Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXXVI, págs. 499-510 (artículo completo)
José G. Etse, Bibiana M. Luccioni, Martín A. Pucheta, Mario A. Storti (Eds.)
San Miguel de Tucumán, 6-9 Noviembre 2018

AUTOMATIZACIÓN DE GENERACIÓN DE MALLAS PARA LA OPTIMIZACIÓN DE PROPULSORES NAVALES

MESH GENERATION AUTOMATISATION FOR OPTIMISATION OF NAVAL PROPELLERS

Gustavo E. Carr[a], Alejandro Vaccari[b], Julio Giménez[b], Yael Sánchez[b], Nicolás Biocca[a], Nicolás Antonelli[b], Juan Francisco Martínez[b] y Santiago A. Urquiza[b]

[a]*CONICET - Mar del Plata, Argentina, http://mardelplata-conicet.gob.ar/*

[b]*Grupo HidroSim, Universidad Tecnológica Nacional, Facultad Regional Mar del Plata, Buque Pesquero Dorrego Nº 281,Mar del Plata, Argentina, email:hidrodinamica@mdp.utn.edu.ar*

Palabras clave: Scripts, Python, Salome, diseño, propulsores navales, hidrodinámica.

Resumen. Las técnicas de modelado computacional son herramientas que hacen posible la introducción a bajo costo de mejoras para optimizar el desempeño de propulsores navales en condiciones realistas de servicio. Debido a que deben ser adaptados a cada embarcación, existe la necesidad de automatizar los modelos CAD de los impulsores y su mallado posterior para los cálculos CFD involucrados en cada lazo de optimización. En este trabajo se elige la plataforma GNU Salome (https://www.salome-platform.org) como entorno de trabajo y se desarrolla código en lenguaje Python para automatizar la reconstrucción geométrica y el mallado asociado al dominio de fluido que envuelve a cada variante de impulsor. Las series que se utilizan son las publicadas por el canal de Wageningen: Serie B (o serie de Troost) y Serie Ka (serie Kaplan). Como resultado de los mismos, se obtienen mallas de elementos finitos realizadas paramétricamente para ambas series y se varían diversos parámetros geométricos ilustrando la robustez de los códigos desarrollados en una amplia gama de alternativas de diseño.

Keywords: Scripts, Python, Salome, Design, naval propellers, hidrodynamics.

Abstract. Nowadays, computer modelling techniques are allowing the introduction of relevant improvements to optimise naval propellers behaviour lowering the costs involved. Given that the propellers must be adapted to each particular vessel, the need of CAD propellers model automatisation and further meshing for CFD calculation arises. In the present work the GNU+Linux Salome Platform was chosen as a working environment. Python language scripting code is developed to automatise the geometric model and meshing, associated to the fluid domain enclosing each propeller variant. The propellers series that are used were published by the Wageningen Canal: B series (or Troost series) and Ka series (Kaplan series). As a result, parametrically generated finite element meshes are obtained for both series and for different geometric parameters, showing the robustness of the code developed for a wide range of design alternatives.

1. INTRODUCCIÓN

En el pasado, el proyecto de buques consistía en adaptar diseños existentes, realizando modificaciones basadas en el conocimiento empírico de los profesionales. La evaluación del desempeño de dichos diseños sólo era posible, a posteriori, por métodos experimentales en canales de ensayos o de acuerdo al comportamiento de los buques en servicio. En las últimas tres décadas, el modelado computacional viene haciendo posible, cada vez con mejor capacidad predictiva, calcular los rendimientos de los sistemas permitiendo evaluar las correcciones y mejoras propuestas de manera virtual y a bajo costo, minimizando, a su vez, la cantidad de los onerosos ensayos en canales de experiencias hidrodinámicas. Por otra parte, es necesario además tener en cuenta que el proyecto de buques es una actividad muy compleja donde intervienen entre otros: velocidad y tiro de la embarcación, comportamiento en el mar y maniobrabilidad, francobordo y longitudes inundables, dimensiones principales, formas, cálculos hidrostáticos, estabilidad en condición intacta y en avería, arreglo general de casco y máquinas, estructura, propulsión y costos. Consecuentemente, hoy en día en la industria naval argentina no es usual someter a análisis exhaustivos, sean éstos experimentales o computacionales, al comportamiento hidrodinámico esperado de los nuevos diseños de propulsores y carenas. Los astilleros nacionales más importantes no suelen realizar optimizaciones de los mismos, sino que se utilizan modelos de series sistemáticas ya probadas y ensayadas, debido a la falta tanto de recursos humanos capacitados como de una tradición de investigación y desarrollo en el área. En el caso de las hélices, los modelos más utilizados son los correspondientes a la serie B y a la serie Ka de Wageningen (Oosterveld. (1970), Bernitsas. et al. (1987) y Kuiper (1992)). Las primeras para el caso de los buques que necesitan alto rendimiento en velocidad (buques de carga como graneleros, buques tanque, buques Ro-Ro) y las segundas para las embarcaciones cuyo principal requisito es el tiro (remolcadores, pesqueros de arrastre). Cada serie, a su vez se divide en "familias" y cada "familia" en "individuos". Las características geométricas que son constantes para todas las familias (de cada serie) son la forma de los perfiles y la ley de espesores máximos. Complementariamente, los parámetros que son constantes dentro de cada familia de hélices son el número de palas y la relación de áreas palas/disco (Fa/F, el área desarrollada total de todas las palas de la hélice dividida por el área de la circunferencia que circunscribe a la hélice (Bernitsas. et al., 1987)). Surge así naturalmente la denominación típica que se expone con un ejemplo: Hélice B5.60, significa que el propulsor pertenece a la serie B, tiene 5 palas y una relación $Fa/F = 0,60$. Luego, dentro de cada familia, los individuos se diferencian entre sí por su paso y su diámetro, lo cual generalmente se define con dos parámetros, la relación paso/diámetro y la dimensión de éste último. Esta manera de brindar la información de estos dos parámetros facilita el poder comparar propulsores y establecer mejor las semejanzas geométricas, ya que, por ejemplo, dos hélices de igual paso, pero distintos diámetros serán mucho más diferentes entre sí que dos hélices de distinto paso y diámetro, pero igual relación entre ambos valores.

