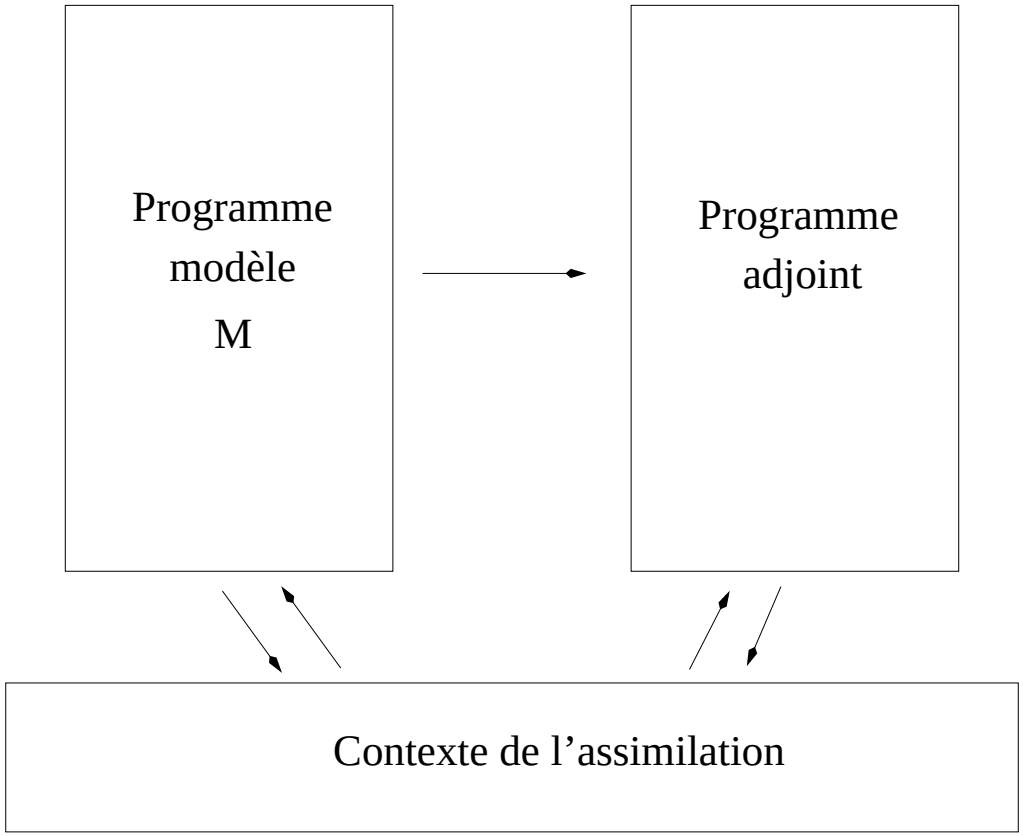
$$\nabla_{\mathbf{x}(t_0)} J = \sum_{i=1}^{n} \underbrace{\mathbf{M}_i^T(\mathbf{x}(t_0))}_{adjoint} \mathbf{H}_i^T [\mathbf{R}_i^{-1}(\mathbf{y}_i - \mathbf{y}_i^o)] + \mathbf{B}^{-1}(\mathbf{x}(t_0) - \mathbf{x}^b).$$

- H_i : opérateur d'observation ;
- $\mathbf{M}_i^T(\mathbf{x}(t_0))$: modèle adjoint.

Résumé assimilation de données



Implémentation informatique



Pourquoi c'est difficile ?

- L'écriture du programme adjoint peut être très lourde ;
- difficulté à tester les nouvelles avancées en assimilation de données ;
- très grandes tailles en jeu, l'ECMWF utilise pour son système opérationnel de prédiction :
  - $2.5 * 10^7$ observations par jour ;
  - vecteur d'état de taille $1.5 * 10^9$;
- optimisation performances :
  - en temps de calcul ;
  - en mémoire allouée (des centaines de To).

Pourquoi c'est difficile ?

- L'écriture du programme adjoint peut être très lourde ;
- difficulté à tester les nouvelles avancées en assimilation de données ;
- très grandes tailles en jeu, l'ECMWF utilise pour son système opérationnel de prédiction :
  - $2.5 * 10^7$ observations par jour ;
  - vecteur d'état de taille $1.5 * 10^9$;
- optimisation performances :
  - en temps de calcul ;
  - en mémoire allouée (des centaines de To).

Pourquoi c'est difficile ?

- L'écriture du programme adjoint peut être très lourde ;
- difficulté à tester les nouvelles avancées en assimilation de données ;
- très grandes tailles en jeu, l'ECMWF utilise pour son système opérationnel de prédiction :
  - $2.5 * 10^7$ observations par jour ;
  - vecteur d'état de taille $1.5 * 10^9$;
- optimisation performances :
  - en temps de calcul ;
  - en mémoire allouée (des centaines de To).

Pourquoi c'est difficile ?

- L'écriture du programme adjoint peut être très lourde ;
- difficulté à tester les nouvelles avancées en assimilation de données ;
- très grandes tailles en jeu, l'ECMWF utilise pour son système opérationnel de prédiction :
  - $2.5 * 10^7$ observations par jour ;
  - vecteur d'état de taille $1.5 * 10^9$;
- optimisation performances :
  - en temps de calcul ;
  - en mémoire allouée (des centaines de To).

Logiciels pour l'assimilation variationnelle

1. Génération du programme adjoint par différentiation automatique : OpenAD, TAF, TAPENADE ;
2. minimiseurs : M1QN3, M2QN1 ;
3. couplage de modèles : OASIS, PALM ;
4. performances : ?

YAO

permet de traiter ces points simultanément.

Logiciels pour l'assimilation variationnelle

1. Génération du programme adjoint par différentiation automatique : OpenAD, TAF, TAPENADE ;
2. minimiseurs : M1QN3, M2QN1 ;
3. couplage de modèles : OASIS, PALM ;
4. performances : ?

YAO

permet de traiter ces points simultanément.

Logiciels pour l'assimilation variationnelle

1. Génération du programme adjoint par différentiation automatique : OpenAD, TAF, TAPENADE ;
2. minimiseurs : M1QN3, M2QN1 ;
3. couplage de modèles : OASIS, PALM ;
4. performances : ?

YAO

permet de traiter ces points simultanément.

Logiciels pour l'assimilation variationnelle

1. Génération du programme adjoint par différentiation automatique : OpenAD, TAF, TAPENADE ;
2. minimiseurs : M1QN3, M2QN1 ;
3. couplage de modèles : OASIS, PALM ;
4. performances : ?

YAO

permet de traiter ces points simultanément.

Logiciels pour l'assimilation variationnelle

1. Génération du programme adjoint par différentiation automatique : OpenAD, TAF, TAPENADE ;
2. minimiseurs : M1QN3, M2QN1 ;
3. couplage de modèles : OASIS, PALM ;
4. performances : ?

YAO

permet de traiter ces points simultanément.

Contribution de la thèse

Un travail de recherche en informatique a été entrepris dans le cadre du logiciel YAO.

Objectif à moyen terme : établir les bases permettant de faire évoluer YAO vers une plateforme générale et opérationnelle pour l'assimilation de données 4D-Var.

- Reformalisation des spécifications de YAO ;
- automatisation de tâches ;
- parallélisation automatique.

Contribution de la thèse

Un travail de recherche en informatique a été entrepris dans le cadre du logiciel YAO.

Objectif à moyen terme : établir les bases permettant de faire évoluer YAO vers une plateforme générale et opérationnelle pour l'assimilation de données 4D-Var.

- Reformalisation des spécifications de YAO ;
- automatisation de tâches ;
- parallélisation automatique.

Sommaire

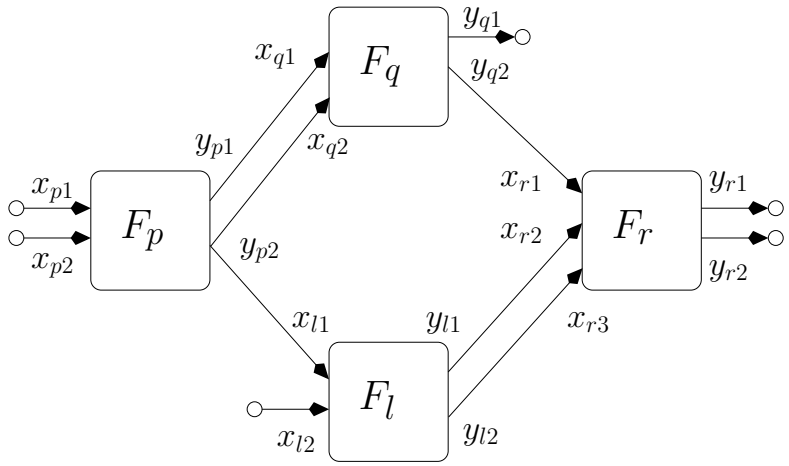
Sommaire

Graphe modulaire : “*divide et impera*”

- *module* : entité de calcul qui représente une fonction ;
- *connexion de base* : transmission de données entre modules.

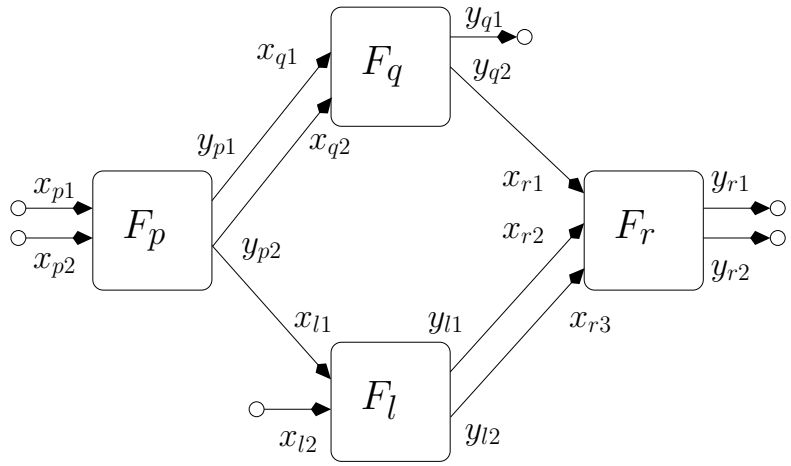
Graphe modulaire : “*divide et impera*”

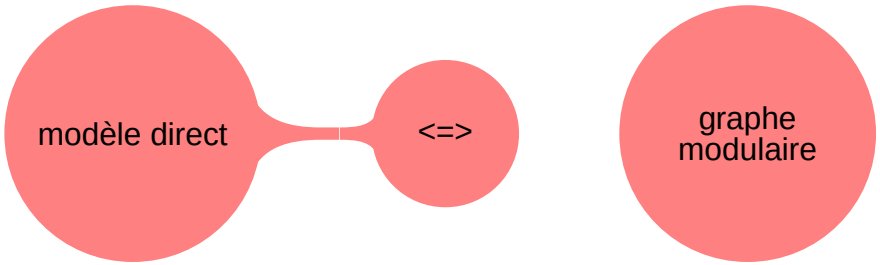
- *module* : entité de calcul qui représente une fonction ;
- *connexion de base* : transmission de données entre modules.



Graphe modulaire : “*divide et impera*”

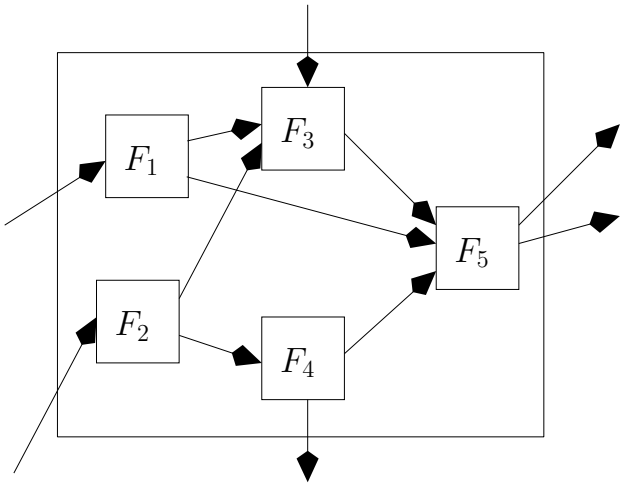
- *module* : entité de calcul qui représente une fonction ;
- *connexion de base* : transmission de données entre modules.





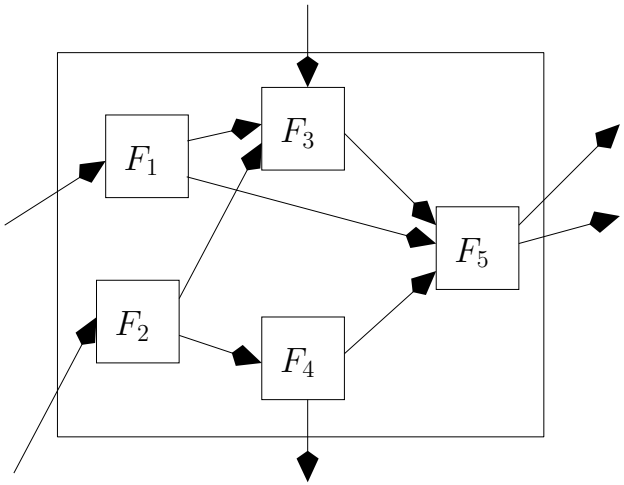
Réplication du graphe

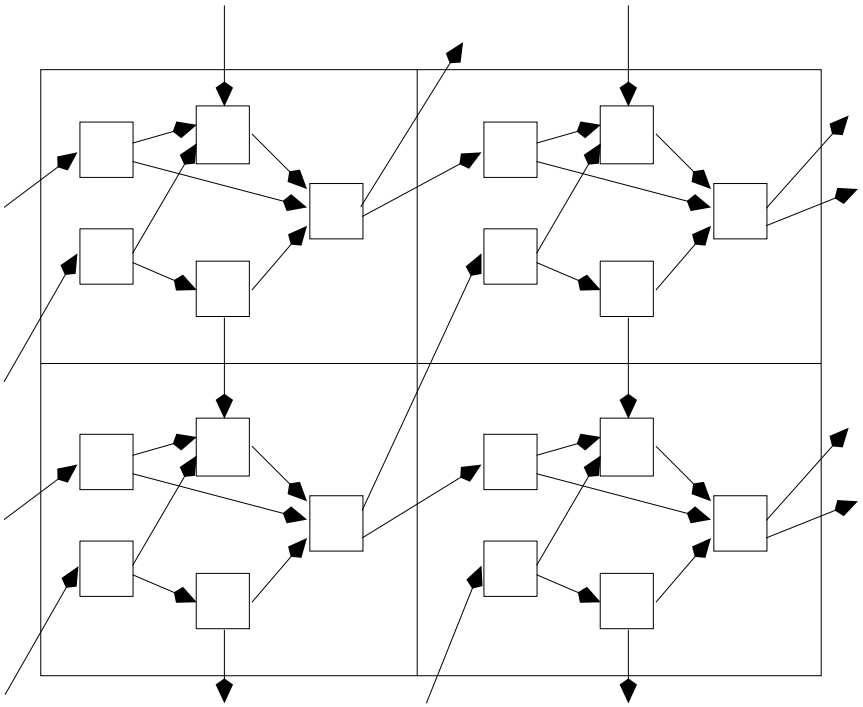
Le même graphe est répété pour chaque point de grille et chaque pas de temps.



