# Learning to Fly

Soutenance de fin de projet

*13 mars 2017*

Etienne Herlaut
Naïl Eljafaari
Houssam Akhmouch

Tâm Le Minh
Adrien Bufort
Ludovic Becq

# Le programme

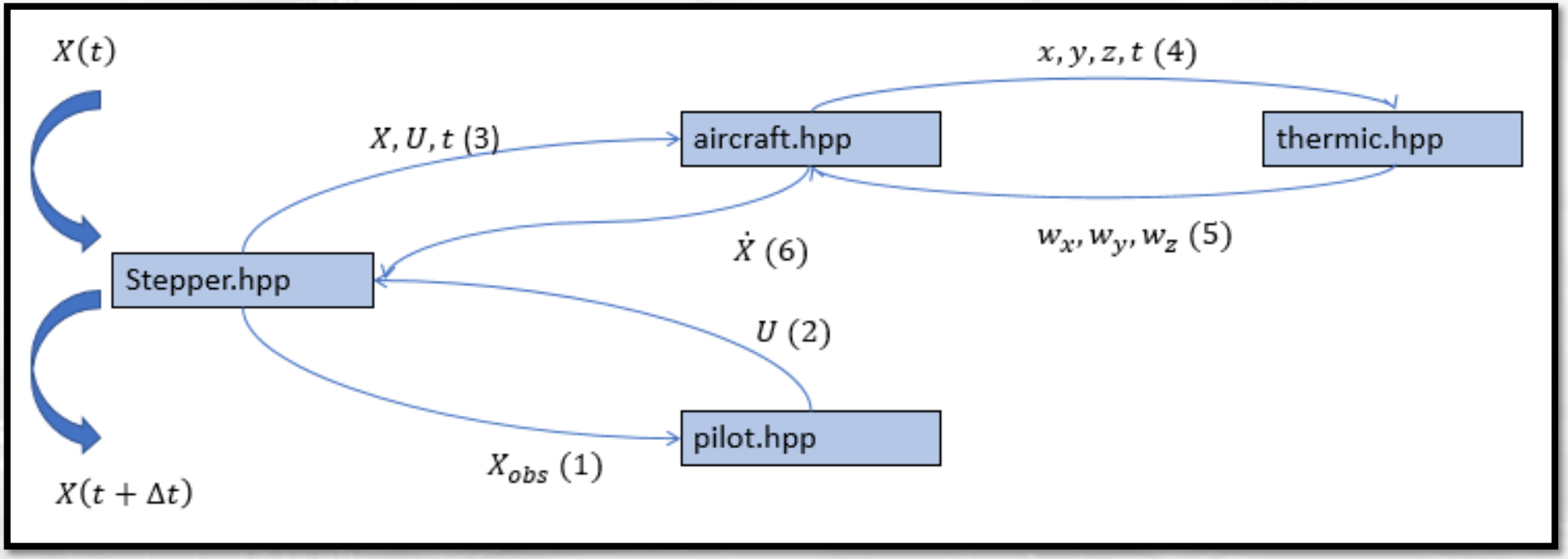
- Principale contribution du projet
- Caractéristique du code :
  - Modulable
  - Documenté
  - Simple d'utilisation
  - C++ (vitesse)
- Simulateur de vol avec :
  - Modélisation des thermiques
  - Modélisation des lois de commandes
  - Modélisation de la dynamique du vol

# Repository

git clone git@github.com:erachelson/l2f-sim.git

- L2Fsim
  - Aircraft
  - Fligth_zone
  - Pilot
  - Stepper
- python_plot
- Demo
  - main.cpp

# Diagramme de classe



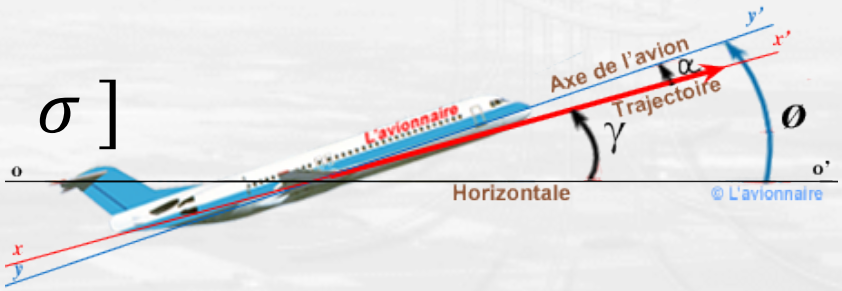
# Modélisation de Aircraft

- Modélisation de la dynamique de vol
- Beeler et al. (2003) :
  - Planeur caractérisé par envergure, allongement et masse
  - Voit le vent relatif comme un point matériel
- Vecteur d'état :