En resumen, la metodología utilizada actualmente en la mayoría de los casos es partir de varios modelos propuestos por ingenieros con experiencia en el área, seleccionándose los de mejor rendimiento basados en la experiencia, sea esta obtenida de datos previos o de ensayos experimentales ad hoc.

Con el objetivo de evaluar la resistencia al avance, tiro y cupla y comportamiento frente a cavitación, se está trabajando sinérgicamente a nivel internacional en modelado y experimentación en canales hidrodinámicos a escala a fin de correlacionar resultados obtenidos con ensayos físicos.

Es así que en la actualidad se propone el uso de técnicas de optimización computacional

utilizando diseños paramétricos, buscando mayor eficiencia por la reducción de tiempos involucrados en el ciclo de diseño y la reducción significativa de los costos, tanto de fabricación como de desempeño en servicio, disminuyendo el uso de criterios subjetivos a la vez que se gana en rapidez y versatilidad (figura. 1).

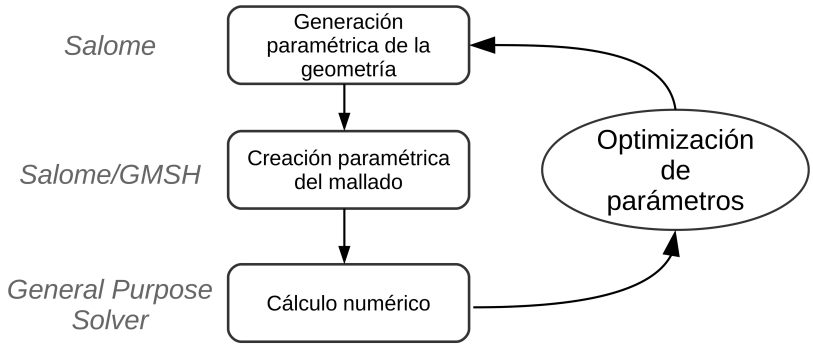

Figura 1: Esquema del proceso de diseño de hélices propulsoras para la industria naval.

El empleo de técnicas de optimización de manera eficaz demanda naturalmente que las geometrías sean parametrizadas de tal forma que los algoritmos de optimización puedan encontrar valores óptimos de los parámetros que gobiernan la representación geométrica de cada individuo dentro de una serie sistemática. Los métodos para optimización de geometrías más utilizados son los parcial y totalmente paramétricos, donde se parte de un diseño inicial que depende de parámetros de entrada que definen su geometría, se realizan los cálculos y a partir de los resultados obtenidos se modifican dichos parámetros para lograr optimizar determinados funcionales costo. Por ende, se utilizan geometrías variables paramétricamente, pre-procesamientos, simulaciones computacionales (Urquiza y Venere, 2001), post-procesamientos y finalmente, algoritmos de optimización, propiamente dichos. Algunas de dichas estrategias, que varían en robustez y precisión, son: heurísticos, multi-estrategia, no diferenciales, y basados en gradientes (Miao y Wan (2017), Harries (2014)).

El paso previo necesario al empleo de algoritmos de optimización es desarrollar herramientas de construcción de formas paramétricamente por medio de su modelado geométrico. De esta manera se obtienen, automática y desatendidamente, las mallas de elementos finitos sobre las que se aplicará la optimización de formas. Consecuentemente, es este trabajo se desarrolla código de automatización paramétrica basado en la suite de libre disponibilidad Salome (Salome, 2015), sobre la que se implementan "scripts" en lenguaje Python (Van Rossum, Guido, 1995) funcionando sobre una distribución Debian 9.4 (SPI and others, 1997-2017) del sistema operativo GNU+Linux, para programar la totalidad de los pasos de la generación de las mallas desde la propia geometría parametrizada. Se espera que esto despierte gran interés por parte de los actores del mercado de la construcción de buques y hélices. Al trabajar en el estudio de las tareas de diseño de propulsores, se contribuiría a mejorar las bases teóricas y procedimientos utilizados por los ingenieros navales en nuestro país y la región.

En el resto del trabajo se describirán brevemente las herramientas desarrolladas, sus principales características y potencialidades, basándonos en el caso de las series sistemáticas de propulsores navales. De forma similar, el software desarrollado puede ser fácilmente extendido al caso de las carenas de buques y otras embarcaciones.

