Coloquio Final Integrador – Tema 91

Facultad: *Facultad de Ingeniería y Ciencias Hídricas*

Materia: *Mecánica del Continuo*

Alumno: *Bartezaghi Catriel*

Fecha Entrega: *31/08/21*

Enunciado:

La teoría de flujo potencial describe el comportamiento cinemático de los fluidos basándose en el concepto matemático de función potencial, asegurando que el campo de velocidades (que es un campo vectorial) del flujo de un fluido es igual al gradiente de una función potencial que determina el movimiento de dicho fluido:

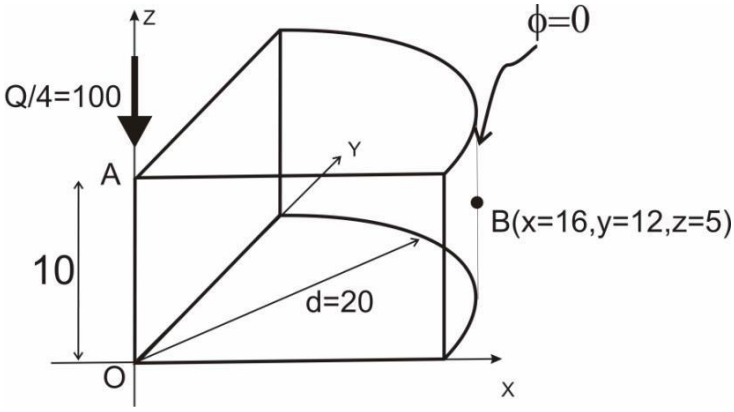
$$\mathbf{v} = -\nabla\phi$$

donde el campo de velocidades queda definido como

$$\mathbf{v} = \begin{Bmatrix} v_x \\ v_y \\ v_z \end{Bmatrix}$$

El signo menos en la ecuación de arriba es una convención de signos sobre la definición de ϕ ; podría definirse sin el signo menos y la formulación que se obtendría sería la misma. A un fluido que se comporta según esta teoría se le denomina *fluido potencial*, que da lugar a un *flujo potencial*.

1. Plantear el Principio de Trabajos Virtuales para la ecuación de Laplace (ecuación de balance de energía, donde ϕ reemplaza al campo de temperatura y la conductividad
   es unitaria). Aplicarlo al problema mostrado en la figura.



   Para imponer el caudal entrante, agregaremos la contribución del caudal concentradoen el Principio de Trabajos Virtuales mediante el término:

$$(\frac{q}{4})\delta\emptyset|_{x=0,y=0,z=10}$$

2. Plantear una aproximación polinomial al campo incógnita ϕ y obtener una solución aproximada utilizando el PTV. Graficar el campo potencial ϕ , y el vector de velocidad $\mathbf{v}$.

3. Graficar la distribución del módulo de velocidad a lo largo de la línea O-A.
4. Graficar la distribución de vectores de velocidad en el volumen.
5. Determinar el valor que debería tener el caudal entrante para que el módulo de la velocidad en el punto B sea igual a 33.
6. **Opcional:** Comparar los resultados con una solución obtenida por elementos finitos.
7. Presentar un informe detallando los pasos seguidos, programa realizado, gráficas de resultados y conclusiones.

Resolución:

1- Principio de trabajos virtuales

En el continuo, existen flujos en el cual sus partículas tienen rotación nula. Por lo tanto, para una función escalar ∅ se cumple:

$$\vec{\nabla} \times \vec{\nabla} \emptyset = 0$$

Dado que esto se cumple para la función escalar, podemos tomar cualquier vector **v** y la rotación también será nula:

$$\vec{\nabla} \times \vec{V} = 0$$

Entonces estas dos igualdades implican que:

$$\vec{V} = \vec{\nabla} \emptyset$$

Lo que nos dice esta ecuación, es que, en el caso de flujo irrotacional, el vector velocidad **v**, es equivalente al gradiente de una función escalar **∅**. A esta función escalar se le llama función potencial de velocidad. Por lo tanto, para calcular el campo velocidad, tenemos que obtener en primera instancia la función potencial.

Analizando el movimiento del fluido en la figura, se puede delimitar dos regiones:

- La primera región es por donde ingresa el flujo $\overline{\varphi}$ la cual llamaremos $\Gamma_{\varphi.}$
- La segunda región es donde se imponen valores a la función potencial ∅ la cual llamaremos $\Gamma_{\emptyset.}$

El campo ∅ satisface las ecuaciones de balance estacionario de energía:

$$\nabla \cdot (\kappa \nabla \emptyset) + q = 0$$

Donde q es la carga distribuida y k conductividad.

También satisface las condiciones de bordes dadas por el flujo que ingresa y por los valores impuestos en cada región. Entonces se tiene:

Región con valor impuesto:

$$\emptyset = \overline{\emptyset}$$

Región en que ingresa flujo:

$$-\kappa \nabla\emptyset = h$$

Ahora vamos a aplicar perturbaciones a la función ∅ que estamos buscando, en base a las condiciones de borde que se imponen en el caso de análisis, por lo que esta variación será igual a 0 en las regiones que la función potencial es nula, y arbitraria en la región que ingresa flujo, que, en este caso, se trata de un caudal que ingresa por el punto x=0, y=0, z=10 de la figura.

