

# Tutoriel de réalisation d'une application **BALTIK** basée sur le logiciel **TRIO\_U**

November 20, 2014

## 1 résumé

Ce tutoriel a comme objectif de décrire la démarche à suivre pour la réalisation d'une application **BALTIK** sur la base du logiciel **TRIO\_U**. Pour ce faire, on choisit à titre d'exercice une équation de convection-diffusion comme problème mathématique à résoudre. Dans cette équation, la vitesse de convection et le coefficient de diffusion sont des données utilisateur.

On décrit en particulier les spécifications des développements à réaliser et on donne quelques recommandations pour résoudre cette équation de convection-diffusion.

Les développements sont ensuite décrits et réalisés dans un environnement proposé par le logiciel **TRIO\_U** appelé application **BALTIK**. La mise en place de cet environnement est également décrit.

Deux tests de vérification des développements et de non-régression sont enfin réalisés en utilisant les outils proposés par le logiciel **TRIO\_U** (`genererCourbes`, `make check_optim`).

## 2 Introduction

Ce tutoriel a comme objectif de décrire la démarche à suivre pour la réalisation d’une application **BALTIK** sur la base du logiciel **TRIO\_U**. Pour ce faire, on choisit à titre d’exercice l’équation de convection-diffusion (1) comme problème mathématique à résoudre:

$$\frac{\partial c}{\partial t} + \nabla \cdot (\vec{V}c) + \nabla \cdot (D\nabla c) = 0 \text{ sur } \Omega = [a, b] \times [c, d] \quad (1)$$

conditions aux limites périodiques

$c(\vec{X}, t)$  : variable d’intérêt (la concentration d’une espèce, la température,...),  
 $\vec{V}(\vec{X}, t)$  : vitesse de convection de la variable  $c$  (donnée utilisateur),  
 $D(\vec{X}, t)$  : coefficient de diffusion de la variable  $c$  (donnée utilisateur),  
 $a, b, c$  et  $d$  : les bornes du domaine de calcul en deux dimensions.

La discrétisation en espace et en temps de l’équation (1) sera réalisée avec les schémas numériques disponibles dans le logiciel **TRIO\_U** [1].

Dans la section 3, nous décrivons les spécifications des développements à réaliser et nous donnons quelques recommandations pour résoudre l’équation (1).

Les développements spécifiés dans la section 3 sont décrits dans la section 4. Ces développements seront réalisés dans un environnement proposé par le logiciel **TRIO\_U** appelé application **BALTIK**. La mise en place de cet environnement sera décrit dans cette même section 4.

La section 5 sera consacrée à la réalisés deux tests de vérification des développements et de non-régression en utilisant les outils proposés par le logiciel **TRIO\_U** (**genererCourbes**, **make check\_optim**).

## 3 Spécification

### 3.1 Notions

- Notion de problème : le rôle d’un problème est la résolution sur un domaine des équations qui composent le problème. Un schéma de discrétisation en temps ainsi qu’un schéma de discrétisation en espace lui sont associés.
- Notion d’équation : le rôle d’une équation est le calcul d’un ou plusieurs champs compte tenu des choix suivants:
  - un schéma de discrétisation en temps,
  - un schéma de discrétisation en espace,

- des conditions aux limites,
- des termes sources et des opérateurs.

Une équation est portée par un problème et possède une référence qui permet de remonter au problème qui la porte.

- Notion d'opérateur : les opérateurs sont des parties d'une équation. Parmi les opérateurs les plus utilisés, on cite les opérateurs de convection et de diffusion.
- Notion de milieu : Description du milieu fluide ou solide qu'on modélise. Le Milieu est associé au problème et aux équations. Parmi les propriétés qui caractérisent le milieu, on a:
  - la masse volumique,
  - la diffusivité,
  - la conductivité,
  - la capacité calorifique,
  - la variation de la masse volumique en fonction de la température (dilatabilité).

Dans notre cas, le problème qu'on souhaite résoudre comprend une seule équation (1). Cette équation est composée de:

- un terme instationnaire  $\frac{\partial c}{\partial t}$ ,
- un opérateur de convection  $\nabla \cdot (\vec{V}c)$ ,
- un opérateur de diffusion  $\nabla \cdot (D\nabla c)$ ,
- un terme source ou puit  $f = 0$ .

### 3.2 Jeu de données - analyse des besoins

Dans le cadre des développements des applications basées sur le logiciel `TRIO_U`, il est recommandé de commencer les spécifications par écrire le jeu de données. La recommandation précédente est liée au fait que les mots clés du jeu données du logiciel `TRIO_U` correspondent à des objets (classes) C++ et donc commencer par le jeu de données sert, d'une part, à se mettre dans la position d'un utilisateur et, d'autre part, à définir les objets (problème, équation(s), ...) à spécifier et à développer.

Nous décrivons ci-dessous les parties de jeu de données spécifiques à notre problème:

- Déclaration du problème

On notera ce nouveau problème `Probleme_Convection_Diffusion`

```
1 Probleme_Convection_Diffusion pb
```

déclaration du problème

- **Déclaration de l'équation de convection-diffusion**

On notera cette nouvelle équation `Convection_Diffusion`. Cette équation est composée de deux opérateurs (convection et diffusion), des conditions initiales et des conditions aux limites

```
1 Convection_Diffusion
2 {
3   diffusion { }
4   convection { montant }
5   conditions_initiales
6   {
7     Concentration champ_fonc_xyz dom 1 (x-0.2)^2+(y-0.2)^2<(0.1)
8     ^2
9   }
10  boundary_conditions
11  {
12    Gauche    periodique
13    Bas       periodique
14  }
15 }
```

déclaration de l'équation

- **Données utilisateur**

Pour l'équation (1), la vitesse de convection  $\vec{V}(\vec{X}, t)$  et le coefficient de diffusion  $D(\vec{X}, t)$  sont donnés par l'utilisateur. Dans les pratiques des développements TRIO\_U, ces paramètres doivent être associés à un objet (équation, problème, ...) du jeu de données. on notera ces paramètres `coefficient_diffusion` et `vitesse_convection`.

```
1 coefficient_diffusion Champ_Uniforme 1 0.001
2 vitesse_convection  Champ_Fonc_txyz dom 2 x*t y*t
```

paramètres d'entrées

- **Post-traitements**

Il nous semble intéressant de post-traiter la variable d'intérêt  $c$ , la vitesse de convection  $\vec{V}(\vec{X}, t)$  et le coefficient de diffusion  $D(\vec{X}, t)$ . Le bloc de post-traitement de ces variables se présente sous la forme suivante:

```

2 Postraitement
3 {
4     format lata
5     champs dt_post 1.e-7
6     {
7         Concentration elem
8         vitesse_convection elem
9         coefficient_diffusion elem
10    }
11 }

```

déclaration du champs à post-traiter

### 3.3 Fonctionnalités TRIO\_U

Il est fortement recommandé d'utiliser au maximum les fonctionnalités existantes dans le logiciel TRIO\_U et d'éviter de dupliquer les lignes de codes. Ceci permet, d'une part, de minimiser le temps de développement de l'application et, d'autre part, de profiter des fonctionnalités couvertes par les tests du logiciel TRIO\_U.

L'identification des fonctionnalités peut se faire via la documentation doxygen du logiciel TRIO\_U. Nous faisons ci-dessous cet exercice d'identification par rapport à nos besoins (3.2).

- **problème**

En analysant le graphe doxygen de la classe `Probleme_base` (cf. figure 1), aucun problème nous parait proche de nos besoins (3.2).