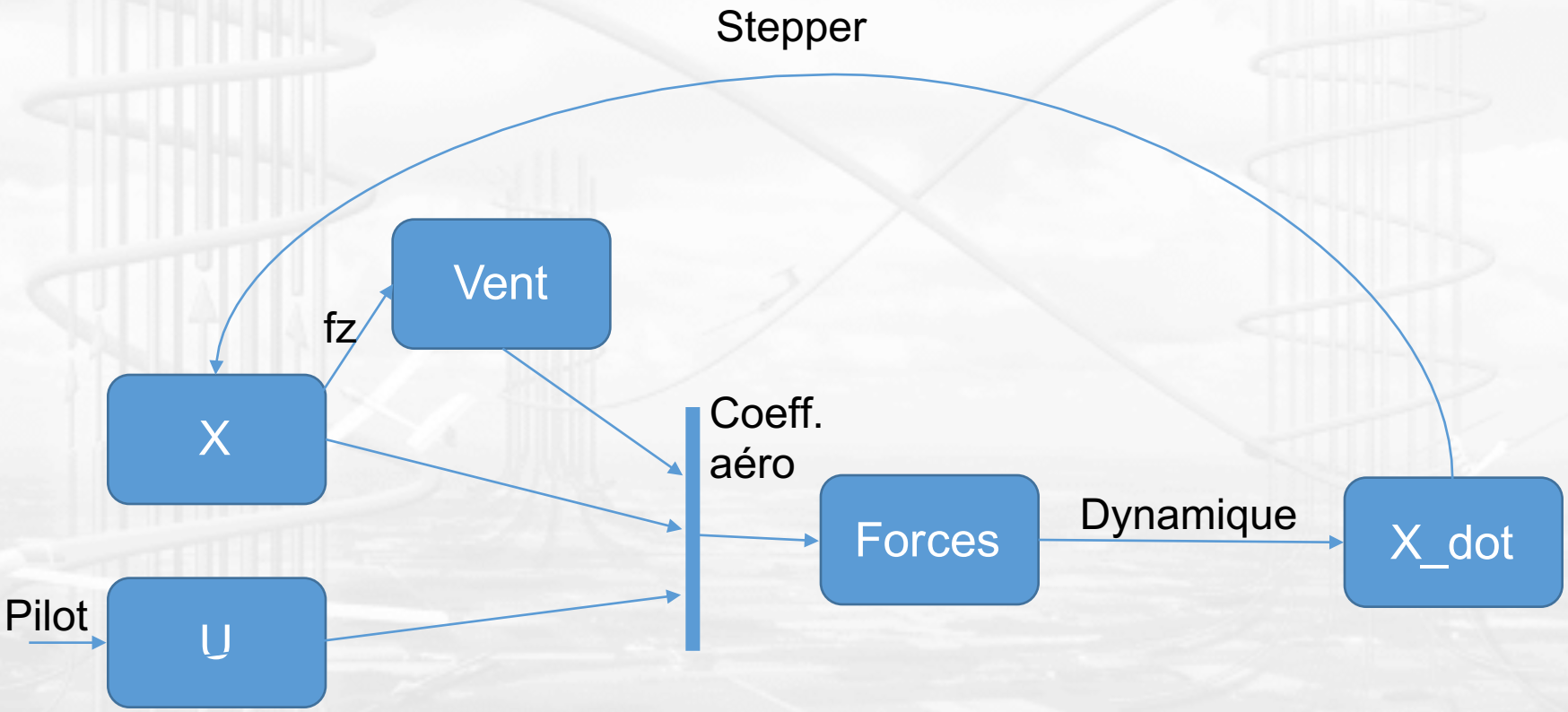
$$X = [\, x \;\; y \;\; z \;\; V \;\; \gamma \;\; \chi \,]$$

- Vecteur de commande :

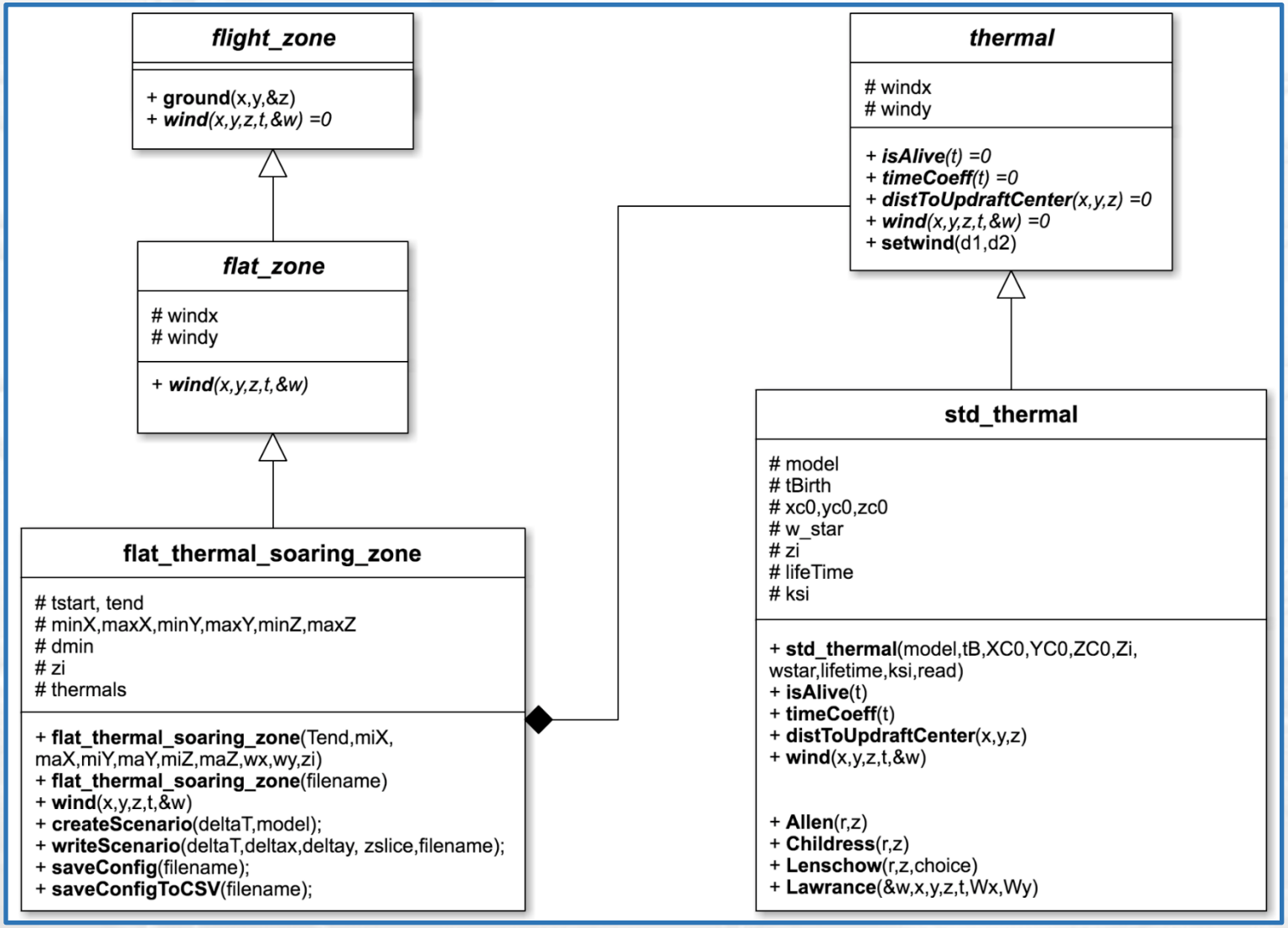
$$U = [\, \alpha \;\; \beta \;\; \sigma \,]$$



