International Journal of Multidisciplinary Research and Growth Evaluation.

Approximation of the data of an evolution equation on FreeFem++

Yosra Annabi
Free Researcher, Street Cheikh Mohamed Hedi Belhadej, Romana, Tunisia

* Corresponding Author: **Yosra Annabi**

Article Info

ISSN (online): 2582-7138
Impact Factor: 5.307 (SJIF)
Volume: 05
Issue: 01
January-February 2024
Received: 10-12-2023;
Accepted: 13-01-2024
Page No: 663-670

Abstract
This article contains the algorithm and the program in Freefem++ for solving a problem of data completion on the edge of a square domain. The software is free and downloadable on the website freefem.org . It is a programming language that allows solving partial differential equations via the finite element method. After an introductory paragraph, the numerical scheme of the Kozlov-Maz'ya-Fomin is explained. The last section exposes the variables and the syntax necessary to program the reconstitution of the data via the resolution of two variational formulations. The code is finalized and functional. The results are presented in the form of numerical values and curves of the approximated functions.

Keywords: PDE, Freefem++, data completion, algorithm

1. Introduction

The numerical study always constitutes the last stage of a study of a mathematically modeled phenomenon. The model studied in this article is represented by the system (1). It also exists as an application in the books [2] and [9]. For this reason, we will not delay on the modeling side. Indeed, the model can represent a phenomenon of the hydrogeology of coastal groundwater as in [2] or a phenomenon of particle transport in hemodynamics as in [9] and more generally, it represents a model of solute (particle) transported in a saturated porous medium by diffusion, convection and dispersion. The inverse problem makes it possible to reconstruct data on an edge of a square domain. The equation formulated represents the transport of a solute by diffusion and convection in a solvent, called by the diffusion-convection equation. The algorithm chosen for the reconstruction of the data is the Kozlov-Maz'ya-Fomin algorithm. We will recall this algorithm and program it on Freefem++. Before approaching the problem in transient mode, we adopt a quasi-stationary scheme i.e. we discretize the temporal term using the implicit Euler scheme. Thus, we bring ourselves back to studying a stationary case at each time step. As for the numerical method considered for the spatial approximation, we apply the finite element method. On the programming side, as mentioned above, we used the FreeFem++ software with [13] as reference. The Freefem++ software can only draw the level lines (isoclines). On the other hand, the curves are plotted on Scilab, using files saved thanks to the command of stream on Freefem++.

2. Model

We are now interested in the inverse problem 1

$$\begin{cases} \frac{\partial C}{\partial t} + div\left(-D\nabla C + C\vec{U}\right) = 0 & \text{in } \Omega_T \\ \left(-D\nabla C + C\vec{U}\right).\vec{n} = \psi & \text{on } \Sigma_m \\ C = \Phi & \text{on } \Sigma_m \\ C(x,0) = C^0 & \text{in } \Omega \end{cases} \quad (1)$$

In which Ω is a domaine, $\partial\,\Omega = \Gamma_i \cup \Gamma_m$ et $T \in \mathbb{R}$.
In addition, $\Omega_T = \Omega$ x [0, T], $\sum_m = \Gamma_m\, x$ [0, T] and $\Sigma_i = \Gamma_i \cup$ [0, T]. In all numerical tests, the velocity vector is fixed at

$$\vec{U} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

In addition, the matrix of the global movement is denoted

$$D = \begin{pmatrix} alpha1 & 0 \\ 0 & alpha2 \end{pmatrix}$$

We try to approximate the flow and the shape of the curve C on Γ_i during the time interval [0, T].

3. Algorithm

In the algorithm for processing the evolution equation, we use a quasi-stationary approach. Let nt ∈ N be. We put dt = T/nt the time discretization step and for i ∈ 0,..., nt, we note Ci := C(., i ∗ dt). By applying the stable implicit Euler scheme, the sequence of systems to be solved is written :

— We pose $C^0 = C_0 \in H^1(\Omega)$.