Luego calculamos el "trabajo virtual" hecho por los flujos externos impuestos bajo perturbación virtual dada por δ∅:

$$W_{ext} = \int_V q\delta\emptyset \, dV \; + \int_{\Gamma_\varphi} \varphi \, \delta\emptyset \, \mathrm{dS}$$

Reemplazando en el segundo término la contribución del caudal:

$$W_{ext} = \int_V q\delta\emptyset \, dV \; + (\frac{q}{4})\delta\emptyset|_{x=0,y=0,z=10}$$

Por otro lado, tenemos la expresión que se corresponde al trabajo virtual interno y que se deduce de las condiciones de balance de energía:

$$W_{int} = \int_V \nabla\delta\emptyset \,.\, (k\nabla\emptyset) \, dV$$

Por lo que la expresión para el trabajo virtual está dada por:

$$\int_V q\delta\emptyset \, dV \; + (\frac{q}{4})\delta\emptyset|_{x=0,y=0,z=10} = \int_V \nabla\delta\emptyset \,.\, (k\nabla\emptyset) \, dV$$

Como no tengo flujos distribuidos:

$$(\frac{q}{4})\delta\emptyset|_{x=0,y=0,z=10} = \int_V \nabla\delta\emptyset \,.\, (k\nabla\emptyset) \, dV$$

Obtenemos finalmente la expresión que verifica el principio de trabajo virtual, es decir, del lado izquierdo de la ecuación tenemos la expresión de trabajo virtual externo, y en el derecho el trabajo virtual interno, donde ambos son equivalentes para un sistema en equilibrio.

Ahora lo que vamos a hacer es aproximar el campo ∅, mediante una aproximación polinomial. Para ello vamos a generar polinomio de grado n, que se van a multiplicar por un polinomio que cumpla con las condiciones de borde del problema. En este caso, el polinomio que cumple con la condición de que en R=20 → ∅=0:

$$\mathrm{x}^2 + \mathrm{y}^2 - 400$$

Por lo que la función buscada queda de la forma:

$$\emptyset(x, y, \mathrm{z}) = (\mathrm{x}^2 + \mathrm{y}^2 - 400) Pn\ (x, y, \mathrm{z})$$

Donde Pn son polinomios de la forma:

$$\mathrm{P}_n(x, y, z) = \sum_{i=0}^{n} \sum_{j=0}^{n-i} \sum_{k=0}^{n-i-j} \alpha_{(\mathrm{k})(\mathrm{j})(\mathrm{i})}\ x^k y^j z^i$$

Por lo que nuestras incógnitas son los valores α_{kji}.

Esto se puede reescribir de forma matricial:

$$\emptyset(x, y, \mathrm{z}) = \mathrm{N}(\mathrm{x},\mathrm{y},\mathrm{z})\alpha$$

Ahora aplicando variaciones al sistema definimos B:

$$\mathrm{B} = \delta\emptyset(x, y, \mathrm{z}) = \mathrm{N}(\mathrm{x},\mathrm{y},\mathrm{z})\delta\alpha$$

Redefino la función gradiente:

$$\nabla\emptyset(x, y, z) = \begin{pmatrix} \frac{\partial\emptyset}{\partial x} \\ \frac{\partial\emptyset}{\partial y} \\ \frac{\partial\emptyset}{\partial z} \end{pmatrix} = \begin{pmatrix} \frac{\partial N}{\partial x} \\ \frac{\partial N}{\partial y} \\ \frac{\partial N}{\partial z} \end{pmatrix} \alpha = \mathrm{B}\alpha$$

Y con las variaciones:

$$\nabla(\delta\emptyset(x,y,z)) = \mathrm{B}\delta\alpha$$

Ahora, partiendo de la ecuación de trabajos virtuales:

$$\int_V \nabla\delta\emptyset \,.\, (k\nabla\emptyset)\, dV \;\; = (\frac{q}{4})\delta\emptyset|_{x=0,y=0,z=10}$$

Reemplazando:

$$\int_0^{10}\int_0^{20}\int_{-f(x)}^{f(x)} ((B\delta\alpha)^T \;\; kB\alpha)\, dxdzdx \;\; = (\frac{q}{4})N|_{x=0,y=0,z=10}\delta\alpha$$

Sacamos los vectores constantes fuera de la integral:

$$(\delta\alpha)^T(\int_0^{10}\int_0^{20}\int_{-f(x)}^{f(x)} (B)^T\, kB)\, dxdzdy)\alpha \;\; = (\delta\alpha)^T(\frac{q}{4}N^T|_{x=0,y=0,z=10})$$