# Diagramme



# Diagramme Flight Zone



# Flight_zone (1)

Il y a un *main* propre à la flight_zone dans demo/

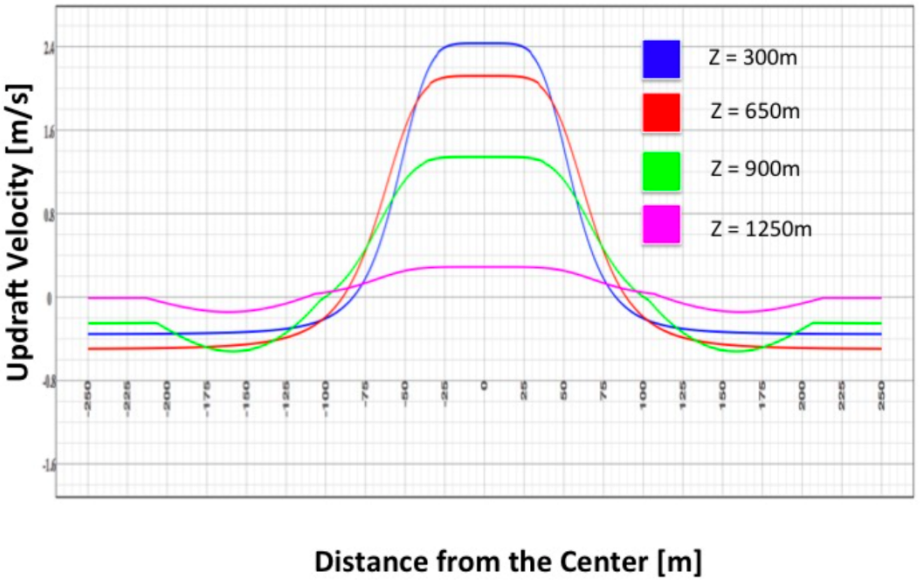
Voici les paramètres sur lesquelles nous pouvons jouer pour définir une flight_zone :

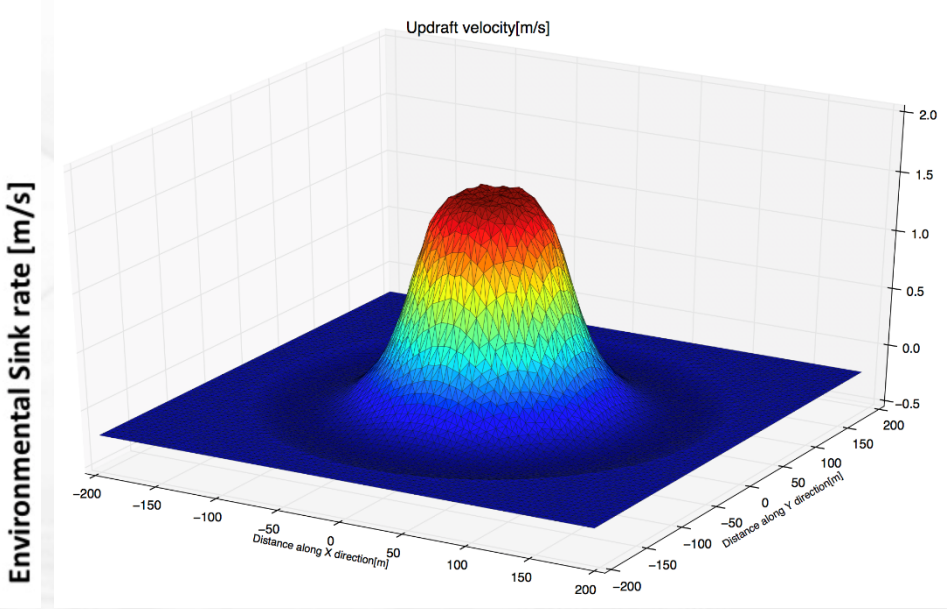
```
/*-------------------------------------
 ------------ FLIGHT ZONE -------------
 -------------------------------------*/
double time_limit = 1000;
double deltaT     = 100;  // actualization of thermals
int minX  = -1000;        // definition of box
int maxX  = 1000;         // definition of box
int minY  = -1000;        // definition of box
int maxY  = 1000;         // definition of box
int minZ  = 0;            // definition of box
int maxZ  = 2000;         // definition of box
double wx = 0.;           // definition of wind
double wy = 0.;           // definition of wind
int zi    = 1400.;        // definition of height of all thermals

int model = 1;            // definition of the model of thermals
// 1 : Allen     model
// 2 : Childress model
// 3 : Lenschow  model
// 4 : Geodon    model
// 5 : Lawrance  model
```

```
//----------------------//
// 1 : create an empty box
//----------------------//
flat_thermal_soaring_zone FZ(time_limit,minX,maxX,minY,maxY,minZ,maxZ,wx,wy,zi);
```