— For i ∈ 0,..., nt, solve system (2)

$$\begin{cases} \frac{C^i - C^{i-1}}{dt} + div\left(-D\nabla C^i + C^i\vec{U}\right) = 0 & \text{in } \Omega \\ \left(-D\nabla C^i + C^i\vec{U}\right).\vec{n} = \Psi^i & \text{on } \Gamma_m \\ C^i = \Phi^i & \text{on } \Gamma_m \end{cases} \quad (2)$$

To solve this inverse problem, we use the same numerical scheme of the Kozlov-Maz'ya-Fomin algorithm.

The scheme of the algorithm

— We pose $C^0 = C_0 \in H^1(\Omega)$.

— For i ∈ 0,,,,, nt

— Given an initial approximation $\eta_0 \in H^{1/2}(\Gamma i)$

— For k ≥ 0

— Solve system (3)

$$\begin{cases} \frac{C^i_{2k}}{dt} + div\left(-D\nabla C^i_{2k} + C^i_{2k}\vec{U}\right) = \frac{C^{i-1}_{2k}}{dt} & \text{in } \Omega \\ \left(-D\nabla C^i_{2k} + C^i_{2k}\vec{U}\right).\vec{n} = \Psi & \text{on } \Gamma_m \\ C^i_{2k} = \eta_k & \text{on } \Gamma_i \end{cases} \quad (3)$$

— Pose $\tau_k = \left(-D\nabla C^i_{2k} + C^i_{2k}\vec{U}\right).\vec{n}$

— Solve

$$\begin{cases} \frac{C^i_{2k+1}}{dt} + div\left(-D\nabla C^i_{2k+1} + C^i_{2k+1}\vec{U}\right) = \frac{C^{i-1}_{2k+1}}{dt} & \text{in } \Omega \\ \left(-D\nabla C^i_{2k+1} + C^i_{2k+1}\vec{U}\right).\vec{n} = \tau_k & \text{on } \Gamma_i \\ C^i_{2k+1} = \Phi & \text{on } \Gamma_m \end{cases} \quad (4)$$

— Pose $\eta_k = C^i_{2k+1\|\Gamma_i}$.

