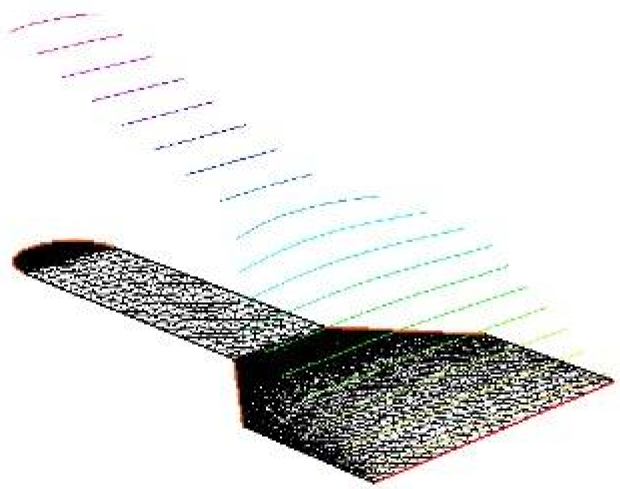
UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE



SIMULATION DE DESCENTE DE FUSÉE EN FREEFEM++

Fait par :
EL BAHRAOUI Imade
KHADRAOUI Mohamed El Bachir
(CHPS)

Introduction

L'objectif de cette simulation est d'observer la descente d'une fusée vers la terre, notamment des températures aux bords de celle-ci. La vitesse de la fusée et l'épaisseur du bouclier thermique déterminant ces températures, elles seront les deux valeurs à faire varier afin d'expérimenter les différentes situations possibles dans cette simulation. Freefem, un outil de simulation permettant la résolution numérique d'équations différentielles et de représentation de formes servira à former le maillage de la fusée ainsi que la définition du solveur permettant d'obtenir les températures au bord de l'engin. Python sera utilisé par la suite afin de procéder à la validation des résultats numériques obtenus avec FreeFem.

1. Partie Freefem

1.1 Construction du maillage :

Le maillage a été construit en suivant une approche géo-schématique en définissant 9 points caractéristiques dans l'espace en plus d'un point représentant le centre du demi-cercle (nez de la fusée).

```
real x1 = 2.0, y1 = 1.0;
real x2 = -2.0, y2 = 1.0;
real x3 = -2.0, y3 = -1.0;
real x4 = -0.75, y4 = -2.5;
real x5 = -0.75, y5 = -6.5;
real x6 = 0.75, y6=-6.5;
real x7 = 0.75, y7 = -2.5;
real x8 = 2.0, y8 = -1.0;
real x9 = x1, y9= y1;

real x0 = 0, y0 = y5; //Demi-cercle
```

Figure 1.1 - Spécification des points

Les bords de Dirichlet du maillage ont été spécifiés avec un label égalant 1, quant aux bords de Fourier, leur label est égal à 2.

```
border a(t=-pi, 0){x=x0+0.75*cos(t); y=y0+0.75*sin(t); label=2;} //Demi-cercle
border b1(t=0,1) {x = (1-t)*x1 + t*x2;y = (1-t)*y1 + t*y2;label= 1;}
border b2(t=0,1) {x = (1-t)*x2 + t*x3;y = (1-t)*y2 + t*y3;label= 0;}
border b3(t=0,1) {x = (1-t)*x3 + t*x4;y = (1-t)*y3 + t*y4;label= 2;}
border b4(t=0,1) {x = (1-t)*x4 + t*x5;y = (1-t)*y4 + t*y5;label= 0;}
border b5(t=0,1) {x = (1-t)*x6 + t*x7;y = (1-t)*y6 + t*y7;label= 0;}
border b6(t=0,1) {x = (1-t)*x7 + t*x8;y = (1-t)*y7 + t*y8;label= 2;}
border b7(t=0,1) {x = (1-t)*x8 + t*x9;y = (1-t)*y8 + t*y9;label= 0;}

mesh th = buildmesh(b1(20) + b2(20) + b3(20) + b4(20) + a(20) + b5(20) + b6(20) + b7(20));
```

Figure 1.2 - Définition des bords du maillage

Le maillage obtenu est le suivant :

