UNIVERSIDAD NACIONAL DE INGENIERÍA
FACULTAD DE INGENIERÍA MECÁNICA
SECCIÓN DE POSGRADO

ANÁLISIS DEL DIMENSIONAMIENTO DE UNA TURBINA HIDRÁULICA, POR REDES NEURONALES PARA UNA MINICENTRAL

TESIS
PARA OPTAR EL GRADO DE:

MAESTRO EN CIENCIAS, MENCIÓN INGENIERÍA MECÁNICA, DISEÑO DE MÁQUINAS

VÍCTOR BEDER VIDAL BARRENA

PROMOCIÓN 2000 - II

LIMA – PERÚ

2010

** * DEDICATORIA * **

A la memoria de mi querido padre: Don Arístides Vidal Márquez
Manténganse unidos, busquen la felicidad y armonía, pues siendo ustedes felices, yo descansaré en paz.

A mi madre: Graciela Barrena de Vidal
Tus brazos siempre se abren cuando necesito un abrazo. Tu corazón sabe comprender cuándo necesito una amiga. Tus ojos sensibles se endurecen cuando necesito una lección. Tu fuerza y tu amor me han dirigido por la vida y me han dado las alas que necesitaba para volar.

AGRADECIMIENTOS.

Agradezco a mis asesores del proyecto, a los doctores ***César Cebreros Delgado de la Flor*** *y* ***Nicolás Kemper Valverde****, directores del proyecto; por su confianza y consejos en el desarrollo de todo el trabajo realizado.*

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

Lista de Símbolos

a_o	Ancho entre álabes
a	Espesor del chorro en la turbina Michell
b	Ancho de la cuchara
b_o	Ancho de la corona directriz
B	Ancho del rodete en la turbina Michell
C_1	Velocidad absoluta en la boquilla
C3	Velocidad de salida en la turbina Francis
1Ch	Una boquilla o un chorro
2Ch	Dos boquillas o dos chorros
4Ch	Cuatro boquillas o cuatro chorros
DGER	Dirección General de Electrificación rural
d	Diámetro del chorro de la boquilla
D	Diámetro del rodete
De	Diámetro exterior de la rueda
D3	Diámetro exterior en el tubo de aspiración de una turbina Francis
F	Fuerza
GWh	Giga vatios por hora
GRNN	General Regression Neural Network
G	Gravedad
H	Altura
h	Altura de la cuchara
KW	Kilo vatios
Logsig	Función de transferencia sigmoidal
LMS	Algoritmo Least Mean Square

MW	Mega vatios
MCH	Mini Central Hidráulica
MEIM	Ministerio de Energía y Minas
MERNA	Método de las redes neuronales artificiales
Ns	Número específico de revoluciones
N	Velocidad de rotación, rpm
P	Potencia
Q	Caudal
RNA	Redes Neuronales Artificiales
RPM	revoluciones por minuto
SEIN	Sistema Eléctrico interconectado
t	Espesor de la cuchara
te	paso o división exterior
u	Velocidad tangencial
u_1	Velocidad Tangencial
Z	Número de cucharas
Zo	Número de alabes

LETRA GRIEGA

η	Eficiencia
φ	Coeficiente
β	Ángulo

RESUMEN

Este trabajo desarrolla un algoritmo computacional, para la realización de la selección de diferentes turbinas hidráulicas, haciendo uso de las técnicas de las numerosas experiencias de práctica experimental y de laboratorio de diferentes expertos, en materias tan específicas como en la selección y diseño de estas turbinas hidráulicas de acción y de reacción, complementado con una red neuronal artificial y ecuaciones de cálculo tabuladas en cuadros estadísticos.

Dicha selección la realiza el Sistema Inteligente que usa procedimientos de conocimiento basada en los cálculos realizados por los expertos. Este campo requiere utilizar un conjunto de conocimientos basados en la experiencia acumulada durante muchos años de trabajo en esa área específica, además de emplear conocimientos y rutinas de trabajo basados en otras disciplinas.

Los conceptos de los parámetros diferentes de las ecuaciones de selección y diseño, tabuladas en cuadros estadísticos, las cuales serán aprovechados por la red neuronal artificial, previamente entrenada para que tenga capacidad de razonar, en el sentido de inferir nueva información, y que por la dificultad del problema de la selección de las turbinas requiera una solución con un grado de inteligencia. Esta cualidad de inteligencia implica una serie de elementos distintivos, tales como la capacidad de aprendizaje, de auto corrección y razonamiento.

Hace veinte años Mark Weiser acuñó el término computación ubicua, y con ello, surgió un nuevo concepto en las relaciones entre el hombre y las computadoras. Los avances tecnológicos actuales hacen posible, que lo que entonces no era más que una visión futurista de la computación, se convierta en una revolución no sólo tecnológica sino también social y cultural.

Palabras Claves: Turbinas Hidráulicas, Experto, Red Neuronal artificial, neuronas, dendritas, jaba, red backpropagation, entrenamiento.

ABSTRACT

This work develops a computational algorithm for the completion of the selection of different hydraulic turbines, using the techniques of the many experiences of experimental and laboratory practice of different experts in specific areas such as the selection and design of these turbines hydraulic action and reaction, supplemented with an artificial neural network and calculation equations tabulated in statistical tables.

This selection is performed by using intelligent knowledge procedures based on the calculations made by experts. This field requires using a set of knowledge based on experience accumulated over many years working in that specific area, in addition to using knowledge and work routines based on other disciplines.

The concepts of the different parameters of the selection and design equations, tabulated in statistical tables, which will be used by the artificial neural network, previously trained to have ability to reason, in the sense of infer new information and that the difficulty of the problem of selection of the turbines required a solution with a degree of intelligence. This quality of intelligence involves a number of distinctive elements, such as learning ability, self-correcting and reasoning.

Twenty years ago Mark Weiser coined the term ubiquitous computing, and with it came a new concept in the relationship between humans and computers. Current technological advances make it possible, then it was just a futuristic vision of computing, it becomes a revolution not only technological but also social and cultural.

Keywords: Hydraulic Turbines, Expert, Artificial neural network, neurons, dendrites, bag, backpropagation network, training.

INTRODUCCIÓN

Desde la antigüedad, el hombre ha buscado medios para que su labor cotidiana sea más cómoda, para ello ha ido desarrollando técnicas e instrumentos que le liberen de los trabajos menos gratos; sin embargo la gran mayoría de ellos precisan que el ser humano supervise e intervenga en su operación.

Puede decirse que el cerebro humano es un elemento de procesamiento de la información extremadamente complejo, cuyo modo de funcionamiento es eminentemente paralelo y cuyo comportamiento no puede describirse por medio de modelos sencillos como son los lineales. De ahí que surgiera, dentro de la Inteligencia Artificial, una rama que intenta imitar el comportamiento del cerebro: los sistemas conexionistas o redes neuronales artificiales.

Los sistemas expertos constituyen una de las principales aplicaciones de la inteligencia artificial, término que fue utilizado por primera vez en 1956, cuando McCarthy, Minsky, Newel y otros estudiosos definieron sus principales características: "La Inteligencia Artificial es el conjunto de técnicas que se aplican en el diseño de programas para computador que tengan capacidad de razonar, en el sentido de inferir nueva información, y que por la dificultad del problema requieren una solución con un grado de inteligencia".

La clasificación como: **grandes centrales** producen energía eléctrica suficiente para abastecer a grandes ciudades ya redes extensas, por ejemplo la central Antúnez de Mayolo que suministra energía a la red nacional por encima de los 10 MW de potencia, **pequeña central** se refiere a unidades con una capacidad de menos de 10 MW, la ***mini central*** se refiere a unidades con una capacidad de 101 a 2000 KW, mientras que la ***micro central*** se refiere a una capacidad del sistema por debajo de los 100 KW.

Este trabajo de investigación para una mejor compresión y con fines netamente didácticos, se ha dividido en varios capítulos, que se detallan a continuación:

En el primer capítulo de este trabajo se trata sobre el estado del arte de las turbinas hidráulicas y de las redes neuronales artificiales, el funcionamiento de una neurona biológica, características de una red neuronal artificial y principales tipos de redes neuronales, como el perceptrón, adaline y las redes Backpropagation.

En el segundo capítulo, se expone en forma detallada la información teórica sobre el cálculo de las turbinas hidráulicas. Actualmente es necesario desarrollar la construcción de turbinas para el aprovechamiento de pequeños saltos hidráulicos, por lo cual se requiere que las mismas sean capaces de transformar eficientemente la energía cinética del agua en energía en forma de electricidad o energía mecánica en el eje.

En el tercer capítulo, se utilizan los métodos: tradicional y el proceso seguido con la aplicación de las Redes Neuronales, basadas en la inteligencia artificial, las RNA han sido aplicadas principalmente como herramienta para la predicción y la clasificación de patrones. Las RNA son capaces de trabajar de forma no lineal con el análisis de grandes masas de datos sujetas a imprecisiones, con suficientes ejemplos reales y para las que no existen reglas generales y rápidas que

puedan ser fácilmente aplicadas y programadas como las que utilizaríamos en un sistema experto.

Luego se exhibe los resultados del proyecto y el análisis comparativo entre los métodos clásicos y el de las redes neuronales. En definitiva se dan las conclusiones y recomendaciones a las que se ha llegado después de la elaboración del proyecto.

Durante la última década, las RNA han atraído la atención de multitud de investigadores y han sido aplicadas con éxito en diferentes ámbitos del conocimiento tales como ingeniería, física, estadística o economía. En medicina, casi todos los sistemas son no lineales, por lo que se considera que en ella los modelos lineales presentan debilidades que podrían ser superadas por sistemas que no impongan a priori esta restricción.

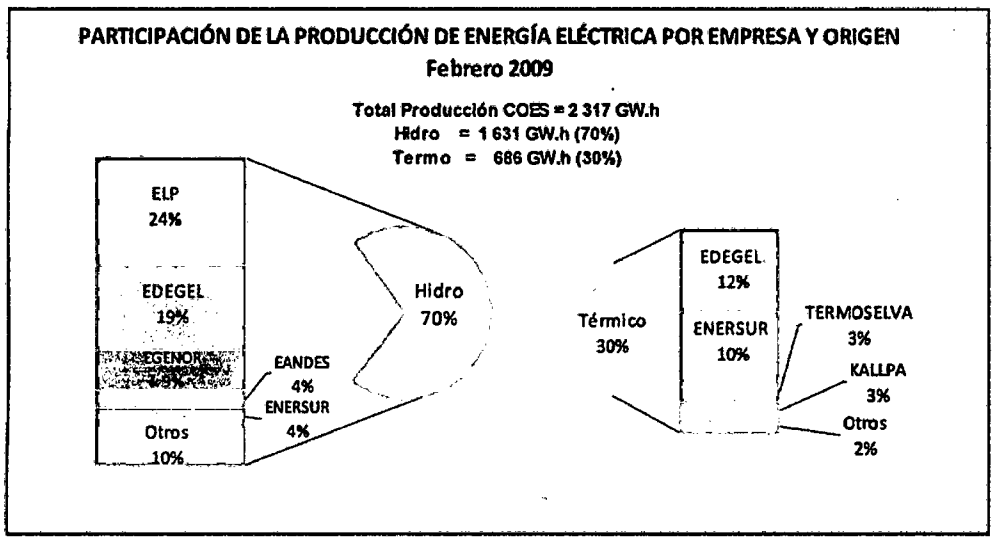
PLANTEAMIENTO DEL PROBLEMA.

¿Como resuelvo el dimensionamiento de las turbinas hidráulicas utilizando las redes neuronales?

¿Como utilizo el lenguaje C++ generado por el NeuroShell 2 en la solución de este problema?

Actualmente la generación de energía eléctrica es tal vez una de las principales fuentes de desarrollo y de mejoramiento de la calidad de vida del hombre actual, ya que gracias a ella, hoy en día es posible llevar a cabo un sin número de actividades que contribuyen al crecimiento integral de la sociedad, tanto desde el punto de vista científico y tecnológico, como industrial, cultural y económico. Por esta razón, la energía eléctrica se ha convertido en uno de los servicios sociales de mayor demanda e importancia en nuestro medio, haciéndose cada vez más indispensable para la ejecución de actividades de gran trascendencia.

Según el Ministerio de Energía y Minas, el Perú requiere de una inversión total en generación y transmisión de unos **3,065** millones de dólares, de los cuales **1,981 millones** corresponden a **generación** y **1,084** a **transmisión**, para atender la demanda de energía eléctrica al 2015. La demanda del Sistema Eléctrico Interconectado Nacional (SEIN) durante el período 2006 - 2015 crecerá a una tasa promedio del 7.3 por ciento anual, por lo que se requiere instalar 3,605 Megavatios (MW), de los cuales 2,540 corresponden a centrales termoeléctricas a gas natural y **1,065 a centrales hidroeléctricas**.



FUENTE: MEM

Fig. I.1 Comparación de la demanda de energía hidráulica – térmica.

El Ministerio de Energía y Minas (MEM), a través de la Dirección General de Electrificación Rural (DGER-MEM), tiene la competencia en materia de **electrificación rural de acuerdo a la Ley N° 28749**, ***"Ley General de Electrificación Rural"***, en la ampliación de la frontera eléctrica en el ámbito nacional, en coordinación con los Gobiernos Regionales y Locales, y entidades públicas y privadas dedicadas a estos fines, permitiendo el acceso del suministro de electricidad a los pueblos del interior del país.

FORMULACIÓN DEL PROBLEMA.

Para resolver el dimensionamiento y selección, ante la gran variedad de turbinas hidráulicas, para un salto y un caudal determinado, ***se ha desarrollado***

un método que permita la selección y dimensionamiento de las dimensiones principales de una turbina hidráulica, sin necesidad de tener a un experto en esta materia; utilizando las redes neuronales. Para dimensionar y seleccionar una turbina hidráulica, se requiere clasificar las diferentes turbinas por el salto, caudal, su velocidad específica, la potencia y sus eficiencias; esta clasificación se muestra en la Tabla 1.1.

Actualmente *no se conoce con* ***información publicada para calcular y determinar cuál es el tipo de turbina para un caudal y altura, más conveniente para el funcionamiento de una mini central hidroeléctrica***, empleando las redes neuronales.

El Software el NeuroShell 2 es un programa para Windows que se utiliza para resolver una amplia variedad de problemas. NeuroShell 2 es una herramienta para la creación de sistemas RNA, que permite predecir en la selección y dimensiones de una turbina. Esta aplicación combina una interfaz de diseño modular con avanzados procedimientos de aprendizaje, está diseñado y entrenado con los datos que figuran en una hoja de cálculo (por ejemplo, Excel o 1-2-3) e incorporada en la hoja de balance (*dentro de la red de desarrollo NeuroShell*).

NeuroShell 2 genera/compila automáticamente una DLL, que podrá ser utilizada en cualquier entorno de programación que soporte el acceso a librerías dinámicas, como por ejemplo: Visual Basic, Access, Visual C++ (Jaba). Los métodos que se ha utilizado para comparar las soluciones son las siguientes:

1. Método analítico o tradicional,
2. Método de las redes neuronales, que llamaremos MERNA.

OBJETO DEL ESTUDIO.

Es desarrollar un software que permita a los usuarios llevar a cabo varias tareas específicas, tal que al ingresar la ***altura y el caudal***, permita seleccionar y calcular las dimensiones principales de las diferentes turbinas hidráulicas para una mini central hidráulica.

OBJETIVO.

El objetivo de la presente investigación, es desarrollar una metodología para mejorar la selección y el cálculo de las dimensiones principales de las turbinas hidráulicas para una mini central, ingresando la altura y el caudal mediante dos métodos: el método analítico o tradicional y el método de las redes neuronales, que en adelante denominaremos MERNA.

ALCANCES.

El desarrollo del presente trabajo comprende:

- Determinación de la potencia y el número específico que se producen al utilizar la altura de la caída de agua y su caudal, mediante los métodos analítico o tradicional y el MERNA.

- Determinación de las dimensiones principales de una turbina hidráulica, mediante los métodos analítico o tradicional y el MERNA.

- Se realizaran predicciones para caudales que varían desde los 0.03 m^3/seg hasta 20 m^3/seg y alturas que varían desde los 3 metros hasta los 300 metros, para seleccionar y calcular las dimensiones principales de las diferentes turbinas hidráulicas que se encuentran en estos rangos.

JUSTIFICACIÓN.

La importancia de este proyecto se basa en lo siguiente:

1. En el Perú existen muchos poblados y se estima que hay un 40% de población rural que no tienen beneficios energéticos. Están lejos de una red de distribución de energía nacional; por requerir estas de una coyuntura especial del erario nacional (presupuesto).

2. En la actualidad en el Perú no se tiene la experiencia necesaria para construir estas turbinas hidráulicas, que si se efectúa en otros países, especialmente los desarrollados que emplean su propia tecnología que no se conocen; en nuestro país lo que se realiza en las fundiciones es copiar un modelo existente sin tener cuenta el salto, caudal, potencia y número específico.

3. Las energías renovables constituyen para nuestro país, nuevas alternativas para suministrar energía a regiones aisladas.

4. Un sistema inteligente es capaz de asesorar al diseñador inexperto en el análisis y selección de alternativas óptimas y en el diseño automatizado, lo que constituye una potente herramienta que puede contribuir a nuestra propia tecnología.

LIMITACIONES.

LIMITACIÓN TEMPORAL.

El proyecto de investigación realizada utiliza el NeuroShell2 que genera una función en C++, con el programa fuente generada se ha desarrollado una aplicación en el lenguaje Java para su funcionamiento, como resultado práctico de la presente investigación, el sistema realizado selecciona y calcula las dimensiones principales de las turbinas hidráulicas de una mini central hidráulica.

LIMITACIÓN ESPACIAL.

La investigación se ha realizado en el laboratorio CIM de la Facultad de Ingeniería de la Universidad Ricardo Palma; en este laboratorio está instalado el NeuroShell2 (donado por Nicolás Kemper) que genera un código fuente en: Visual Basic, Access, Visual C++ (Java); y en los laboratorios de la Escuela de Ingeniería Informática, en donde se ha desarrollado el software en el lenguaje Java, para

la predicción en la selección y cálculo de las dimensiones principales de las diferentes turbinas hidráulicas.

LIMITACIÓN CONCEPTUAL.

En cuanto a la dirección de la investigación realizada, será teórico y experimental utilizando, las herramientas del NeuroShell2 que recomienda: **Probabilístico** (probabilistic) y **regresión general neta** (general regresión net) o múltiples placas ocultas. La red Backpropagation genera un archivo en C que ocupa 180kb, la red GRNN genera un archivo en **C** que ocupa 1.9 MB, este archivo no es posible compilarlos directamente; por tal motivo he seleccionado la red Backpropagation.

HIPÓTESIS.

Es posible generar una función utilizando datos reales mediante el uso NeuroShell2 que permite predecir, la selección y el cálculo de las dimensiones principales de una turbina hidráulica; empleando las variables de entrada, como la altura y el caudal.

METODO DE INVESTIGACIÓN.

La investigación realizada es un proceso que, mediante la aplicación del método científico, procura obtener información relevante y fidedigna, para entender, verificar, corregir o aplicar el conocimiento; para hallar resultados de manera clara, en la selección y el cálculo de las dimensiones principales de las diferentes turbinas hidráulicas, aplicando el método de una red neuronal artificial.

Método Analítico.

Se utiliza las ecuaciones de forma y dimensiones de Bovet, F. de Siervo y A. Lugaresi en máquinas hidráulicas.

Método de las Redes Neuronales.

El método que pretende en esta tesis es el diseño de una estrategia de una red neuronal, que permita la ejecución de acciones que optimicen realizando buenas predicciones en la selección y diseño de turbinas hidráulicas. En este trabajo, se utilizó la red neuronal backpropagation para predecir la variable turbinas (selecciones y dimensiones del rodete); en donde existe una capa de entrada con "x" neuronas y una capa de salida con "y" neuronas y al menos una capa oculta de "z" neuronas internas.

Método experimental.

Utilizar un software comercial, tal como el Java para diseñar un software que utilice la función generada en el NeuroShell2, de tal manera que al utilizar los parámetros de entrada como la altura y el caudal, seleccione y calcule la turbina que corresponde a esas condiciones.

Procedimiento.

El procedimiento realizado es el siguiente:

- Se calcula la potencia y su velocidad específica, que son las características que diferencian a las turbinas de acción y de reacción.
- Se calcula las dimensiones principales de las diferentes turbinas por el método tradicional.
- Con los datos obtenidos en el método tradicional, aplicamos el método de las redes neuronales artificiales (MERNA) en donde se generamos una función.
- Evaluar el análisis comparativo de la selección y cálculo de las dimensiones principales de una turbina.

TIPO DE INVESTIGACIÓN.

El proyecto, se ubica como un nuevo paradigma de una investigación académica y tecnológica en ciencias de ingeniería, al desarrollar una herramienta software con una tecnología relativamente nueva del MERNA en la selección y cálculo de turbinas que sustituya al método tradicional y que sea igualmente fiable.

La innovación en el trabajo realizado con redes neuronales artificiales que tienen la habilidad de predecir y abordar problemas de una elevada complejidad entre variables, problemas que, mediante métodos tradicionales no pueden ser resueltos.

NIVEL DE INVESTIGACIÓN.

Por la abundancia de conocimientos explayados, se trata de una investigación con la finalidad de obtener información nueva orientada hacia el logro de nuevos conocimientos de manera metodológica, sistemática y comprobable; que contribuyan a resolver problemas y que permite seguir desarrollando y profundizando conocimiento en investigaciones posteriores.

NOVEDAD CIENTÍFICA.

Las redes neuronales operan sobre la base de reconocimiento de patrones, y que pueden adquirir, almacenar y utilizar conocimiento experimental, obtenido a partir de ejemplos. Esta forma de adquirir el conocimiento es una de sus características más destacables: no se programa de forma directa, como en los sistemas expertos, sino que se adquiere a partir de ejemplos, por ajuste de parámetros de las neuronas mediante un algoritmo de aprendizaje. Por esta razón, se desarrolla una nueva metodología que este acorde con los avances tecnológicos, como desarrollar un software de programación especializado, con el propósito que al aplicar las redes neuronales artificiales simule la forma de razonar del cerebro humano.

CAPÍTULO I

ESTADO DEL ARTE

1.1 REVISIÓN BIBLIOGRÁFICA.

Juan Cevallos Ampuero [33], en su artículo: "Aplicación de Redes Neuronales para Optimizar Problemas Multi respuesta en mejora de la Calidad", trata sobre la aplicación de redes neuronales en problemas Multi respuesta para el mejoramiento de la calidad, a partir de un análisis conceptual se establece una aplicación que demuestra la validez de las redes neuronales utilizando la ***red backpropagation*** con entrenamiento.

Gustavo Ovando, Mónica Bocco y **Silvina Sayago [34]**, en su artículo: "Redes Neuronales para Modelar Predicción de Heladas" En este trabajo se desarrollaron modelos basados en redes neuronales del tipo ***"backpropagation"***, para predecir la ocurrencia de heladas, a partir de datos meteorológicos de temperatura, humedad relativa, nubosidad, dirección y velocidad del viento. El entrenamiento y la validación de las redes se realizaron utilizando 24 años de datos meteorológicos correspondientes a la estación de Río Cuarto, Córdoba, Argenti-

na, separados en 10 años como conjunto de datos de entrenamiento y 14 como conjunto de datos de validación. Se construyeron diferentes modelos para evaluar el comportamiento de las redes cuando se usan distintos números de variables de entrada y/o neuronas en la capa oculta y las probabilidades de aciertos en los resultados de predicción para los mismos, al considerar distintas variables de entrada. En los análisis realizados por Ceballos **[33]** y Ovando **[34]** demuestran la validez de la utilización de las redes neuronales, utilizando la ***red backpropagation*** con entrenamiento que genera un archivo en **C** que ocupa 180kb, que son factibles de ser generados.

Patricia González Serrano [35], en su artículo "Simulación Técnico-Económica Del Mercado Eléctrico Español" El proyecto tiene como objeto el estudio de la evolución del mercado eléctrico español a partir del comportamiento hidráulico del parque de generación. Partiendo de las ofertas presentadas al Pool durante los años 2002 y 2003 de las distintas centrales de generación hidráulica en régimen ordinario, proporcionadas por la Compañía Operadora del Mercado Español de Electricidad, y con la información obtenida del Ministerio del Medio Ambiente sobre la pluviometría, reserva y energía disponible de las distintas cuencas hidrográficas españolas durante esos mismos años, se ha analizado la forma de ofertar de las distintas unidades hidráulicas en el Mercado Eléctrico Español mediante la aplicación de Redes Neuronales. Una vez que las Redes Neuronales han aprendido a ofertar con las condiciones hidráulicas reales del 2003, se ha analizado la variación del precio de la energía en el Mercado Diario simulando otras condiciones hidráulicas del año 2003.

Hernández López Leonor, [36] en su artículo "Desarrollo de una metodología para la predicción y optimización de emisiones contaminantes y consumo en motores Diesel de automoción mediante redes neuronales artificiales" Las estrictas normativas europeas en cuanto a límite de emisiones de escape permitidas, junto con el requerimiento de reducción del consumo de combustible impuesto por el mercado, someten a los motores Diesel de automoción a unos altos niveles de exigencia. El desarrollo tecnológico llevado a cabo como respuesta a estas demandas en el campo de los motores Diesel, ha supuesto un aumento notable en la complejidad de estos motores, incrementando de forma importante el número de parámetros operativos de motor. Este hecho complica la predicción de emisiones, a la vez que impone a la optimización en motor unas dimensiones elevadas. Los trabajos realizados en la presente tesis doctoral se plantearon con el objetivo de desarrollar herramientas que permitieran la predicción emisiones con un tiempo de cálculo corto y que proporcionaran resultados de buena calidad. Las redes neuronales artificiales (RNA) presentan ciertas características ventajosas a la hora de abordar el problema planteado, como son el modelado Multi variable, la gestión de comportamientos no lineales y la rapidez de cálculo, con lo que el estudio se centró en estos modelos empíricos.

Jorge Enrique Rodríguez Rodríguez, [37] en su artículo "Redes Neuronales Artificiales para la clasificación de Imágenes Satelitales" En este artículo se presenta el análisis hecho a un conjunto de datos que representan diferentes imágenes, clasificadas como: Tierra roja, Cosecha de algodón, Tierra gris, Tierra gris húmeda, Tierra con vegetación, Cada terreno gris húmedo. El artículo se es-

tructura en: una introducción en la cual se destaca la importancia del modelo de los mapas auto-organizativos de Kohonen (SOM) y la red de resonancia adaptativa (ART2) para la clasificación de imágenes; descripción de los algoritmos utilizados por las dos redes neuronales artificiales en mención; información relevante al problema; uso de las redes SOM y ART2 en la clasificación de imágenes satelitales; y planteamiento de conclusiones y trabajos futuros.

Ismael González García, [38] en su artículo "Control Neuronal De Un Generador De Inducción Para Generación Eólica" Dado el impacto para la mejora ecológica que representa el empleo del viento como fuente de energía, surge la necesidad de hacer económicamente más atractiva esta opción energética para los gobiernos e industriales de todo el mundo. De aquí que surge la necesidad de aumentar la eficiencia de los sistemas de conversión de energía. Una forma de aumentar la eficiencia en esto sistemas, es mediante el desarrollo de mejores técnicas de control, como lo han demostrado ser las técnicas inteligentes basadas en redes neuronales. En este trabajo de tesis se presenta la elaboración de un control neuronal del tipo perceptrón multicapa, aplicando la técnica de adecuación potencia, para controlar un generador de inducción empleado en plantas Eolo eléctricas a través de la variación del índice de modulación de inversor electrónico, perteneciente al sistema rectificador inversor (REC-INV).

Martínez Estudillo, F. J. y Hervás Martínez, C. [39] en su artículo "Modelo no lineal basado en redes neuronales de unidades producto para clasificación. Una aplicación a la determinación del riesgo en tarjetas de crédito". El principal objetivo de este trabajo es mostrar un tipo de redes neuronales denomina-

das *redes neuronales basadas en unidades producto* (RNUP) como un modelo no lineal que puede ser utilizado para la resolución de problemas de clasificación en aprendizaje. Proponemos un método evolutivo en el que simultáneamente se diseña la estructura de la red y se calculan los correspondientes pesos.

PATRICIA GONZÁLEZ SERRANO [40] en su artículo "Departamento De Sistemas Energéticos, Simulación Técnico- Económica Del Mercado Eléctrico Español" El proyecto tiene como objeto el estudio de la evolución del mercado eléctrico español a partir del comportamiento hidráulico del parque de generación. Partiendo de las ofertas presentadas al Pool durante los años 2002 y 2003 de las distintas centrales de generación hidráulica en régimen ordinario, proporcionadas por la Compañía Operadora del Mercado Español de Electricidad, y con la información obtenida del Ministerio del Medio Ambiente sobre la pluviometría, reserva y energía disponible de las distintas cuencas hidrográficas españolas durante esos mismos años, se ha analizado la forma de ofertar de las distintas unidades hidráulicas en el Mercado Eléctrico Español mediante la aplicación de *Redes Neuronales*. Una vez que las *Redes Neuronales han aprendido* a ofertar con las condiciones hidráulicas reales del 2003, se ha analizado la variación del precio de la energía en el Mercado diario simulando otras condiciones hidráulicas del año 2003. De los estudios realizados, se concluye que las ofertas presentadas al Pool por las centrales de generación hidráulica en régimen ordinario, y por tanto la mayor o menor disponibilidad de estas centrales para producir energía, influyen en gran medida en el precio de la energía eléctrica casada en el Mercado Diario.

ANTONIO MUÑOZ SAN ROQUE [41] en su artículo "Aplicación De Técnicas De Redes Neuronales Artificiales Al Diagnóstico De Procesos Industriales" La tesis doctoral que aquí se presenta se enmarca dentro de las áreas de trabajo de diagnóstico y mantenimiento de procesos industriales, y propone un nuevo sistema de detección de anomalías incipientes basado en el modelado conexionista del funcionamiento normal de los componentes. El sistema propuesto está especialmente dirigido a resolver el problema de la detección de anomalías en aquellos casos en los que no existe una completa base de datos de fallo, y en los que el modelado físico del comportamiento de los componentes resulta inviable. La solución propuesta consiste en caracterizar el comportamiento normal de los componentes involucrados mediante la aplicación de técnicas de modelado de procesos dinámicos no lineales con aproximadores funcionales. Como aproximadores funcionales se propone utilizar *Redes Neuronales Artificiales supervisadas, tales como el Perceptrón Multicapa y la red PRBFN* (aportación original de esta tesis). Estas herramientas, además de ofrecer una elevada capacidad de representación, poseen una estructura modular que las hacen altamente paralelizables y realizables en *"hardware"*.

