Tutorial RMOP - XFOIL

CIMNE - Servei de Transferència de Teconologia

CIMNE - Aeronàutica

Centre Internacional de Mètodes Numèrics a l'Enginyeria

Terrassa, 30 de Enero de 2013

Índice

1. Introducción **2**

2. Descripción del problema **2**

3. Preparación de la optimización **3**

3.1. Variables de diseño 4

3.2. Magnitudes a optimizar 4

3.3. Funciones objetivo 5

3.4. Diagrama de flujo de datos 5

3.5. Programación de los módulos 6

3.5.1. Analyser (PRE) 7

3.5.2. Analyser (POST) 7

3.6. Comprobaciones previas a la optimización 8

4. Introducción de los datos en la GUI de RMOPv1.0 **8**

4.1. Población 9

4.2. Rango de valores de la DVs 10

4.3. Optimización 11

4.4. Analyzer 12

4.5. Ejecución 13

5. Análisis de los resultados **13**

Referencias **17**

1. Introducción

A continuación se presenta una aplicación sencilla para ilustrar el funcionamiento de RMOPv1.0 a modo de tutorial. El tutorial se puede utilizar tanto para conocer el funcionamiento general del optimizador, aprender a usarlo y analizar resultados, una vez esta vinculado al *solver*, como para profundizar en la creación del vínculo con el programa y dominar la herramientas que se disponen para ello.

Para ello se vinculará el optimizador con el programa XFOIL, ver [1], un programa de paneles para calcular las fuerzas aerodinámicas sobre un perfil alar.

La vinculación entre los dos programas, y en casos con más *solvers* también, se realizará mediante un programa, llamado *Analyser* en este texto, el cual se debe crear específicamente para cada aplicación. En este tutorial se da un ejemplo en C/C++, aunque se puede realizar con otros lenguajes de programación o incluso integrarlo en el propio *solver*.

Aunque en este documento se muestra detalladamente el uso de la interfaz gráfica (GUI) para este caso, se puede profundizar más sobre su uso consultando el manual de la GUI hallado en la web de RMOP, ver [2].

2. Descripción del problema

En primer lugar se va a describir el problema de optimización que se quiere resolver.

El objetivo de la optimización es encontrar un perfil aerodinámico que se que comporte eficientemente en los rangos de operación de la aeronave. Que a la velocidad de crucero tenga una relación sustentación/resistencia alta así como un coeficiente de sustentación alto cuando vuela a baja velocidad para aterrizar o despegar. Como ejemplo de aplicación se tomará una avioneta ligera con las siguientes características relevantes para el problema:

- Cuerda media aerodinámica: $MAC = 1{,}5m$
- Velocidad de crucero: $V_{CR} = 150\frac{km}{h} = 41{,}7\frac{m}{s}$
- Velocidad de despegue/aterrizaje: $V_{TO} = 75\frac{km}{h} = 20{,}8\frac{m}{s}$

Con los datos anteriores se pueden obtener los números de *Reynolds* aproximados de crucero, Ec. 1, y despegue/aterrizaje, Ec. 2. Siendo $\nu = 1{,}5 \cdot 10^{-5}$ la

viscosidad cinemática del aire a nivel del mar.

$$Re_{CR} = \frac{V \cdot MAC}{\nu} = \frac{41{,}7 \cdot 1{,}5}{1{,}5 \cdot 10^{-5}} \approx 4 \cdot 10^6 \tag{1}$$

$$Re_{TO} = \frac{V \cdot MAC}{\nu} = \frac{20{,}8 \cdot 1{,}5}{1{,}5 \cdot 10^{-5}} \approx 2 \cdot 10^6 \tag{2}$$

El programa XFOIL se encargará de calcular las fuerzas aerodinámicas, o los coeficientes de fuerza que son equivalentes, para todos los diseños, que dentro de unos rangos el optimizador proponga. Para ello habrá que crear el programa *Analyser* que vincule los dos programas. La comunicación entre los distintos programas se realiza mediante archivos de texto.

Para ejecutar el proceso de optimización hay que definir unas funciones objetivo que se deberán calcular a partir de los resultados obtenidos de XFOIL. En el apartado 3.4 se detalla el flujo de datos y los módulos que intervienen.

3. Preparación de la optimización

Para vincular un *solver* con el optimizador hay que seguir los siguientes pasos:

1. Definir las variables de diseño (DVs) y sus respectivos rangos de valores.
2. Definir las magnitudes que se quieren optimizar.
3. Definir las funciones objetivo (FOs).
4. Crear el diagrama de flujo de datos.
5. Programar los distintos módulos del diagrama.
6. Comprobaciones previas a la optimización.

Los pasos descritos están detallados en los apartados 3.1, 3.2, 3.3, 3.4, 3.5 y 3.6, respectivamente . Para casos con mayor complejidad, en los que se pueden estar trabajando con varios archivos de texto y con más de un *solver*, facilitará la tarea de programar los módulos del *Analyser*.

3.1. Variables de diseño

Las variables de diseño que se utilizarán en este ejemplo son las especificadas en la tabla 1. Los valores del *baseline* son los del perfil de partida, un *NACA2412*. Las distancias se trabajan adimensionalizadas con la cuerda media aerodinámica del ala (MAC).