Réplication du graphe

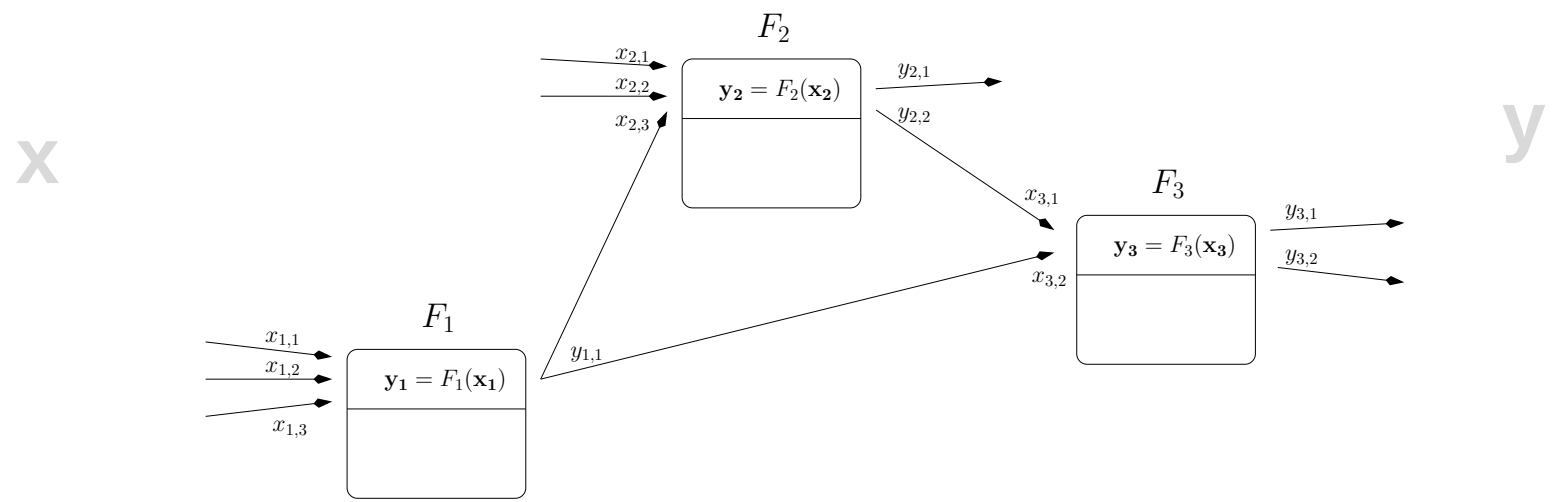
Le même graphe est répété pour chaque point de grille et chaque pas de temps.





Modèle direct

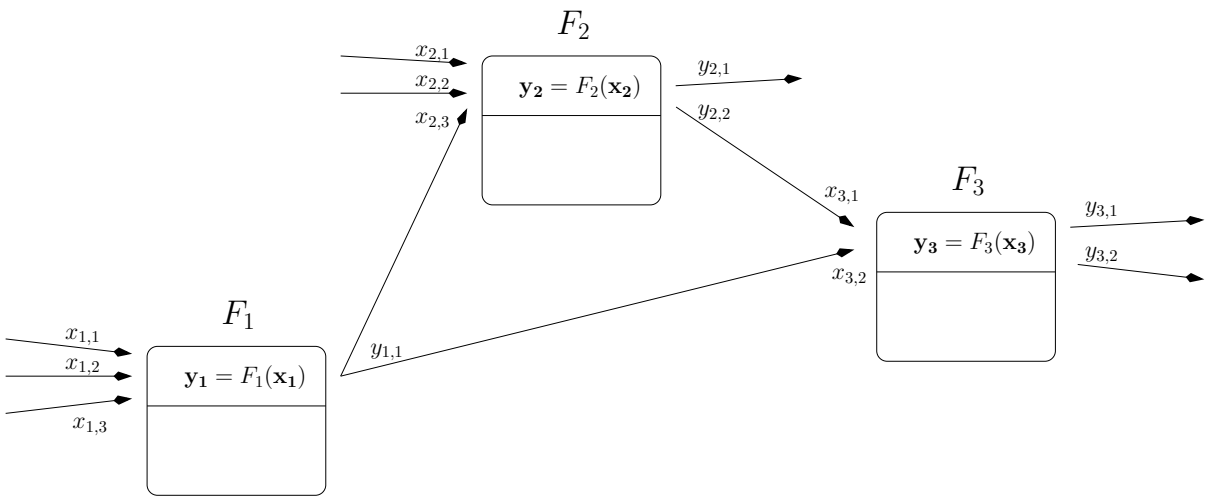
En passe avant on calcule le modèle direct :
procédure *forward*.



Modèle direct

En passe avant on calcule le modèle direct :
procédure *forward*.

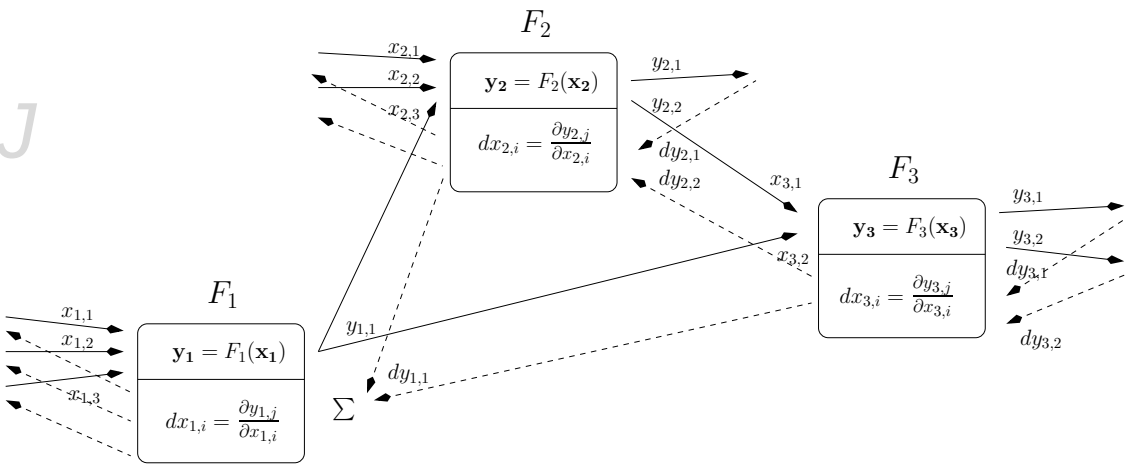
x



y

Modèle adjoint

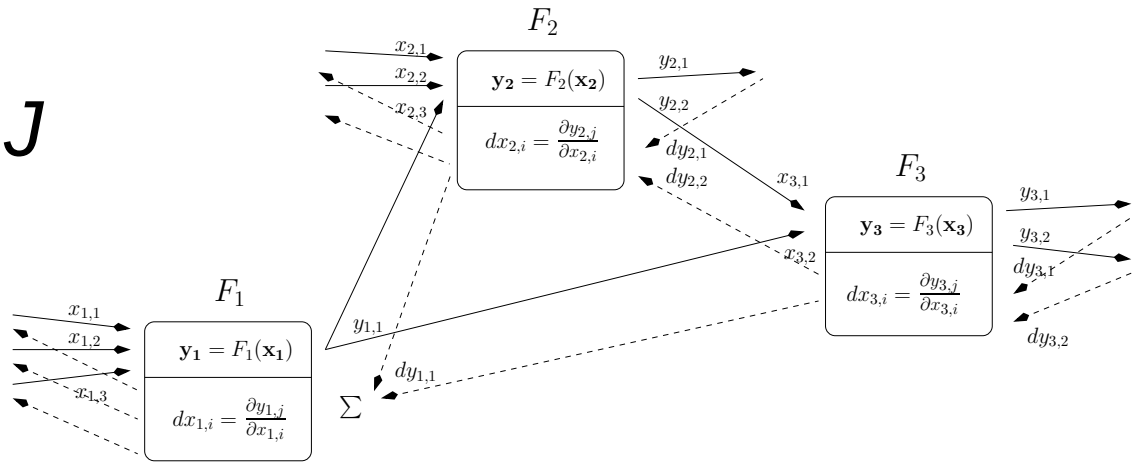
En passe arrière on calcule le modèle adjoint :
procédure *backward*.



Modèle adjoint

En passe arrière on calcule le modèle adjoint :
procédure *backward*.

$\nabla_{\mathbf{x}(t_0)} J$



$\nabla_{\mathbf{y_i}} J$

Le modèle numérique 2D *Shallow-water*

Variables de la dynamique :

$$u_{ijt} = \hat{u}_{ijt-2} + 2\Delta t \left(\frac{-g^*}{\Delta x} \left[h_{i+1jt-1} - h_{ijt-1} \right] + \frac{f}{4} \left[v_{ijt-1} + v_{ij+1t-1} + v_{i+1jt-1} + v_{i+1j+1t-1} \right] - \gamma \cdot \hat{u}_{ijt-2} \right)$$

$$v_{ijt} = \hat{v}_{ijt-2} + 2\Delta t \left(\frac{-g^*}{\Delta y} \left[h_{ijt-1} - h_{ij-1t-1} \right] - \frac{f}{4} \left[u_{i-1j-1t-1} + u_{i-1jt-1} + u_{ij-1t-1} + u_{ijt-1} \right] - \gamma \cdot \hat{v}_{ijt-2} \right)$$

$$h_{ijt} = \hat{h}_{ijt-2} - 2\Delta t \cdot H \left(\frac{u_{ijt-1} - u_{i-1jt-1}}{\Delta x} + \frac{v_{ij+1t-1} - v_{ijt-1}}{\Delta y} \right)$$

Filtre d'Asselin :

$$\begin{aligned} \hat{u}_{ijt} &= u_{ijt-1} + \alpha(\hat{u}_{ijt-1} - 2u_{ijt-1} + u_{ijt}) \\ \hat{v}_{ijt} &= v_{ijt-1} + \alpha(\hat{v}_{ijt-1} - 2v_{ijt-1} + v_{ijt}) \\ \hat{h}_{ijt} &= h_{ijt-1} + \alpha(\hat{h}_{ijt-1} - 2h_{ijt-1} + h_{ijt}) \end{aligned}$$

u,v : vitesses horizontales.

h : hauteur de l'eau.

Le modèle numérique 2D *Shallow-water*

Variables de la dynamique :

$$u_{ijt} = \hat{u}_{ijt-2} + 2\Delta t \left(\frac{-g^*}{\Delta x} \left[h_{i+1jt-1} - h_{ijt-1} \right] + \frac{f}{4} \left[v_{ijt-1} + v_{ij+1t-1} + v_{i+1jt-1} + v_{i+1j+1t-1} \right] - \gamma \cdot \hat{u}_{ijt-2} \right)$$

$$v_{ijt} = \hat{v}_{ijt-2} + 2\Delta t \left(\frac{-g^*}{\Delta y} \left[h_{ijt-1} - h_{ij-1t-1} \right] - \frac{f}{4} \left[u_{i-1j-1t-1} + u_{i-1jt-1} + u_{ij-1t-1} + u_{ijt-1} \right] - \gamma \cdot \hat{v}_{ijt-2} \right)$$

$$h_{ijt} = \hat{h}_{ijt-2} - 2\Delta t \cdot H \left(\frac{u_{ijt-1} - u_{i-1jt-1}}{\Delta x} + \frac{v_{ij+1t-1} - v_{ijt-1}}{\Delta y} \right)$$

Filtre d'Asselin :

$$\begin{aligned} \hat{u}_{ijt} &= u_{ijt-1} + \alpha(\hat{u}_{ijt-1} - 2u_{ijt-1} + u_{ijt}) \\ \hat{v}_{ijt} &= v_{ijt-1} + \alpha(\hat{v}_{ijt-1} - 2v_{ijt-1} + v_{ijt}) \\ \hat{h}_{ijt} &= h_{ijt-1} + \alpha(\hat{h}_{ijt-1} - 2h_{ijt-1} + h_{ijt}) \end{aligned}$$

u,v : vitesses horizontales.

h : hauteur de l'eau.

Langage de description YAO

```
trajectory 100
space 50 50

module Ĥ input 3 output 1
module Û input 3 output 1
module V̂ input 3 output 1
module H input 5 output 1
module U input 7 output 1
module V input 7 output 1

ctin Ĥ from Ĥ i j t-1
ctin Ĥ from H i j t-1
ctin Ĥ from H i j t
ctin Û from Û i j t-1
ctin Û from U i j t-1
ctin Û from U i j t
ctin V̂ from V̂ i j t-1
ctin V̂ from V i j t-1
ctin V̂ from V i j t
ctin H from Ĥ i j t-1
ctin H from U i j t-1
...
...

order YA1
  order YA2
     H U V Ĥ Û V̂
```

Langage de description YAO

```
trajectory 100
space 50 50

module Ĥ input 3 output 1
module Û input 3 output 1
module V̂ input 3 output 1
module H input 5 output 1
module U input 7 output 1
module V input 7 output 1

ctin Ĥ from Ĥ i j t-1
ctin Ĥ from H i j t-1
ctin Ĥ from H i j t
ctin Û from Û i j t-1
ctin Û from U i j t-1
ctin Û from U i j t
ctin V̂ from V̂ i j t-1
ctin V̂ from V i j t-1
ctin V̂ from V i j t
ctin H from Ĥ i j t-1
ctin H from U i j t-1
...
...

order YA1
  order YA2
     H U V Ĥ Û V̂
```

Langage de description YAO

```
trajectory 100
space 50 50

module Ĥ input 3 output 1
module Û input 3 output 1
module V̂ input 3 output 1
module H input 5 output 1
module U input 7 output 1
module V input 7 output 1

ctin Ĥ from Ĥ i j t-1
ctin Ĥ from H i j t-1
ctin Ĥ from H i j t
ctin Û from Û i j t-1
ctin Û from U i j t-1
ctin Û from U i j t
ctin V̂ from V̂ i j t-1
ctin V̂ from V i j t-1
ctin V̂ from V i j t
ctin H from Ĥ i j t-1
ctin H from U i j t-1
...
...

order YA1
  order YA2
    H U V Ĥ Û V̂
```

Langage de description YAO

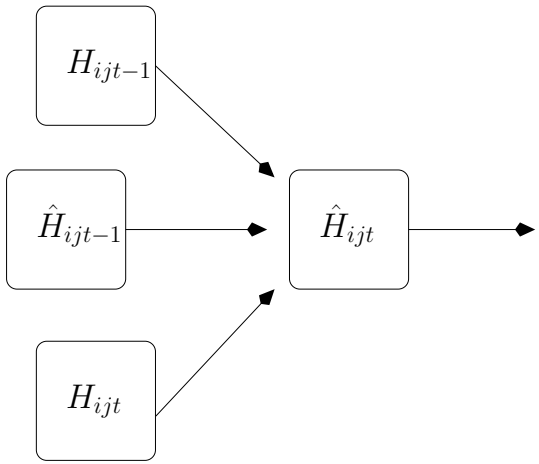
```
trajectory 100
space 50 50

module Ĥ input 3 output 1
module Û input 3 output 1
module V̂ input 3 output 1
module H input 5 output 1
module U input 7 output 1
module V input 7 output 1

ctin Ĥ from Ĥ i j t-1
ctin Ĥ from H i j t-1
ctin Ĥ from H i j t
ctin Û from Û i j t-1
ctin Û from U i j t-1
ctin Û from U i j t
ctin V̂ from V̂ i j t-1
ctin V̂ from V i j t-1
ctin V̂ from V i j t
ctin H from Ĥ i j t-1
ctin H from U i j t-1
...
...

order YA1
  order YA2
    H U V Ĥ Û V̂
```

$$\hat{h}_{ijt} = h_{ijt-1} + \alpha(\hat{h}_{ijt-1} - 2h_{ijt-1} + h_{ijt})$$



Langage de description YAO

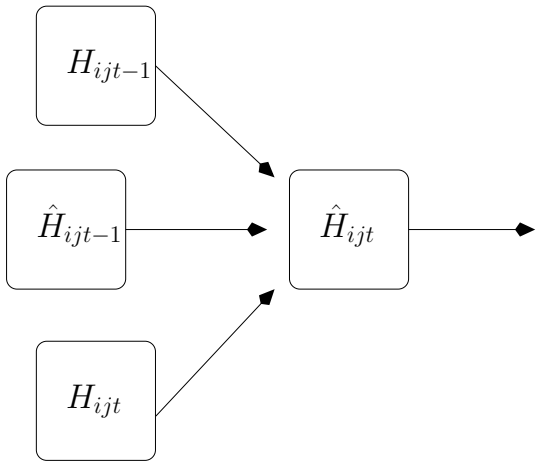
```
trajectory 100
space 50 50

module Ĥ input 3 output 1
module Û input 3 output 1
module V̂ input 3 output 1
module H input 5 output 1
module U input 7 output 1
module V input 7 output 1

ctin Ĥ from Ĥ i j t-1
ctin Ĥ from H i j t-1
ctin Ĥ from H i j t
ctin Û from Û i j t-1
ctin Û from U i j t-1
ctin Û from U i j t
ctin V̂ from V̂ i j t-1
ctin V̂ from V i j t-1
ctin V̂ from V i j t
ctin H from Ĥ i j t-1
ctin H from U i j t-1
...
...

order YA1
  order YA2
     H U V Ĥ Û V̂
```

$$\hat{h}_{ijt} = h_{ijt-1} + \alpha(\hat{h}_{ijt-1} - 2h_{ijt-1} + h_{ijt})$$



Applications YAO

Testé avec succès sur plusieurs modèles en océanographie :

- **Acoustique marine** : inversion variationnelle du profil de la vitesse du son et récupération de paramètres géoacoustiques (célérité, densité, atténuation, . . .) ;
- **ISBA** : assimilation variationnelle de données dans le modèle hydrologique d'interface sol - végétation - atmosphère ;
- **NEMO** (*Nucleus for European Modelling of the Ocean*) : modèle adjoint de la configuration GYRE.
- **NeuroVaria** : inversion variationnelle des mesures multi-spectrales satellitaires de la couleur de l'océan pour la restitution de la chlorophylle-a ;
- **PISCES** : assimilation de données variationnelle sur la couleur de l'océan dans un modèle biogéochimique ;
- **Shallow-water** : assimilation variationnelle de données sur le modèle 2D Shallow-water ;

Sommaire

Directive *ctin*

Connexion de base entre deux modules :

ctin F_d **from** F_s **i** $+ d_i$ **[j** $+ d_j$ **[k** $+ d_k$**]] t** $+ d_t$**.**

$$F_s(\underbrace{i + d_i, j + d_j, k + d_k}_{I'}, \underbrace{t + d_t}_{t'}) \rightarrow F_d(\underbrace{i, j, k}_{I}, \underbrace{t}_{t})$$

Vecteur distance 3D : $I' - I = (d_i, d_j, d_k)$.