L. Quantz [31] en su publicación ***"Motores Hidráulicos"*** Elementos para el estudio, construcción y cálculo de las instalaciones modernas de fuerza hidráulica. Editorial Gustavo Gili, SA. Se desarrollan las teorías bidimensionales para los rodetes axiales y centrífugos, indicando los efectos tridimensionales que pueden producirse. Se afronta el estudio de la semejanza dinámica en las turbo máquinas a partir de leyes semi empíricas. Se amplía el estudio de la cavitación.

Se obtienen coeficientes asociados con el diseño de las máquinas. Nociones sobre el diseño y comportamiento de las turbo máquinas. Se presta especial atención a regímenes de funcionamiento no deseados por su comportamiento inestable o periódico. Se analiza el proceso de arranque deseable así como los dispositivos que permiten un equilibrado axial y radial aceptable para la máquina. Se estudian los ventiladores. En la Tabla 1.1 (**L. Quantz [31]**) se muestra la clasificación de las diferentes turbinas hidráulicas.

Tabla 1.1 Clasificación de Turbinas Hidráulicas.

TIPO DE TURBINA	n_s	n_q	$H_{máx}$ adm.
Turbina Pelton 1 CH	10 a 13	3 a 4	**1800 a 1300 m**
Turbina Pelton 2 CH	12 a 20	4 a 6	**1300 a 550 m**
Turbina Pelton 4 CH	20 a 30	6 a 9	**550 a 300 m**
Francis lenta	60 a 125	18 a 38	**350 a 150 m**
Francis normal	125 a 175	38 a 53	**150 a 120 m**
	175 a 225	53 a 68	**120 a 80 m**
Francis rápida	225 a 350	68 a 105	**80 a 35 m**
	350 a 450	105 a 135	**35 a 20 m**
Kaplan	300 a 600	105 a 180	**35 a 18 m**
Tubular	300 a 800	180 a 240	**18 a 12 m**
Hélice	500 a 1000	240 a 300	**12 a 5 m**

ITDG Perú **[32].** "Manual de Mini y Micro centrales Hidráulicas" Una guía para el desarrollo de proyectos. Intermediate Technology Development Group, ITDG-PERÜ. Este manual proporciona información sobre el diseño de sistemas de energía hidráulica en pequeña escala. Estos sistemas se clasifican, por lo general, en tres rangos de potencia: en gran escala, mini y micro generación. En la

tabla 1.2 (**ITDG** Perú **[32]**) se muestra las características principales de turbinas hidráulicas.

Tabla 1.2 Características principales de turbinas hidráulicas

TURBINA		Inventor y año de patente	Ns rpm	Q m^3/s	H m	P kW	ηmax %
ACCIÓN	PELTON	Lester Pelton (EE.UU) 1880	1 Ch: 30 2 Ch: 30-50 4 Ch: 30-50 6 Ch: 50-70	0.05-50	30-1800	2-300000	91
	TURGO	Eric Crewdson (G. Bretaña) 1920	60 – 260	0.025-10	15-300	5-8000	85
	MICHELL-BANKI	A.G. Michell (Australia) 1903 D. Banki (Hung) 1917-1919	40-160	0.025-5	1-50 (200)	1-750	82
REACCIÓN	Bomba Rotodinámica	Dionisio Papin (Francia) 1689	30-170	0.05-0.25	10-250	5-500	80
	FRANCIS	James Francis (G. Bretaña) 1848	L: 60-150 N: 150-250 R: 250-400	1-500	2-750	2-750000	92
	DERIAZ	P. Deriaz (Suiza) 1956	60-400	500	30-130	100,000	92
	KAPLAN y de hélice	V. Kaplan (Austria) 1912	300-800	1000	5-80	2-200000	93
	AXIALES: - Tubular - Bulbo - Generador Periférico	 Kuhne-1930 Hugenin-1933 Harza-1919	300-800	600	5-30	100,00	93

Nota: NS: velocidad específica
Ch: chorro
L: lento
N: normal
R: rápida

En la Tabla 1.3 (**ITDG** Perú **[32]**) se muestra la Eficiencia del grupo de generación y en la Tabla 1.4 (**ITDG** Perú **[32]**) se muestra la clasificación de mini centrales hidráulicas (M.C.H.) según la potencia.

TABLA 1.3 Eficiencia del grupo de generación.

Potencia	TIPO DE TURBINA			
kW	Pelton	Michell - Banki	Francis	Axial
< 50	58 – 65 %	54 – 62 %	59 – 65 %	58 – 66 %
51 – 500	65 – 69 %	62 – 65 %	66 – 70 %	66 – 70 %
501 – 5000	69 – 73 %	65* %	70 – 74 %	70 – 74 %

* Limitación por máxima potencia de 1000w.
*Fuente: ONUDI Mini Hydro Power Stations, UNIDO/OS, Viena (1981)

En la **tabla 1.4** se observan los valores de potencia establecidos para la clasificación de una mini central hidráulica (MCH) por organismos internacionales.

Tabla 1.4 Clasificación de MCH según la potencia.

REGIÓN	INSTITUCIÓN	MICRO CENTRAL	MINI CENTRAL	PEQUEÑA CENTRAL	GRANDES CENTRALES
Mundial	ONUDI[1]	< 100 kW	101 - 2,000 Kw	>2,000 – >10,000 kW	>10,000 kW
Latinoamérica	OLADE[2]	< 50 kW	51 – 500 Kw	>500 – <5,000 kW	>5,0000 kW

1. Organización de las Naciones Unidas para el Desarrollo Industrial.
2. Organización Latinoamericana de la Energía.

Para el desarrollo del trabajo de investigación se ha propuesto los siguientes valores característicos de las turbinas hidráulicas (turbinas de acción y turbinas de reacción), que se resumen en la **Tabla 1.5**. Las turbinas hidráulicas modernas están clasificadas hoy en día en dos grandes grupos y que se emplean en las centrales hidráulicas, sean estas micro, mini, pequeña o grandes.

TABLA 1.5 Valores característicos de las turbinas hidráulicas.

Clases de Turbinas		Tipos de Rodete	Ns (rpm)	Q (m^3/seg)	H_{max} adm. (m)	η %	P kW
TURBINAS DE ACCIÓN	PELTON	1 Ch	10 – 30	0.03 – 0.41	90 – 300	70-91	30 – 900
		2 Ch	30 – 50	0.07 – 1.1	60 - 300		30 - 2500
		4 Ch	30 – 50	0.65 – 2.0	150 - 300		750 - 4000
	TURGO		60 – 260	0.83 – 4.5	35 – 175	65-85	750 – 1500
	MICHELL-BANKI		40 – 160	0.12 – 1.1	11 – 90	65-82	30 - 150
Turbinas de Reacción	BOMBA ROTODINAMICA		30 – 170	0.05 - 0.25	10 – 250	60-80	5 – 50
	FRANCIS	Lenta	60 - 125	1.3 – 7	50 – 200	80-92	2000 - 4000
		Normal	125 – 225	0.25 – 2.5	20 – 150		150 - 750
		Rápida	225 – 350	0.6 – 12	10 - 55		30 - 4000
		Extra rápida	350 - 450	0.7 – 3.0	5 - 9		30 - 180
	KAPLAN	Kaplan	300 – 600	5 – 25	8,5 – 35	90	400 - 4000
		Hélice	500 – 1000	1.4 – 11	2,5 – 10	85	30 – 400
		Tubular	300 – 1000	7.4 – 25	2,5 – 7	93	400 – 1500

Wilfredo Jara T. [43] (1998). "Máquinas Hidráulicas" Fondo Editorial INI-FIM. Instituto de Investigación de la Facultad de Ingeniería Mecánica. Este manual está estructurado en dos partes. La primera desarrolla lo referente a las turbinas Hidráulicas, el estudio teórico a los diversos tipos de turbinas y un análisis detallado de las turbinas de reacción (Francis, Kaplan, Hélice) y de Acción (Pelton). La segunda parte está dedicada a las bombas centrífugas.

TSUGUO NOZAKI, [46]. "Guía para la Elaboración de Proyectos de Pequeñas Centrales Hidroeléctricas destinadas a la Electrificación Rural del Perú" Julio de 1968. Esta guía tiene por objeto proporcionar a los Ingenieros Civiles y Mecánicos, bases y capacidad como proyectistas de Centrales Hidráulicas pequeñas, por cuanto son pocos los ingenieros especialistas en proyectos de Centrales Hidráulicas. Esta guía incluye varias tablas para la estimación de costos. En

la Tabla 1.6 se muestra la relación del número de polos con el de revoluciones y la frecuencia del generador.

Tabla 1.6 Número de polos con la frecuencia del generador.

Número de polos	Generador Velocidad de Rotación RPM		Número de polos	Generador Velocidad de Rotación RPM	
	Frecuencia 50	Frecuencia 60		Frecuencia 50	Frecuencia 60
4	1,500	1,800	20	300	360
6	1,000	1,200	22	272	327
8	750	900	24	250	300
10	600	720	26	231	277
12	500	600	28	214	257
14	428	514	30	200	240
16	375	450	32	187.5	225
18	333	400			

Tsuguo Nozaki sostiene que se elige el tipo de turbina en función de la caída y el caudal empleando la figura 1.1.

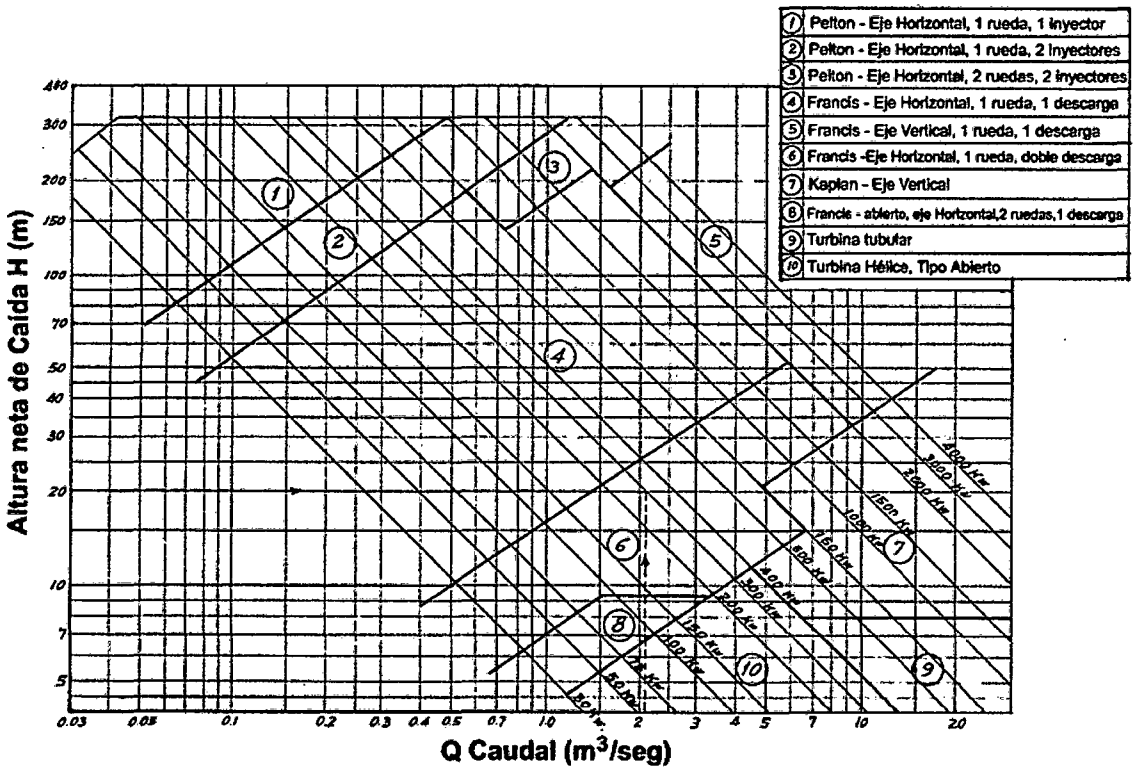


Fig. 1.1 Alcance y aplicación del tipo de turbina.

En el desarrollo del trabajo de investigación y empleando el gráfico mostrado en la figura 1.1, se ha propuesto la **figura 1.2** en donde podemos realizar

una selección rápida de una turbina para una mini central; para utilizar este diagrama se requiere conocer la altura y caudal.

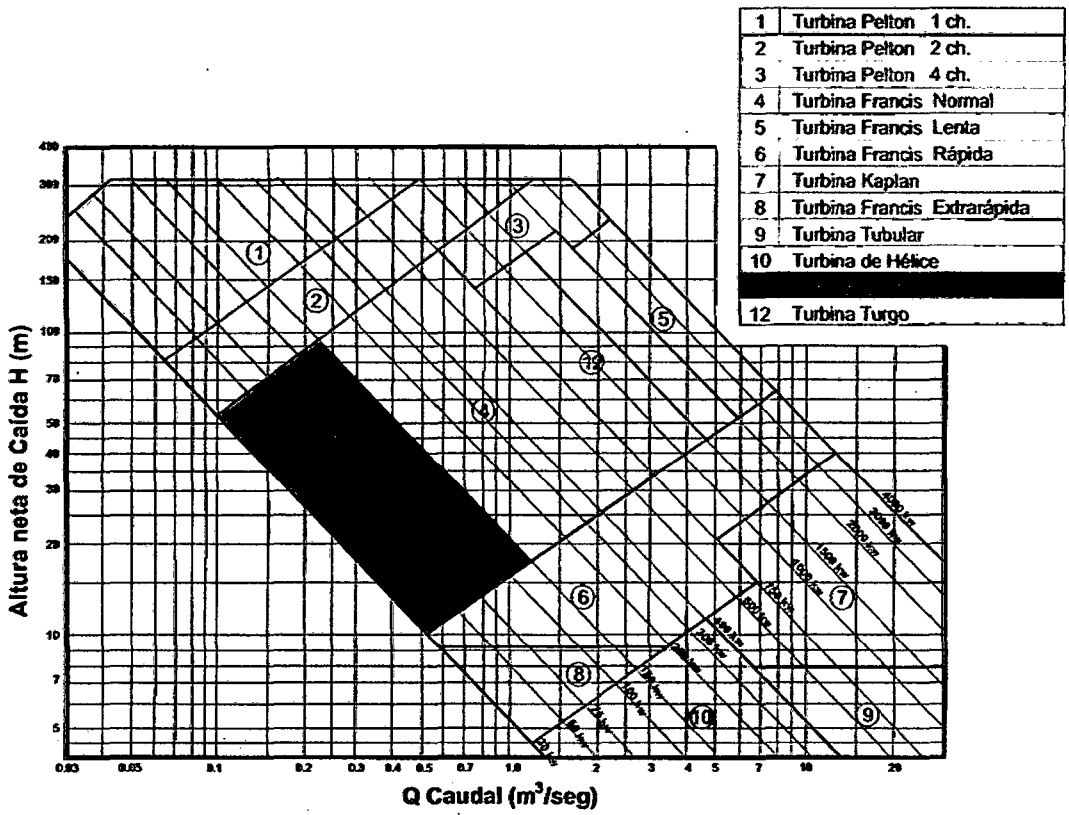


Fig. 1.2 Diagrama de selección de turbinas hidráulicas.

TSUGUO NOZAKI, [47]. "Guía para la Elaboración de Estudios Preliminares de Factibilidad de Proyectos de Pequeñas Centrales de Mediana o Grande Capacidad en el Perú" Enero de 1969. Esta guía tiene por objeto proporcionar a los ingenieros proyectistas e inspectores de proyectos de Centrales Hidráulicas, bases y capacidad en estas grandes centrales. La presente guía capacita al proyectista a confeccionar los estudios de mercado de energía eléctrica, hidrología del río, geología, topografía, ubicación del proyecto, estimación del costo de construcción, comparación de las alternativas y justificación del proyecto.

A.R. BARRON. [7] *"Las R*edes Neuronales" El cerebro consta de un gran número (aproximadamente 10^{11}) de elementos altamente interconectados

(aproximadamente 10^4 conexiones por elemento), llamados **neuronas**. Estas neuronas tienen tres componentes principales:

1. ***dendritas***,
2. ***cuerpo de la célula*** o soma,
3. ***axón***.

Estas componentes tienen las siguientes funciones:

Las **dendritas**, son el árbol receptor de la red, son como fibras nerviosas que cargan de señales eléctricas el cuerpo de la célula.

El **cuerpo de la célula**, realiza la suma de esas señales de entrada.

El **axón** es una fibra larga que lleva la señal desde el cuerpo de la célula hacia otras neuronas. El punto de contacto entre un ***axón*** de una célula y una ***dendrita*** de otra célula es llamado **sinapsis**. En la figura **1.3** se muestra un esquema simplificado de la interconexión de dos neuronas biológicas.

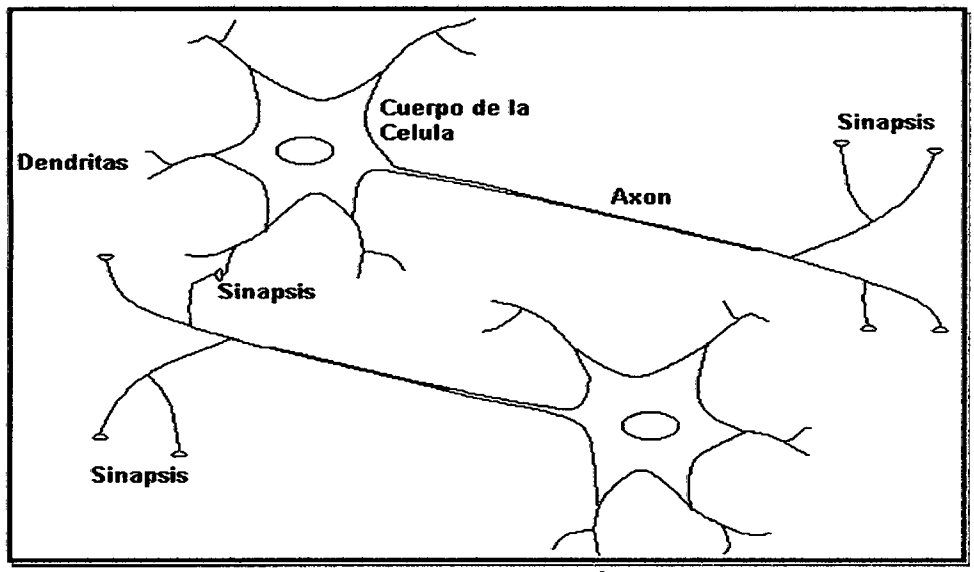


Fig. 1.3 Neuronas Biológicas.

Existen varias formas de nombrar una neurona artificial, es conocida como nodo, celda, unidad o ***elemento de procesamiento*** (**EP**); En la **figura 1.4** se observa un **EP** en forma general y su similitud con una neurona biológica.

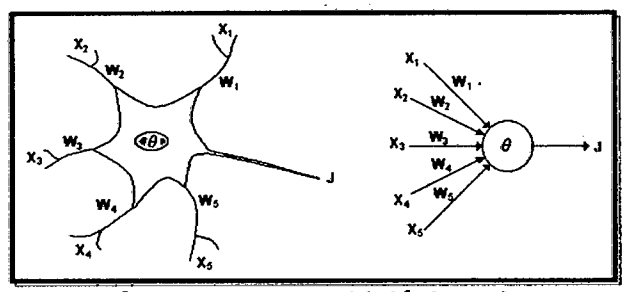


Fig. 1.4 Comparación de la neurona biológica y la neurona artificial.

Dentro de una red neuronal, los elementos de procesamiento se encuentran agrupados por capas, como se muestra en la **figura 1.5**; una capa es una colección de neuronas; de acuerdo a la ubicación de la capa en la **RNA**, esta recibe diferentes nombres:

1. **Capa de entrada:** Recibe las señales de la entrada de la red, algunos autores no consideran el vector de entrada como una capa pues allí no se lleva a cabo ningún proceso.
2. **Capas ocultas:** Estas capas son aquellas que no tienen contacto con el medio exterior, sus elementos pueden tener diferentes conexiones y son estas las que determinan las diferentes topologías de la red.
3. **Capa de salida:** Recibe la información de la capa oculta y transmite la respuesta al medio externo.

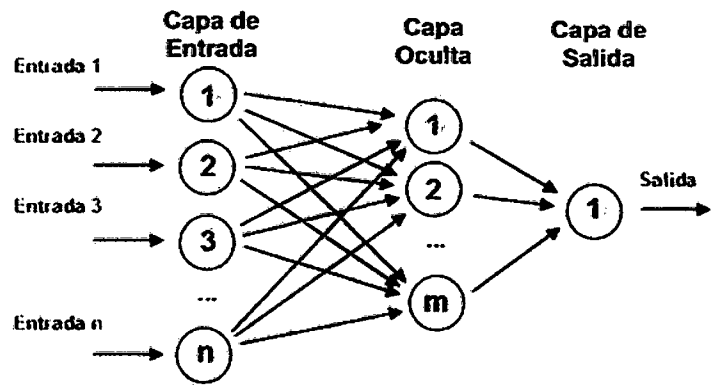


Fig. 1.5 Capas de una red.

Los principales tipos de redes se clasifican en:

1. **PERCEPTRÓN.**
2. **ADALIN**
3. **BACKPROPAGATION.**

CAPÍTULO 2

MODELOS MATEMÁTICOS PARA LAS DIMENSIONES DE LAS TURBINAS

2.1 MODELO MATEMÁTICO POR EL MÉTODO TRADICIONAL

2.1.1 SELECCIÓN DE LAS TURBINAS HIDRÁULICAS

Las turbinas hidráulicas no pueden fabricarse en serie. Cada salto (**H, Q**) requiere un diseño concreto. La velocidad especifica salto $\mathbf{n_s}$ es el parámetro clave para fijar en primer lugar el tipo de turbina y en segundo lugar la forma y el dimensionamiento correspondientes.

En Europa la frecuencia de la corriente eléctrica es de **50 Hz** (en América es de **60Hz**, por lo que la velocidad (el número de revoluciones) en **rpm** será de **3000** (para 1 par de polos en el alternador), 1500 (para 2 pares de polos), 1000 (para 3 pares de polos), 750 (para 4 pares de polos), y así sucesivamente.

Los datos que se necesita son la altura neta **H** y el caudal normal, o de diseño Q^*. Para calcular la potencia normal P_e^* que vamos a disponer a partir de H y Q , tenemos la expresión antes desarrollada:

1. POTENCIA DE LA TURBINA (P).

La potencia en hp de la turbina se determina utilizando la siguiente relación:

$$P = \frac{1000 \times Q \times H \times \eta_t}{76} \qquad (2.1)$$

En donde: **P** = Potencia en Hp

Q = Caudal en metros cúbicos por segundo (m^3/seg)

H = Altura del salto (salto neto) en metros (m),

η_t = Rendimiento de la turbina, en tanto por uno; $\eta_t/100$

1000 = Cantidad de litros de agua (o kg) en un metro cúbico (m^3)

76 = Proviene de la equivalencia existente entre el HP y el kgm/seg, como unidades de potencia;

Así: 76 kgm/seg 1 HP

100xQxHxη_t kgm/seg P

De donde: $P = \frac{1000 \times Q \times H \times \eta_t}{76}$

La potencia en KW de la turbina se utiliza la siguiente relación:

$$P_{kw} = \frac{1000 \times Q \times H \times \eta_T \times HP}{76} \times \frac{1kw}{1.340Hp}$$

$$P_{kW} = \frac{1000 \times Q \times H \times \eta_T}{1.340 \times 76} \qquad (2.2)$$

Ampliando el cálculo al alternador y considerando el rendimiento total η_T, producto del rendimiento η_t en la turbina por el rendimiento η_a en el alternador tenemos: $\eta_T = \eta_t + \eta_a$

2. NÚMERO ESPECÍFICO DE REVOLUCIONES DE POTENCIA (Ns).

La velocidad específica representada normalmente por N_s, denominada también **velocidad específica absoluta o velocidad angular**

específica, corresponde al número de revoluciones por minuto que daría una turbina semejante a la que se desea proyectar (de igual forma pero dimensiones reducidas), la cual, instalada en un salto de 1 m. de altura, proporcionaría una potencia de 1CV (0.9862 HP). Para calcular el número específico de revoluciones, utilizamos la siguiente relación:

$$N_s = \frac{n\sqrt{\frac{P}{i}}}{H.\sqrt[4]{H}} \qquad (2.3)$$

En donde: $\mathbf{N_s}$ = Velocidad específica en rpm
$\mathbf{n}$ = Velocidad de rotación en rpm
$\mathbf{P}$ = Potencia en HP
$\mathbf{H}$ = Altura del salto en metros
$\mathbf{i}$ = Número de boquillas

$$\text{Si i} = 1Ch, \quad N_s = 0.03572 \times n \qquad (2.3a)$$

$$\text{Si i} = 2Ch, \quad N_s = 0.02526 \times n \qquad (2.3b)$$

$$\text{Si i} = 4Ch, \quad N_s = 0.01786 \times n \qquad (2.3c)$$

2.1.2 MODELO TRADICIONAL EN EL DISEÑO DE UNA TURBINA PELTON ([31] Quantz L, [43] Jara W., [32] ITDG Perú)

Las turbinas Pelton son turbinas de chorro libre que se acomodan a la utilización de saltos de agua con mucho desnivel y caudales relativamente pequeños, con márgenes de empleo entre 60 y 300 metros, consiguiéndose rendimientos máximos del orden del 90%.

1. Salto neto en la Turbina Pelton de un inyector (H_n)

En el caso de un solo inyector y eje de la turbina horizontal, si se considera la zona comprendida desde inmediatamente antes del inyector,

punto A de la figura 2.1, hasta el punto de tangencia del chorro con la circunferencia media de la rueda, punto A1, de acuerdo con la definición dada de salto neto, se tiene:

$$H_n = \frac{C_o^2}{2g} + \frac{p_o}{\gamma} + z_o - z_a$$

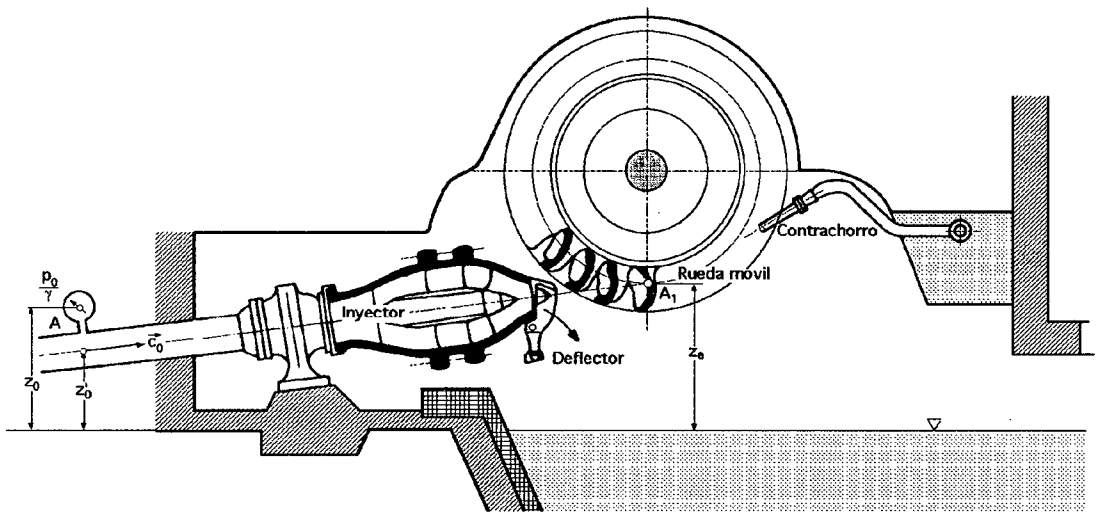


Fig. 2.1 Turbina Pelton de un inyector.