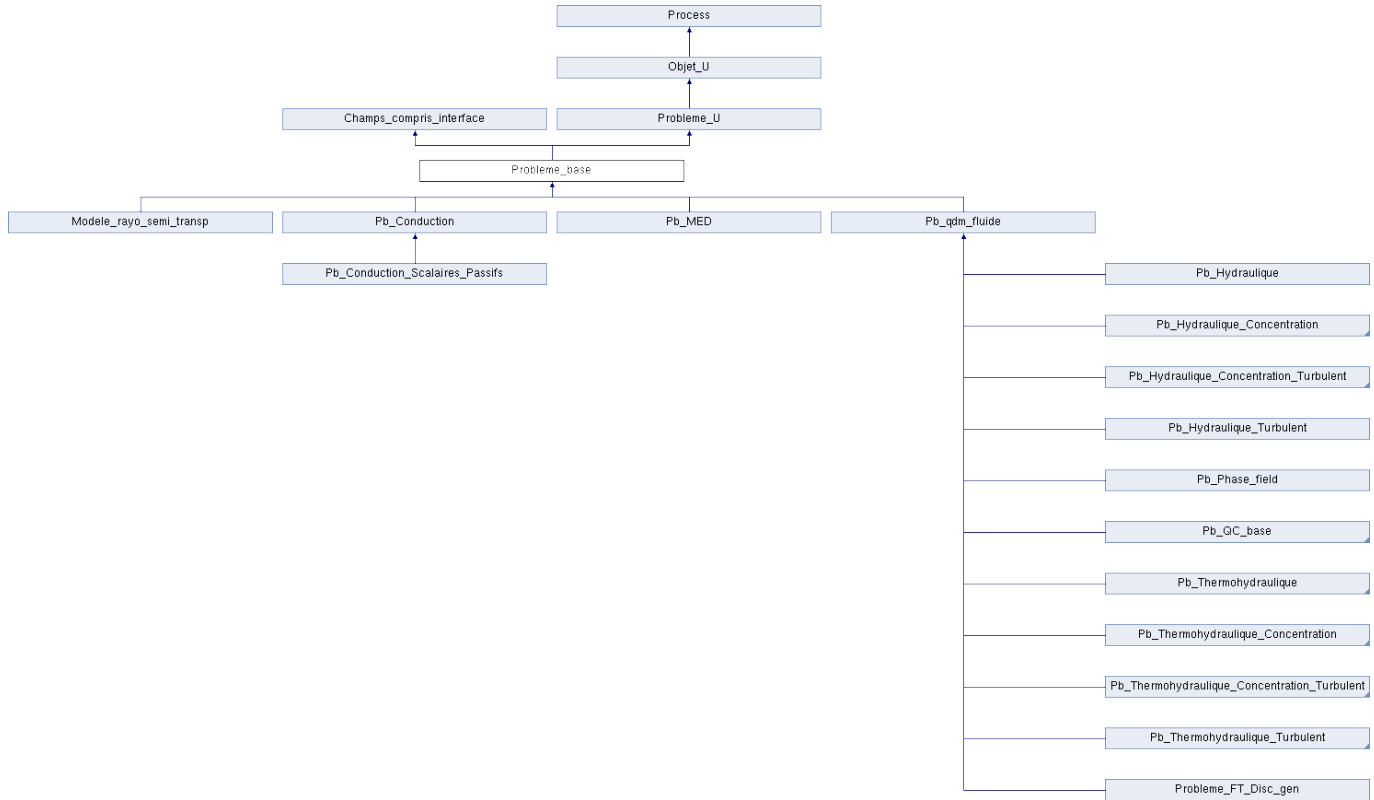


Figure 1: classe `Probleme_base`

- **équation de convection-diffusion**

En analysant le graphe doxygen de la classe `Equation_base` (cf. figure 2), on constate qu'il existe une équation de type convection-diffusion nommée `Convection_Diffusion_std` (cf. figure 3). C'est une classe de base pour l'équation de transport d'un scalaire en régime laminaire et elle porte une référence au champ de la vitesse transportante [2]. Cette classe d'équation est abstraite pure et donc non instanciable. L'implémentation de cette classe n'est pas complète et elle sert de base à d'autres classes dérivées (héritées).

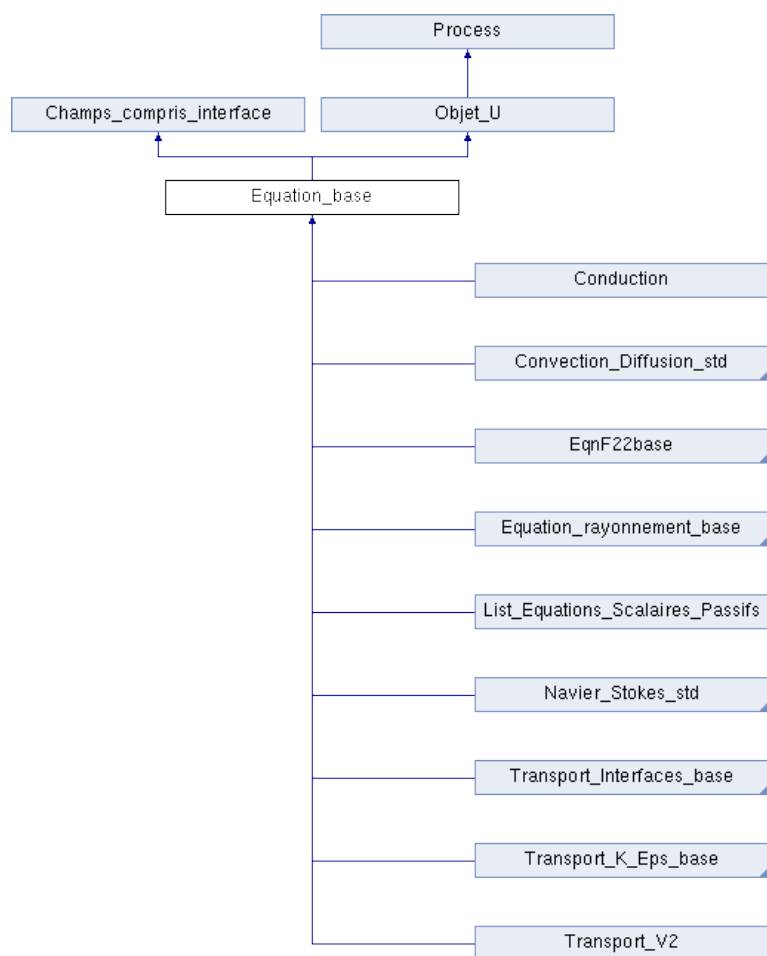


Figure 2: classe Equation\_base

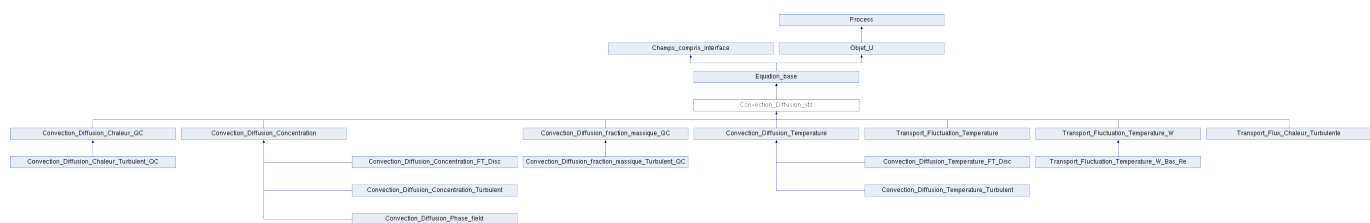


Figure 3: classe Convection\_Diffusion\_std

En se basant sur le graphe doxygen de la classe `Convection_Diffusion_Concentration` (cf. figure 4), la classe `Convection_Diffusion_Concentration` fille (dérivée) de la classe `Convection_Diffusion_std` paraît très proche de l'équation (1). Cette classe est un cas particulier de `Convection_Diffusion_std` pour le transport d'un ou plusieurs constituants [2].

