Quelques tendances en Informatique du Calcul Scientifique (1)

T. Dumont V. Louvet.

Journées Mathrice, Lille, Oct. 2004.

Plan

L’ objet : modèles approchés et calculables

Equations rendues calculables par dicrétisation.

L’ objet : modèles approchés et calculables

Equations rendues calculables par dicrétisation.

Equations aux dérivées partielles.

L' objet : modèles approchés et calculables

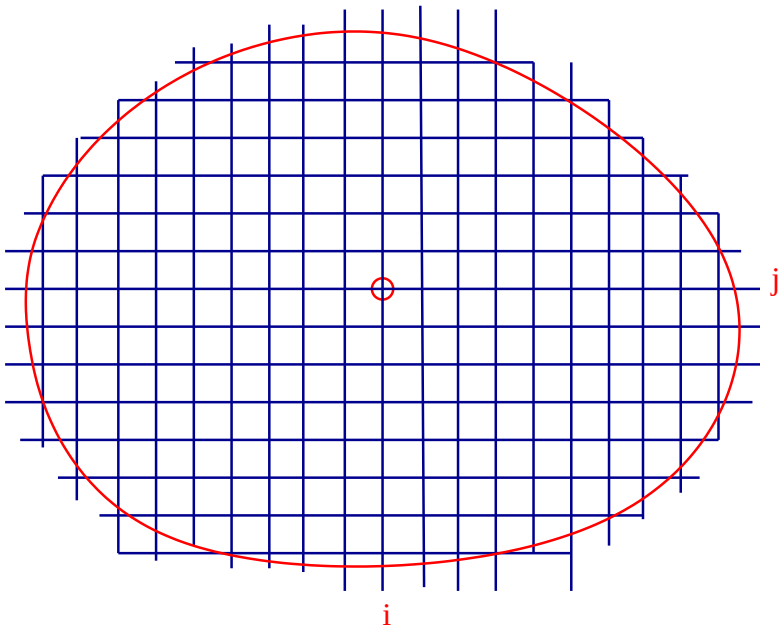
Equations rendues calculables par dicrétisation.

Equations aux dérivées partielles.
Exemple (équation de la chaleur) : $T(t;x,y,z)$?

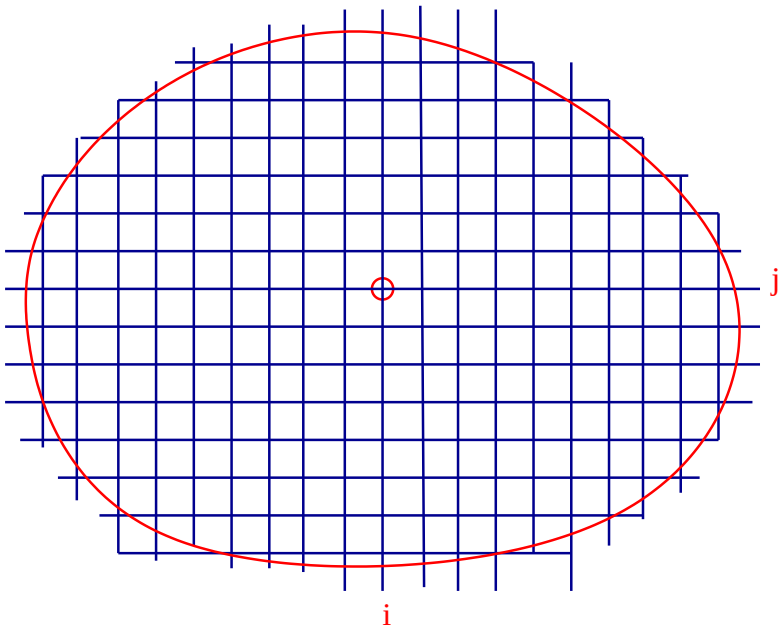
$$\frac{dT}{dt} - \Delta\, T = 0,$$

$$\Delta\, T = \frac{\partial^2\, T}{\partial\, x^2} + \frac{\partial^2\, T}{\partial\, y^2} + \frac{\partial^2\, T}{\partial\, z^2}.$$

exemple :