2. PARÁMETROS PRINCIPALES DE DEFINICIÓN DE LA GEOMETRÍA DE PROPULSORES

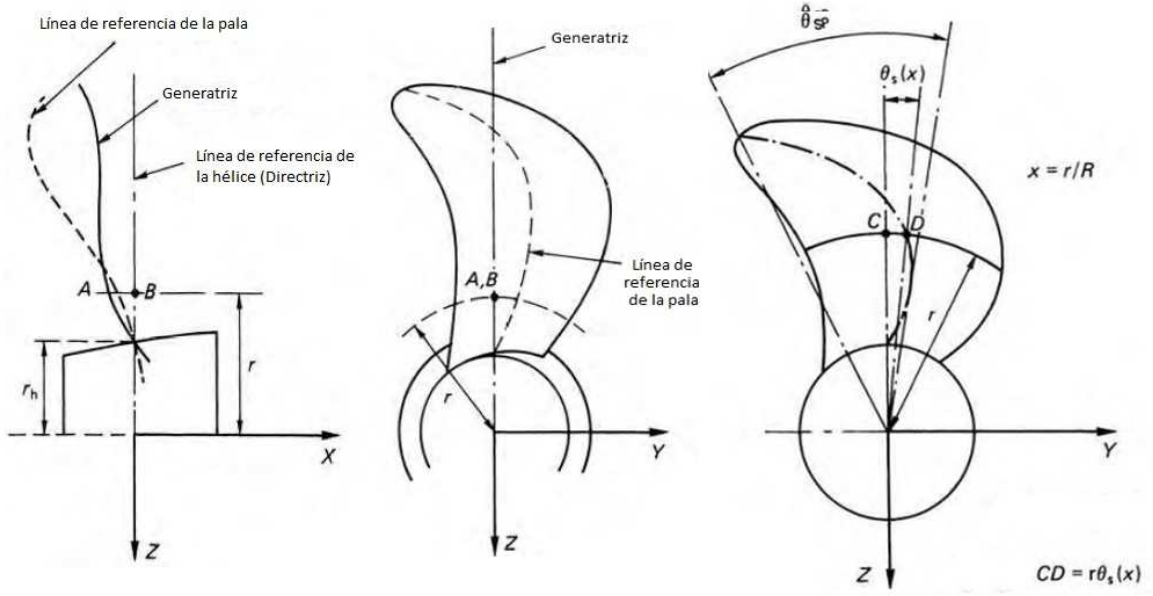

Figura 2: Elementos básicos para la descripción geométrica de una pala de propulsor naval ((Carlton, 2007), figura 3.2, capítulo 3, página 34 y figura 3.7, capítulo, página 38).

Antes de definir los parámetros principales definiremos las líneas de referencia que nos permitirán describir la geometría de la hélice. Estas líneas son:

Líneas de referencia

- **Directriz** (o línea de referencia de la hélice): es una línea recta que es perpendicular al eje (principal) de la hélice.
- **Generatriz**: es una línea que, también es perpendicular al eje de la hélice pero no es recta, ya que, vista en perfil suele formar una curva en el plano x-z.
- **Línea de referencia de la pala**: es una línea que pasa por los puntos medios de cada sección cilíndrica.

Diámetro: junto con el número de palas, es el parámetro más simple de visualizar en una hélice, ya que es el segmento de recta que pasa por el centro y une los extremos de las palas, es decir es el correspondiente a la circunferencia circunscripta en la hélice.

Paso: es la distancia, en el sentido axial, que se desplaza un punto cualquiera cuando la hélice completa una revolución. Esto se visualiza en la figura 3, donde se adoptó la rotación alrededor del eje $\hat{x}$.

Skew: es un parámetro que define las inclinaciones (conjuntamente con el Rake), y por ende las transformaciones que sufre cada sección. Es el ángulo entre la directriz y la recta que pasa por el centro de la hélice y por el punto de cuerda media de la sección. Es importante destacar que la distancia CD se calcula como el arco de radio r y ángulo θ_S, y es lo que se conoce como "distancia skew"de cada sección, o simplemente "skew". Cada uno de esos puntos (desde el inicial en la raíz, hasta el final en el extremo de la pala) define lo que se conoce como curva de

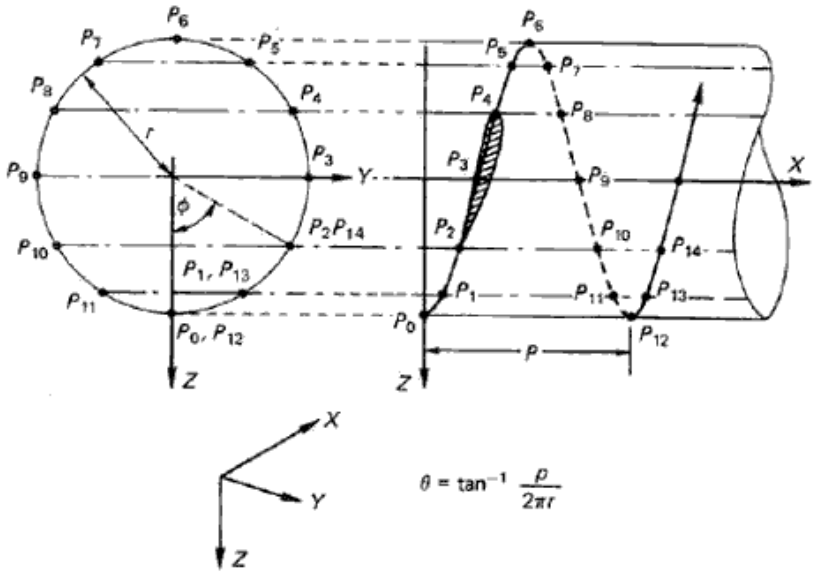

Figura 3: Descripción del paso de una hélice (Carlton (2007), figura 3.4, capítulo 3, página 35).

skew, o de cuerdas medias (Midchord curve), de suma importancia. Esto puede apreciarse en la sección de la derecha de la figura 2.