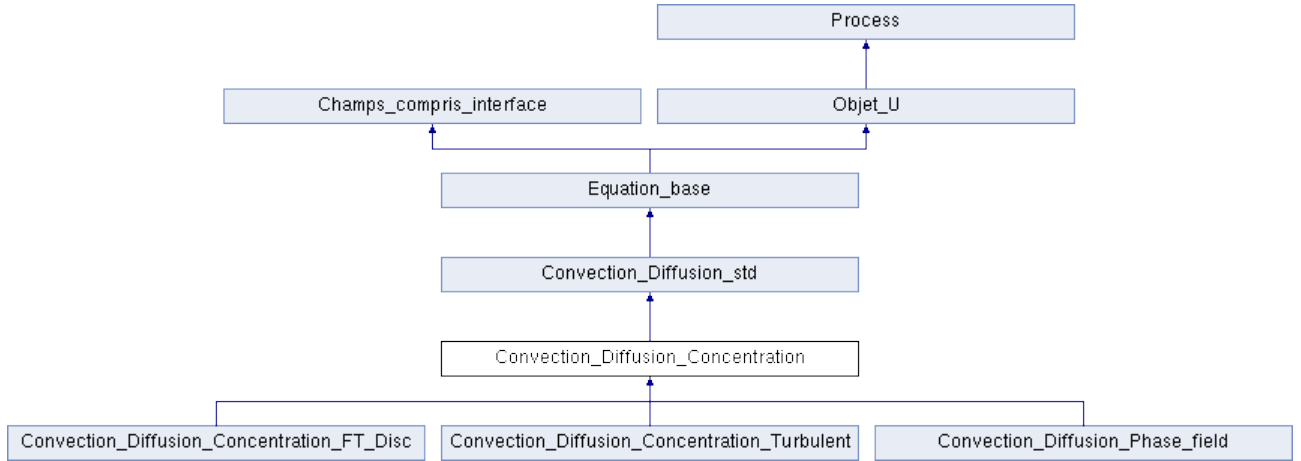


Figure 4: classe `Convection_Diffusion_Concentration`

Les spécificités de l'équation `Convection_Diffusion_Concentration` sont [2]:

- coefficient de diffusion [3]: ce coefficient constitue une propriété d'un constituant associé à cette équation. Cette propriété est donnée par l'utilisateur dans le jeu de données de la manière suivante:

```

Constituant constituant
2 Read constituant
{
4   coefficient_diffusion  Champ_Uniforme 1 0.001
}

```

déclaration du constituant standard `TRIO_U`

- vitesse de convection [2]: cette vitesse est obtenue par la méthode `vitesse_pour_transport` qui récupère cette vitesse à partir de l'équation du problème qui la porte. En effet, dans une utilisation standard du logiciel `TRIO_U`, cette vitesse de convection est obtenue par l'équation de l'hydraulique Navier-Stokes.

### 3.4 Fonctionnalités spécifiques

L'analyse des fonctionnalités du logiciel `TRIO_U` dans la section précédente a permis d'identifier une équation `Convection_Diffusion_Concentration` très proche de l'équation (1) avec un coefficient de diffusion donné par l'utilisateur à travers un constituant. Nous faisons donc le



choix d'utiliser cette équation comme base pour développer une nouvelle équation qu'on notera **Convection\_Diffusion**. A cette équation, on associe un nouveau constituant **Constituant\_Avec\_Vitesse** basé sur l'ancien en lui rajoutant comme propriété la vitesse de convection.

La nouvelle classe de constituant **Constituant\_Avec\_Vitesse** sera appelée dans le jeu de données de la manière suivante:

```
1 Constituant_Avec_Vitesse constituant
  Read constituant
3 {
    coefficient_diffusion  Champ_Uniforme 1 0.001
5 vitesse_convection  Champ_Fonc_txyz dom 2 x*t y*t
  }
```

déclaration du constituant spécifique à l'application BALTIK

## 4 Développements

### 4.1 Environnement de développement : Application BALTIK

Un environnement de développement Application BALTIK est mis à disposition pour les applications reposant, soit sur le logiciel **TRIO\_U** soit, sur son noyau numérique « Kernel » soit sur une autre application BALTIK. Cet environnement permet une gestion de compilation de ces applications (sources spécifiques) ainsi que les sources modifiées et/ou rajoutées du logiciel **TRIO\_U**. Dans ce cadre, une application est constituée :

- du noyau numérique « Kernel » ou du logiciel **TRIO\_U** ou d'une autre application BALTIK,
- des sources spécifiques à l'application.

En pratique, les sources spécifiques à l'application sont rassemblées dans un répertoire de travail appelé **src**<sup>1</sup>, celui-ci est compilé suivant le processus classique de compilation du logiciel **TRIO\_U**. On peut inclure à cette application des sources du « Kernel » ou de **TRIO\_U** que l'on souhaite modifier par rapport à leurs versions standards.

#### 4.1.1 Création et configuration de l'application BALTIK

La création d'une application BALTIK consiste à créer un répertoire qui prend le nom de cette application (**convection\_diffusion** par exemple).

```
$ mkdir convection_diffusion
```

La mise en place d'une application BALTIK se traduit par la création dans le répertoire **convection\_diffusion** d'un fichier de configuration **project.cfg**.

<sup>1</sup>L'emplacement du répertoire **src** dans l'arborescence de l'application BALTIK est indiqué dans la section 4.1.1

```

1 # description section (required)
  [description]
3 name      : convection_diffusion      # name of the project      (
    required)
  author     : MEKKAS                    # author of the project   (
    optional)
5 executable : convection_diffusion_exe # name of the executable (
    optional)

```

project.cfg

Dans ce fichier de configuration, seul le nom de l'application est obligatoire. En revanche, il existe un certain nombre de paramètres optionnels tels que l'auteur, le nom de l'exécutable à générer, etc.

Pour configurer notre application BALTIK, il faut exécuter les instructions suivantes:

- initialisation de l'environnement du logiciel:

```

1 $ cd convection_diffusion
  $ mkdir src
3
  src est le répertoire contenant soit des sources modifiées du
    logiciel
5 TRIO_U soit du Kernel TRIO_U soit des sources supplémentaires
    spécifiques
  à notre application.

```

- configuration de l'application BALTIK:

```

$ source $TRIO_U_ROOT/bin/Init_Trio_U
2 $ cd convection_diffusion
  $ baltik_build_configure
4 La variable d'environnement $TRIO_U_ROOT est le chemin où le
    logiciel
  TRIO_U est installé.
6 La première instruction sert à initialiser l'environnement
  TRIO_U.
  La deuxième instruction génère un fichier configure. Il est
    important de
8 noter que cette instruction doit être exécuter dans le
    répertoire
  convection_diffusion.

```

### 4.1.2 Compilation de l'application BALTIK

La compilation de notre application BALTIK est assez standard. En effet, la génération du fichier Makefile se fait via l'exécution dans le répertoire `convection_diffusion` de l'instruction suivante:

```
1 $ cd convection_diffusion
2 $ ./configure
3
4 Il est important de noter que lors du rajout de nouvelles sources
5 dans
6 le répertoire src, il est impératif de re-exécuter cette
7 instruction
8 pour mettre à jour le fichier Makefile.
```

Une fois le fichier Makefile généré, deux modes de compilation de l'application BALTIK sont possibles:

- compilation en mode debug:

```
1 $ cd convection_diffusion
2 $ make debug
```

- compilation en mode optimisé:

```
1 $ cd convection_diffusion
2 $ make optim
```

## 4.2 Développements

### 4.3 Classe constituant `Constituant_Avec_Vitesse`

Comme spécifié dans la section 3, les paramètres utilisateurs vitesse de convection et coefficient de diffusion sont attribués à la nouvelle classe de constituant `Constituant_Avec_Vitesse` basée (héritage) sur la classe `Constituant` du logiciel TRIO\_U. La spécificité de cette nouvelle classe est que cette dernière possède un champ de vitesse de convection du constituant.