2. Salto neto en la turbina Pelton de varios inyectores.

Si por ejemplo se considera que la turbina tiene dos inyectores, figura **2.2**, de diferentes características que proporcionan los caudales Q_1 y Q_2, (caso poco frecuente), el estudio se puede hacer como si el conjunto constase de dos turbinas, para los respectivos caudales Q_1 y Q_2, saltos correspondientes H_{n1} y H_{n2}, y potencias respectivas N_{n1} y N_{n2}, de la forma:

$$H_{n1} = \frac{C_{o1}^2}{2g} + \frac{p_{o1}}{\gamma} + z_{o1} - z_{a1} \quad ; N_{n1} = \gamma Q_1 H_{n1}$$

$$H_{n2} = \frac{C_{o2}^2}{2g} + \frac{p_{o2}}{\gamma} + z_{o2} - z_{a2} \quad ; N_{n2} = \gamma Q_2 H_{n2}$$

$$N_n = \gamma Q_1 H_{n1} + \gamma Q_2 H_{n2}$$

$$N_n = \gamma Q_1 (\frac{C_{o1}^2}{2g} + \frac{p_{o1}}{\gamma} + z_{o1} - z_{a1}) + \gamma Q_2 (\frac{C_{o2}^2}{2g} + \frac{p_{o2}}{\gamma} + z_{o2} - z_{a2})$$

En este caso se puede tomar como salto neto el salto neto promediado H_n, que es el que tendría una turbina de un solo inyector que con el caudal total, **Q = Q1 + Q2**, diese la misma potencia, es decir:

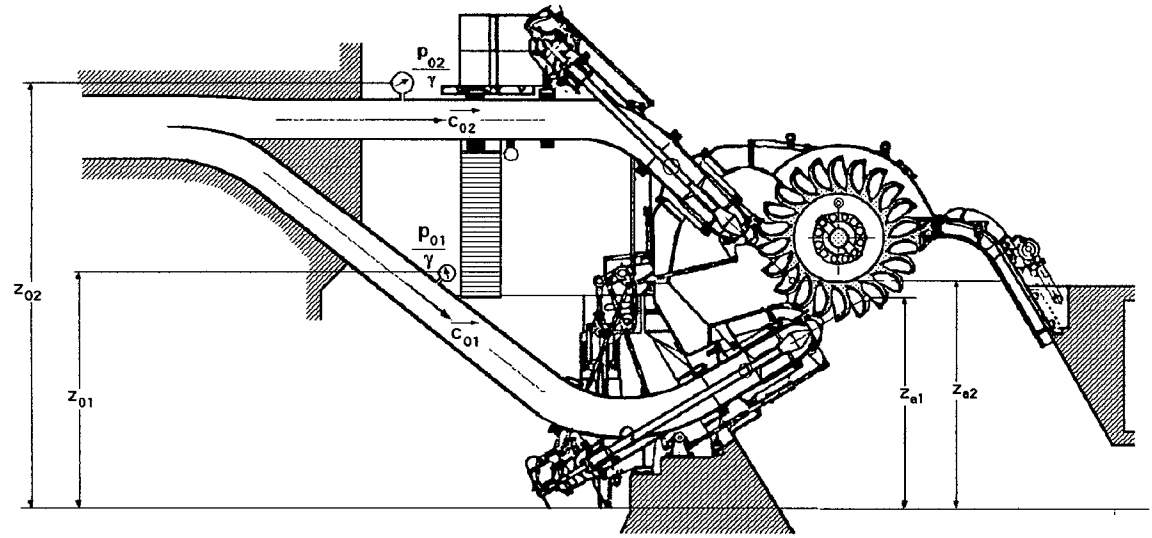


Fig. 2.2 Turbina Pelton de dos inyectores.

$$\gamma Q_1 H_{n1} + \gamma Q_2 H_{n2} = \gamma (Q_1 + Q_2) H_n = \gamma Q H_n$$

$$H_n = \frac{Q_1 (\frac{C_{o1}^2}{2g} + \frac{p_{o1}}{\gamma} + z_{o1} - z_{a1}) + Q_2 (\frac{C_{o2}^2}{2g} + \frac{p_{o2}}{\gamma} + z_{o2} - z_{a2})}{Q_1 + Q_2}$$

$$H_n = \frac{Q_1 H_{n1} + Q_2 H_{n2}}{Q}$$

Que se puede ampliar fácilmente para una turbina de eje horizontal y cualquier número de inyectores. Si la turbina fuese de eje vertical, las expresiones se simplifican, (Hn1 = Hn2 =...), sobre todo, en el caso de tener los inyectores la misma sección, (Q1 = Q2 =...), caso cada día más frecuente.

2.1.2.1 DISEÑO DE LOS DIÁMETRO PRINCIPALES.

El rodete de una turbina Pelton mostrado en las figuras **2.3** y **2.4**, se observa que se necesita calcular el diámetro del chorro (**d** ó $\mathbf{d_{ch}}$), diámetro del rodete ($\mathbf{D_2}$) y el diámetro externo ($\mathbf{D_3}$).

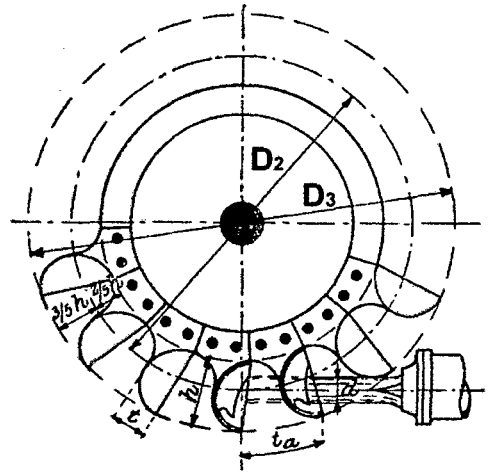


Fig. 2.3 Rodete de una turbina Pelton.

1. Diámetro del chorro (d_{ch})

$$d_{ch} = 550\sqrt{\frac{Q}{i\sqrt{H}}} \qquad (2.4)$$

En donde: $\mathbf{d_{ch}}$ = Diámetro del chorro, en mm

Q = Caudal en metros cúbicos por segundo (m^3/seg)

H = Altura del salto en metros (m),

i = Número de boquillas

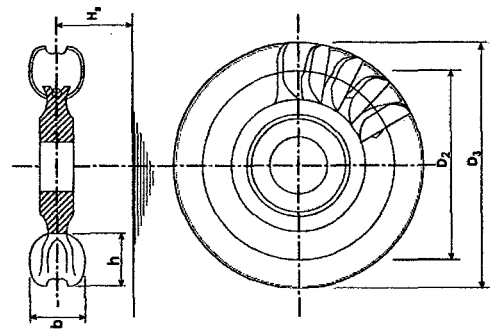


Fig. 2.4 Diámetro del rodete.

2. Diámetro medio del Rodete (D_2)

$$D_2 = k_{D2} \times \frac{1000\sqrt{H}}{n} \qquad (2.5)$$

En donde: D_2 = Diámetro medio del rodete en mm
H = Altura del salto en metros (m),
K_{D2} = Constante, $37 \geq k_{D2} \leq 39$
n = Velocidad de rotación en rpm

3. Diámetro externo del Rodete (D_3)

$$D_3 = D_2 + 2\left[\frac{3}{5} \times h\right] \qquad (2.6)$$

En donde: D_3 = Diámetro externo del rodete, en mm
D_2 = Diámetro del rodete, en mm
h = Altura de la cuchara, en mm

4. Diseño de las dimensiones de las paletas

Las dimensiones de la cuchara son proporcionales al diámetro del chorro, la figura **2.5** muestra las proporciones básicas de una turbina Pelton.

Las paletas o cazoletas, en las versiones más modernas, tienen forma de elipsoide; la arista que las divide en dos puede quedar al ras de los bordes de las mismas, o a veces se queda algo adentro, como se observa en la figura **2.6**. Las medidas se adoptan en función del diámetro del chorro.

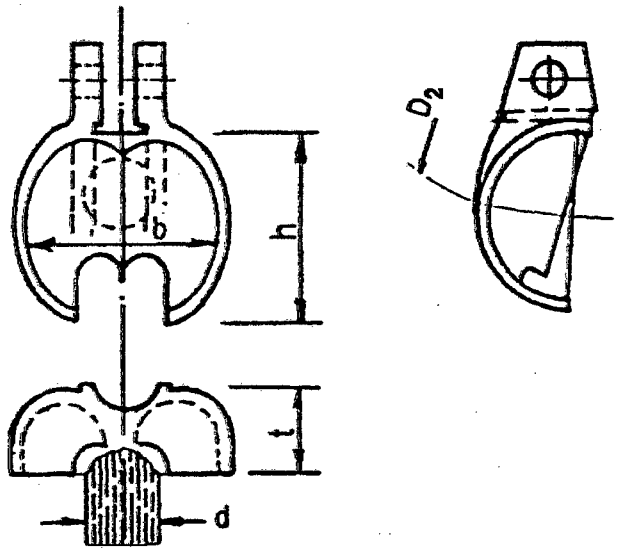


Fig. 2.5: Dimensiones básicas de una paleta Pelton.

1. Ancho de la paleta: b = 3.75 x d_{ch} (2.7a)
2. Altura de la paleta: h = 3.50 x d_{ch} (2.7b)
3. Espesor de la paleta: t = 1.50 x d_{ch} (2.7c)
4. El número de cucharas la calculamos de la siguiente relación:

$$Z = \frac{1}{2}\left(\frac{D_2}{d_{ch}}\right) + k_z \qquad (2.8)$$

En donde: D_2 = Diámetro medio del rodete, en mm
D_{ch} = Diámetro del chorro, en mm
Kz = Constante, $14 \leq k_z \leq 16$
Z = Número de paletas

5. El paso t_e, entre cucharas se calcula utilizando la siguiente relación:

$$t_e = \frac{\pi \cdot D_3}{Z} \qquad (2.9)$$

En donde: D_3 = Diámetro exterior del rodete, en mm
t_e = Paso entre paletas, en mm
Z = Número de paletas

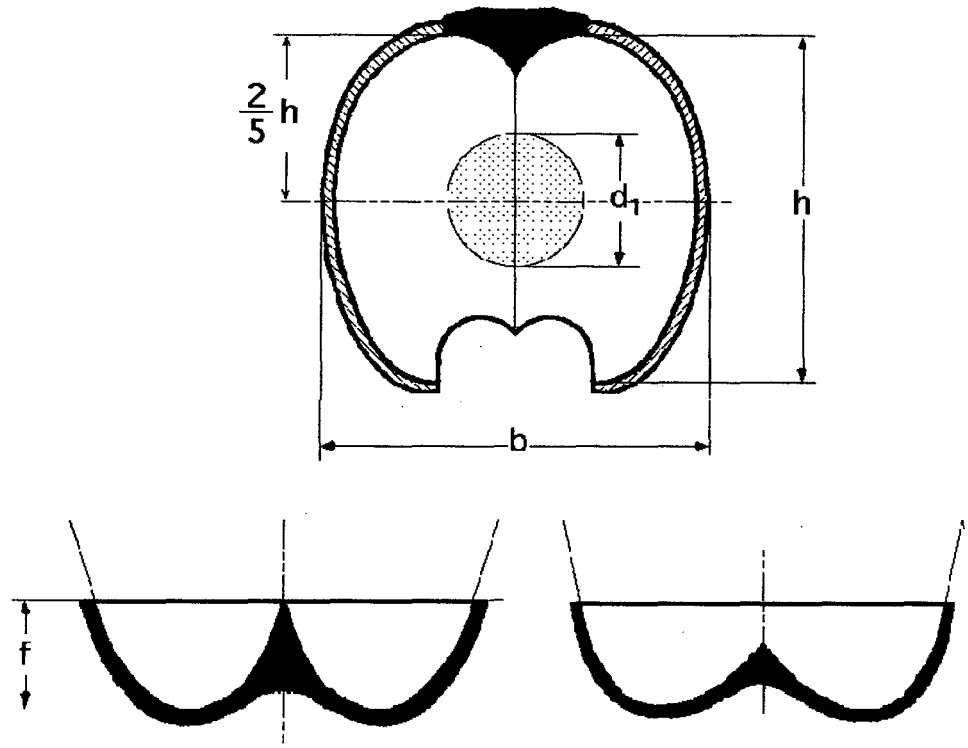


Fig. 2.6: Formas modernas de una paleta Pelton.

Las paletas no se colocan exactamente en sentido radial, sino en forma tal que el chorro al alcanzar de lleno una de ellas, se halle perpendicular a la arista de la misma, quedando separada la paleta del inyector el mínimo que permita la construcción, atacándola el chorro lo más cerca posible de la corona del rodete, para que las pérdidas a la salida resulten más pequeñas, haciendo que la circunferencia tangente al chorro (circunferencia Pelton), corte a las cazoletas a 2h/5 medido desde el interior. Las paletas tienen que ir dispuestas de tal forma, que su separación no permita que se pierda agua, es decir, cuando el chorro abandone una, debe encontrarse con la siguiente, figura **2.7**.

La paleta en la posición (a) entra en contacto con el agua, en la (b) está en un punto intermedio, de forma que capta una parte del chorro, y en la (c) capta todo el chorro. El tiempo que tardaría una partícula ficticia de agua en recorrer el espacio (AF) sería el mismo que tardaría el borde de la cazoleta en recorrer el espacio (AE), por lo que:

$$t = \frac{\overline{AF}}{c_1} = \frac{\overset{\frown}{AE}}{u}$$

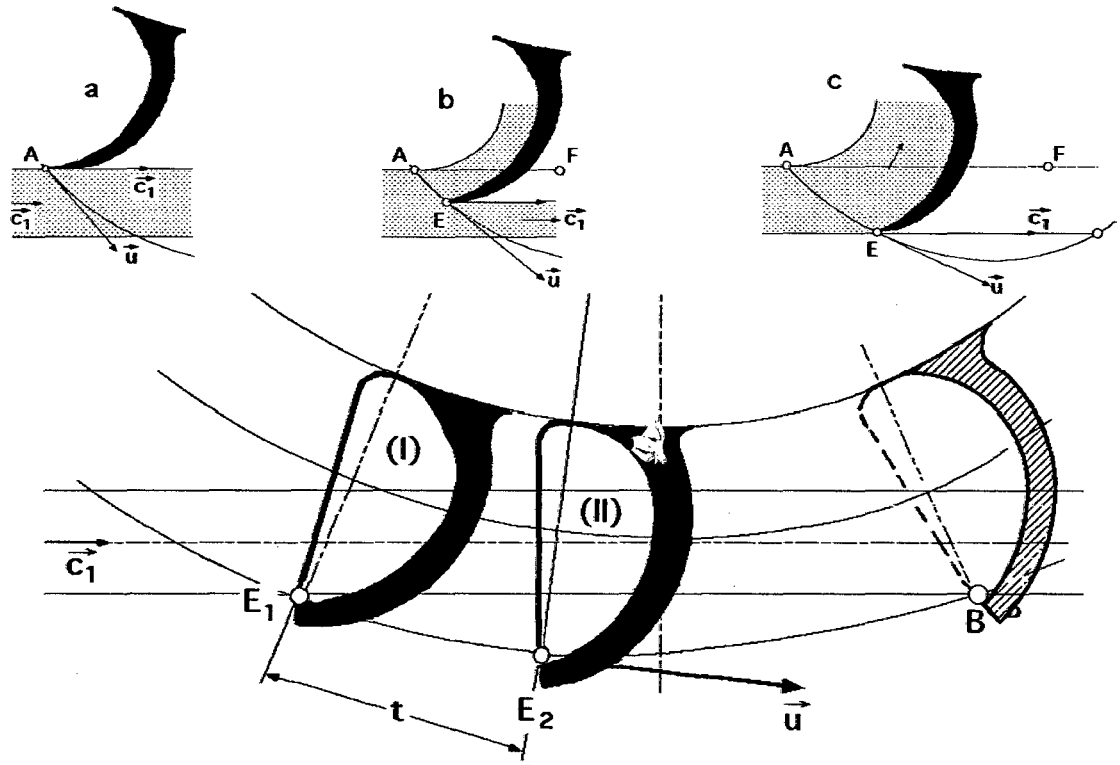


Fig. 2.7: Separación entre paletas Pelton.

2.1.3 MÉTODO TRADICIONAL EN EL DISEÑO DE UNA TURBINA FRANCIS [31] Quantz L, [43] Jara W., [32] ITDG Perú.

Las turbinas Francis, son de tipo radial, admisión centrípeta y tubo de aspiración; siempre se construyen en condiciones de rendimiento máximo, dando lugar a cuatro tipos fundamentales, lentas, normales, rápidas y extra rápidas, diferenciándose unas de otras en la forma del rodete. Haciendo uso de la ecuación fundamental de las turbinas en condiciones de rendimiento máximo $\alpha_2 = 90°$ resulta:

$$c_1 u_1 Cos\alpha_1 = \eta_{hid} g H_n$$

ó

$$c_{1n} u_1 = \eta_{hid} g H_n$$

2.1.3.1 TIPOS DE RODETES

1. Rodetes lentos:

Se utilizan en los grandes saltos, figura **2.8**; con ellos se tiende a reducir el número de revoluciones, lo cual supone un aumento del diámetro D_1 del rodete respecto al del tubo de aspiración D_3 ($D_1 > D_3$). El ángulo a la entrada $\beta_1 < 90^o$, ($\alpha_1 < 15^o$) y su número de revoluciones específico está comprendido entre 60 y 125. En estas turbinas se obtienen velocidades tangenciales reducidas. Los álabes tienen forma especial, aumentando su espesor a fin de que su cara posterior guíe mejor el chorro que atraviesa el rodete deslizándose en contacto con las paredes de los álabes, ya que de no ser así el chorro se despegaría de la cara posterior de los mismos, originando remolinos.

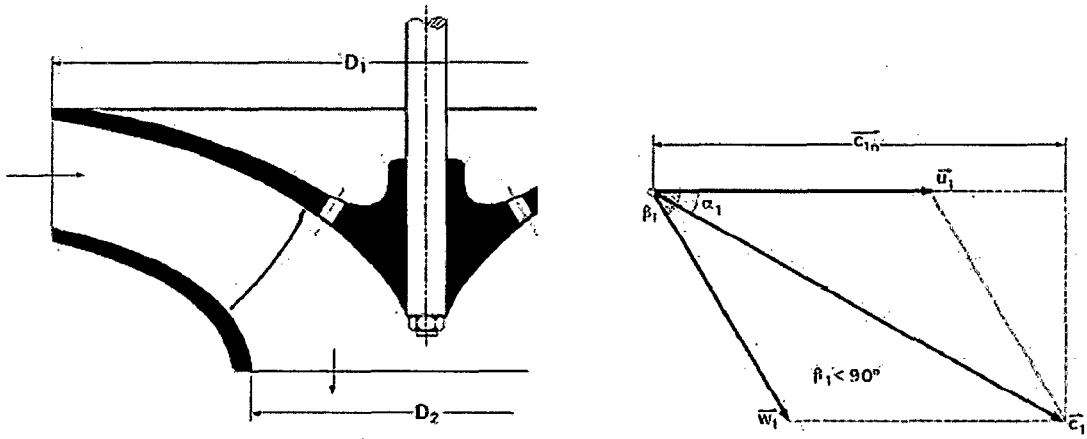


Fig. 2.8 Rodete Francis Lento, $\beta_1 > 90^o$

2. Rodetes normales:

Se caracterizan porque el diámetro D_1 es ligeramente superior al del tubo de aspiración D_3, figura **2.9**. El agua entra en el rodete radialmente y sale de él axialmente, entrando así en el tubo de aspiración. El valor de β_1 es del orden de 90°, ($15^o < \alpha_1 < 30^o$) y se alcanza un n_s comprendido entre 125 y 225 rpm. No existen apenas huelgos entre el distribuidor y la rueda.

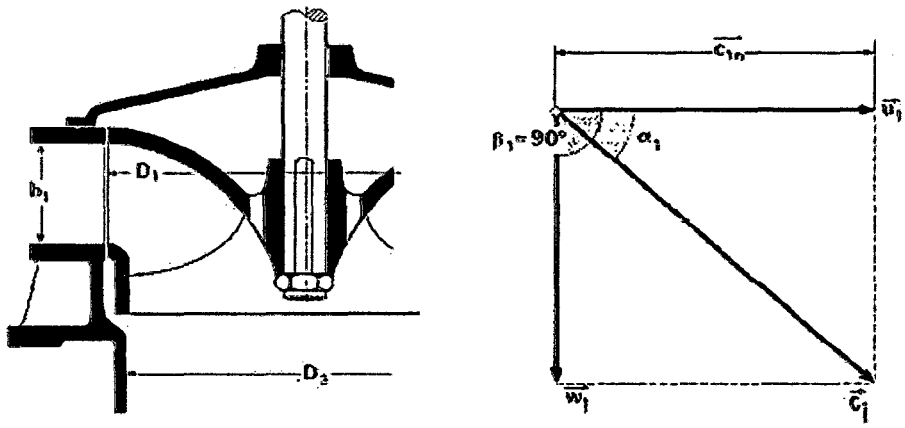


Fig. 2.9 Rodete Francis Normal, $\beta_1 = 90^o$

En estas turbinas, en el triángulo de velocidades a la entrada, al ser $\beta_1 = 90^o$, se cumple:

$$u_1 = c_1 Cos\alpha_1$$

$$u_1^2 = \eta_{hid} g H_n$$

3. Rodetes rápidos.-

Permiten obtener elevadas velocidades de rotación para valores de n_s comprendidos entre 225 y 450 (rápida: 225 y 350, extra rápidos: 350 y 450), figura **2.10** y **2.11**. El diámetro del rodete D_1 es menor que el D_3 del tubo de aspiración y el cambio de dirección del agua se efectúa más bruscamente que en las turbinas normales.

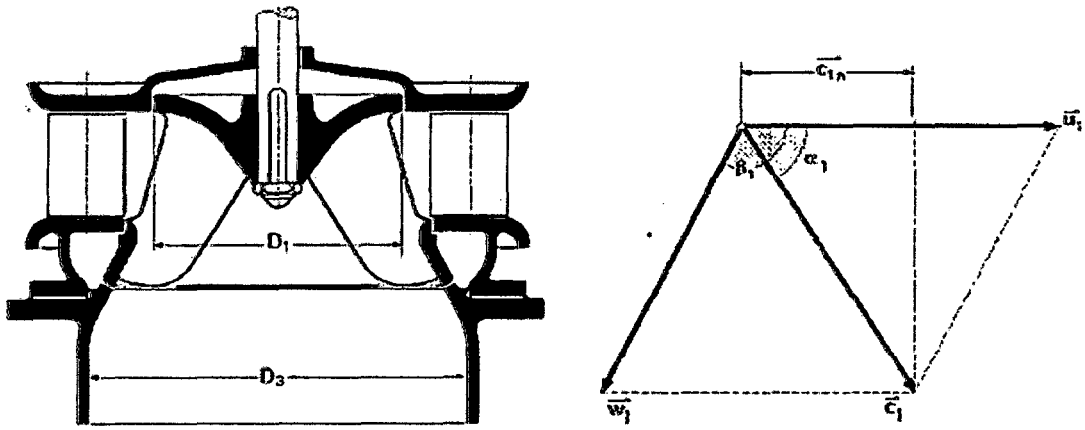


Fig. 2.10 Rodete Francis Rápido, $\beta_1 < 90^o$

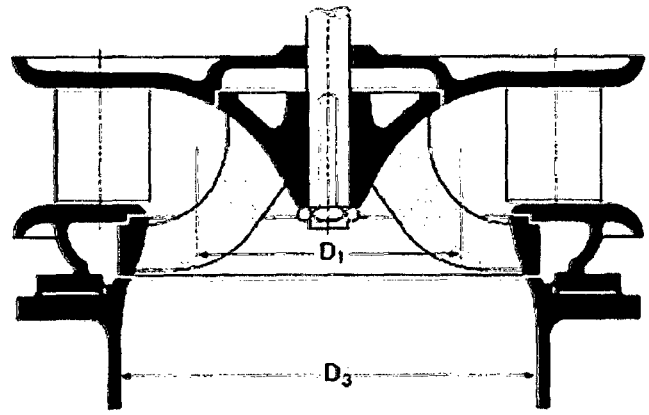


Fig. 2.11 Rodete Francis Extra Rápido, $\beta_1 < 90^o$

El rodete de una turbina Francis mostrado en la figura **2.12**, se observa que se necesita calcular los diámetros del rodete D_1, D_2 y D_3.

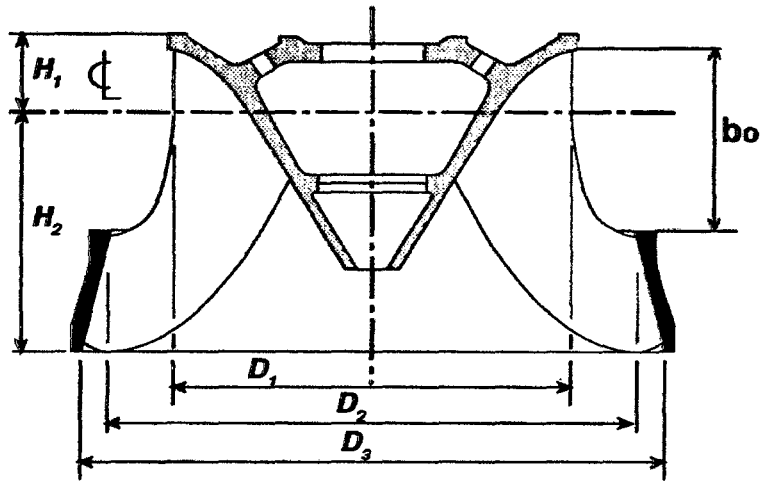


Fig. 2.12 Dimensiones principales de un rodete.

2.1.3.2 DISEÑO DE LOS DIÁMETRO PRINCIPALES.

1. DIÁMETRO DEL TUBO DE ASPIRACIÓN (D_3)

$$D_3 = 1000\sqrt{\frac{4Q}{\pi C_3}} \qquad (2.10)$$

En donde: D_3 = Diámetro en el tubo de aspiración, en mm

Q = Caudal en metros cúbicos por segundo (m^3/seg)

C_3 = Velocidad de salida, m/seg

Para determinar la velocidad de salida C_3 utilizamos la ecuación (2.11):

$$C_3 = \sqrt{\frac{2 \times g \times k_c \times H}{100}} \qquad (2.11)$$

En donde: C_3 = Velocidad de salida, m/seg
g = Gravedad, en m/seg^2
H = Altura del salto en metros (m),
K_c = Constante según la turbina, en tanto por uno; $\eta_t/100$
Considerar 4% para Francis Lenta, 6% para Francis Normal, 12% para Francis Rápida y 25% para la Francis Extra rápida.

2. DIÁMETRO EXTERNO DEL RODETE (D_1)

Para calcular el diámetro externo del rodete D_1 (Lenta, Normal, Rápida y Extra rápida) de la turbina mostrada en la figura **2.10**, utilizamos la siguiente relación:

$$D_1 = D_3 \times (0.4 + \frac{94.5}{N_S}) \quad (2.12)$$

En donde: D_1 = Diámetro externo del rodete, mm
D_3 = Diámetro del tubo de aspiración, en mm
N_s = Velocidad específica, en rpm

3. DIÁMETRO INTERNO DEL RODETE (D_2)

Para calcular el diámetro interno del rodete D_2 de la turbina mostrada en la figura **2.10**, utilizamos la siguiente relación:

$$D_2 = D_3 \times (0.96 + 0.00038 \times N_s) \quad (2.13)$$

En donde: D_2 = Diámetro interno del rodete, mm

D_3 = Diámetro del tubo de aspiración, en mm

N_s = Velocidad específica, en rpm

2.1.3.3 ANCHO DE LA CORONA DIRECTRIZ (bo)

El ancho de la corona directriz se calcula según la turbina Francis seleccionada, utilizando las siguientes ecuaciones:

1. FRANCIS LENTA (F_L)

$$bo = D_1 \left[\frac{(N_S - 60) \times 0.07}{65} + k_L \right] \quad (2.14a)$$

En donde: b_o = Ancho de la corona directriz, en mm

D_2 = Diámetro interno del rodete, en mm

N_s = Velocidad específica, en rpm

k_L = Constante de diseño para Francis Lenta, $0.08 \leq k_L \leq 0.119$ (seleccionar uno de los dos valores)

2. FRANCIS NORMAL (F_N)

$$bo = D_1 \left[\frac{(N_S - 125) \times 0.15}{100} + k_N \right] \quad (2.14b)$$

En donde: b_o = Ancho de la corona directriz, en mm

D_2 = Diámetro externo del rodete, en mm

N_s = Velocidad específica, en rpm

k_N = Constante de diseño para Francis Normal.

$0.15 \leq k_N \leq 0.30$ (seleccionar uno de los dos valores)

3. FRANCIS RÁPIDA (F_R)

$$bo = D_1\left[\frac{(N_S - 225)\times 0.22}{125} + k_R\right] \qquad (2.14c)$$

En donde: $\mathbf{b_o}$ = Ancho de la corona directriz, en mm

$\mathbf{D_1}$ = Diámetro interno del rodete, en mm

$\mathbf{N_s}$ = Velocidad específica, en rpm

$\mathbf{K_R}$ = Constante de diseño para Francis rápida.

$0.1582 \leq \mathbf{k_R} \leq 0.30$ (seleccionar uno de los dos valores)

4. FRANCIS EXTRA RÁPIDA (F_{ER})

$$bo = D_1\left[\frac{(N_S - 225)\times 0.19}{100} + k_{ER}\right] \qquad (2.14d)$$

En donde: $\mathbf{b_o}$ = Ancho de la corona directriz, en mm

$\mathbf{D_2}$ = Diámetro interno del rodete, en mm

$\mathbf{N_s}$ = Velocidad específica, en rpm

$\mathbf{K_{ER}}$ = Constante de diseño para Francis Normal.

$0.1807 \leq \mathbf{k_{ER}} \leq 0.52$ (seleccionar uno de los dos valores)

2.1.3.4 CÁLCULO DEL NÚMERO DE ÁLABES (Z_o)

Perfil de los álabes de las directrices.- Las directrices son superficies desarrollables cilíndricas de generatrices paralelas al eje de rotación de la turbina; su perfil se determina teniendo en cuenta que no hay transformación de energía hidráulica en mecánica al paso del agua por el distribuidor, procurando evitar al máximo las pérdidas por rozamiento y torbellinos. Para calcular este perfil se determina la trayectoria ideal de la vena fluida; para ello, como el paso del agua por el distribuidor no genera ningún tipo de energía, si consideramos un punto A cualquiera de la trayectoria (0A1) del agua en el distribuidor, figura **2.11**, la condición:

$$dN = \gamma Q \eta_{hid} dH_n = \left| H_n = \frac{u_1 c_{1n} - u_2 c_{2n}}{g \eta_{hid}} \right|$$

$$dN = \gamma Q \frac{d(uc_n)}{g} = 0 \Rightarrow uc_n = cte$$

$$uc_n = rwc_n = |w = cte| = Cte \Rightarrow rc_n = k$$

Por lo que la circulación por el distribuidor es rotacional.

La componente C_n no proporciona caudal alguno, por lo que el caudal que atraviesa el distribuidor es:

$$Q = 2\pi r b_1 c_r = Cte$$

$$rc_r = \frac{Q}{2\pi b_1} = Cte$$

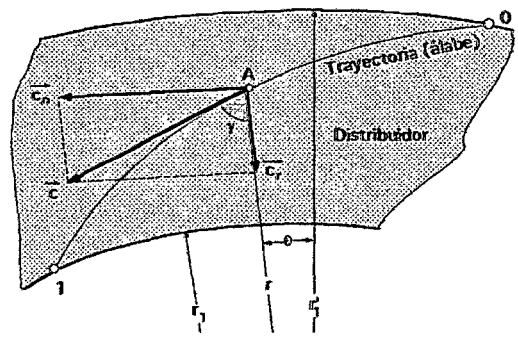


Fig. 2.13 Trayectoria de la vena fluida en el distribuidor.

