Práctica #1: Manipulación de Campos Escalares y Vectoriales

Carlos A. Vásquez Castañeda 1155057 Grupo 395

Febrero 22, 2020

Introducción

Como es costumbre en las ramas de ingeniería, las matemáticas son de gran utilidad para lograr la abstracción de distitnos fenómenos físicos y así poder llegar a modelos que sean fáciles de comprender. El modelado de sistemas físicos como el de los fluidos no es excepción. Si consideramos cada partícula que conforma el fluido estudidado podemos asignarle una posición, componentes de velocidad, de aceleración y otras propiedades físicas de interés. Independientemente de aquello que se quiera modelar, ya es posible contar con una base a partir de esta abstracción del fluido, otorgándonos un vector para cada partícula que represente su estado físico en el espacio.

En este punto de la abstracción podemos notar que la utilización de la mecánica y cálculo vectorial es de gran ayuda. La introducción de funciones multivariables, campos escalares y campos vectoriales nos permiten tomar cualquier subcojunto del conjunto $\mathbb{R}^n$ y generar una imagen de éste con m-dimensiones, lo cual efectivamente nos otorgaría una aplicación de la forma:

$$f : D \subset \mathbb{R}^n \to \mathbb{R}^m \tag{1}$$

De esta manera formalizamos la noción de funciones multivariables, donde cada punto $X \in D$ le hace corresponer un único punto $Y \in \mathbb{R}^m$, que podemos denotar de la forma $Y = f(X)$ y lo llamaremos la *imagen* del punto X mediante la aplicación f. Esta aplicación

se puede aplicar a cualquier conjunto que sea topológicamente homomorfo a $\mathbb{R}^n$

La ecuación (1) resulta ser la forma de las aplicaciones multivariables más general, y de aquí podemos partir para definir los campos escalares y campos vectoriales.

Un **campo escalar** es una función real de varias variables en la que a cada punto de su dominio se le asigna el valor que toma una determinada magnitud escalar sobre dicho punto. Expresado de manera análoga a la ecuación (1):

$$f : D \subset \mathbb{R}^n \to \mathbb{R} \tag{2}$$

La imagen de X es reubicada en la recta de número reales. Por este motivo es que se les llama campos escalares. Podemos iniciar con n-dimensiones pero siempre debemos de obtener un valor concreto dentro de la recta real para que sea considerado un campo escalar.

Por otro lado, un **campo vectorial** es una función vectorial de varias variables en la que a cada punto de su dominio se le asigna el vector correspondiente a una determinada magnitud vectorial que actúa sobre dicho punto. Formalmente se expresa de la misma manera que la ecuación (1).

Como se decía a inicios de esta sección, estas herramientas resultan útiles para representar partículas con distintas propiedades físicas. Ejemplos de campos escalares pueden ser: densidad, temperatura, altura, etc. Ejemplos de campos vectoriales podrían ser: campos de fuerzas (eléctricos, gravitacionales), campos de velocidades (velocidad de un fluido, movimiento del viento, líneas de corriente). En las siguientes secciones veremos el comportamiento de estos campos y sus gráficas resultantes.

Procedimiento de la Práctica

Campo Escalar

Como primer acercamiento al entorno de Wolfram Mathematica, se declaró una función multivariable la cual regresa un valor numérico, y es representada mediante una gráfica de contornos. La declaración de variables es muy sencilla. Por otro lado, la función para graficar contornos necesita ciertos parámetros. Se necesita especificar el rango de la función de las

variables independientes, el tamaño de la imagen, el valor de los contornos que se desea visualizar, los colores y el nombre de los ejes. Cada contorno representa una línea donde el valor de de la función es el mismo. Por lo tanto, si lo visualizamos como una altura, estas líneas representan los puntos en donde la altura es la misma.

```
In[1]:= p1[x_, y_] := x^2 + y^3
In[2]:= graph1 = ContourPlot[p1[x, y], {x, -3, 3}, {y, -3, 3}, ImageSize → 350, Contours → {-5, -10, 0, 10, 15},
          ColorFunction → Function[u, Blend[{Blue, Cyan, Green, Yellow, Red}, u]], PlotLegends → Automatic]
```

Out[2]=

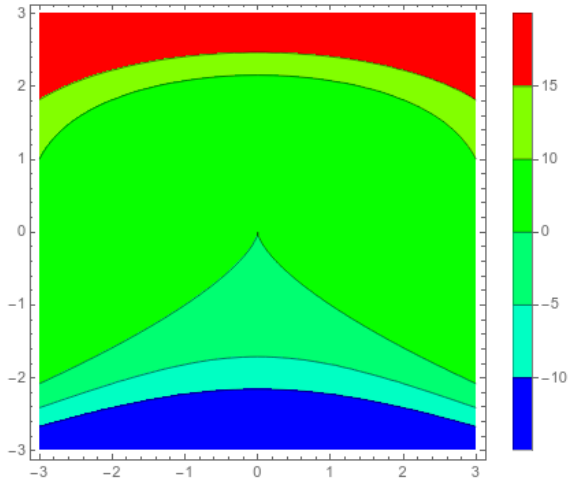


Figure 1: Código y gráfica de la función.

Campo Vectorial

Es posible declarar una función vectorial y manipularla al igual que con la función escalar definida anteriormente. La declaración de la función es similar, sin embargo se utilizan corchetes para especificar que se trata de un vector, lo cual se acostumbra en la programación orientada a objetos. De manera similar, esta gráfica se genera mediante la función VectorPLot, la cual toma como parámetros al campo vectorial, los intervalos de las variables independientes y especificaciones de los vectores que se dibujan en pantalla.

```
In[3]:= v1[x_, y_] := {Sin[y], x^2 - y}
In[4]:= graph2 = VectorPlot[v1[x, y], {x, -3, 3}, {y, -3, 3}, VectorScale → 0.3, VectorStyle → {Arrowheads[0.02], Black}]
```

Out[4]=

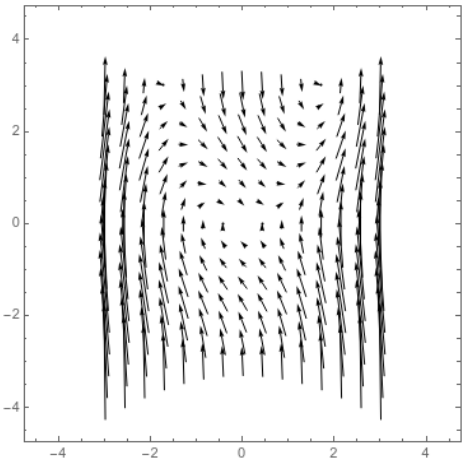


Figure 2: Código y gráfica del campo vectorial.

También es posible obtener la magnitud de estas funciones mediante la función de norma euclidiana ya integrada en Mathematica.

```
In[5]:= magv1 = Norm[v1[x, y]]
Out[5]= √(Abs[x^2 - y]^2 + Abs[Sin[y]]^2)

In[6]:= graph3 = ContourPlot[magv1, {x, -3, 3}, {y, -5, 5}, ImageSize → 350, Contours → 15, ColorFunction → Function[u, Blend[{Blue, Cyan, Green, Yellow, Red}, u]],
  PlotLegends → Automatic, PlotLabel → "Magnitud de la Velocidad |v1|"]
```

Out[6]=

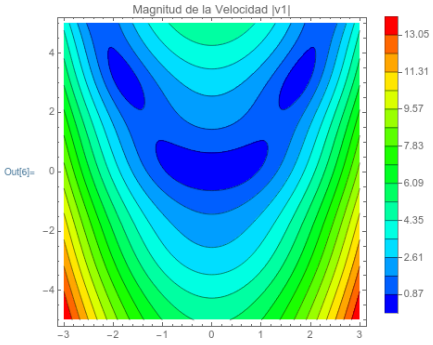


Figure 3: Cálculo y representación gráfica de la norma de la función vectorial.