# Flight_zone (1)

## Modèle Allen

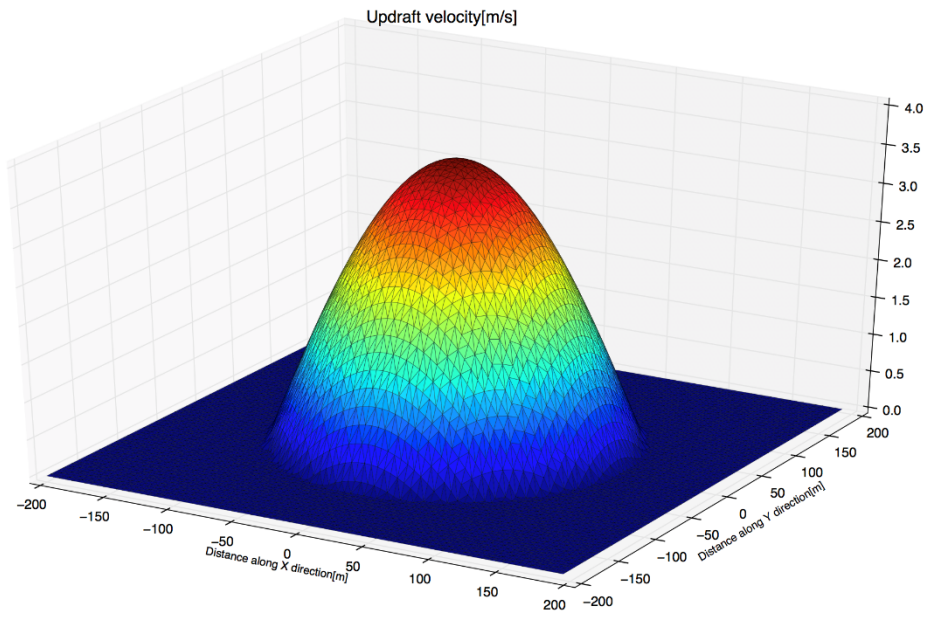


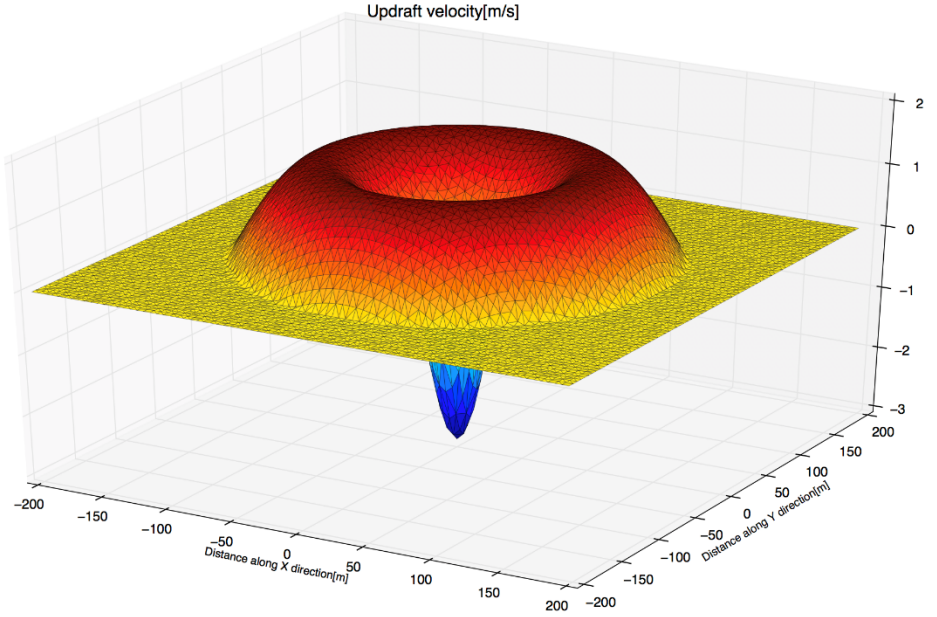

*z = 650*

# Flight_zone (1)

## Modèle Childress



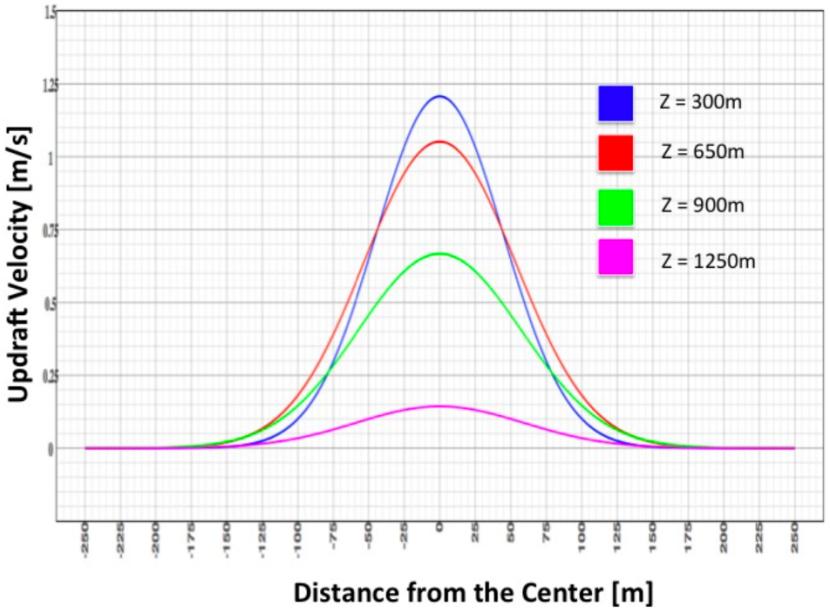

*z = 300*

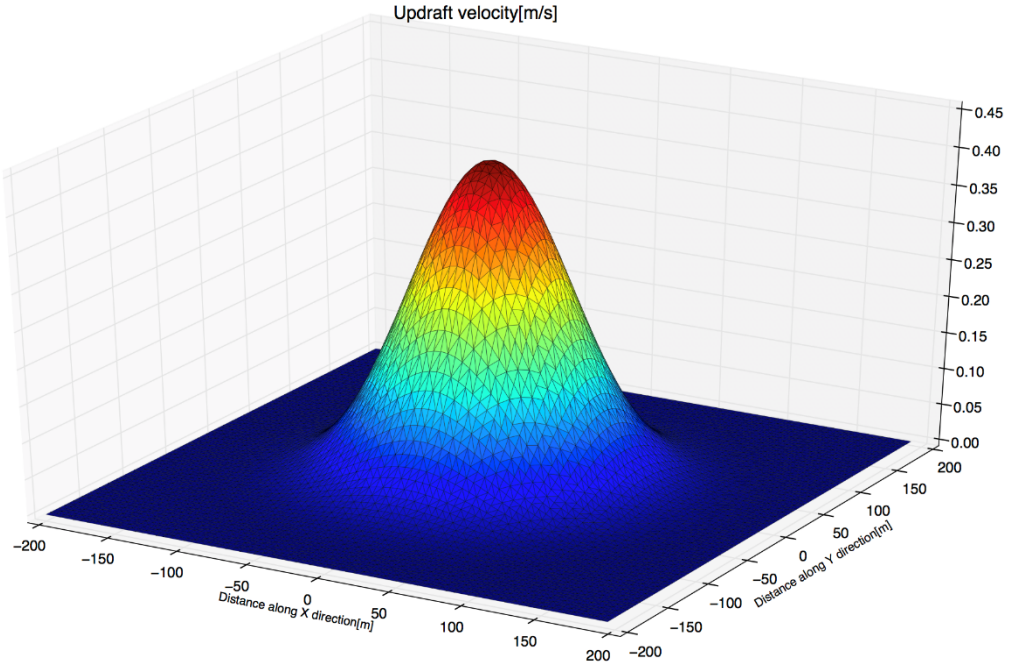



*z = 900*

# Flight_zone (1)

## Modèle Lenshow

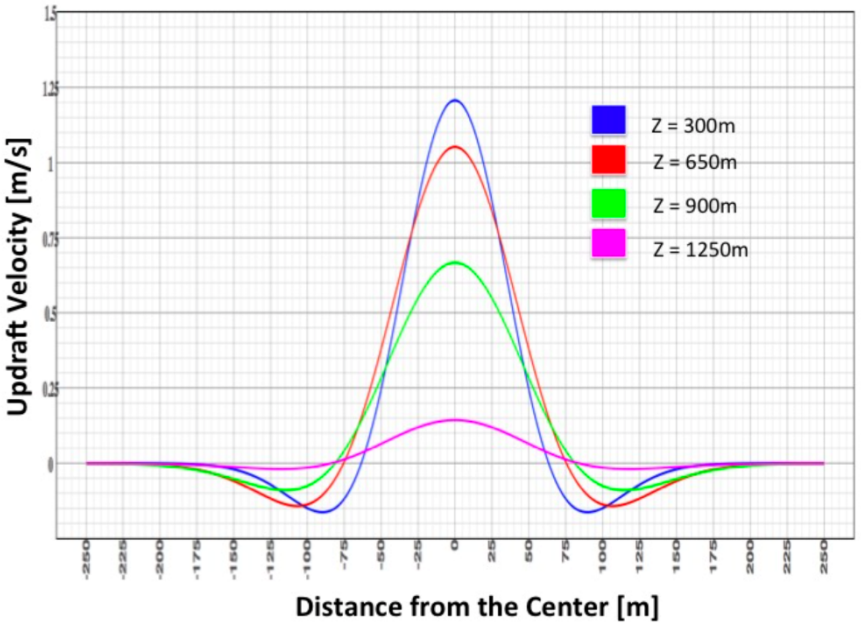


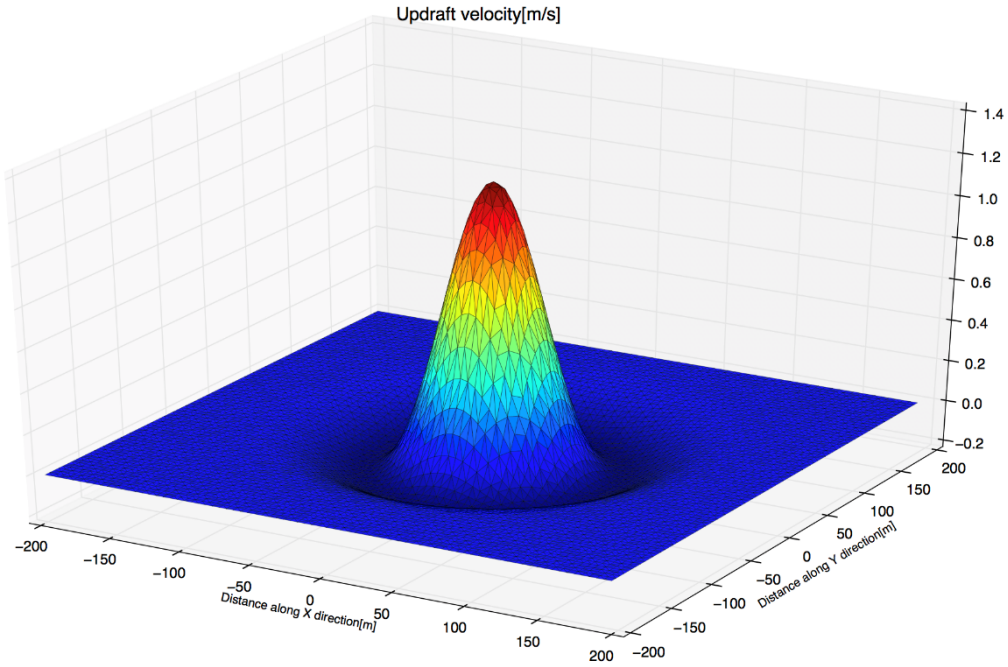




*z = 900*

# Flight_zone (1)

## Modèle Gordon

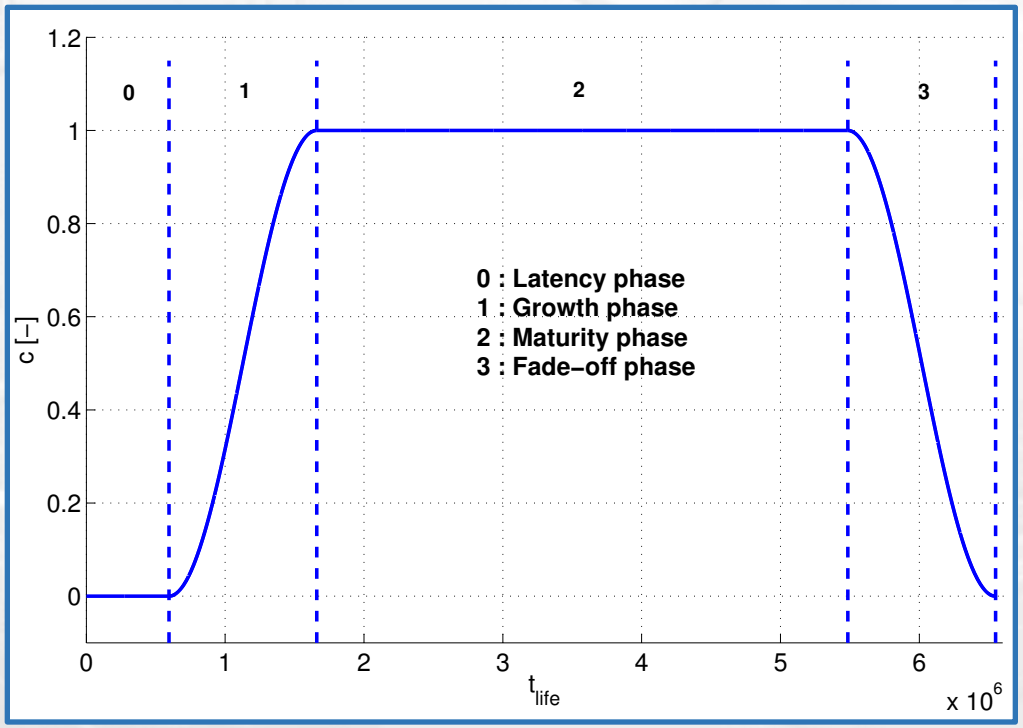



*z = 300*

# Flight_zone (2)

```
//-------------------------------------//