Le traitement de cet attribut supplémentaire nécessite les actions suivantes:

1. le rajout d'attributs:

- `C`: il contiendra le contenu du `vitesse_convection` du jeu de données (cf. section 3).
- `vitesse_transport`: il contiendra la vitesse de convection discrétisée.

- `champs_compris_`: il contiendra les champs post-traitables.

2. la surcharge des méthodes suivantes<sup>2</sup>:

- `initialiser`: évaluer la vitesse de convection en espace au temps initial.
- `set_param`: affecter le contenu de `vitesse_convection` du jeu de données (cf. section 3) à l'attribut `C` de cette classe.
- `mettre_a_jour`: évaluer la vitesse de convection en espace au temps courant.
- `discrétiser`: discrétise la vitesse de convection à partir de la vitesse de convection donné par l'utilisateur.
- `get_champ` et `get_noms_champs_postraitables`: rendre la vitesse de convection discrétisée post-traitable à partir du jeu de données.

3. le développement de nouvelles méthodes:

- `vit_convection_constituant`: renvoie l'attribut contenant la vitesse de convection non discrète.
- `vitesse_pour_transport`: renvoie la vitesse de convection discrétisée sur les faces du maillage.

4. le développement des méthodes `printOn` et `readOn`: la classe `Constituant_Avec_Vitesse` est utilisable dans le jeu de données et donc instantiable. Pour déclarer ce type de classe, il suffit d'utiliser la macro `Declare_instanciable`. Dans cette macro, on déclare les entêtes des méthodes `printOn` et `readOn` qui sont nécessaires à la lecture des objets sur les flux d'entrée (jeu de données) et l'écriture des objets sur les flux de sortie.

#### 4.3.1 Fichier entête

```

1  #ifndef Constituant_Avec_Vitesse_included
2  #define Constituant_Avec_Vitesse_included
3
4  #include <Constituant.h>
5  #include <Champ_Inc.h>
6
7  class Constituant_Avec_Vitesse : public Constituant
8  {
9      Declare_instanciable(Constituant_Avec_Vitesse);
10 public :
11     int    initialiser(const double& temps);
12     void    set_param(Param& param);
13     void    mettre_a_jour(double temps);
14     const Champ_Don& vit_convection_constituant() const;

```

<sup>2</sup>Seules les fonctionnalités spécifiques de la nouvelle classe `Constituant_Avec_Vitesse` sont décrites.

```

16     const Champ_base& vitesse_pour_transport() ;
    virtual void discretiser(const Probleme_base& pb, const
        Discretisation_base& dis);
protected :
18     Champ_Don C;
private :
20     Champ_Inc vitesse_transport;
    };
22 #endif

```

Constituant\_Avec\_Vitesse.h

### 4.3.2 Fichier source

Il est à noter que dans le fichier source, qui contient le corps des méthodes définies dans le fichier entête, la macro `Implemente_instanciable` sert, d'une part, à définir le mot clé utilisable dans le jeu données pour représenter la classe `Constituant_Avec_Vitesse` et, d'autre part, la classe mère de `Constituant_Avec_Vitesse`. Dans notre cas, on choisit le nom de la classe `Constituant_Avec_Vitesse` comme mot clé dans le jeu de données.

```

#include <Constituant_Avec_Vitesse.h>
2 #include <Probleme_base.h>
#include <Discretisation_base.h>
4 #include <Equation_base.h>
#include <Param.h>
6 #include <Schema_Temps_base.h>

8 Implemente_instanciable(Constituant_Avec_Vitesse,"
    Constituant_Avec_Vitesse",Constituant);

10 Sortie& Constituant_Avec_Vitesse::printOn(Sortie& os) const
{
12     return os;
}

14 Entree& Constituant_Avec_Vitesse::readOn(Entree& is)
16 {
    Constituant::readOn(is);
18     return is;
}

20 void Constituant_Avec_Vitesse::set_param(Param& param)
22 {
    Constituant::set_param(param);
24     param.ajouter("vitesse_convection",&C,Param::REQUIRED);

```

```

}
26
void Constituant_Avec_Vitesse::discretiser(const Probleme_base& pb,
28                                     const Discretisation_base&
        dis)
{
30     const Zone_dis_base& zone_dis=pb.equation(0).zone_dis();
    dis.nommer_completer_champ_physique(zone_dis,"
        coefficient_diffusion","m2/s",D.valeur(),pb);
32     dis.nommer_completer_champ_physique(zone_dis,"
        vitesse_convection_lu","m/s",C.valeur(),pb);
    champs_compris_.ajoute_champ(D.valeur());
34     champs_compris_.ajoute_champ(C.valeur());
    const Schema_Temps_base& sch=pb.schema_temps();
36     dis.discretiser_champ("vitesse",zone_dis,"vitesse_convection","m/
        s",
                                dimension,sch.nb_valeurs_temporelles(),
38                                sch.temps_courant(),vitesse_transport);
    champs_compris_.ajoute_champ(vitesse_transport.valeur());
40     Milieu_base::discretiser(pb,dis);
    if (C.non_nul())
42         vitesse_transport.valeur().affecter_(C.valeur());
    else
44     {
        Cerr<<"To must read "<<que_suis_je()<<" before discretiser "<<
        finl;
46         exit();
    }
48 }

50 const Champ_base& Constituant_Avec_Vitesse::vitesse_pour_transport
    ()
{
52     return vitesse_transport;
}

54 const Champ_Don& Constituant_Avec_Vitesse::
    vit_convection_constituant() const
56 {
    return C;
58 }

60 void Constituant_Avec_Vitesse::mettre_a_jour(double temps)
{

```

```

62  Constituant::mettre_a_jour(temps);
    if (C.non_nul())
64      C.mettre_a_jour(temps);
    vitesse_transport.valeur().affecter_(C.valeur());
66  vitesse_transport.changer_temps(temps);
}

68
int Constituant_Avec_Vitesse::initialiser(const double& temps)
70 {
    Constituant::initialiser(temps);
72  if (C.non_nul())
        C.initialiser(temps);
74  return 1;
}

```

Constituant\_Avec\_Vitesse.cpp

#### 4.4 Classe équation Convection\_Diffusion

La spécificité de cette nouvelle classe d'équation `Convection_Diffusion` par rapport à la classe de base `Convection_Diffusion_Concentration` (3) est que la classe `Convection_Diffusion` possède une référence sur la nouvelle classe de constituant `Constituant_Avec_Vitesse` et que la vitesse de convection est donnée par cette dernière classe et non pas donnée par une résolution d'une équation de l'hydraulique Navier-Stokes.