Rake: es el otro parámetro importante que, junto con el skew, define las transformaciones que sufre cada sección. El Rake, o ángulo de lanzamiento, de la hélice se divide en dos componentes: el rake de la generatriz (Generator line rake), y el rake inducido por el skew. De forma que tenemos: $i_T(r) = i_G(r) + i_S(r)$, siendo: $i_G(r)$, el rake que se mide en la vista de perfil, simplemente como la distancia AB (en la Figura 2, centro) que se mide paralela al eje x, desde la directriz hasta el punto donde la sección de radio r corta al plano XZ; y $i_S(r)$ es el rake generado por el skew de la sección, más concretamente, por el desenvolvimiento de dos secciones cilíndricas entre sí, una en la raíz de la pala y otra en cualquier punto de radio r (figura. 2, derecha). Es decir, para obtener el rake total de la sección r, simplemente se suman ambas componentes de forma directa.

3. MÉTODOS DE GENERACION DE LOS MODELOS GEOMETRICOS Y DE LAS MALLAS ASOCIADAS

3.1. Representación geométrica

Las hélices son generadas algorítmicamente utilizando el programa de software libre Salome (Salome, 2015), que implementa la creación y manipulación de sólidos B-rep (boundary representation, (Braid et al., 1978) a través de la tecnología OpenCascade (Open Cascade Company, 2017) implementada dentro del software. Esta representación B-rep genera al sólido a partir del espacio encerrado en una o varias superficies en un espacio tridimensional. Dichas superficies están generadas a partir de secuencias de segmentos encadenados, y son además definidas por funciones matemáticas paramétricas. A su vez, los segmentos quedan determinados en el espacio tridimensional a partir de sus vértices y una función paramétrica. El continuo completo consiste en un espacio cilíndrico que rodea a la hélice y sus geometrías accesorias (cono y eje). Este sólido es generado restando de un cilindro, que representa al fluído, la unión de los sólidos correspondientes a la unión de la palas, el cono y el eje.

Dado que la forma de cada pala depende de la cantidad de ellas para una dada hélice, este

sistema brinda la posibilidad de testear el mallado usando una sola pala calculada como perteneciente a una hélice de mayor número de palas. Esto acorta los tiempos de procesamiento y mallado en varias veces.

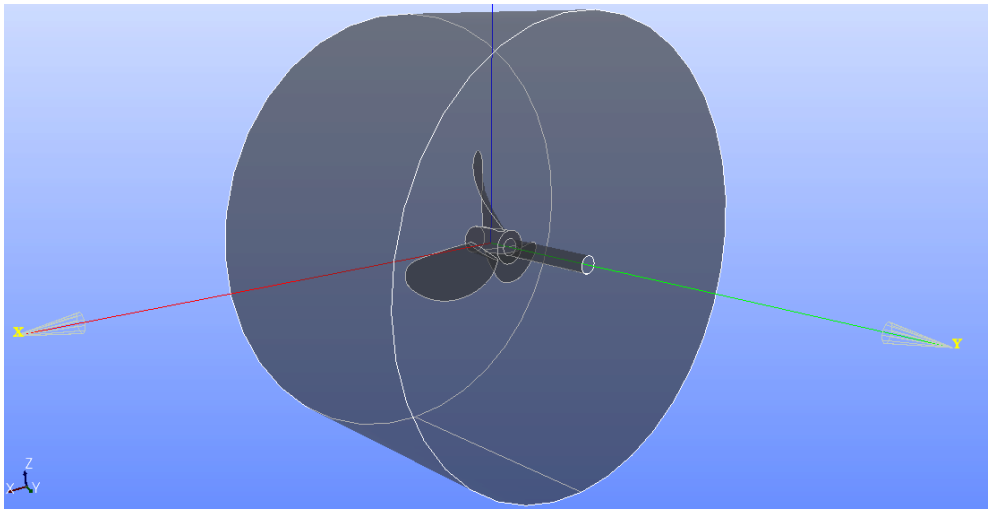

Figura 4: Vista del continuo a mallar, para un caso de hélice de tres palas.

La creación de las palas de hélice se realiza a partir de las series de puntos en tres dimensiones que se obtienen mediante la implementación de “The Wageningen Propeller Series” en un código en lenguaje Python desarrollado especialmente. La geometría, a partir de los puntos, es definida en base a la cantidad de palas, el diámetro de la hélice, y la relación Fa/F (relación entre el área desarrollada total de todas las palas de la hélice y el área de la circunferencia que circunscribe a la hélice). Las series de puntos corresponden a secciones transversales a la pala de la hélice a distancias determinadas, ordenadas radialmente desde el eje.

