|  |  |
| --- | --- |
| Aerotaxi ZATA:  Programa de Octave de la hélice | |
|  |  |
|  | |
|  | |

**CARACTERÍSTICAS**

|  |  |  |
| --- | --- | --- |
| NIVEL DE SEGURIDAD | NIVEL DE DIFUSIÓN | NIVEL DE REPRODUCCIÓN |
| Confidencial  Restringido  Corporativo  Público | [Persona/s, Departamento, Equipo, etc.] | No Autorizada  Controlada  Autorizada |

**ELABORACIÓN, REVISIÓN Y APROBACIÓN DEL DOCUMENTO**

|  |  |  |
| --- | --- | --- |
| ELABORADO POR: | REVISADO POR: | APROBADO POR: |
| Emilio Martín Pedrero |  |  |
| FECHA : 28/05/2020 | FECHA: | FECHA : |

**HISTÓRICO DE VERSIONES**

|  |  |  |  |
| --- | --- | --- | --- |
| EDICIÓN | FECHA | AUTOR | DESCRIPCIÓN |
| 1 | 28/05/2020 | Emilio Martín Pedrero | Versión inicial |
| 2 |  |  |  |
| 3 |  |  |  |

Tabla de contenido

[Índice de Tablas 4](#_Toc37770469)

[Índice de Figuras 5](#_Toc37770470)

[1. Corrección de las curvas de sustentación y resistencia aerodinámica en función del ángulo de ataque para el Perfil NACA 6412 6](#_Toc37770471)

Índice de Tablas

**No se encuentran elementos de tabla de ilustraciones.**

Índice de Figuras

**No se encuentran elementos de tabla de ilustraciones.**

## Introducción

Con el objetivo de poder realizar cálculos de actuaciones de manera eficiente en Matlab, se ha tomado la decisión de crear un programa en Matlab implementando el modelo de la hélice usado por el departamento de propulsión para hacer el cálculo y diseño preliminar de las hélices. Este modelo consiste en la combinación entre la teoría de cantidad de movimiento de la hélice (TCM) y la teoría del elemento pala (TEP).

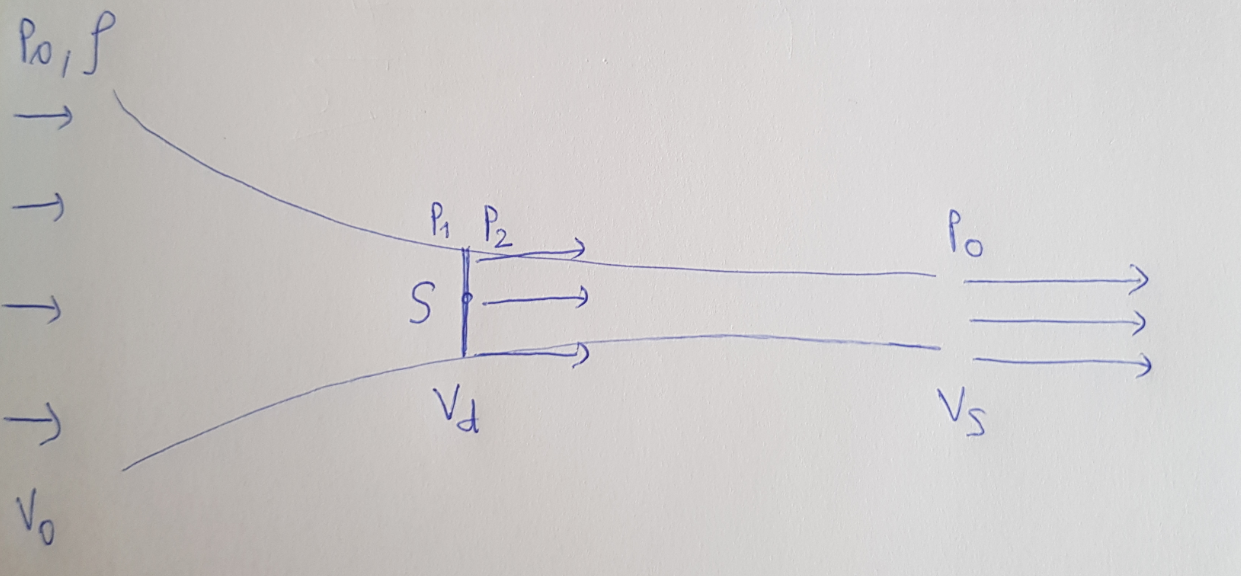
Por otro lado, este programa también sirve no solo para el calculo de actuaciones y control, sino también para realizar el calculo y diseño preliminar de la hélice de manera más sencilla y rápida.

En los siguientes apartados se va a comenzar por describir el modelo matemático de la hélice usado y posteriormente su implementación en Matlab y Octave (el programa funciona en ambos). A continuación, se darán unas instrucciones para el fácil uso del programa y por último se comentará su comparación con el Excel previo de propulsión.

## Modelos físico-matemáticos de la hélice

Con el objetivo de condensar toda la información en un solo informe, y que no sea necesario buscar en otros informes, se va a resumir brevemente los modelos matemáticos empleados.