Directive *order*

Parcours de la grille par des boucles imbriquées :

order *sensAxe listeInstructions*.

sensAxe est l'axe à parcourir et son sens :

- *YA1* : axe *i* traversée ascendante ;
- *YB2* : axe *j* traversée descendante.

listeInstructions est une suite d'instructions :

- *fonctions de base* ;
- une nouvelle directive *order*.

```
1: order YA1
2:       F1
3:       order YB2
4:             F2 F3
```

for $i = 1$ **à** N_i **do**
  $F_1(i)$
  for $j = N_j$ **à** 1 **do**
    $F_2(i,j)$
    $F_3(i,j)$
  end for
end for

Directive *order*

Parcours de la grille par des boucles imbriquées :

order *sensAxe listeInstructions*.

sensAxe est l'axe à parcourir et son sens :

- *YA1* : axe *i* traversée ascendante ;
- *YB2* : axe *j* traversée descendante.

listeInstructions est une suite d'instructions :

- *fonctions de base* ;
- une nouvelle directive *order*.

```
1: order YA1
2:       F1
3:       order YB2
4:             F2 F3
```

for $i = 1$ **à** N_i **do**
  $F_1(i)$
  for $j = N_j$ **à** 1 **do**
    $F_2(i,j)$
    $F_3(i,j)$
  end for
end for

Directive *order*

Parcours de la grille par des boucles imbriquées :

order *sensAxe listeInstructions*.

sensAxe est l'axe à parcourir et son sens :

- *YA1* : axe *i* traversée ascendante ;
- *YB2* : axe *j* traversée descendante.

listeInstructions est une suite d'instructions :

- *fonctions de base* ;
- une nouvelle directive *order*.

```
1: order YA1
2:       F1
3:       order YB2
4:             F2 F3
```

```
for i = 1 à Ni do
    F1(i)
    for j = Nj à 1 do
        F2(i, j)
        F3(i, j)
    end for
end for
```

Directive *order*

Parcours de la grille par des boucles imbriquées :

order *sensAxe listeInstructions*.

sensAxe est l'axe à parcourir et son sens :

- *YA1* : axe *i* traversée ascendante ;
- *YB2* : axe *j* traversée descendante.

listeInstructions est une suite d'instructions :

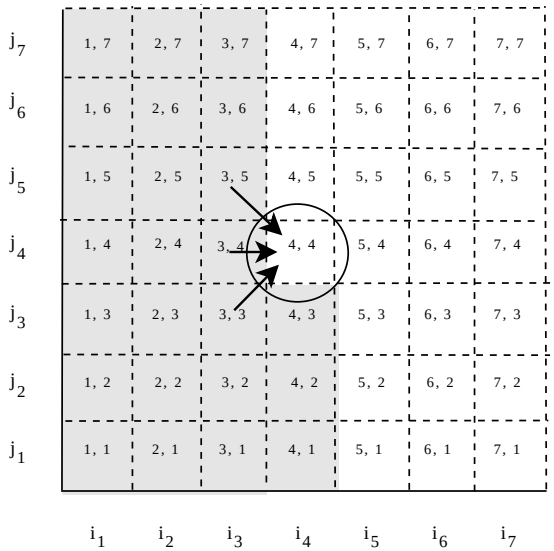
- *fonctions de base* ;
- une nouvelle directive *order*.

```
1: order YA1
2:       F1
3:       order YB2
4:             F2 F3
```

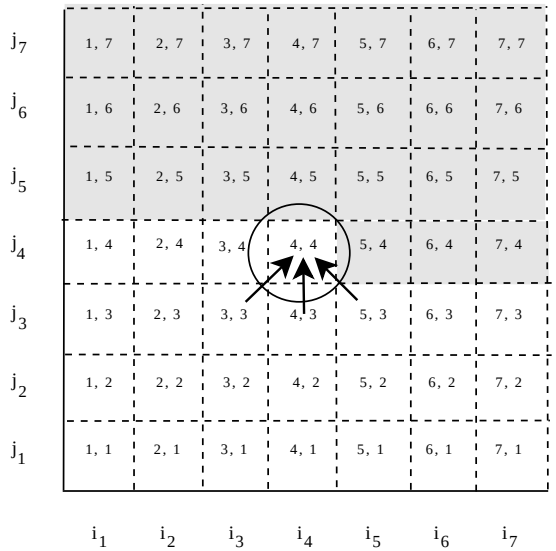
for $i = 1$ **à** N_i **do**
  $F_1(i)$
  for $j = N_j$ **à** 1 **do**
    $F_2(i,j)$
    $F_3(i,j)$
  end for
end for

Cohérence des *ctin* vis-à-vis du parcours choisi

order YA1
order YA2
F_1 F_2



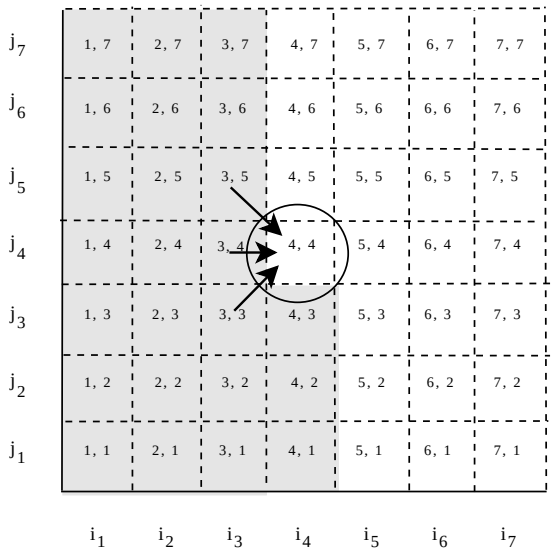
order YB2
order YB1
F_1 F_2



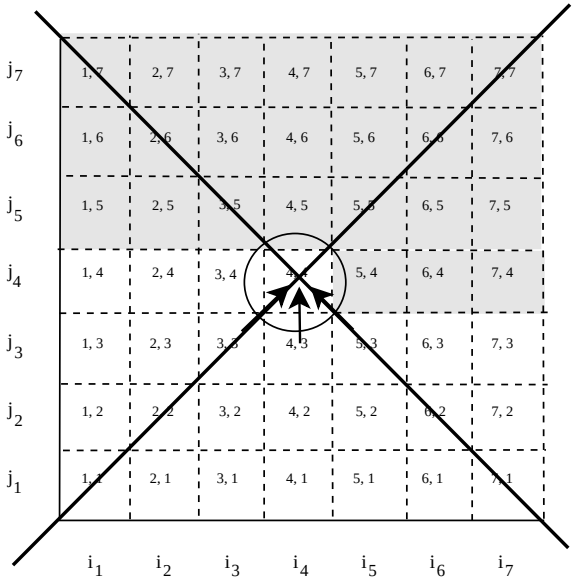
En gris les points de grille calculés et en blanc ceux à calculer.

Cohérence des *ctin* vis-à-vis du parcours choisi

order YA1
order YA2
F_1 F_2



order YB2
order YB1
F_1 F_2



On s'intéresse aux signes et pas à la norme du vecteur distance.

Blocs de directives *order*

order YA1
    F_1
    order YA2
        F_2 F_3
order YA2
    order YA1
        F_4
        order YB3
            F_5
        order YA3
            F_6
order YB2
    order YB3
        order YA1
            F_7

Un bloc est composé d'une ou de plusieurs directives *order* imbriquées.

Chaque bloc est relatif à une boucle extérieure.

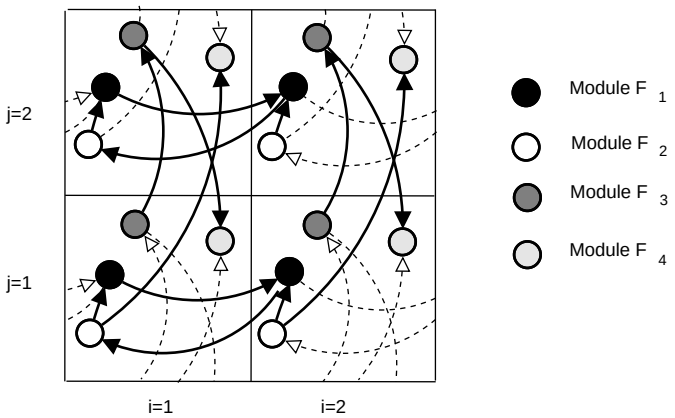
Tous les blocs sont dans la boucle du temps.

Génération automatique de directives *order*

Objectif : générer des blocs de directives *order* qui permettent la propagation du calcul dans le graphe modulaire.

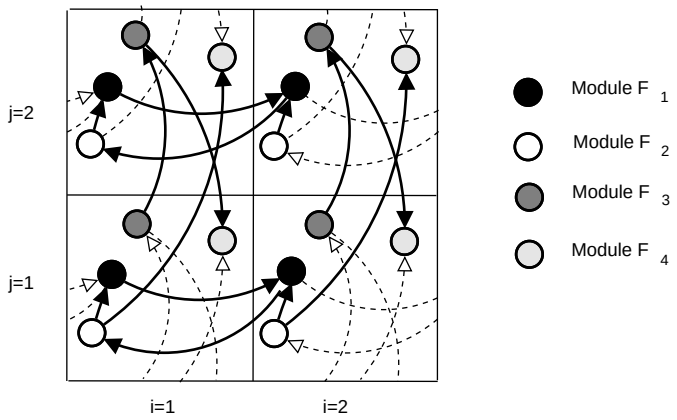
Graphes de dépendance

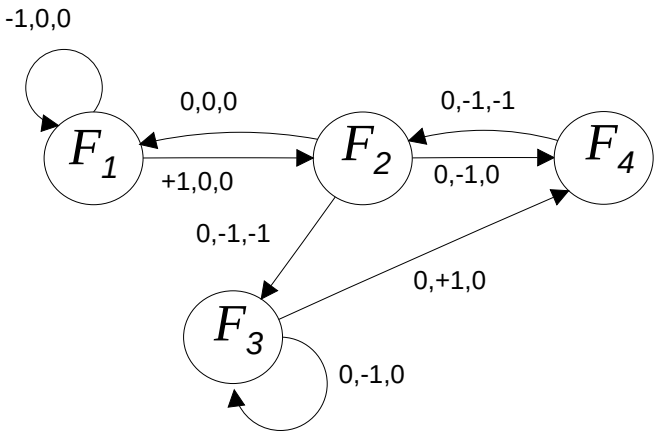
$$F_1(i-1,j,t) \longrightarrow F_1(i,j,t)$$
$$F_1(i+1,j,t) \longrightarrow F_2(i,j,t)$$
$$F_2(i,j,t) \longrightarrow F_1(i,j,t)$$
$$F_2(i,j-1,t) \longrightarrow F_4(i,j,t)$$
$$F_2(i,j-1,t-1) \longrightarrow F_3(i,j,t)$$
$$F_3(i,j-1,t) \longrightarrow F_3(i,j,t)$$
$$F_3(i,j+1,t) \longrightarrow F_4(i,j,t)$$
$$F_4(i,j-1,t-1) \longrightarrow F_2(i,j,t)$$



Graphes de dépendance

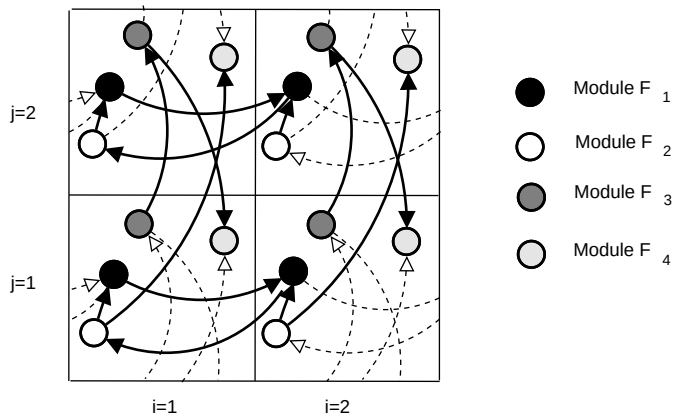
$$F_1(i-1,j,t) \longrightarrow F_1(i,j,t)$$
$$F_1(i+1,j,t) \longrightarrow F_2(i,j,t)$$
$$F_2(i,j,t) \longrightarrow F_1(i,j,t)$$
$$F_2(i,j-1,t) \longrightarrow F_4(i,j,t)$$
$$F_2(i,j-1,t-1) \longrightarrow F_3(i,j,t)$$
$$F_3(i,j-1,t) \longrightarrow F_3(i,j,t)$$
$$F_3(i,j+1,t) \longrightarrow F_4(i,j,t)$$
$$F_4(i,j-1,t-1) \longrightarrow F_2(i,j,t)$$

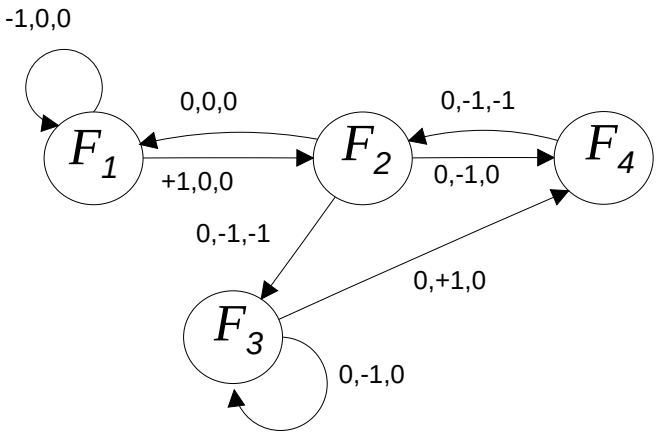




Graphes de dépendance

$$F_1(i-1,j,t) \longrightarrow F_1(i,j,t)$$
$$F_1(i+1,j,t) \longrightarrow F_2(i,j,t)$$
$$F_2(i,j,t) \longrightarrow F_1(i,j,t)$$
$$F_2(i,j-1,t) \longrightarrow F_4(i,j,t)$$
$$F_2(i,j-1,t-1) \longrightarrow F_3(i,j,t)$$
$$F_3(i,j-1,t) \longrightarrow F_3(i,j,t)$$
$$F_3(i,j+1,t) \longrightarrow F_4(i,j,t)$$
$$F_4(i,j-1,t-1) \longrightarrow F_2(i,j,t)$$



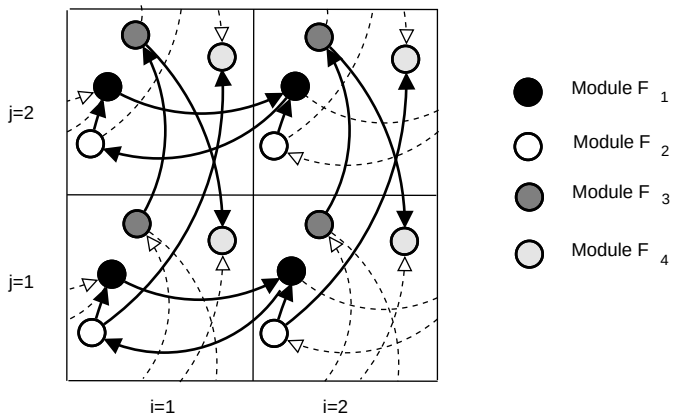


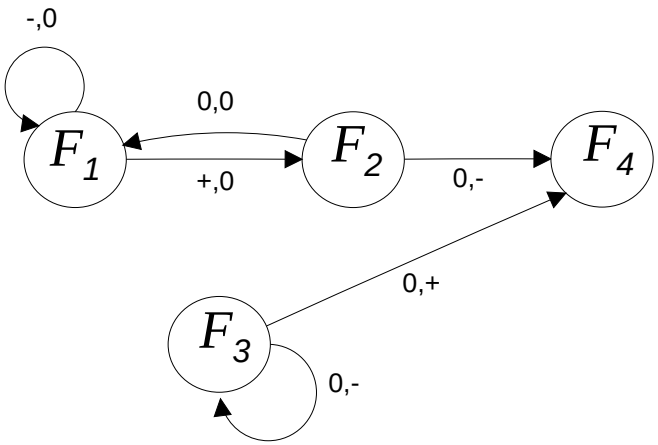
Graphe de Dépendance Réduit *GDR* :

- issu directement des *ctin* ;
- circuits et boucles ;
- pas de coordonnée temporelle.