Dichas series son interpoladas mediante curvas tipo spline polinómicas de alto orden de interpolación, para ser posteriormente “envueltas” en una superficie tridimensional del mismo orden que las curvas splines. Estas superficies, intersectadas con la cáscara cónica que conforma el núcleo de la hélice, son utilizadas para generar un sólido el cual es restado a un cilindro sólido de mayor diámetro que la hélice y que resulta ser el recinto de cálculo a mallar. Asimismo, son utilizadas para generar los segmentos de intersección entre cono y palas.

3.2. Implementación computacional en lenguage Python

La plataforma Salome trae provistas copias de las bibliotecas de lenguaje Python, como numpy, sys, math, scipy y otras. Al importar dichas bibliotecas el usuario accede a éstas y no a las instaladas en el sistema operativo. Esto es igual a la inversa: no es posible acceder desde un script corriendo en el sistema oprerativo a las clases que Salome contiene. Por ello, el script principal debe ser ejecutado desde Salome. Éste corre autónomamente y genera las geometrías y la creación de mallas con sus respectivos parámetros. La finalización de la ejecución deja al usuario la opción de ejecutar a mano el cómputo de la malla, para posibilitar una inspección previa de todo lo realizado en cuanto a modelado de sólidos y dimensionamiento.

El programa generador de mallados paramétricos utiliza el paradigma de objetos para la gestión de tamaños de elemento, entre otras características de las mallas. Debido a que las palas de esta serie definen una topología constante de una pala a otra, es posible hallar cada segmento paramétricamente y definir un tamaño de elemento para la malla a generar. Esto se realiza a través

de instancias de dos clases definidas ad-hoc: `Segmento` y `Cara`, las cuales contienen como atributos tanto los parámetros de la malla (tamaño de elementos, tipo de algoritmo de semilleo) como también los objetos pertenecientes a la geometría homóloga de la cual forman parte. El programa comienza su ejecución leyendo desde un único archivo de texto los parámetros para la generación de la geometría de la pala y los parámetros del mallado. A partir de estos datos el programa realiza tres actividades: primero efectúa el cálculo del grupo de series de puntos, luego genera la geometría y finalmente crea y computa la malla. El cálculo de series de puntos se realiza como se describe a continuación. Se define primero un objeto `geompy`, constructor de geometrías propio de Salome, luego de haber importado el módulo correspondiente.

```
import GEOM
from salome.geom import geomBuilder
geompy = geomBuilder(myStudy)
```

A continuación se presenta el núcleo de cálculo, perteneciente al método `generarPuntos()` de la clase `Pala`, para obtener la posición en el espacio de cada punto de las series que definen la pala patrón. Se genera el punto P requerido según la geometría de la sección:

```
P = geompy.MakeVertex (Xi, Yi, 0)
```

Como la geometría de las series sistemáticas de Wageningen toman como origen de coordenadas el punto de mayor espesor de cada sección, se debe desplazar todos los puntos hasta la generatriz. Este desplazamiento se realiza en dos etapas, primero se lo lleva al punto correspondiente al skew de la sección y luego, desde ese punto, a la generatriz. Los desplazamientos son siempre en el sentido horizontal, es decir en el eje de las abscisas.

```
geompy.TranslateDXDYDZ(P, Xdesp, 0, 0)
geompy.TranslateDXDYDZ(P, skew, 0, 0)
```

Se rota según el eje $\hat{z}$ (saliente al plano de dibujo), el ángulo de giro correspondiente al paso de cada sección.

```
geompy.Rotate(P, OZ, alfa)
```

Para trasladar cada punto a su posición correcta, se los coloca en $x = 0$ y se los eleva la distancia correspondiente a su radio. Para colocarlos en $x = 0$, simplemente se extrae el valor de x correspondiente, y se lo traslada, en la dirección $\hat{x}$, una distancia igual a ese valor en sentido contrario.

```
coordsP= geompy.PointCoordinates(P)
Ps = coordsP[0]
geompy.TranslateDXDYDZ(P, -coordsP[0], 0, ((n+1.0)/10)*(D/2))
```

Luego, se debe girar cada punto según el ángulo que resulta del sector circular de cuerda Ps y radio r, como se muestra en la figura 3, de forma tal de transformar la sección de plana a cilíndrica.

```
angP = Ps/(((n+1.0)/10)*(D/2))
geompy.Rotate(P, OY, angP)
```

Finalmente, el último paso es trasladar el punto hacia atrás, la distancia igual al ángulo de Rake correspondiente. Además, el punto se agrega a la lista de puntos que luego conformarán la sección.

```
geompy.TranslateDXDYDZ(P, 0, -rake, 0)
listPuntos.append(P)
```

Durante la generación de geometrías, el programa crea objetos vértice (según se describe anteriormente, puntos en el espacio materializados en instancias de la clase `Vertex`, nativa de Salome) pertenecientes a cada uno de los segmentos y superficies que se van generando para luego ubicarlos al momento de realizar un explotado (método `Explode()` en Salome) del contínuo a mallar y entonces asignar los parámetros de semillas y tamaños de elemento a cada elemento. Esto debe ser así ya que el programa Salome genera nuevos objetos de memoria para cada geometría resultante de una operación booleana o de una extracción de entidades a partir de un elemento existente. Para ello, los objetos de la clase `Vertex` son utilizados a modo de handle o referente respecto a cuál segmento o superficie corresponden. Para que los grupos de elementos de malla aparezcan identificados en el archivo resultante se asocian las geometrías derivadas del continuo con los correspondientes objetos handle.