* 1. **Teoria de Cantidad de Movimiento (TCM)**



La TCM estudia el intercambio de cantidad de movimiento y energía entre una hélice y el fluido contenido en el tubo de corriente que atraviesa la hélice, en la situación más ideal en la que no hay pérdidas de energía mecánica por disipación y toda la energía mecánica se dirige en la dirección axial, sin movimiento rotatorio del fluido.

Con estas hipótesis se puede poner de manera aproximada las siguientes expresiones para el gasto másico, el empuje y la potencia:

G = ρ \* S \* vd

T = G \* (vs – v0)

P = 0.5 \* G \* (vs^2 – v0^2)

Siendo:

S = π \* R^2 = superficie de la hélice ; R = radio de la hélice

La hélice está representada por un disco en el cual aparece una diferencia de presiones a ambos lados. Esta diferencia de presiones se puede calcular aplicando la ecuación de bernouilli delante y detrás de la helice:

p0 + 0.5 \* ρ \* v0^2 = p1 + 0.5 \* ρ \* vd^2

p2 + 0.5 \* ρ \* vd^2 = p0 + 0.5 \* ρ \* vs^2

Restando estas dos ecuaciones se puede obtener:

p2 – p1 = 0.5 \* ρ \* (vs^2 – v0^2)

Ahora, el empuje de la hélice se puede poner también así:

T = (p2 – p1) \* S = 0.5 \* ρ \* (vs^2 - v0^2) \* S

Igualando las dos expresiones del empuje, queda:

0.5 \* ρ \* (vs^2 – v0^2) \* S = ρ \* S \* vd \* (vs – v0)

Operando y simplificando queda finalmente:

2 \* vd = vs – v0

Si se expresan las velocidades del siguiente modo:

vd = v0 + vi

vs = v0 + vi2

Y se sustituyen en la ecuación superior, se obtiene:

vi2 = 2 \* vi

Por lo tanto, vd y vs se pueden poner en función vi (velocidad inducida) como:

vd = v0 + vi

vs = v0 + 2 \* vi

Ahora se puede sustituir este último resultado en las expresiones de G,T,P y dejarlos en función de la velocidad inducida:

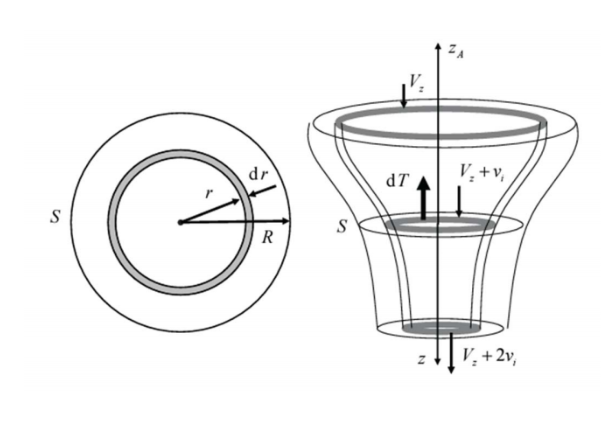
G = ρ \* S \* (v0 + vi)

T = 2 \* ρ \* S \* vi \* (v0 + vi)

P = 2 \* ρ \* S \* vi \* (v0 + vi)^2

**Caso diferencial**

También es posible aplicar la teoría de cantidad de movimiento a una cascara diferencial del tubo de corriente que atraviesa la hélice. Esto es de utilidad para luego combinar esta teoría con la teoría del elemento pala.

****

Los resultados obtenidos anteriormente en forma diferencial resultan :

dG = ρ \* dS \* vd

dT = dG \* (vs – v0)

Siendo el diferencial de superficie:

dS = 2 \* π \* r \* dr

Éstas también se pueden poner en función de la velocidad inducida en esa cascara diferencial. Resolviendo el problema igual que en el caso no diferencial, queda para la cascara diferencial:

vd = v0 + vi

vs = v0 + 2 \* vi

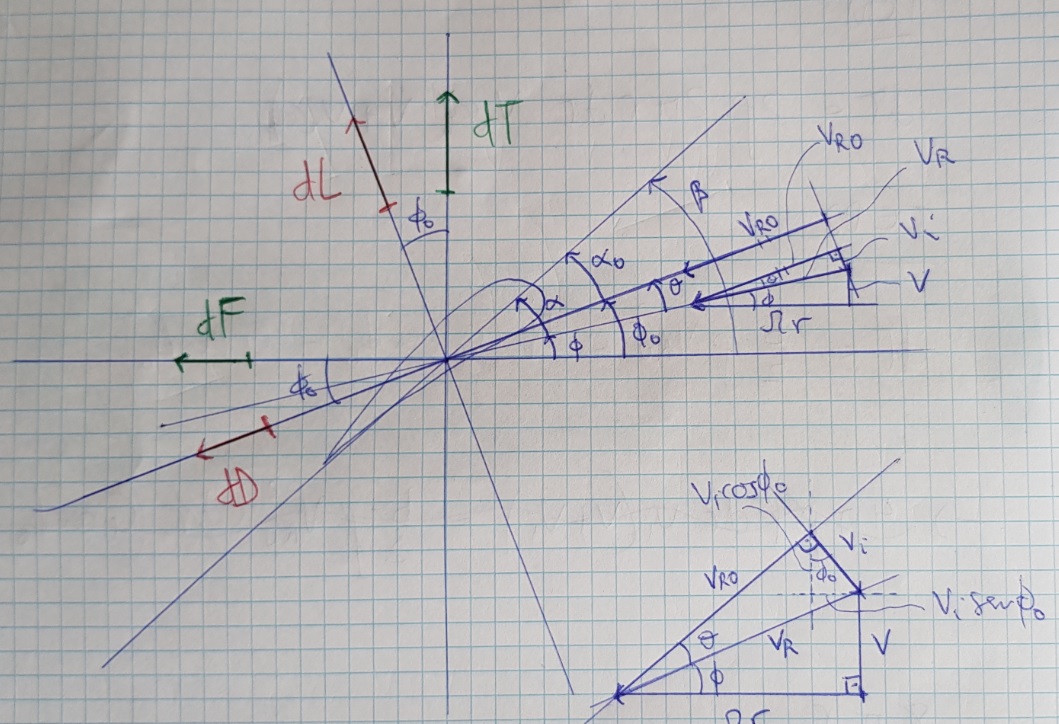
Sustituyendo en las fórmulas de arriba queda:

dG = 2 \* π \* ρ \* r \* dr \* (v0 + vi)

dT = 4 \* π \* ρ \* r \* dr \* vi \* (v0 + vi)

**2.2 Teoría del Elemento Pala (TEP)**

En este modelo se trata de aumentar la complejidad y el realismo de los cálculos respecto del modelo anterior. Para ello se considera un perfil genérico de una pala, “elemento pala”, y se supone que la velocidad inducida es perpendicular a la velocidad incidente que ve el perfil. El diagrama vectorial de velocidades queda como se muestra en la figura inferior.



En dicha figura, “v” es la velocidad de vuelo, “Ω.r” es la componente de la velocidad debida a la rotación de la hélice. Ambas forman “vR”, de tal manera que:

vR = (v^2 + (Ω.r)^2)^0.5

Esta velocidad, junto con la velocidad inducida, forman la velocidad incidente que ve el perfil “vR0”:

vR0 = (vR ^2 – vi^2)^0.5

Los ángulos que forman estas velocidades son: φ, el ángulo que forma Ω.r con vR; θ, el ángulo que forma vR0 con vR.

φ = arctg(v / Ω.r)

θ = arcsen(vi / vR)

Por otro lado, se considera que el elemento pala va a estar operando en condiciones nominales a pequeños ángulos de ataque por lo que las curvas de sustentación y resistencia del perfil se aproximan por una recta y una parábola:

cl = cl0 + clα0 \* α0

cd = cd0 + cd1 \* α0 + cd2 \* α0^2

El ángulo de ataque (α0) se puede calcular en función del ángulo de torsión del elemento pala (β), y los ángulos ya definidos φ y θ :

α = β – φ – θ

Estos dos últimos ángulos, a su vez, pueden servir para definir el ángulo de asiento de la velocidad (φ0):

φ0 = φ + θ

Con todas estas relaciones geométricas y aerodinámicas se pueden expresar ahora la sustentación y la resistencia del elemento pala como:

dL = 0.5 \* ρ \* vR0^2 \* c \* dr \* cl

dD = 0.5 \* ρ \* vR0^2 \* c \* dr \* cd

Y a partir de ellas de pueden obtener la tracción y fuerza tangencial proyectando adecuadamente:

dT = dL \* cos(φ0) – dD \* sen(φ0)

dF = dL \* sen(φ0) + dD \* cos(φ0)

El par aerodinámico del elemento pala se obtiene multiplicando la fuerza tangencial por el radio:

dQ = dF \* r

Con todo ello, las expresiones de tracción y par para el elemento pala se pueden poner como:

dT = 0.5 \* ρ \* vR0^2 \* c \* dr \* (cl \* cos(φ0) – cd \* sen(φ0))

dQ = 0.5 \* ρ \* vR0^2 \* c \* dr \* (cl \* sen(φ0) + cd \* cos(φ0)) \* r

**Combinación TCM y TEP**

Para combinar la TEP con la TCM se igualan los empujes diferenciales (dT) predichos por ambas teorías, para un elemento diferencial del radio de la hélice.

La tracción diferencial según la TEP será el resultado anterior multiplicado por el número de palas (b):

dT = 0.5 \* b \* ρ \* vR0^2 \* c \* dr \* (cl \* cos(φ0) – cd \* sen(φ0))

Por otro lado, la tracción predicha por la TCM, para un elemento diferencial de radio r y espesor dr es:

dT = 4 \* π \* ρ \* r \* dr \* vi \* cos(φ0) \* (v + vi \* cos(φ0))

Igualando ambos términos se obtiene finalmente la ecuación para calcular la velocidad inducida en función de r:

8 \* π \* r \* vi \* cos(φ0) \* (v + vi \* cos(φ0)) – b \* vR0^2 \* c \* (cl \* cos(φ0) – cd \* sen(φ0)) = 0

## Resolución del problema de la hélice

En lo expuesto hasta ahora, se han determinado las ecuaciones del modelo matemático. En este apartado se va a tratar del planteamiento de problema y su resolución. Como modelo para realizar los cálculos preliminares para la hélice, se adopta el modelo TCM + TEP.