Graphes de dépendance

$$F_1(i-1,j,t) \longrightarrow F_1(i,j,t)$$
$$F_1(i+1,j,t) \longrightarrow F_2(i,j,t)$$
$$F_2(i,j,t) \longrightarrow F_1(i,j,t)$$
$$F_2(i,j-1,t) \longrightarrow F_4(i,j,t)$$
$$F_2(i,j-1,t-1) \longrightarrow F_3(i,j,t)$$
$$F_3(i,j-1,t) \longrightarrow F_3(i,j,t)$$
$$F_3(i,j+1,t) \longrightarrow F_4(i,j,t)$$
$$F_4(i,j-1,t-1) \longrightarrow F_2(i,j,t)$$





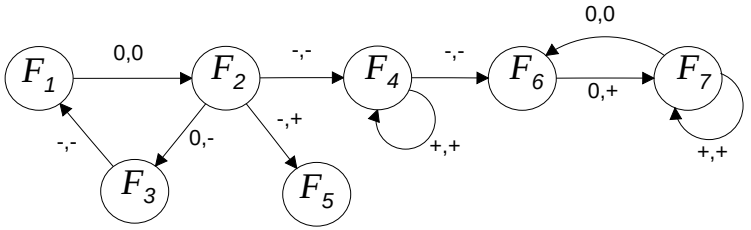
Graphe de Dépendance Réduit *GDR* :

- issu directement des *ctin* ;
- circuits et boucles ;
- pas de coordonnée temporelle.

Notions de la théorie des graphes

Définition

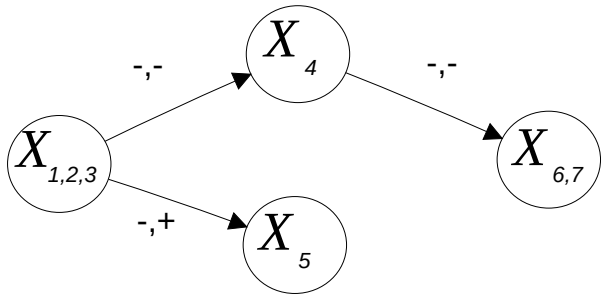
Une Composante Fortement Connexe (CFC) X d'un graphe orienté G est un sous-graphe maximal de G tel que pour tout couple de sommets F_u et F_v dans ce sous-graphe, il existe un chemin de F_u à F_v et un chemin de F_v à F_u.



Définition

Le graphe réduit est défini de la manière suivante :

- *les sommets sont les X de G ;*
- *pour tout arc (F_u, F_v) de G telle que F_u et F_v sont dans deux composantes distinctes X_u et X_v, on ajoute un arc de X_u à X_v.*

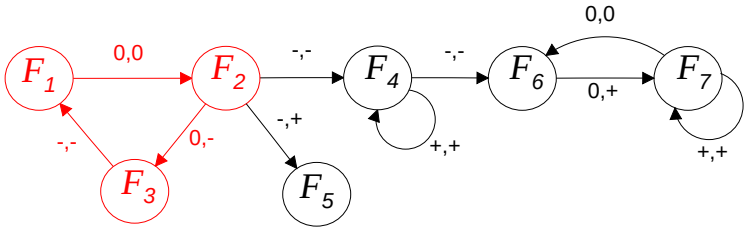


Le graphe réduit est toujours acyclique.

Notions de la théorie des graphes

Définition

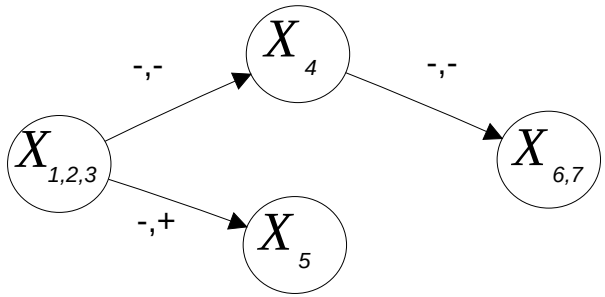
Une Composante Fortement Connexe (CFC) X d'un graphe orienté G est un sous-graphe maximal de G tel que pour tout couple de sommets F_u et F_v dans ce sous-graphe, il existe un chemin de F_u à F_v et un chemin de F_v à F_u.



Définition

Le graphe réduit est défini de la manière suivante :

- *les sommets sont les X de G ;*
- *pour tout arc (F_u, F_v) de G telle que F_u et F_v sont dans deux composantes distinctes X_u et X_v, on ajoute un arc de X_u à X_v.*

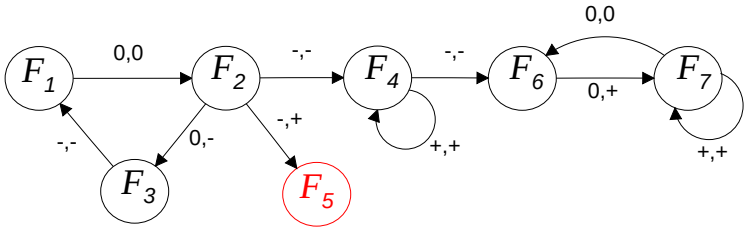


Le graphe réduit est toujours acyclique.

Notions de la théorie des graphes

Définition

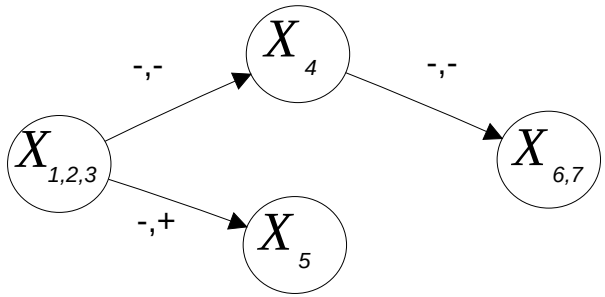
Une Composante Fortement Connexe (CFC) X d'un graphe orienté G est un sous-graphe maximal de G tel que pour tout couple de sommets F_u et F_v dans ce sous-graphe, il existe un chemin de F_u à F_v et un chemin de F_v à F_u.



Définition

Le graphe réduit est défini de la manière suivante :

- *les sommets sont les X de G ;*
- *pour tout arc (F_u, F_v) de G telle que F_u et F_v sont dans deux composantes distinctes X_u et X_v, on ajoute un arc de X_u à X_v.*

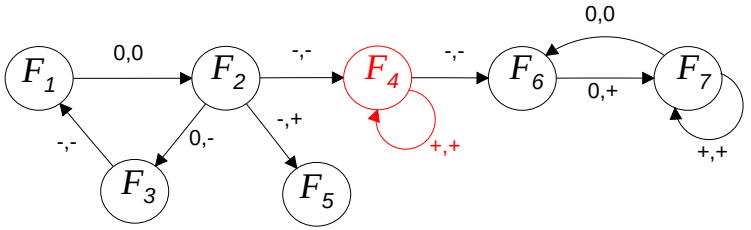


Le graphe réduit est toujours acyclique.

Notions de la théorie des graphes

Définition

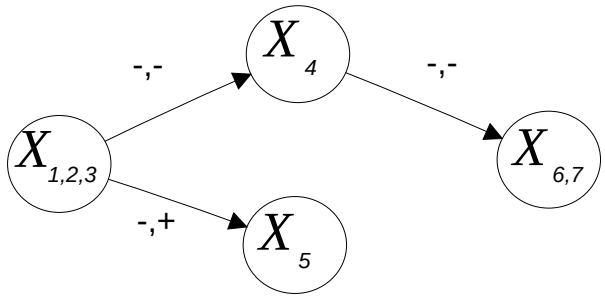
Une Composante Fortement Connexe (CFC) X d'un graphe orienté G est un sous-graphe maximal de G tel que pour tout couple de sommets F_u et F_v dans ce sous-graphe, il existe un chemin de F_u à F_v et un chemin de F_v à F_u.



Définition

Le graphe réduit est défini de la manière suivante :

- *les sommets sont les X de G ;*
- *pour tout arc (F_u, F_v) de G telle que F_u et F_v sont dans deux composantes distinctes X_u et X_v, on ajoute un arc de X_u à X_v.*

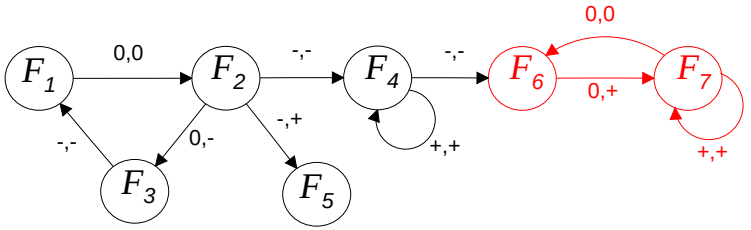


Le graphe réduit est toujours acyclique.

Notions de la théorie des graphes

Définition

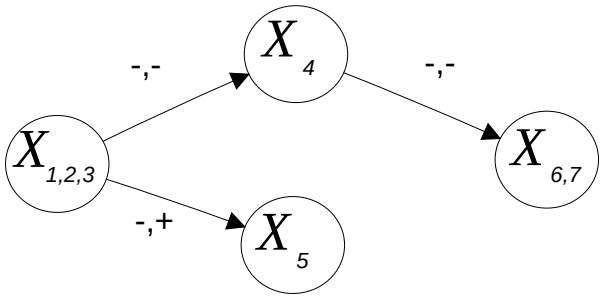
Une Composante Fortement Connexe (CFC) X d'un graphe orienté G est un sous-graphe maximal de G tel que pour tout couple de sommets F_u et F_v dans ce sous-graphe, il existe un chemin de F_u à F_v et un chemin de F_v à F_u.



Définition

Le graphe réduit est défini de la manière suivante :

- *les sommets sont les X de G ;*
- *pour tout arc (F_u, F_v) de G telle que F_u et F_v sont dans deux composantes distinctes X_u et X_v, on ajoute un arc de X_u à X_v.*

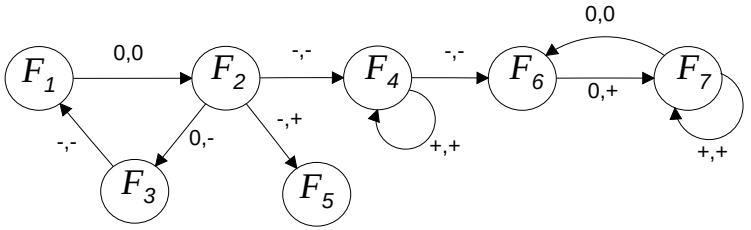


Le graphe réduit est toujours acyclique.

Notions de la théorie des graphes

Définition

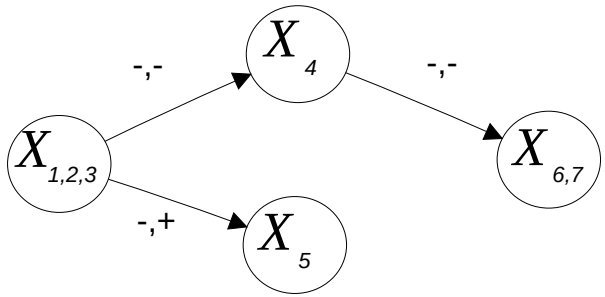
Une Composante Fortement Connexe (CFC) X d'un graphe orienté G est un sous-graphe maximal de G tel que pour tout couple de sommets F_u et F_v dans ce sous-graphe, il existe un chemin de F_u à F_v et un chemin de F_v à F_u.



Définition

Le graphe réduit est défini de la manière suivante :

- *les sommets sont les X de G ;*
- *pour tout arc (F_u, F_v) de G telle que F_u et F_v sont dans deux composantes distinctes X_u et X_v, on ajoute un arc de X_u à X_v.*

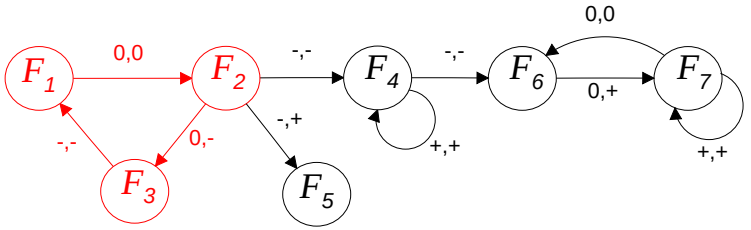


Le graphe réduit est toujours acyclique.

Notions de la théorie des graphes

Définition

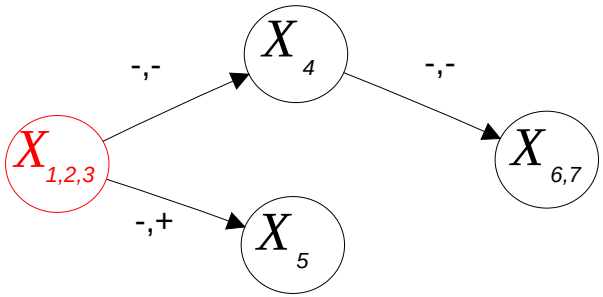
Une Composante Fortement Connexe (CFC) X d'un graphe orienté G est un sous-graphe maximal de G tel que pour tout couple de sommets F_u et F_v dans ce sous-graphe, il existe un chemin de F_u à F_v et un chemin de F_v à F_u.



Définition

Le graphe réduit est défini de la manière suivante :

- *les sommets sont les X de G ;*
- *pour tout arc (F_u, F_v) de G telle que F_u et F_v sont dans deux composantes distinctes X_u et X_v, on ajoute un arc de X_u à X_v.*

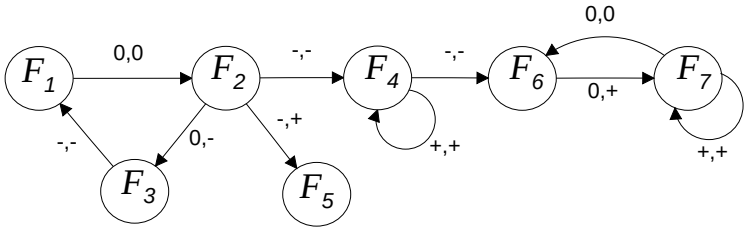


Le graphe réduit est toujours acyclique.