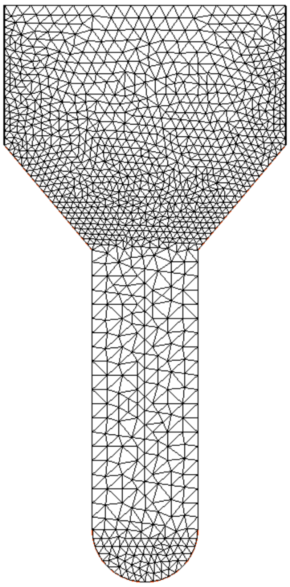


Figure 1.3 - Maillage de la fusée

1.2 Définition du solveur :

Les données de la simulation sont les suivantes :

- **f : La fonction source**
- **V : La vitesse en mètre par seconde (m/s)**
- **e : L'épaisseur en mètre (m)**
- k0 : La conductivité thermique de l'air
- k1 : La conductivité thermique du matériau utilisé pour le bouclier thermique (l'amiante dans notre cas)
- alpha : $\frac{k1}{e}$
- Beta : 10^8
- uE : $10^{-3} * V^2$
- uD : 20 (en degrés Celsius)

```
func f = -0.5*exp(-(x^2 + y^2)); //Fonction source
int V = 100; //Vitesse m/s
real e = 0.001; //Epaisseur du bouclier en m
real k0 = 0.026; //Conductivité de l'air
real k1 = 0.168; // Conductivité thermique de l'amiante
real alpha = k1/e
real beta = 10^8;
real uE = 0.001 * (V^2);
real uD = 20;
```

Figure 1.4 - Données du solveur

Après la construction du maillage vient la définition du solveur pour la mise en place de la formulation variationnelle qui permettra au final d'obtenir uh qui elle contiendra les températures aux bords de la fusée lors de sa descente vers la terre en fonction de la vitesse **V** et de l'épaisseur du bouclier **e**, ces deux derniers paramètres étant à **varier** dans la simulation afin de pouvoir observer les résultats.

La formulation variationnelle dans le cas de cette simulation est la suivante :

$$\int\int K\nabla u * \nabla v * dx + \int \alpha * U * Vds + \int \beta * U * Vds = \int\int f * dx + \int \alpha * Ue * Vds + \int \beta * Ud * Vds$$

Avec :

- α : Correspondant à alpha
- β : Correspondant à Beta
- $\int \alpha * Ue * Vds$: Terme de bord de Fourier
- $\int \beta * Ud * Vds$: Terme de bord de Dirichlet

Le solveur au niveau de Freefem possède la forme suivante :

```
fespace Vh(th, P1); // Espace EF
Vh uh, vh;

problem PbLAPLACE(uh, vh, solver = Cholesky) // Mise en place de la formulation variationnelle
    = int2d(th)(dx(uh)*dx(vh) + dy(uh)*dy(vh))
    - int2d(th)(f*vh)
    +int1d(th,2)(alpha*uh*vh) //Ajout des termes de bords ayant pour condition Fourier-Robin
    -int1d(th,2)(alpha*uE*vh)
    +int1d(th,1)(beta*uh*vh) //Ajout de la pénalisation dans le cas de la condition de Dirichlet
    -int1d(th,1)(beta*uD*vh);

PbLAPLACE;
```

Figure 1.5 - Solveur FreeFem

L'affichage du maillage avec une représentation visuelle des températures permet d'obtenir le schéma suivant :

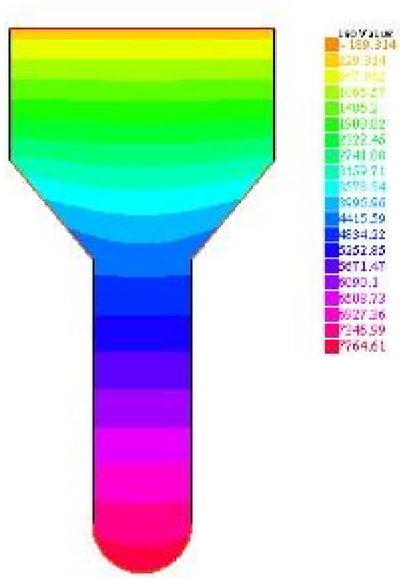
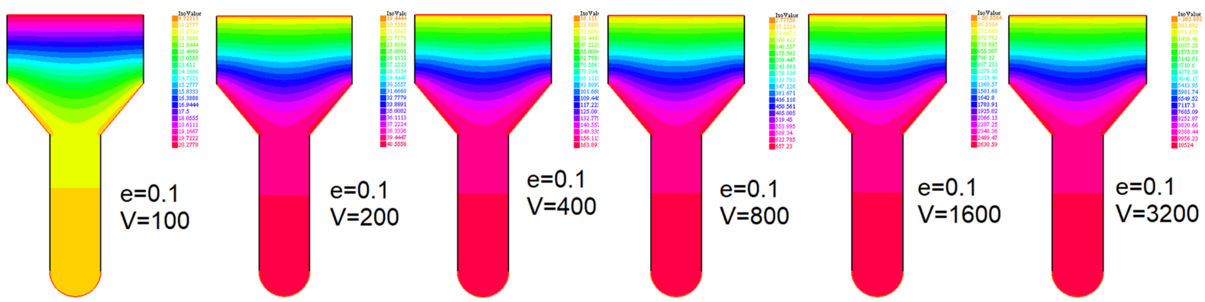