Para empezar, se ha de decir que existen dos posibles enfoques del problema y que, antes de nada, debemos tener totalmente definida la hélice, su geométrica y su aerodinámica. Por un lado, podemos suponer conocidas las condiciones de vuelo, altitud y velocidad de vuelo, y la velocidad angular de rotación de la hélice y a partir de ello calcular la tracción, el par y la potencia de la hélice en esas condiciones. Por otro lado, podemos suponer conocidas las condiciones de vuelo (v y h) y una de las siguientes variables: tracción, par o potencia. En este caso la velocidad angular es desconocida e incógnita del problema.

A continuación, se van a exponer cada uno de estos dos enfoques, primero entrando con Ω y calculando T, Q y P, y después entrando con T, Q o P y calculando Ω y las otras dos.

* 1. **Entrando con Ω**

El procedimiento de resolución es el siguiente: Supongamos que partimos de unas condiciones de vuelo conocidas (v y h), unas revoluciones de la hélice conocidas (Ω). Además, conocemos la geometría y la aerodinámica de la hélice, esto es, conocemos:

* v, velocidad de vuelo
* h, altitud de vuelo
* Ω, velocidad angular del rotor
* β = β(r) , distribución de torsión
* c = c(r) , distribución de cuerda
* cl0, clα, cd0, cd, cd2 = f(r) , distribución de perfiles aerodinámicos
* b, numero de palas
* R, radio de la hélice
* k, fracción del radio de la hélice que ocupa el buje central

Siendo r el radio hasta la sección de elemento pala correspondiente, pudiendo r variar entre kR (límite del buje), hasta R (radio de la hélice).

Se trata de calcular las siguientes incógnitas:

* T, tracción de la hélice
* Q, par de la hélice
* P, potencia de la hélice en su eje

Para ello se dispone de las siguientes ecuaciones, vistas anteriormente:

P = Q \* Ω

dT = 0.5 \* b \* ρ \* vR0^2 \* c \* dr \* (cl \* cos(φ0) – cd \* sen(φ0))

dQ = 0.5 \* b \* ρ \* vR0^2 \* c \* dr \* (cl \* sen(φ0) + cd \* cos(φ0)) \* r

vR0 = (vR^2 – vi^2)^0.5

vR = (v^2 + (Ω.r)^2)^0.5

ρ = atmosferaISA(h)

cl = cl0 + clα \* α

cd = cd0 + cd1 \* α + cd2 \* α^2

α = β – φ0

φ0 = φ + θ

φ = arctg(v / Ω.r)

θ = arcsen(vi / vR)

8 \* π \* r \* vi \* cos(φ0) \* (v + vi \* cos(φ0)) – b \* vR0^2 \* c \* (cl \* cos(φ0) – cd \* sen(φ0)) = 0

Para resolver el problema se procede desde abajo hacia arriba, calculando en primer lugar vi y φ en función de r con las últimas ecuaciones y a partir de la distribución de vi, se pueden calcular las distribuciones de θ, α, vR0, dT, dQ. Por último, se puede utilizar la formula compuesta del trapecio para hacer las integrales y obtener los valores de T, Q y P.

* 1. **Entrando con T, Q o P**

Puede resultar más útil resolver el problema a la inversa, es decir, entrar en el problema con la tracción como dato y calcular la Ω que produce esa tracción. Para ello se debe resolver la siguiente ecuación:

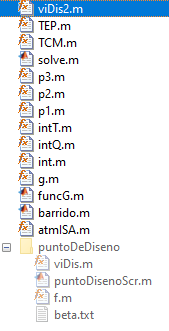
Donde Treq es la tracción que introducimos como dato y dT dependerá de la incognita (Ω). Esta ecuación puede resolverse numéricamente usando las fórmulas anteriores para dT y la fórmula compuesta del trapecio para la integral, y empleando un algoritmo numérico, por ejemplo, el algoritmo de Newton, para resolver la ecuación.

Análogamente si se quiere entrar con Q o P se deben resolver las siguientes ecuaciones respectivamente:

## Implementación en Matlab

* 1. **Introducción**

Para la implementación de este modelo en Matlab, se han programado a una serie de funciones y scripts que se muestran en la siguiente imagen:



Como se puede ver el programa está dividido en dos partes: por un lado, los scripts y funciones que están fuera de la carpeta “puntoDeDiseño” y por otro aquellos que están dentro.

Lo que está en dicha carpeta sirve exclusivamente para obtener la distribución de torsión (beta), a partir de unas condiciones de vuelo determinadas de velocidad, altitud y velocidad angular, para obtener un ángulo de ataque uniforme específico a lo largo de la pala en esas condiciones de vuelo. A esto se le llama “punto de diseño de la hélice” y se emplean como tal las condiciones de vuelo en crucero.

