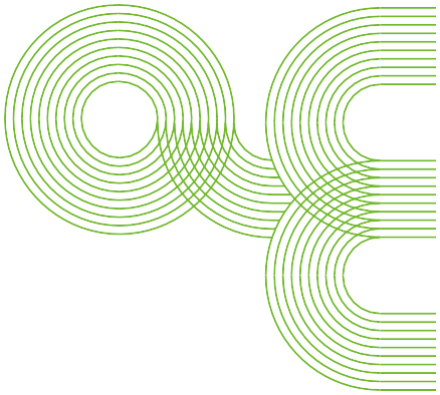
Universidade Vigo

Manual de uso de funciones Hipersónico

Definición y uso de objetos de la clase ShapeHipersonic - Matlab

Escola de Enxeñaría Aeronáutica e do Espazo



Autor/a: Christian Nicolás La Banca Lotter

1. Definición de objetos de la clase

El constructor de la clase es un constructor genérico sin argumentos, por lo que al llamar a ***shapeHipersonic()*** y asignarlo a una variable del Workspace, se crea un objeto vacío:

```
>> nuevoCuerpo = ShapeHipersonic()

nuevoCuerpo =

  ShapeHipersonic with properties:

          X_data: []
          Y_data: []
     Z_data_upper: []
     Z_data_lower: []
     dS_vec_upper: []
     dS_vec_lower: []
    Cp_data_upper: []
    Cp_data_lower: []
```

Ilustración 1.1: Uso del constructor genérico

Si queremos conocer qué funciones (métodos) tiene asociada la clase de nuestro nuevo cuerpo, podemos usar el comando methods():

```
>> methods(nuevoCuerpo)

Methods for class ShapeHipersonic:

ShapeHipersonic         setGeometry             solveForces
drawCP                  setPremadeShape         solveForcesAndMoments
drawGeometry            solveCp
```

Ilustración 1.2: Funciones disponibles para el usuario

2. Descripción del flujo de trabajo

El workflow se basa en definir la geometría y posteriormente obtener la distribución de CP y las fuerzas resultantes en base a esa geometría.

La geometría estará definida por dos superficies, una superficie inferior y otra superior, y está basada en la definición del plano XY como el plano base, teniendo la posibilidad

de definir puntos para Z_upper y Z_lower asociados a cada uno de los puntos definidos en el plano XY.

Ambas superficies deben tener un formato meshgrid (ver documentación de Matlab para meshgrid). Por lo que para su generación se pueden utilizar la función meshgrid de Matlab y las funciones customizadas como triangle_meshgrid o f_meshgrid.

a. Preset de geometría

La clase tiene un **preset** de geometría preconstruido, que es un con su eje de revolución situado a lo largo del eje X, con su punta en (0,0,0). Para acceder a este preset, se utilizará la función ***setPremadeShape(****settings****)***.

```
cone  = ShapeHipersonic();
cone = cone.setPremadeShape("shape",'Cone','coneLength',5,'coneAngle',deg2rad(11),'nDivisions_X',50,'nDivisions_angle',250);
```

Ilustración 2.1: Definición del cono preestablecido

Se nos muestra el siguiente mensaje que indica que la geometría se ha definido correctamente:

```
Preset geometry selected
Normals set
```

b. Visualización de geometría

Podemos utilizar la función ***drawGeometry()*** para visualizar la geometría que hemos definido, el código de colores se basa en le valor de la variable Z.

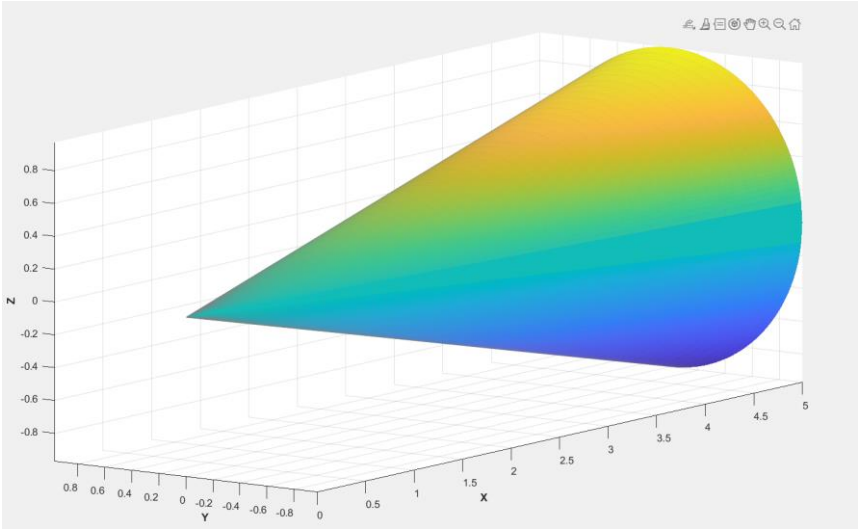


Ilustración 2.2: Visualización de la geometría

c. Obtención de CP

Podemos utilizar la función ***solveCP(***alpha,beta***)*** para calcular la distribución de CP en la geometría para un ángulo de ataque **alpha** y un ángulo de guiñada **beta**, ambos en radianes.