3.3. Descripción de los principales objetos creados para este script

El script es un sistema para la creación de mallas paramétricas que consta de diferentes clases, instanciadas en objetos. La clase principal, `Generador` (su instancia es ejecutada en el bloque principal del programa en ćodigo Python), contiene por composición a las instancias de las clases que generan los diferentes elementos necesarios para la creación de las geometrías sobre las cuales se elaborarán las mallas correspondientes.

Existen clases para lectura y escritura de archivos. Estas son: `LectorDeParámetros` y `Escritor`. El objeto de la clase `LectorDeParámetros` lee desde un archivo de texto los parámetros necesarios para crear las geometrías y las mallas.

La clase principal está provista de un atributo, `dataDic`, el cual es del tipo diccionario y contiene los parámetros leídos por el objeto `LectorDeParámetros`. De esta manera están disponibles para los demás objetos.

Como se muestra en los siguientes fragmentos del archivo de configuración, los parámetros se encuentran codificados de la siguiente manera:

```
# Los comentarios van con '#' al inicio del renglón.

# > Parámetro = valor (entero o float, los flags (bool) van como int=0|1). Estos parámetros,
# determinados por '>' al comienzo del renglón, van como claves del diccionario dataDic{}.
# Los parámetros pueden estar en cualquier lugar y orden, ya que se leen antes que el resto
# de los datos.

# estos son flags:
> PublicacionDeBordes = 0
> PublicacionDePuntos = 0
> DebugEntidades = 1

# --------------------------- Geometría ---------------------------------
# Cantidad total de palas de la hélice a representar, independientemente de las palas
```

```
# generadas por geometría.
> Npalas = 3

# -- Definición de densidades globales paramétricas. Estos parámetros sí se pueden definir
# a gusto y en cantidad y reutilizarse en las definiciones de tamaños de elemento para bordes
# y/o caras.

> DensDetalle = 0.00012
> DensFina = 0.0025
> DensMedia = 0.025
> DensGruesa = 0.25

# Las claves a nombres de borde van con '/' al inicio del renglón. Son coherentes con la
# geometría generada:
# /nombre del borde, tamaño de elemento inicial, tamaño de elemento final   (en los edges)

/nervio,        0.05, 0.001
/bordeAtaque,   0.0075, 0.00012

# Las caras están definidas por '@' al inicio del renglón.
# @nombre de cara, tamaño de elemento (máxima longitud de algún lado).

@supCaraExtPala,    0.05,    DELAUNAY
@supCaraIntPala,    0.05,    DELAUNAY
@supTapitaPala,     0.01,    DELAUNAY

@supLateralCono,    0.05,    FRONTAL
@supEje,                DensMedia,  FRONTAL

@supEntrada,            DensGruesa, FRONTAL
```

El método `Procesar()`, perteneciente a la clase `Generador`, es el encargado de crear las instancias de los objetos que crean las geometrías involucradas. Este método ejecuta la cantidad de instancias de clase necesarias: `Cilindro`, `Cono`, `Pala`, `Eje` (todos elementos del tipo `SOLID` de Salome) y realiza las operaciones booleanas correspondientes para obtener el sólido a ser restado de la instancia `Cilindro`. Este sólido lleva por nombre "elContinuo". Es necesario destacar que, para el correcto mallado del continuo, es necesario identificar biunívocamente a los diferentes bordes que encierran las superficies que conforman cada cáscara que define a los sólidos.

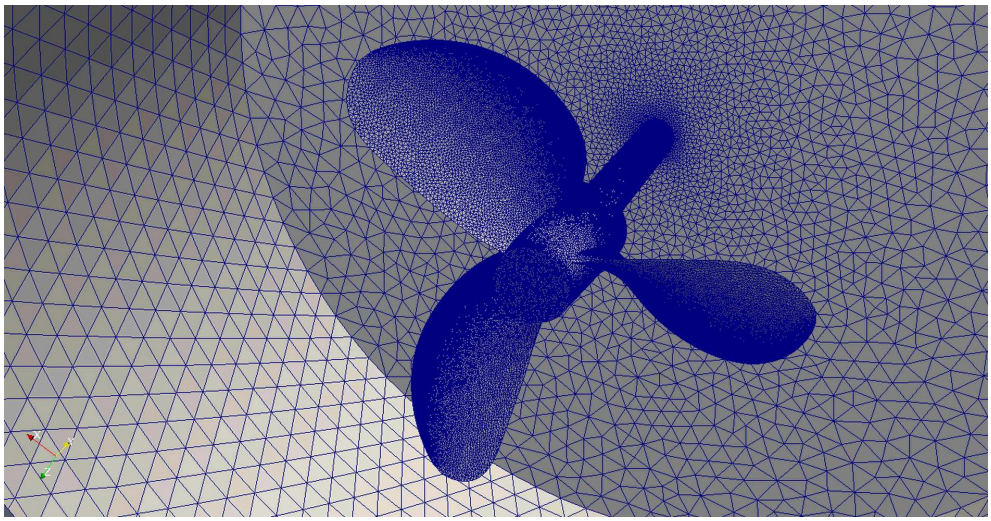

Figura 5: Mallado paramétrico de una hélice de tres palas, relación paso a diámetro = 1,2.