Tabla 1: Variables de diseño.

Variable	Baseline	Mínimo	Máximo	Real / Entero
Espesor máx. $t\,[adim]$	0.12	0.05	0.19	Real
Curvatura máx. $cmb\,[adim]$	0.02	0.01	0.03	Real
Pos. espesor máx. $x_{m_t}\,[adim]$	0.3	0.2	0.4	Real
Pos. curvatura máx. $x_{m_{cmb}}\,[adim]$	0.4	0.3	0.5	Real

En la figura 1 se muestra un esquema de la variables de diseño.

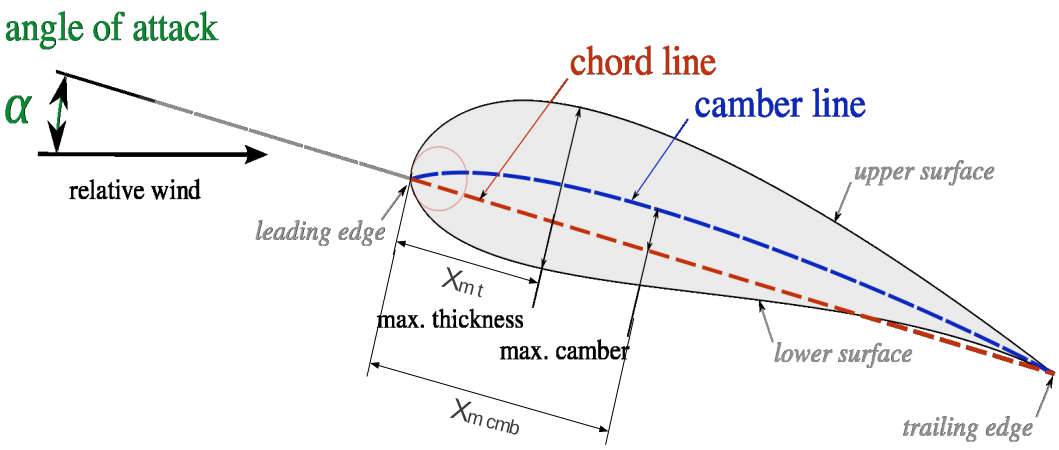

Figura 1: Esquema de las variables de diseño [3].

3.2. Magnitudes a optimizar

Se va a optimizar dos magnitudes del problema. Se quiere maximizar el coeficiente de sustentación máximo C_{lmax} a un número de *Reynolds*, $Re = 2 \cdot 10^6$, de

despegue y aterrizaje. Por otro lado se requiere una eficiencia, E, del perfil alta,

$$E = \frac{C_l}{C_d}$$

para *Reynolds* de crucero, $Re = 4 \cdot 10^6$, con $\alpha = 3[^{\text{o}}]$ de ángulo de ataque.

3.3. Funciones objetivo

Hay que crear una función objetivo para cada magnitud a optimizar. Las funciones objetivo deben ser mínimas para los valores óptimos de los parámetros a optimizar. En el caso que se está tratando se proponen las dos FOs siguientes, que son las inversas de los valores a maximizar, aunque se podrían utilizar funciones distintas:

$$f_1 = \frac{1}{C_{lmax}} \tag{3}$$

$$f_2 = \left.\frac{C_d}{C_l}\right|_{\alpha=3[^{\text{o}}]} \tag{4}$$

3.4. Diagrama de flujo de datos

En la figura 2 se presenta el diagrama de flujo de datos propuesto.

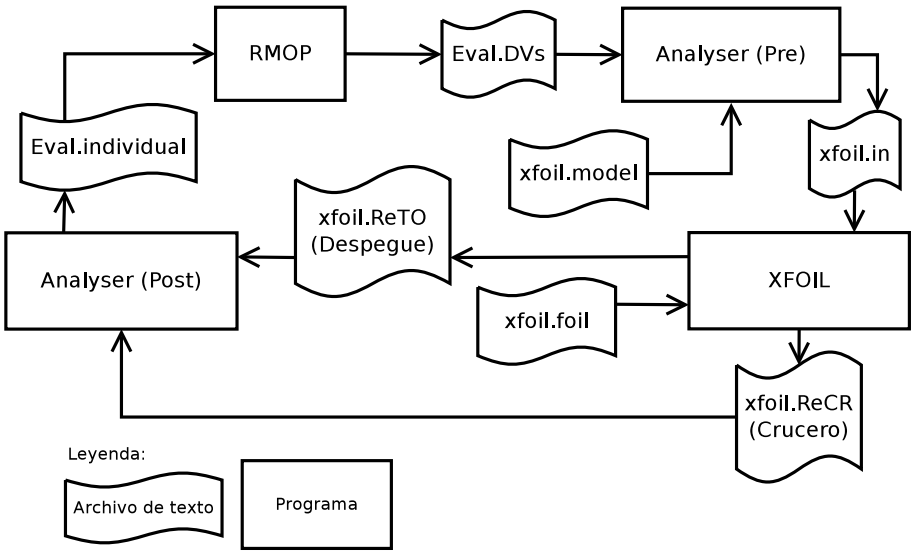

Figura 2: Diagrama de flujo de datos.