Le traitement de cette nouvelle référence nécessite les actions suivantes:

1. le rajout de la référence `le_constituant`. Pour ce faire, il est nécessaire d'implémenter les deux sources ci-dessous. Cette technique, non standard C++, est utilisée dans le logiciel `TRIO_U` pour éviter aux développeurs de manipuler des pointeurs qui peuvent être source des fuites mémoires.

```

1  #ifndef Ref_Constituant_Avec_Vitesse_included
    #define Ref_Constituant_Avec_Vitesse_included
3
    #include <Ref.h>
5  class Constituant_Avec_Vitesse;
    Declare_ref(Constituant_Avec_Vitesse);
7  #endif

```

Ref\_Constituant\_Avec\_Vitesse.h

```

1  #include <Ref_Constituant_Avec_Vitesse.h>
    #include <Constituant_Avec_Vitesse.h>
3  Implemente_ref(Constituant_Avec_Vitesse);

```

2. la surcharge des méthodes suivantes <sup>3</sup>:

- **associer\_milieu\_base**: associer le milieu à l'équation de convection diffusion et plus précisément associer le nouveau constituant **Constituant\_Avec\_Vitesse** à l'équation.
- **associer\_constituant**: associer le constituant à l'équation **Convection\_Diffusion**. Cette méthode est appelée par la méthode précédente **associer\_milieu\_base**.
- **lire\_motcle\_non\_standard**: lire les mots clé non standard du jeu de données tel que **convection** pour associer la vitesse de convection à l'équation de convection diffusion

3. le développement des méthodes **printOn** et **readOn**.

#### 4.4.1 Fichier entête

```
1  #ifndef Eq_Conv_Diff_included
3  #define Eq_Conv_Diff_included
5  #include <Convection_Diffusion_Concentration.h>
6  #include <Ref_Constituant_Avec_Vitesse.h>
7
9  class Eq_Conv_Diff : public Convection_Diffusion_Concentration
10 {
11     Declare_instanciable(Eq_Conv_Diff);
12 public :
13     void associer_milieu_base(const Milieu_base& un_milieu);
14     void associer_constituant(const Constituant_Avec_Vitesse&
15         un_constituant);
16     int lire_motcle_non_standard(const Motcle& mot, Entree& is);
17 protected :
18     REF(Constituant_Avec_Vitesse) le_constituant;
19 };
20 #endif
```

Eq\_Conv\_Diff.h

#### 4.4.2 Fichier source

---

<sup>3</sup>Seul les fonctionnalités spécifiques de la nouvelle classe **Convection\_Diffusion** sont décrites.



```

#include <Eq_Conv_Diff.h>
2 #include <Constituant_Avec_Vitesse.h>
#include <Discret_Thyd.h>
4 #include <Probleme_base.h>

6 Implemente_instanciable(Eq_Conv_Diff,"Convection_Diffusion",
    Convection_Diffusion_Concentration);

8 Sortie& Eq_Conv_Diff::printOn(Sortie& is) const
{
10     return Convection_Diffusion_Concentration::printOn(is);
}

12 Entree& Eq_Conv_Diff::readOn(Entree& is)
{
14     Convection_Diffusion_Concentration::readOn(is);
16     return is;
}

18 int Eq_Conv_Diff::lire_motcle_non_standard(const Motcle& mot,Entree
    & is)
20 {
    if (mot=="convection")
22     {
        Cerr <<"Reading and typing of the convection operator : "<< finl;
24     terme_convectif.associer_vitesse(le_constituant->
        vitesse_pour_transport());
        is >> terme_convectif;
26     return 1;
    }else
28     {
        Cout << mot << finl;
30     return Convection_Diffusion_Concentration::
        lire_motcle_non_standard(mot,is);
    }
32     return 1;
}

34 void Eq_Conv_Diff::associer_milieu_base(const Milieu_base&
    un_milieu)
36 {
    Convection_Diffusion_Concentration::associer_milieu_base(
        un_milieu);
}

```

```

38  const Constituant_Avec_Vitesse& un_constituant=ref_cast(
    Constituant_Avec_Vitesse,un_milieu);
40  if (un_constituant.vit_convection_constituant().non_nul())
    associer_constituant(un_constituant);
    else
42  {
        Cerr <<"The dye (constituant) convection has not been defined."
        << endl;
44        exit();
    }
46 }

48 void Eq_Conv_Diff::associer_constituant(const
    Constituant_Avec_Vitesse& un_constituant)
{
50     le_constituant = un_constituant;
}

```

Eq\_Conv\_Diff.cpp

## 4.5 Classe Problème Probleme\_Convection\_Diffusion

Dans la section 3 dédiée à la spécification, aucun problème n'a été identifié pour servir de base au nouveau problème `Probleme_Convection_Diffusion`. Nous prenons donc la classe abstraite pure `Probleme_base` comme classe de base (mère) pour la nouvelle classe `Probleme_Convection_Diffusion`. Pour utiliser cette nouvelle classe de problème `Probleme_Convection_Diffusion`, il faut:

1. rajouter l'attribut `eq_conv_diff` qui représente l'équation de convection-diffusion (1).
2. compléter l'implémentation de la classe `Probleme_base` en développant les méthodes suivantes:
  - `nombre_d_equations`: renvoie le nombre d'équation qui compose ce problème. Dans notre cas c'est une seule équation.
  - `equation`: renvoie l'attribut contenant l'équation de convection-diffusion (`eq_conv_diff`).
3. surcharger la méthode `associer_milieu_base` pour associer le milieu à l'équation de convection de diffusion `eq_conv_diff`.
4. le développement des méthodes `printOn` et `readOn`.

### 4.5.1 Fichier entête

```

1  #ifndef Pb_Conv_Diff_included

```

```

3 #define Pb_Conv_Diff_included
5 #include <Probleme_base.h>
6 #include <Eq_Conv_Diff.h>
7
9 class Pb_Conv_Diff : public Probleme_base
10 {
11     Declare_instanciable(Pb_Conv_Diff);
12 public:
13     int nombre_d_equations() const;
14     const Equation_base& equation(int i) const ;
15     Equation_base& equation(int i);
16     void associer_milieu_base(const Milieu_base& un_milieu);
17 protected:
18     Eq_Conv_Diff eq_conv_diff;
19 };
20 #endif

```

Pb\_Conv\_Diff.h

#### 4.5.2 Fichier source

```

1 #include <Pb_Conv_Diff.h>
2
3 Implemente_instanciable(Pb_Conv_Diff,"Probleme_Convection_Diffusion",Probleme_base);
4
5 Sortie& Pb_Conv_Diff::printOn(Sortie& os) const
6 {
7     return Probleme_base::printOn(os);
8 }
9
10 Entree& Pb_Conv_Diff::readOn(Entree& is)
11 {
12     return Probleme_base::readOn(is);
13 }
14
15 int Pb_Conv_Diff::nombre_d_equations() const
16 {
17     return 1;
18 }
19
20 const Equation_base& Pb_Conv_Diff::equation(int i) const
21 {

```

```

23     assert (i==0);
    return eq_conv_diff;
}
25
Equation_base& Pb_Conv_Diff::equation(int i)
27 {
    assert (i==0);
29     return eq_conv_diff;
}
31
void Pb_Conv_Diff::associer_milieu_base(const Milieu_base&
    un_milieu)
33 {
    eq_conv_diff.associer_milieu_base(un_milieu);
35 }

```

Pb\_Conv\_Diff.cpp

## 5 Processus de vérification des développements

Dans cette section, nous présentons:

- la mise en place des tests de vérification,
- la génération automatique d'un rapport de vérification des développements via l'outil de génération `genererCourbes`,
- la vérification de la non-régression.

### 5.1 Mise en place des tests de vérification

La mise en place des tests de vérification consiste à créer:

- un répertoire pour chaque test de vérification,

```

1 $ cd convection_diffusion
$ mkdir -p share/Validation/Rapports_automatiques/Cas-tests/src
  /cas-test1
3 $ mkdir -p share/Validation/Rapports_automatiques/Cas-tests/src
  /cas-test2
5 Cas-tests est le répertoire contenant les tests de vérification
.
cas-test1 est le répertoire contenant le premier cas de
  vérification.