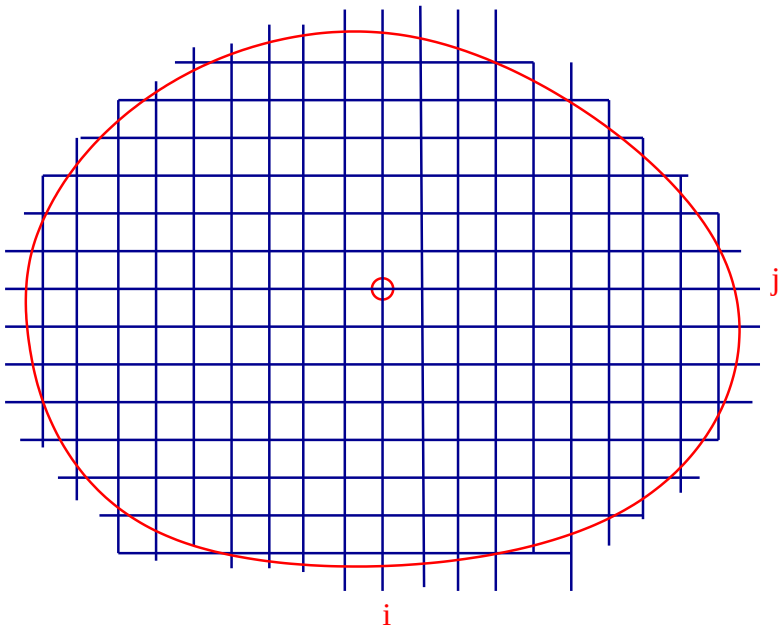


exemple :



$$\frac{\partial T}{\partial x} \longrightarrow \frac{T_{i,j} - T_{(i-1),j}}{h_x},$$

exemple :



$$\frac{\partial\ T}{\partial\ x} \longrightarrow \frac{T_{i,j} - T_{(i-1),j}}{h_x}, \frac{\partial^2\ T}{\partial\ x^2} \longrightarrow \frac{T_{(i+1),j} - 2\ T_{i,j} + T_{(i-1),j}}{h_x^2}.$$

exemple :

On se ramène donc à un nombre fini d'inconnues.

- Différences finies,

exemple :

On se ramène donc à un nombre fini d'inconnues.

- Différences finies,
- Polynômes,

exemple :

On se ramène donc à un nombre fini d'inconnues.

- Différences finies,
- Polynômes,
- Polynômes par morceaux (éléments finis)

exemple :

On se ramène donc à un nombre fini d'inconnues.

- Différences finies,
- Polynômes,
- Polynômes par morceaux (éléments finis)
- etc.

Objets : grands tableaux

- Chaleur : n^d inconnues ($d = 2, 3$).

Objets : grands tableaux

- Chaleur : n^d inconnues ($d = 2, 3$).
- Rayleigh–Bénard :
  $(T, \vec{U}, P)$: $(d+2)\, n^d$ inconnues...

Objets : grands tableaux

- Chaleur : n^d inconnues ($d = 2, 3$).
- Rayleigh–Bénard :
  $(T, \vec{U}, P)$: $(d+2)\, n^d$ inconnues...
- Mmmmh...

Objets : grands tableaux

- Chaleur : n^d inconnues ($d = 2, 3$).
- Rayleigh–Bénard :
  $(T, \vec{U}, P)$: $(d+2)\, n^d$ inconnues...
- Mmmmh...

Objets : grands tableaux

- Chaleur : n^d inconnues ($d = 2, 3$).
- Rayleigh–Bénard :
  $(T, \vec{U}, P)$: $(d+2)\, n^d$ inconnues...
- Mmmmh... $n = 100,\ d = 3$...