4. Program

```
//————————————————————————————————//
// This program is to solve the problem
// transient with Cauchy data :
// c = ((y + 1)2)/6 + 2 ∗ x and Flux (dx(c), c, dy(c))
// The numerical scheme is explained above
//————————————————————————————————//
// construction of the domaine
border gamma1(s = 0, 2 ∗ pi) {x = cos(s); y = sin(s); label = 1; } ;
border gamma0(s = 0, 2 ∗ pi) {x = 0.6 ∗ cos(s); y = 0.6 ∗ sin(s); label = 2; } ;
// construction of the mesh :
mesh Th=buildmesh (gamma1(50) + gamma0(−50)) ;

plot (T h, wait = 1) ;
// construction of Lagrange polynomials on the mesh
fespace V h(T h, P1) ;

//Time data
//T temps final , nt nbre de noeuds , dt= pas temporel
real T = 2, nt = 8, dt, alpha ;
dt = T/nt ;
alpha = 1./dt ;

// The elements of Vh
Vh u1 , u2 , v1 , v2 , oldu1 , oldu2 ;
//u1 solution of the problem where the Dirichlet condition is fixed on gamma1 ( connu )
//u2 solution of the problem where the Neumann condition is fixed on gamma1 ( connu )
//v1 , v2 tests functions .

//Cauchy data defined on gamma1
func c = ((y + 1)2)/6 + 2 ∗ x ;
func dcx = 2 ;
func dcy = (y + 1)/3 ;
```

```
macro cc(s)((T ∗ (s ∗ dt)) + c)//
macro dccx(s)(dcx)//
macro dccy(s)(dcy)//

// coefficients of the matrix D
real alpha1 = 30, alpha2 = 3 ;

// initialization
int k = 0 ;// counter number of iterations
real err = 1 ;
oldu1 = T + c ; // time initialization
oldu2 = T + c ;

//Resolution
for (ints = 1; s < 9; s + +)
{

// definition of the variational problem u1 :
problem FV 1(u1, v1, solver = LU, eps = ∗ 1.0e ∗ 6) = int2d (T h)(al pha ∗ u1 ∗ v1) - int2d(T h)((−al pha1 ∗ dx(u1) + u1) ∗
dx(v1) − al pha2 ∗ dy(u1) ∗ dy(v1)) + on(1, u1 = cc(s)) + int1d(T h, 2)(((−al pha1 ∗ dx(u2) + u2) ∗ N.x − al pha2 ∗ dy(u2)
∗ N.y) ∗ v1) - int2d(T h)(al pha ∗ oldu1 ∗ v1) ;

// definition of the variational problem u2 :
problem FV2(u2, v2, solver = LU, eps = ∗ 1.0e ∗ 6) = int2d (T h)(al pha ∗ u2 ∗ v2) - int2d(T h)((−al pha1 ∗ dx(u2) + u2) ∗
dx(v2) − alpha2 ∗ dy(u2) ∗ dy(v2)) + int1d(T h, 1)(((−alpha1 ∗ dccx(s) + cc(s)) ∗ N.x − alpha2 ∗ dccy(s) ∗ N.y) ∗ v2) +
on(2, u2 = u1) - int2d(T h)(al pha ∗ oldu2 ∗ v2) ;

// initialization problem
problem FV(u2, v2, solver = LU) = int2d (Th) (alpha ∗ u2 ∗ v2) - int2d(T h)((−alpha1 ∗ dx(u2) + u2) ∗ dx(v2) − alpha2 ∗
dy(u2) ∗ dy(v2)) + on(2, u2 = 0) + int1d(T h, 1)(((∗ alpha1 ∗ dccx(s) + cc(s)) ∗ N.x − alpha2 ∗ dccy(s) ∗ N.y) ∗ v2) − int2d(T
h)(alpha ∗ oldu2 ∗ v2) ;
FV ;
while (err > 1e − 5)
{
FV1 ;

// erreur
err = (int1d(T h, 2)(u1 − u2) ∧ 2)/(int1d(T h, 2)(u2 ∧ 2)) ;
FV2 ;
k = k + 1 ;
}
oldu1 = u1 ;
oldu2 = u2 ;
err = 1 ;
}

// storing the results in a file
int ni = 10000 ; //number of points on gamma0 .
real [int] xx(ni + 1), yu1(ni + 1), yu2(ni + 1), yu(ni + 1),
yderivu1(ni + 1), yderivu2(ni + 1), yderivT (ni + 1) ;
int f = 8 ; //at the final moment

for (int i = 0; i < ni + 1; i + +)
{
xx[i] = i ∗ (2 ∗ pi)/ni ; / / discretization of gamma0

// evaluation of u1, u2 and c in these points
yu1[i] = u1(0.6 ∗ cos(xx[i]), 0.6 ∗ sin(xx[i])) ;
yu2[i] = u2(0.6 ∗ cos(xx[i]), 0.6 ∗ sin(xx[i])) ;
yu[i] = cc( f )(0.6 ∗ cos(xx[i]), 0.6 ∗ sin(xx[i])) ;

// evaluation of the outgoing flow.
yderivu1[i] = (- alpha1(u1)(0.6 ∗ cos(xx[i]), 0.6 ∗ sin(xx[i]))+u1(0.6 ∗ cos(xx[i]), 0.6 ∗ sin(xx[i]))) ∗ cos(xx[i]) - al pha2 ∗
```

```
dy(u1)(0.6 ∗ cos(xx[i]), 0.6 ∗ sin(xx[i])) ∗ sin(xx[i]) ;

yderivu2[i] = (- al pha1 ∗ dx(u2)(0.6 ∗ cos(xx[i]), 0.6 ∗ sin(xx[i])) + u2(0.6 ∗ cos(xx[i]), 0.6 ∗ sin(xx[i]))) ∗ cos(xx[i])
- al pha2(u2)(0.6 ∗ cos(xx[i]), 0.6 ∗ sin(xx[i])) ∗ sin(xx[i]) ;

yderivT [i] = (- al pha1∗ dcx+cc( f )(0.6∗ cos(xx[i]), 0.6∗ sin(xx[i])))∗ cos(xx[i]) - al pha2∗ dccy( f )(0.6∗ cos(xx[i]), 0.6 ∗
sin(xx[i])) ∗ sin(xx[i]) ;
}

// registration
ofstream fichier (”valeurs.gp”) ;
for ( int i =0 ; i< ni +1 ; i++)
{
fichier << xx[i] << ”” << yu1[i] << ”” << yu2[i] << ”” << yu[i] << endl ;
}
ofstream fichier1 (" derivnor . gp " ) ;
for (inti = 0; i < ni + 1; i + +)
{
fichier1 << xx[i] << ”” << yderivu1[i] << ”” << yderivu2[i] << ”” << yderivT [i] << endl ;
}
```