```

```
7 cas-test2 est le répertoire contenant le deuxième cas de
   vérification.
```

- un fichier PRM dans le répertoire `convection_diffusion/share/Validation/Rapports_automatiques/Ca` qui sert à générer automatiquement un rapport de vérification (section 6). Ce fichier PRM est présenté dans la section 6.4.
- un jeu de donnée pour chaque test de vérification (sections 6.5.1 et 6.5.2).

## 5.2 Génération de rapport de vérification

La génération consiste à exécuter la commande `make validation` dans le répertoire `convection_diffusion`. Dans ce même répertoire, l'exécution de cette commande générera le rapport de vérification `validation.pdf`. Ce rapport est inclus dans cette note technique à la section 6.

## 5.3 Vérification de la non-régression

Pour utiliser les tests mis en place précédemment dans le rapport de vérification comme des tests de non-régression, il suffit de:

- créer un répertoire qui contiendra les résultats de référence dans le répertoire `convection_diffusion`:

```
1 $ cd convection_diffusion
   $ mkdir -p tests/Reference/Validation
```

- lancement des tests et vérification de la non-régression en exécutant la commande `make check_optim` dans le répertoire `convection_diffusion`. Le résultat de cette commande est illustré dans la figure (cf. figure 5.3). Cette figure montre que les résultats en séquentiel et en parallèle ne présentent aucune régression.

```

-----
Complete results into the file:
/export/home/am217485/applications/convection_diffusion/build/tests/.tests_convection_diffusion_exe_opt
Mon Jan 27 23:04:17 CET 2014 : Test results with the version dated Jan 27 17:29
/export/home/am217485/applications/convection_diffusion/convection_diffusion_exe_opt
-----
| START | END | CPU | NP | SPU | NDT | GCP | ELEM | MEM | PLOTS | STATE | NAME
-----
| 23:04:17 | 23:04:17 | ...0s | ..1. | ..... | ....3 | .....0 | .10000 | ..18Mo | ..... | OK | Cas-tests_jdd1
| 23:04:17 | 23:04:19 | ...2s | ..2. | 1.691 | ....3 | .....0 | .10000 | ..34Mo | ....2/2 | OK | PAR_Cas-tests_jdd1
| 23:04:20 | 23:04:21 | ...1s | ..1. | ..... | ....3 | .....0 | .10000 | ..18Mo | ..... | OK | Cas-tests_jdd2
| 23:04:21 | 23:04:22 | ...1s | ..2. | 1.691 | ....3 | .....0 | .10000 | ..34Mo | ....2/2 | OK | PAR_Cas-tests_jdd2
-----

Successful tests cases :2/2
Successful tests cases in parallel mode :2/2

4 test cases run in 4 s ( 2 s/test), so less than 1 hours
Parallel tests cases :
  2 PARALLEL OK
Successful result.
Info_global test seq 2/2
Info_global test par 2/2

```

Figure 5: Résultats des tests de non-régression

## 6 Rapport de vérification

### 6.1 Introduction

Validation made by : A. MEKKAS.

Report generated 20/11/2014.

#### 6.1.1 Description

Dans cette fiche de tests, deux cas-tests ont été développés pour une vérification qualitative des développements d'une équation de convection-diffusion.

#### 6.1.2 Parameters Trio\_U

- Version Trio\_U :
- Version Trio\_U from out: /users/fauchet/VUES/beta/trio\_u/Trio\_U/doc/Trio\_U/exercices/equation\_ (1.7.0\_beta)

#### 6.1.3 Test cases

- cas-test1/cas-test1.data :
- cas-test2/cas-test2.data :

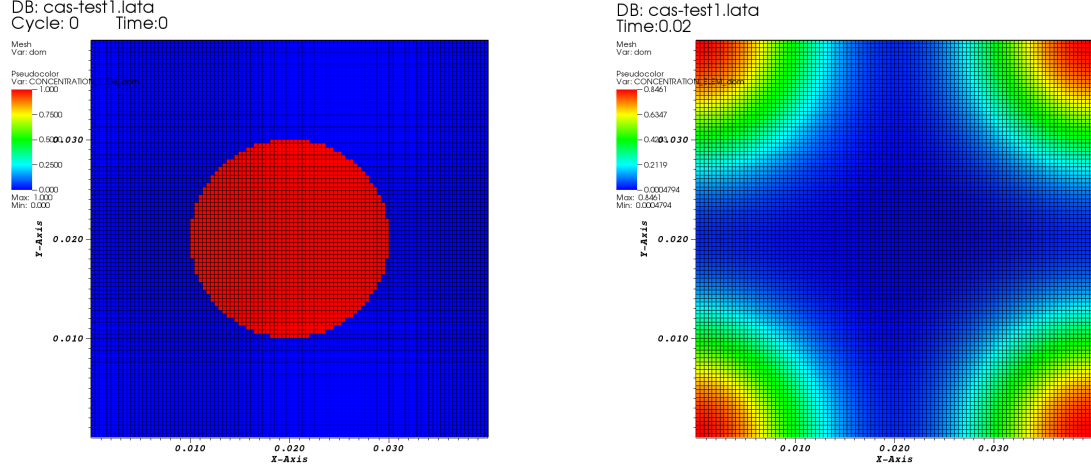
### 6.2 Cas-test 1

Ce premier cas-test vise à vérifier la solution d'une équation de convection-diffusion sur un domaine 2D  $[0, 0.04] \times [0, 0.04]$  pour une vitesse de convection constante, un coefficient de diffusion constant et des conditions aux limites périodiques:

- $\vec{V}(\vec{X}, t) = (1, 1)$ ,
- $D(\vec{X}, t) = 0.0005$ .

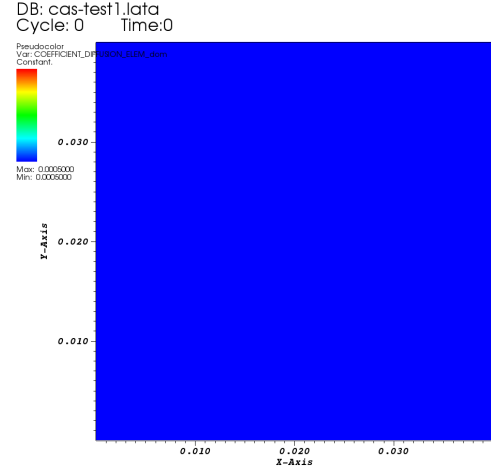
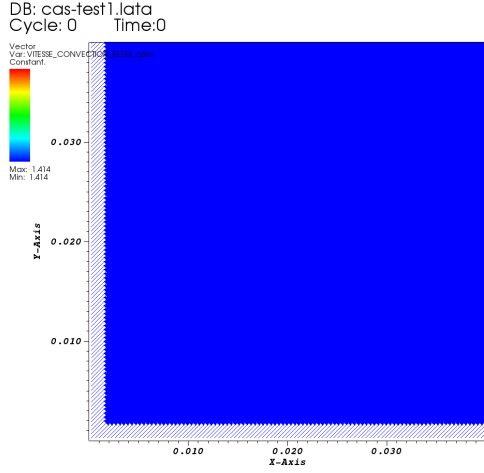
### 6.2.1 Concentration

Les figures ci-dessous représentent la concentration à l'instant initial et à l'instant  $t = 0.02$ . La figure représentant la solution à  $t = 0.02$  montre la bonne prise en compte de la vitesse de convection, de la diffusion de la concentration initiale et des conditions aux limites périodiques.



### 6.2.2 Vitesse de convection et coefficient de diffusion

Les figures ci-dessous représentent la vitesse de convection et le coefficient de diffusion à l'instant initial. Ces figures montrent la bonne prise en compte de la vitesse de convection et du coefficient de diffusion. Ceci vérifie également que le rajout de la vitesse de convection en tant que variable post-traitable est bien pris en compte.