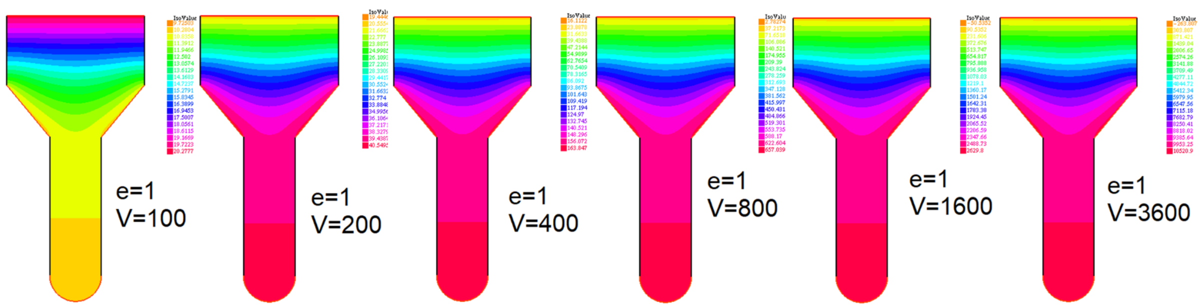
Figure 1.6 - Maillage avec température

Dans les figures suivantes, des variations de la vitesse **V** et l'épaisseur du bouclier **e** ont appliquées afin d'observer les résultats :

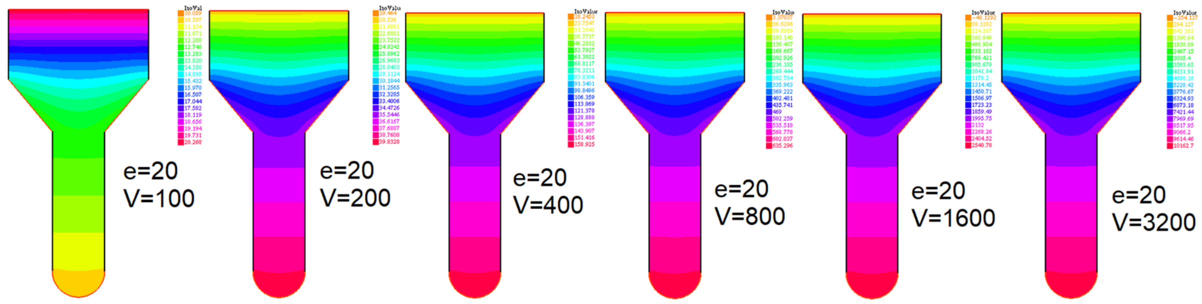
Pour e = 0.1cm :



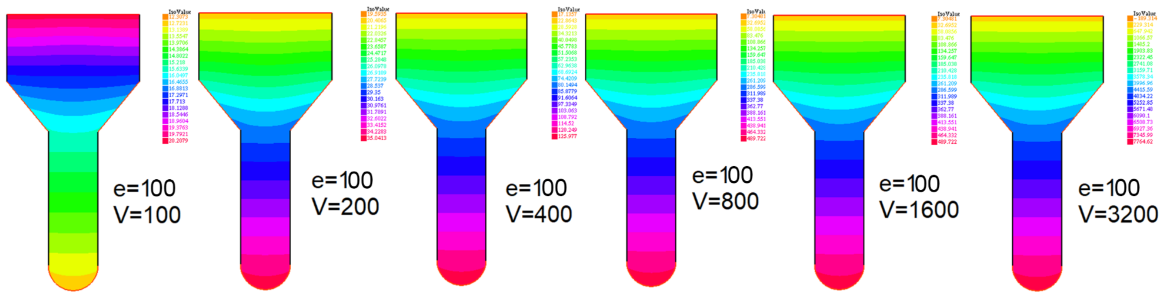
Pour e = 1cm :



Pour e = 20cm :



Pour e = 100cm :



1.3 Observations :

Dans les schémas ci-dessus, le rouge désigne la température la plus élevée tandis que l'orange lui représente la moins élevée.