Gradiente

Una vez explorada la declaración de estos distintos tipos de funciones y arreglos, algunas operaciones que es posible hacer con éstos son las de gradiente, divergencia y rotacional. La operación de gradiente se aplica a los campos escalares para hallar la dirección en la cual se

halla el cambio más rápido de la función. Es análogo a las derivadas de cálculo de una sola variable.

```
In[9]:= gradp1 = Grad[p1[x, y], {x, y}]
Out[9]= {2 x, 3 y^2}

In[10]:= graph4 = VectorPlot[Evaluate[gradp1], {x, -3, 3}, {y, -3, 3}, VectorStyle → {Arrowheads[0.02], Black}]
```

Out[10]=

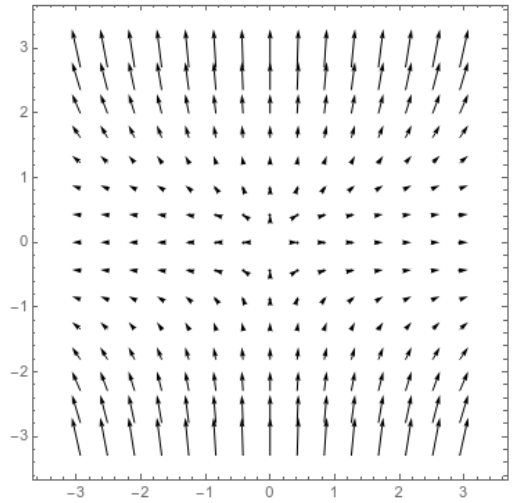


Figure 4: Gradiente de la función escalar definida anteriormente.

Podemos superponer las gráficas para ver cómo embonan entre ellas

```
In[11]:= Show[graph1, graph4]
```

Out[11]=

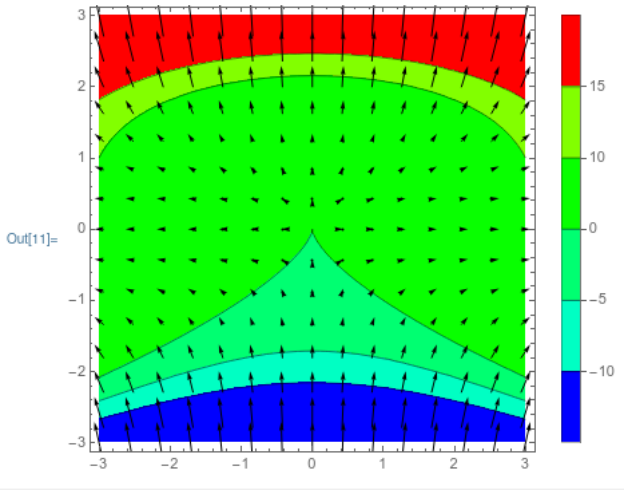


Figure 5: Superposición de las gráficas de campo escalar y su gradiente.

Divergencia

Una cantidad de interés para la mecánica de fluidos y de sustentación es la de divergencia. En un campo vectorial es posible medir la diferencia de flujo entrante y saliente de una región dada. Esta operación es la que llamamos divergencia. La sintaxis es bastante similar a la de las otras funciones.

```
In[13]:= diverv1 = Div[v1[x, y], {x, y}]
Out[13]= -1

In[14]:= diverGradp1 = Div[gradp1, {x, y}]
Out[14]= 2 + 6 y

In[15]:=

In[16]:= ContourPlot[diverGradp1, {x, -3, 3}, {y, -3, 3}, ImageSize → 350,
          Contours → 15,
          ColorFunction → Function[u, Blend[{Blue, Cyan, Green, Yellow, Red}, u]],
          PlotLegends → Automatic]
```

Out[16]=

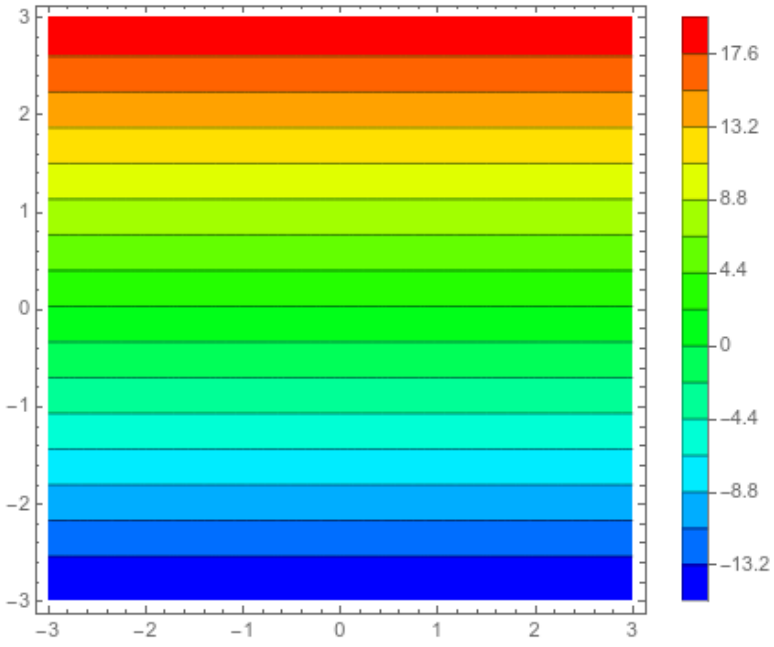


Figure 6: Divergencia del campo vectorial y el gradiente de la función escalar.

La interpretación es, si la divergencia es menor a 0, se dice que el fluido se comprime, o entra a la región. Y viceversa si la divergencia es mayor a 0.

Campo Vectorial Numérico

Los campos vectoriales se pueden generar de muchas maneras, la que hemos explorado hasta el momento es mediante la definición de funciones arbitrarias. Sin embargo, en la realidad estos campos vectoriales se modelan mediante la designación de coordenadas a distintos puntos mediante programas de simulación. En éstos podemos modelar objetos y fluidos, los cuales no proporcionarán un campo escalar único con propiedades físicas definidas. Con estos datos podemos calcular distintas propiedades físicas como el campo de velocidades, de mucha utilidad en dinámica de fluidos.