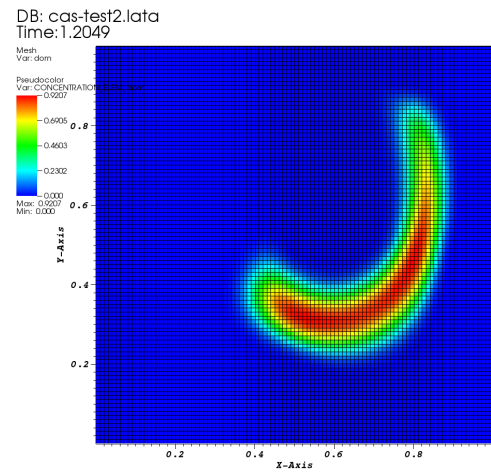
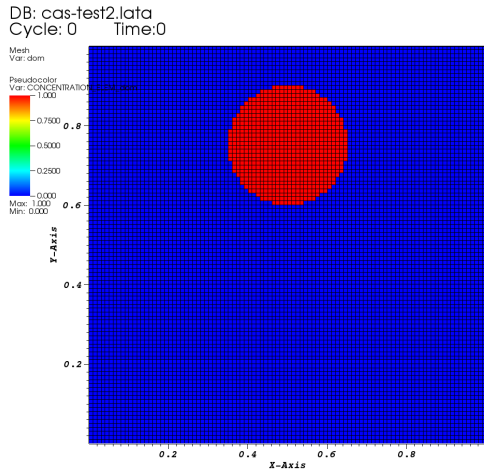
### 6.3 Cas-test 2

Le deuxième cas-test vise à vérifier la solution d'une équation de convection-diffusion sur un domaine 2D  $[0, 1] \times [0, 1]$  pour une vitesse de convection variable en temps et en espace et un coefficient de diffusion nul:

- $\vec{V}(\vec{X}, t) = (-2 \sin(\pi x)^2 \sin(\pi y) \cos(\pi y) \cos(t \frac{\pi}{2}), 2 \sin(\pi y)^2 \sin(\pi x) \cos(\pi x) \cos(t \frac{\pi}{2}))$ ,
- $D(\vec{X}, t) = 0$ .

#### 6.3.1 Concentration

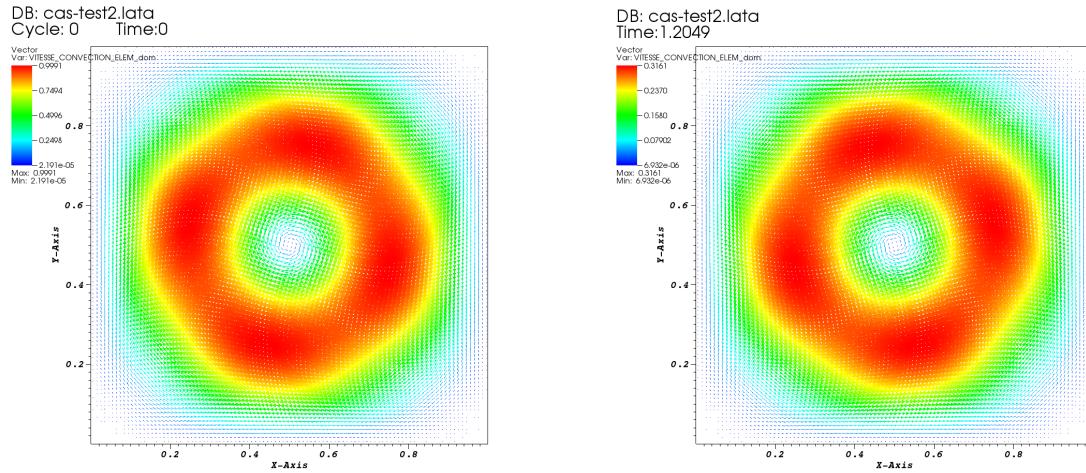
Les figures ci-dessous représentent la concentration à l'instant initial et à l'instant  $t = 1.2049$ .





### 6.3.2 Vitesse de convection

Les figures ci-dessous représentent la vitesse de convection à l'instant initial et à l'instant  $t = 1.2049$ . La figure représentant la solution à  $t = 1.2049$  montre la bonne prise en compte d'une vitesse de convection variable en temps et en espace. En effet, Les figures ci-dessous mettent en évidence le changement de signe à l'instant  $t = 1.2049$  dû au changement de signe du terme  $\cos(t\frac{\pi}{2})$ .



### 6.4 Fichier prm

```
Parametres {
  Titre "Tests de vérification"
  Description "Dans cette fiche de tests , deux cas-tests ont été développés
    pour une vérification qualitative des développements d'une équation de
    convection-diffusion."
  IncludeData 0
  Auteur "A. MEKKAS"
  CasTest "cas-test1" "cas-test1.data"
  CasTest "cas-test2" "cas-test2.data"
}
Chapitre {
  Titre "Cas-test 1"
  Description "Ce premier cas-test vise à vérifier la solution d'une
    équation de convection-diffusion sur un domaine 2D \latex_([0,0.04]\
    times[0,0.04]\latex_) pour une vitesse de convection constante , un
    coefficient de diffusion constant et des conditions aux limites
    périodiques:"
    Description "\latex_(\begin{itemize} \item $\overrightarrow{V}(\
    overrightarrow{X},t)=(1,1)$, \item $\mathcal{D}(\overrightarrow{X},t)=0.0005$. \
    end{itemize}\latex_)"
  Table {
```

```

titre "Concentration"
description "Les figures ci-dessous représentent la concentration à l'
instant initial et à l'instant  $t=0.02$ . La figure
représentant la solution à  $t=0.02$  montre la bonne
prise en compte de la vitesse de convection, de la diffusion de la
concentration initiale et des conditions aux limites périodiques."
description
description
}
Visu {
Width 7cm
mesh cas-test1/cas-test1.lata dom
pseudocolor cas-test1/cas-test1.lata dom CONCENTRATION ELEM
cycles 0 350
# MultiVisu 1 #
nb_img_without_newline 2
}
Table {
titre "Vitesse de convection et coefficient de diffusion"
description "Les figures ci-dessous représentent la vitesse de convection
et le coefficient de diffusion à l'instant initial. Ces figures montrent
la bonne prise en compte de la vitesse de convection et du coefficient
de diffusion. Ceci vérifie également que le rajout de la vitesse de
convection en tant que variable post-traitable est bien pris en compte."
description
}
Visu {
vector cas-test1/cas-test1.lata dom VITESSE_CONVECTION ELEM
Width 7cm
cycles 0
nb_img_without_newline 2
}
Visu {
pseudocolor cas-test1/cas-test1.lata dom COEFFICIENT_DIFFUSION ELEM
Width 7cm
cycles 0
}
}
Chapitre {
Titre "Cas-test 2"
Description "Le deuxieme cas-test vise à vérifier la solution d'une
équation de convection-diffusion sur un domaine 2D  $[0,1] \times [0,1]$  pour une vitesse de convection variable en temps et en
espace et un coefficient de diffusion nul:"
Description "\begin{itemize} \item  $\overrightarrow{V}(\overrightarrow{X},t)=(-2\sin(\pi x)^2\sin(\pi y)\cos(\pi y)\cos(t\frac{\pi}{2}), 2\sin(\pi y)^2\sin(\pi x)\cos(\pi x)\cos(t\frac{\pi}{2}))$ , \item  $D(\overrightarrow{X},t)=0$ . \end{itemize}"
Table {