Para asignar tamaños de malla y algoritmos de mallado a los objetos, el programa Salome ofrece el método Explode() aplicado al objeto elContinuo, del cual se obtienen instan-

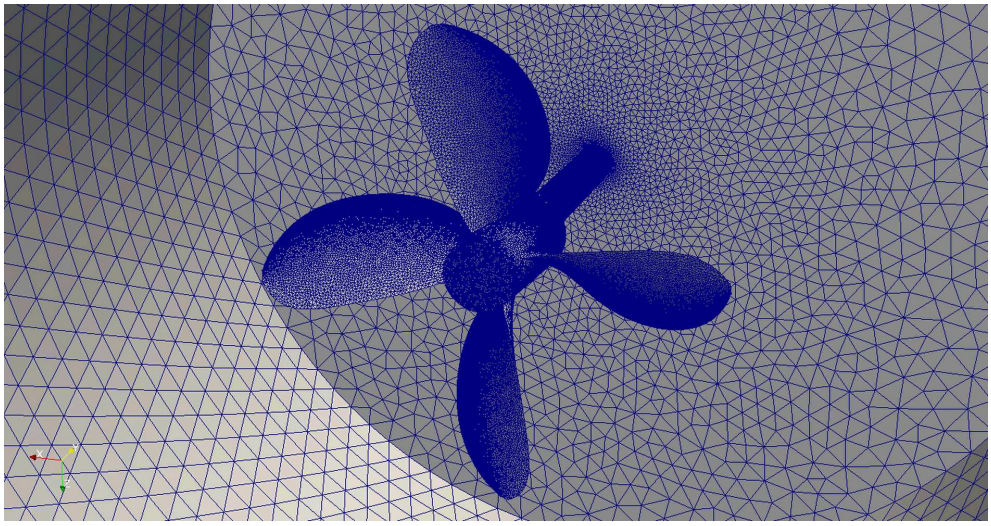

Figura 6: Mallado paramétrico de una hélice de cuatro palas, relación paso a diámetro = 1,2.

cias requeridas derivadas de las clases `EDGE`, `FACE`, y `SOLID` de Salome. Estas instancias, si bien son idénticas geométricamente, no son los mismos objetos que los que definieron inicialmente la geometría sino que son objetos nuevos. Es por ello que es necesario tener una manera de identificar las nuevas geometrías con las originales. Para ello, se generan puntos (llamados "handles") durante la creación de las geometrías para determinar su identidad y correspondencia con las mismas del grupo de objetos resultantes de la ejecución del método `elContinuo.Explode()`. Para ello, se crearon especialmente dos clases: `Segmento` y `Cara`. Las instancias de estas clases contienen, como atributos, tanto un vértice que hace las veces de punto de ubicación unívoca de la geometría en cuestión como también del tipo de algoritmo de mallado y sus parámetros correspondientes. De esta manera, se pueden aplicar de manera sencilla los algoritmos de mallado a todos los objetos de tipo `EDGE` y `FACE` derivados de la explosión del contínuo.

Es interesante señalar que, con pocas modificaciones, es posible utilizar este programa para la creación de geometrías y mallas para modelado numérico de otros procesos, como por ejemplo, la soldadura de punto por fricción–agitación. Esto se lograría seleccionando un número nulo de palas a generar y asignando un diámetro de eje mayor al diámetro de cono, por lo cual se genera un contínuo de geometría de revolución que contiene un hueco equivalente al de una herramienta para tal proceso de soldadura. Asimismo, de manera muy sencilla se podría adaptar el código para la generación automática de series de carenas de embarcaciones navales.

4. RESULTADOS

En las figuras 4 a 8 se pueden apreciar diferentes resultados de mallados paramétricos.

La figura 4 muestra el mallado de una hélice de tres palas y paso 1,2. En la figura 6 se detalla el mallado de una hélice del mismo paso pero incrementando a cuatro el número de palas. En la figura 7 se observa, para el caso de tres palas, el continuo correspondiente a un paso de 1,2 en superposición con el mallado para una hélice de paso 1,9.

El programa asimismo genera grupos de elementos de superficie y volumen paramétricamente según lo configurado a través del archivo de entrada. Estos grupos se identifican en el archivo de salida tipo .unv, de formato abierto. El investigador puede, de esta manera, reconocer biunívocamente los grupos para realizar los cálculos que requiera en su modelado.