El número de alabes para la turbina Francis seleccionada, se calcula con las siguientes ecuaciones:

1. FRANCIS LENTA (F_L)

$$Z_{FL} = 18 - \left[\frac{(N_S - 60)}{65} \right] \qquad (2.18)$$

En donde: Z_{FL} = Número de alabes para una turbina Francis lenta

N_s = Velocidad específica, en rpm

2. FRANCIS NORMAL (F_N)

$$Z_{FN} = 17 - \left[\frac{(N_S - 125)\times 2}{100}\right] \qquad (2.19)$$

En donde: Z_{FN} = Número de alabes para una turbina Francis normal
N_s = Velocidad específica, en rpm

3. FRANCIS RÁPIDA (F_R)

$$Z_{FR} = 15 - \left[\frac{(N_S - 225)\times 2}{125}\right] \qquad (2.20)$$

En donde: Z_{FR} = Número de alabes para una turbina Francis rápida
N_s = Velocidad específica, en rpm

4. FRANCIS EXTRA RÁPIDA (F_{ER})

$$Z_{FER} = 13 - \left[\frac{(N_S - 350)}{100}\right] \qquad (2.21)$$

En donde: Z_{FER} = Número de alabes para una turbina Francis extra rápida
N_s = Velocidad específica, en rpm

2.1.4 MODELO TRADICIONAL EN EL DISEÑO DE LA TURBINAS DE ACCIÓN TOTAL [31] Quantz L, [43] Jara W., [32] ITDG Perú

La importancia de las turbinas Hélice, Tubular y Kaplan en pequeños saltos con grandes caudales, las hacen idóneas tanto en posición horizontal como vertical. La tendencia a la construcción de turbinas cada vez más rápidas, para velocidades específicas ***Ns*** mayores de 450, conduce a las turbinas indicadas, ya que en las turbinas Francis con *Ns* del orden de 400, el agua no se puede guiar y conducir con precisión. El rodete está compuesto por unas pocas palas, que le

confieren forma de hélice de barco; cuando éstas sean fijas, se llama turbina hélice, mientras que si son orientables se denominan turbinas Kaplan; en ambos casos las turbinas funcionan con un único sentido de giro de rotación; son pues turbinas irreversibles.

La trayectoria del fluido recorre líneas contenidas en superficies cilíndricas de revolución en torno al eje de la turbina. Son de este tipo las turbinas tipo Kaplan, Hélice y Tubulares. Para una turbina hélice del tipo que sea, si se supone una velocidad de entrada c_1 uniforme para toda la altura del perfil, las distintas curvaturas de las palas se deducen de las distintas velocidades periféricas **u** que tiene la rueda en los diversos puntos, figura **2.14**, de forma que siempre se cumpla que: $r\,u = Cte$

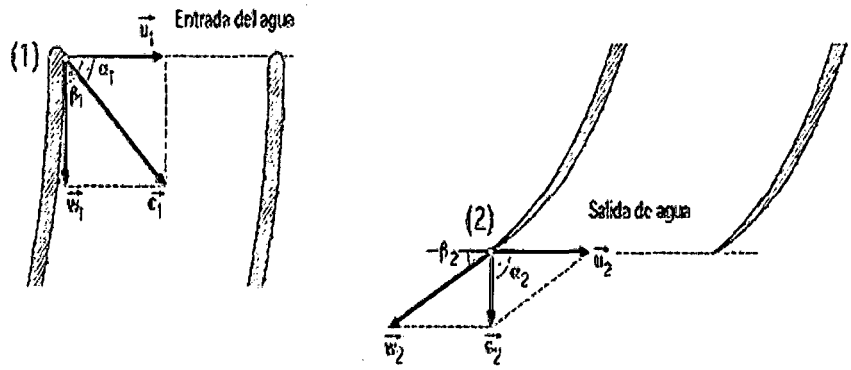


Fig. 2.14 Triángulo de velocidades

2.1.4.1 DISEÑO DE LOS DIÁMETRO PRINCIPALES DE LA TURBINA DE ACCIÓN TOTAL.

El rodete de las turbinas de acción total mostrada en la figura **2.15**, se observan que se necesitan calcular los diámetros del rodete **D_1, D_2, D_3 y Dn**.

1. **DIÁMETRO DEL TUBO DE ASPIRACIÓN (D_3):**

   Estas fórmulas empíricas son aplicables para las turbinas Kaplan, Hélice y Tubular.

$$D_3 = 1000\sqrt{\frac{4Q}{\pi C_3}} \quad (2.22)$$

En donde: D_3 = Diámetro en el tubo de aspiración, en mm

Q = Caudal en metros cúbicos por segundo (m^3/seg)

H = Altura del salto en metros (m),

C_3 = Velocidad de salida, m/seg

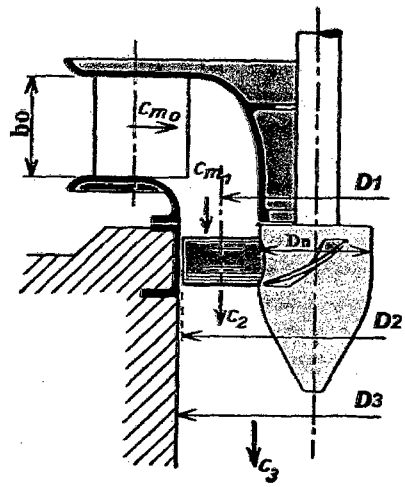


Fig. 2.15 Dimensiones principales de un rodete.

Para determinar la velocidad de salida C_3 utilizamos la siguiente ecuación:

$$C_3 = \sqrt{\frac{2 \times g \times k_c \times H}{100}} \quad (2.23)$$

En donde: C_3 = Velocidad de salida, m/seg

g = Gravedad, en m/seg^2

H = Altura del salto en metros (m),

K_c = Constante, en tanto por uno; η_t/100

Considerar para la Kaplan, Hélice y Tubular el 30%.

2. DIÁMETRO DEL RODETE (D_2)

Para calcular el diámetro externo del rodete D_1 de la turbina (Kaplan, Hélice y Tubular) mostrada en la figura **2.15**, utilizamos la siguiente relación:

$$D_2 = 0.98 D_3 \qquad (2.24)$$

En donde: D_2 = Diámetro del rodete, mm

D_3 = Diámetro del tubo de aspiración, en mm

3. DIÁMETRO DEL CUBO DEL RODETE (D_N)

Para calcular el diámetro del cubo del rodete D_N de la turbina (Kaplan, Hélice y Tubular) mostrada en la figura **2.15**, utilizamos la siguiente relación:

$$D_N = D_3(0.25 + \frac{94.64}{N_S}) \qquad (2.25)$$

En donde: D_N = Diámetro del cubo del rodete, mm

D_3 = Diámetro del tubo de aspiración, en mm

N_s = Velocidad específica, en rpm

4. DIÁMETRO MEDIO DEL RODETE (D_1)

Para calcular el diámetro medio del rodete D_1 de la turbina (Kaplan, Hélice y Tubular) mostrada en la figura **2.15**, utilizamos la siguiente relación:

$$D_1 = D_N + (\frac{D_2 - D_N}{2}) \qquad (2.26)$$

En donde: D_1 = Diámetro medio del rodete, mm

D_N = Diámetro del cubo del rodete, en mm

D_2 = Diámetro del rodete, en mm

2.1.4.2 ANCHO DE LA RUEDA DIRECTRIZ (Bo)

Si la admisión está al 80%, el gasto será de **0.8 Q** y se considera que la disminución de sección por el espesor de las paletas directrices, alcanzará al

10%; la anchura de la rueda directriz (**Bo**) se calcula según la turbina de acción total seleccionada, utilizando la siguiente ecuación:

$$B_0 = \frac{0.8\ x\ Q}{0.9\ x\ D_2\ x\ \pi\ x\ Cmo} \qquad (2.27)$$

En donde: $\mathbf{B_o}$ = Ancho de la rueda directriz, en mm
$\mathbf{D_2}$ = Diámetro del rodete, en mm
$\mathbf{C_{mo}}$ = Velocidad meridiana, en m/seg
Q = Caudal, en m^3/seg

La sección libre de salida debe ser mayor que la superficie de entrada en el rodete: $\mathbf{0.6\ c_{m1} \leq c_{mo} \leq 0.7\ c_{m1}}$

Utilizar en el diseño: $C_{mo} = 0.65 \times C_{m1} \qquad (2.28)$

En donde: $\mathbf{C_{m1}}$ = Componente de la velocidad de salida, m/seg

Para calcular esta componente utilizamos la siguiente relación:

$$C_{m1} = \frac{4 \times 0.8 \times Q \times 10^6}{(D_2^2 - D_N^2)\ \pi}\ m/seg \qquad (2.29)$$

En donde: $\mathbf{C_{m1}}$ = Componente de la velocidad de salida, en m/seg
$\mathbf{D_2}$ = Diámetro del rodete, en mm
$\mathbf{D_N}$ = Diámetro del cubo del rodete, en mm
Q = Caudal, en m^3/seg

2.1.4.3 CÁLCULO DEL NÚMERO DE ALABES (Z)

El número de alabes para la turbina de acción total seleccionada, se calcula con la siguiente ecuación:

$$Z = \frac{2170 - 1.2 \times N_s}{250} \qquad (2.30)$$

En donde: **Z** = Número de álabes

N_s = Velocidad específica, en rpm

2.1.5 MODELO TRADICIONAL EN EL DISEÑO DE LA TURBINA MICHELL

31] Quantz L, [43] Jara W., [32] ITDG Perú

La turbina de flujo transversal o Michell-Banki es una máquina utilizada principalmente para pequeños aprovechamientos hidroeléctricos. Sus ventajas principales están en su sencillo diseño y su fácil construcción lo que la hace atractiva en el balance económico de un aprovechamiento a pequeña escala. No obstante esto no impide que la turbina se utilice en grandes instalaciones. Aunque la turbina de flujo transversal se conoce como una máquina de pequeña escala, existen actualmente máquinas de este tipo de hasta 6 MW.

La turbina consta de dos elementos principales: un inyector y un rotor (figura **2.16**). El agua es restituida mediante una descarga a presión atmosférica. El rotor está compuesto por dos discos paralelos a los cuales van unidos los álabes curvados en forma de sector circular.

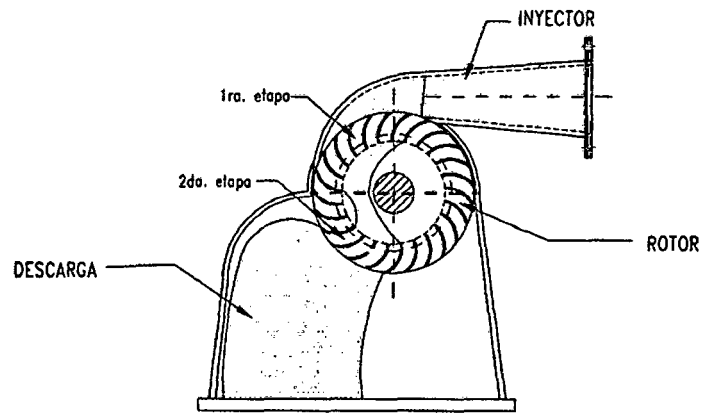


Fig. 2.16 Partes principales de una turbina Michell.

La energía del agua es transferida al rotor en dos etapas, lo que también da a esta máquina el nombre de turbina de doble efecto, y de las cuales la primera etapa entrega un promedio del 70% de la energía total transferida al rotor y la segunda alrededor del 30% restante (figura **2.17**).

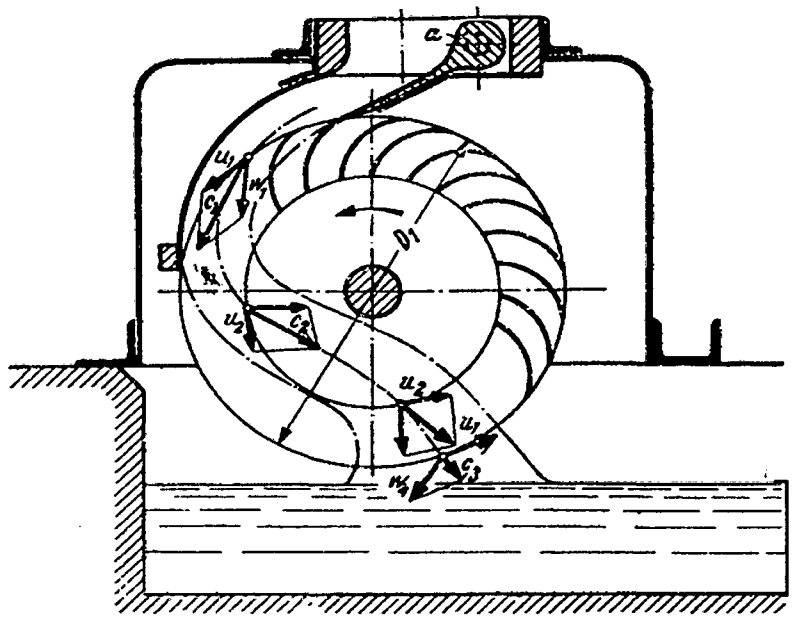


Fig. 2.17 Turbina Michell.

Los ensayos realizados por distintos investigadores sitúan el rendimiento hidráulico de esta máquina entre un 65-70%, otros autores mencionan un 61% aclarando que la segunda etapa entrega un 17%, y en general muchos autores indican un 70% hasta un 84%. Una característica atractiva de esta máquina es la forma aplanada de su curva de rendimiento. Esto se logra con un diseño de la turbina con admisión parcial.

2.1.5.1 DISEÑO DE LOS DIÁMETRO PRINCIPALES DE LA TURBINA

El rodete de la turbina Mitchell-Banki de acción total mostrada en la figura **2.18**, se observan que se necesitan calcular los diámetros del rodete D_1 y D_2.

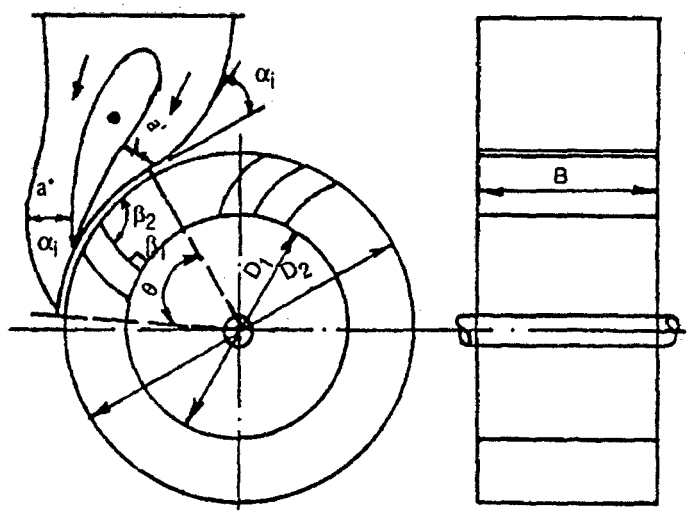


Fig. 2.18 Diámetro del rodete.

1. DIÁMETRO EXTERIOR DEL RODETE (D_2)

El diámetro exterior del rodete mostrado en la figura **2.18**, la podemos calcular utilizando la ecuación (2.31):

$$D_2 = K_m \times \frac{1000\sqrt{H}}{N} \qquad (2.31)$$

En donde: D_2 = Diámetro exterior del rodete, en mm
H = Altura del salto, en metros (m),
N = Velocidad de rotación, en rpm
K_m = Constante de la turbina Mitchell, $37 \leq K_m \leq 39$

2. DIÁMETRO INTERIOR DEL RODETE (D_1)

El diámetro interior del rodete mostrado en la figura **2.18**, la podemos calcular utilizando la ecuación (2.32):

$$D_1 = 0.66 \times D_2 \qquad (2.32)$$

3. DIÁMETRO DEL CHORRO (d_{chorro})

Suponiendo que se adopta un chorro cilíndrico, el diámetro del chorro la calculamos con la ecuación (2.33):

$$d_{chorro} = 1000 \times \sqrt{\frac{4 \times Q}{C_1 \times \pi}} \qquad (2.33)$$

En donde: D_{chorro} = Diámetro del chorro, mm
Q = Caudal, en m^3/seg
C_1 = Velocidad del chorro, en m/seg

4. VELOCIDAD DEL CHORRO (C_1)

Dada la cercanía entre el inyector y el rodete, y en el intersticio comprendido bajo el arco de admisión, existe una pequeña sobre presión despreciable y comprendida bajo el arco de admisión dado por el ángulo **θ**. La velocidad del chorro, se calcula utilizando la siguiente relación:

$$C_1 = \varphi \sqrt{2.g.H} \qquad (2.34)$$

En donde: C_1 = Velocidad del chorro, m/seg
g = Aceleración de la gravedad, 9,80665 m/seg^2
H = Altura, en m
φ = Coeficiente para la salida del agua:
$0.95 \leq \varphi \leq 0.98$

5. DIÁMETRO MEDIO DEL RODETE (D_m)

El diámetro medio del rodete mostrado en la figura **2.18**, la podemos calcular utilizando la ecuación 2.35:

$$D_m = \frac{D_2 + D_1}{2} \qquad (2.35)$$

En donde: D_m = Diámetro medio del rodete, en mm

D_2 = Diámetro exterior del rodete, en mm

D_1 = Diámetro interior del rodete, en mm

6. ESPESOR DEL CHORRO.

Para calcular el espesor del chorro utilizamos la ecuación 2.36.

$$a = k_a \times D_2 \qquad (2.36)$$

Donde: **a** = Espesor del chorro, en mm,

D_2 = Diámetro exterior del rodete, en mm

k_a = Coeficiente que depende del ángulo del inyector α_i y del ángulo de admisión **θ**.

Para α_i = 16° se pueden tomar los siguientes valores:

θ	60°	90°	120°
ka	0.1443	0.2164	0.2886

En el caso de usar una paleta directriz central, como se observa en la figura **2.16**, el espesor del chorro: **a = a´ + a´´**.

7. VELOCIDAD ANGULAR DE LA RUEDA (n).

Para calcular el número de revoluciones de la rueda, utilizamos la ecuación **2.37**.

$$n = 1000 \times (\frac{60.u_1}{D_2.\pi}) \qquad (2.37)$$

En donde: **n** = Velocidad angular de la rueda, en rpm
u_1 = Velocidad tangencial, en m/seg
D_2 = Diámetro exterior del rodete, en mm

8. VELOCIDAD TANGENCIAL (u_1).

La velocidad tangencial u_1 en el rodete se calcula con la ecuación 2.38.

$$u_1 = 2.1 \times \sqrt{H} \qquad (2.38)$$

En donde: **H** = Altura, en m
u_1 = Velocidad tangencial, en m/seg

9. ANCHO DEL RODETE (B).

Si el ángulo del inyector del rodete es α_i = 16°, este ángulo varía entre **$15° \leq \alpha_i \geq 20°$**;

El ancho del rodete (figura 2.18) la calculamos con la ecuación (2.39).

$$B = 98.8 \times \frac{Q}{D_2 \times \sqrt{H}} \times \frac{1000}{\theta} \quad (mm) \qquad (2.39)$$

En donde: **B** = Ancho del rodete, mm
Q = Caudal, m^3/seg
H = Altura, en m
D_2 = Diámetro exterior del rodete, mm
θ = Ángulo de admisión, valores de 30°,60°,120°

10. MEDIDAS Y NÚMERO DE PALETAS

El número de alabes para la turbina Michell seleccionada, se calcula con las siguientes ecuaciones:

$$Z = \frac{D_2 + 900}{50} \qquad (2.40)$$

$$Z = \frac{\pi \times D_2}{t_a} \qquad (2.41)$$

En donde: **Z** = Número de álabes
D_2 = Diámetro exterior del rodete, en mm
Ta = Paso o división exterior, en mm

Las dimensiones de la cuchara son proporcionales al diámetro del chorro, para evitar una destrucción rápida de la arista media, no puede ser más pequeño pues el agua que sale de una cuchara no debe golpear la siguiente.

$$b = 2.8 \times d \qquad (2.42a)$$
$$h = 3 \times d \qquad (2.42b)$$
$$t = 0.8 \times d \qquad (2.42c)$$

11. PASO O DIVISIÓN EXTERIOR

Escogemos ahora un paso o división exterior de:

$$t_a \cong \frac{h}{4.5} \qquad (2.43)$$

2.2 MODELO MATEMÁTICO POR EL MÉTODO DE LAS REDES NEURONALES

2.2.1 CÁLCULO DE LAS TURBINAS HIDRÁULICAS

Los datos obtenidos se tabulan en una hoja del cálculo con formato Excel (Excel 97), los valores hallados son transferidos al Datagrid del NeuroShell 2; que es el mecanismo de introducción de datos; tal como se observa en la figura **2.19**.

I2 =(10*A2*B2*D2)/76

	A	B	C	D	E	F	G	H	I	J	K
1	H_m	Q_m3_seg	Vel_Rot.n_rpm	Ef_Pelton	Ef_Francis	Ef_kaplan	Ef_Helice	Ef_Michell	P_Pelton_hp	P_Pelton_kW	P_Francis
2	300	0,05	900	70,00	80,00	90,00	85,00	65,00	138,158	100,184	157,8
3	300	0,05	1200	70,00	80,00	90,00	85,00	65,00	138,158	100,184	157,8
4	300	0,05	1800	70,00	80,00	90,00	85,00	65,00	138,158	100,184	157,8
5	300	0,07	720	70,00	80,00	90,00	85,00	65,00	193,421	140,258	221,(
6	300	0,07	1200	70,00	80,00	90,00	85,00	65,00	193,421	140,258	221,(
7	300	0,07	1800	70,00	80,00	90,00	85,00	65,00	193,421	140,258	221,(
8	300	0,1	720	70,00	80,00	90,00	85,00	65,00	276,316	200,368	315,
9	300	0,1	900	70,00	80,00	90,00	85,00	65,00	276,316	200,368	315,
10	300	0,1	1200	70,00	80,00	90,00	85,00	65,00	276,316	200,368	315,
11	300	0,2	514	70,00	80,00	90,00	85,00	65,00	552,632	400,736	631,
12	300	0,2	600	70,00	80,00	90,00	85,00	65,00	552,632	400,736	631,
13	300	0,2	720	70,00	80,00	90,00	85,00	65,00	552,632	400,736	631,
14	300	0,2	900	70,00	80,00	90,00	85,00	65,00	552,632	400,736	631,
15	300	0,2	1200	70,00	80,00	90,00	85,00	65,00	552,632	400,736	631,
16	300	0,2	1800	70,00	80,00	90,00	85,00	65,00	552,632	400,736	631,
17	300	0,4	360	70,00	80,00	90,00	85,00	65,00	1105,263	801,473	1.263,

SelTurb

Fig. 2.19 Tabulación de datos.

1. POTENCIA DE LA TURBINA (P).

La potencia de la turbina se determinan utilizando una hoja Excel (Excel 97), como se observa en la figura **2.19**; los valores calculados son transferidas al Datagrid del NeuroShell2, por ejemplo para la fila 2 utilizamos la siguiente relación:

$$P = (10*A2*B2*D2) / 76, \text{ en hp} \quad (2.44a)$$

$$P = I2 / (1,014*1,36), \text{ en kW} \quad (2.44b)$$

Cada letra en las ecuaciones **2.44** significa la columna utilizada por los datos de entrada y el número 2 se refiere a la fila; la letra **A** corresponde a la altura, la letra **B** corresponde al caudal, etc. Se utiliza una columna para cada turbina diferente, por ejemplo la columna I para la turbina Pelton, la columna k para la turbina Francis, y así sucesivamente.

2. NÚMERO ESPECÍFICO (Ns).

El número específico de revoluciones de la turbina se determina utilizando una hoja Excel (Excel 97), como se observa en la figura **2.20**; los valores calculados son transferidas al Datagrid del NeuroShell2, por ejemplo para la fila 2 utilizamos la ecuación **2.45**; en la columna **S** que corresponde al Ns de la turbina Pelton 1CH.

Ns = (C2*POTENCIA (I2; 1/2)) / (POTENCIA (A2; 5/4)) **(2.45)**

S2 =(C2*POTENCIA(I2;1/2))/(POTENCIA(A2;5/4))

	A	B	C	D	E	F	G	H	I	S	T
1	H_m	Q_m3_seg	Vel_Rot.n_rpm	Ef_Pelton	Ef_Francis	Ef_kaplan	Ef_Helice	Ef_Michell	P_Pelton_hp	Ns_Pelton_1Ch	Ns_Pelto
2	300	0,05	900	70,00	80,00	90,00	85,00	65,00	138,158	8,47	5,9
3	300	0,05	1200	70,00	80,00	90,00	85,00	65,00	138,158	11,30	7,9
4	300	0,05	1800	70,00	80,00	90,00	85,00	65,00	138,158	16,95	11,9
5	300	0,07	720	70,00	80,00	90,00	85,00	65,00	193,421	8,02	5,6
6	300	0,07	1200	70,00	80,00	90,00	85,00	65,00	193,421	13,37	9,4
7	300	0,07	1800	70,00	80,00	90,00	85,00	65,00	193,421	20,05	14,1
8	300	0,1	720	70,00	80,00	90,00	85,00	65,00	276,316	9,59	6,7
9	300	0,1	900	70,00	80,00	90,00	85,00	65,00	276,316	11,98	8,4
10	300	0,1	1200	70,00	80,00	90,00	85,00	65,00	276,316	15,98	11,2
11	300	0,2	514	70,00	80,00	90,00	85,00	65,00	552,632	9,68	6,8
12	300	0,2	600	70,00	80,00	90,00	85,00	65,00	552,632	11,30	7,9
13	300	0,2	720	70,00	80,00	90,00	85,00	65,00	552,632	13,56	9,5
14	300	0,2	900	70,00	80,00	90,00	85,00	65,00	552,632	16,95	11,9
15	300	0,2	1200	70,00	80,00	90,00	85,00	65,00	552,632	22,59	15,9
16	300	0,2	1800	70,00	80,00	90,00	85,00	65,00	552,632	33,89	23,9
17	300	0,4	360	70,00	80,00	90,00	85,00	65,00	1105,263	9,59	6,7

SelTurb

Fig. 2.20 Tabulación de datos.

2.2.2 DISEÑO DE UNA TURBINA PELTON

2.2.2.1 CÁLCULO DE LAS DIMENSIONES PRINCIPALES.

El rodete de una turbina Pelton mostrado en la figura **2.3**, se observa que se necesita calcular el diámetro del chorro (d_{ch}), diámetro del rodete (D_2) y el diámetro externo (D_3).

1. Diámetro del chorro (d_{ch})

Utilizando la ecuación **2.46** calculamos el diámetro del chorro de una turbina Pelton, como se observa en la figura **2.21**; los valores calculados y tabulada en una hoja de cálculo son transferidas al Datagrid del NeuroShell2, por ejemplo para la fila **2** y en la columna **P**, calculamos el d_{ch} de la turbina Pelton **2CH**.

$$d_{ch} = (0{,}55*1000)*RAIZ((B2/(2*RAIZ(A2)))) \qquad (2.46)$$

P2 | =(0,55*1000)*RAIZ((B2/(2*RAIZ(A2))))

	A	B	C	D	E	F	G	O	P	Q
1	H_m	Q_m3_seg	Vel_Rot.n_rpm	Ef_Pelton	Cs_Pelton	Kdr_Pelton	Kch_Pelton	dch_Pelton_1ch	dch_Pelton_2ch	dch_Pelt
2	300	0,05	900	70,00	0,97	37,00	14,00	29,55	20,90	14,
3	300	0,05	1200	70,00	0,97	37,00	14,00	29,55	20,90	14,
4	300	0,05	1800	70,00	0,97	37,00	14,00	29,55	20,90	14,
5	300	0,07	720	70,00	0,97	37,00	14,00	34,96	24,72	17,
6	300	0,07	1200	70,00	0,97	37,00	14,00	34,96	24,72	17,
7	300	0,07	1800	70,00	0,97	37,00	14,00	34,96	24,72	17,
8	300	0,1	720	70,00	0,97	37,00	14,00	41,79	29,55	20,
9	300	0,1	900	70,00	0,97	37,00	14,00	41,79	29,55	20,
10	300	0,1	1200	70,00	0,97	37,00	14,00	41,79	29,55	20,
11	300	0,2	514	70,00	0,97	37,00	14,00	59,10	41,79	29,
12	300	0,2	600	70,00	0,97	37,00	14,00	59,10	41,79	29,
13	300	0,2	720	70,00	0,97	37,00	14,00	59,10	41,79	29,
14	300	0,2	900	70,00	0,97	37,00	14,00	59,10	41,79	29,
15	300	0,2	1200	70,00	0,97	37,00	14,00	59,10	41,79	29,
16	300	0,2	1800	70,00	0,97	37,00	14,00	59,10	41,79	29,
17	300	0,4	360	70,00	0,97	37,00	14,00	83,58	59,10	41

Pelton

Fig. 2.21 Cálculo del diámetro del chorro.

2. Diámetro medio del Rodete (D_2 ó D_r)

Utilizando la ecuación **2.47** calculamos el diámetro medio del rodete de una turbina Pelton, como se observa en la figura **2.22**; los valores calculados y tabulada en una hoja de cálculo son transferidas al Datagrid

del NeuroShell2, por ejemplo para la fila 2 y en la columna **R**, calculamos el D_r de la turbina Pelton.

$$D_r = ((F2*1000)*RAIZ(A2))/C2 \qquad (2.47)$$

R2 =((F2*1000)*RAIZ(A2))/C2

	A	B	C	D	E	F	G	R	S	
1	H_m	Q_m3_seg	Vel_Rot.n_rpm	Ef_Pelton	Cs_Pelton	Kdr_Pelton	Kcb_Pelton	Dr_Pelton_mm	Vt_Pelton_m_s	Vr_Pel
2	300	0,05	900	70,00	0,97	37,00	14,00	712,07	37,190	997
3	300	0,05	1200	70,00	0,97	37,00	14,00	534,05	37,190	132
4	300	0,05	1800	70,00	0,97	37,00	14,00	356,03	37,190	199
5	300	0,07	720	70,00	0,97	37,00	14,00	890,08	37,190	797
6	300	0,07	1200	70,00	0,97	37,00	14,00	534,05	37,190	132
7	300	0,07	1800	70,00	0,97	37,00	14,00	356,03	37,190	199
8	300	0,1	720	70,00	0,97	37,00	14,00	890,08	37,190	797
9	300	0,1	900	70,00	0,97	37,00	14,00	712,07	37,190	997
10	300	0,1	1200	70,00	0,97	37,00	14,00	534,05	37,190	132
11	300	0,2	514	70,00	0,97	37,00	14,00	1246,81	37,190	569
12	300	0,2	600	70,00	0,97	37,00	14,00	1068,10	37,190	664
13	300	0,2	720	70,00	0,97	37,00	14,00	890,08	37,190	797
14	300	0,2	900	70,00	0,97	37,00	14,00	712,07	37,190	997
15	300	0,2	1200	70,00	0,97	37,00	14,00	534,05	37,190	132
16	300	0,2	1800	70,00	0,97	37,00	14,00	356,03	37,190	199
17	300	0,4	360	70,00	0,97	37,00	14,00	1780,16	37,190	398

Pelton

Fig. 2.22 Cálculo del diámetro medio del rodete.