Notions de la théorie des graphes

Définition

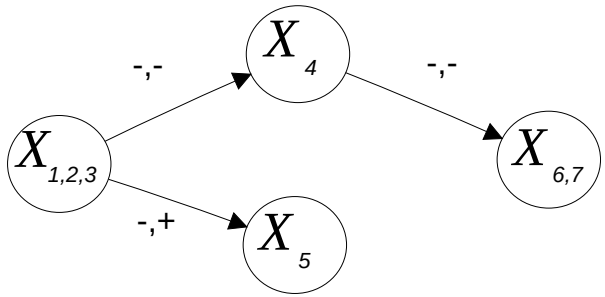
Une Composante Fortement Connexe (CFC) X d'un graphe orienté G est un sous-graphe maximal de G tel que pour tout couple de sommets F_u et F_v dans ce sous-graphe, il existe un chemin de F_u à F_v et un chemin de F_v à F_u.



Définition

Le graphe réduit est défini de la manière suivante :

- *les sommets sont les X de G ;*
- *pour tout arc (F_u, F_v) de G telle que F_u et F_v sont dans deux composantes distinctes X_u et X_v, on ajoute un arc de X_u à X_v.*



Le graphe réduit est toujours acyclique.

Anomalies dans les *ctin*

Peut-on toujours générer des directives *order* par rapport aux directives *ctin* ?

Définition

Des directives ctin *particulières présentent une anomalie s'il n'est pas possible de parcourir le graphe modulaire par un système de boucles imbriquées.*

Types d'anomalies

- **Anomalie structurelle** : le graphe modulaire généré contient des circuits ;

- **anomalie non structurelle** : le graphe modulaire est sans circuit mais sa structure ne permet pas de lui associer un ordre topologique sous la forme de boucles imbriquées.

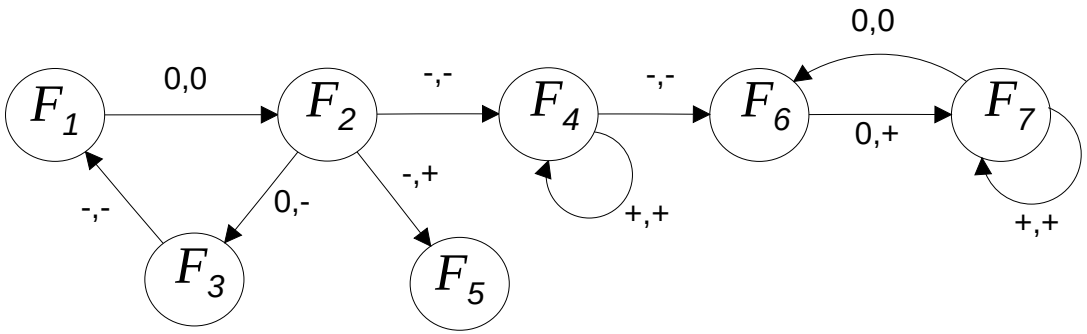
Types d'anomalies

- **Anomalie structurelle** : le graphe modulaire généré contient des circuits ;
- **anomalie non structurelle** : le graphe modulaire est sans circuit mais sa structure ne permet pas de lui associer un ordre topologique sous la forme de boucles imbriquées.

Appartenance des fonctions à une CFC

Résultat

Si X est une CFC du GDR, alors toutes les fonctions de base *de X doivent appartenir à un même bloc de directives* order*.*



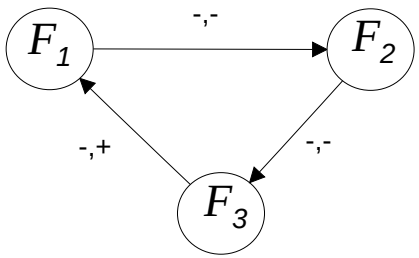
Signe commun dans une CFC

Signe commun

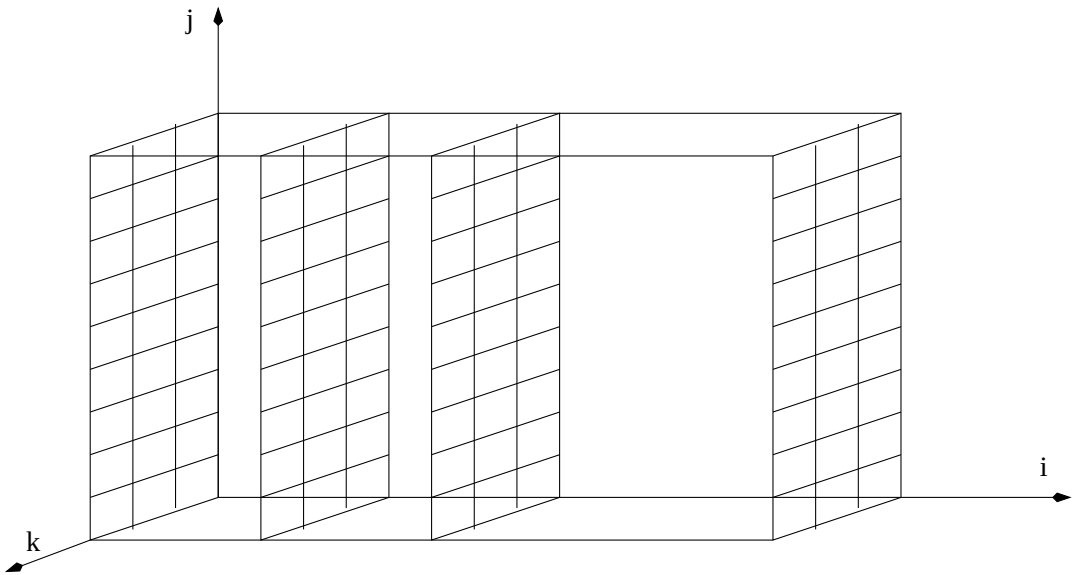
Etant donné un ensemble d'arcs du *GDR*, un axe *l* a un signe commun (ou un même signe) si ses distances d_l sont toutes ≤ 0 ou ≥ 0.

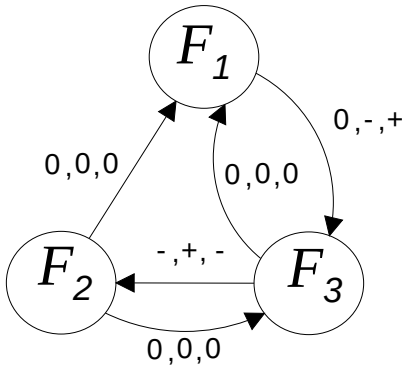
Résultat

Soit X une CFC. Si on considère les arcs de X, il doit exister au moins un axe l avec distances d_l de même signe.

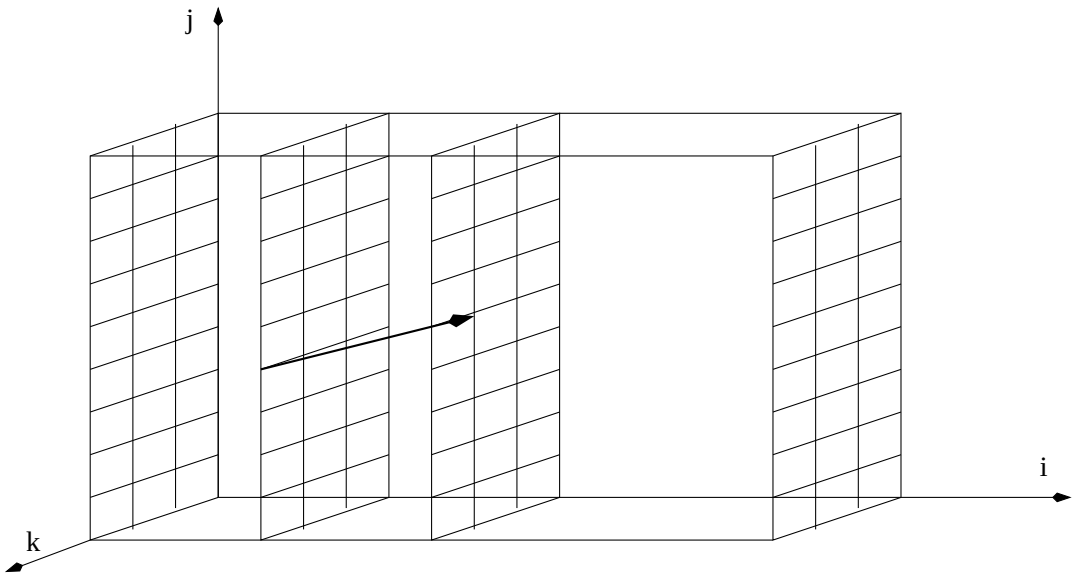


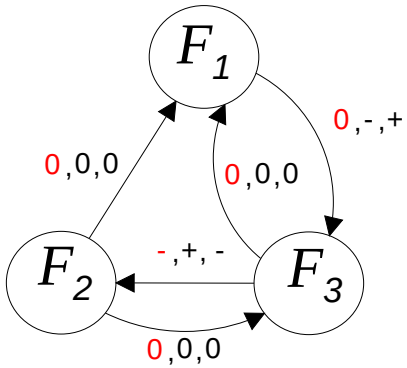
Sous-espaces affines I



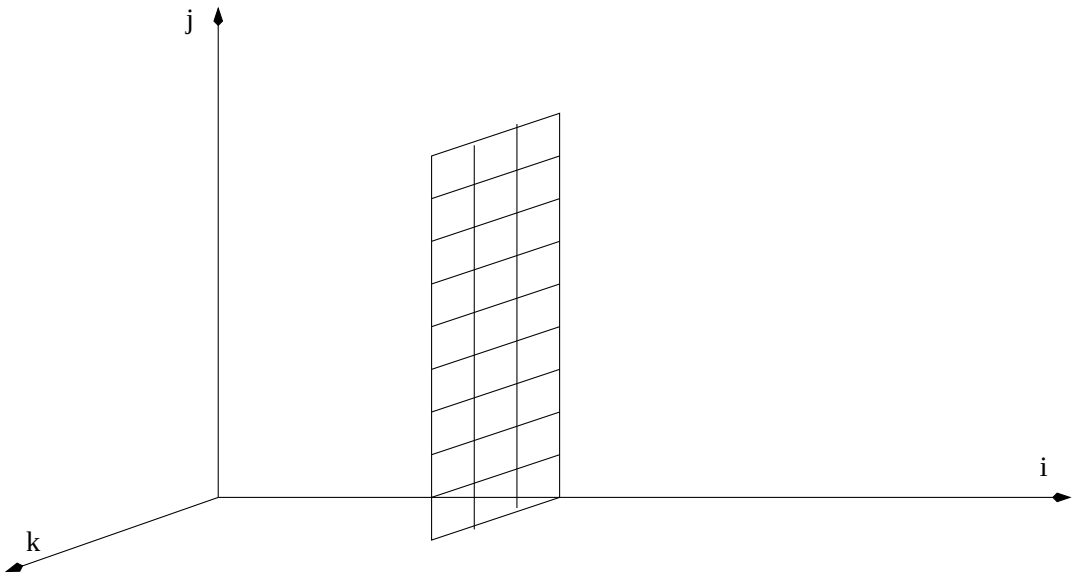


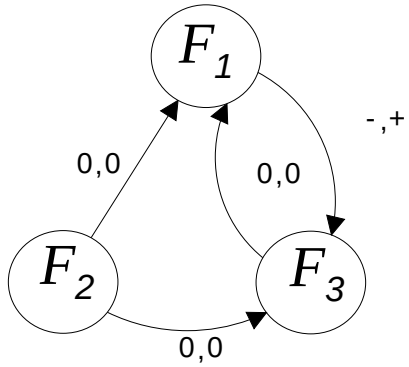
Sous-espaces affines II



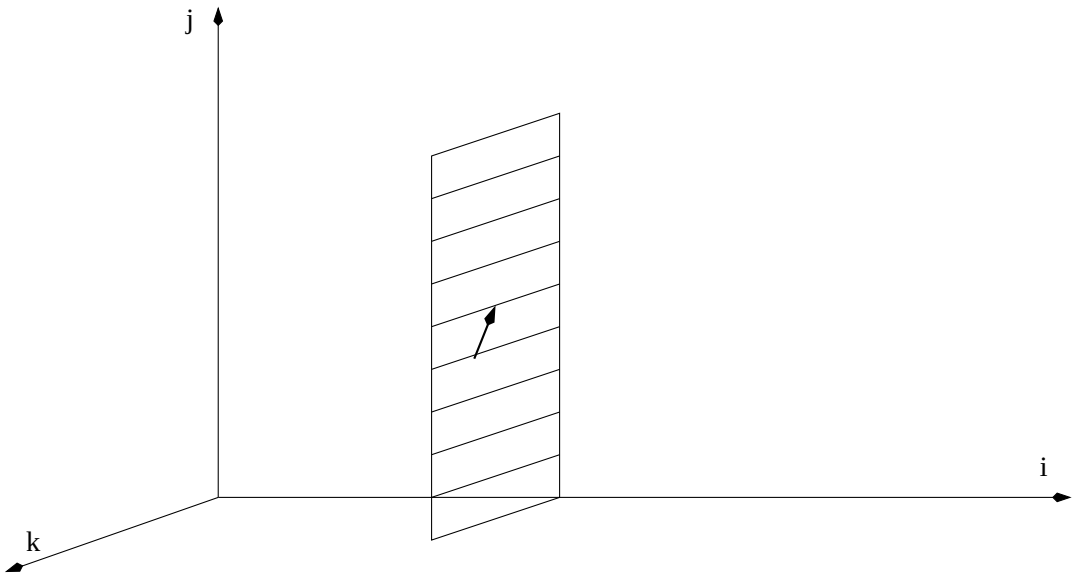


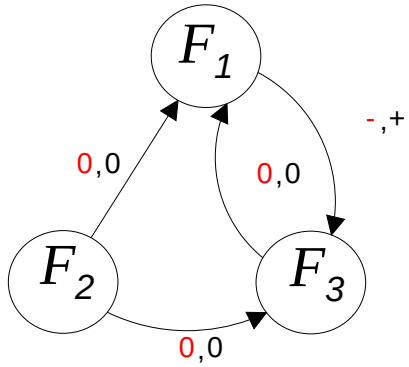
Sous-espaces affines III



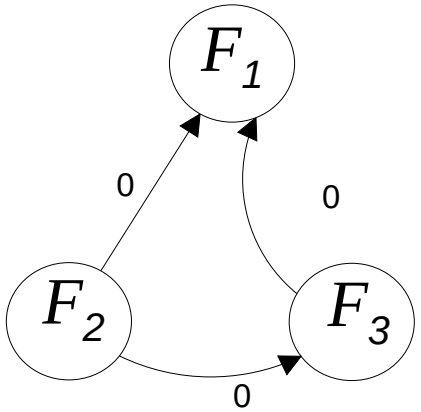


Sous-espaces affines IV





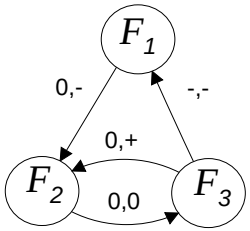
Sous-espaces affines V



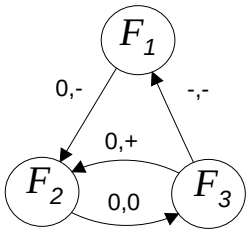
Algorithme de traitement d'un axe

- chercher les CFC ;
- ordonnancer les CFC du graphe réduit selon un ordre topologique ;
- en suivant cet ordre pour chaque CFC X :
  - chercher un axe l_X avec les distances d_{l_X} de signe commun ;
  - générer une directive *order* en fonction de cet axe ;
  - transformer le graphe correspondant à X.