5. Results

With this program, we carry out two tests on a crown and a square. So, we approximate the solution of the inverse problem by giving Cauchy data (5) and (7) fixed on $\sum_m$. For the first test, thedata are

$$\begin{cases} \psi_1 &= (T-t+\frac{1}{2}(y+1)^2+2x) \\ \phi_1 &= (-D\nabla\phi_1+\phi_1\vec{U}).\vec{n} \end{cases} \quad (5)$$

The coefficients of the matrix D are:

$$alpha1 = alpha2 = 1 \quad (6)$$

For the second test, the following data are considered:

$$\begin{cases} \psi_2 &= (T-t+\frac{1}{6}(y+1)^2+2x) \\ \phi_2 &= (-D\nabla\phi_2+\phi_1\vec{U}).\vec{n} \end{cases} \quad (7)$$

and the coefficients of the matrix *D*:

$$\begin{cases} alpha1 &= 30 \\ alpha2 &= 3 \end{cases} \quad (8)$$

In the examples tested, the final instant T is equal to 2.
In addition, the stopping criterion is realized when a stationarity of the sequence η_k is reached. That is, when the relative error defined by (9):

$$err(\eta_k, \eta_{k+1}) = \frac{\|\eta_k-\eta_{k+1}\|_{L^2(\Gamma_i)}}{\|\eta_k\|_{L^2(\Gamma_i)}} \quad (9)$$

is smaller than a certain fixed constant in advance. In all numerical tests, we set to $1e^{-05}$. However, the tests carried out below have shown that experimentally the stopping criterion is applicable only in the case of a crown. Indeed, we do not obtain a good approximation with this criterion in the case of a square. For this reason, we make an automatic stop after a hundred iterations, at each time step.
The results of the first test carried out on a crown, and obtained at different times are summarized in the table (1) and the curves are plotted in the figures (**1**). For the second test, it is necessary to consult the table (2) and the figure (2).

Table 1: Analysis of the program for a crown: test 1.

t =	Number of iterations	Relative Error	CPU time
0.5	24	$1.59533e^{-06}$	8.31
1	40	$1.46176e^{-06}$	8.841
1.5	54	$8.15392e^{-06}$	8.516
2	68	$4.48832e^{-06}$	8.08

Table 2: Analysis of the program for a crown: test 2.

t =	Number of iterations	Relative Error	CPU time
0.5	12	$2.81838e^{-06}$	7.462
1	20	$3.05951e^{-06}$	8.471
1.5	28	$3.14228e^{-06}$	7.445
2	36	$2.32936e^{-06}$	7.372

Interpretation 1 *The results of tests 1 and 2 show that we obtain a good convergence when the domain is a crown. The change of the matrix D does not taint the results. In addition, the two tests require 36 iterations with an error of the order of* 10^{-6}.

We use the same test functions as for a crown but we only change the domain to a square. So we start with the data 5 and 6. The results obtained for four different instants are in the table 3. The curves are plotted in the figure 4. In the second test, we consider the data 7 and 8. The curves plotted on Scilab are in the figure 5. The numerical results are in the table 4.