Lorsqu'on fixe une épaisseur **e**, plus la vitesse **V** augmente, plus la température augmente aux bords de la fusée.

Il est possible de constater que pour une vitesse fixée, plus l'épaisseur augmente, plus la température baisse aux bords de la fusée.

Le bord de Dirichlet a une température de 20 degrés à chaque début de cas de variation de vitesse pour une épaisseur fixe comme cela a été défini (uD = 20). La température de ce bord seul diminue avec l'augmentation de la vitesse de la fusée.

2. Partie Python

2.1 Préparation de l'Algorithme d'assemblage

La partie Python a pour but d'effectuer les calculs de températures en utilisant le fichier *.msh* généré de la première partie et d'effectuer une comparaison avec les résultats obtenus précédemment . Pour ce faire, le code du TP2 a été repris. Afin de s'accorder à notre cas d'étude, les apports sont les suivants lui ont été fait :

- Redéfinition de la fonction source f
- Définition de la fonction uE avec pour paramètre la vitesse **V**
- Définition d'une fonction pour retourner k1
- Définition d'une fonction prenant l'épaisseur **e** comme argument pour retourner alpha
- Définition d'une fonction pour retourner Beta
- Définition de la fonction retournant uD égalant 20
- Définition d'une fonction pour retourner k0

```
def fct_uE(V):
    return 0.001 * (V**2)

def fct_f(x : float, y : float):
    result = -0.5*exp(-(x**2 + y**2))
    return result

def fct_k1():
    return 0.168

def fct_alpha(e):
    k1 = 0.168
    return k1/e

def fct_beta():
    return 10**8

def fct_uD():
    return 20

def fct_k0():
    return 0.026
```

Figure 2.1 - Données calculs coefficients

Afin que l'algorithme d'assemblage s'exécute correctement, il faut rajouter le terme de Dirichlet aux matrices A et F. Pour faire cela, il faut calculer les coefficients de ce bord, à savoir, le poids et le flux extérieur de la même manière que cela a été effectué pour les termes de Fourier-Robin. Il a donc fallu en faire les fonctions.

```
def coeffelem_P1_poids_dirichlet(self,alpha):

        nbn, nbe, nba, coord, tri, ar, refn, reft, refa =
        lit_fichier_msh(self.NomduFichier)

        ai1, ai2 = [int(j) for j in ar[alpha]] #Les indices de chaque
        sommets de l'arete

        a1 = [coord[ai1,0], coord[ai1,1]]

        a2 = [coord[ai2,0], coord[ai2,1]]

        mes = sqrt(abs((a2[0] - a1[0])**2 - (a2[1] - a1[1])**2))

        val = mes/6


        mat = [[2,1],
                [1,2]]

        p = [[0,0],[0,0]]


        for i in range(0,2):
            for j in range(0,2):

                p[i][j] = (val * fct_beta() * mat[i][j])


        return p
```

Figure 2.2 - Calcul coefficient de poids du terme de Dirichlet

```
def coeffelem_P1_transf_dirichlet(self,alpha):
        nbn, nbe, nba, coord, tri, ar, refn, reft, refa =
        lit_fichier_msh(self.NomduFichier)

        ai1, ai2 = [int(j) for j in ar[alpha]] #Les indices de chaque
        sommets de l'arete

        a1 = [coord[ai1,0], coord[ai1,1]]

        a2 = [coord[ai2,0], coord[ai2,1]]

        mes = sqrt(abs((a2[0] - a1[0])**2 - (a2[1] - a1[1])**2))

        val = mes/2

        mat = [1,1]
        p =[]

        for i in range(0,2):

            p.append(val * fct_beta() * fct_uD() * mat[i])

        return p
```