// 2 : Simulate a scenario of thermals
//-------------------------------------//
FZ.createScenario(deltaT,model);
```



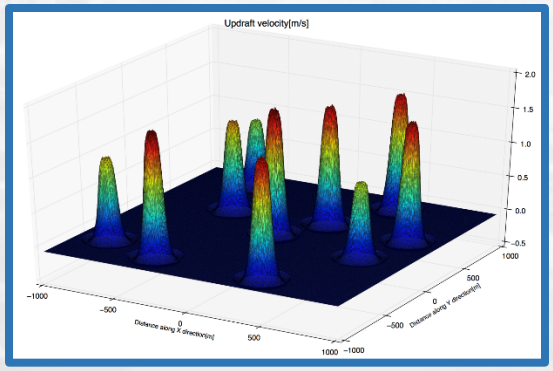
# Flight_zone (3)

```
//---------------------------------------//
// 3 : write DATA for visualization zslice
//---------------------------------------//
// Write the whole wind DATA of a zslice in a file
double deltax = 5.;     // definition of the mesh precision in x direction
double deltay = 5.;     // definition of the mesh precision in y direction
double zslice = 650.;   // height of the windfield you want to write
FZ.writeScenario(deltaT,deltax,deltay,zslice,"DATA/wind.txt");
```

Commandes therminal :

➔ make therm
(va compiler et executer FZmain.cpp)

➔ make thermvis
(va lancer un script python permettant de visualiser DATA/wind.txt)



# Flight_zone (4)

```
//---------------------------//
// 4 : save this configuration
//---------------------------//
FZ.saveConfig("DATA/config2.txt");
```

Avec cette commande vous avez sauvegardé la configuration que vous avez créée avec createScenario() dans *DATA/config2.txt*