Objets : grands tableaux

- Chaleur : n^d inconnues ($d = 2, 3$).
- Rayleigh–Bénard :
  $(T, \vec{U}, P)$: $(d+2)\, n^d$ inconnues...
- Mmmmh... $n = 100,\ d = 3$... ça fait, heu...

Objets : grands tableaux

- Chaleur : n^d inconnues ($d = 2, 3$).
- Rayleigh–Bénard :
  $(T, \vec{U}, P)$: $(d+2)\, n^d$ inconnues...
- Mmmmh... $n = 100,\ d = 3$... ça fait, heu...

5. 10^6 inconnues.

Objets : grands tableaux

- Chaleur : n^d inconnues ($d = 2, 3$).
- Rayleigh–Bénard :
  $(T, \vec{U}, P)$: $(d+2)\, n^d$ inconnues...
- Mmmmh... $n = 100,\ d = 3$... ça fait, heu...

  5. 10^6 inconnues.
- Mais les tableaux sont souvents creux.

tableaux= matrices.

- Grands systèmes linéaires à résoudre,

tableaux= matrices.

- Grands systèmes linéaires à résoudre,
- En très grand nombre.

tableaux= matrices.

- Grands systèmes linéaires à résoudre,
- En très grand nombre.
- Algorithmes adaptés.

tableaux= matrices.

- Grands systèmes linéaires à résoudre,
- En très grand nombre.
- Algorithmes adaptés.
- Non linéaire => Linéaire.

tableaux= matrices.

- Grands systèmes linéaires à résoudre,
- En très grand nombre.
- Algorithmes adaptés.
- Non linéaire => Linéaire.

tableaux= matrices.

- Grands systèmes linéaires à résoudre,
- En très grand nombre.
- Algorithmes adaptés.
- Non linéaire => Linéaire.

un grand nombre de boucles imbriquées.

flottants.

- Amplification des erreurs.

flottants.

- Amplification des erreurs.
- flottant 32 bits= 6 chiffres significatifs

flottants.

- Amplification des erreurs.
- flottant 32 bits= 6 chiffres significatifs
- flottant 64 bits= 16 chiffres significatifs

flottants.

- Amplification des erreurs.
- flottant 32 bits= 6 chiffres significatifs
- flottant 64 bits= 16 chiffres significatifs

flottants.

- Amplification des erreurs.
- flottant 32 bits= 6 chiffres significatifs
- flottant 64 bits= 16 chiffres significatifs

flottant 64 bits obligatoire.

optimisations et accès mémoire.

Durée des calculs : de ε secondes à plusieurs semaines

optimisations et accès mémoire.

Durée des calculs : de ε secondes à plusieurs semaines

- optimiser

optimisations et accès mémoire.

Durée des calculs : de ε secondes à plusieurs semaines

- optimiser

optimisations et accès mémoire.

Durée des calculs : de ε secondes à plusieurs semaines

- optimiser au moins les boucles internes

optimisations et accès mémoire.

Durée des calculs : de ε secondes à plusieurs semaines

- optimiser au moins les boucles internes
- optimiser les accès mémoire,

optimisations et accès mémoire.

Durée des calculs : de ε secondes à plusieurs semaines

- optimiser au moins les boucles internes
- optimiser les accès mémoire,
- taille mémoire.

mémoire

temps d'accès :

Registres	2	ns
L1 On–Chip	4	ns
L2 On–Chip	5	ns
L3 Off–Chip	30	ns
Mémoire	220	ns

(Alpha 21164.)

mémoire

- minimiser les défauts de cache

mémoire

- minimiser les défauts de cache

mémoire

- minimiser les défauts de cache : parcourir les tableaux par adresses conjointes.

mémoire

- minimiser les défauts de cache : parcourir les tableaux par adresses conjointes.
- gérer de tableaux de plus de 2 Giga octets.

mémoire

- minimiser les défauts de cache : parcourir les tableaux par adresses conjointes.
- gérer de tableaux de plus de 2 Giga octets.

mémoire

- minimiser les défauts de cache : parcourir les tableaux par adresses conjointes.
- gérer de tableaux de plus de 2 Giga octets.

l'avenir immédiat : machines 64 bits.