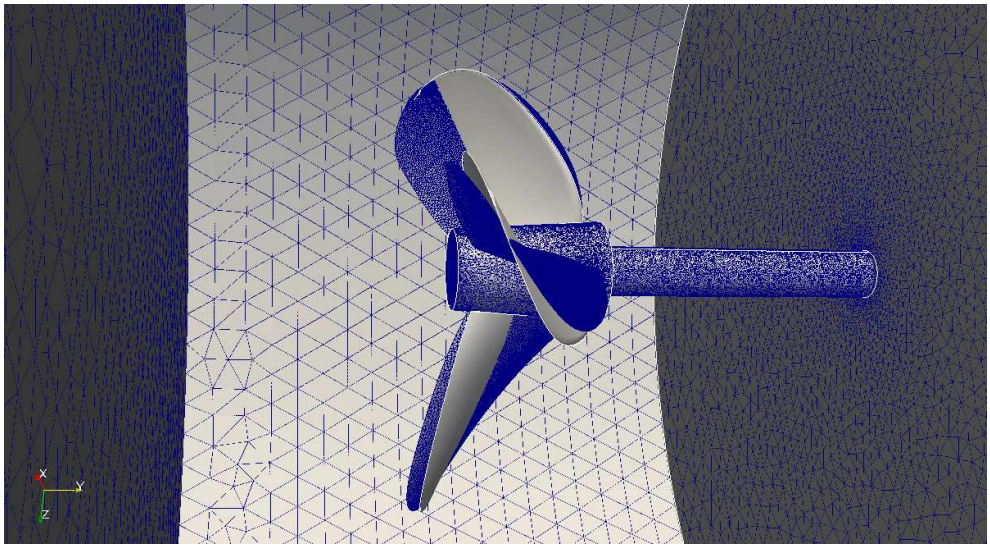

Figura 7: Superposición del mallado paramétrico y de la geometría de una hélice de tres palas con dos valores diferentes de relación paso a diámetro: 1,2 y 1,9.

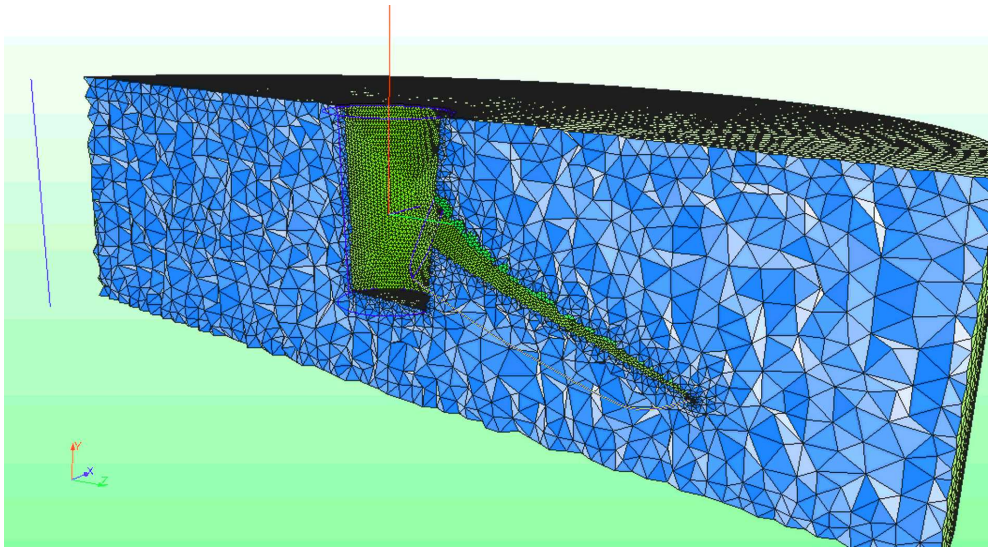

Figura 8: Corte de mallado tridimensional de una hélice de una sola pala, sin eje, para examen de densidad de elementos en el continuo.

5. CONCLUSIONES

Con los ejemplos de geometrías y mallados provistos en la sección de resultados, se muestra la capacidad de generar parametricamente mallas válidas para análisis de elementos finitos en todos los casos mostrados, que incluyeron cambios en el número de palas y otros parámetros significativos en el diseño de las mismas. Los tamaños mínimos y máximos para los diversos parámetros y tamaños de elementos deben analizarse para cada caso en particular y en función de la capacidad de cómputo del sistema informático sobre el cual se implemente.

La utilización de la plataforma Salome mediante su interfaz de programación en lenguaje Python sobre un sistema Debian GNU+Linux ha demostrado una potencialidad y versatilidad muy aceptable. El empleo de software libre, además evita la onerosa circunstancia de depender de costosas licencias de permisos de uso de otros software privativos con código cerrado, obscuro al usuario, mientras que sigue brindando la posibilidad de acceder a herramientas de buena

calidad y con la posibilidad de aprender de la escritura de su código y la potencial expansión de las capacidades actuales.

REFERENCIAS

Bernitsas. M.M., Ray. D., y Kinley P. Kt, kq and efficiency curves for the wageningen b-series propellers. I, 1987.

Braid I.C., Hillyard R.C., y Stroud I.A. Stepwise construction of polyhedra in geometric modelling. *Mathematical Methods in Computer Graphics and Design*, I, 1978.

Carlton J.S. Marine propellers and propulsion. second edition. 2007.

Harries S. Practical shape optimization using cfd. 2014.

Kuiper G. The wageningen propeller series. I, 1992.

Miao A. y Wan D. Ship hull form design using a kriging-based global optimization algorithm. 2017.

Oosterveld. D. Wake adapted ducted propellers. publication no. 345. 1970.

Open Cascade Company. OpenCascade Technology . 2017.

Salome. The Open Source Integration Platform for Numerical Simulation. Version 7.7.1. OPEN CASCADE SAS, 1 place des frères Montgolfier, France. url: https://www.salome-platform.org. 2015.

SPI and others. Debian Operative System. http://www.debian.org. *http://www.debian.org*, 1997-2017.

Urquiza S.A. y Venere M.J. An application framework architecture for FEM and other related solvers. página 11, 2001.

Van Rossum, Guido. Python tutorial. *Technical Report CS-R9526*, 1995.