Tal como se aprecia en el diagrama, el *Analyser* se divide en dos partes, la previa al cálculo, *Analyser (PRE)* y la que evalúa las FOs después del cálculo de XFOIL, *Analyser (POST)*. Las dos partes estarán contenidas en un solo programa, que preparará los archivos, ejecutará XFOIL y evaluará las FOs.

Adjuntos a este documento se proporcionan los siguientes archivos:

1. **xfoil.model:** Contiene el modelo de archivo de cálculo de XFOIL para calcular los casos que se requieren.
2. **xfoil.foil:** Contiene los puntos del perfil de referencia, geometría *baseline*.

El archivo *xfoil.in* se debe crear desde el *Analyser (PRE)*, con los valores de las DVs substituidas en el archivo *xfoil.model*.

XFOIL creará los archivos *xfoil.ReTO* y *xfoil.ReCR*, con los resultados del cálculo para el *Reynolds* de despegue y el de crucero respectivamente.

A partir de de los resultados de XFOIL, en el *Analyser (POST)* se deben calcular los valores de las FOs y escribir el archivo *Eval.individual*.

A modo de recordatorio de la documentación de RMOPv1.0 (`http://tts.cimne.com/RMOP/`), los archivos *Eval.DVs* y *Eval.individual* son los archivos mediante los que se establece la comunicación con el optimizador independientemente del *solver* utilizado.

- **Eval.DVs:** Contiene los valores de las variables de diseño con las que el optimizador quiere que se evalúen las FOs.
- **Eval.individual:** Es el archivo en el que el optimizador lee los valores de las FOs correspondientes a las DVs del archivo *Eval.DVs*.

3.5. Programación de los módulos

En este punto se explican los puntos importantes a tener en cuenta en la programación de los módulos, pero no se va a explicar con detalle la algoritmia. Se propone que el lector, programe los módulos en caso de querer profundizar en el tema, o por el contrario utilice el posible código adjunto, *AnalyserXFOIL.cpp*, que cumple las especificaciones requeridas.

3.5.1. Analyser (PRE)

El módulo PRE solo debe realizar la substitución de los valores del archivo *Eval.DVs* en el archivo *xfoil.model* y escribir el archivo *xfoil.in*. Ésto se puede realizar mediante la librería *AnalyserTools*, usando la clase *SUBS*. Para más información consultar la documentación de la clase *SUBS* [4].

3.5.2. Analyser (POST)

La programación del módulo POST se debe realizar tal que se obtengan los datos de los archivos *xfoil.ReTO* y *xfoil.ReCR*, se calculen las funciones objetivo Ec. 3 y Ec. 4, y se escriba el resultado en el archivo *Eval.individual*.

Es muy importante asegurar que en caso de que el cálculo de alguno de los individuos, no converja, no de los resultados suficientes para evaluar las FOs, o haya cualquier tipo de problema en los resultados, los valores que se escriban en el archivo *Eval.individual*, sean valores que penalicen dichos individuos, es decir, valores altos.

***Reynolds* de despegue**

Para el *Reynolds* de despegue se debe obtener el coeficiente de sustentación máximo. Para ello se propone ajustar una parábola a los tres puntos superiores de la curva $C_l - \alpha$, calculados por XFOIL y buscar el máximo de dicha parábola. En la figura 3 se muestra el problema gráficamente.

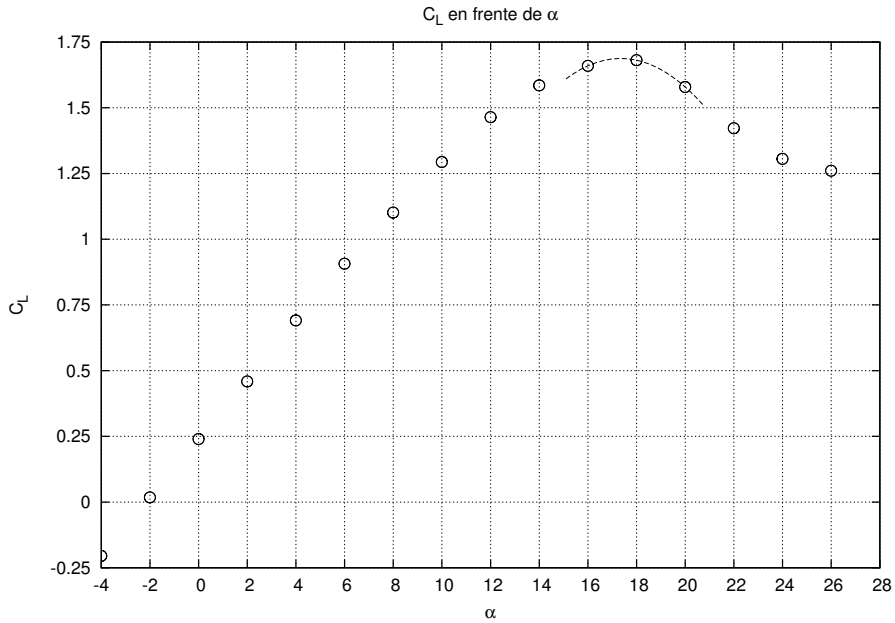

Figura 3: Curva C_l en frente de α.