Calculamos CP para el cono a un Alpha de 15º y se nos enseña en consola el siguiente mensaje que indica que se ha calculado CP correctamente:

```
CP calculated using Newton hipersonic model for aoa: 15 ,beta: 0 in 0.027461 seconds
```

d. Obtención de la fuerzas y momentos

Tenemos a nuestra disposición dos funciones, distintas.

[CF]= *solveForces(*alpha,beta***)***

calcula los coeficientes de fuerzas, **en ejes cuerpo,** para un ángulo de ataque alpha y un ángulo de guiñada beta, ambos en radianes, mientras que

[CF, CM] = *solveForcesAndMoments(*alpha,beta,punto***)***

calcula las fuerzas y los momentos, **en ejes cuerpo,** para un ángulo de ataque alpha y un ángulo de guiñada beta, ambos en radianes.

3. Definición de una geometría customizada

Existen 4 propiedades asociadas a la geometría: X_data, Y_data, Z_data_upper, Z_data_lower que se pueden visualizar pero no se pueden definir. Para definirlos, hay que utilizar la función ***setGeometry(***X,Y,Z_up,Z_low***)***

Esta función acepta 4 matrices en formato **meshgrid**, por lo que pueden definir cualquier geometría.

Por ejemplo:

a. Definición de una cuña

Para definir una cuña definiremos un rectángulo en el plano XY que tendrá asociado una Z_up y una Z_down para cada punto x,y. Para definir estas matrices, utilizaremos la función meshgrid de Matlab:

```
xvector = linspace(0,4,100);
yvector = linspace(0,2,100);
[X,Y] = meshgrid(xvector,yvector);
```

Ilustración 3.1: Definición del plano XY mallado en Matlab

Conocido el plano XY, definiremos una cuña con dos superficies, una superior y otra inferior

```
Z_up = 0.2*X;
Z_down = -X;
```

Se define la superior con una pendiente de 0.2 y la inferior de -1.

Ahora, debemos asociar estas matrices a nuestro nuevo objeto, para ello, generamos el objeto vacío y asociamos la geometría con ***setGeometry***.

```
>> cuna = ShapeHipersonic;
>> cuna = cuna.setGeometry(X,Y,Z_up,Z_down)
Custom geometry defined
Normals set

cuna =

  ShapeHipersonic with properties:

          X_data: [100x100 double]
          Y_data: [100x100 double]
    Z_data_upper: [100x100 double]
    Z_data_lower: [100x100 double]
     dS_vec_upper: {100x100 cell}
     dS_vec_lower: {100x100 cell}
   Cp_data_upper: []
   Cp_data_lower: []
```

Ilustración 3.2: Asociación de la geometría al objeto

Ahora, calcular y dibujar CP alrededor de la geometría es ahora simplemente utilizar las funciones de la clase:

```
>> cuna = cuna.solveCp(0,0);
CP calculated using Newton hipersonic model for aoa: 0 ,beta: 0 in 0.0060384 seconds
>> cuna.drawCP
```

Resultado:

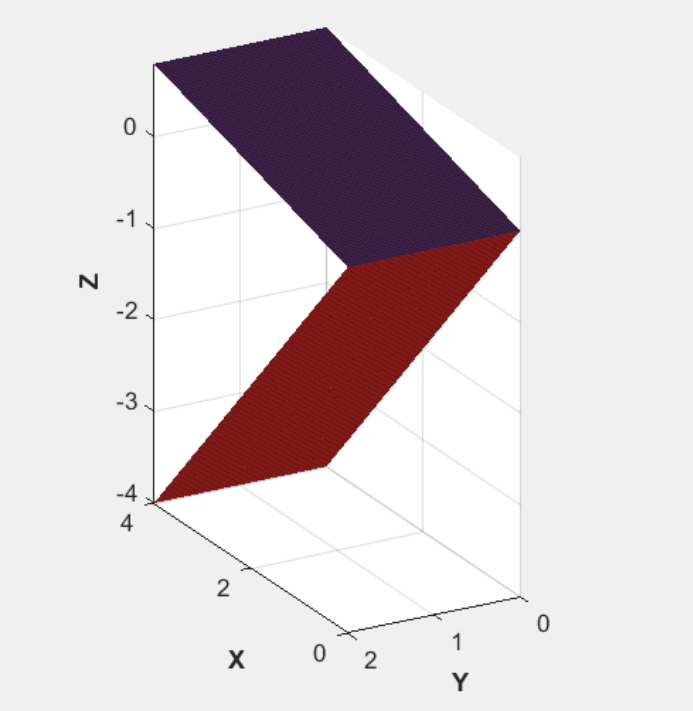


Ilustración 3.3: Distribución de CP para alpha = beta = 0 en la cuña que hemos definido

Para obtener las fuerzas utilizaremos la función ***solveForces***.

```
>> CF = cuna.solveForces(0,0)
CP calculated using Newton hipersonic model for aoa: 0 ,beta: 0 in 0.0089851 seconds

CF =

    8.2880
         0
    7.5346
```