Mediante los archivos proporcionados en clase, hemos importado las coordenadas de los puntos, sus velocidades y presiones. La manipulación de estos datos nos ayudará a visualizar el estado físico del sistema.

```
In[19]:= nodos =
          Import[
           "~/Documents/Academic/Uni/6to\ Semestre/Mecanica\ de\
             Sustentacion/LAB/PR1/nodos.dat"]
```

```
Out[19]= {{0.049251, -0.0058826}, {0.049222, -0.0059485}, {0.048219, -0.0054325},
          {0.048191, -0.0054981}, {0.049293, -0.005783}, {0.048261, -0.0053335},
          ··· 31 946 ···, {0.34281, 0.1235}, {0.34521, 0.1235}, {-0.14792, -0.1235},
          {-0.14792, 0.1235}, {0.3476, -0.1235}, {0.3476, 0.1235}}

          large output | show less | show more | show all | set size limit...
```

```
In[20]:= campos =
          Import[
           "~/Documents/Academic/Uni/6to\ Semestre/Mecanica\ de\
             Sustentacion/LAB/PR1/fieldQ_xv_yv_dp_tp.dat"]
```

```
Out[20]= {{0.22288, -0.23176, 0.12349, 101 290.}, {0., 0., 0.96389, 101 290.},
          {-0.26445, 0.048373, 0.094226, 101 290.}, {0., 0., 0.054211, 101 290.},
          ··· 31 950 ···, {0., 0., 175.95, 101 340.}, {0., 0., 175.95, 101 330.},
          {0., 0., 78.883, 101 330.}, {0., 0., 66.575, 101 320.}}

          large output | show less | show more | show all | set size limit...
```

Figure 7: Importación de los nodos y datos de los campos vectoriales del perfil alar.

Al renderizar estos puntos importados obtendremos la ubicación de los puntos individuales.

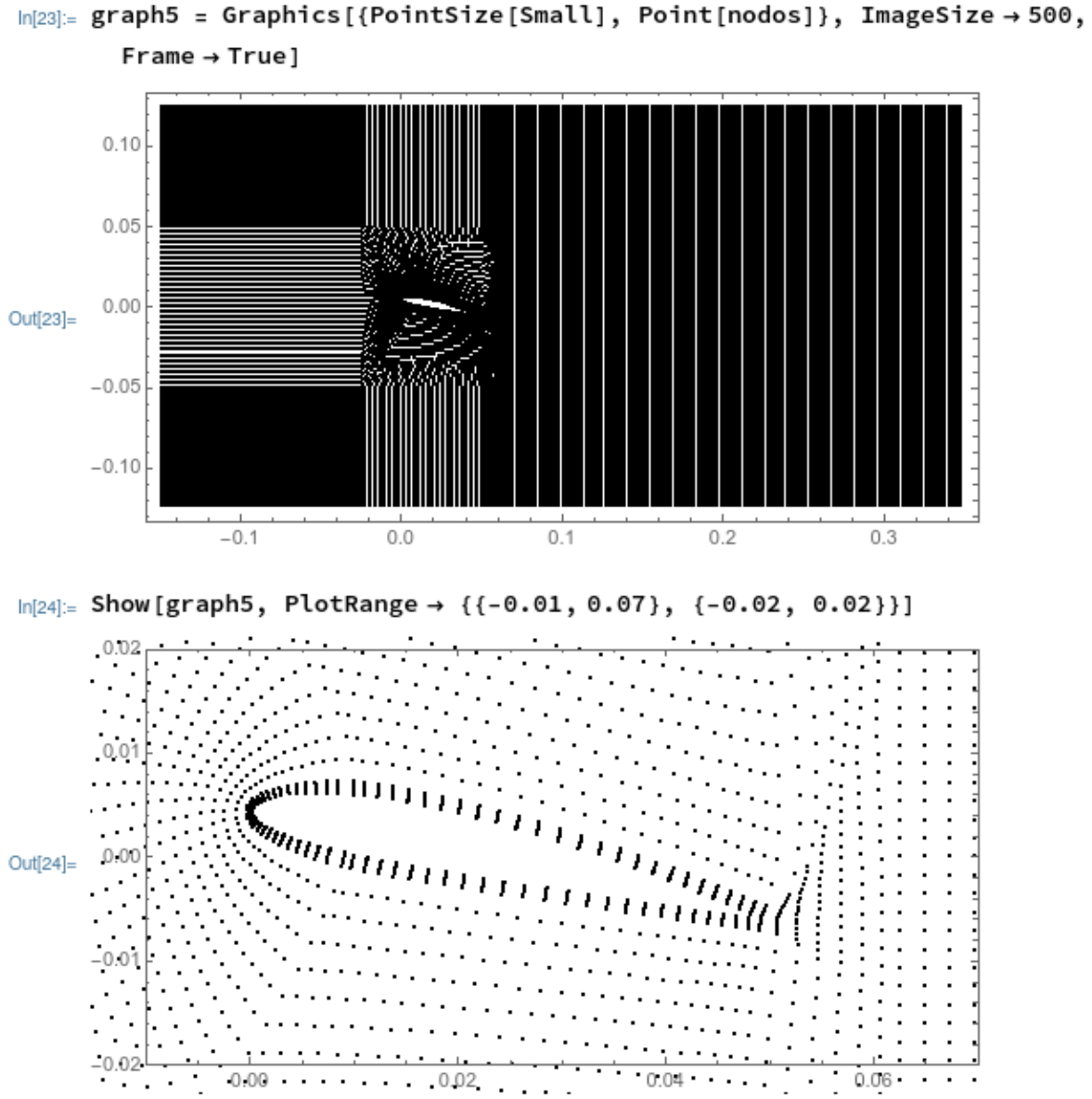


Figure 8: Visualización de los puntos.

Podemos ver cómo los puntos delinean el contorno del perfil alar, pero esto sólo son los puntos específicos que se utilizaron para la discretización de las líneas que representaban al modelo de perfil alar. A partir de los demás datos importados, podemos mezclarlos para visualizar las distintias propiedades del sistema como lo es la velocidad en la componente direccional x.

```
In[26]:= velY = campos[[All, 2]];

In[27]:= dimP = campos[[All, 23]];
      ··· Part: Part 23 of {0.22288, -0.23176, 0.12349, 101290.} does not exist.

In[28]:= totP = campos[[All, 4]];

In[29]:= dimP = campos[[All, 3]];

In[30]:= coordX = nodos[[All, 1]];

In[31]:= coordY = nodos[[All, 2]];

In[32]:= graph6 =
          Graphics3D[{PointSize[Small], Point[Transpose[{coordX, coordY, velX}]]},
           ImageSize → 700, BoxRatios → {2, 1, 1 / 2}, Axes → True,
           AxesLabel → {"x", "y", "Vx"}]
```