3. Diámetro externo del Rodete (D_3)

$$D3=R2+2*((3*V2)/5) \qquad (2.48)$$

4. DIMENSIONES DE LAS PALETAS

Ancho de la paleta: $b = 3,75*O2$ **(2.49)**

Altura de la paleta: $h = 3,5*O2$ **(2.50)**

Espesor de la paleta: $t=1,5*O2$ **(2.51)**

5. NÚMERO DE LAS PALETAS

El número de cucharas: $Z = 0,5*(R2/O2)+G2$ **(2.52)**

6. SELECCIÓN DE LA TURBINA PELTON

P1ch = SI (Y (A2=300; B2>=0,042; B2<=0,4); 10; 0) (2.53)

2.2.3 DISEÑO DE UNA TURBINA FRANCIS

2.2.3.1 CÁLCULO DE LOS DIÁMETRO PRINCIPALES.

1. DIÁMETRO DEL TUBO DE ASPIRACIÓN (D_3)

$$D_3 = 1000 * RAIZ ((4*B2) / (PI ()*M2)) \quad (2.54)$$

En donde: D_3 = Diámetro en el tubo de aspiración, en mm
B_2 = Columna B y la fila 2, caudal en (m^3/seg)
M_2 = Columna M y la fila 2, velocidad de salida, m/seg

Para determinar la velocidad de salida C_3 utilizamos la siguiente ecuación:

$$C_3 = RAIZ ((2*9,8*E2*A2)/100) \quad (2.55)$$

En donde: C_3 = Velocidad de salida, m/seg
g = Gravedad, 9.8 en m/seg^2
A_2 = Columna A y la fila 2, altura del salto en metros (m),
E_2 = Columna E y la fila 2, constante según la turbina. Considerar 4% para Francis Lenta, 6% para Francis Normal, 12% para Francis Rápida y 25% para la Francis Extra rápida.

2. DIÁMETRO EXTERNO DEL RODETE (D_1)

Para calcular el diámetro externo del rodete D_1 de la turbina mostrada en la figura **2.10**, utilizamos la siguiente relación:

$$D_1 = Q2*(0,4+(94,5/K2)) \qquad (2.56)$$

En donde: D_1 = Diámetro externo del rodete, mm

D_3 = Diámetro del tubo de aspiración, columna Q, R, S, T, fila 2; en mm

N_s = Velocidad específica, columna K, fila 2; en rpm

3. DIÁMETRO INTERNO DEL RODETE (D_2)

Para calcular el diámetro interno del rodete D_2 de la turbina mostrada en la figura **2.10**, utilizamos la siguiente relación:

$$D_2 = Q2*(0,96+0,00038*K2) \qquad (2.57)$$

En donde: D_2 = Diámetro interno del rodete, mm

D_3 = Diámetro del tubo de aspiración, columna Q, R, S y T, fila 2; en mm

N_s = Velocidad específica, columna K, fila 2; en rpm

4. ANCHO DE LA CORONA DIRECTRIZ (bo)

El ancho de la corona directriz se calcula según la turbina Francis seleccionada, utilizando las siguientes ecuaciones:

$$bo = ABS(Y2*(((K2-60)*0,07)/65+0,08)) \qquad (2.58)$$

En donde: b_o = Ancho de la corona directriz, en mm

D_2 = Diámetro interno del rodete, en mm

N_s = Velocidad específica, en rpm

5. CÁLCULO DEL NÚMERO DE ÁLABES (Z_o)

El número de alabes para una turbina Francis seleccionada, se calcula con las siguientes ecuaciones:

a. FRANCIS LENTA (F_L)

$$Z_{FL} = 18-((K2-60)*1/65) \quad (2.59)$$

En donde: Z_{FL} = Número de alabes para una turbina Francis lenta

N_s = Velocidad específica; columna K, fila 2, en rpm

b. FRANCIS NORMAL (F_N)

$$Z_{FN} = 17-(2*(K2-125)/100) \quad (2.60)$$

En donde: Z_{FN} = Número de alabes para una turbina Francis normal

N_s = Velocidad específica; columna K, fila 2, en rpm

c. FRANCIS RÁPIDA (F_R)

$$Z_{FR} = 15-(2*(K2-225)/125) \quad (2.61)$$

En donde: Z_{FR} = Número de alabes para una turbina Francis rápida

N_s = Velocidad específica; columna K, fila 2, en rpm

d. FRANCIS EXTRA RÁPIDA (F_{ER})

$$Z_{FER} = 13-(1*(K2-350)/100) \quad (2.62)$$

En donde: Z_{FER} = Número de alabes para una turbina Francis extra rápida

N_s = Velocidad específica; columna k, fila 2, en rpm

6. SELECCIÓN DE LA TURBINA FRANCIS

FL = SI (Y (A2=200; B2>=1,4; B2<=2,5; J2>=1800; J2<=4000; K2>=60; K2<=125); 10; 0) **(2.63)**

2.2.4 DISEÑO DE UNA TURBINA DE ACCIÓN TOTAL

2.2.4.1 CÁLCULO DE LAS DIMENSIONES PRINCIPALES.

El rodete de las turbinas de acción total mostrada en la figura **2.15**, se observan que se necesitan calcular los diámetros del rodete **D_1, D_2, D_3 y Dn**.

1. DIÁMETRO DEL TUBO DE ASPIRACIÓN (D_3):

$$D_3 = 1000*\text{RAIZ}((4*B2)/(\text{PI}()*K2)) \quad (2.64)$$

En donde: **D_3** = Diámetro en el tubo de aspiración, en mm
B2 = Caudal, Columna B, fila 2; m^3/seg
K2 = Velocidad de salida; columna K, fila 2, en m/seg

Para determinar la velocidad de salida **C_3** (K2) utilizamos la siguiente ecuación:

$$C_3 = \text{RAIZ}(2*9{,}8*0{,}3*A2) \quad (2.65)$$

En donde: **C_3** = Velocidad de salida, (K2) m/seg
g = Gravedad, (9.8) en m/seg^2
A2 = Altura del salto en metros (m),
K_c = Constante, 0.3

2. DIÁMETRO DEL RODETE (D_2)

Para calcular el diámetro externo del rodete **D_1** de la turbina (Kaplan, Hélice y Tubular) mostrada en la figura **2.15**, utilizamos la siguiente relación:

$$D_2 = 0,98*\mathrm{L2} \qquad (2.66)$$

En donde: D_2 = Diámetro del rodete, mm

D_3 = Diámetro del tubo de aspiración (L2), columna L y la fila 2; en mm

3. DIÁMETRO DEL CUBO DEL RODETE (D_N)

$$D_N = \mathrm{L2}*((0,25+(94,64/\mathrm{I2}))) \qquad (2.67)$$

En donde: D_N = Diámetro del cubo del rodete, mm

D_3 = Diámetro del tubo de aspiración (L2), columna L y la fila 2, en mm

N_s = Velocidad específica (I2), en rpm

4. DIÁMETRO MEDIO DEL RODETE (D_1)

Para calcular el diámetro medio del rodete D_1 de la turbina (Kaplan, Hélice y Tubular) mostrada en la figura **2.15**, utilizamos la siguiente relación:

$$D_1 = \mathrm{N2}+(\mathrm{M2}-\mathrm{N2})/2 \qquad (2.68)$$

En donde: D_1 = Diámetro medio del rodete, mm

D_N = Diámetro del cubo del rodete (N2), columna N y la fila 2, en mm

D_2 = Diámetro del rodete (D2), columna M y la fila 2, en mm

5. ANCHO DE LA RUEDA DIRECTRIZ (Bo)

El ancho de la corona directriz se calcula utilizando la siguiente ecuación:

$$B_0 = (0,8*\mathrm{B2})/(0,9*\mathrm{M2}*\mathrm{PI}()*\mathrm{V2})*10\text{^}6 \qquad (2.69)$$

En donde: B_o = Ancho de la rueda directriz, en mm

D_2 = Diámetro del rodete (M2), columna M y la fila 2, en mm

C_{mo} = Velocidad meridiana, columna V y la fila 2, en m/seg

Q = Caudal, columna B y la fila 2, en m^3/seg

6. VELOCIDAD MERIDIANA (Cmo)

La velocidad meridiana se calcula utilizando la siguiente ecuación:

$$C_{mo} = 0{,}65*(\text{ABS}((4*0{,}8*\text{B2})/((\text{M2}\text{\^{}}2-\text{N2}\text{\^{}}2)*\text{PI}())*10\text{\^{}}6)) \quad (2.70)$$

En donde: C_{mo} = Velocidad meridiana, en mm

D_2 = Diámetro del rodete (M2), columna M y la fila 2, en mm

D_N = Diámetro del cubo del rodete (N2), columna N y la fila 2, en mm

Q = Caudal, columna B y la fila 2, en m^3/seg

2.2.5 ALGORITMO DE APRENDIZAJE BACKPROPAGATION.

1. Regla de Aprendizaje.

El algoritmo Backpropagation para redes multicapa es una generalización del algoritmo LMS, ambos algoritmos realizan su labor de actualización de pesos y ganancias con base en el error medio cuadrático. La red Backpropagation trabaja bajo aprendizaje supervisado y por tanto necesita un set de entrenamiento que le describa cada salida y su valor de salida esperado de la siguiente forma:

$$\{p_1,t_1\},\{p_2,t_2\},.....,\{p_Q,t_Q\} \quad (2.71)$$

Donde: $\mathbf{p_q}$ = es una entrada a la red

t_Q = es la correspondiente salida deseada para el patrón q-ésimo.

El algoritmo debe ajustar los parámetros de la red para minimizar el error medio cuadrático.

El entrenamiento de una red neuronal multicapa se realiza mediante un proceso de aprendizaje, para realizar este proceso se debe inicialmente tener definida la topología de la red esto es: número de neuronas en la capa de entrada el cual depende del número de componentes del vector de entrada, cantidad de capas ocultas y número de neuronas de cada una de ellas, número de neuronas en la capa de la salida el cual depende del número de componentes del vector de salida o patrones objetivo y funciones de transferencia requeridas en cada capa, con base en la topología escogida se asignan valores iniciales a cada uno de los parámetros que conforma la red.

La regla de entrenamiento es la siguiente:

1. Aprende los $\mathbf{w_i}$ para una red neuronal multicapa, con funciones de activación derivables.
2. Función de error:

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in D} (t_{kd} - O_{kd})^2 \qquad (2.72)$$

Donde D es el conjunto de ejemplos y O el conjunto de nodos de salida.

t_{kd} , será la salida esperada para el ejemplo en el nodo,

O_{kd}, la salida obtenida para el nodo y el ejemplo.

3. Convergencia a mínimos locales

2. Deducción Matemática.

Se desea entrenar una red para que reconozca potencia de la turbina, números específico, y selección del tipo de turbina. La deducción matemática de este procedimiento se realizará para una red con una capa de entrada, una capa oculta y una capa de salida y luego se generalizará para redes que tengan más de una capa oculta (figura **2.23**)

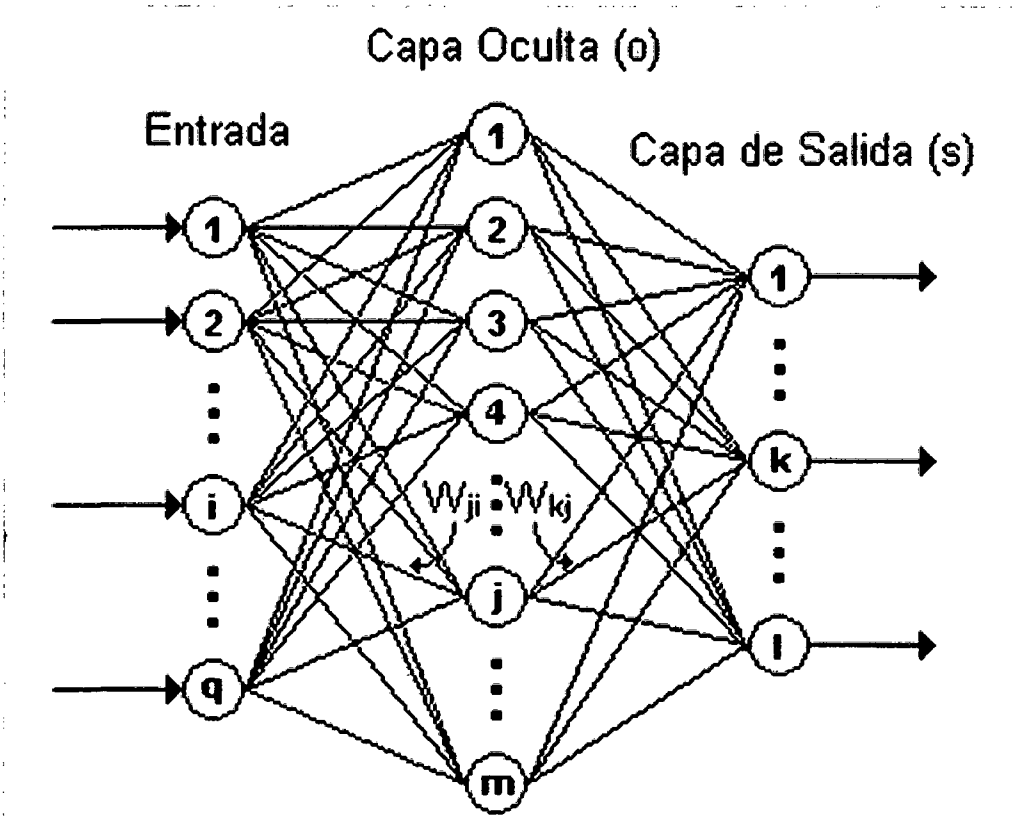


Fig. 2.23 Disposición de una red de tres capas.

En la figura **2.23**:

q : Equivale al número de componentes el vector de entrada.

m : Número de neuronas de la capa oculta

l : Número de neuronas de la capa de salida

En la tabla 2.1 se muestran los valores de entrada y las salidas deseadas.

Tabla 2.1 Valores de entrada y salida

ENTRADAS					SALIDAS DESEADAS			
X_1	X_2	X_3	X_4	X_5				
H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Tpelton	Coe. Sal.P:(j)	Pot.Pelton (hp)	Pot.Pelton (kW)	Ns Pel(1Ch)	Pelton 1ch
300	0.05	1200	70.00	0,97	138.158	100.184	11.30	10
300	0.1	900	70.00	0,97	276.316	200.368	11.98	10
300	0.2	720	70.00	0,97	552.632	400.736	13.56	10
200	0.04	1200	70,00	0,97	73.684	53.432	13.70	10
200	0.07	1200	70,00	0,97	128.947	93.505	18.12	10
200	0.1	720	70,00	0,97	184.211	133.579	12.99	10
150	0.04	900	70,00	0,97	55.263	40.074	12.75	10
150	0.06	720	70,00	0,97	82.895	60.110	12.49	10
150	0.15	450	70,00	0,97	207.237	150.276	12.34	10
100	0.06	600	70,00	0,97	55.263	40.074	14.10	10
100	0.08	450	70,00	0,97	73.684	53.432	12.22	10
90	0.06	514	70.00	0.97	49.737	36.066	13.08	10

Para iniciar el entrenamiento se le presenta a la red un patrón de entrenamiento, el cual tiene *q* componentes como se describe en la ecuación (2.73).

$$p = \begin{bmatrix} p_1 \\ p_2 \\ . \\ . \\ . \\ p_i \\ . \\ p_q \end{bmatrix} \quad (2.73)$$

Cuando se le presenta a la red una patrón de entrenamiento, este se propaga a través de las conexiones existentes produciendo una entrada neta ***n*** en cada una las neuronas de la siguiente capa, la entrada neta a la neurona ***j*** de la siguiente capa debido a la presencia de un patrón de entrenamiento en la entrada está dada por la ecuación (2.74), nótese que la entrada neta es el valor justo antes de pasar por la función de transferencia.

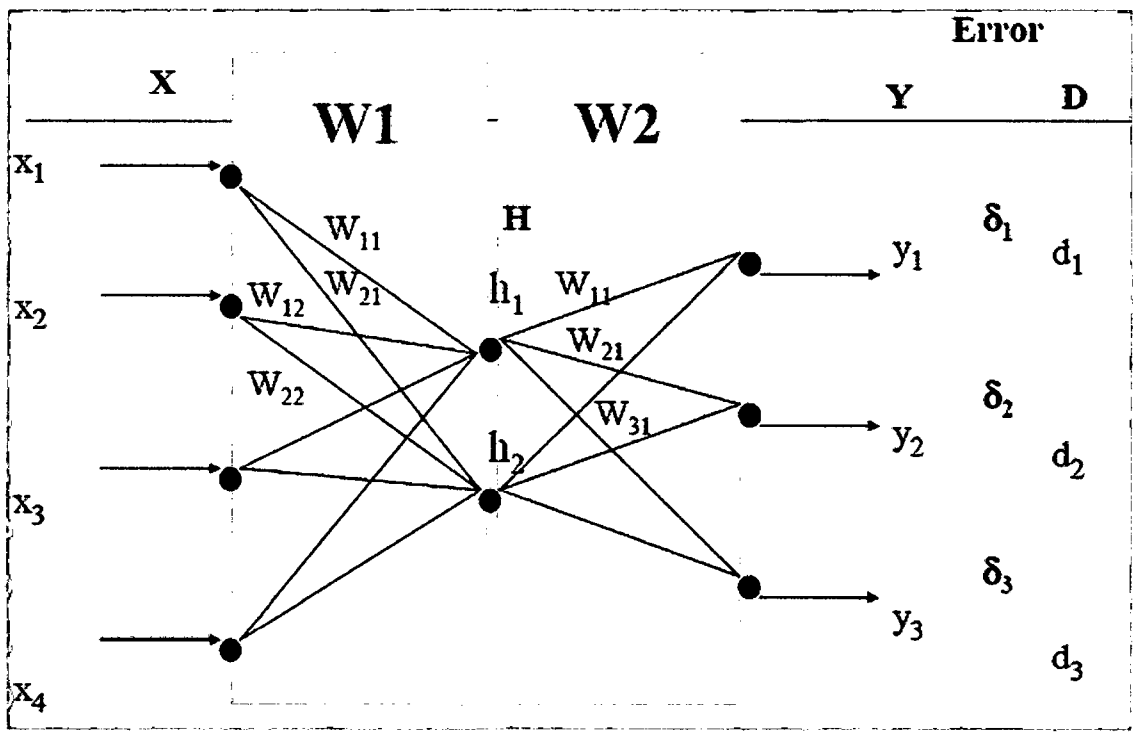


Fig. 2.24 Patrón de entrenamiento de la red.

$$n_j^o = \sum_{i=1}^{q} W_{ji}^o p_i + b_j^o \qquad (2.74)$$

W_{ji}^o : Peso que une la componente ***i*** de la entrada con la neurona ***j*** de la capa oculta

p_i : *Componente* ***i*** del vector ***p*** que contiene el patrón de entrenamiento de ***q*** componentes

b^o_j : Ganancia de la neurona ***j*** de la capa oculta

Donde el superíndice (°) representa la capa a la que pertenece cada parámetro, es este caso la capa oculta. Cada una de las neuronas de la capa oculta tiene como salida a^o_j que está dada por la ecuación (2.75).

$$a^o_j = f^o\left[\sum_{i=1}^{q} W^o_{ji} p_i + b^o_j\right] \quad (2.75)$$

f^o : Función de transferencia de las neuronas de la capa oculta

Las salidas a^o_j de las neuronas de la capa oculta (de las componentes) son las entradas a los pesos de conexión de la capa de salida, este comportamiento esta descrito por la ecuación (2.74)

$$n^s_k = \sum_{j=1}^{m} W^s_{kj} a^o_j + b^s_k \quad (2.76)$$

W^s_{kj} : Peso que une la neurona ***j*** de la capa oculta con la neurona ***k*** de la capa de salida, la cual cuenta con ***s*** neuronas.

a^o_j : Salida de la neurona ***j*** de la capa oculta, la cual cuenta con ***m*** neuronas.

b^s_k : Ganancia de la neurona *k* de la capa de salida.

n^s_k : Entrada neta a la neurona *k* de la capa de salida.

La red produce una salida final descrita por la ecuación (2.77)

$$a^s_k = f^s(n^s_k) \quad (2.77)$$

f^s : Función de transferencia de las neuronas de la capa de salida.

Reemplazando (2.74) en (2.75) se obtiene la salida de la red en función de la entrada neta y de los pesos de conexión con la última capa oculta.

$$a_k^s = f^s\left(\sum_{j=1}^{m} W_{kj}^s a_j^o + b_k^s\right) \quad (2.78)$$

La salida de la red de cada neurona a^s_k se compara con la salida deseada t_k para calcular el error en cada unidad de salida (2.79)

$$\delta_k = (t_k - a_k^s) \quad (2.79)$$

El error debido a cada patrón *p* propagado esta dado por (2.81)

$$ep^2 = \frac{1}{2}\sum_{k=1}^{s}(\delta_k)^2 \quad (2.80)$$

ep^2 : Error medio cuadrático para cada patrón de entrada *p*

δ_k : Error en la neurona ***k*** de la capa de salida con ***l*** neuronas

Este proceso se repite para el número total de patrones de entrenamiento (***r***), para un proceso de aprendizaje exitoso; el objetivo del algoritmo es actualizar todos los pesos y ganancias de la red minimizando el error medio cuadrático total descrito en (2.81).

$$e^2 = \sum_{p=1}^{r} ep^2 \quad (2.81)$$

e^2 : Error total en el proceso de aprendizaje en una iteración luego de haber presentado a la red los ***r*** patrones de entrenamiento

El error que genera una red neuronal en función de sus pesos, genera un espacio de **n** dimensiones, donde **n** es el número de pesos de conexión de la red, al evaluar el gradiente del error en un punto de esta superficie se obtendrá la dirección en la cual la función del error tendrá un mayor crecimiento, como el objetivo del proceso de aprendizaje es minimizar el error debe tomarse la dirección negativa del gradiente para obtener el mayor decremento del error y de esta forma su minimización, condición requerida para realizar la actualización de la matriz de pesos en el algoritmo Backpropagation.

$$W_{k+1} = W_k - \alpha \nabla ep^2 \qquad (2.82)$$

El gradiente negativo de ep^2 se denotara como $-\nabla ep^2$ y se calcula como la derivada del error respecto a todos los pesos de la red

En la capa de salida el gradiente negativo del error con respecto a los pesos es:

$$-\frac{\partial ep^2}{\partial W_{kj}^s} = -\frac{\partial}{\partial W_{kj}^s}\left(\frac{1}{2}\sum_{k=1}^{l}(t_k - a_k^s)^2\right)$$

$$-\frac{\partial ep^2}{\partial W_{kj}^s} = (t_k - a_k^s)^2 \times \frac{\partial a_k^s}{\partial W_{kj}^s} \qquad (2.83)$$

$\frac{\partial ep^2}{\partial W_{kj}^s}$: Componente del gradiente $-\nabla ep^2$ respecto al peso de la conexión de la neurona de la capa de salida y la neurona j de la capa oculta W_{kj}^s

$\frac{\partial a_k^s}{\partial W_{kj}^s}$: Derivada de la salida de la neurona *k* de la capa de salida respecto, al peso W_{kj}^s

Para calcular $\frac{\partial a_k^s}{\partial W_{kj}^s}$ se debe utilizar la regla de la cadena, pues el error no es una función explícita de los pesos de la red, de la ecuación (2.75) puede verse que la salida de la red $\boldsymbol{a^s_k}$ esta explícitamente en función de $\boldsymbol{n^s_k}$ y de la ecuación (2.76) puede verse que $\boldsymbol{n^s_k}$ esta explícitamente en función de W_{kj}^s considerando esto se genera la ecuación (2.84)

$$\frac{\partial a_k^s}{\partial W_{kj}^s} = \frac{\partial a_k^s}{\partial n_k^s} \times \frac{\partial n_k^s}{\partial W_{kj}^s} \qquad (2.84)$$

Tomando la ecuación (2.83) y reemplazándola en la ecuación (2.84) se obtiene,

$$-\frac{\partial ep^2}{\partial W_{kj}^s} = (t_k - a_k^s) \times \frac{\partial a_k^s}{\partial n_k^s} \times \frac{\partial n_k^s}{\partial W_{kj}^s} \qquad (2.85)$$

$\frac{\partial n_k^s}{\partial W_{kj}^s}$: Derivada de la entrada neta a la neurona ***k*** de la capa de salida respecto a los pesos de la conexión entre las neuronas de la capa oculta y la capa de salida.

$\frac{\partial a_k^s}{\partial n_k^s}$: Derivada de la salida de la neurona *k* de la capa de salida respecto a su entrada neta.

Reemplazando en la ecuación (2.85) las derivadas de las ecuaciones (2.76) y (2.77) se obtiene:

$$-\frac{\partial ep^2}{\partial W_{kj}^s} = (t_k - a_k^s) \times f'^s(n_k^s) \times a_j^o \qquad (2.86)$$

Como se observa en la ecuación (2.86) las funciones de transferencia utilizadas en este tipo de red deben ser continuas para que su derivada exista en todo el intervalo, ya que el término $f'^s(n^s_k)$ es requerido para el cálculo del error.

Las funciones de transferencia f más utilizadas y sus respectivas derivadas son las siguientes:

$$\log sig: \quad f(n) = \frac{1}{1+e^{-n}} \qquad (2.87)$$

$$f'(n) = f(n)(1-f(n)) \qquad (2.88)$$

$$f'(n) = a(1-a) \qquad (2.89)$$

$$\tan sig: \quad f(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}} \qquad (2.90)$$

$$f'(n) = 1-(f(n))^2 \qquad (2.91)$$

$$f'(n) = (1-a^2) \qquad (2.92)$$

$$purelin: \quad f(n) = n \qquad (2.93)$$

$$f'(n) = 1 \qquad (2.94)$$

De la ecuación (2.86), los términos del error para las neuronas de la capa de salida están dados por la ecuación (2.95), la cual se le denomina comúnmente sensitividad de la capa de salida.

$$\partial_k^s = (t_k - a_k^s) \times f'^s(n_k^s) \qquad (2.95)$$

Este algoritmo se denomina Backpropagation o de propagación inversa debido a que el error se propaga de manera inversa al funcionamiento normal de la red, de esta forma, el algoritmo encuentra el error en el proceso de aprendizaje desde las capas más internas hasta llegar a la entrada; con base en el cálculo de este error se actualizan los pesos y ganancias de cada capa.

Después de conocer (2.95) se procede a encontrar el error en la capa oculta el cual está dado por:

$$-\frac{\partial ep^2}{\partial W_{ji}^o} = -\frac{\partial}{\partial W_{ji}^o}\left(\frac{1}{2}\sum_{k=1}^{l}(t_k - a_k^s)^2\right)$$

$$-\frac{\partial ep^2}{\partial W_{ji}^s} = \sum_{k=1}^{l}(t_k - a_k^s) \times \frac{\partial a_k^s}{\partial W_{ji}^s} \qquad (2.96)$$

Para calcular el último término de la ecuación (2.96) se debe aplicar la regla de la cadena en varias ocasiones como se observa en la ecuación

(2.97) puesto que la salida de la red no es una función explícita de los pesos de la conexión entre la capa de entrada y la capa oculta.

$$\frac{\partial a_k^s}{\partial W_{ji}^o} = \frac{\partial a_k^s}{\partial n_k^s} \times \frac{\partial n_k^s}{\partial a_k^o} \times \frac{\partial a_k^o}{\partial n_j^o} \times \frac{\partial n_j^o}{\partial W_{kj}^s} \quad (2.97)$$

Todos los términos de la ecuación (2.98) son derivados respecto a variables de las que dependan explícitamente, reemplazando (2.97) en (2.96) tenemos:

$$-\frac{\partial ep^2}{\partial W_{ji}^o} = \sum_{k=1}^{l} (t_k - a_k^s) \times \frac{\partial a_k^s}{\partial n_k^s} \times \frac{\partial n_k^s}{\partial a_k^o} \times \frac{\partial a_k^o}{\partial n_j^o} \times \frac{\partial n_j^o}{\partial W_{kj}^s} \quad (2.99)$$

Tomando las derivas de las ecuaciones (2.74), (2.75), (2.76) y (2.77), y reemplazándolas en la ecuación (2.99) se obtiene la expresión del gradiente del error en la capa oculta.

$$-\frac{\partial ep^2}{\partial W_{ji}^o} = \sum_{k=1}^{l} (t_k - a_k^s) \times f'^s(n_k^s) \times W_{kj}^s \times f'^o(n_j^o) \times p_i \quad (2.100)$$

Reemplazando las ecuaciones (2.95), (2.96) y (2.97) en la ecuación (2.100) se tiene:

$$-\frac{\partial ep^2}{\partial W_{ji}^o} = \sum_{k=1}^{l} \partial_k^s \times W_{kj}^s \times f'^o(n_j^o) \times p_i \quad (2.101)$$

Los términos del error para cada neurona de la capa oculta esta dado por la ecuación (2.102), este término también se denomina sensitividad de la capa oculta.

$$\partial_j^o = f'^o(n_j^o) \times \sum_{k=1}^{l} \partial_k^s W_{kj}^s \quad (2.102)$$

Luego de encontrar el valor del gradiente del error se procede a actualizar los pesos de todas las capas empezando por la de salida, para la capa de salida la actualización de pesos y ganancias está dada por las ecuaciones (2.101) y (2.102).

$$W_{kj}(t+1) = W_{kj}(t) - 2\alpha\partial_k^s \quad (2.101)$$

$$b_k(t+1) = b_k(t) - 2\alpha\partial_k^s \quad (2.102)$$

α : Rata de aprendizaje que varía entre 0 y 1 dependiendo de las características del problema a solucionar.