Figure 2.3 - Calcul coefficient du flux extérieur du terme de Dirichlet

Dans l’addition des bords de l’algorithme d’assemblage, les termes de Fourier-Robin sont récupérés grâce à leur label égalant 2 et celui de Dirichlet grâce au label 1 et ce en utilisant une condition.

```
if(int(refa[i]) == 2):

  p = self.coeffelem_P1_poids_fourier(i)
  e = self.coeffelem_P1_transf_fourier(i)

  i1 = ar[i][0]
  i2 = ar[i][1]

  A[i1][i1] += p[0][0]
  A[i1][i2] += p[0][1]
  f[i1] += e[0]

  A[i2][i1] += p[1][0]
  A[i2][i2] += p[1][1]
  f[i2] += e[1]

if(int(refa[i]) == 1):

  p = self.coeffelem_P1_poids_dirichlet(i)
  e = self.coeffelem_P1_transf_dirichlet(i)

  i1 = ar[i][0]
  i2 = ar[i][1]

  A[i1][i1] += p[0][0]
  A[i1][i2] += p[0][1]
  f[i1] += e[0]

  A[i2][i1] += p[1][0]
  A[i2][i2] += p[1][1]
  f[i2] += e[1]
```

Figure 2.4 - Ajout des bords

Après exécution de l'algorithme d'assemblage, les résultats obtenus sont les suivants :

Pour e = 0.1cm :

Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :
100	200	400	800	1600	3200
Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :
0.1	0.1	0.1	0.1	0.1	0.1
min Uh =	min Uh =	min Uh =	min Uh =	min Uh =	min Uh =
-0.039165402988206456	-0.03916540298228848	-0.03916540295861657	-0.03916540286392891	-0.03916540248517828	-0.039165400970175894
max Uh =	max Uh =	max Uh =	max Uh =	max Uh =	max Uh =
19.999999796010226	39.90668705677661	159.6267482271064	638.5069929084256	2554.0279716337022	10216.111886534809

Pour e = 1cm :

Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :
100	200	400	800	1600	3200
Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :
1.0	1.0	1.0	1.0	1.0	1.0
min Uh =	min Uh =	min Uh =	min Uh =	min Uh =	min Uh =
-0.03916540298875136	-0.03916540298446811	-0.0391654029673351	-0.03916540289880303	-0.039165402624674786	-0.03916540152816175
max Uh =	max Uh =	max Uh =	max Uh =	max Uh =	max Uh =
19.999999796010226	37.94459663766362	151.7783865506544	607.1135462026175	2428.45418481047	9713.81673924188

Pour e = 20cm :

Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :
100	200	400	800	1600	3200
Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :
20.0	20.0	20.0	20.0	20.0	20.0
min Uh =	min Uh =	min Uh =	min Uh =	min Uh =	min Uh =
-0.0391654029899332	-0.03916540298919546	-0.03916540298624445	-0.039165402974440426	-0.03916540292722427	-0.039165402738359
max Uh =	max Uh =	max Uh =	max Uh =	max Uh =	max Uh =
19.999999796010226	19.999999796010226	56.992753692523415	227.97101477009392	911.884059080376	3647.536236321504

Pour e = 100cm :

Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :	Vitesse en m/s :
100	200	800	400	1600	3200
Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :	Epaisseur en cm :
100	100	100	100	100	100
min Uh =	min Uh =	min Uh =	min Uh =	min Uh =	min Uh =
-0.03916540299012384	-0.03916540298995801	-0.03916540298664143	-0.03916540298929469	-0.03916540297602838	-0.0391654029335759
max Uh =	max Uh =	max Uh =	max Uh =	max Uh =	max Uh =
19.999999796010226	19.999999796010226	61.84538977254302	19.999999796010226	247.38155909017254	989.5262363606905

2.2 Observations :

minUh et maxUh désignent respectivement la température minimum et la température maximum obtenue. minUh représente la température au bord de Dirichlet et maxUh représente la température au nez de la fusée.

Ces résultats sont cohérents. Pour chaque épaisseur fixée, plus la vitesse augmente, et plus la température augmente aussi.

Il est possible de constater que pour une vitesse fixée, plus l'épaisseur augmente et plus la température diminue.

Les résultats ne sont pas égaux à ceux obtenus avec FreeFem, toutefois, ils s'en rapprochent grandement.

Conclusion :

Les résultats obtenus via FreeFem et Python sont valides dans le cas d'une descente d'une fusée. La température augmente avec l'augmentation de la vitesse sur tous les bords sauf sur celui de Dirichlet qui est le plus froid vu qu'il ne connaît pas de friction avec l'air lors de la descente, ce qui est le contraire du bord le plus chaud qui est celui du nez de la fusée.