Out[32]=

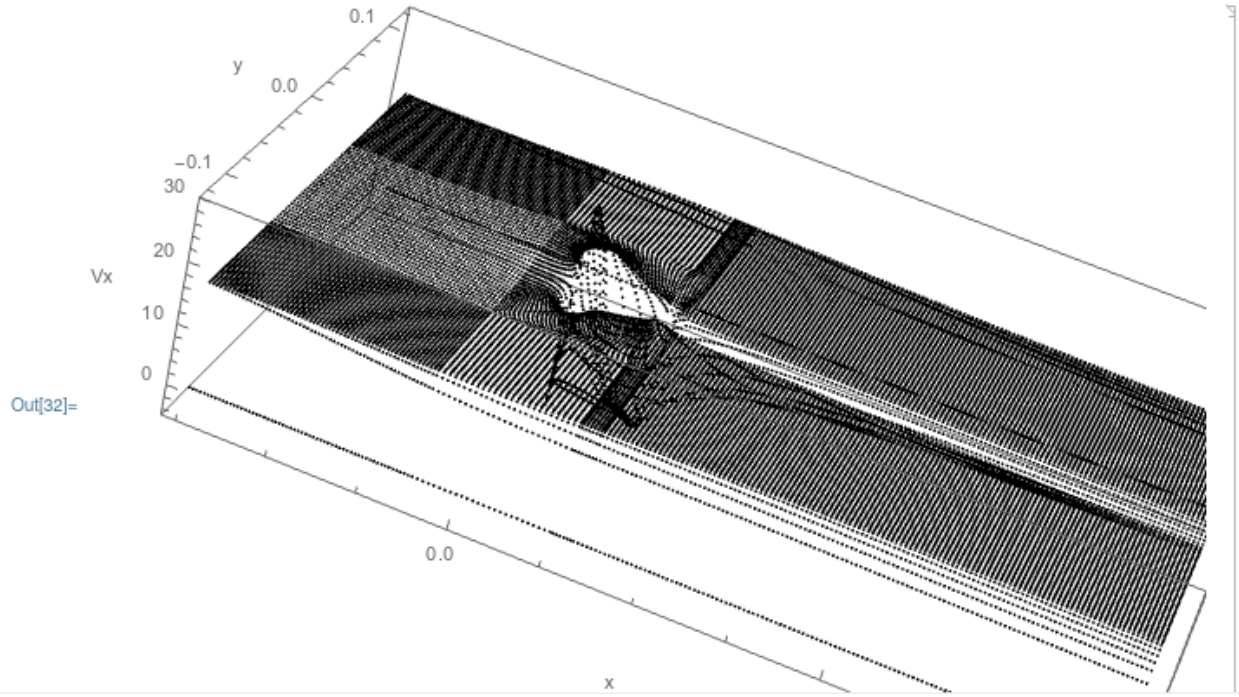


Figure 9: En las primeras líneas del código se almacenan los datos en variables para una referencia sencilla, posteriormente se grafican los puntos y la velocidad en la componente x.

A pesar de tener esta visualización, resultad difícil discernir los valores máximos y mínimos de la velocidad. Otra opción para representar estos valores es utilizando contornos como en los primeros ejemplos. Para hacer esto y que la gráfica que obtengamos sea reconocible, necesitamos obtener el *aspect ratio* de la región que graficaremos. Esto lo obtendremos mediante los valores de la coordenada X máxima, la mínima y los valores de la coordenada Y

máxima y mínima. Se muestra a continuación el bloque de código que se encarga de esto.

```
In[33]:= {entradaX, salidaX} = MinMax[coordX]
Out[33]= {-0.14792, 0.3476}

In[34]:=

In[35]:= {parediY, paredsY} = MinMax[coordY]
Out[35]= {-0.1235, 0.1235}

In[36]:= Subtract[salidaX, entradaX]
Out[36]= 0.49552

In[37]:= paredsY - parediY
Out[37]= 0.247

In[38]:= graph7 = ListContourPlot[Transpose[{coordX, coordY, velX}],
  ColorFunction → Function[u, Blend[{Blue, Cyan, Green, Yellow, Red}, u]],
  PlotLegends → Automatic,
  AspectRatio → {(paredsY - parediY) / (salidaX - entradaX)},
  PlotRange → All, PlotLabel → "Campo Escalar Vx"]
```

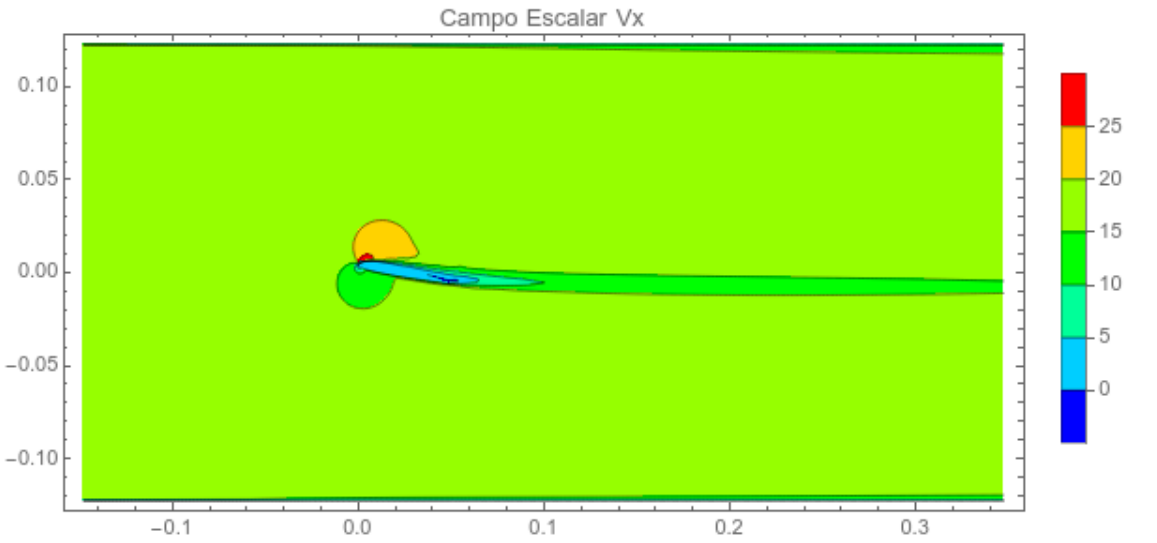


Figure 10: Obtención de las coordenadas, el *aspect ratio*, y gráfica de la función en contornos.

Aún resulta un poco complejo visualizar en qué punto se encuentra el valor máximo y mínimo de la velocidad en la componente x, para esto podemos ordenar al arreglo de número que tenemos de manor a mayor, o viceversa, y extraer el primer punto con sus valores, como

se muestra a continuación.

```
In[39]:= SortBy[Transpose[{coordX, coordY, velX}], Last][[1]]
Out[39]= {0.00037197, 0.0038626, -4.319}

In[40]:= SortBy[Transpose[{coordX, coordY, velX}], Last][[-1]]
Out[40]= {0.0017281, 0.0061127, 29.426}
```

Figure 11: Reordenamiento e impresión de los valores máximos y mínimos de la velocidad en la componente en x.

Posteriormente, para un mejor análisis de nuestro perfil, es posible generar una superficie, donde se unen los puntos, lo cual nos ayudará aún más a visualizar nuestros datos.

```
In[42]:= graph8 = ListPlot3D[Transpose[{coordX, coordY, velX}],
           ColorFunction → Function[{x, y, z},
             Blend[{Blue, Cyan, Green, Yellow, Red}, z]],
           BoxRatios → {salidaX - entradaX, paredsY - parediY, 0.2},
           PlotRange → All, PlotLabel → "Campo Escalar Vx", ImageSize → 500,
           Axes → {"x", "y", "Vx"}]
```

Out[42]=

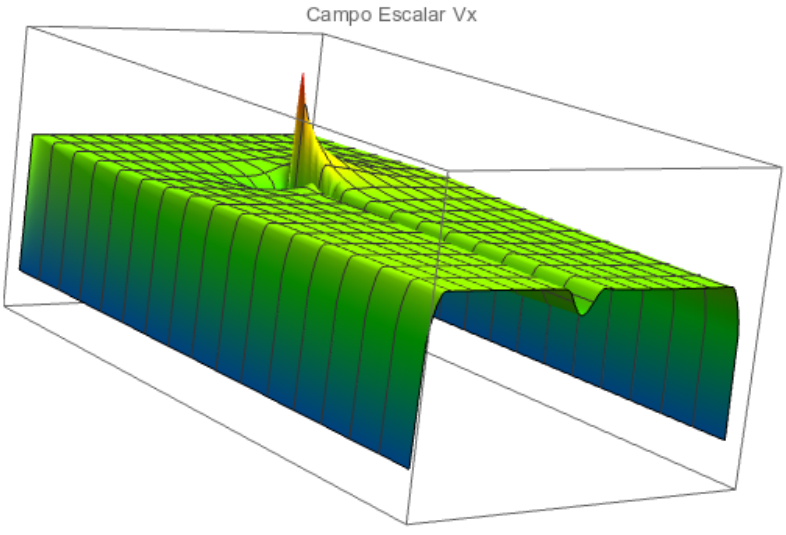


Figure 12: Superficie de los datos de perfil alar.