***Reynolds* de crucero**

Para la condición de vuelo de crucero el procedimiento es más sencillo, dado que solo hay que leer los valores de los coeficientes de sustentación y resistencia, directamente del archivo *xfoil.ReCR*. Para calcular la segunda función objetivo, Ec. 4 y escribirla en el archivo *Eval.individual*.

3.6. Comprobaciones previas a la optimización

Las comprobaciones previas a la optimización que se recomienda realizar son, ejecutar el *Analyser* con los valores máximos y mínimos de las DVs, así como con los valores del *Baseline*. Se debe comprobar que los casos ejecutan correctamente, tanto en el módulo del *solver*, XFOIL en este caso, como los módulos del *Analyser*.

4. Introducción de los datos en la GUI de RMOPv1.0

A continuación se describe paso a paso cómo introducir los datos del problema en la GUI de RMOPv1.0 . Para resolver cualquier duda sobre el uso de la interfaz,

se puede consultar el manual, ver [2].

Al ejecutar el programa se abre la ventana principal, figura 4.

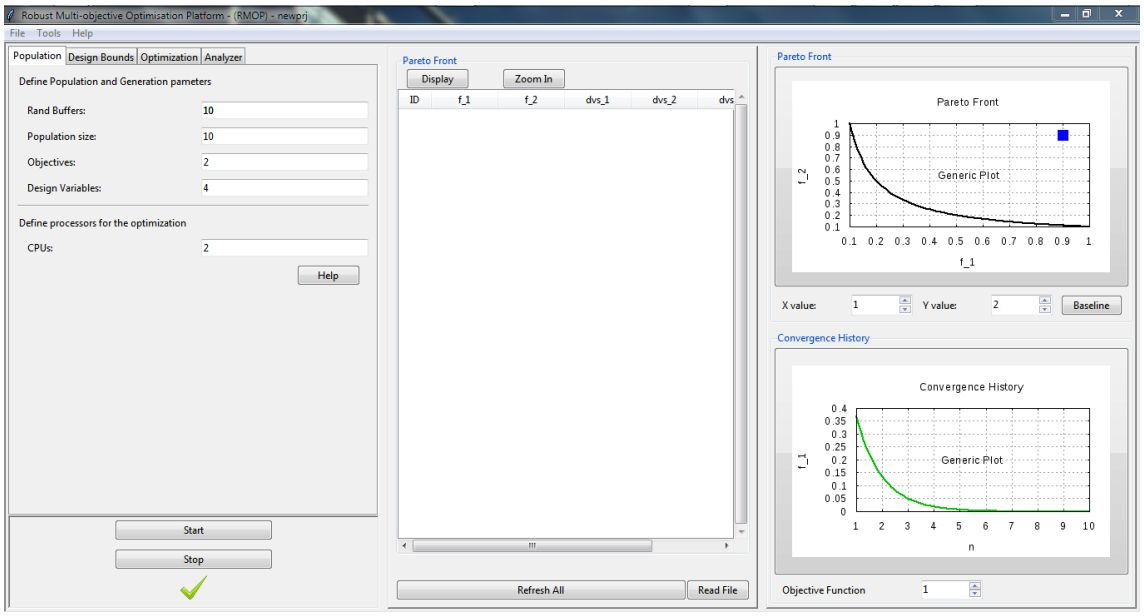

Figura 4: Ventana principal de la GUI-RMOPv1.0 .

4.1. Población

En primer lugar se informa el tamaño de la población en la pestaña llamada "*Population*". Además en esta pestaña se definen los parámetros relacionados con la población de individuos y el número de procesadores que se encargarán de ejecutar en paralelo todo el proceso de optimización.

El tamaño que se asignará a cada generación "*Population size*" es de 10 individuos. El *buffer* de individuos en este caso será de 10 ("*Rand Buffers*" de 10). En el campo "*Objectives*" se debe introducir el número de funciones objetivo, que en este caso son 2, como se ha especificado en el apartado 3.3. De la misma manera el número de variables de diseño son 4, especificadas en apartado 3.1. El número de *CPUs* no afectará a los resultados, pero si al tiempo de ejecución, se recomienda dejar, como mínimo un procesador libre, en este caso se utilizaran 2, como se muestra en la figura 5, junto con el resto de parámetros.