Fortran(s) ?

“Je ne sais pas quel sera le langage de l’an 2000, mais il s’appelera Fortran”

S. Cray, années 80

Fortran(s) ?

"Je ne sais pas quel sera le langage de l'an 2000, mais il s'appelera Fortran"

S. Cray, années 80

- Stock gigantesque de bibliothèques de base en Fortran 77,

Fortran(s) ?

"Je ne sais pas quel sera le langage de l'an 2000, mais il s'appelera Fortran"

S. Cray, années 80

- Stock gigantesque de bibliothèques de base en Fortran 77,
- Fortran 77 : souvent le seul langage maîtrisé,

Fortran(s) ?

"Je ne sais pas quel sera le langage de l'an 2000, mais il s'appelera Fortran"

S. Cray, années 80

- Stock gigantesque de bibliothèques de base en Fortran 77,
- Fortran 77 : souvent le seul langage maîtrisé,
- Fortran 9x :

Fortran(s) ?

"Je ne sais pas quel sera le langage de l'an 2000, mais il s'appelera Fortran"

S. Cray, années 80

- Stock gigantesque de bibliothèques de base en Fortran 77,
- Fortran 77 : souvent le seul langage maîtrisé,
- Fortran 9x :
  - pas de compilateurs libres,

Fortran(s) ?

"Je ne sais pas quel sera le langage de l'an 2000, mais il s'appelera Fortran"

S. Cray, années 80

- Stock gigantesque de bibliothèques de base en Fortran 77,
- Fortran 77 : souvent le seul langage maîtrisé,
- Fortran 9x :
  - pas de compilateurs libres,
  - manipulation aisée de tableaux,

Fortran(s) ?

"Je ne sais pas quel sera le langage de l'an 2000, mais il s'appelera Fortran"

S. Cray, années 80

- Stock gigantesque de bibliothèques de base en Fortran 77,
- Fortran 77 : souvent le seul langage maîtrisé,
- Fortran 9x :
  - pas de compilateurs libres,
  - manipulation aisée de tableaux,
  - un peu hybride.

Fortran(s) ?

"Je ne sais pas quel sera le langage de l'an 2000, mais il s'appelera Fortran"

S. Cray, années 80

- Stock gigantesque de bibliothèques de base en Fortran 77,
- Fortran 77 : souvent le seul langage maîtrisé,
- Fortran 9x :
  - pas de compilateurs libres,
  - manipulation aisée de tableaux,
  - un peu hybride.
- Fortran 2000 ?

autres langages ?

- **C :** disqualifié.

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais :

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais :

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais : *virtual functions*,

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais : *virtual functions*, *surcharge des opérateurs* (copies),

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais : *virtual functions*, *surcharge des opérateurs* (copies),
  - mais tout peut devenir bon...

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais : *virtual functions*, *surcharge des opérateurs* (copies),
  - mais tout peut devenir bon...

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais : *virtual functions*, *surcharge des opérateurs* (copies),
  - mais tout peut devenir bon... au prix d'une complexité importante,

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais : *virtual functions*, *surcharge des opérateurs* (copies),
  - mais tout peut devenir bon... au prix d'une complexité importante,
  - généricité (*templates*).

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais : *virtual functions*, *surcharge des opérateurs* (copies),
  - mais tout peut devenir bon... au prix d'une complexité importante,
  - généricité (*templates*).
  - allocation dynamique.

autres langages ?

- **C :** disqualifié.
  - modèle d'accès mémoire,
  - aliasing (peut disparaitre avec la nouvelle norme),
  - trop bas niveau.
- **C++ :**
  - objets, masquer l'accès mémoire,
  - par défaut, tout est mauvais : *virtual functions*, *surcharge des opérateurs* (copies),
  - mais tout peut devenir bon... au prix d'une complexité importante,
  - généricité (*templates*).
  - allocation dynamique.
  - c'est dur !

bibliothèques

Briques de base : on n'utilise pratiquement *que* des bibliothèques libres.