Esta gráfica es útil, pero si deseamos ser aún más precisos es conveniente realizar cortes de la gráfica para lograr inspeccionarlos más a fondo. Esto es posible si seleccionamos un rango

de valores, digamos valores de la coordenada Y, cuyas abscisas sean una cierta coordenada X establecida. Si deseamos analizar el corte de la entrada del flujo del fluido podemos hacerlo de la siguiente manera.

```
In[43]:= perfil0a = Select[Transpose[{coordX, coordY, velX}],
           Function[y, First[y] == entradaX]]
In[44]:= perfil0 = SortBy[perfil0a, Function[y, y[[2]]]]
```

Figure 13: Obtención de las coordenadas y su reordenamiento.

Esto resultará en la siguiente gráfica.

```
In[45]:= graph9 = ListLinePlot[perfil0[[All, {2, 3}]], Frame → True,
           FrameLabel → {"Coordenada y", "Vx"}]
```

Out[45]=

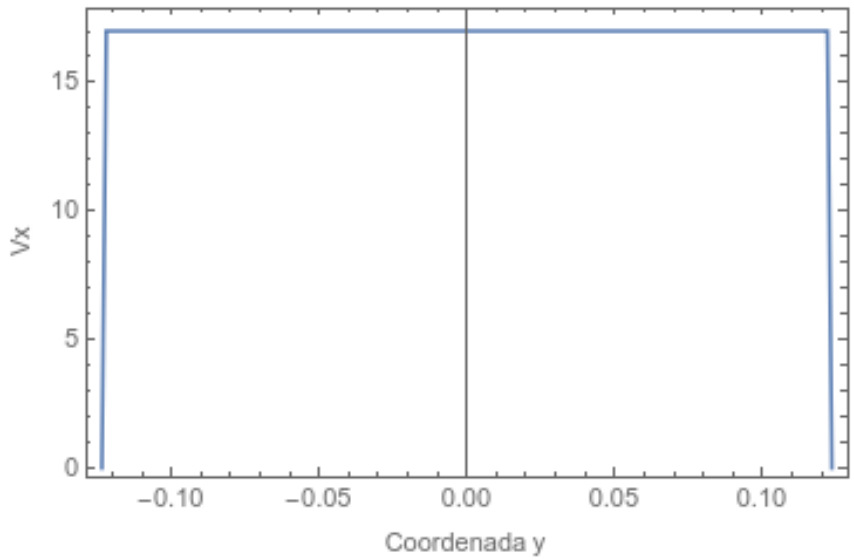


Figure 14: Corte de la superficie de la figura 12 en la entrada.

Se sigue un procedimiento similar para encontrar el corte de la superficie en la salida del sistema.

```
In[46]:= perfil1a = Select[Transpose[{coordX, coordY, velX}],
           Function[y, First[y] == salidaX]];

In[47]:= perfil1 = SortBy[perfil1a, Function[y, y[[2]]]];

In[48]:= graph10 = ListLinePlot[{perfil0[[All, {2, 3}]], perfil1[[All, {2, 3}]]},
           Frame → True, FrameLabel → {"Coordenada y", "Vx"},
           PlotLegends → {"VxEntrada", "VxSalida"}]
```

Out[48]=

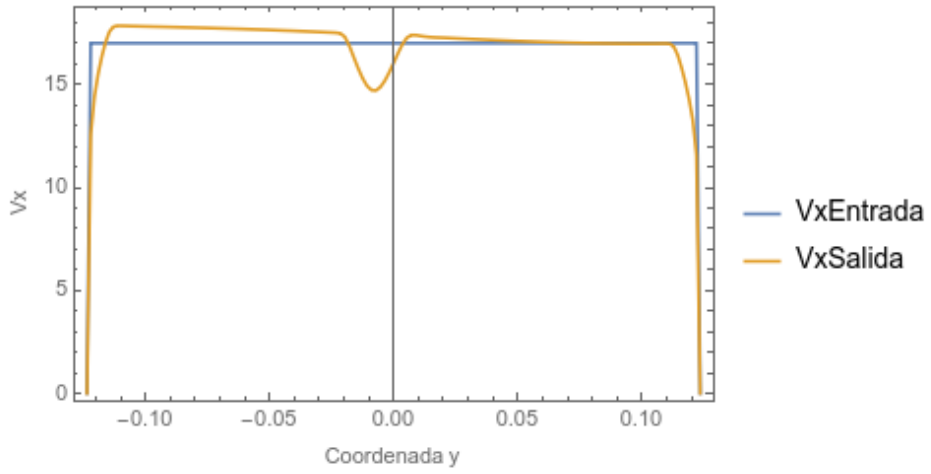
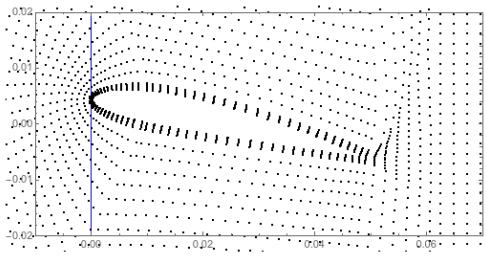


Figure 15: Superposición de las gráficas de los cortes hechos de la superficie.

Estos dos cortes son de gran utilidad, sin embargo si deseeamos trazar un perfil en un punto específico, digamos, cuando $x = 0$ obtendremos un conjunto vacío de puntos. Esto debido a que la discretización de los puntos del fluido no son exactamente 0 en su componente de posición X. Para encontrar este corte, lo que se deberá de hacer es hallar aquél conjunto de puntos cuyas coordenadas en X se aproximen al cero. Para encontrar esto y visualizarlo procedemos a escribir el siguiente código.

```
In[49]:= perfil2a = Select[Transpose[{coordX, coordY, velX}], Function[y, First[y] == 0]]
Out[49]= {}
In[50]:= Show[graph5, PlotRange -> {{-0.01, 0.07}, {-0.02, 0.02}}, GridLines → {{0}, None}, GridLinesStyle → Blue]
```



```
In[51]:= perfil2a = Select[Transpose[{coordX, coordY, velX}], Function[y, -1 / 1000 ≤ First[y] ≤ 1 / 1000]];
In[52]:= Show[graph5, Graphics[{Red, Point[perfil2a[[All, {1, 2}]]]}], PlotRange -> {{-0.01, 0.07}, {-0.02, 0.02}}, GridLines → {{0}, None}, GridLinesStyle → Blue]
```

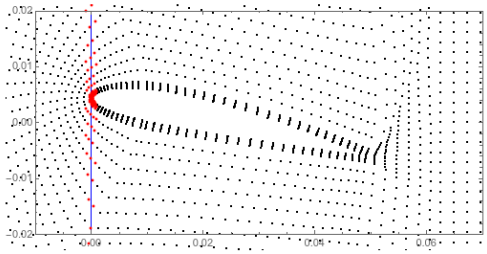


Figure 16: Obtención de los puntos que aproximandamente son $x = 0$, así como su visualización.