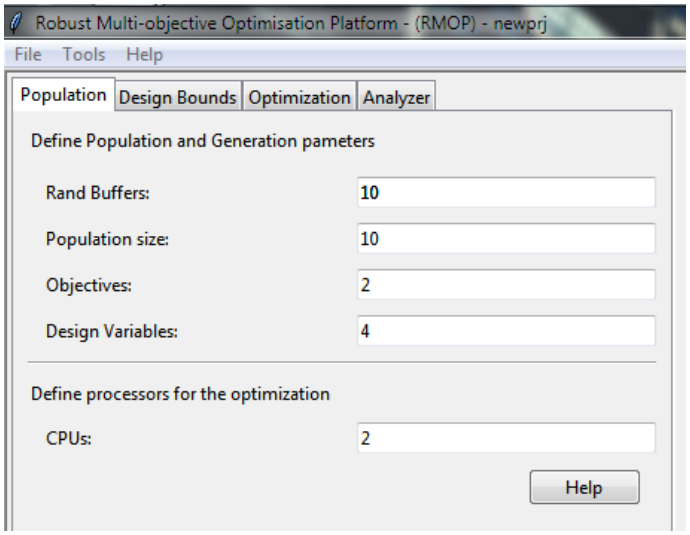

Figura 5: Pestaña ”Population”.

4.2. Rango de valores de la DVs

El rango de la variables de diseño se informa en la pestaña ”*Design Bounds*”, figura 6. Además se asigna si se trata de una variable entera o real. Para añadir una nueva variable de diseño, se debe apretar el botón ”*add*”.

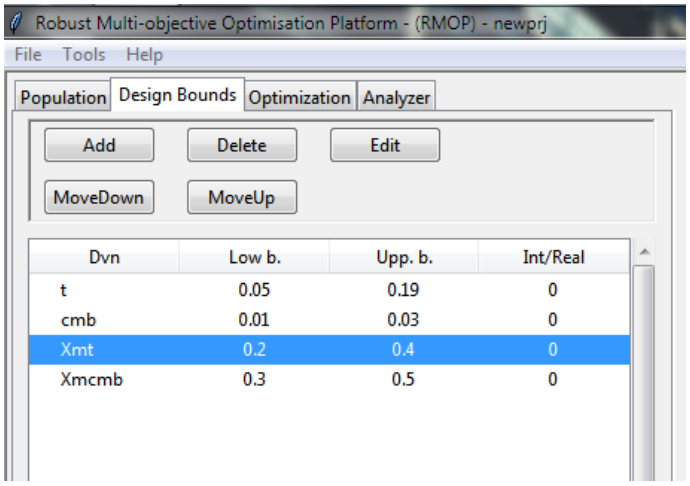

Figura 6: Pestaña ”*Design Bounds*”.

Aparecerá una nueva ventana, figura 7, en la que se pueden introducir los datos. El nombre de la variable, campo ”*DV identifier*”, no se usa en el cálculo, está destinada a facilitar la introducción y revisión de los datos. Se asigna el máximo valor y el mínimo que puede asumir la variable de diseño concreta en

los campos "*Upper Bound*" y "*Lower Bound*" respectivamente. Las variables de diseño y los límites de cada una están especificadas en el apartado 3.1, tabla 1.

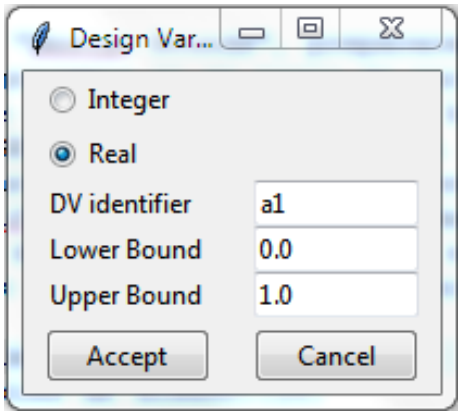

Figura 7: Ventana "*Design Variable*".

4.3. Optimización

La pestaña "*Optimization*", figura 8, se configuran los parámetros del optimizador. En "*Game Strategy*" se debe seleccionar "*Pareto Game*", en el subapartado "*Choice of optimization method*", "*Genetic Algorithim*" y finalmente en el subapartado "*Choice of optimization progress monitoring*" se selecciona la opción "*Update to file*".

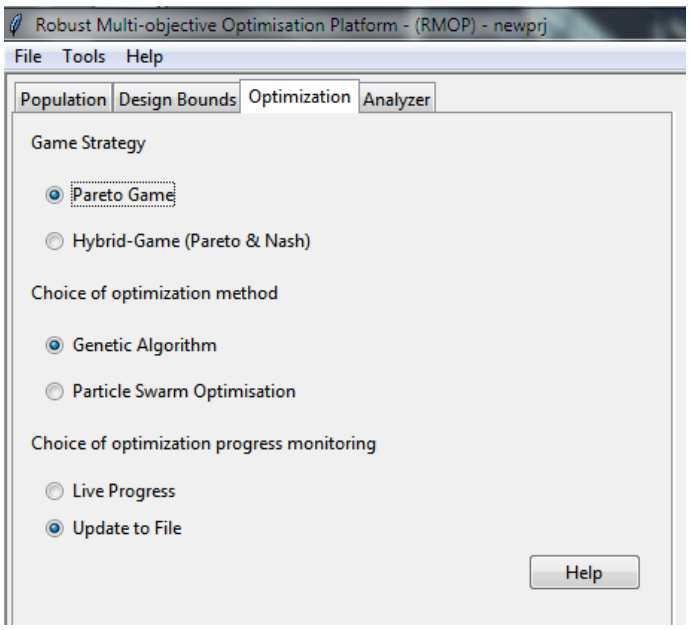

Figura 8: Pestaña "*Optimization*".