Vous pouvez aller le chercher dans le dossier et même le modifié à la main, par exemple changer le centre d'une thermique ? Ou autres…

# Flight_zone (5)

Il est possible une flight_zone uniquement grâce au fichier créé en (4).
La preuve:

```
//---------------------------------//
// 5 : call a previous configuration
//---------------------------------//
// Constructor 2
//flat_thermal_soaring_zone FZ("DATA/config2.txt");
//flat_thermal_soaring_zone FZ("DATA/configUneSeuleTherm.txt");
```

Choisissez l'une des deux dernières lignes et remplacer alors la ligne de (1) par cette dernière, ainsi par exemple :

```
//-------------------------//
// 1 : create an empty box
//-------------------------//
flat_thermal_soaring_zone FZ("DATA/config2.txt");
```

Commentez la ligne (2), de cette façon :

```
//-----------------------------------//
// 2 : Simulate a scenario of thermals
//-----------------------------------//
FZ.createScenario(deltaT,model);
```

# Flight_zone (5)

Le *main* doit alors prendre la forme suivante :

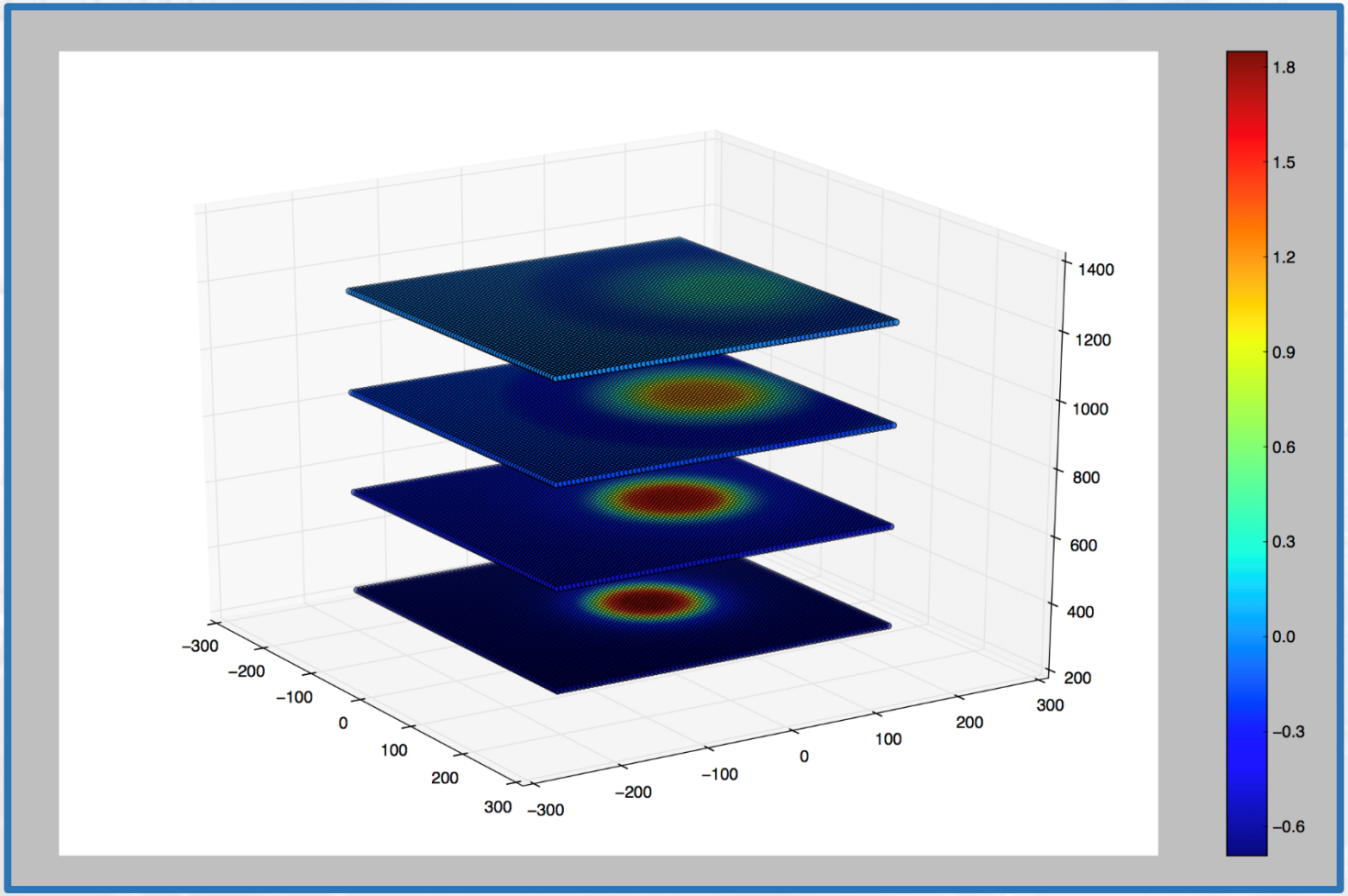
```
//------------------------//
// 1 : create an empty box
//------------------------//
flat_thermal_soaring_zone FZ("DATA/config2.txt");

//-----------------------------------//
// 2 : Simulate a scenario of thermals
//-----------------------------------//
//FZ.createScenario(deltaT,model);

//---------------------------------------//
// 3 : write DATA for visualization zslice
//---------------------------------------//
// Write the whole wind DATA of a zslice in a file
double deltax = 5.;    // definition of the mesh precision in x direction
double deltay = 5.;    // definition of the mesh precision in y direction
double zslice = 650.;  // height of the windfield you want to write
FZ.writeScenario(deltaT,deltax,deltay,zslice,"DATA/wind.txt");

//--------------------------//
// 4 : save this configuration
//--------------------------//
//FZ.saveConfig("DATA/config2.txt");

//-----------------------------------//
// 5 : call a previous configuration
//-----------------------------------//
// Constructor 2
//flat_thermal_soaring_zone FZ("DATA/config2.txt");
//flat_thermal_soaring_zone FZ("DATA/configUneSeuleTherm.txt");
```