Lo cual se puede reescribir de la siguiente forma:

$$(\delta\alpha)^T\, K\, \alpha = (\delta\alpha)^T\, f$$

Dado que la variación δø es arbitraria, esto implica que $(\delta\alpha)^T$ también lo es. Por lo que el problema se reduce al siguiente sistema matricial:

$$K\, \alpha = \; f$$

El orden del sistema, va a estar dado por el grado del polinomio usado para la aproximación, y de las dimensiones espaciales del problema, en nuestro caso 3D:

$$r = \frac{(n+1)(n+2)(n+3)}{6}$$

2- Resolución del sistema matricial

Para resolver el sistema de ecuaciones y obtener los coeficientes aproximados de la función buscada se utilizó la herramienta Matlab. El código implementado es el **Algoritmo 1**, que se puede visualizar en el Anexo.
Para mostrar los resultados obtenidos, se adjuntan distintos gráficos con grillas de colores, para distintos valores de **z**, y en este caso utilizando polinomios de grado 2.

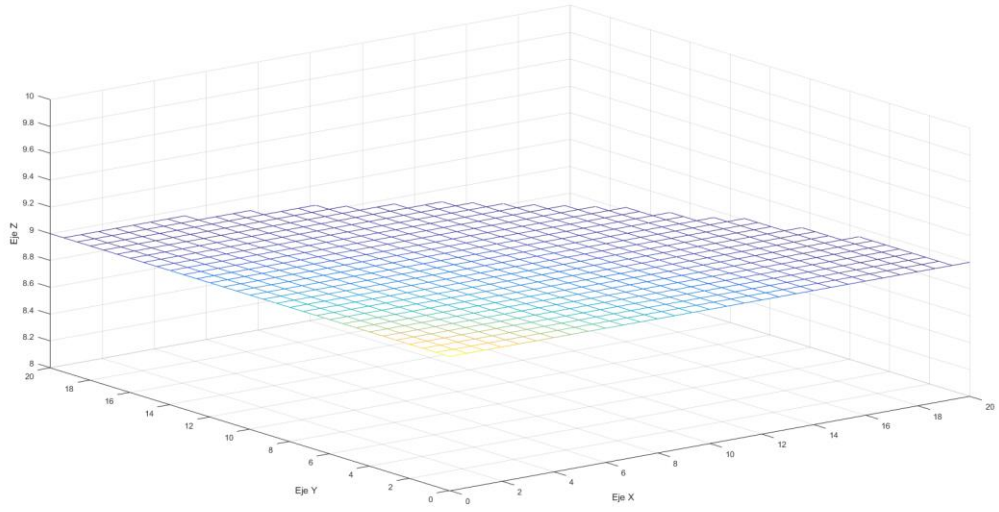


Figura 1: Gráfico de la función potencial en Z = 9

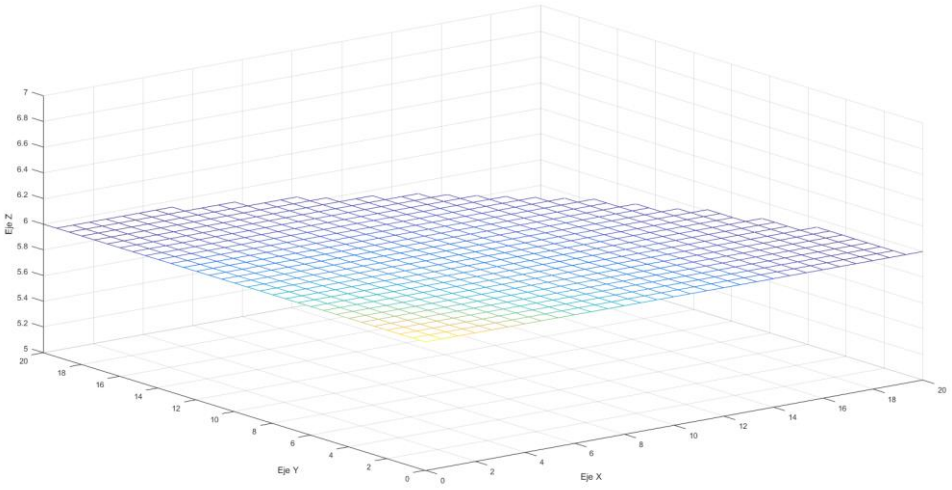


Figura 2: Gráfico de la función potencial en Z = 6

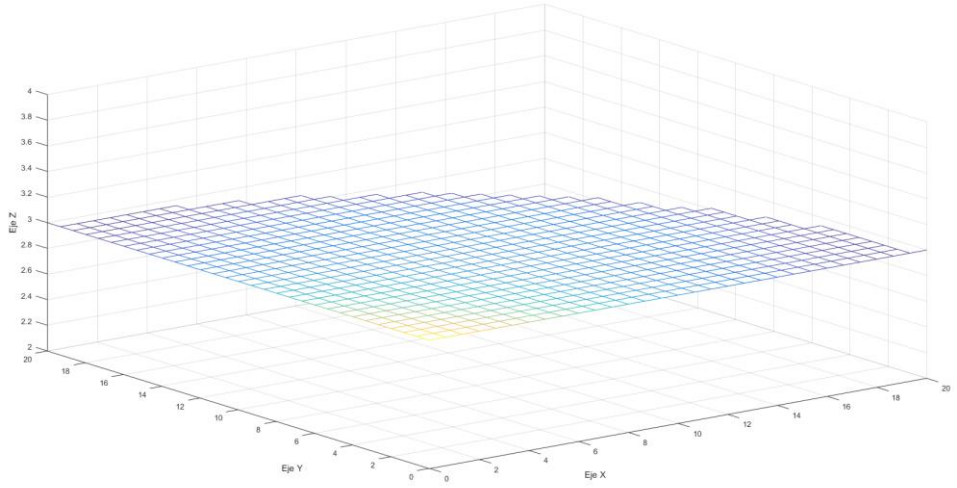


Figura 3: Gráfico de la función potencial en Z = 3

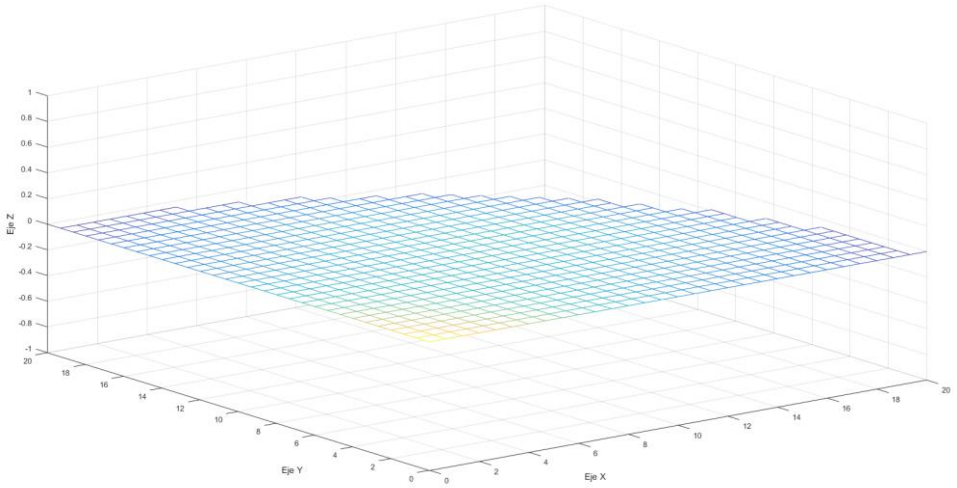


Figura 4: Gráfico de la función potencial en Z = 0

En las graficas se puede apreciar que se cumplen los valores iniciales. Se tiene una concentración de valores altos cerca del eje Z, y que va decayendo progresivamente a medida que nos acercamos a las fronteras. Además, a mayor z, se puede ver una mayor concentración de flujo, esto se debe a que el flujo ingresa por un único punto. Pero, a medida que Z se aproxima 0, el fluido se va distribuyendo en mayor medida. Esto ultimo se puede observar mejor cuando se realiza una aproximación polinómica de mayor grado:

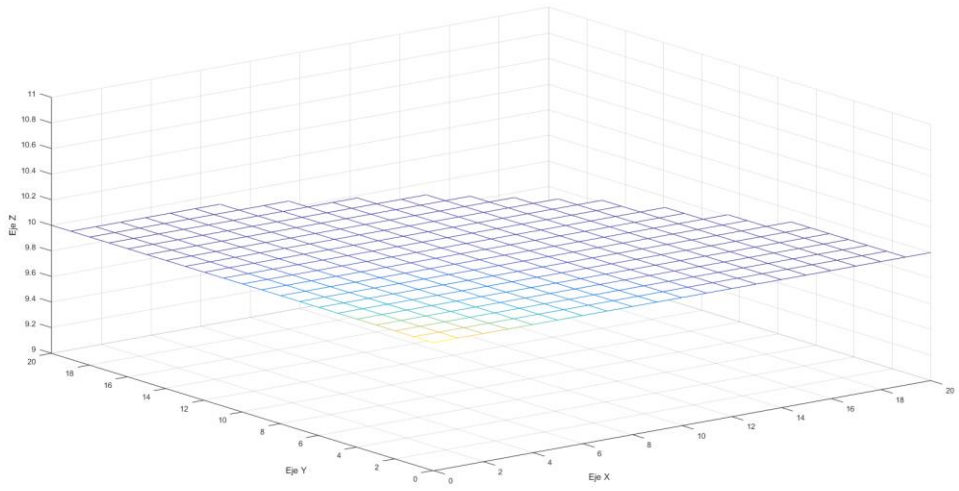


Figura 5: Gráfico de la función potencial en Z = 10 con polinomios grado 3

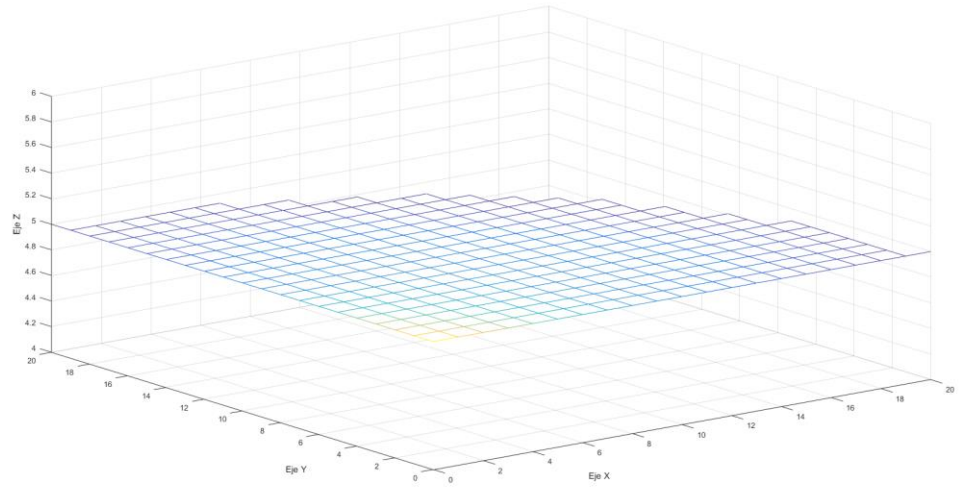


Figura 6: Gráfico de la función potencial en Z = 5 con polinomios grado 3

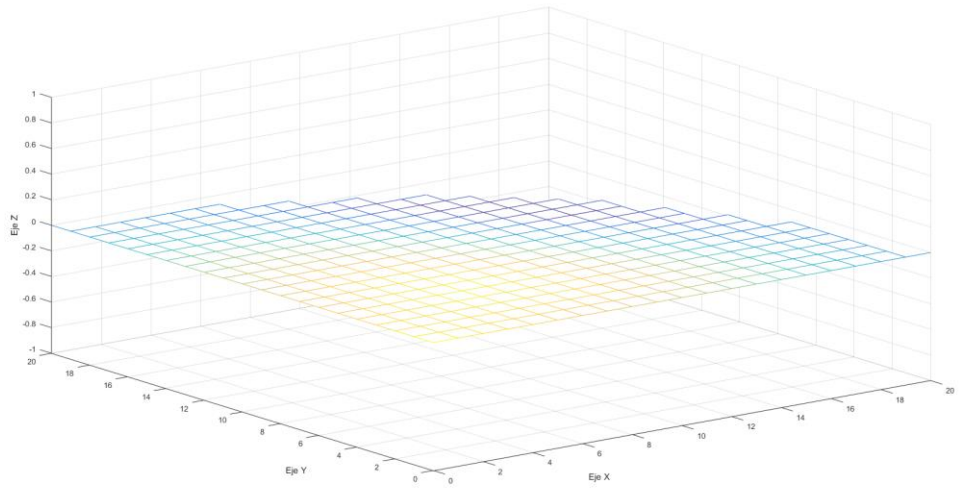


Figura 7: Gráfico de la función potencial en Z = 0 con polinomios grado 3

3- Grafica distribución del módulo de velocidad en O-A

A continuación, se muestra la grafica del módulo velocidad con respecto al eje **z**, en la recta que se conforma por el origen y el punto (x=0, y=0, z=10).

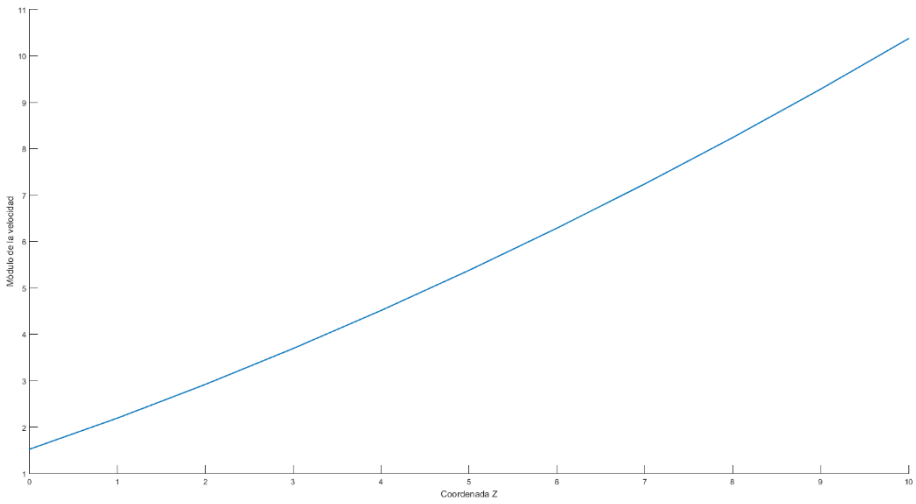


Figura 8: Módulo velocidad en O-A

En la misma se puede apreciar, como al ingresar el caudal (en z=10) tiene la velocidad máxima, y a medida que se adentra en la figura comienza a disminuir la velocidad de forma casi lineal. En el anexo se puede ver el código que realiza el cálculo (**Algoritmo 2**). Lo que se hizo, fue evaluar el sistema en diferentes puntos a lo largo del eje Z y obtener el módulo.