4.4. Analyzer

En la última pestaña ”*Analyzer*”, figura 9, se vincula el optimizador con el *Analyser* que se ha programado anteriormente, junto con los criterios de parada de la optimización. En el subapartado ”*Termination Criteria*”, donde se elige que la señal de parada del proceso de optimización venga dada por un tiempo en horas, para ello se debe seleccionar la opción ”*Elapse Time (Hours)*”, y en el campo ”*Termination Value:*” se asignarán las horas que estará funcionando el optimizador, los resultados que se presentan en es te documento son con 0.5 horas.

En el apartado ”*Essential Files*”, se añadirán los dos archivos que se necesitarán para este caso, que son ”*xfoil.foil*” y ”*xfoil.model*”, especificados en la página 6. Para ello apretamos el botón ”*Add*” y se abrirá una nueva ventana donde se puede introducir el nombre del archivo en cuestión que se puede añadir. Se pulsa ”*Accept*” y ya estará. En el caso de que se quiera eliminar un essential file concreto, se debe apretar el botón ”delete” y se eliminará el ”*Essential File*” que se tenga seleccionado, y lo mismo para el botón ”*edit*”, con la diferencia de que en lugar de eliminar el essential file que se tenga elegido, se podrá editar.

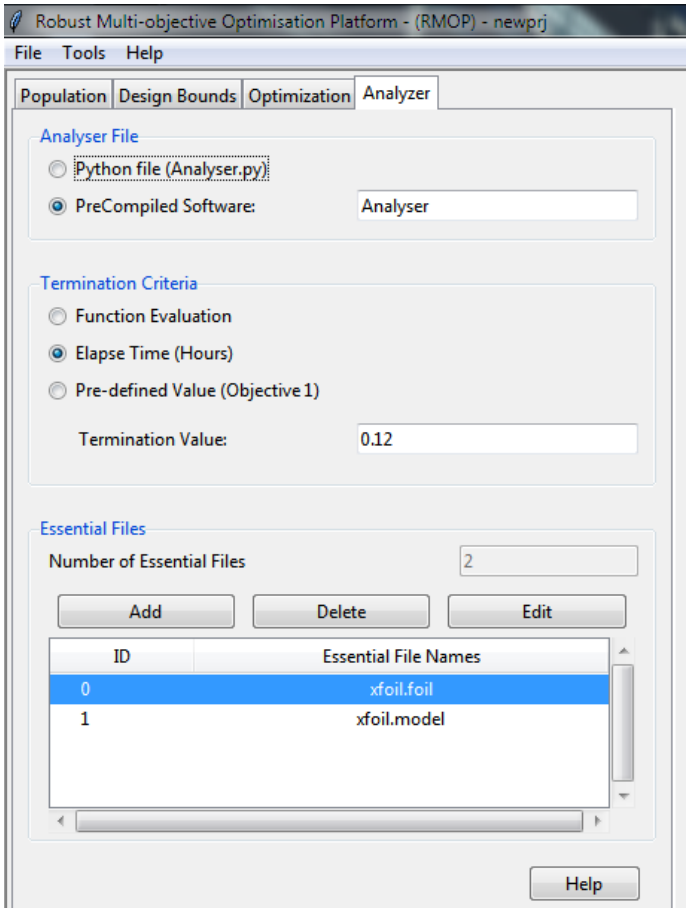

Figura 9: Pestaña "*Analyzer*".

4.5. Ejecución

Una vez preparado el caso, para ejecutarlo se debe pulsar en "*Start*" y comenzará el cálculo. Durante la ejecución se puede monitorizar la convergencia de f_1 *Pareto front* pulsando el botón *Refresh All* y en la parte derecha de la ventana principal se actualizan los gráficos.

5. Análisis de los resultados

En este apartado se analizan brevemente los resultados de la optimización. En primer lugar se comprueba la convergencia de la 1ª función objetivo, que está disponible en la *GUI* de HPRMOP. En la figura 10 se muestra el gráfico de convergencia obtenido para la optimización con 410 individuos evaluados. En el gráfico se puede observar la rápida mejora en unos 100 individuos y que con 290 indivi-

duos se ha obtenido la mejor mejora de f_1.