- lapack,

bibliothèques

Briques de base : on n'utilise pratiquement *que* des bibliothèques libres.

- lapack,
- fftpack,

bibliothèques

Briques de base : on n'utilise pratiquement *que* des bibliothèques libres.

- lapack,
- fftpack,
- souvent de vieux programmes (lsode).

bibliothèques auto–adaptatives

résoudre dynamiquement le problème de l'accès mémoire.

bibliothèques auto–adaptatives

résoudre dynamiquement le problème de l'accès mémoire.

- fftw *(Fastest Fourier transform of the west)*,

bibliothèques auto–adaptatives

résoudre dynamiquement le problème de l'accès mémoire.

- fftw *(Fastest Fourier transform of the west)*,
- atlas *(blas)*,

bibliothèques auto–adaptatives

résoudre dynamiquement le problème de l'accès mémoire.

- fftw *(Fastest Fourier transform of the west)*,
- atlas *(blas)*,
- et sûrement d'autres...

bibliothèques auto–adaptatives

résoudre dynamiquement le problème de l'accès mémoire.

- fftw *(Fastest Fourier transform of the west)*,
- atlas *(blas)*,
- et sûrement d'autres...

bibliothèques auto–adaptatives

résoudre dynamiquement le problème de l'accès mémoire.

- fftw *(Fastest Fourier transform of the west)*,
- atlas *(blas)*,
- et sûrement d'autres...

Linux, bien sur.

ce qu'on n'a pas besoin d'optimiser

les boucles externes n'ont pas à être optimisées.

ce qu'on n'a pas besoin d'optimiser

les boucles externes n'ont pas à être optimisées.

Python.

- langage interprèté,

ce qu'on n'a pas besoin d'optimiser

les boucles externes n'ont pas à être optimisées.

Python.

- langage interprèté,
- à objets

ce qu'on n'a pas besoin d'optimiser

les boucles externes n'ont pas à être optimisées.

Python.

- langage interprèté,
- à objets
- extensible,

ce qu'on n'a pas besoin d'optimiser

les boucles externes n'ont pas à être optimisées.

Python.

- langage interprèté,
- à objets
- extensible,
- simple.

Python

- ajout d'un langage de commande (de script) à un code existant.
  Exemple : *Aster*.

Python

- ajout d'un langage de commande (de script) à un code existant.
  Exemple : *Aster*.
- encapsuler des bibliothèques numériques : faites votre propre `Matlab` ;

Python

- ajout d'un langage de commande (de script) à un code existant.
  Exemple : *Aster*.
- encapsuler des bibliothèques numériques : faites votre propre `Matlab` ;
- faire communiquer des codes entre eux.

XML

Besoin d'échanger des données structurées.

XML

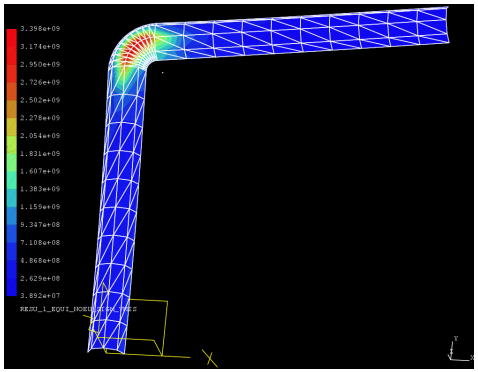
Besoin d'échanger des données structurées.

Exemple :
un code de calcul fournit un résultat :

XML

Besoin d'échanger des données structurées.

Exemple :
un code de calcul fournit un résultat :



(Code *Aster (EdF)*)

XML

1. un graphe ;

XML

1. un graphe ;
2. chaque nœud du graphe a des coordonnées ;

XML

1. un graphe ;
2. chaque nœud du graphe a des coordonnées ;
3. à chaque nœud sont associées les valeurs des inconnues.