Nótese que se da un rango a los valores de la coordenada X, para así obtener la franja roja mostrada en la figura. Ahora, a partir de estos datos, podemos realizar la superposición de los tres cortes para su análisis posterior.

```
In[53]:= perfil2 = SortBy[perfil2a, Function[y, y[[2]]]];
In[54]:= graph11 =
          ListLinePlot[{perfil0[[All, {2, 3}]], perfil1[[All, {2, 3}]],
            perfil2[[All, {2, 3}]]}, Frame → True, FrameLabel → {"Coordenada y", "Vx"},
           PlotLegends → {"VxEntrada", "VxSalida", "x = 0"}, PlotRange → All]
```

Out[54]=

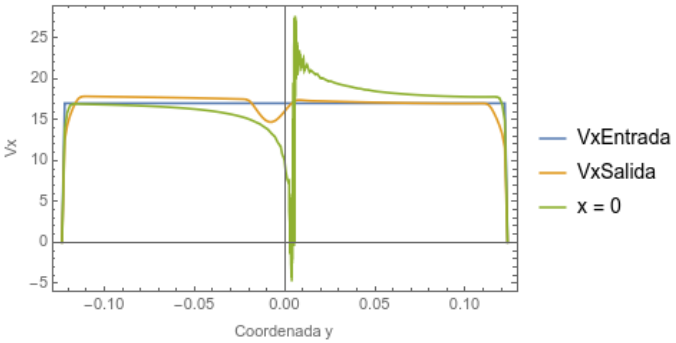


Figure 17: Gráficas superpuestas de los distintos cortes creados.

Como última forma de visualizar las partículas es mediante la construcción del campo vectorial.

```
In[56]:= dataVxy = Table[{nodos[[i]], {velX[[i]], velY[[i]]}}, {i, Length[nodos]}];

In[57]:= graph12 = ListVectorPlot[dataVxy, VectorStyle → {Arrowheads[0.01], Black},
           AspectRatio → {(paredsY - parediY) / (salidaX - entradaX)},
           VectorPoints → 50, ImageSize → 900]
```

Out[57]=

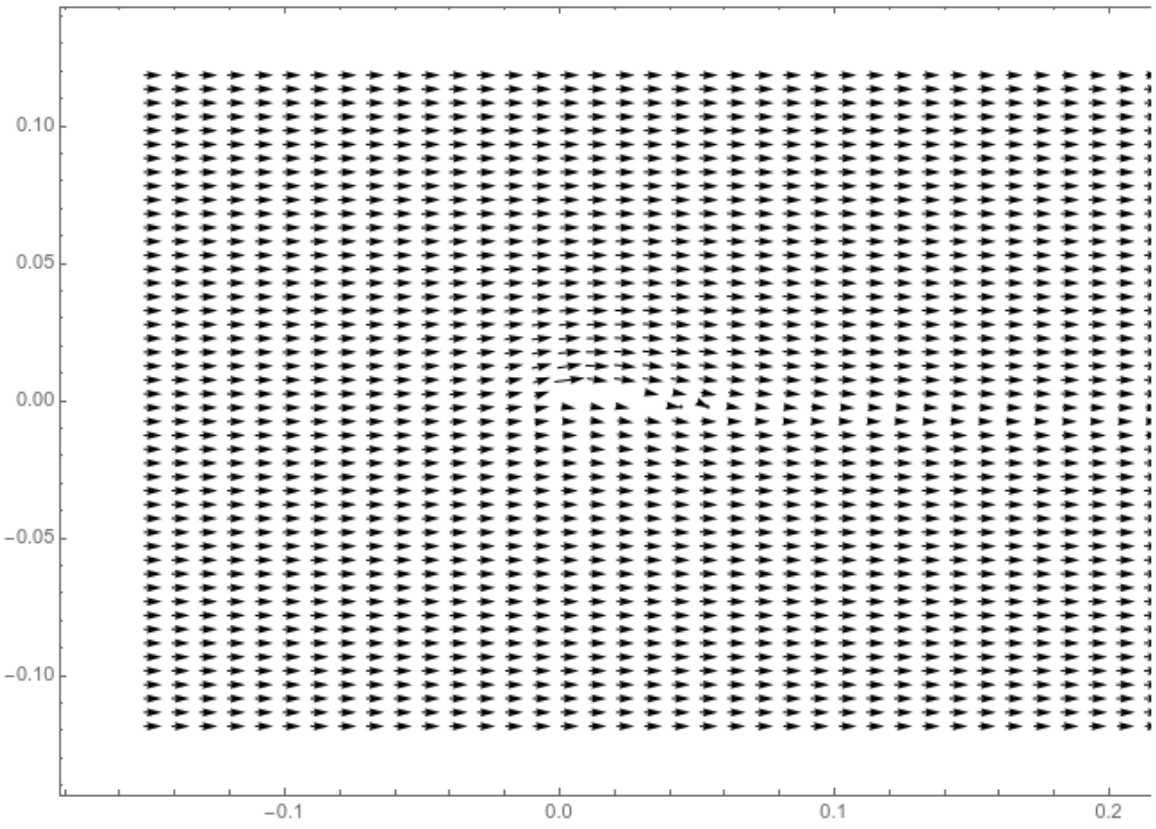


Figure 18: Campo vectorial de las partículas del fluido.

Si deseamos enfocarnos en el perfil alar es posible realizar un acercamiento.