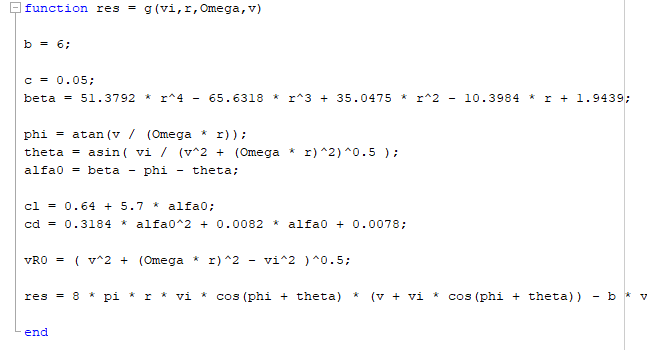
Lo que está fuera de la carpeta es el programa principal, en el que hay que introducir todos los datos geométricos y aerodinámicos de la hélice como datos del problema, incluyendo la distribución de beta calculada. Este programa se puede usar de 2 formas distintas, se puede introducir como dato la velocidad angular (Ω) y, dada la geometría y aerodinámica de la hélice, además de las condiciones de vuelo, calcula la tracción (T), el par (Q) y la potencia (P). La otra forma de usarlo sería introducir como dato T,Q ó P en lugar de Ω, y calcula la velocidad angular que produce ese T,Q o P introducida.

* 1. **Funciones del programa principal**

Voy a hacer ahora un breve repaso por las funciones del programa, explicando cada una de ellas.

**g**

En primer lugar, tenemos la función g, que se muestra a continuación:



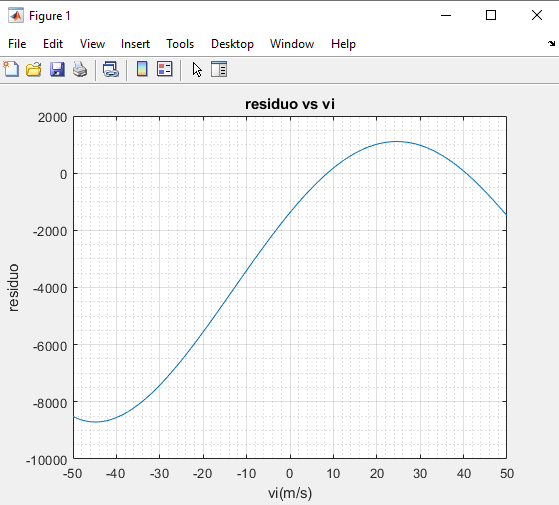
En esta función deben introducirse los datos geométricos y aerodinámicos de la hélice:

* Numero de palas, b
* Distribución de cuerda, c(r)
* Distribución de torsión, β(r)
* Coeficiente de sustentación, cl(r,α0) = cl0(r) + clα(r) \* α0
* Coeficiente de resistencia, cd(r,α0) = cd0(r) + cd1(r) \* α0 + cd2(r) \* α0^2

Esta función calcula el residuo de la ecuación de la velocidad inducida, introduciendo un valor arbitrario de ella en la misma:

8 \* π \* r \* vi \* cos(phi0) \* (v + vi \*cos(phi0)) – b \* vR0^2 \* c \* (cl \* cos(phi0) – cd \* sen(phi0)) = residuo

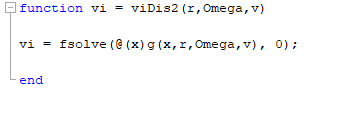
Cuando se introduzca un valor de vi que haga que el residuo sea cero, ese valor de vi será una solución de la ecuación. Si se representa la función del residuo frente a vi, se puede comprobar que corta varias veces al eje x, teniendo por tanto varias soluciones, como se puede ver en la gráfica inferior.



Con respecto a que solución es la correcta, en principio sería la que se encuentra entre 0 y 10 ya que estos son los valores típicos que suele tomar la velocidad inducida.

**viDis2**

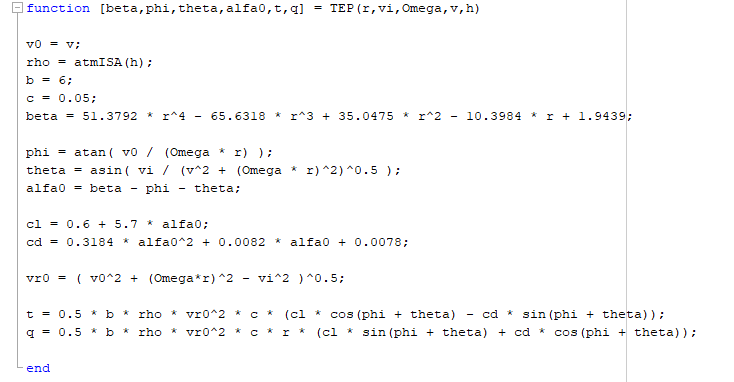
La siguiente función es viDis2. A esta función se le entregan los valores de entrada de r, Ω y v, y resuelve la ecuación de la velocidad inducida haciendo el valor de salida de g cero. Después, devuelve la solución. Se debe calcular una solución para cada valor de r de la discretización elegida a lo largo de la envergadura de la pala.



La condición inicial de fsolve debe situarse próxima a la solución que se desea obtener, ya que, como se ha dicho, g tiene varias soluciones. Si esto no se hace bien, puede dar lugar a resultados erróneos, al converger fsolve a una solución que no es correcta.

**TEP**

Esta función tiene como entradas las condiciones de operación de la hélice (v, h, Ω) y además el radio r del elemento pala correspondiente y su velocidad inducida ya calculada. A partir de estos datos, calcula los ángulos que forman las velocidades en dicho elemento pala (φ, θ) y la tracción y par por unidad de longitud en el mismo (t, q).