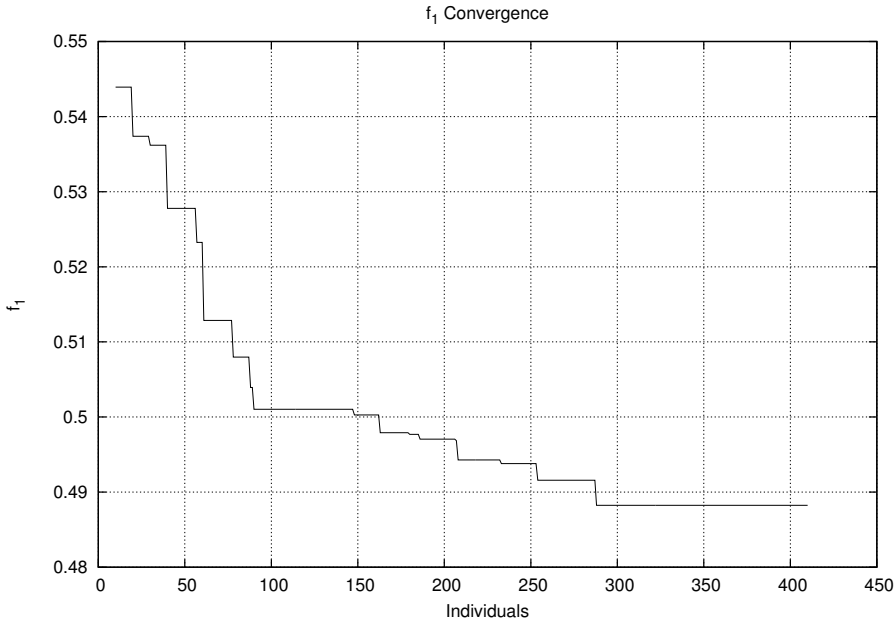

Figura 10: Gráfico de convergencia de f_1.

En la figura 11 se muestra el gráfico del *Pareto front*. En éste se aprecia mejoría, respecto al *Baseline* en las dos funciones objetivo, aunque no todos los individuos mejoran en ambos aspectos. Dado que el *Pareto front* contiene los individuos tales que no se puede mejorar en un aspecto sin perjudicar el resto, el individuo del *Pareto front* más adecuado para una determinada aplicación se deberá seleccionar mediante otros criterios, no contemplados en este proceso de optimización.

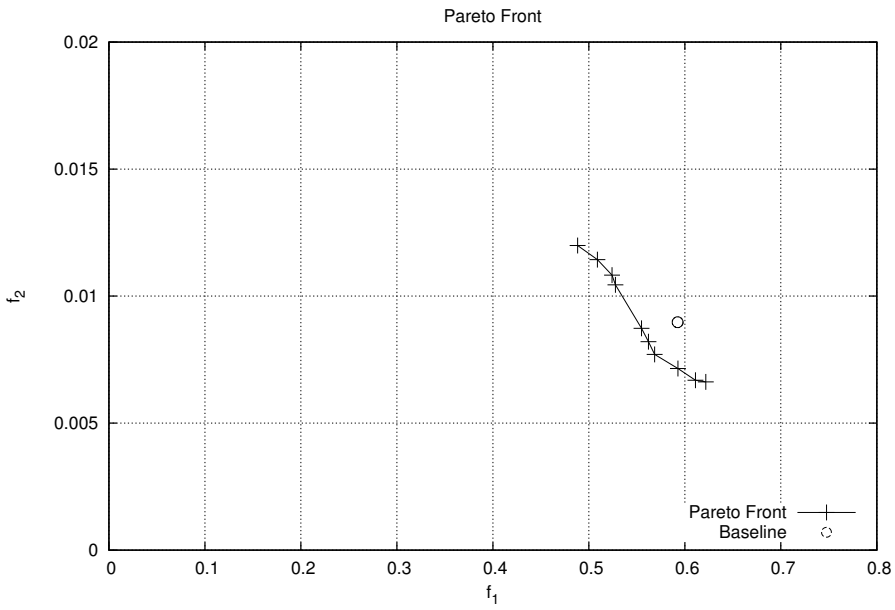

Figura 11: Pareto front.

La forma de los perfiles 1, 5 y 10 del *Pareto front* (1: Menor valor de f_1, 10: mayor valor de f_1) se muestran en la figura 12.

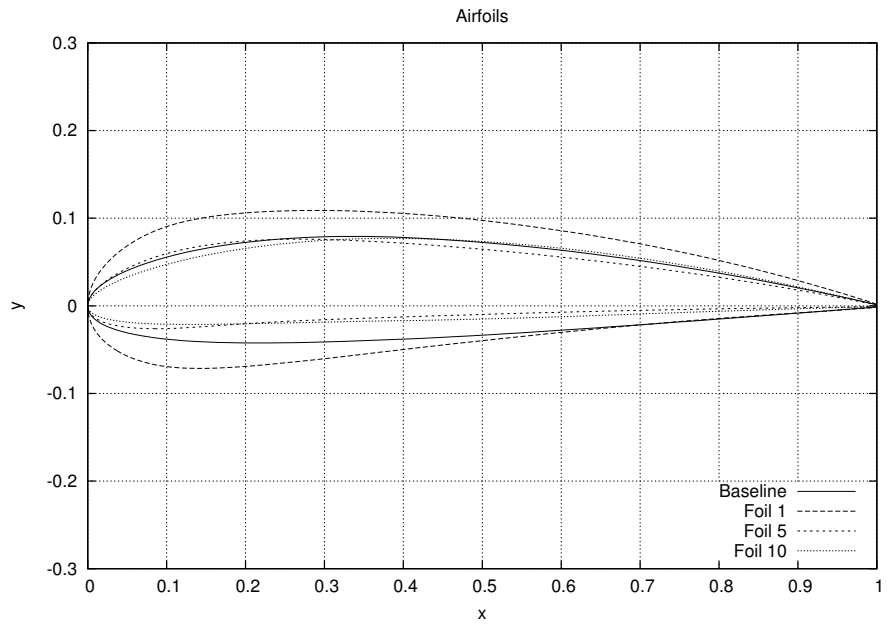

Figura 12: Geometría de los perfiles obtenidos en la optimización.