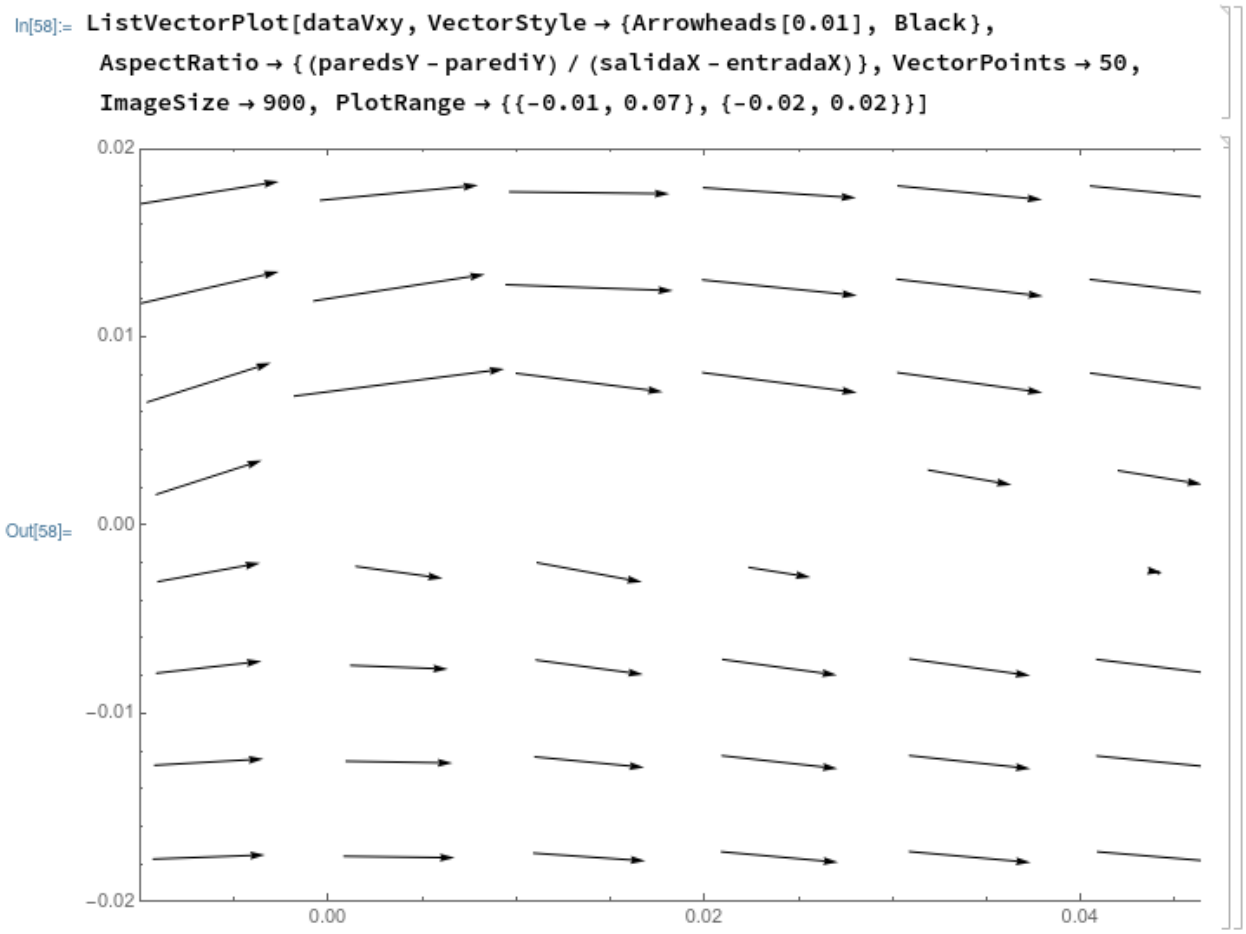


Figure 19: Acercamiento del campo vectorial.

Ejercicios

1. Discuta acerca del resultado anterior, ¿es correcto obtener valores de V_x negativos?

   Pensando en el fenómeno físico, es natural encontrar estos valores. El simple hecho de la creación de vórtices y el choque de las partículas con el perfil alar implican un cambio de dirección en las partículas del fluido, lo cual indicaría que es posible tener estos cambios de velocidad en un sentido negativo y así modificar el movimiento, llevándolo a hacer estructuras como los vórtices al final del perfil alar.

2. Adicional a V_x, obtena para el punto donde sucede la **máxima** velocidad x de los campos restantes: V_y, dp, tp.

3. Adicional a V_x, obtena para el punto donde sucede la **mínima** velocidad x de los campos restantes: V_y, dp, tp.

   Podemos responder el ejercicio **2 y 3** al mismo tiempo mediane la siguientes líneas de código:

```
In[68]:= SortBy[Transpose[{velY, dimP, totP, velX}], Last][[1]]
Out[68]= {11.136, 94.972, 101400., -4.319}

In[69]:= SortBy[Transpose[{velY, dimP, totP, velX}], Last][[-1]]
Out[69]= {10.095, 594.43, 100910., 29.426}
```

Figure 20: Valores de V_y, presión dinámica y presión total en los puntos donde V_x es máxima y mínima, respectivamente.

4. Obtenga las representaciones gráficas o trazos de los campos escalares restantes: V_y, dp, tp.

5. Calcule los valores máximos y mínimos de los campos restantes, así como las coordenadas de los puntos donde suceden.

   Los ejercicios 4 y 5 es posible contestarlos ambos al mismo tiempo. A continuación se muestran los resultados.

```
In[59]:= SortBy[Transpose[{coordX, coordY, velY}], Last][[1]]
Out[59]= {0.004745, 0.0012278, -3.1143}

In[60]:= SortBy[Transpose[{coordX, coordY, velY}], Last][[-1]]
Out[60]= {0.00029002, 0.0046634, 26.023}

In[61]:= graph71 = ListContourPlot[Transpose[{coordX, coordY, velY}],
  ColorFunction → Function[u, Blend[{Blue, Cyan, Green, Yellow, Red}, u]],
  PlotLegends → Automatic,
  AspectRatio → {(paredsY - parediY) / (salidaX - entradaX)},
  PlotRange → All, PlotLabel → "Campo Escalar Vy"]
```

Out[61]=

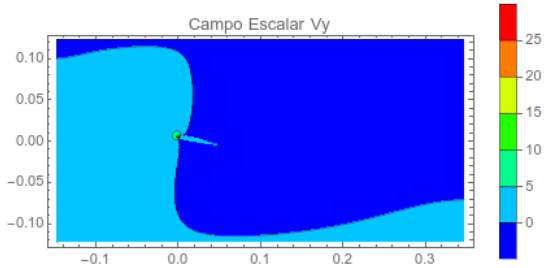


Figure 21: V_y

```
In[62]:= SortBy[Transpose[{coordX, coordY, dimP}], Last][[1]]
Out[62]= {0.033559, 0.00016819, 0.00025813}

In[63]:= SortBy[Transpose[{coordX, coordY, dimP}], Last][[-1]]
Out[63]= {0.00064538, 0.0051905, 616.98}

In[66]:= graph72 = ListContourPlot[Transpose[{coordX, coordY, dimP}],
  ColorFunction → Function[u, Blend[{Blue, Cyan, Green, Yellow, Red}, u]],
  PlotLegends → Automatic,
  AspectRatio → {(paredsY - parediY) / (salidaX - entradaX)},
  PlotRange → All, PlotLabel → "Campo Escalar dimP"]
```

Out[66]=

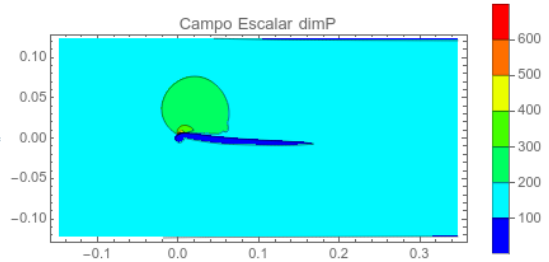


Figure 22: Presión dinámica.

```
In[64]:= SortBy[Transpose[{coordX, coordY, totP}], Last][[1]]
Out[64]= {0.00073961, 0.0050713, 100700.}

In[65]:= SortBy[Transpose[{coordX, coordY, totP}], Last][[-1]]
Out[65]= {0.0011647, 0.0030988, 101510.}

In[67]:= graph73 = ListContourPlot[Transpose[{coordX, coordY, totP}],
  ColorFunction → Function[u, Blend[{Blue, Cyan, Green, Yellow, Red}, u]],
  PlotLegends → Automatic,
  AspectRatio → {(paredsY - parediY) / (salidaX - entradaX)},
  PlotRange → All, PlotLabel → "Campo Escalar totP"]
```

Out[67]=

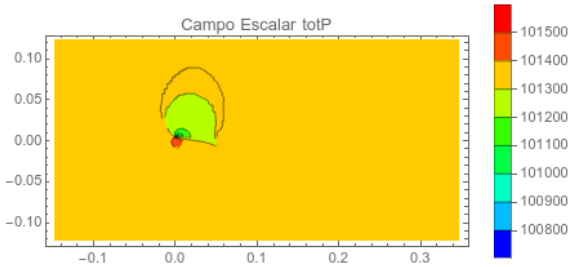


Figure 23: Presión total.