En esta función deben introducirse, al igual que en g, los datos geométricos y aerodinámicos de la hélice:

* Numero de palas, b
* Distribución de cuerda, c(r)
* Distribución de torsión, β(r)
* Coeficiente de sustentación, cl(r,α0) = cl0(r) + clα(r) \* α0
* Coeficiente de resistencia, cd(r,α0) = cd0(r) + cd1(r) \* α0 + cd2(r) \* α0^2

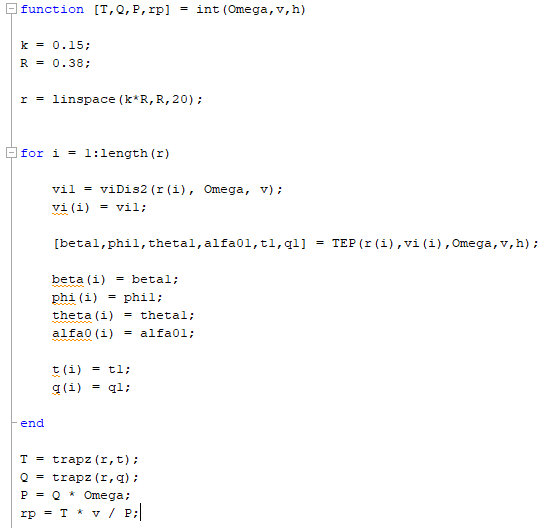
**int**

La función int es la encargada de generar las distribuciones radiales de ángulos y fuerzas, utilizando las funciones viDis y TEP, para posteriormente integrarlas para calcular la tracción, el par y la potencia totales de la hélice. Además, dentro de esta función, hay secciones de código encargado de generar gráficas para visualizar estas distribuciones. Si no se desea que produzcan estas gráficas estas secciones de código pueden ser “comentadas”.

En esta función deben introducirse los datos geométricos de la hélice necesarios para la integración, en este caso:

* k: fracción del radio de la hélice que ocupa en buje central (en tanto por uno)
* R: radio de la hélice.

Como puede verse en el código en la imagen inferior, esta función primero genera una discretización radial de la hélice tomando determinados valores de r entre kR y R. Después calcula la velocidad inducida para cada r usando viDis2, y los ángulos y fuerzas para cada r usando TEP y el valor ya calculado de velocidad inducida en cada sección. Una vez calculadas las distribuciones, las integra en r usando la función de matlab “trapz” (regla del trapecio).



Al observar estas funciones puede deducirse que, dada toda la geometría y aerodinámica de la hélice, incluida dentro de las funciones, las actuaciones globales de la hélice (tracción, par, potencia y rendimiento propulsivo) son solo función de las condiciones de vuelo (v, h) y la velocidad angular del rotor (Ω).

**intQ, intT**

Además de int, se han añadido dos versiones reducidas de esta última, cuyo propósito es servir para el segundo enfoque de resolución del problema, que sería entrar con T,Q o P, sin llevar tanto lastre como lleva la función int. Es conveniente que estas 3 funciones sean 3 funciones distintas por motivos que luego se verán.

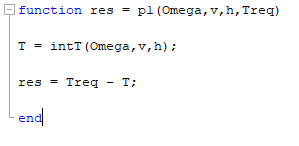
**p1, p2, p3**

Como ya se ha comentado, se pueden adoptar dos enfoques a la hora de resolver el problema de la hélice. Dada toda la geometría y la aerodinámica de la hélice, y las condiciones de vuelo (v, h), se puede entrar en el problema con Ω, y tener por incógnitas ya despejadas T, Q y P. Por otro lado, se puede también entrar con T, Q o P y calcular la Ω como incógnita del problema. El primer enfoque es más sencillo, y con la función int y anteriores es suficiente. El segundo enfoque es más complejo y requiere la definición de funciones adicionales.

Las funciones p1, p2 y p3 tienen como misión formar los residuos de las respectivas ecuaciones a resolver en las tres versiones del segundo enfoque, dependiendo de si se quiere entrar en el problema con T, Q o P.

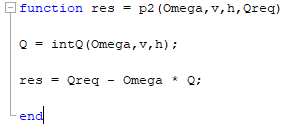
En la versión 1 del segundo enfoque entramos con T, y debemos resolver la siguiente ecuación:

Para ello empleamos la funcion p1, que devuelve el residuo de esta ecuación al introducir un valor de Ω arbitrario.



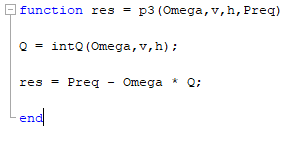
En la versión 2 del segundo enfoque entramos con Q, y debemos resolver la ecuación:

Para ello empleamos la función p2, que devuelve el residuo de esta última ecuación cuando se introduce un valor de Ω arbitrario.



Análogamente, en la versión 3 del segundo enfoque entramos con P y la ecuación a resolver es:

La función p3 es la encargada de formar el correspondiente residuo:



Como puede verse, las funciones p1, p2 y p3 llaman a las funciones intT e intQ ya que estas solo calculan y devuelven un argumento, el requerido en cada caso. Esto es importante para aumentar la velocidad del calculo de Ω, y no realizar operaciones innecesarias en el algoritmo iterativo que va a resolver estas ecuaciones. Por ello, la función int, que calcula todas las distribuciones y fuerzas, en esta segunda parte no se utiliza.