Table 3: Analysis of the program for a square: test 1.

t =	Number of iterations	Relative Error	CPU time
0.5	100	$1.54886e^{-18}$	61.804
1	100	$1.49199e^{-18}$	100.95
1.5	100	$1.47461e^{-18}$	136.469
2	100	$1.45261e^{-18}$	176.202

Table 4: Analysis of the program for a square: test 2.

t =	Number of iterations	Relative Error	CPU time
0.5	100	$1.34578e^{-28}$	61.822
1	100	$4.98735e^{-28}$	98.545
1.5	100	$1.40005e^{-27}$	143.954
2	100	$3.66783e^{-27}$	178.261

Interpretation 2 *In the case of a square, the first test reveals that 100 iterations are satisfactory to obtain a good convergence except in the first test at time 0.5 as indicated in the figure* ***4****. On the other hand, in the second test, although the reconstruction of the concentration is convincing, that of the flow is far from the exact solution; moreover, the even sequence and the odd sequence do not coincide.*

Table 5: Comparison calculation time: test 1.

	t =	Number of iterations	Relative Error	CPU time
Crown	2	100	$7.77186e^{-33}$	19.344
Square	2	100	$1.45261e^{-18}$	176.202

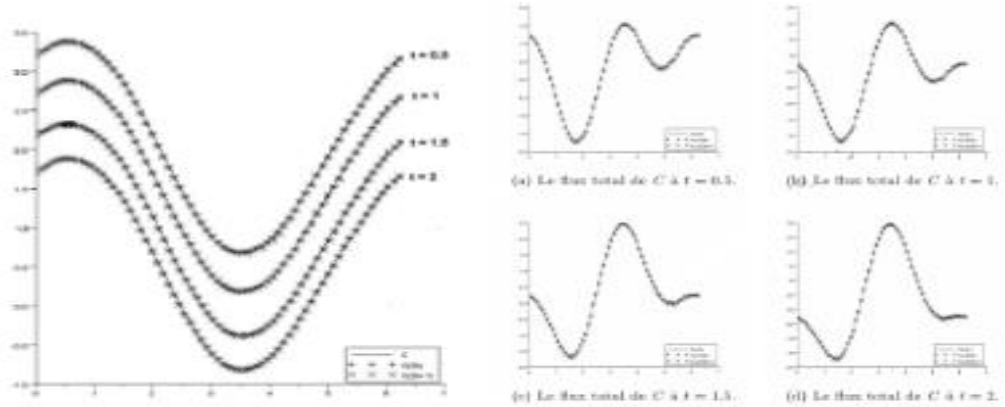

FIGURE 1 – Test 1.

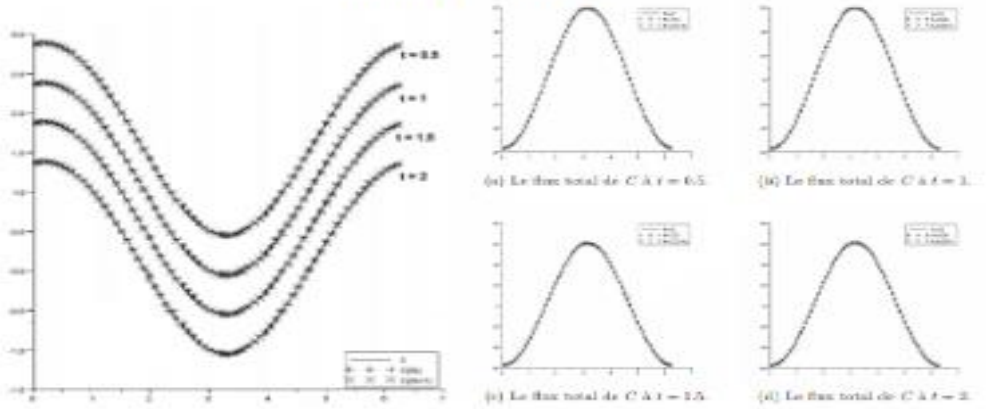

FIGURE 2 – Test 2.