Así como se hizo en la figura 10 y 11, se realizó el mismo procedimiento pero esta vez utilizando los datos de las distintas cantidades físicas con las que se contaban.

6. Discuta sobre los resultados obtenidos en el campo escalar V_y.

   La gráfica muestra mayormente una variación en los puntos que están al inicio del perfil alar. Debido a la geometría del perfil, es razonable que las partículas tengan un aumento en su componente de velocidad vertical. Es por eso que estas partículas cercanas presentan mayor velocidad vertical en este punto, sin embargo, las demás partíciulas del fluido a su alrededor cambian sus trayectorias, normalizándolas y dándoles una dirección similar a las del fluido en general, la cual es casi puramente horizontal. En otras palabras, la ausencia de componente vertical en la velocidad es bastante grande debido a la dirección del flujo que se eligió.

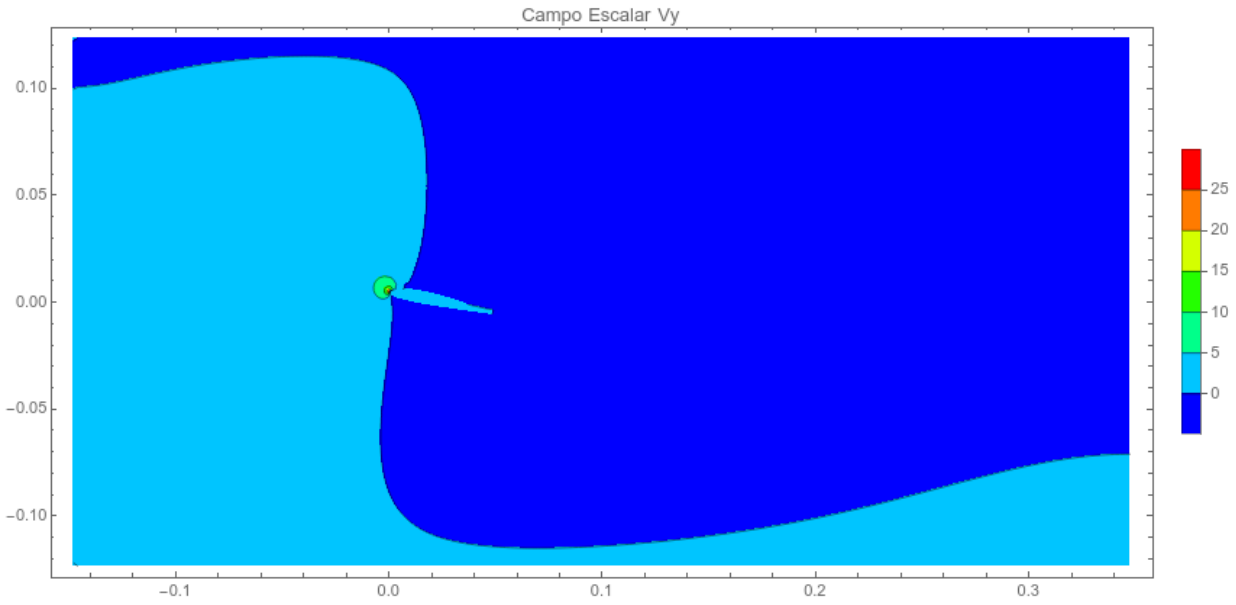


Figure 24: Detalle de la gráfica de contorno para V_y.

7. Trace los perfiles V_y para las mismas localizaciones (entrada, salida y $x = 0$).

```
In[60]:= perfil3a = Select[Transpose[{coordX, coordY, velY}],
           Function[y, First[y] == entradaX]];

In[61]:= perfil3 = SortBy[perfil3a, Function[y, y[[2]]]];

In[65]:= perfil4a = Select[Transpose[{coordX, coordY, velY}],
           Function[y, First[y] == salidaX]];

In[66]:= perfil4 = SortBy[perfil4a, Function[y, y[[2]]]];

In[67]:= perfil5a = Select[Transpose[{coordX, coordY, velY}],
           Function[y, -1 / 1000 ≤ First[y] ≤ 1 / 1000]];

In[68]:= perfil5 = SortBy[perfil5a, Function[y, y[[2]]]]

In[69]:= graph50 =
          ListLinePlot[{perfil3[[All, {2, 3}]], perfil4[[All, {2, 3}]],
            perfil5[[All, {2, 3}]]}, Frame → True, FrameLabel → {"Coordenada y", "Vy"},
           PlotLegends → {"VxEntrada", "VxSalida", "x = 0"}, PlotRange → All]
```

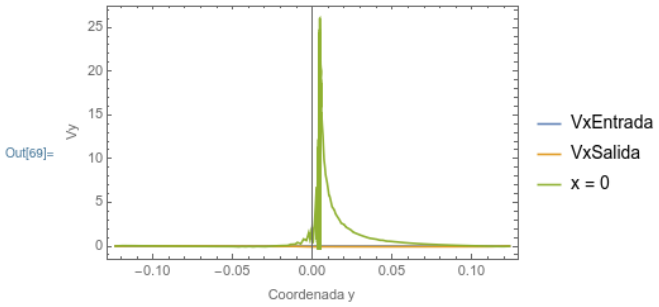


Figure 25: Cortes para V_y.

8. Trace los perfiles $|V|$ para las mismas localizaciones (entrada, salida y $x = 0$).

   Esta pregunta no supe cömo llegar a la norma de ambas magnitudes porque se encontraban en arreglos distintos (V_x y V_y).

9. Discuta sobre los perfiles de velocidad, ¿cuál es el efecto de la obstrucción del perfil alar sobre éstos?

   Aumenta dramáticamente la velocidad en ambas componentes de la velocidad.

10. Comente cuál es el comportamiento de la velocidad en las paredes y su cercanía.

    La velocidad es efectivamente cero, y eso es algo que hemos de esperar de estas simulaciones ya que hemos definido como condición de frontera esta particularidad.

Conclusión

A partir de los resultados obtenidos mediante el programa Mathematica, es sencillo ver cómo los resultados necesitan de interpretación después de haber llegado a estos campos vectoriales. Estas herramientas proporcionadas por Mathematica nos permiten realizar estas intepretaciones de manera mucho más simple. La versatilidad de las funciones y el modo en el que se modifican los arreglos y datos resulta muy amigable para llegar a la toma de decisiones adecuada.

La utilización de las distintas operaciones vectoriales son utilizadas con mucha frecuencia, en especial en la mecánica de sustentación y esto puede hacer la diferencia entre un perfil alar que sea altamente fiable a uno que sea inestable.