Luego de actualizar los pesos y ganancias de la capa de salida se procede a actualizar los pesos y ganancias de la capa oculta mediante las ecuaciones (2.105) y (2.106).

$$W_{ji}(t+1) = W_{ji}(t) - 2\alpha\partial_j^o p_i \quad (2.105)$$

$$b_j(t+1) = b_j(t) - 2\alpha\partial_j^o \quad (2.106)$$

Esta deducción fue realizada para una red de tres capas, si se requiere realizar el análisis para una red con dos o más capas ocultas, las expresiones pueden derivarse de la ecuación (2.100) donde los términos que se encuentran dentro de la sumatoria pertenecen a la capa inmediatamente superior, este algoritmo es conocido como la regla Delta Generalizada desarrollada por Rumelhart D, la cual es una extensión de la regla delta desarrollada por Widrow en 1930

CAPÍTULO 3

DISEÑO DE LAS DIMENSIONES PRINCIPALES DE LAS TURBINAS

3.1 DISEÑO DE LAS DIMENSIONES DE UNA TURBINA POR EL MÉTODO TRADICIONAL.

3.1.1 DISEÑO DE UNA TURBINA PELTON

3.1.1.1 SELECCIÓN DE UNA TURBINA PELTON

Para un salto de **H = 120 m** y con un caudal de **Q = 0.15 m^3/seg**, proyectar una turbina Pelton o rueda tangencial, mostrada en la figura **3.1** y determinar las principales dimensiones de la turbina.

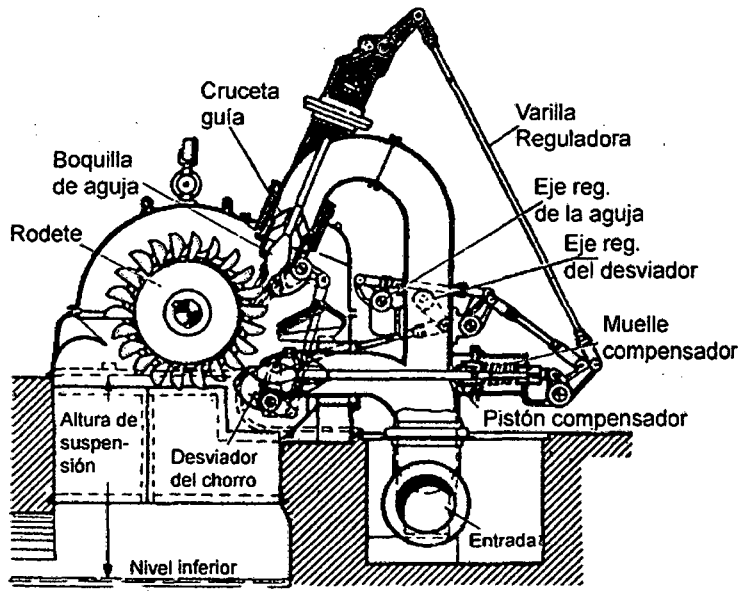


Fig. 3.1: Turbina Pelton.

1. POTENCIA DE LA TURBINA (P).

Datos: **Q = 0.15 m^3/seg, H = 120 m, η = 0.85**

Reemplazando valores en la ecuación (2.1)

$$P = \frac{1000\ x\ 0.15\ x120\ x\ 0.85}{76}$$

$$P = 201.32\,HP$$

Para calcular la potencia en KW, reemplazando valores en la ecuación (2.2):

$$P_{kW} = \frac{1000 \times 0.15 \times 120 \times 0.85}{1.340 \times 76}$$

$$P = 150.23\,kW$$

La potencia obtenida de 150.23kW corresponde al rango de una mini central, según la **figura 1.2** página **22.**

2. NÚMERO ESPECÍFICO DE REVOLUCIONES DE POTENCIA (Ns).

Para calcular el número específico de revoluciones, utilizamos la ecuación (2.3):

Datos: P = 201.32 HP, H = 120 m, i= 2 boquillas

Reemplazando valores en la ecuación (2.3):

$$N_s = \frac{n\sqrt{\frac{201.32}{i}}}{120\ x\ \sqrt[4]{120}};$$

$$\text{Si i} = 1, \quad N_s = 0.03572 \times n \quad (3.1a)$$

$$\text{Si i} = 2, \quad N_s = 0.02526 \times n \quad (3.1b)$$

$$\text{Si i} = 4, \quad N_s = 0.01786 \times n \quad (3.1c)$$

Los resultados obtenidos al aplicar la ecuación (2.3), se muestran en la ecuación 3.1; los valores hallados de esta ecuación han sido tabulados en tabla **Nº 3.1**, de estos valores seleccionamos la turbina Pelton de dos boquillas, que se encuentre dentro de los rangos establecidos.

Tabla 3.1 Selección de una turbina Pelton

Altura	Caudal	Tipo Turbina		Efic. Turb.	Pot. Turb.	Pot. Turb.	Vel. Rot.	Vel. Espcif.
H	Q	Nombre	Clase	η	P	P	n	Ns
m	m^3/seg				HP	kW	rpm	rpm
120	0.15	Pelton	1 Ch	0,85	201,32	150,23	1800	64,30
120	0.15	Pelton	2 Ch	0,85	201,32	150,23	1800	45,47
120	0.15	Pelton	4 Ch	0,85	201,32	150,23	1800	32,15

En la figura **3.2**, buscamos en el eje de las abscisas el caudal de **0.15m^3/seg**, y en el eje de las ordenadas la altura de **120metros**, de estos puntos trazamos líneas vertical y horizontal que se cortan en el campo **(2)**, que corresponde a una Turbina de Pelton de dos boquillas, cuya potencia corresponde a **140KW**, en los cálculos hallamos **150.23KW**, la condición:

$P_{calculado} > = < P_{Gráfico}$;

por lo tanto la selección es correcta.

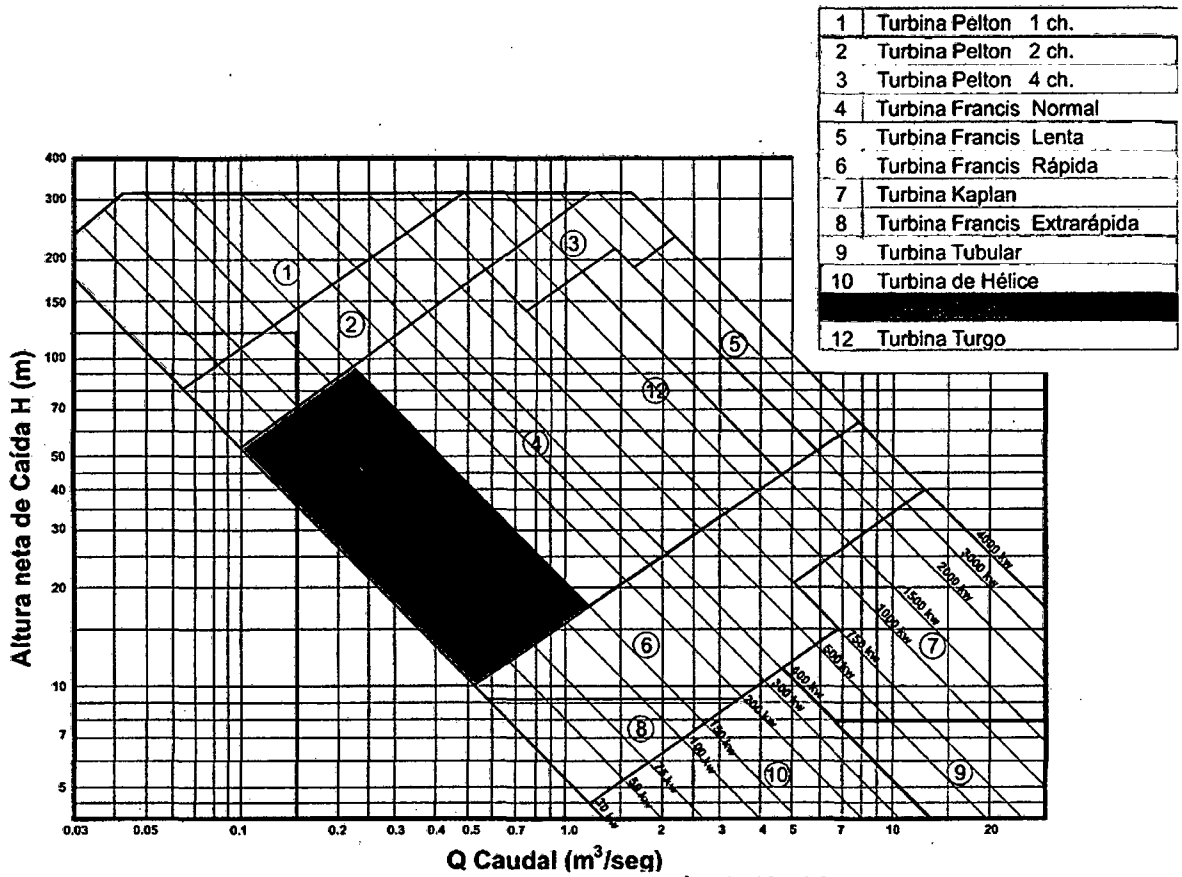


Fig. 3.2 Diagrama de selección de Turbinas.

3.1.1.2 DISEÑO DE LAS DIMENSIONES PRINCIPALES

[31] Quantz L, [43] Jara W., [32] ITDG Perú

1. DIÁMETROS PRINCIPALES.

a. Diámetro del chorro (d_{ch}): Utilizamos la ecuación 2.4:

$$d_{ch} = 550\sqrt{\frac{Q}{i\sqrt{H}}} \qquad (2.4)$$

Datos: Q = 0.15 m³/seg

H = 120 m

i = 1, 2 y 4Ch

Reemplazando valores en la ecuación (2.4) obtenemos:

$$d_{ch} = 550\sqrt{\frac{0.15}{1\sqrt{120}}} = 64.36\ mm$$

$$d_{ch} = 550\sqrt{\frac{0.15}{2\sqrt{120}}} = 45.51\,mm$$

b. Diámetro medio del Rodete (D_2): Utilizamos la ecuación (2.5):

$$D_2 = k_{D2} \times \frac{1000\sqrt{H}}{n} \qquad (2.5)$$

En donde: D_2 = Diámetro medio del rodete en mm

H = 120 m

K_{D2} = Constante, $37 \geq k_{D2} \leq 39$

n = 1800, rpm

Reemplazando valores en la ecuación (2.5) obtenemos:

$$D_2 = 37 \times \frac{1000\sqrt{120}}{1800} = 225.17mm$$

c. Dimensiones de las paletas

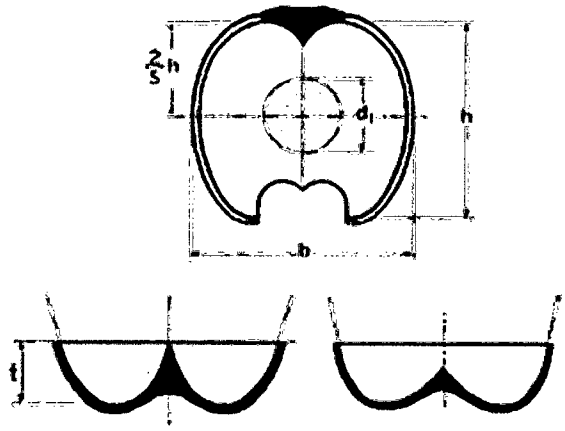


Fig. 3.3 Dimensiones de la paleta.

1. **Ancho de la paleta: b = 3.75 x d_{ch}** (2.7a)

   Pelton 1Ch: b = 3.75 x 64.36 = 241.35 mm

   Pelton 2Ch: b = 3.75 x 45.51 = 170.66 mm

2. **Altura de la paleta: h = 3.50 x d_{ch}** (2.7b)

   Pelton 1Ch: h = 3.50 x 64.36 = 225.26 mm

   Pelton 2Ch: h = 3.50 x 45.51 = 159.29 mm

3. **Espesor de la paleta: t = 1.50 x d_{ch}** (2.7c)

   Pelton 1Ch: t = 1.50 x 64.36 = 96.54 mm

   Pelton 2Ch: t = 1.50 x 45.51 = 68.27 mm

d. Diámetro externo del Rodete (D_3): Utilizamos la ecuación (2.6):

$$D_3 = D_2 + 2\left[\frac{3}{5} \times h\right] \qquad (2.6)$$

En donde: **D_3** = Diámetro externo del rodete, en mm
D_2 = Diámetro del rodete, en mm
h = Altura de la cuchara, en mm

$$D_3 = 225.17 + 2\left[\frac{3}{5} \times 159.29\right] = 416.318\,mm$$

e. El número de cucharas o paletas (Z):

Utilizamos la ecuación (2.8):

$$Z = \frac{1}{2}\left(\frac{D_2}{d_{ch}}\right) + k_z \qquad (2.8)$$

En donde: Z = Número de paletas
D_2 = Diámetro medio del rodete, 225.17 mm
D_{ch} = Diámetro del chorro, 45.51 mm
Kz = Constante, $14 \leq k_z \leq 16$

$$Z = \frac{1}{2}\left(\frac{225.17}{45.51}\right) + 14 = 16.5\ paletas$$

3.1.2 DISEÑO DE UNA TURBINA FRANCIS CON RODETE NORMAL

3.1.2.1 SELECCIÓN DE UNA TURBINA FRANCIS NORMAL

En un salto de **H = 20 m** y con un caudal de **Q = 1m^3/seg**, proyectar una turbina Francis con rodete normal de eje vertical, mostrado en la **figura 3.4**; y determinar sus dimensiones principales.

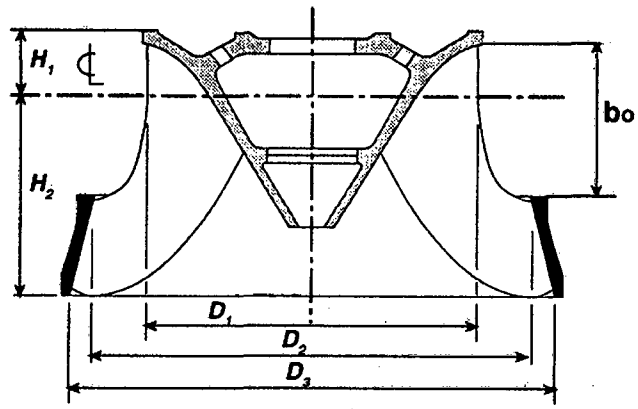


Fig. 3.4: Turbina Francis con rodete normal.

1. POTENCIA DE LA TURBINA (P):

La potencia de la turbina se determina utilizando la ecuación (2.1):

$$P = \frac{1000 \times Q \times H \times \eta_t}{76} \quad (2.1)$$

Datos: Q = 1 m^3/seg

H = 20 m

η = 0.80

Reemplazando valores en (2.1)

$$P = \frac{1000\ x\ 1.0\ x\ 20\ x\ 0.80}{76} = 210.53\,HP$$

La potencia en KW se determina utilizando la ecuación (2.2):

$$P_{kW} = \frac{1000 \times 1 \times 20 \times 0.80}{1.340 \times 76} = 157.11$$

La potencia obtenida de **157.11 KW** corresponde al rango de una mini central, según la **figura 1.2** página 22.

2. NÚMERO ESPECÍFICO (Ns):

Para calcular el número específico de revoluciones, utilizamos la ecuación (2.3):

$$N_s = \frac{n\sqrt{P}}{H.\sqrt[4]{H}} \qquad (2.3)$$

Datos: P = 210.53 HP

H = 20 m

Reemplazando valores en la ecuación (2.3)

$$N_s = \frac{n\sqrt{210.53}}{20\ x\ 4\sqrt{20}}$$

$$N_s = 0.343n \qquad (3.2)$$

Utilizando el resultado de (3.2), tabulamos en la tabla Nº 3.2 para las diferentes turbinas Francis y seleccionamos la turbina Francis Normal, que se encuentra dentro de los rangos establecidos.

Tabla Nº 3.2 Selección de una turbina Francis Normal.

Altura	Caudal	Tipo Turbina		Efic. Turb.	Pot. Turb.	Pot. Turb.	Vel. Rot.	Vel. Espcif.
H	Q	Nombre		η	P	P	n	Ns
m	m^3/seg				HP	kW	rpm	rpm
20	1	Francis	Lenta	0,80	210.53	157.11	600	205.83
20	1	Francis	Normal	0,80	210.53	157.11	600	205.83
20	1	Francis	Rápida	0,80	210.53	157.11	600	205.83
20	1	Michell		0,65	171.05	127.65	600	185.54

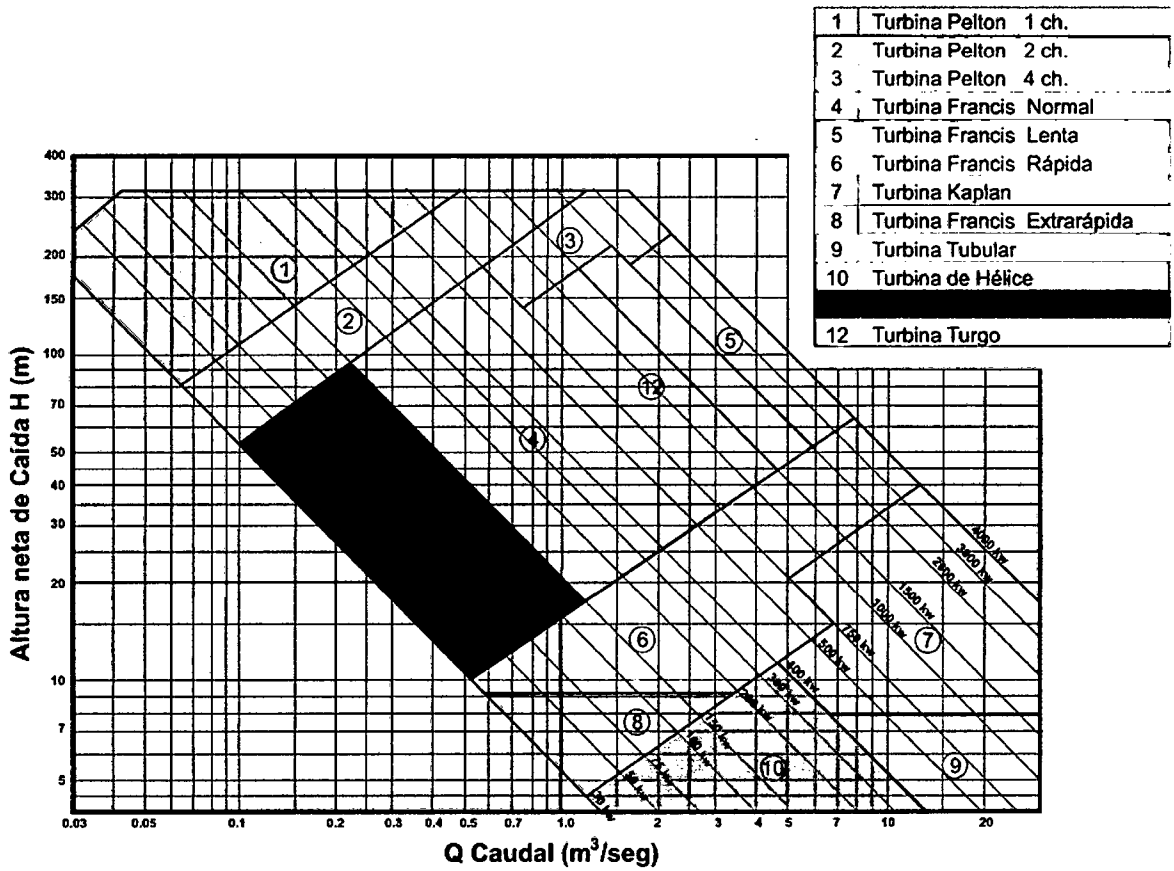


Fig. 3.5 Gráfico de selección de turbinas.

En la figura **3.5**, buscamos en el eje de las abscisas el caudal de **1m³/seg**, y en el eje de las ordenadas la altura de **20 metros;** de estos puntos trazamos líneas vertical y horizontal que se cortan en el campo de (4) ó (11), que corresponde a una Turbina Michell ó Francis; y por el **Ns** calculado (205.83) corresponde a una turbina Francis Normal, según la Tabla Nº 1.5 página 20 del capítulo 1.

3.1.2.2 DISEÑO DE LAS DIMENSIONES PRINCIPALES

1. DIÁMETRO DEL TUBO DE ASPIRACIÓN (D_3)

Para calcular el diámetro del tubo de aspiración (D_3), utilizamos la ecuación (2.10):

$$D_3 = 1000\sqrt{\frac{4Q}{\pi C_3}} \quad (2.10)$$

En donde: D_3 = Diámetro en el tubo de aspiración, en mm

Q = Caudal, 1m³/seg

C_3 = Velocidad de salida, m/seg

Para determinar la velocidad de salida C_3 utilizamos la ecuación (2.11):

$$C_3 = \sqrt{\frac{2 \times g \times k_c \times H}{100}} \quad (2.11)$$

En donde: C_3 = Velocidad de salida, m/seg
g = Gravedad, 9.8 m/seg^2
H = Altura, 20 metros
K_c = Constante según la turbina, en tanto por uno; $\eta_t/100$
Considerar 4% para Francis Lenta, 6% para Francis Normal, 12% para Francis Rápida y 25% para la Francis Extra rápida.

$$C_3 = \sqrt{2\left(\frac{9.8m}{seg^2}\right)\left(\frac{6}{100}\ x\ 20m\right)} = 4.85m/seg$$

Reemplazando valores en la ecuación (2.10) obtenemos:

$$D_3 = 1000\sqrt{\frac{4(1m^3/seg)}{\pi x 4.85m/seg}} = 512.37mm$$

D_3 = 512.37 mm

2. DIÁMETRO EXTERNO DEL RODETE (D_1)

Para calcular el diámetro externo del rodete D_1 utilizamos la siguiente relación:

$$D_1 = D_3 \times (0.4 + \frac{94.5}{N_S}) \quad (2.12)$$

En donde: D_1 = Diámetro externo del rodete, mm
D_3 = Diámetro del tubo de aspiración, 512 mm
N_s = Velocidad específica, 205.83 rpm

$$D_1 = 512 \times (0.4 + \frac{94.5}{205.83}) = 439.87mm$$

3. DIÁMETRO INTERNO DEL RODETE (D_2)

Para calcular el diámetro interno del rodete D_2, utilizamos la ecuación (2.13):

$$D_2 = D_3 \times (0.96 + 0.00038 \times N_s) \quad (2.13)$$

En donde: D_2 = Diámetro interno del rodete, mm
D_3 = Diámetro del tubo de aspiración, 512 mm
N_s = Velocidad específica, 205.83 rpm

$$D_2 = 512 \times (0.96 + 0.00038 \times 205.83) = 531.56mm$$

3.1.2.3 ANCHO DE LA CORONA DIRECTRIZ (bo)

El ancho de la corona directriz se calcula según la turbina Francis seleccionada, utilizando la siguiente ecuación:

1. **FRANCIS NORMAL (F_N):** Utilizamos la ecuación (2.15):

$$bo = D_1 \left[\frac{(N_S - 125) \times 0.15}{100} + 0.302 \right] \quad (2.15)$$

En donde: b_o = Ancho de la corona directriz, en mm
D_1 = Diámetro externo del rodete, 439.87 mm
N_s = Velocidad específica, 205.83 rpm

$$bo = 439.87 \left[\frac{(205.83 - 125) \times 0.15}{100} + 0.15 \right] = 119.31mm$$

2.1.3.4 CÁLCULO DEL NÚMERO DE ÁLABES (Z_o)

1. FRANCIS NORMAL (F_N): Utilizamos la ecuación (2.19):

$$Z_{FN} = 17 - \left[\frac{(N_S - 125)\times 2}{100}\right] \qquad (2.19)$$

En donde: Z_{FN} = Número de alabes para una turbina Francis normal

N_s = Velocidad específica, 205.83 rpm

$$Z_{FN} = 17 - \left[\frac{(205.83 - 125)\times 2}{100}\right] = 15.38\ alabes$$

3.1.3 DISEÑO DE UNA TURBINA FRANCIS CON RODETE RÁPIDO.

En un salto de **H = 20 m** y con un caudal de **Q = 2.1 m^3/seg**, proyectar una turbina Francis con rodete rápido, mostrada en la figura **3.6** y determinar sus dimensiones principales.

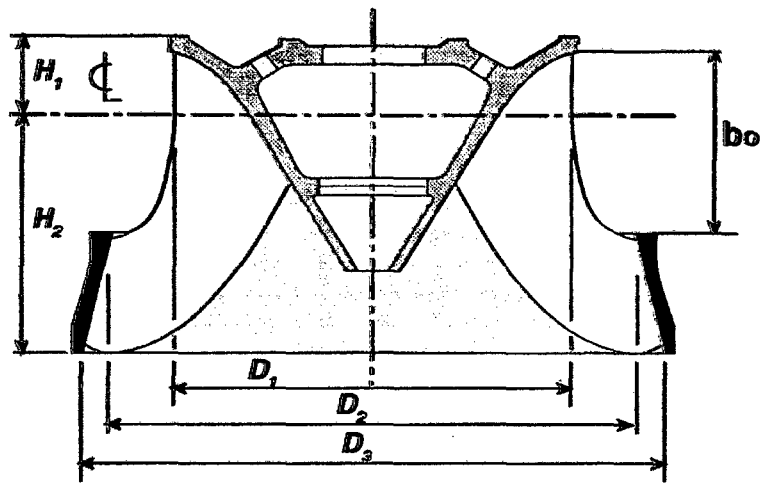


Fig. 3.6: Turbina Francis con rodete rápido.

3.1.3.1 SELECCIÓN DE UNA TURBINA FRANCIS RÁPIDA

1. POTENCIA DE LA TURBINA (P):

La potencia de la turbina se determina utilizando la ecuación (2.1):

$$P = \frac{1000 \times Q \times H \times \eta_t}{76} \qquad (2.1)$$

Datos: Q = 2.1 m^3/seg
H = 20 m
η = 0.80

Reemplazando valores en (2.1):

$$P = \frac{1000\ x\ 2.1\ x\ 20\ x\ 0.80}{76} = 442.105HP$$

La potencia en KW se determina utilizando la ecuación (2.2):

$$P_{kW} = \frac{1000 \times 2.1 \times 20 \times 0.80}{1.340 \times 76} = 329.93$$

La potencia obtenida de **329.93 KW** corresponde al rango de una mini central, según la **figura 1.2** página 22.

2. NÚMERO ESPECÍFICO (Ns):

Para calcular el número específico de revoluciones, utilizamos la ecuación (2.3):

$$N_s = \frac{n\sqrt{P}}{H.\sqrt[4]{H}} \qquad (2.3)$$

Datos: P = 442.105 HP
H = 20 m

Reemplazando valores en la ecuación (2.3)

$$N_s = \frac{n\ \sqrt{442.105}}{20\ x\ \sqrt[4]{20}}$$

$$N_s = 0.4971n \qquad (3.2)$$

Utilizando el resultado de (3.2), tabulamos en la tabla Nº 3.3 para las diferentes turbinas Francis y seleccionamos la turbina Francis Rápida, que se encuentra dentro de los rangos establecidos.

Tabla Nº 3.3 Selección de una turbina Francis Rápida.

Altura	Caudal	Turbina		Efic. Turb.	Pot. Turb.	Pot. Turb.	Vel. Rot.	Vel. Espcif.
H	Q	Nombre	Tipo	η	P	P	n	Ns
m	m³/seg				HP	KW	rpm	rpm
20	2.1	Francis	Lenta	0,80	442.11	329.93	600	298.28
20	2.1	Francis	Normal	0,80	442.11	329.93	600	298.28
20	2.1	Francis	Rápida	0,80	442.11	329.93	600	298.28
20	2.1	Francis	Extra Rápida	0,80	442.11	329.93	600	298.28

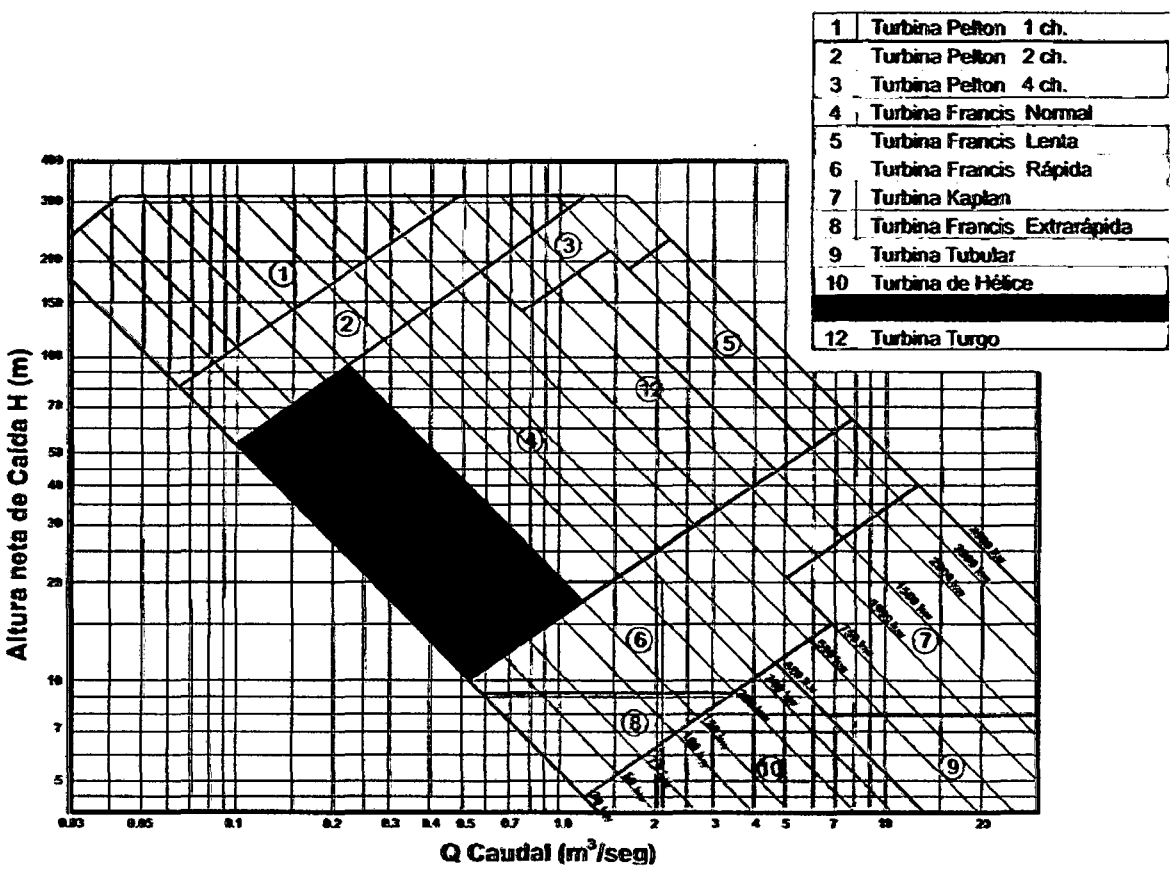


Fig. 3.7 Gráfico de selección de turbinas.