FIGURE 3 – Approximation of the curve (left) and the flow (right) on corona.

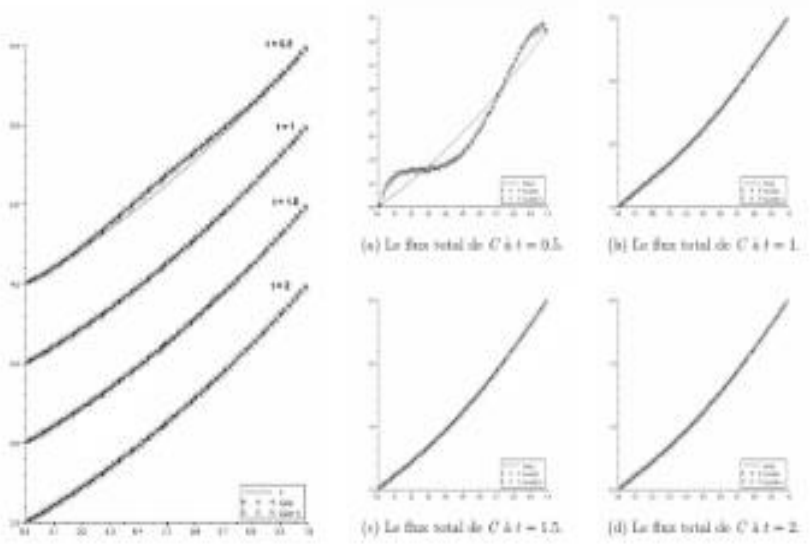

FIGURE 4 – Test 1.

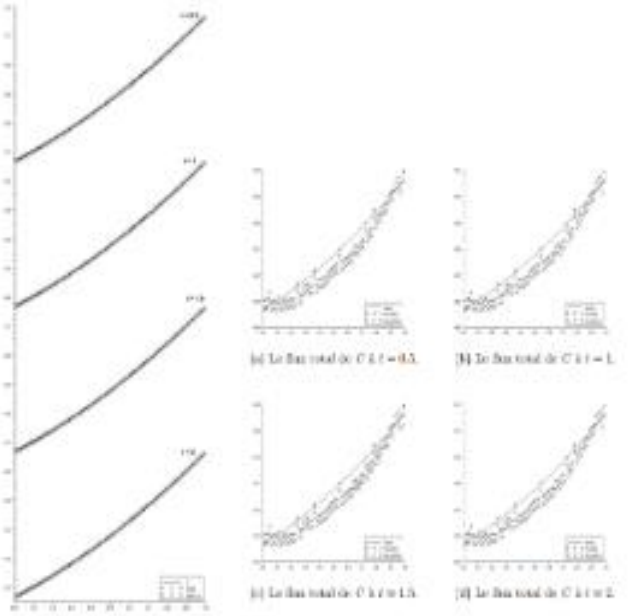

FIGURE 5 – Test 2.

FIGURE 6 – Approximation of the curve (left) and the flow (right) on square.

Interpretation 3

Our first observation concerns the choice of the stopping criterion. Indeed, we note that in the case of a corona it suffices that the relative error is smaller than a certain value fixed in advance to obtain a good convergence, which is not the case for a square domain. For this, we have chosen to make a sudden stop at 100 iterations, in the stationary case this condition is sufficient on the other hand in the transient case this is not always valid as in test 1 at time 0.5. This may be due to the regularity of the domain which is smoother in a crown than in a square.

A second observation which supports the effect of the choice of the domain is the irregularity of the curves, both for the reconstruction of the concentration and for the total flux, noticed in the case of the square domain. We conclude, therefore, that the angles present in a square delay the convergence of the algorithm. This confirms the remark quoted in [14], concerning the regularity of the domain: the more regular the domain, the faster the convergence.