**TCM y atmISA**

Por último, también podemos encontrar la función TCM, que calcula la potencia y el rendimiento propulsivo de la hélice en función del empuje introducido, para unas determinadas condiciones de vuelo. Esta función está aquí solo para poder comparar los resultados que ofrece la teoría de cantidad de movimiento aplicada a la hélice (el modelo más sencillo), con los de el modelo que se esta aplicando a la hélice que combina la teoría del elemento pala con la teoría de cantidad de movimiento. Por otro lado también necesitamos la función atmISA que transforma la altitud de vuelo en densidad del aire.

**Scripts barrido y solve**

Vistas ya todas las funciones importantes, los scripts barrido y solve sirven para hacer barridos y sacar graficas utilizando las funciones definidas anteriormente. El script barrido se utiliza para el primer enfoque, dando las condiciones de vuelo y Ω, y el script solve se utiliza en el segundo enfoque dando las condiciones de vuelo y T, Q o P y resolviendo las ecuaciones descritas anteriormente para obtener Ω.

* 1. **Funciones en punto de diseño**

En la carpeta punto de diseño se encuentran las siguientes funciones:

**f**

Tiene una función similar a g pero a diferencia de ella, aquí se entra con α0 = cte, y se calcula el residuo de la ecuación de la velocidad inducida con esa condición.

**viDis**

viDis tiene una función análoga a viDis2, del programa principal. Esta función resuelve la ecuación de la velocidad inducida en f, con α0 = cte, y posteriormente calcula el ángulo de torsión β del elemento pala en esa posición r, que produce ese ángulo de ataque.

**Script puntoDisenoSrc**

El script en este caso lo que hace es usar las dos funciones anteriores para, dadas las condiciones de operación de la hélice (v, h, Ω), calcular la distribución de torsión β que produzca una distribución determinada de ángulo de ataque α0, a lo largo de la envergadura de la pala. Una vez determinada la distribución de manera discreta, realiza una aproximación por mínimos cuadrados para dar una expresión polinómica que se ajuste a los puntos. Los coeficientes luego son anotados en un documento de texto “beta.txt”, pues luego deberán ser introducidos en el programa principal en g y TEP para definir la distribución de torsión que ha sido diseñada en el punto de diseño.

## Como usar el programa

Usar este programa es sencillo, incluso si no se comprende como funciona por dentro. Solo se deben seguir las instrucciones que se indican a continuación. Como se ha dicho el problema de la hélice admite 2 enfoques de resolución, se va por tanto a explicar como usar el programa con cada enfoque. Para ambos enfoques se asume que se debe conocer toda la geometría y aerodinámica de la hélice, esto es, deben haberse introducido en g y TEP:

* El número de palas, b
* La distribución de torsión, β(r)
* La distribución de cuerda, c(r)
* El coeficiente de sustentación, cl(r, α0) = cl0(r) + clα0(r) \* α0
* El coeficiente de resistencia, cd(r, α0) = cd0(r) + cd1(r) \* α0 + cd2(r) \* α0^2

Y en int:

* k: fracción del radio de la hélice que ocupa el buje
* R: radio de la hélice;

Además, se debe introducir un valor inicial apropiado en fsolve en la función viDis2, siendo valores típicos 0, 10 ó 20.

**Enfoque 1**

El enfoque uno consistía en introducir las condiciones de vuelo(v,h) y la velocidad angular de la hélice(Omega) como datos y obtener la tracción (T), el par (Q), la potencia(P) y el rendimiento propulsivo(rp) en función de ellos. Para ello se introducen las siguientes líneas en la línea de comando de Matlab:

v = 70;

h = 2000;

Omega = 400;

[T,Q,P,rp] = int(Omega, v,h);

La velocidad, altitud y velocidad angular introducidas aquí son a modo de ejemplo, pudiendo el usuario elegir los valores de v, h y Omega que desee. La función int calculará T, Q, P y rp en función de dichos valores.

**Enfoque 2**

El enfoque 2 consistía en introducir como datos las condiciones de vuelo (v,h) y una de las siguientes tres variables T, Q o P, y calcular después la velocidad angular correspondiente. Una vez conocida la velocidad angular se vuelve al enfoque 1 para calcular el resto de variables.

Para hacer todo esto introducimos en la línea de comando de Matlab las siguientes líneas:

**Para entrar con la tracción:**

v = 70;

h = 2000;

T = 625;

Omega = fsolve(@(x)p1(x,v,h,T), 400);

[T2,Q,P,rp] = int(Omega,v,h);

Se puede comprobar si la ecuación se ha resuelto bien viendo si T2 = T. Tambien se puede calcular el valor del residuo poniendo:

res = p1(Omega,v,h,T);

Y comprobando se res = 0;

**Para entrar con el par:**

v = 70;

h = 2000;

Q = 150;

Omega = fsolve(@(x)p2(x,v,h,Q), 400);

[T,Q2,P,rp] = int(Omega,v,h);

De nuevo, se puede comprobar si ha salido bien viendo si Q2 = Q, o bien calculando el residuo y comprobando que sea cero:

res = p2(Omega,v,h,Q);

**Para entrar con la potencia:**

v = 70;

h = 2000;

P = 60000;

Omega = fsolve(@(x)p3(x,v,h,P), 400);

[T,Q,P2,rp] = int(Omega,v,h);

Aquí de nuevo de puede comprobar si ha salido bien, fijándose si P2 = P o bien calculando el residuo:

res = p3(Omega,v,h,P);

En los tres casos, los valores introducidos son a modo de ejemplo y el usuario puede introducir los que quiera. El programa calculará Ω y los restantes en función de ellos.