4- Grafica de distribución de vectores de velocidad

Para analizar la velocidad a lo largo del modelo, se codifico un algoritmo que calcula el vector velocidad para cada punto de la figura. También se dibujo el modelo para una mejor apreciación. Todo esto se puede ver en el **Algoritmo 3**.

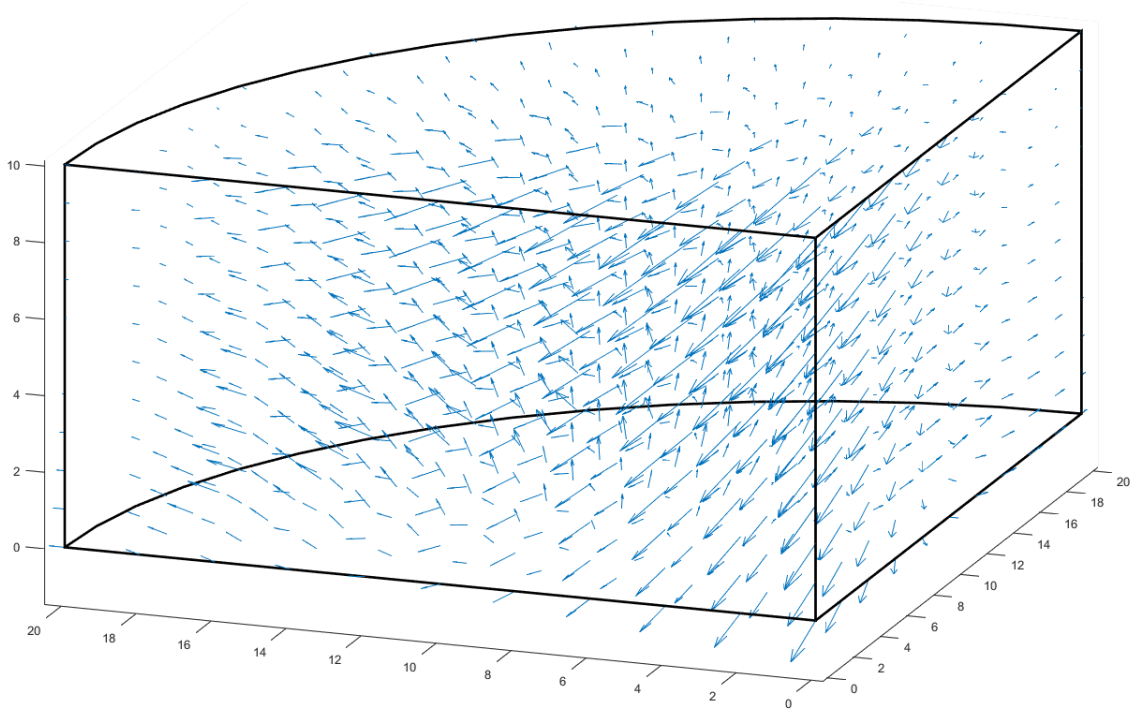


Figura 9: Vectores velocidad en el volumen

Se puede observar, al igual que en incisos anteriores, como se cumplen las condiciones iniciales. El flujo tiende a ir en dirección de la cara curva, mientras que, a lo largo de las paredes rectas, el flujo es cero por la condición adiabática impuesta.

5- Estimación valor caudal entrante

Por último, se pide calcular el valor que debería tener el caudal entrante para que el modulo de la velocidad en el punto (x=16,y=12,z=5) sea igual a **33**. Para ello de despeja la incógnita buscada del sistema, lo cual se puede ver en el **Algoritmo 4**. Con una aproximación polinómica de grado 3, se obtuvo que el caudal entrante debe ingresar con una velocidad de **1.3526e+05**

6- Conclusiones:

- El tiempo de ejecución del algoritmo depende directamente del grado

del polinomio utilizado para aproximar la función buscada. Pero también se puede ver que a medida que elevamos el grado del polinomio, y por ende aumentamos el numero de coeficientes utilizados, el sistema se va aproximando cada vez más a la solución ideal.

- El principio de trabajos virtuales nos permitió reescribir el problema de forma matricial, lo cual es una gran ventaja porque nos permite resolverlo de forma más óptima con herramientas de cálculo como Matlab.
- Se puede verificar que los resultados obtenidos se aproximan a la solución de la ecuación diferencial, ya que en las gráficas se visualiza el cumplimiento de las condiciones iniciales. También podemos decir que el comportamiento del sistema es razonable y esperado.
- Por último, concluir que las técnicas y herramientas utilizadas en este coloquio son de gran utilidad en la aplicación práctica para modelar el comportamiento de fluidos y otros flujos. Y que la suficiencia o no de este método dependerá de cada caso, de la precisión buscada y del poder computacional disponible.