Relancer alors le programme et la visualisation :

- make therm
- make thermvis

# Flight_zone

## Effet du vent horizontal



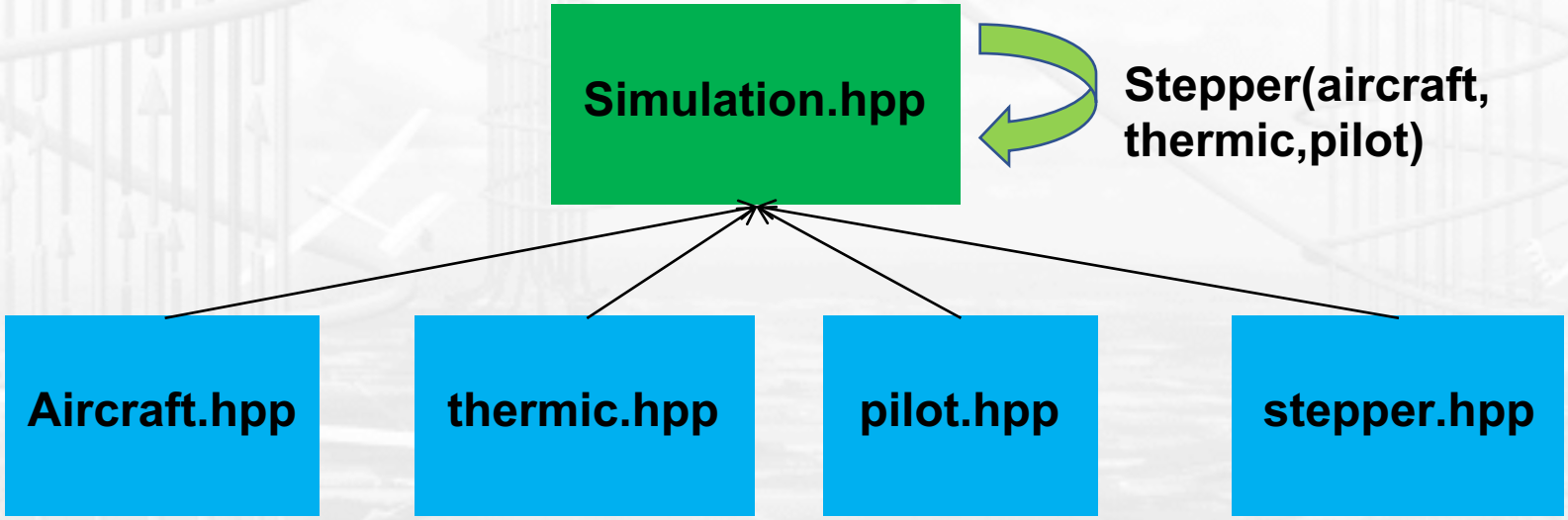
# Stepper

- Gère l'occurance de la loi de pilotage et l'intégration temporelle du modèle physique

- Intégration temporelle
  - Schéma d'Euler
  - Runge-kutta

# Lancer une simulation

- On charge les classes de modèles que l'on veut charger et les conditions initiales. On lance ensuite le modèle avec le stepper.



# Lancement d'une simulation

```
int main()
{
    srand (time(NULL));

    /*--------------------------------------
     ------------ FLIGHT ZONE -------------
     --------------------------------------*/

    //flat_zone my_zone;
    flat_thermal_soaring_zone my_zone("DATA/config1.txt");

    /*--------------------------------------
     --------------- AIRCRAFT --------------
     --------------------------------------*/

    // mass, wind spam, aspect ratio
    double m = 1.35;
    double ws = 2.;
    double ar=16.;

    // Type of aircraft
    BeelerGlider my_glider(m,ws,ar);

    /*--------------------------------------
     ---------------- PILOT ----------------
     --------------------------------------*/
    //passive_pilot my_pilot;
    pilot_genQlearn my_pilot;

    /*--------------------------------------
     --------------- STEPPER ---------------
     --------------------------------------*/
    euler_integrator my_stepper;

    /*--------------------------------------
     ------------ SIMULATION ---------------
     --------------------------------------*/
    // Initialization of the main class
    simulation mysim;

    // We attach those class the main class simulation
    mysim.fz = &my_zone;
    mysim.ac = &my_glider;
    mysim.st = &my_stepper;
    mysim.pl = &my_pilot;
```

# Lancement d'une simulation

```
// Initialization of the main parameter
// Here we choose the time limit for our simulation (second)
double dt = 0.01; // 1 ms as dt
double time_current = 0;

// Initilation of the position // TODO with a config file
double x0 = -500.;// m
double y0 = 0.;// m
double z0 = 1000.;// m
double V0 = 20.; // m/s
double gamma0 = 0.; // rad
double khi0 = 0.; // rad
double time_limit = 1000;

std::vector<double> state_init = {x0,y0,z0,V0,gamma0,khi0};
mysim.ac->set_state(state_init);

// for loop on the stepper

while(time_current < time_limit)
{

    printf("time stamp : %f\n", time_current);
    // We increment the system -> evolution of the aircraft
    mysim.step(dt);
    time_current += dt;

}
```