XML

1. un graphe ;
2. chaque nœud du graphe a des coordonnées ;
3. à chaque nœud sont associées les valeurs des inconnues.

XML

1. un graphe ;
2. chaque nœud du graphe a des coordonnées ;
3. à chaque nœud sont associées les valeurs des inconnues.

XML !

Exemple

XML + C++ + Python

- VTK *VisualToolkit* en C++.

Exemple

XML + C++ + Python

- VTK *VisualToolkit* en C++.
- VTK lit du XML.

Exemple

XML + C++ + Python

- VTK *VisualToolkit* en C++.
- VTK lit du XML.
- programmation en C++ : lent et pénible

Exemple

XML + C++ + Python

- VTK *VisualToolkit* en C++.
- VTK lit du XML.
- programmation en C++ : lent et pénible

Exemple

XML + C++ + Python

- VTK *VisualToolkit* en C++.
- VTK lit du XML.
- programmation en C++ : lent et pénible : Python !

Exemple

XML + C++ + Python

- VTK *VisualToolkit* en C++.
- VTK lit du XML.
- programmation en C++ : lent et pénible : Python !
  VTK + Python +TkInter = Mayavi.

Exemple

XML + C++ + Python

- VTK *VisualToolkit* en C++.
- VTK lit du XML.
- programmation en C++ : lent et pénible : Python !
  VTK + Python +TkInter = Mayavi.
- Mayavi peut être appelé dans un programme Python..

Exemple

XML + C++ + Python

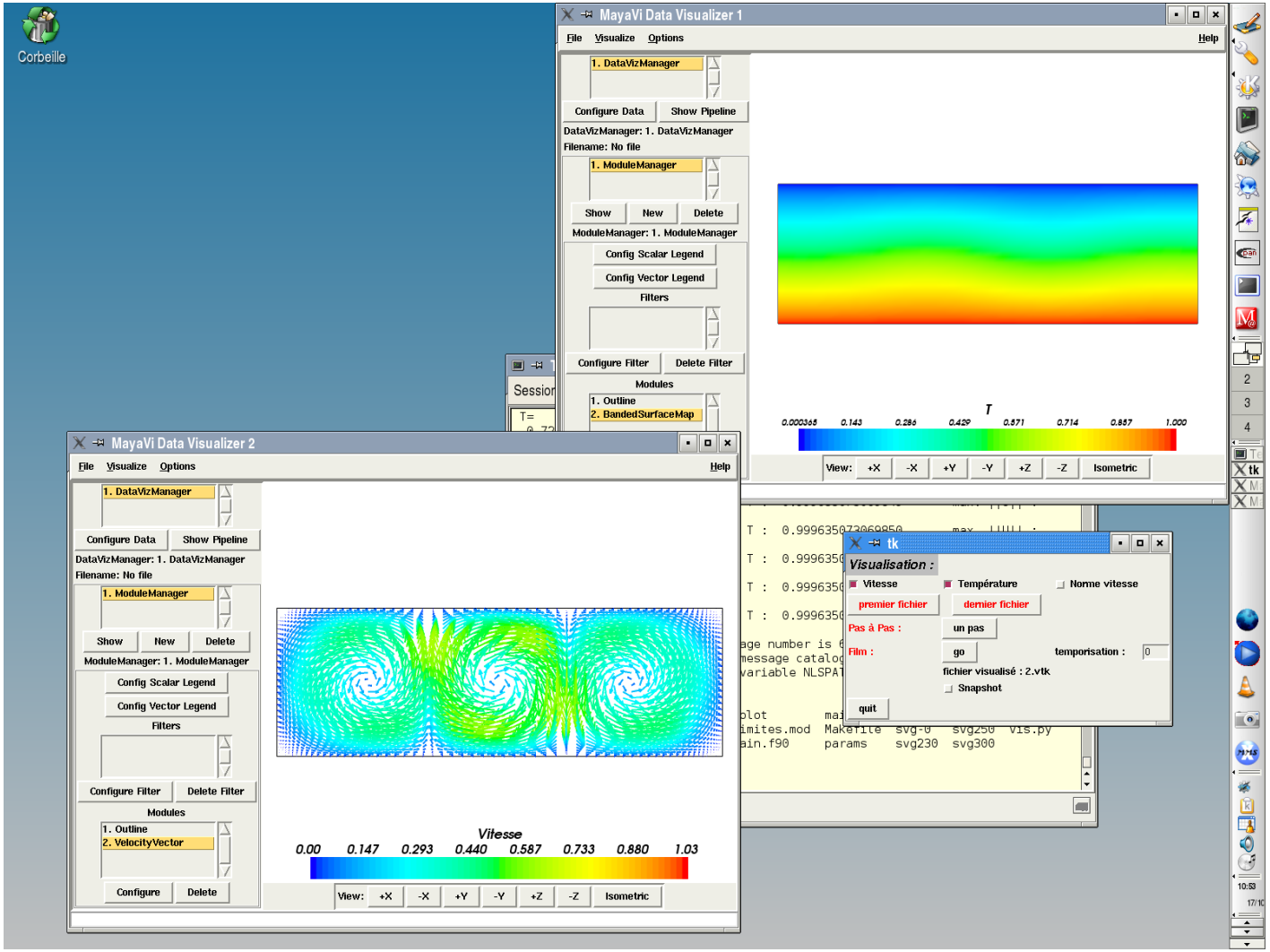
- VTK *VisualToolkit* en C++.
- VTK lit du XML.
- programmation en C++ : lent et pénible : Python !
  VTK + Python +TkInter = Mayavi.
- Mayavi peut être appelé dans un programme Python..