A third observation, emphasizes the same trait, the calculation time for a crown is much less than that of the square, as indicated by the table 5. Although the CPU calculation time depends on the power of the machine, this finding is a motivation for studying the complexity of the algorithm. In addition, this can confirm the dependence of this algorithm on the thickness of the studied domain.

In the case of the crown, it only takes a few iterations, between 2 and 4, to obtain a relative error of the order of 10^{-6}; and even when we extend to 100 iterations in the case of a square, we obtain a relative error which varies between 10^{-26} and 10^{-23}. This does not contradict the slowness of the Kozlov-Mazy'à-Fomin algorithm because the test functions considered are polynomial.

In addition, we note that the choice of the matrix D, and consequently the choice of the operator, does not affect the convergence of the algorithm, nor the quality of the results obtained. We also notice that we cannot predict the variations of the relative error and the CPU time when we provide the Cauchy data or when the program constructs them itself.

Since we are studying a transient case, we have applied a quasi-stationary scheme. We note that the accumulation of errors at each time step does not alter the convergence of the algorithm at the final instant, as illustrated in the figures of the first test with a square domain (4). On the other hand, in the second test, although the concentration curve is reconstructed in the figure (5), that of the flow is poorly approximated as shown in the figure.

6. Conclusion

Finally, this article presents a program on the Frefem++ software, for the numerical resolution of an inverse problem. Thus, theoretically, we applied the Kozlov-Maz'ya-Fomin algorithm to the presented data completion problem. In practice, we have varied different parameters: the domains, the Cauchy data and the coefficients of the diffusion-dispersion matrix. We have shown that the stationarity of the relative error is not a valid stopping condition for the two domains: crown and square. In addition, we numerically show the importance of the regularity of the domain as well as its thickness. In addition, we find that the choice of the matrix does not affect the results.

7. References

1. M Kern. Problèmes Inverses, 2002-2003. http://www-rocq.inria.fr/~kern/ Teaching/ESILV/inverse.pdf
2. Y Annabi. Introduction aux mathématiques appliquées, Editions Universitaires Européennes, 2012, ISBN: 978-3-8417-8154-3.
3. MI Edouard, ML Doris, BB Paulin. Modélisation physique et mathématique de l'écoule- ment sanguindans l'anevrisme de l'aorte abdominale, International Journal of Innovation and Scien- tific Reasearch, 2016, 26(1).
4. Y Annabi. Hemodynamics: Macroscopic and Microscopic Modeling, International Journal of Innovation in Science and Mathematics, 2022, 10(4).
5. J Happel, et H. Brenner. Low Reynolds number hydrodynamics, Martinus Nijhoff Publishers, 1983.
6. G ADDE0 Méthodes de Traitement d'Image Appliquées au Problème Inverse en Magnéto- Electro-Encéphalographie, École nationale des ponts et chaussées, 2005.
7. Jishan Fan, Michele Di Cristo, Yu Jiang, Gen Nakamura, Inverse viscosity problem for the NavierS- tokes equation, Journal of Mathematical Analysis and Applications. 2010; 365(2):750-757.
8. M Bonnet. Problèmes inverses, 2008.
9. Y Annabi. Introduction aux écoulements en canalisations et milieux poreux, Editions Univer- sitaires Européennes, 2022, ISBN: 978-620-3-43992-2.
10. Y Annabi. Introduction aux opérateurs linéaires et aux opérateurs non linéaires, Book, Editions Universitaires Européennes, 2016, ISBN: 978-3-639-50779-9.
11. Y Annabi. Introduction à la modélisation multi-échelle, Editions Universitaires Européennes, 2022, ISBN: 978-6-139-50445-9.

12. Y Annabi. Introduction à l'hémodynamie, Editions Universitaires Européennes, 2022, ISBN: 978-620-3-43779-9.
13. F Hecht. *Freefem++*, 2010. http://www.freefem.org/ff++/ftp/freefem++doc. pdf
14. M Farah. Problèmes inverses de sources et lien avec l'électro-encéphalo-graphie. PhD thesis, Université de technologie de Compiègne, 2007.