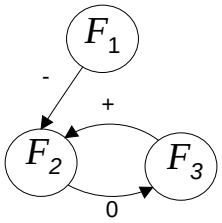
Exemple de génération



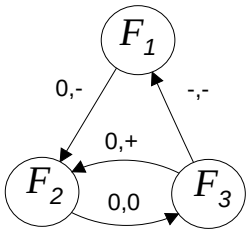
Exemple de génération



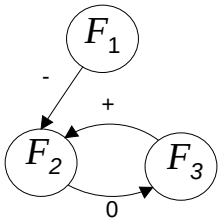
order YA1



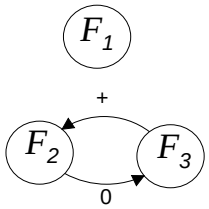
Exemple de génération



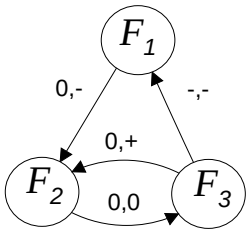
order YA1



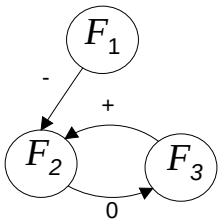
order YA1
order YX2



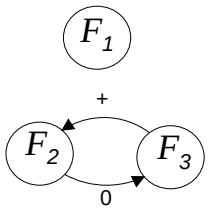
Exemple de génération



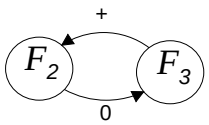
order YA1



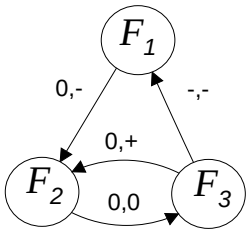
order YA1
order YX2



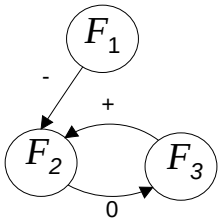
order YA1
order YX2
F_1



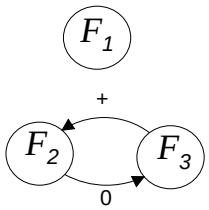
Exemple de génération



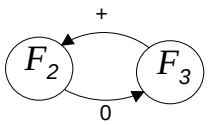
order YA1



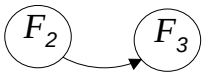
order YA1
order YX2



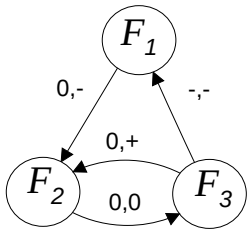
order YA1
order YX2
F_1



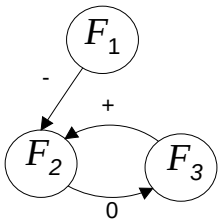
order YA1
order YX2
F_1
order YB2



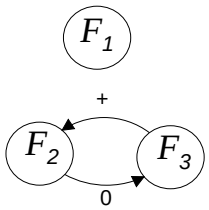
Exemple de génération



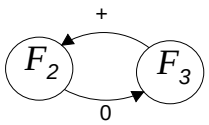
order YA1



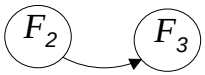
order YA1
order YX2



order YA1
order YX2
F_1



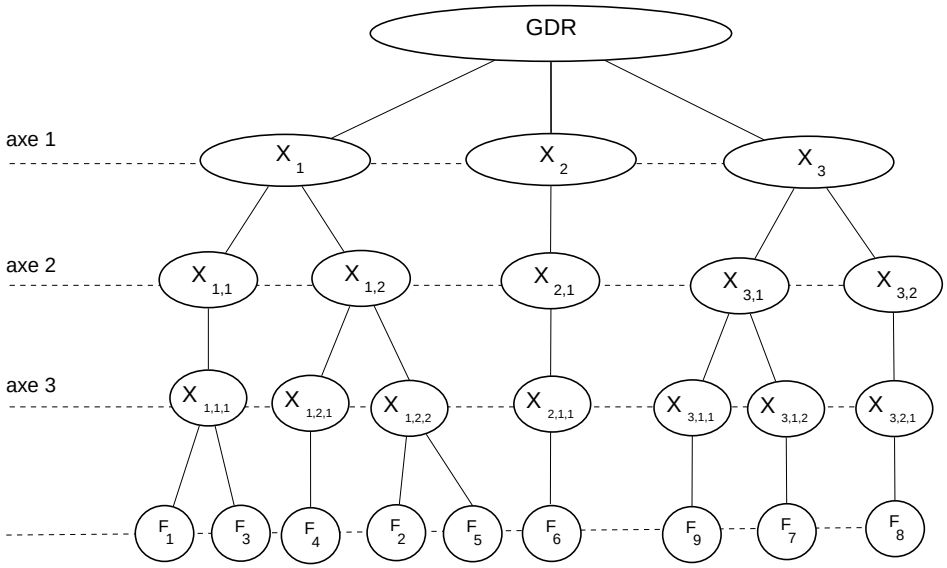
order YA1
order YX2
F_1
order YB2



order YA1
order YX2
F_1
order YB2
F_2 F_3

Algorithme testé sur plusieurs applications.

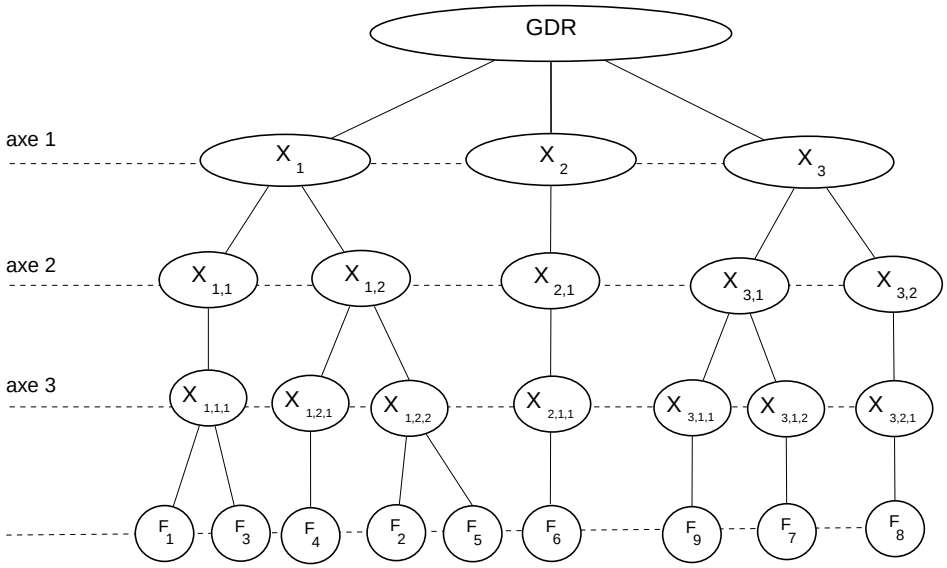
Les blocs réalisent une partition de l'ensemble des *fonctions de base*.



Peut-on faire une fusion des blocs générés ?

Algorithme testé sur plusieurs applications.

Les blocs réalisent une partition de l'ensemble des *fonctions de base*.



Peut-on faire une fusion des blocs générés ?

Arguments pour la fusion de blocs I

Liste de boucles possibles

Etant donné un ensemble d'arcs du *GDR* on lui associe une liste L d'éléments $(axe, signe)$.

Exemple 1

L'axe i de l'ensemble des arcs a des $d_i = 0$:

$$L = ((i, 0), (i, -), (i, +)).$$

Exemple 2

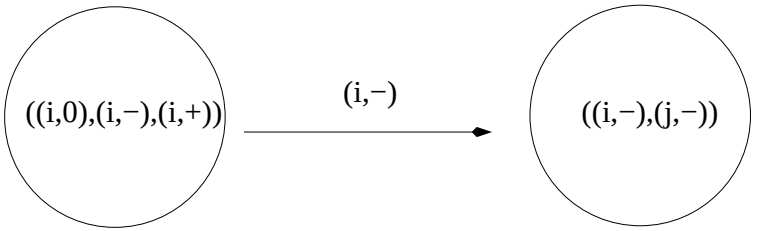
Les axes i et j de l'ensemble ont des d_i et des d_j qui sont ≤ 0 (avec au moins un d_i et un $d_j < 0$) :

$$L = ((i, -), (j, -)).$$

Arguments pour la fusion de blocs II

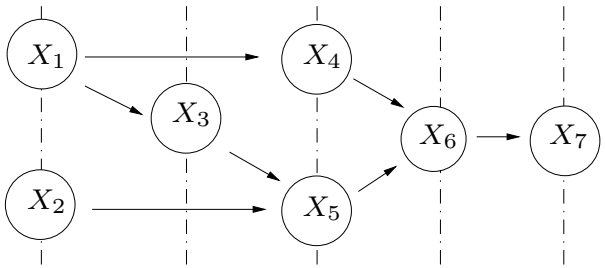
On considère le graphe réduit avec sommets X_i :

- à chaque X_i on associe la liste L_i correspondant au sous-ensemble d'arcs qui ont été réduits ;
- étant donné X_u et X_v, si l'ensemble d'arcs de X_u vers X_v est non vide :
  - on lui associe sa liste $L_{u,v}$;
  - on considère un arc $L_u \longrightarrow L_v$ valué par $L_{u,v}$.



Graphe réduit par niveaux

Afin de bien mener le processus de fusion on organise le graphe réduit par niveaux.

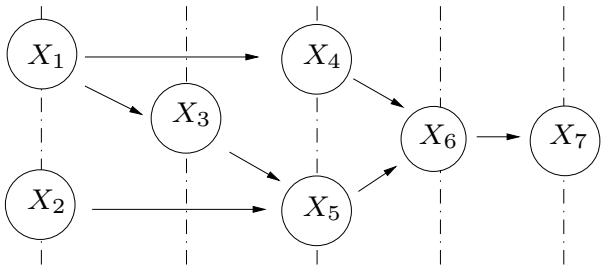


La fusion de X_u et X_v peut se faire si :

1. il n'y a pas d'arc et $L_u \cap L_v \neq \emptyset$;
2. il y a un arc $X_u \rightarrow X_v$ et $L_u \cap L_{u,v} \cap L_v \neq \emptyset$;
3. X_u et X_v ne sont pas dans deux niveaux voisins et il n'y a pas de chemin entre eux (en suivant les points 1 et 2).

Graphe réduit par niveaux

Afin de bien mener le processus de fusion on organise le graphe réduit par niveaux.

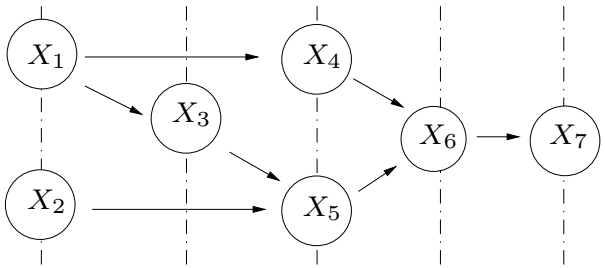


La fusion de X_u et X_v peut se faire si :

1. il n'y a pas d'arc et $L_u \cap L_v \neq \emptyset$;
2. il y a un arc $X_u \rightarrow X_v$ et $L_u \cap L_{u,v} \cap L_v \neq \emptyset$;
3. X_u et X_v ne sont pas dans deux niveaux voisins et il n'y a pas de chemin entre eux (en suivant les points 1 et 2).

Graphe réduit par niveaux

Afin de bien mener le processus de fusion on organise le graphe réduit par niveaux.

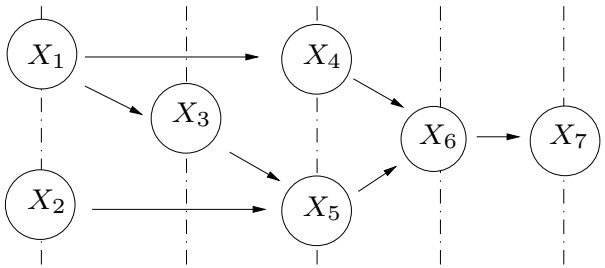


La fusion de X_u et X_v peut se faire si :

1. il n'y a pas d'arc et $L_u \cap L_v \neq \emptyset$;
2. il y a un arc $X_u \rightarrow X_v$ et $L_u \cap L_{u,v} \cap L_v \neq \emptyset$;
3. X_u et X_v ne sont pas dans deux niveaux voisins et il n'y a pas de chemin entre eux (en suivant les points 1 et 2).

Graphe réduit par niveaux

Afin de bien mener le processus de fusion on organise le graphe réduit par niveaux.



La fusion de X_u et X_v peut se faire si :

1. il n'y a pas d'arc et $L_u \cap L_v \neq \emptyset$;
2. il y a un arc $X_u \rightarrow X_v$ et $L_u \cap L_{u,v} \cap L_v \neq \emptyset$;
3. X_u et X_v ne sont pas dans deux niveaux voisins et il n'y a pas de chemin entre eux (en suivant les points 1 et 2).

Génération de directives order : conclusion

L'écriture d'une application YAO devient plus simple pour l'utilisateur.

Possibilité d'optimiser le code généré selon certains critères à définir selon le langage utilisé et le modèle machine.

Modèle machine

Contient les informations de type : nombre de registres, nombre de mémoires cache (L1, L2, L3) et leur taille, taille de la mémoire RAM, type d'architecture (32 ou 64 bits), etc..

Une fusion très importante des boucles :

- réduit immédiatement le surcoût dû au fonctionnement des boucles ;
- pourrait empêcher un fonctionnement optimal de la hiérarchie mémoire.

Est-il possible d'exploiter cette génération automatique pour du calcul haute performance ?

Génération de directives order : conclusion

L'écriture d'une application YAO devient plus simple pour l'utilisateur.

Possibilité d'optimiser le code généré selon certains critères à définir selon le langage utilisé et le modèle machine.

Modèle machine

Contient les informations de type : nombre de registres, nombre de mémoires cache (L1, L2, L3) et leur taille, taille de la mémoire RAM, type d'architecture (32 ou 64 bits), etc..

Une fusion très importante des boucles :

- réduit immédiatement le surcoût dû au fonctionnement des boucles ;
- pourrait empêcher un fonctionnement optimal de la hiérarchie mémoire.

Est-il possible d'exploiter cette génération automatique pour du calcul haute performance ?

Génération de directives order : conclusion

L'écriture d'une application YAO devient plus simple pour l'utilisateur.

Possibilité d'optimiser le code généré selon certains critères à définir selon le langage utilisé et le modèle machine.

Modèle machine

Contient les informations de type : nombre de registres, nombre de mémoires cache (L1, L2, L3) et leur taille, taille de la mémoire RAM, type d'architecture (32 ou 64 bits), etc..

Une fusion très importante des boucles :

- réduit immédiatement le surcoût dû au fonctionnement des boucles ;
- pourrait empêcher un fonctionnement optimal de la hiérarchie mémoire.

Est-il possible d'exploiter cette génération automatique pour du calcul haute performance ?

Sommaire

Notions de base en parallélisme

Système à *mémoire partagée*

Désigne un système composé d'une mémoire accédée par différentes unités de traitement au sein d'un même ordinateur.

Thread

Est un processus.
Les sous-tâches d'un programme parallèle sont souvent appelées threads.

Data race condition

Les *threads* accèdent (lecture/écriture) à des variables qu'ils partagent entre eux, ce qui peut causer un phénomène de corruption de données.

Speedup

L'accélération est le rapport entre les temps d'exécution des programmes séquentiel et parallèle. Loi théorique *speedup* maximum : Amdahl.

Extraction automatique de parallélisme

Objectif :
générer automatiquement un code parallèle
pour architectures à mémoire partagée.

Mais. . .
le code généré par YAO dépend du modèle numérique.

- Tous les codes générés ont la même structure ;
- le *GDR* nous donne les dépendances.

Extraction automatique de parallélisme

Objectif :
générer automatiquement un code parallèle
pour architectures à mémoire partagée.

Mais. . .
le code généré par YAO dépend du modèle numérique.

- Tous les codes générés ont la même structure ;
- le *GDR* nous donne les dépendances.