Anexo: Códigos Utilizados

Algoritmo 1:

```
%% Datos del problema
grPol = 3;
v=100;


%% Datos gráfico
%Número de divisiones.
     nPartx = 10;
     nParty = 10;
     nPartz = 10;
     %Número de divisiones de la línea O-A.
     nPartOA = (nPartx+nParty+nPartz)/2-4;
     %% Matriz simbólica de producto coordenadas polinomio
     syms x y z

     nCoef=(grPol+1)*(grPol+2)*(grPol+3)/6;
     ms_N = sym(zeros(nCoef,1));
     ind = 0;
     for iz = 0:grPol
      for iy = 0:grPol-iz
        for ix = 0:grPol-iz-iy
         ind = ind+1;
         ms_N(ind) = (x^2+y^2-400)*x^ix*y^iy*z^iz;
```

```
24.         end
25.       end
26.     end
27.     %% Matriz de deformación simbólica
28.     syms ms_B
29.     ms_B =
   simplify([diff(ms_N.',x);diff(ms_N.',y);diff(ms_N.',z)]);
30.     %% Matriz de rigidez (matriz de coeficiente del
   sistema lineal)
31.     syms ms_K
32.
33.     funcion_arriba=sqrt(400-(x^2));
34.     ms_K =
   int(int(int(ms_B.'*ms_B,y,0,funcion_arriba),x,0,20),z,0,10
   );
35.     m_K = double(ms_K);
36.     %vpa(ms_K,10);
37.
38.     %% Vector de fuerza (vector de términos
   independientes)
39.     syms ms_Fr ms_F
40.
41.     ms_Fr = subs(ms_N,[x,y,z],[0,0,10]);
42.     ms_F = sym(v)*ms_Fr;
43.     m_F = double(ms_F);
44.     %m_F = vpa(ms_F);
45.     %% Resolución del sistema en forma simbólica
46.     syms ms_A
47.     ms_A = ms_K\ms_F;
48.     %% Resolución del sistema en forma numérica
49.     m_A = m_K\m_F;
50.     %% Gráfico de la función potencial
51.     figure(1)
52.
53.     %La grilla tiene que ser 3d.
54.     [m_CoordX,m_CoordY,m_CoordZ] = meshgrid(linspace(0,20
   ,2*nPartx+1),linspace(0,20,2*nParty+1),linspace(0,10,2*nPa
   rtz+1));
55.     %Para graficar la L se pone NaN en las coordenadas
   fuera del dominio.
56.     m_CoordYAux=m_CoordY;
57.     m_CoordXAux=m_CoordX;
58.     m_CoordZAux=m_CoordZ;
59.     m_CoordY(m_CoordYAux>sqrt(400-
   (m_CoordXAux.^2))) = NaN;
60.     m_CoordY(m_CoordYAux<-sqrt(400-
   (m_CoordXAux.^2))) = NaN;
61.     m_CoordY(m_CoordYAux<0) = NaN;
62.     m_CoordX(m_CoordXAux<0) = NaN;
63.
```

```
64.      nPuntos = numel(m_CoordX);
65.
66.      m_XGraf = zeros(nPuntos,nCoef);
67.      %Se evalua cada uno de los elementos de ms_X para
   todos los puntos de grilla de la gráfica.
68.      for iCoef = 1:nCoef
69.       m_XGraf(:,iCoef) =
   subs(ms_N(iCoef),{x,y,z},{m_CoordX(:),m_CoordY(:),m_CoordZ
   (:)});
70.
71.      end
72.      %Función potencial
73.      m_FunPot = reshape(m_XGraf*(m_A),size(m_CoordX));
74.      %Gráfico
75.      for i=1:size(m_CoordX(:,1,1))
76.          figure(i)
77.          mesh(m_CoordX(:,:,i),m_CoordY(:,:,i),m_CoordZ(:,:
   ,i),m_FunPot(:,:,i))
78.          xlabel('Eje X'); ylabel('Eje Y'); zlabel('Eje
   Z');
79.      end
```