La condición inicial introducida en fsolve debe estar próxima el valor esperado de velocidad angular, por si las funciones tuvieran varias soluciones o por si fsolve no convergiera para cualquier condición inicial. En este caso se ha introducido 400 rad/s en la condición inicial.

**Barridos**

Si se quiere, por ejemplo, realizar un barrido en velocidad, se puede hacer del siguiente modo:

v = 30:10:90;

h = 2000;

Omega = 400;

for i = 1:length(v)

[T1,Q1,P1,rp1] = int(Omega, v(i),h);

T(i) = T1;

Q(i) = Q1;

P(i) = P1;

rp(i) = rp1;

end

De esta forma se construyen los vectores T, Q, P, rp en función del vector creado de velocidad. Luego se pueden representar estos puntos en una gráfica escribiendo, para la gráfica T – v, por ejemplo:

figure

plot(v,T,”marker”,”+”)

grid on

grid minor

title(“T vs v”)

xlabel(“v(m/s)”)

ylabel(“T(N)”)

Si se quiere hacer el barrido en Ω, puede hacerse del mismo modo, pero generando esta vez un vector en Omega:

v = 70;

h = 2000;

Omega = 300:20:500;

for i = 1:length(Omega)

[T1,Q1,P1,rp1] = int(Omega(i),v,h);

T(i) = T1;

Q(i) = Q1;

P(i) = P1;

rp(i) = rp1;

end

Y para representar la gráfica T – Omega se haría, igual que en el caso anterior:

figure

plot(Omega,T,”marker”,”+”)

grid on

grid minor

title(“T vs Omega”)

xlabel(“Omega(rad/s)”)

ylabel(“T(N)”)

Por último, para hacer un barrido en velocidad, por ejemplo, desde el segundo enfoque se puede hacer del mismo modo, en este caso habría que escribir lo siguiente:

v = 30:5:90;

h = 2000;

T = 625;

for i = 1:length(v)

Omega(i) = fsolve(@(x)p1(x,v(i),h,T), 400);

[T1,Q1,P1,rp1] = int(Omega(i),v(i),h);

T2(i) = T1;

Q(i) = Q1;

P(i) = P1;

rp(i) = rp1;

end

Podemos comprobar si ha salido bien, viendo si T2 = T, para todo i, o bien calcular el valor del residuo para cada Omega:

for i = 1:length(Omega)

res(i) = p1(Omega(i),v(i),h);

end

Que a su vez luego se puede representar.

**Punto de diseño**

Utilizar puntoDeDiseño para diseñar la hélice y obtener la distribución de torsión adecuada es aún más sencillo. Solamente hay que introducir en puntoDisenoScr las condiciones de operación de la hélice en las que se desea realizar el diseño (v, h, Ω) y la distribución de ángulo de ataque de diseño (α0).

Es importante asegurarse de que se ha introducido en la condición inicial de fsolve, en viDis, un valor razonable y cercano a la solución esperada para que fsolve converja hacia ella (normalmente 0 ó 5 ó 10)

Después hay que ejecutar el script y se generará la distribución de torsión y de velocidad inducida correspondientes. Los coeficientes del polinomio de β calculado quedarán anotados en un documento de texto “beta.txt” para que puedan ser copiados al programa principal para el cálculo de actuaciones de esa hélice.

Se puede comprobar, al introducir el β calculado en el programa principal, que para las condiciones de operación de la hélice en el punto de diseño, sale efectivamente la distribución de α0 que se había impuesto previamente. Esto se puede hacer descomentando la gráfica correspondiente a las distribuciones de ángulos en int, para que int las represente y poder ver como son.

**Unidades**

Se pueden introducir las magnitudes en el programa en las unidades que se quiera, siempre y cuando se sea luego coherente y consistente con las unidades de las magnitudes de salida, dadas las relaciones que existen entre las diferentes magnitudes a través de las ecuaciones del problema.

Lo más recomendable es usar siempre las unidades del sistema internacional, y en concreto medir los ángulos en radianes y las velocidades angulares en radianes por segundo. Si se introducen todas las magnitudes de entrada en unidades del sistema internacional, las unidades de las magnitudes de salida también serán las del sistema internacional.

## Excel

## Conclusiones

Con este programa se pueden hacer cálculos preliminares para el diseño y para el calculo de actuaciones de una hélice de una manera fácil y rápida.

Además, este programa puede servir no solo para este proyecto sino para otros ya que se plantea el calculo de una hélice genérica. También puede servir para el diseño preliminar de turbinas eólicas puesto que el modelo matemático sería el mismo.

Por su rapidez y sencillez, sería interesante validar este programa con datos de CFD para comprobar su fiabilidad y ver maneras de mejorarlo en caso de que sea necesario.