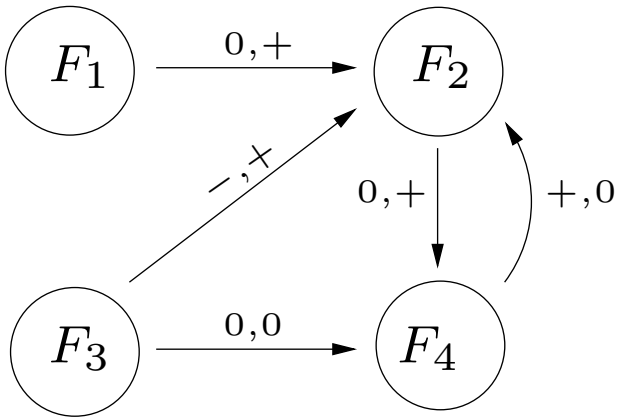
Extraction automatique de parallélisme

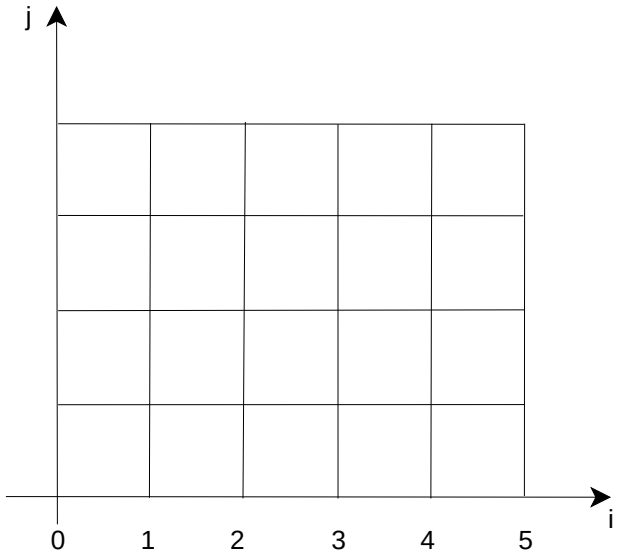
Objectif :
générer automatiquement un code parallèle
pour architectures à mémoire partagée.

Mais. . .
le code généré par YAO dépend du modèle numérique.

- Tous les codes générés ont la même structure ;
- le *GDR* nous donne les dépendances.

Décomposition de domaine



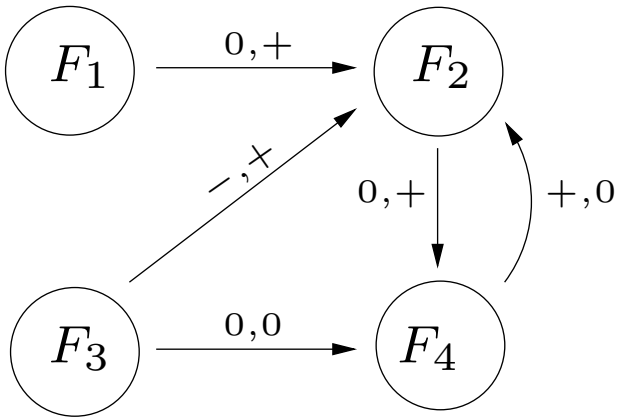


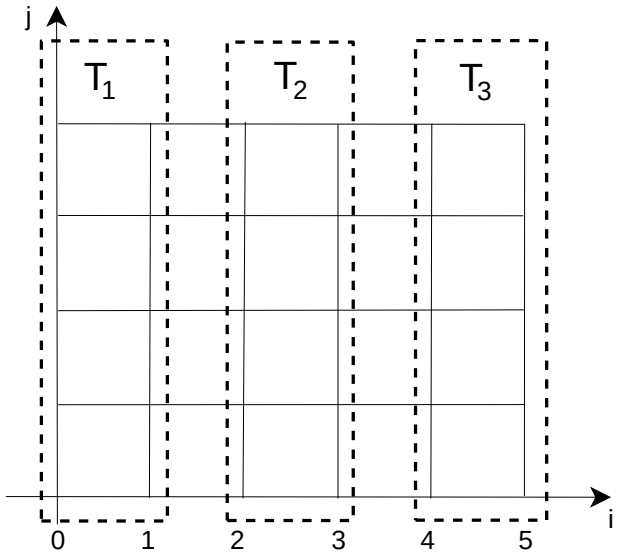
L'espace est partitionné logiquement : décomposition de domaine.
Chaque thread T_i exécute la même tâche sur son propre sous-espace.

Problèmes : *race conditions*.

Objectif : maximiser le nombre de *fonctions de base* exécutées en parallèle.

Décomposition de domaine



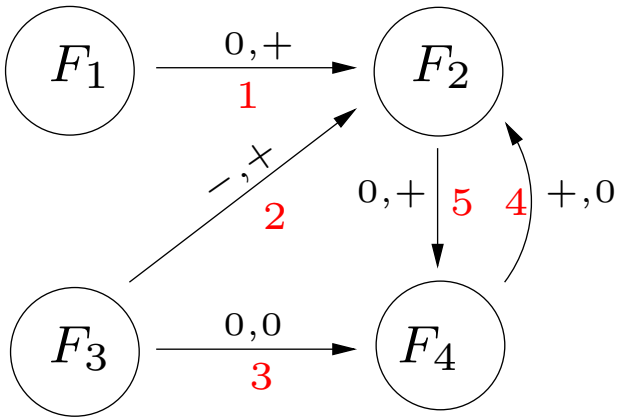


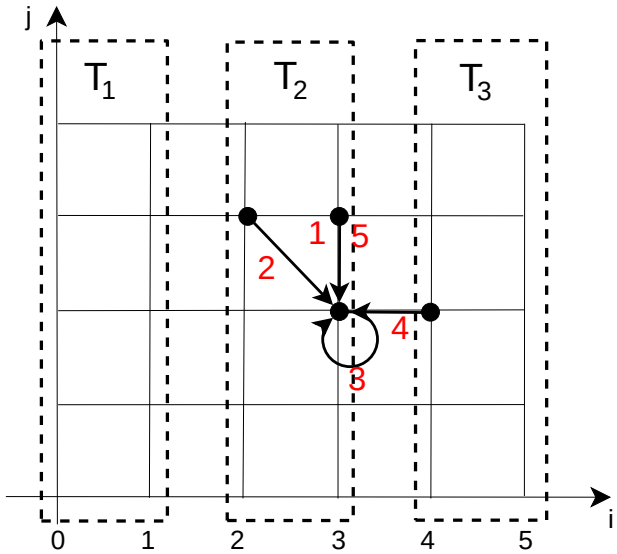
L'espace est partitionné logiquement : décomposition de domaine.
Chaque thread T_i exécute la même tâche sur son propre sous-espace.

Problèmes : *race conditions*.

Objectif : maximiser le nombre de *fonctions de base* exécutées en parallèle.

Décomposition de domaine



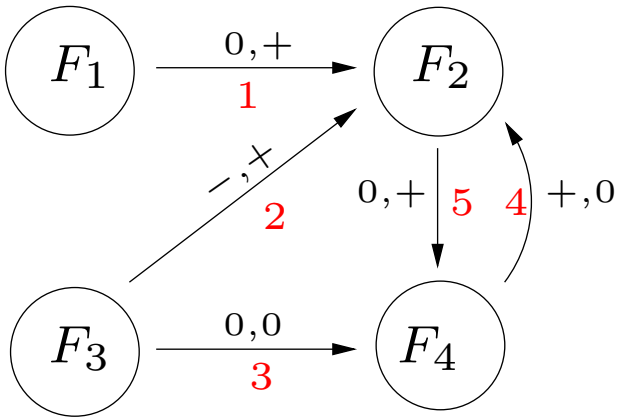


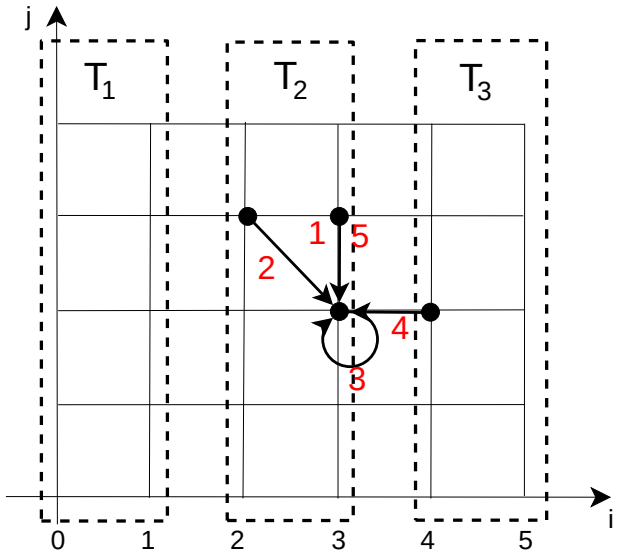
L'espace est partitionné logiquement : décomposition de domaine.
Chaque thread T_i exécute la même tâche sur son propre sous-espace.

Problèmes : *race conditions*.

Objectif : maximiser le nombre de *fonctions de base* exécutées en parallèle.

Décomposition de domaine



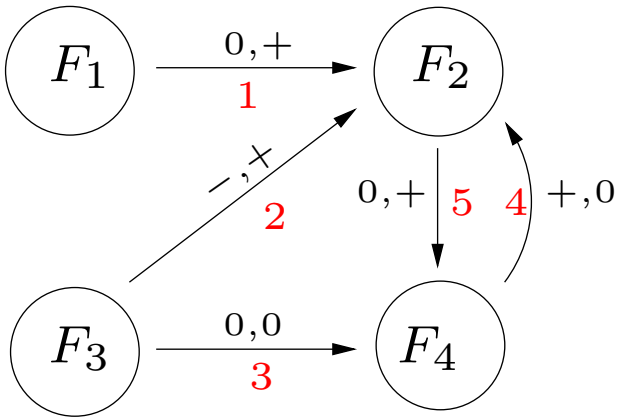


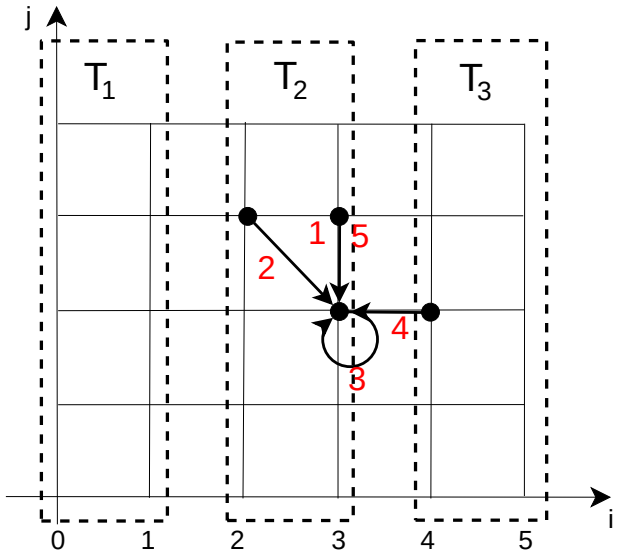
L'espace est partitionné logiquement : décomposition de domaine.
Chaque thread T_i exécute la même tâche sur son propre sous-espace.

Problèmes : *race conditions*.

Objectif : maximiser le nombre de *fonctions de base* exécutées en parallèle.

Décomposition de domaine





L'espace est partitionné logiquement : décomposition de domaine.
Chaque thread T_i exécute la même tâche sur son propre sous-espace.

Problèmes : *race conditions*.

Objectif : maximiser le nombre de *fonctions de base* exécutées en parallèle.

Synchronisation

CFC parallélisable

Pour chaque CFC X du graphe GDR on considère tous les arcs qui ont pour extrémités deux fonctions de base faisant partie de X. S'il existe au moins un arc avec signe $d_l \neq 0$ on considère que X n'est pas parallélisable selon l'axe l.

Un bloc, caractérisé par une boucle extérieure l, est parallélisable si tous les d_l sont nuls.

Cela revient à considérer la liste de boucles possibles définie lors de la génération des directives *order*. Si cette liste ne contient pas de couple $(l, 0)$ la CFC ne sera pas parallélisable.

Synchronisation

CFC parallélisable

Pour chaque CFC X du graphe *GDR* on considère tous les arcs qui ont pour extrémités deux fonctions de base faisant partie de X. S'il existe au moins un arc avec signe $d_l \neq 0$ on considère que X n'est pas parallélisable selon l'axe l.

Un bloc, caractérisé par une boucle extérieure l, est parallélisable si tous les d_l sont nuls.

Cela revient à considérer la liste de boucles possibles définie lors de la génération des directives *order*. Si cette liste ne contient pas de couple $(l, 0)$ la CFC ne sera pas parallélisable.

Algorithme de parallélisation

- Chercher les CFC X_i ;
- pour chaque X_i :
  - déterminer la liste L_i ;
  - chercher un axe l_i dans la liste L_i pour lequel les distances $d_{l_i} = 0$;
  - s'il en existe au moins un, marquer X_i comme parallélisable. Sinon marquer X_i comme non parallélisable.

Cet algorithme permet d'extraire le plus haut degré de parallélisme.

Cette forte décomposition n'est pas la meilleure solution en terme de performance.

Algorithme de parallélisation

- Chercher les CFC X_i ;
- pour chaque X_i :
  - déterminer la liste L_i ;
  - chercher un axe l_i dans la liste L_i pour lequel les distances $d_{l_i} = 0$;
  - s'il en existe au moins un, marquer X_i comme parallélisable. Sinon marquer X_i comme non parallélisable.

Cet algorithme permet d'extraire le plus haut degré de parallélisme.

Cette forte décomposition n'est pas la meilleure solution en terme de performance.

Algorithme de parallélisation

- Chercher les CFC X_i ;
- pour chaque X_i :
  - déterminer la liste L_i ;
  - chercher un axe l_i dans la liste L_i pour lequel les distances $d_{l_i} = 0$;
  - s'il en existe au moins un, marquer X_i comme parallélisable. Sinon marquer X_i comme non parallélisable.

Cet algorithme permet d'extraire le plus haut degré de parallélisme.

Cette forte décomposition n'est pas la meilleure solution en terme de performance.

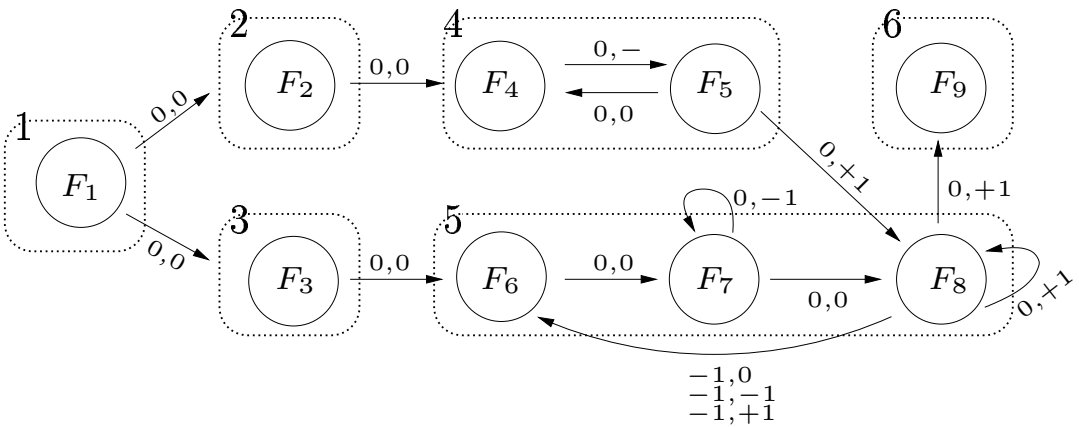
Fusion dans la parallélisation automatique

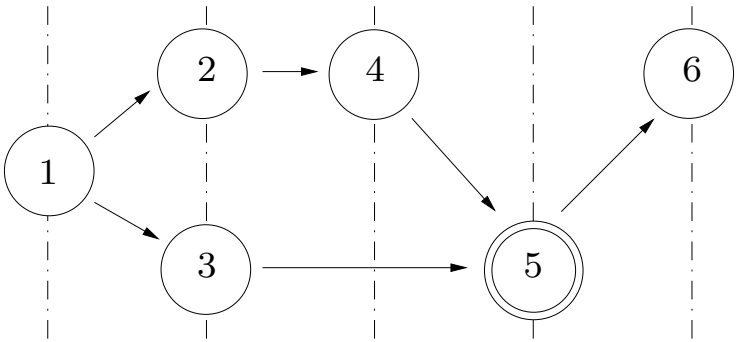
La fusion dans le cadre de la parallélisation se fait simultanément à la fusion de la génération des directives *order*.