Seleccionamos una turbina Francis en rodete rápido, con **n = 600rpm**, del tipo mostrado en la figura **3.6.** Si utilizamos la figura **3.7**, observamos que si seleccionamos en el eje de las abscisas el caudal de **2.1m^3/seg**, y en el eje de las ordenadas la altura de **20 metros**, estas líneas se intersecan en el campo de **(6)**, que corresponde a una Turbina Francis Rápida, cuya potencia corresponde a **300KW**, en los cálculos hallamos **298.28KW**; por lo tanto la selección es correcta.

3.1.3.2 DISEÑO DE LAS DIMENSIONES PRINCIPALES

1. DIÁMETRO DEL TUBO DE ASPIRACIÓN (D_3)

Para calcular el diámetro del tubo de aspiración (D_3), utilizamos la ecuación (2.10):

$$D_3 = 1000\sqrt{\frac{4Q}{\pi C_3}} \quad (2.10)$$

En donde: **D_3** = Diámetro en el tubo de aspiración, en mm
Q = Caudal, 2.1m^3/seg
C_3 = Velocidad de salida, m/seg

Para determinar la velocidad de salida **C_3** utilizamos la ecuación (2.11):

$$C_3 = \sqrt{\frac{2 \times g \times k_c \times H}{100}} \quad (2.11)$$

En donde: **C_3** = Velocidad de salida, m/seg
g = Gravedad, 9.8 m/seg^2
H = Altura, 20 metros
K_c = Constante según la turbina, en tanto por uno; **η_t**/100
Considerar 4% para Francis Lenta, 6% para Francis Normal, 12% para Francis Rápida y 25% para la Francis Extra rápida.

$$C_3 = \sqrt{2\,(\frac{9.8m}{seg^2})\,(\frac{12}{100}\ x\ 20m)} = 6.86m/seg$$

Reemplazando valores en la ecuación (2.10) obtenemos:

$$D_3 = 1000\sqrt{\frac{4(2.1m^3/seg)}{\pi\, x\, 6.86\, m/seg}} = 624.31mm$$

2. DIÁMETRO EXTERNO DEL RODETE (D_1)

Para calcular el diámetro externo del rodete D_1 utilizamos la siguiente relación:

$$D_1 = D_3 \times (0.4 + \frac{94.5}{N_S}) \quad (2.12)$$

En donde: D_1 = Diámetro externo del rodete, mm
D_3 = Diámetro del tubo de aspiración, 624.31 mm
N_s = Velocidad específica, 298.28 rpm

$$D_1 = 624.31 \times (0.4 + \frac{94.5}{298.28}) = 447.52mm$$

3. DIÁMETRO INTERNO DEL RODETE (D_2)

Para calcular el diámetro interno del rodete D_2, utilizamos la siguiente relación:

$$D_2 = D_3 \times (0.96 + 0.00038 \times N_s) \quad (2.13)$$

En donde: D_2 = Diámetro interno del rodete, mm
D_3 = Diámetro del tubo de aspiración, 624.31 mm
N_s = Velocidad específica, 298.28 rpm

$$D_2 = 624.31 \times (0.96 + 0.00038 \times 298.28) = 670.1mm$$

3.1.3.3 ANCHO DE LA CORONA DIRECTRIZ (bo)

El ancho de la corona directriz se calcula según la turbina Francis seleccionada, utilizando la siguiente ecuación:

1. FRANCIS RÁPIDA (F_R)

$$bo = D_1\left[\frac{(N_S - 225)\times 0.22}{125} + k_R\right] \qquad (2.14c)$$

En donde: b_o = Ancho de la corona directriz, en mm
D_1 = Diámetro interno del rodete, 447.52mm
N_s = Velocidad específica, 298.28 rpm
K_R = Constante de diseño para Francis rápida.
$0.1582 \leq k_R \leq 0.30$ (seleccionar uno de los dos valores)

$$bo = 447.52\left[\frac{(298.28 - 225)\times 0.22}{125} + 0.30\right] = 191.97mm$$

3.1.3.4 CÁLCULO DEL NÚMERO DE ÁLABES (Z_o)

1. FRANCIS RÁPIDA (F_R)

$$Z_{FR} = 15 - \left[\frac{(N_S - 225)\times 2}{125}\right] \qquad (2.15c)$$

En donde: Z_{FR} = Número de alabes para una turbina Francis rápida
N_s = Velocidad específica, 298.28 rpm

$$Z_{FR} = 15 - \left[\frac{(298.28 - 225)\times 2}{125}\right] = 13.83\,alabes$$

3.1.4 CÁLCULO DE UNA TURBINA DE HÉLICE

En un salto de **H = 6metros** y con un caudal de **Q = 3.5 m^3/seg**, proyectar una turbina de hélice, de eje vertical, mostrado en la **figura 3.8**; y determinar sus dimensiones principales.

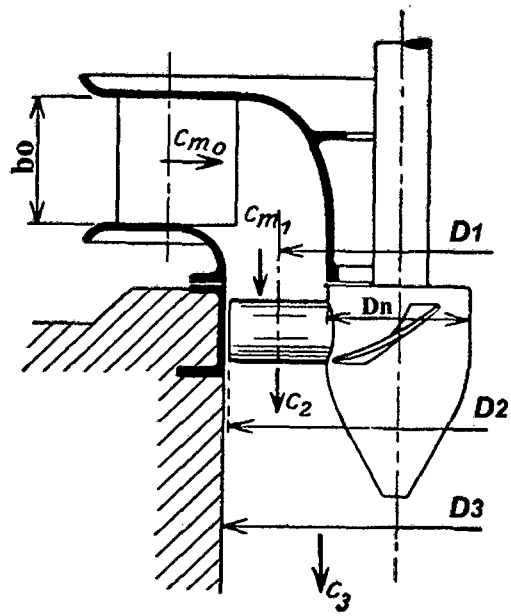


Fig. 3.8 Turbina de hélice.

3.1.4.1 SELECCIÓN DE UNA TURBINA DE HÉLICE.

1. POTENCIA DE LA TURBINA (P):

La potencia de la turbina se determina utilizando la ecuación (2.1):

$$P = \frac{1000 \times Q \times H \times \eta_t}{76} \quad (2.1)$$

Datos: Q = 3.5 m^3/seg
H = 6 m
η = 0.80

Reemplazando valores en (2.1)

$$P = \frac{1000\ x\ 3.5\ x 6\ x\ 0.80}{76} = 221.053\,HP$$

La potencia en KW se determina utilizando la ecuación (2.2):

$$P_{kW} = \frac{1000 \times 3.5 \times 6 \times 0.80}{1.340 \times 76} = 164.96$$

La potencia obtenida de **164.96 KW** corresponde al rango de una mini central, según la **tabla 4.3** página 22.

2. NÚMERO ESPECÍFICO (Ns):

Para calcular el número específico de revoluciones, utilizamos la ecuación (2.3):

$$N_s = \frac{n\sqrt{P}}{H.\sqrt[4]{H}} \qquad (2.3)$$

Datos: P = 221.053 HP

H = 6 m

Reemplazando valores en la ecuación (2.3)

$$N_s = \frac{n\sqrt{221.053}}{6\ x\ \sqrt[4]{6}}$$

$$N_s = 1.5833n \qquad (3.2)$$

Utilizando el resultado de (3.2), tabulamos en la tabla Nº 3.4 para las diferentes turbinas seleccionamos la turbina de Hélice, que se encuentra dentro de los rangos establecidos.

Tabla Nº 3.4 Selección de una turbina Francis Normal.

Altura	Caudal	Tipo Turbina		Efic. Turb.	Pot. Turb.	Pot. Turb.	Vel. Rot.	Vel. Espcif.
H	Q	Nombre	Clase	η	P	P	n	Ns
m	m³/seg				HP	kW	rpm	rpm
6	3.5	Pelton	1 Ch	0.70	193,42	140,26	600	888,62
6	3.5	Pelton	2 Ch	0,70	193,42	140,26	600	628,35
6	3.5	Pelton	4 Ch	0,70	193,42	140,26	600	444,31
6	3.5	Michell		0,65	179,61	130,24	600	856,29
6	3.5	Francis	Lenta	0,80	221,05	160,29	600	949,97
6	3.5	Francis	Normal	0,80	221,05	160,29	600	949,97
6	3.5	Francis	Rápida	0,80	221,05	160,29	600	949,97
6	3.5	De		0,80	221,05	164,96	600	949.97

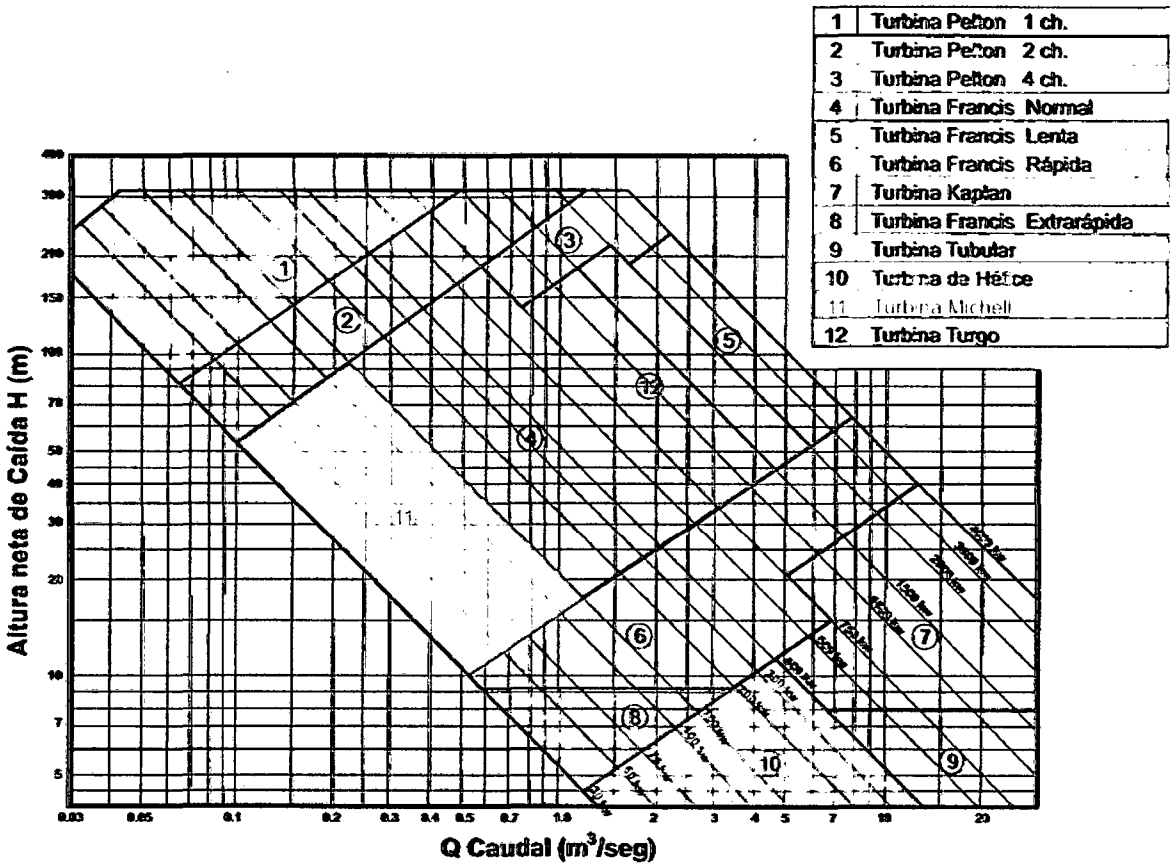


Fig. 3.9 Gráfico de selección de turbinas.

Seleccionamos una turbina de Hélice, con **n = 600rpm**, del tipo mostrado en la figura **3.8.** En la figura **3.9**, observamos que si seleccionamos en el eje de las abscisas el caudal de **3.5m³/seg**, y en el eje de las ordenadas la altura de **6metros**, estas líneas se intersecan en el campo de **(10)**,

que corresponde a una Turbina de Hélice, cuya potencia corresponde a **160kW**, en los cálculos hallamos **164.30kW**; por lo tanto la selección es correcta.

3.1.4.2 DISEÑO DE LOS DIMENSIONES PRINCIPALES.

El rodete de las turbinas de acción total mostrada en la figura **3.8**, se observan que se necesitan calcular los diámetros del rodete **D_1, D_2, D_3 y Dn**.

1. DIÁMETRO DEL TUBO DE ASPIRACIÓN (D_3):

Estas fórmulas empíricas son aplicables para las turbinas Kaplan, Hélice y Tubular, utilizamos la ecuación (2.22):

$$D_3 = 1000\sqrt{\frac{4Q}{\pi C_3}} \qquad (2.22)$$

En donde: **D_3** = Diámetro en el tubo de aspiración, en mm
Q = Caudal, 3.5 m^3/seg
H = Altura del salto, 6 metros,
C_3 = Velocidad de salida, m/seg

Para determinar la velocidad de salida **C_3** utilizamos la ecuación (2.23):

$$C_3 = \sqrt{\frac{2 \times g \times k_c \times H}{100}} \qquad (2.23)$$

En donde: **C_3** = Velocidad de salida, m/seg
g = Gravedad, 9.8 m/seg^2
H = Altura del salto, 6 metros,
K_c = Constante, 30%; $\eta_t/100$
Considerar para la Kaplan, Hélice y Tubular el 30%.

$$C_3 = \sqrt{\frac{2 \times 9.8 \times 30 \times 6}{100}} = 5.94\, m/seg$$

Reemplazando valores en la ecuación (2.22):

$$D_3 = 1000\sqrt{\frac{4\times 3.5}{\pi \times 5.94}} = 866.16 mm$$

2. DIÁMETRO DEL RODETE (D_2)

Para calcular el diámetro externo del rodete D_1 de la turbina Hélice mostrada en la figura **3.8**, utilizamos la ecuación (2.24):

$$D_2 = 0.98 D_3 \qquad (2.24)$$

En donde: D_2 = Diámetro del rodete, mm
D_3 = Diámetro del tubo de aspiración, 866.16 mm

Reemplazando valores en la ecuación (2.24):

$$D_2 = 0.98 \times 866.16 = 848.84\ mm$$

3. DIÁMETRO DEL CUBO DEL RODETE (D_N)

Para calcular el diámetro del cubo del rodete D_N de la turbina Hélice mostrada en la figura **3.8**, utilizamos la ecuación (2.25):

$$D_N = D_3(0.25 + \frac{94.64}{N_S}) \qquad (2.25)$$

En donde: D_N = Diámetro del cubo del rodete, mm
D_3 = Diámetro del tubo de aspiración, 866.16 mm
N_S = Velocidad específica, 949.97 rpm

Reemplazando valores en la ecuación (2.25) obtenemos:

$$D_N = 866.16 \times (0.25 + \frac{94.64}{949.97}) = 302.83 mm$$

4. DIÁMETRO MEDIO DEL RODETE (D_1)

Para calcular el diámetro medio del rodete D_1 de la turbina Hélice mostrada en la figura **3.8**, utilizamos la ecuación (2.26):

$$D_1 = D_N + (\frac{D_2 - D_N}{2}) \qquad (2.26)$$

En donde: D_1 = Diámetro medio del rodete, mm
D_N = Diámetro del cubo del rodete, 302.83 mm
D_2 = Diámetro del rodete, 848.84 mm

Reemplazando valores en la ecuación (2.19) obtenemos:

$$D_1 = 302.83 + (\frac{848.84 - 302.83}{2}) = 575.84 mm$$

3.1.4.3 ANCHO DE LA RUEDA DIRECTRIZ (Bo)

El ancho de la corona directriz se calcula según la turbina de acción total seleccionada, utilizando la ecuación (2.27):

$$B_0 = \frac{0.8 \; x \; Q \times 10^6}{0.9 \; x \; D_2 \; x \; \pi \; x \; Cmo} \qquad (2.27)$$

En donde: B_o = Ancho de la rueda directriz, en mm
D_2 = Diámetro del rodete, 848.84 mm
C_{mo} = Velocidad meridiana, en m/seg
Q = Caudal, 3.5 m^3/seg

La sección libre de salida debe ser mayor que la superficie de entrada en el rodete:

0.6 $c_{m1} \leq c_{mo} \leq$ **0.7** c_{m1}

Utilizar en el diseño: $C_{mo} = 0.65 \times C_{m1}$ (2.28)

En donde: C_{m1} = Componente de la velocidad de salida, m/seg

Para calcular esta componente utilizamos la siguiente relación:

$$C_{m1} = \frac{4 \times 0.8 \times Q \times 10^6}{(D_2^2 - D_N^2)\,\pi}\, m/seg \qquad (2.29)$$

En donde: C_{m1} = Componente de la velocidad de salida, en m/seg
D_2 = Diámetro del rodete, 848.84 mm
D_N = Diámetro del cubo del rodete, 302.83 mm
Q = Caudal, 3.5 m^3/seg

Reemplazando valores en la ecuación (2.23) obtenemos:

$$C_{m1} = \frac{4 \times 0.8 \times 3.5 \times 10^6}{(848.84^2 - 302.83^2)\,\pi} = 5.675\, m/seg$$

Reemplazando valores en la ecuación (2.21) obtenemos:

$$B_0 = \frac{0.8\ x\ 3.5 \times 10^6}{0.9\ x\ 848.84\ x\ \pi\ x\ 0.65 \times 5.675} = 316.272\, mm$$

3.1.4.4 CÁLCULO DEL NÚMERO DE ALABES (Z)

El número de alabes para la turbina de acción total seleccionada, se calcula con la ecuación (2.30):

$$Z = \frac{2170 - 1.2 \times N_s}{250} \qquad (2.30)$$

En donde: Z = Número de álabes
N_s = Velocidad específica, 949.97 rpm

Reemplazando valores en la ecuación (2.30) obtenemos:

$$Z = \frac{2170 - 1.2 \times 949.97}{250} = 4.12\, alabes$$

3.1.5 DISEÑO DE UNA TURBINA MICHELL.

En un salto de **H = 20 metros** y con un caudal de **Q = 0.4 m^3/seg**, proyectar una turbina Mitchell-Banki, mostrada en la figura **3.10** y determinar las dimensiones principales de la turbina.

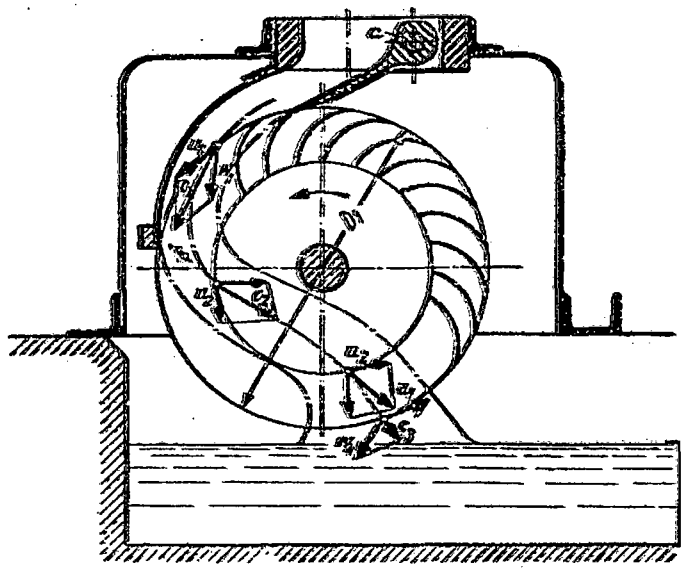


Fig. 3.10: Turbina Michell.

3.1.5.1 SELECCIÓN DE UNA TURBINA MICHELL.

1. POTENCIA DE LA TURBINA (P).

Datos: Q = 0.4 m^3/seg, H = 20 m, η = 0.65 (Tabla 1.5)

Reemplazando valores en la ecuación (2.1)

$$P = \frac{1000\ x\ 0.4\ x20\ x\ 0.65}{76}$$

$$P = 68.421\,HP$$

Para calcular la potencia en KW, reemplazando valores en la ecuación (2.2):

$$P_{kW} = \frac{1000 \times 0.4 \times 20 \times 0.65}{1.340 \times 76}$$

$$P = 51.06\,KW$$

La potencia obtenida de 51.06 KW corresponde al rango de una mini central, según la **figura 1.2** página **22.**

2. NÚMERO ESPECÍFICO DE REVOLUCIONES (Ns).

Para calcular el número específico de revoluciones, utilizamos la ecuación (2.3):

Datos: P = 68.421 HP,

H = 20 m,

i= 1 boquilla

Reemplazando valores en la ecuación (2.3)

$$N_s = \frac{n\sqrt{P}}{H.\sqrt[4]{H}} \qquad (2.3)$$

Datos: P = 68.42 HP, H = 20 m

Reemplazando valores en la ecuación (2.3):

$$N_s = \frac{n\sqrt{68.42}}{20\ x\ \sqrt[4]{20}}$$

$$N_s = 0.1956n \qquad (3.4)$$

Utilizando el resultado de (3.4), tabulamos en la tabla **Nº 3.5** con las diferentes turbinas y seleccionamos la turbina Michell.

Tabla Nº 3.5 Selección de una turbina Mitchell.

Altura	Caudal	Turbina		Efic. Turb.	Pot. Turb.	Pot. Turb.	Vel. Rot.	Vel. Espcif.
H	Q	Nombre	Clase	η	P	P	n	Ns
m	m³/seg				HP	KW	rpm	rpm
20	0,4	Pelton	1 Ch	0.70	73,68	53,43	450	91,33
20	0,4	Pelton	2 Ch	0,70	73,68	53,43	450	91,33
20	0,4	Pelton	4 Ch	0,70	73,68	53,43	450	91,33
20	0,4	Michell Banki		0,65	68,42	51,06	450	88,01
20	0,4	Francis	Lenta	0,80	84,21	61,06	450	97,63
20	0,4	Francis	Normal	0,80	84,21	61,06	450	97,63
20	0,4	Francis	Rápida	0,80	84,21	61,06	450	97,63
20	0,4	Kaplan y de Hélice		0,82	97,89	70,98	450	105,27

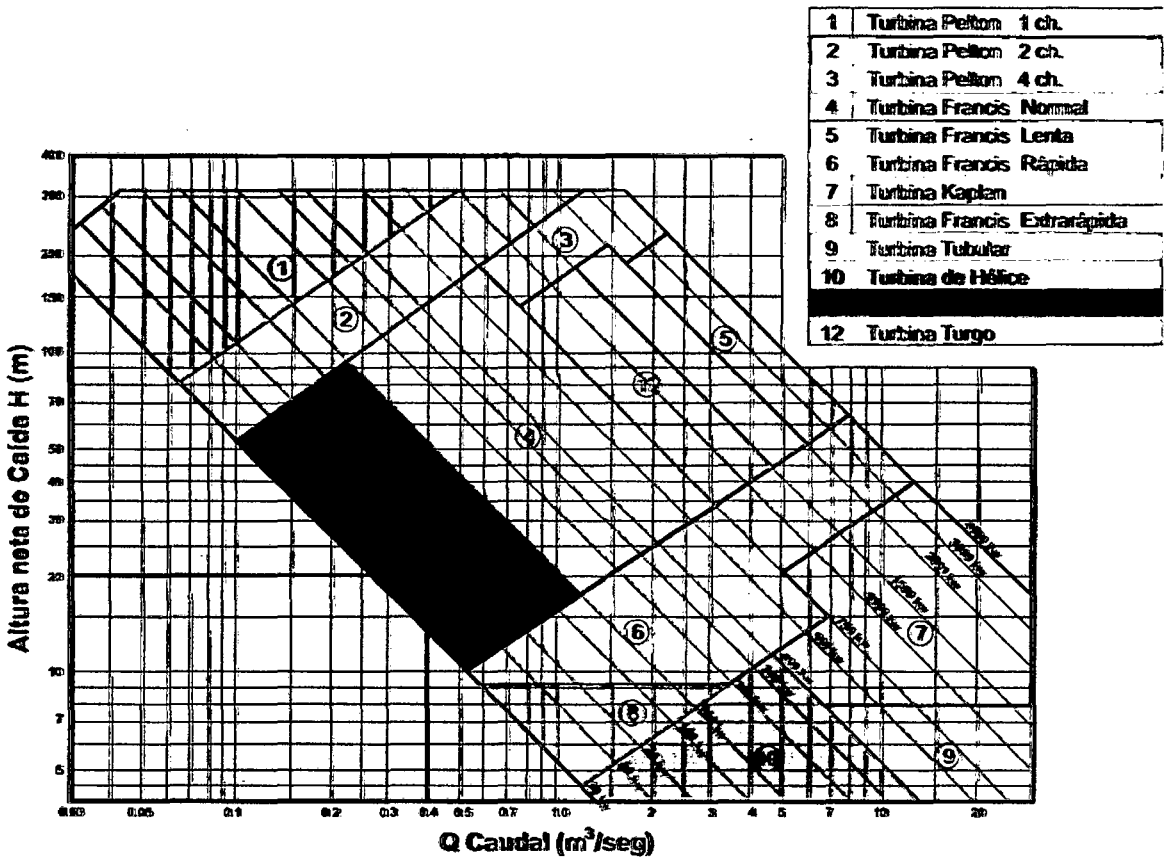


Fig. 3.11: Gráfico de selección de Turbinas.

Seleccionamos una turbina Michell Banki de una boquilla, con **n = 450rpm**, del tipo mostrado en la figura **3.10.** Si utilizamos la figura **3.11**, observamos que si seleccionamos en el eje de las abscisas el caudal de **0.4m³/seg** y en el eje de las ordenadas la altura de **20metros**, estas líneas

se intersecan en el campo de **(11)**, que corresponde a una Turbina de Michell, cuya potencia corresponde a **50KW**, en los cálculos hallamos **51.06KW**; por lo tanto la selección es correcta.

3.1.5.2 DISEÑO DE LAS DIMENSIONES PRINCIPALES

El rodete de la turbina Mitchell-Banki de acción total mostrada en la figura **3.12**, se observan que se necesitan calcular los diámetros del rodete D_1 y D_2.

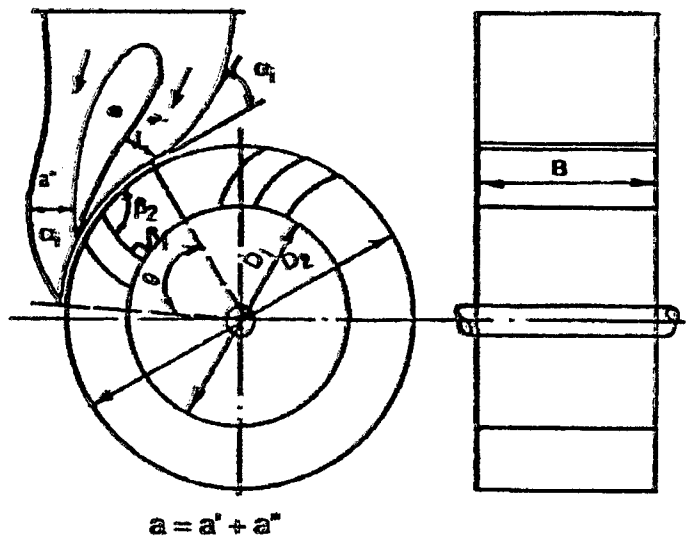


Fig. 3.12 Diámetro del rodete.

1. DIÁMETRO EXTERIOR DEL RODETE (D_2)

El diámetro exterior del rodete mostrado en la figura **3.12**, la podemos calcular utilizando la ecuación (2.31):

$$D_2 = K_m \times \frac{1000\sqrt{H}}{N} \qquad (2.31)$$

En donde: D_2 = Diámetro exterior del rodete, en mm

H = Altura del salto, 20 metros,

N = Velocidad de rotación, 450 rpm

K_m = Constante de la turbina Mitchell, $37 \leq K_m \leq 39$

K_m = 37 (asumir este valor)

Reemplazando valores en la ecuación (2.31) obtenemos:

$$D_2 = 37 \times \frac{1000\sqrt{20}}{450} = 367.709mm$$

2. DIÁMETRO INTERIOR DEL RODETE (D_1)

El diámetro interior del rodete mostrado en la figura **3.12**, la podemos calcular utilizando la ecuación (2.32):

$$D_1 = 0.66D_2 \quad (2.32)$$

$$D_1 = 0.66 \times 367.709 = 242.688mm$$

3. DIÁMETRO DEL CHORRO (d_{chorro})

Suponiendo que se adopta un chorro cilíndrico, el diámetro del chorro la calculamos con la ecuación (2.33):

$$d_{chorro} = 1000 \times \sqrt{\frac{4 \times Q}{C_1 \times \pi}} \qquad (2.33)$$

En donde: D_{chorro} = Diámetro del chorro, mm
Q = Caudal, 0.4 m^3/seg
C_1 = Velocidad del chorro, en m/seg

4. VELOCIDAD DEL CHORRO (C_1)

La velocidad del chorro, se calcula utilizando la ecuación (2.34):

$$C_1 = \varphi\sqrt{2.g.H} \qquad (2.34)$$

En donde: C_1 = Velocidad del chorro, m/seg

g = Aceleración de la gravedad, 9,80665 m/seg^2

H = Altura, 20 m

φ = Coeficiente de salida del agua: 0.95

Reemplazando valores en la ecuación (2.34) obtenemos:

$$C_1 = 0.95\sqrt{2x9.8\times 20} = 18.81\ m/seg$$

Reemplazando valores en la ecuación (2.33):

$$d_{chorro} = 1000\times\sqrt{\frac{4\times 0.4}{18.81\times\pi}} = 164.55mm$$

5. DIÁMETRO MEDIO DEL RODETE (D_m)

El diámetro medio del rodete mostrado en la figura **3.12**, la podemos calcular utilizando la ecuación (2.35):

$$D_m = \frac{D_2 + D_1}{2} \qquad (2.35)$$

En donde: **D_m** = Diámetro medio del rodete, en mm

D_2 = Diámetro exterior del rodete, 367.709 mm

D_1 = Diámetro interior del rodete, 242.688 mm

Reemplazando valores en la ecuación (2.35):

$$D_m = \frac{367.709 + 242.688}{2} = 305.198mm$$

6. ESPESOR DEL CHORRO.

Para calcular el espesor del chorro utilizamos la ecuación (2.36).

$$a = k_a \times D_2 \qquad (2.36)$$

Donde: **a** = Espesor del chorro, en mm,

$\mathbf{D_2}$ = Diámetro exterior del rodete, en mm

k_a = Coeficiente que depende del ángulo del inyector α_i y del ángulo de admisión θ.