Exemple

XML + C++ + Python

- VTK *VisualToolkit* en C++.
- VTK lit du XML.
- programmation en C++ : lent et pénible : Python !
  VTK + Python +TkInter = Mayavi.
- Mayavi peut être appelé dans un programme Python..
  création de “films”.

Exemple


Corbeille
MayaVi Data Visualizer 1
File Visualize Options Help
1. DataVizManager
Configure Data
Show Pipeline
DataVizManager: 1. DataVizManager
Filename: No file
1. ModuleManager
Show
New
Delete
ModuleManager: 1. ModuleManager
Config Scalar Legend
Config Vector Legend
Filters
Configure Filter
Delete Filter
Modules
1. Outline
2. BandedSurfaceMap
T
0.000365 0.143 0.286 0.429 0.571 0.714 0.857 1.000
View: +X -X +Y -Y +Z -Z Isometric
MayaVi Data Visualizer 2
File Visualize Options Help
1. DataVizManager
Configure Data
Show Pipeline
DataVizManager: 1. DataVizManager
Filename: No file
1. ModuleManager
Show
New
Delete
ModuleManager: 1. ModuleManager
Config Scalar Legend
Config Vector Legend
Filters
Configure Filter
Delete Filter
Modules
1. Outline
2. VelocityVector
Configure
Delete
Vitesse
0.00 0.147 0.293 0.440 0.587 0.733 0.880 1.03
View: +X -X +Y -Y +Z -Z Isometric
tk
Visualisation :
Vitesse
Température
Norme vitesse
premier fichier
dernier fichier
Pas à Pas :
un pas
Film :
go
temporisation :
fichier visualisé : 2.vtk
Snapshot
quit

Fin...

C’est presque fini...

Fin...

C’est presque fini...

il reste la biblio

Références livres/web I

K. Dowd et C. Severance,
High Performance Computing.
O'Reilly.

- T. Veldhuizen
  Techniques for Scientific C++
  `http://osl.iu.edu/~tveldhui/papers/`
- Python.
  `http://www.boost.org/libs/python/doc/index.html`
- Mayavi.
  `http://mayavi.sourceforge.net/`