Les principes de la fusion dans la génération de directives *order* restent valables. On ajoute des contraintes supplémentaires dans l'analyse du graphe réduit :

- deux sommets non parallélisables peuvent être fusionnés ;
- deux sommets parallélisables peuvent être fusionnés si le sommet résultant est parallélisable ;
- deux sommets parallélisable et non parallélisable ne sont jamais fusionnés.

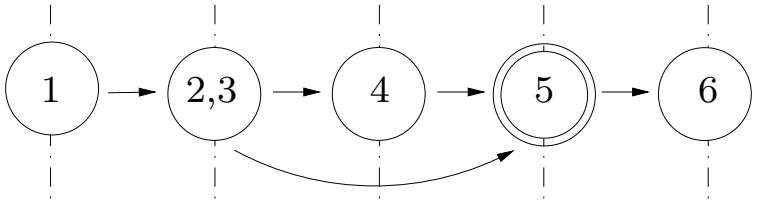
Exemple de l'acoustique marine I

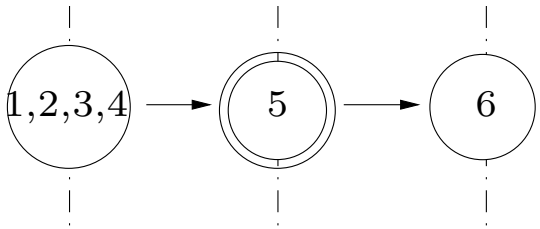




$$L_4 = ((i,-),(i,+),(i,0),(j,-))$$
$$L_5 = ((i,-))$$

Exemple de l'acoustique marine II

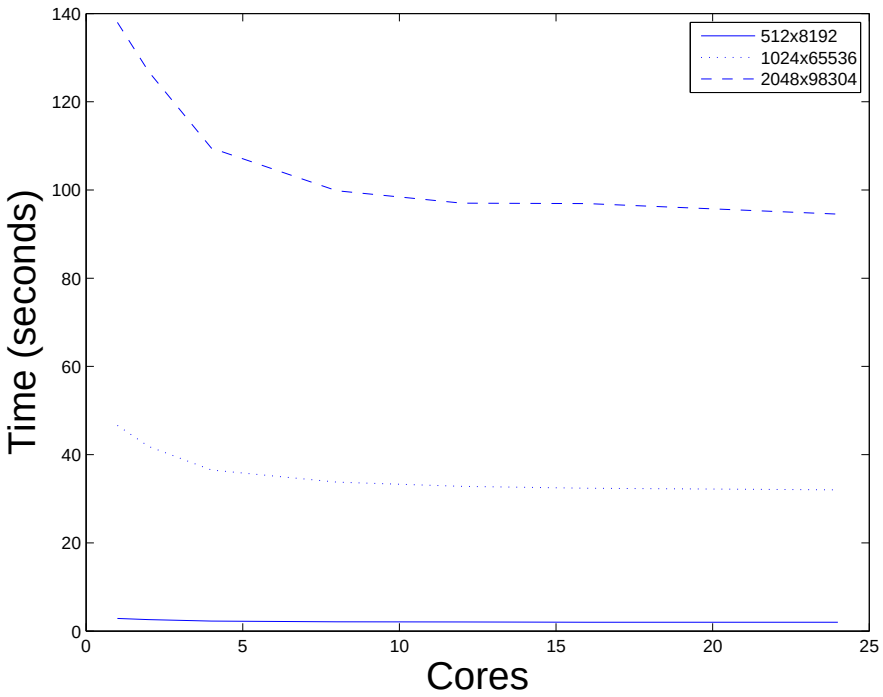


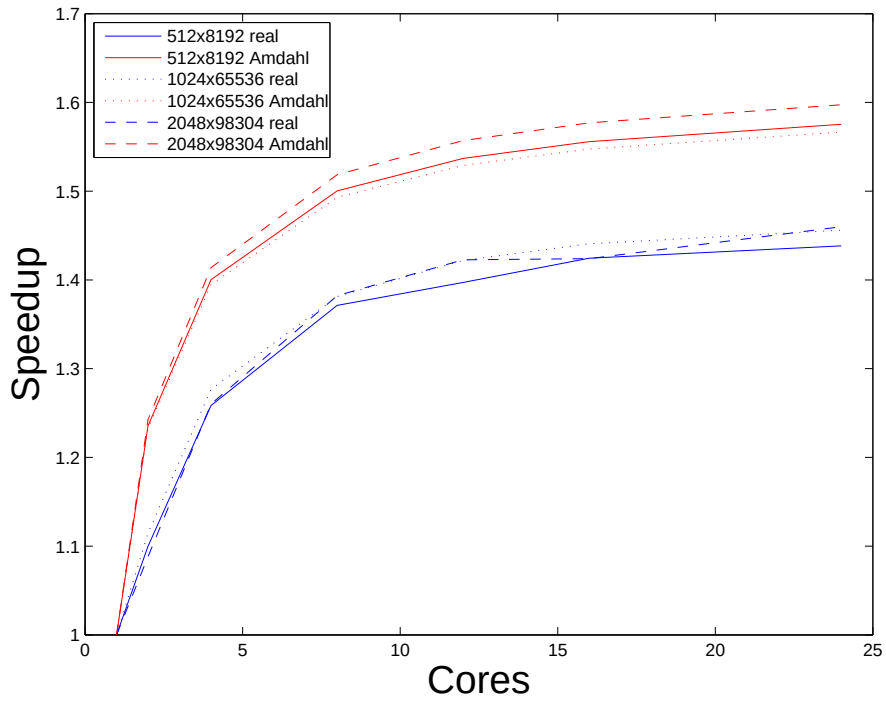


$$L_{1-4} = ((i,-),(i,+),(i,0),(j,-))$$
$$L_5 = ((i,-))$$

Performance acoustique marine

3 espaces 2D de tailles : 512 x 8192, 1024 x 65536, 2048 x 98304.



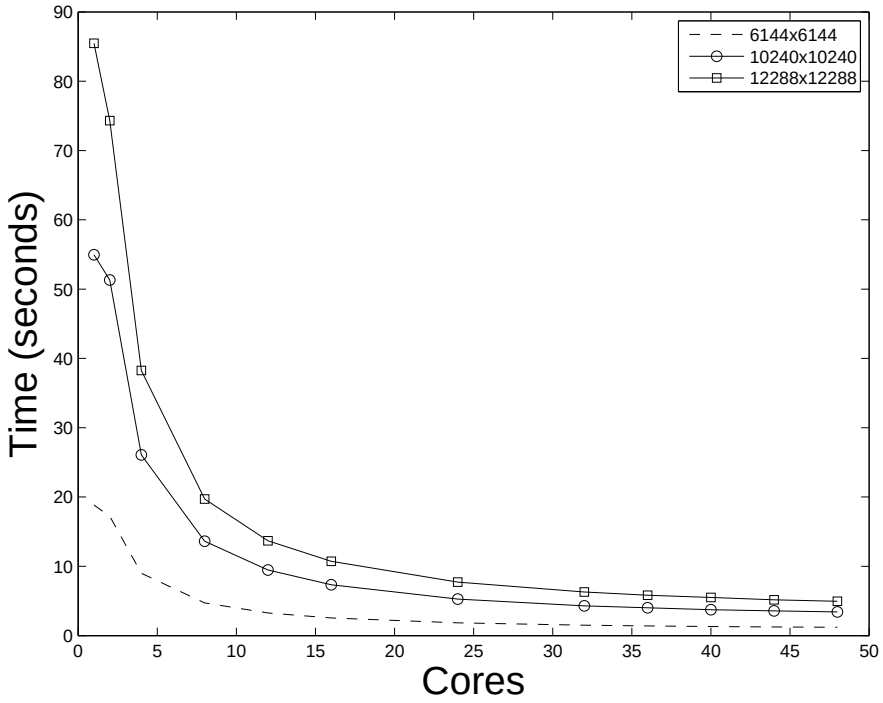


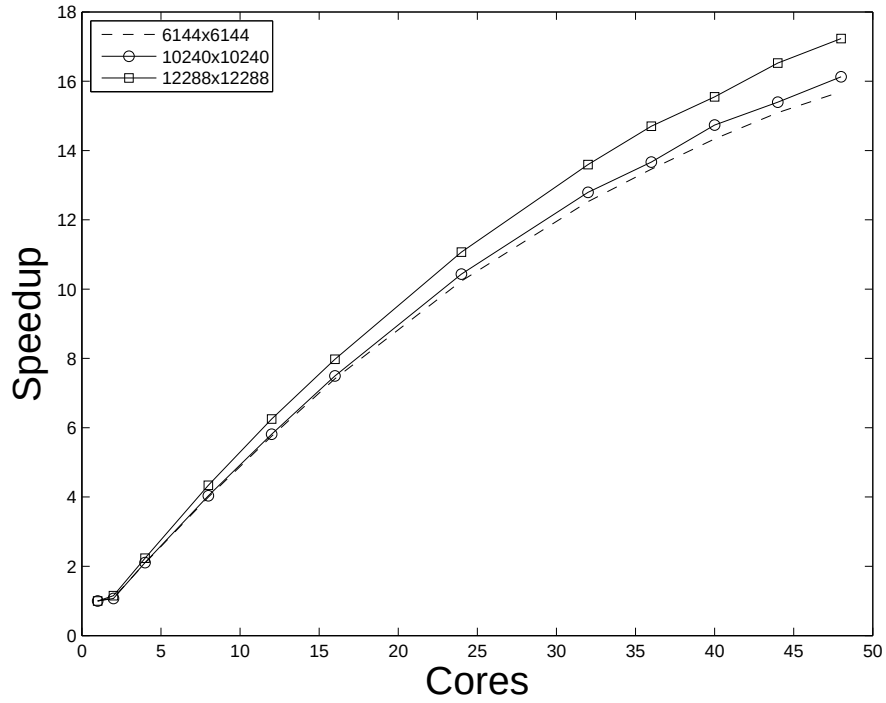
Procs : 4 AMD Opteron 2,3 GHz (6176 SE) dodéca-cœurs
RAM : 256 Go
Caches L1 : 12 x 128 ko, L2 : 12 x 512 ko, L3 : 2 x 6 Mo

speedup théorique maximum
speedup réél

Performances Shallow-water

3 espaces 2D de tailles : 6144 x 6144, 10240 x 10240, 12288 x 12288.





Procs : 4 AMD Opteron 2,3 GHz (6176 SE) dodéca-cœurs
RAM : 256 Go
Caches L1 : 12 x 128 ko, L2 : 12 x 512 ko, L3 : 2 x 6 Mo

Pas de *speedup* théorique maximum

Sommaire

Conclusion

Les simulations ont montré l'efficacité en terme de performance.

Les algorithmes proposés donnent la possibilité d'optimiser le code en suivant des critères donnés, en fonction du langage de programmation et du modèle machine.

Le travail de reformalisation, de synthèse et de développement algorithmique d'automatisation de tâches a permis :

- de donner des réponses immédiates et opérationnelles (YAO est sous licence CeCILL) ;
- de faire le lien avec des domaines actuels de recherche en informatique.

Conclusion

Les simulations ont montré l'efficacité en terme de performance.

Les algorithmes proposés donnent la possibilité d'optimiser le code en suivant des critères donnés, en fonction du langage de programmation et du modèle machine.

Le travail de reformalisation, de synthèse et de développement algorithmique d'automatisation de tâches a permis :

- de donner des réponses immédiates et opérationnelles (YAO est sous licence CeCILL) ;
- de faire le lien avec des domaines actuels de recherche en informatique.

Méthodes avancées en optimisation automatique

La parallélisation correspond au cas où les espaces sont représentés par des parallélépipèdes rectangles (dans le cas d'espaces 3D) définis par des contraintes du type $1 \leq i \leq N_i$.

Ce travail doit être étendu en se basant sur des *modèles polyédriques* qui considèrent l'espace de calcul sous la forme de polyèdre convexe (défini par des inéquations linéaires).

Envisager le passage à l'échelle des architectures parallèles à mémoire distribuée.

Méthodes avancées en optimisation automatique

La parallélisation correspond au cas où les espaces sont représentés par des parallélépipèdes rectangles (dans le cas d'espaces 3D) définis par des contraintes du type $1 \leq i \leq N_i$.

Ce travail doit être étendu en se basant sur des *modèles polyédriques* qui considèrent l'espace de calcul sous la forme de polyèdre convexe (défini par des inéquations linéaires).

Envisager le passage à l'échelle des architectures parallèles à mémoire distribuée.

Méthodes avancées en optimisation automatique

La parallélisation correspond au cas où les espaces sont représentés par des parallélépipèdes rectangles (dans le cas d'espaces 3D) définis par des contraintes du type $1 \leq i \leq N_i$.

Ce travail doit être étendu en se basant sur des *modèles polyédriques* qui considèrent l'espace de calcul sous la forme de polyèdre convexe (défini par des inéquations linéaires).

Envisager le passage à l'échelle des architectures parallèles à mémoire distribuée.

Evolution à court terme

Formalisation

- Introduction des opérateurs entre espaces différents afin de pouvoir traiter le couplage de modèle et les problèmes de changement d'échelle de discrétisation ;
- reformalisation des *ctin* entre deux espaces de dimensions différentes.

Opérationnel

- Introduction de technique dite du *checkpointing* et sa gestion automatique par YAO pour le traitement d'applications de taille importante ;
- introduction de la bibliothèque CADNA pour la précision numérique.

YAO s'étend aussi à la seule utilisation du modèle direct.

Evolution à court terme

Formalisation

- Introduction des opérateurs entre espaces différents afin de pouvoir traiter le couplage de modèle et les problèmes de changement d'échelle de discrétisation ;
- reformalisation des *ctin* entre deux espaces de dimensions différentes.

Opérationnel

- Introduction de technique dite du *checkpointing* et sa gestion automatique par YAO pour le traitement d'applications de taille importante ;
- introduction de la bibliothèque CADNA pour la précision numérique.

YAO s'étend aussi à la seule utilisation du modèle direct.

Evolution à court terme

Formalisation

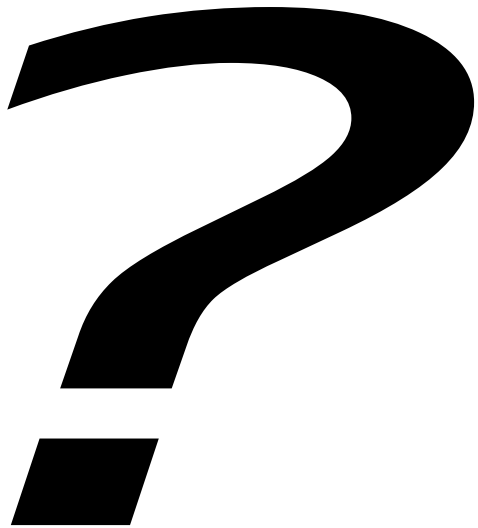
- Introduction des opérateurs entre espaces différents afin de pouvoir traiter le couplage de modèle et les problèmes de changement d'échelle de discrétisation ;
- reformalisation des *ctin* entre deux espaces de dimensions différentes.

Opérationnel

- Introduction de technique dite du *checkpointing* et sa gestion automatique par YAO pour le traitement d'applications de taille importante ;
- introduction de la bibliothèque CADNA pour la précision numérique.

YAO s'étend aussi à la seule utilisation du modèle direct.

Questions



YAO home page : *http://www.locean-ipsl.upmc.fr/∼yao*
email : *lnalod@locean-ipsl.upmc.fr*