```

```

titre "Concentration"
description "Les figures ci-dessous représentent la concentration à l'
instant initial et à l'instant \latex_{$t=1.2049$\latex_}."
description
description
}
Visu {
Width 7cm
mesh cas-test2/cas-test2.lata dom
pseudocolor cas-test2/cas-test2.lata dom CONCENTRATION ELEM
cycles 0 88
nb_img_without_newline 2
}
Table {
titre "Vitesse de convection"
description "Les figures ci-dessous représentent la vitesse de convection
à l'instant initial et à l'instant \latex_{$t=1.2049$\latex_}. La
figure représentant la solution à \latex_{$t=1.2049$\latex_} montre la
bonne prise en compte d'une vitesse de convection variable en temps et
en espace. En effet, Les figures ci-dessous mettent en évidence le
changement de signe à l'instant \latex_{$t=1.2049$\latex_} d\latex_{$\wedge u\backslash
latex_} au changement de signe du terme \latex_{$\cos(t\frac{\pi}{2})$\backslash
latex_}."
description
description
}
Visu {
Width 7cm
vector cas-test2/cas-test2.lata dom VITESSE_CONVECTION ELEM
cycles 0 88
nb_img_without_newline 2
}
}
}
Chapitre {
Titre "Fichier prm"
Description \latex_{$\backslash label{prm}\backslash latex_}
Description include_text_file(fiche.prm)
}
Chapter {
Title Data Files
Table {
titre cas-test1
Description \latex_{$\backslash label{cas1}\backslash latex_}
Description include_text_file(cas-test1/cas-test1.data)
}
Table {
titre cas-test2
Description \latex_{$\backslash label{cas2}\backslash latex_}
Description include_text_file(cas-test2/cas-test2.data)
}
}

```

```
}  
}
```

preserve/fiche.prm

## 6.5 Data Files

### 6.5.1 cas-test1

```
# Thermohydraulique 2D VDF #  
# PARALLEL OK #  
dimension 2  
Probleme_Convection_Diffusion pb  
Domaine dom  
# BEGIN MESH #  
Mesh dom  
{  
  Pave Plaques  
  {  
    origine 0. 0.  
    longueurs 0.04 0.04  
    nombre_de_noeuds 101 101  
  }  
  {  
    bord Gauche X = 0. 0. <= Y <= 0.04  
    bord Gauche X = 0.04 0. <= Y <= 0.04  
    bord Bas Y = 0. 0. <= X <= 0.04  
    bord Bas Y = 0.04 0. <= X <= 0.04  
  }  
}  
# END MESH #  
# BEGIN PARTITION  
Partition dom  
{  
  Partitionneur tranche { tranches 2 1 }  
  Larg_joint 1  
  Nom_Zones DOM  
  periodique 2 Bas Gauche  
}  
End  
END PARTITION #  
# BEGIN SCATTER  
Scatter DOM.Zones dom  
END SCATTER #  
vdf dis  
schema_Euler_explicite sch  
Read sch  
{  
  tinit 0.
```

```

nb_pas_dt_max 500
tmax .04
dt_min 1.e-7
dt_max 0.1
dt_impr 1.e-7
dt_sauv 100.
seuil_statio 1.e-8
}
Constituant_Avec_Vitesse constituant
Read constituant
{
  coefficient_diffusion  Champ_Uniforme 1 0.0005
  vitesse_convection  Champ_Uniforme 2 1. 1.
}
Associate pb dom
Associate pb sch
Associate pb constituant
Discretize pb dis
Read pb
{
  Convection_Diffusion
  {
    diffusion { }
    convection { montant }
    conditions_initiales
    {
      Concentration champ_fonc_xyz dom 1 (x-0.02)^2+(y-0.02)^2<(0.01)^2
    }
    boundary_conditions
    {
      Gauche periodique
      Bas periodique
    }
  }
  Postraitement
  {
    format lata
      champs dt_post 1.e-7
    {
      Concentration elem
      vitesse_convection elem
      coefficient_diffusion elem
    }
  }
}
Solve pb
End

```

preserve/cas-test1/cas-test1.data

### 6.5.2 cas-test2

```
# Thermohydraulique 2D VDF #
# PARALLEL OK #
dimension 2
Probleme_Convection_Diffusion pb
Domaine dom
# BEGIN MESH #
Mesh dom
{
  Pave Plaques
  {
    origine 0. 0.
    longueurs 1. 1.
    nombre_de_noeuds 101 101
  }
  {
    bord Gauche X = 0. 0. <= Y <= 1.
    bord Gauche X = 1. 0. <= Y <= 1.
    bord Bas Y = 0. 0. <= X <= 1.
    bord Bas Y = 1. 0. <= X <= 1.
  }
}
# END MESH #
# BEGIN PARTITION
Partition dom
{
  Partitionneur tranche { tranches 2 1 }
  Larg_joint 1
  Nom_Zones DOM
  periodique 2 Bas Gauche
}
End
END PARTITION #
# BEGIN SCATTER
Scatter DOM.Zones dom
END SCATTER #
vdf dis
schema_Euler_explicite sch
Read sch
{
  tinit 0.
  nb_pas_dt_max 500
  tmax 8.
  dt_min 1.e-7
  dt_max 0.1
  dt_impr 1.e-7
  dt_sauv 100.
  seuil_statio 1.e-8
}
```

```

}
Constituant_Avec_Vitesse constituant
Read constituant
{
  coefficient_diffusion  Champ_Uniforme 1 1.E-15
  vitesse_convection  Champ_Fonc_txyz dom 2 -2*sin(pi*x)^2*sin(pi*y)*cos(pi*
    y)*cos(2.*t*pi/4.) 2*sin(pi*y)^2*sin(pi*x)*cos(pi*x)*cos(2.*t*pi/4.)
}
Associate pb dom
Associate pb sch
Associate pb constituant
Discretize pb dis
Read pb
{
  Convection_Diffusion
  {
    diffusion { }
    convection { amount }
    conditions_initiales
    {
      Concentration champ_fonc_xyz dom 1 ((x-0.5)^2+(y-0.75)^2<(0.15)^2)
    }
    boundary_conditions
    {
      Gauche  periodique
      Bas     periodique
    }
  }
  Postraitement
  {
    format lata
    champs dt_post 1.e-7
    {
      Concentration elem
      vitesse_convection elem
      coefficient_diffusion elem
    }
  }
}
Solve pb
End

```

preserve/cas-test2/cas-test2.data

## 7 Conclusion

Dans ce tutoriel, nous avons décrit à travers la résolution d'une équation de convection-diffusion les aspects suivants:

- l'environnement de développement du logiciel `TRIO_U` en réalisant une nouvelle application `BALTIK`,
- la démarche de spécifications et de développements. Dans cette démarche, il est nécessaire de réaliser des choix techniques tels que l'association de la vitesse de convection à un constituant et non pas à l'équation ou problème). Ces choix ne sont ni uniques ni meilleurs que d'autres. Il est à noter que le but principal de ce tutoriel est de présenter une démarche accompagnée de ces propres choix techniques en donnant quelques recommandations ou méthodologies à suivre lors de sa phase de spécification.
- la mise en place des tests de non-régression et la génération automatique d'un rapport de vérification des développement via des outils proposés par le logiciel `TRIO_U`.



## References

- [1] U. Bieder, V. Barthel, F. Ducros, F. Perdu, and P. Quéméré. Trio\_u users manual : Methodology for incompressible single phase flow in industrial applications. Note Technique DER/SSTH/LMDL/2007-058, CEA, 2007.
- [2] M. Farvaque, Ph. Emonot, and O. Cueto. Document de conception trio\_u version 1. Note Technique STR/LTML 96-20, CEA, 1996.
- [3] Trio\_U teams. Trio\_u user's manual v1.6.8. Note technique, CEA, 2013.