Algoritmo 2:

```
1. %% Gráfico de módulo de velocidad sobre la línea OA
2. hold on
3. figure()
4. hold on
5. m_CoordXOA = linspace(0,0,nPartOA);
6. m_CoordYOA = linspace(0,0,nPartOA);
7. m_CoordZOA = linspace(0,10,nPartOA);
8. nPuntos = numel(m_CoordXOA);
9. %Se divide en dos líneas la matriz de deformación.
10.      m_BGraf1 = zeros(nPuntos,nCoef);
11.      m_BGraf2 = zeros(nPuntos,nCoef);
12.      m_BGraf3 = zeros(nPuntos,nCoef);
13.      %Se evalua cada uno de los elementos de ms_B para
   todos los puntos degrilla de la gráfica.
14.      for iCoef = 1:nCoef
15.       m_BGraf1(:,iCoef) =
   subs(ms_B(1,iCoef),{x,y,z},{m_CoordXOA,m_CoordYOA,m_CoordZ
   OA});
16.       m_BGraf2(:,iCoef) =
   subs(ms_B(2,iCoef),{x,y,z},{m_CoordXOA,m_CoordYOA,m_CoordZ
   OA});
17.       m_BGraf3(:,iCoef) =
   subs(ms_B(3,iCoef),{x,y,z},{m_CoordXOA,m_CoordYOA,m_CoordZ
   OA});
18.      end
19.      %Vector de velocidad
```

```
20.      m_VecVelX = -m_BGraf1*m_A;
21.      m_VecVelY = -m_BGraf2*m_A;
22.      m_VecVelZ = -m_BGraf3*m_A;
23.      %Gráfico del módulo de velocidad
24.      %Se grafica en función de las coordenadas x de los
   puntos.
25.
26.      mod =
   sqrt(abs(m_VecVelX).^2+abs(m_VecVelY).^2+abs(m_VecVelZ).^2
   );
27.      plot(m_CoordZOA,mod,'LineWidth',1.5);
28.
29.      xlabel('Coordenada Z'); ylabel('Módulo de la
   velocidad');
```

Algoritmo 3:

```
1. %% Gráfico de la distribución de vectores de velocidad en
   el dominio
2.
3. figure();
4. %Datos para graficar el cuarto de circunferencia
5. t = 0:pi/30:pi/2;
6. xc = 20*cos(t);
7. yc = 20*sin(t);
8. z0 = zeros(1,16);
9. z10 = 10*ones(1,16);
10.      %Gráfico del contorno de la figura
11.      plot3([0,20,20 ,0,0 ,0,0,0],...
12.            [0,0 ,0 ,0 ,20,20,0,0],...
13.            [0,0 ,10,10,10 ,0,0,10],'k','LineWidth',2)
14.      hold on
15.      plot3(xc,yc,z0,'k','LineWidth',2);
16.      plot3(xc,yc,z10,'k','LineWidth',2);
17.      axis square
18.
19.      [m_CoordX,m_CoordY,m_CoordZ] =
   meshgrid(linspace(0,20,11),linspace(0,20,11),linspace(0,10
   ,11));
20.      m_CoordYAux=m_CoordY;
21.      m_CoordXAux=m_CoordX;
22.      m_CoordZAux=m_CoordZ;
23.      m_CoordY(m_CoordYAux>sqrt(400-(m_CoordXAux.^2))) =
   NaN;
24.      m_CoordY(m_CoordYAux<-sqrt(400-(m_CoordXAux.^2))) =
   NaN;
25.      m_CoordY(m_CoordYAux<0) = NaN;
26.      m_CoordX(m_CoordXAux<0) = NaN;
27.
```

```
28.      nPuntos = numel(m_CoordX);
29.      %Se divide en dos líneas la matriz de gradientes
30.      m_BGraf1 = zeros(nPuntos,nCoef);
31.      m_BGraf2 = zeros(nPuntos,nCoef);
32.      m_BGraf3 = zeros(nPuntos,nCoef);
33.      %Se evalua cada uno de los elementos de ms_B para
   todos los puntos de grilla de la gráfica.
34.      for iCoef = 1:nCoef
35.      m_BGraf1(:,iCoef) =
   subs(ms_B(1,iCoef),{x,y,z},{m_CoordX(:),m_CoordY(:),m_Coor
   dZ(:)});
36.      m_BGraf2(:,iCoef) =
   subs(ms_B(2,iCoef),{x,y,z},{m_CoordX(:),m_CoordY(:),m_Coor
   dZ(:)});
37.      m_BGraf3(:,iCoef) =
   subs(ms_B(3,iCoef),{x,y,z},{m_CoordX(:),m_CoordY(:),m_Coor
   dZ(:)});
38.      end
39.      %Vector de flujo
40.      m_VecVelX = -m_BGraf1*m_A;
41.      m_VecVelY = -m_BGraf2*m_A;
42.      m_VecVelZ = -m_BGraf3*m_A;
43.
44.      %El quiver escala a partir del tamaño automático que
   pone automáticamente
45.      quiver3(m_CoordX(:),m_CoordY(:),m_CoordZ(:),m_VecVelX
   ,m_VecVelY,m_VecVelZ,5);
46.      hold off
47.      axis equal
```

Algoritmo 4:

```
1.  %% Determinación de la velocidad de entrada para que un
   cierto punto tenga un módulo de velocidad dado
2. modVelBusc = 33;
3. posVelBusc = [16,12,5];
4. m_BGrafposVelBusc =
   subs(ms_B,{x,y,z},{posVelBusc(1),posVelBusc(2),posVelBusc(
   3)});
5. m_VecVelposVelBusc = m_BGrafposVelBusc*m_A;
6. absVelEntr = modVelBusc/norm(m_VecVelposVelBusc)*abs(v)
```