En la figura 13 se presenta el gráfico $C_l - \alpha$ de tres perfiles del *Pareto front* además del *baseline*.

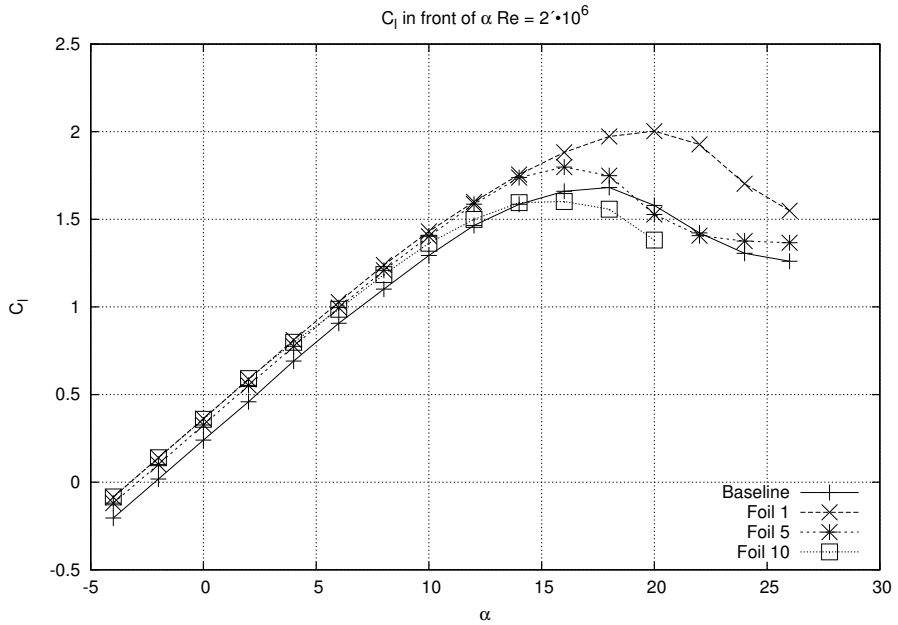

Figura 13: $C_l - \alpha$ para distintos perfiles obtenidos, $Re = 2 \cdot 10^6$.

En la figura 14 se presenta el gráfico $E - \alpha$ de tres perfiles del *Pareto front* además del *baseline*.

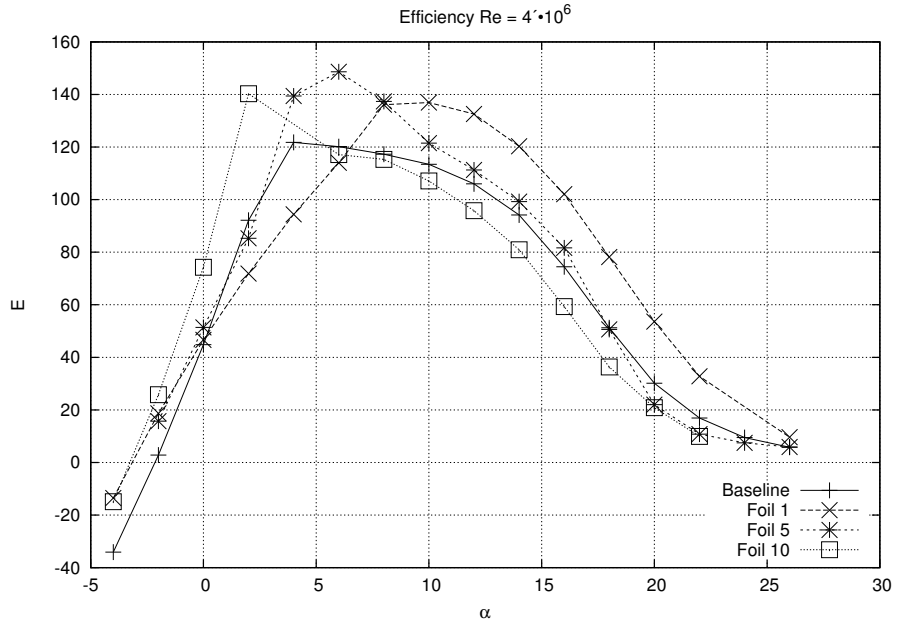

Figura 14: $E - \alpha$ para distintos perfiles obtenidos, $Re = 4 \cdot 10^6$.

Referencias

[1] XFOIL web site: http://web.mit.edu/drela/Public/web/xfoil/

[2] RMOP-GUI Manual: (http://tts.cimne.com/RMOP/Docs/gui_manual/gui_manual.pdf)

[3] http://en.wikipedia.org/wiki/Airfoil

[4] Documentación de la clase SUBS-RMOP: (http://tts.cimne.com/RMOP/Docs/analyser_tools/SUBS-RMOP_ES.pdf)