Para $\alpha_i = 16°$ se pueden tomar los siguientes valores:

θ	60°	90°	120°
k_a	0.1443	0.2164	0.2886

En el caso de usar una paleta directriz central, como se observa en la figura **2.23**, el espesor del chorro: $a = a' + a''$.

Reemplazando valores en la ecuación (2.30) obtendremos:

$$a = 0.1443 \times 367.709 = 53.06\,mm$$
$$a = 0.2164 \times 367.709 = 79.57\,mm$$
$$a = 0.2886 \times 367.709 = 106.121\,mm$$

7. VELOCIDAD ANGULAR DE LA RUEDA (n).

Para calcular el número de revoluciones de la rueda, utilizamos la ecuación **2.37**.

$$n = 1000 \times \left(\frac{60.u_1}{D_2.\pi}\right) \qquad (2.37)$$

8. VELOCIDAD TANGENCIAL (u_1).

La velocidad tangencial $\mathbf{u_1}$ en el rodete se calcula con la ecuación 2.32.

$$u_1 = 2.1 \times \sqrt{H} \qquad (2.38)$$

$$u_1 = 2.1 \times \sqrt{H} = 2.1 \times \sqrt{20} = 9.3915 m/seg$$

Para calcular el número de revoluciones de la rueda, utilizamos la siguiente relación:

$$n = 1000 \times \left(\frac{60 \; x \; 9.3915 \; m/seg}{367.709 \; mm \; x \; \pi} \right) = 487.789 \, vueltas/\min$$

$$n = \frac{N_s \times H \times \sqrt[4]{H}}{\sqrt{P}} = \frac{88.01 \; x \; 20 \times \sqrt[4]{20}}{\sqrt{68.42}} = 450 rpm$$

No requiere utilizar un engranaje para conseguir para el alternador la velocidad angular deseada de 450 rpm.

9. ANCHO DEL RODETE (B).

Si el ángulo del inyector α_i = 16°, este ángulo varía entre 15° ≤α_i ≥ 20°; el ancho del rodete la calculamos con la siguiente relación:

$$B = 98.8 \times \frac{Q}{D_2 \times \sqrt{H}} \times \frac{10^6}{\theta} \quad (mm) \qquad (2.39)$$

En donde:
- **B** = Ancho del rodete, mm
- **Q** = Caudal, 0.4 m^3/seg
- **H** = Altura, 20 m
- **D_2** = Diámetro exterior del rodete, 367.709 mm
- **θ** = Ángulo de admisión, valores de 30°,60°,120°

Remplazando valores en la ecuación (2.39) tendremos:

$$B = 98.8 \times \frac{0.4}{367.709 \times \sqrt{20}} \times \frac{10^6}{60} = 400.540 mm$$

$$B = 98.8 \times \frac{0.4}{367.709 \times \sqrt{20}} \times \frac{10^6}{90} = 267.027\, mm$$

$$B = 98.8 \times \frac{0.4}{367.709 \times \sqrt{20}} \times \frac{10^6}{120} = 200.270\, mm$$

10. MEDIDAS Y NÚMERO DE PALETAS

El número de alabes para la turbina Michell seleccionada, se calcula con la siguiente ecuación:

$$Z = \frac{D_2 + 900}{50} \qquad (2.40)$$

$$Z = \frac{\pi \times D_2}{t_a} \qquad (2.41)$$

En donde: **Z** = Número de álabes
D_2 = Diámetro exterior del rodete, 367.709 mm
t_a = Paso o división exterior, en mm

Las dimensiones de la cuchara son proporcionales al diámetro del chorro, para evitar una destrucción rápida de la arista media, no puede ser más pequeño pues el agua que sale de una cuchara no debe golpear la siguiente.

b = 2.8 d = 2.8 x 260.2 = 728.56 mm

h = 2.8 d = 2.8 x 260.2 = 728.56 mm

t = 0.8 d = 0.8 x 260.2 = 208.16 mm

Escogemos ahora un paso o división exterior de:

$$t_a \cong \frac{h}{4.5} = \frac{208.16}{4.5} = 46.26mm$$

El valor calculado corresponde aproximadamente a los tipos normales; por lo tanto el número de paletas la calculamos de la siguiente relación:

$$Z = \frac{\pi \ x \ 367.709}{46.26} = 24.97 paletas$$

$$Z = \frac{367.709 + 900}{50} = 25.35 paletas$$

3.2 DISEÑO DEL SOFTWARE UTILIZANDO EL MÉTODO DE LAS REDES NEURONALES

El **Método de las Redes Neuronales Artificiales (MERNA)**, es un método que presenta un gran número de características semejantes a las del cerebro humano, son capaces de aprender de la experiencia. La utilización del MERNA es un método importante que en general permite:

- Seleccionar a la turbina hidráulica.
- Diseñar las dimensiones principales de las turbinas hidráulicas.

Como la energía de origen hidráulico ha sido la de mayor acogida hasta el momento, debido a esta gran aceptación que se logró de la generación hidroeléctrica, y a la importancia de la electricidad, cada vez se ha vuelto más especializado el estudio de este proceso, convirtiéndose en un amplio campo de acción de la ingeniería, debido a la magnitud y frecuencia de problemas que suelen presentarse y que deben resolverse.

Ante la gran variedad de turbinas hidráulicas para la generación de energía eléctrica, uno de estos problemas, que se presenta a la hora de seleccionar cuál es el tipo de turbina más conveniente, para un salto y un caudal, pues debe lograrse realizar una instalación con la que se obtenga el mejor aprovechamiento de los recursos, con facilidades de mantenimiento, y al precio más favorable.

Por esta ***razón se ha desarrollado un método en el proceso de selección y cálculo de las turbinas hidráulicas,*** que permita seleccionar y calcular a la de turbina sin necesidad de tener a un experto en esta materia. Para seleccionar una turbina hidráulica, se requiere clasificar las diferentes turbinas por el salto, caudal, su velocidad específica, la potencia y sus eficiencias; esta clasificación se muestra en la página 20, Tabla 1.5.

3.2.1 NEUROSHELL 2.

El Software el NeuroShell 2 (la página de inicio se muestra en la figura 3.13) es un programa para Windows que se utiliza para tratar para resolver una amplia variedad de problemas en materia de modelación de datos de negocio, investigación básica y entornos industriales. NeuroShell 2 es una herramienta para la creación de sistemas RNA, que permite predecir en la selección y dimensiones de una turbina.

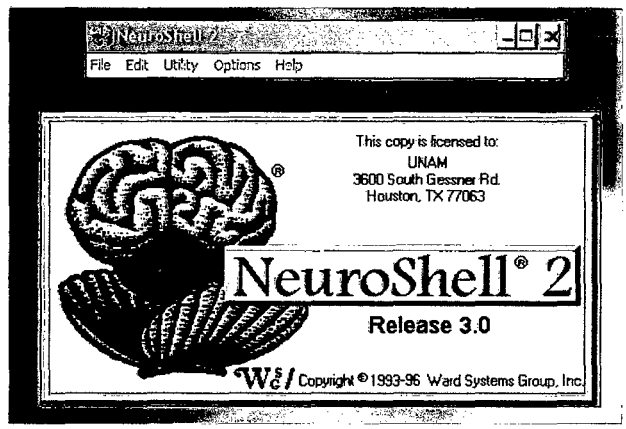


Fig. 3.13 NeuroShell 2.

Esta aplicación combina una interfaz de diseño modular con avanzados procedimientos de aprendizaje, está diseñado y entrenado con los datos que figuran en una hoja de cálculo (por ejemplo, Excel o 1-2-3) e incorporada en la hoja de balance (*dentro de la red de desarrollo NeuroShell*).

NeuroShell 2 genera/compila automáticamente una DLL, que podrá ser utilizada en cualquier entorno de programación que soporte el acceso a librerías dinámicas, como por ejemplo: Visual Basic, Access, Visual C++ y Jaba.

3.2.2 DISEÑO DEL MÉTODO DE LAS REDES NEURONALES (MERNA)

Las **RNA** han demostrado conseguir buenos resultados en el campo de la predicción. La estructura del ***MERNA*** está realizada en diferentes fases sucesivas, cada fase con una función específica y diseñada para generar información de forma independiente, cada una de ellas necesita los resultados del razonamiento inferido en las etapas anteriores; estas fases son:

- Fase 1: Adquisición del conocimiento.
- Fase 2: Desarrollo del sistema.
- Fase 3: Obtención de datos a partir del MERNA.

3.2.2.1 Fase 1: Adquisición del conocimiento.

La adquisición de conocimiento se refiere al conocimiento desde alguna fuente a la base de conocimientos del sistema. Las fuentes del conocimiento son los expertos humanos, libros, y datos experimentales agrupados en bases de datos.

La adición de conocimiento puede ser directa, o como resultado de una inducción a partir de ejemplos, o como deducción de otros conocimientos ya al-

macenados en la base de conocimientos. Las técnicas manuales consumen mucho tiempo y son costosas.

3.2.2.2 Fase 2: Desarrollo del Sistema.

En este paso se elige la arquitectura de red y se eligen los valores de sus parámetros y los criterios seguidos (caudal, la altura, la velocidad de rotación, la potencia, la eficiencia, etc.).

Los resultados obtenidos en la fase 1 son los datos de partida y el proceso metodológico es el siguiente: el planteamiento del problema, el análisis, el diseño y la implementación, aquí es necesario manejar un lenguaje de programación y convertir nuestra respuesta diseñada en código ejecutable por la computadora. Existen muchos lenguajes que nos servirían para implementar el programa entre los más usados están el C, el C++ y el Java; el MERNA utiliza el Java.

3.2.2.3 Fase 3: Obtención de datos a partir del MERNA

Según las condiciones del requerimiento suministradas por el usuario, el MERNA selecciona entre todas las turbinas posibles que técnicamente cumpla con estos requerimientos; de ahí la importancia que tiene la fiabilidad de los datos incorporados.

Esta es la fase el usuario puede realizar la selección final de la turbina para una mini central. El **MERNA** selecciona 10 posibles turbinas, como se observa en la **Tabla 3.6** Clasificación de las diferentes turbinas hidráulicas.

Tabla 3.6 Clasificación de las diferentes turbinas hidráulicas

Clases de Turbinas	Abrev.	Tipos de Rodete	Ns (rpm)	Q (m^3/seg)	H_{max} adm. (m)	η %	P KW
PELTON	TP1CH	1 Ch	10 – 30	0.03 – 0.41	90 – 300	70-91	30 – 900
	TP2CH	2 Ch	30 – 50	0.07 – 1.1	60 – 300		30 - 2500
	TP4CH	4 Ch	30 – 50	0.65 – 2.0	150 - 300		750 - 4000
MICHELL	TMB		40 – 160	0.12 – 1.0	12 – 80	65-82	30 - 150
FRANCIS	TFL	Lenta	60 - 125	1.3 – 7	50 – 180	80-92	1500 - 4000
	TFN	Normal	125 – 225	0.25 – 2.5	20 – 150		150 - 750
	TFR	Rápida	225 – 350	0.6 – 12	10 - 55		30 - 4000
	TFER	Extra rápida	350 - 450	0.7 – 3.0	5 - 9		30 - 180
KAPLAN y de Hélice	TK	Kaplan	300 – 600	5 – 25	8,5 – 35	80-93	400 - 4000
	TDH	De Hélice	500 – 1000	1.4 - 11	2,5 – 10		30 – 400

Cada una de estas posibles soluciones tiene una determinada zona de aplicación en el diagrama técnico "caudal – altura" de selección del tipo de turbina, como se observa en la **figura 3.2**.

3.2.3 GESTIÓN DE LAS BASES DE CONOCIMIENTOS DEL MERNA.

La gestión de la base de conocimiento abarca no solo la recopilación de datos de proyectos reales, sino también la extracción del conocimiento a los expertos. La RNA requiere el conocimiento especializado extraído de la experiencia de expertos humanos; por lo tanto es necesario tener ejemplos correctamente desarrollados de las diferentes turbinas; tal como se observan en las tablas siguientes, que permita al software transformar los datos operacionales en información útil.

Tabla 3.7 Cálculo de selección de una turbina Pelton 1 Ch.

H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Tpelton	Coe. Sal.P:(j)	Pot.Pelton (hp)	Pot.Pelton (kW)	Ns Pel(1Ch)	Vel. Sal. Pelton (m/seg)	Pelton 1ch
300	0.05	1200	70.00	0,97	138.158	100.184	11.30	37.190	10
300	0.2	720	70.00	0,97	552.632	400.736	13.56	37.190	10
200	0.04	1200	70,00	0,97	73.684	53.432	13.70	30.366	10
200	0.1	720	70,00	0,97	184.211	133.579	12.99	30.366	10
150	0.04	900	70,00	0,97	55.263	40.074	12.75	26.298	10
150	0.15	450	70,00	0,97	207.237	150.276	12.34	26.298	10
100	0.06	600	70,00	0,97	55.263	40.074	14.10	21.472	10
100	0.08	450	70,00	0,97	73.684	53.432	12.22	21,472	10
90	0.06	514	70.00	0.97	49.737	36.066	13.08	20.370	10

Tabla 3.8 Cálculo de selección de una turbina Pelton 2 Ch.

H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Tpelton	Coe. Sal.P:(j)	Pot.Pelton (hp)	Pot.Pelton (kW)	Ns Pel(2Ch)	Vel. Sal. Pelton (m/seg)	Pelton 2ch
300	0.5	1800	70.00	0.97	1381.579	1001.841	37.89	37.190	10
200	0.3	1800	70.00	0.97	552.632	400.736	39.78	30.366	10
200	0.5	1200	70.00	0.97	921.053	667.894	34.24	30.366	10
150	0.2	1800	70.00	0.97	276.316	200.368	40.30	26.298	10
100	0.1	1800	70.00	0.97	92.105	66.789	38.63	21.472	10
100	0.2	1200	70.00	0.97	184.211	133.579	36.42	21.472	10
90	0.08	1800	70.00	0.97	66.316	48.088	37.39	20.370	10
90	0.15	1200	70.00	0.97	124.342	90.166	34.13	20.370	10
80	0.07	1800	70.00	0.97	51.579	37.402	38.21	19.205	10
80	0.15	1200	70.00	0.97	110.526	80.147	37.29	19.205	10
70	0.1	1200	70.00	0.97	64.474	46.753	33.65	17.965	10
70	0.13	1200	70.00	0.97	83.816	60.778	38.37	17.965	10
60	0.09	1200	70.00	0.97	49.737	36.066	35.84	16.632	10
60	0.1	1200	70.00	0.97	55.263	40.074	37.77	16.632	10

Tabla 3.9 Cálculo de selección de una turbina Pelton 4 Ch.

H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Tpelton	Coe. Sal.P:(j)	Pot. Pel-ton (hp)	Pot. Pel-ton (kW)	Ns Pel.(4Ch)	Vel. Sal. Pelton (m/seg)	Pelton 4ch
300	1.2	1800	70.00	0.97	3315.789	2404.419	41.51	37.190	10
300	1.6	1200	70.00	0.97	4421.053	3205.892	31.95	37.190	10
250	0.9	1800	70.00	0.97	2072.368	1502.762	41.21	33.950	10
250	1	1800	70.00	0.97	2302.632	1669.735	43.44	33.950	10
250	2	1200	70.00	0.97	4605.263	3339.470	40.96	33.950	10
200	0.7	1800	70.00	0.97	1289.474	935.052	42.97	30.366	10
200	1.2	1200	70.00	0.97	2210.526	1602.946	37.51	30.366	10
150	0.7	1200	70.00	0.97	967.105	701.289	35.54	26.298	10
150	0.8	1200	70.00	0.97	1105.263	801.473	38.00	26.298	10

Tabla 3.10 Cálculo de selección de una turbina Francis Lenta.

H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Tfrancis	Pot. Francis (hp)	Pot. Francis (kW)	Ns Francis	Vs Francis C3 (m/seg)	VTg. Francis (m/seg)	Francis Lenta
200	1.3	900	80.00	2,736.842	1,984.600	62.60	12.522	29.6984848	10
200	2.5	900	80.00	5,263.158	3,816.538	86.81	12.522	29.6984848	10
150	1.5	1200	80.00	2,368.421	1,717.442	111.25	10.844	25.7196423	10
125	4	600	80.00	5,263.158	3,816.538	104.14	9.899	23.4787138	10
100	2.3	720	80.00	2,421.053	1,755.607	112.03	8.854	21.0000000	10
100	5	514	80.00	5,263.158	3,816.538	117.92	8.854	21.0000000	10
90	2.5	514	80.00	2,368.421	1,717.442	90.24	8.854	19.9223493	10
80	3	514	80.00	2,526.316	1,831.938	107.98	7.920	18.7829710	10
70	3	450	80.00	2,210.526	1,602.946	104.49	7.408	17.5698606	10
60	4	360	80.00	2,526.316	1,831.938	108.36	6.859	16.2665301	10
60	6	360	80.00	3,789.474	2,747.907	132.71	6.859	16.2665301	10
50	4.5	300	80.00	2,368.421	1,717.442	109.81	6.261	14.8492424	10

Tabla 3.11 Cálculo de selección de una turbina Francis Normal.

H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Tfrancis	Pot. Francis (hp)	Pot. Francis (kW)	Ns Francis	Vs Francis C3 (m/seg)	VTg. Francis (m/seg)	Francis Normal
150	0.42	2700	80.00	663.158	480.884	132.45	13.282	25.7196423	10
150	0.7	2500	80.00	1,105.263	801.473	158.33	13.282	36.0070000	10
125	0.35	2500	80.00	460.526	333.947	128.36	12.124	32.8700000	10
125	0.9	900	80.00	1,184.211	858.721	74.10	12.124	32.8700000	10
100	0.25	1200	80.00	263.158	190.827	61.56	10.844	29.4000000	10
100	1	1200	80.00	1,052.632	763.308	123.12	10.844	29.4000000	10
90	0.25	1200	80.00	236.842	171.744	66.62	10.288	27.8910000	10
90	1	1800	80.00	947.368	686.977	199.86	10.288	27.8910000	10
80	0.28	900	80.00	235.789	170.981	57.76	9.699	26.2960000	10
80	1.3	1200	80.00	1,094.737	793.840	165.95	9.699	26.2960000	10
70	0.4	1200	80.00	294.737	213.726	101.75	9.073	24.5980000	10
70	0.9	900	80.00	663.158	480.884	114.47	9.073	24.5980000	10

Tabla 3.12 Cálculo de selección de una turbina Francis Rápida.

H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Tfrancis	Pot. Francis (hp)	Pot. Francis (kW)	Ns Francis	Vs Francis C3 (m/seg)	VTg. Francis (m/seg)	Francis Rápida
60	8	600	80.00	5,052.632	3,663.876	255.40	11.879	24.787	10
50	6	514	80.00	3,157.895	2,289.923	217.24	10.844	22.627	10
50	10	514	80.00	5,263.158	3,816.538	280.46	10.844	22.627	10
40	5	450	80.00	2,105.263	1,526.615	205.25	9,699	20,239	10
40	12	360	80.00	5,052.632	3,663.876	254.38	9,699	20,239	10
35	3.5	514	80.00	1,289.474	935.052	216.81	9.073	18.931	10
35	10	225	80.00	3,684.211	2,671.576	160.42	9.073	18.931	10
30	3	450	80.00	947.368	686.977	197.27	8.400	17.527	10
30	7	514	80.00	2,210.526	1,602.946	344.20	8.400	17.527	10
25	2.0	600	80.00	526.316	381.654	246.23	7.668	16.000	10
25	4.0	600	80.00	1,052.632	763.308	348.23	7.668	16.000	10
20	1.5	514	80.00	315.789	228.992	215.96	6.859	14.311	10

Tabla 3.13 Cálculo de selección de una turbina Francis Extra Rápida.

H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Tfrancis	Pot. Francis (hp)	Pot. Francis (kW)	Ns Francis	Vs Francis C3 (m/seg)	VTg. Francis (m/seg)	Francis Rápida
9	0.7	720	80.00	66.316	48.088	376.13	6.641	10,500	10
9	2.0	450	80.00	189.474	137.395	397.36	6,641	10,500	10
8	0.7	720	80.00	58.947	42.745	410.87	6.261	9.899	10
8	1.5	450	80.00	126.316	91.597	375.91	6.261	9.899	10
7	0.8	600	80.00	58.947	42.745	404.59	5,857	9,260	10
7	2	300	80.00	147.368	106.863	319.85	5,857	9,260	10
6	0.9	450	80.00	56.842	41.219	361.29	5,422	8,573	10
6	1.8	360	80.00	113.684	82.437	408.76	5,422	8,573	10
5	1.3	327	80.00	68.421	49.615	361.77	4.950	7.826	10
5	1.3	400	80.00	68.421	49.615	442.53	4.950	7.826	10

Tabla 3.14 Cálculo de selección de una turbina Hélice.

H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Hélice	Pot. Hélice (hp)	Pot. Hélice (kW)	Ns Hélice	Vs Hélice C3 (m/seg)	VTg. Hélice U1 (m/seg)	Hélice
10	4.0	514	85.00	447.368	324.406	611.36	7.67	16.01	10
10	5.0	514	85.00	559.211	405.507	683.52	7.67	16.27	10
9	3.5	450	85.00	352.303	255.469	541.84	7.27	14.87	10
9	4.0	600	85.00	402.632	291.965	772.33	7.27	15.67	10
9	5.0	450	85.00	503.289	364.956	647.62	7.27	15.32	10
8	3.0	450	85.00	268.421	194.643	547.97	6.86	14.05	10
8	4.0	600	85.00	357.895	259.525	843.66	6.86	14.91	10
8	6.0	400	85.00	536.842	389.287	688.84	6.86	14.57	10
7	2.5	514	85.00	195.724	141.927	631.56	6.42	13.46	10
7	3.0	450	85.00	234.868	170.313	605.69	6.42	13.37	10
7	5.0	400	85.00	391.447	283.855	695.06	6.42	13.64	10

Tabla 3.15 Cálculo de selección de una turbina Hélice.

H (m)	Q (m3/seg)	Vel.Rot. n(rpm)	Ef. Hélice	Pot. Hélice (hp)	Pot. Hélice (kW)	Ns Hélice	Vs Hélice C3 (m/seg)	VTg. Hélice U1 (m/seg)	Hélice
7	6.0	360	85.00	469.737	340.626	685.26	6.42	13.62	10
6	2.0	600	85.00	134.211	97.322	740.21	5.94	12.73	10
6	7.0	327	85.00	469.737	340.626	754.72	5.94	12.76	10
6	8.0	400	85.00	536.842	389.287	986.95	5.94	13.09	10
5	10.0	277	85.00	559.211	405.507	876.10	5.42	11.83	10
3	1.5	360	85.00	50.329	36.496	646.86	4.20	8.84	10
3	10.0	225	85.00	335.526	243.304	1043.86	4.20	9.29	10

Tabla 3.16 Cálculo de selección de una turbina Kaplan.

H (m)	Q (m3/seg)	Vel. Rot. n(rpm)	Ef. TKaplan	Pot. Kaplan (hp)	Pot. Kaplan (kW)	Ns Kaplan	C3 Kaplan (m/seg)	VTg. Kaplan (m/seg)	Kaplan
35	12	514	80,00	4421.053	3205.892	401.46	14,35	27,24	10
35	13	450	80,00	4789.474	3473.049	365.82	14,35	26,38	10
30	10	450	80,00	3157.895	2289.923	360.17	13,28	24,28	10
30	16	300	80,00	5052.632	3663.876	303.72	13,28	22,45	10
25	7	450	80,00	1842.105	1335.788	345.50	12,12	21,80	10
25	20	450	80,00	5263.158	3816.538	584.00	12,12	25,12	10
20	6	450	80,00	1263.158	915.969	378.14	10,84	20,18	10
15	9	327	80,00	1421.053	1030.465	417.58	9,39	18,05	10
12	10	327	80,00	1263.158	915.969	520.36	8,40	17,04	10
10	7	300	80,00	736.842	534.315	457.94	7,67	15,11	10
9	7	300	80,00	663.158	480.884	495.59	7,67	14,60	10
8.5	10	240	80,00	894.737	648.811	494.64	7,07	14,19	10
8	8	300	80,00	673.684	488.517	578.74	6,86	14,19	10

Tabla 3.17 Cálculo de selección de una turbina Michell.

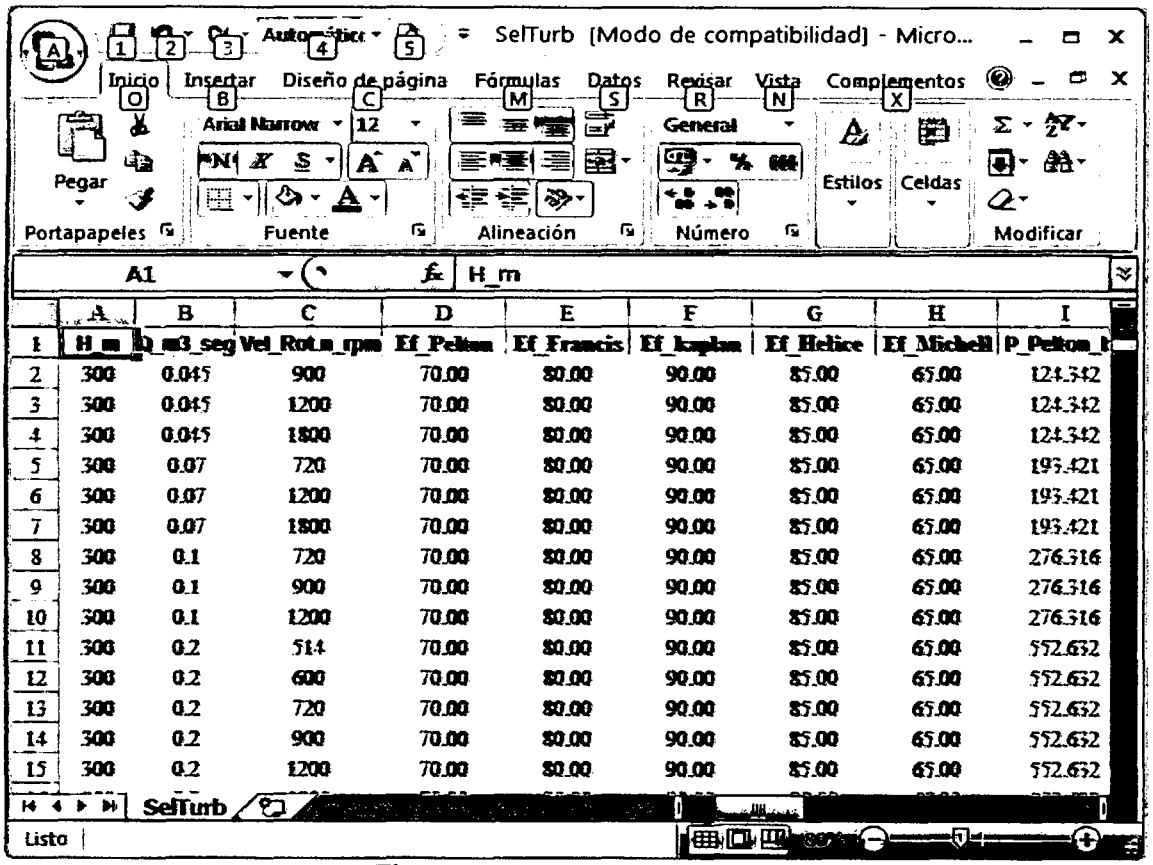
H (m)	Q (m3/seg)	Vel. Rot. n(rpm)	Ef. T.Michell	Pot. Michell (hp)	Pot. Michell (kW)	Ns Michell	C1 Michell (m/seg)	VTg. Michell (m/seg)	Michell
90	0.22	1200	65.00	169.342	122.797	56.33	39.90	19.922	10
80	0.25	1800	65.00	171.053	124.037	98.40	37.62	18.783	10
70	0.15	1200	65.00	89.803	65.120	56.16	35.19	17.570	10
70	0.3	1200	65.00	179.605	130.239	79.43	35.19	17.570	10
60	0.12	1800	65.00	61.579	44.653	84.59	32.58	16.267	10
50	0.11	1800	65.00	47.039	34.110	92.85	29.74	14.849	10
50	0.41	1200	65.00	175.329	127.138	119.51	29.74	14.849	10
45	0.47	900	65.00	180.888	131.170	103.86	28.21	14.087	10
40	0.51	900	65.00	174.474	126.518	118.18	26.60	13.282	10
35	0.6	360	65.00	179.605	130.239	56.67	24.88	12.424	10
30	0.2	600	65.00	51.316	37.211	61.22	23.04	11.502	10
30	0.5	450	65.00	128.289	93.028	72.59	23.04	11.502	10
25	0.7	600	65.00	149.671	108.533	131.31	21.03	10.500	10
20	0.27	360	65.00	46.184	33.490	57.84	18.81	9.391	10
15	0.5	514	65.00	64.145	46.514	139.45	16.29	8.133	10

3.2.3.1 Análisis.

La investigación realizada en el presente trabajo es de calcular y seleccionar el tipo de turbina a utilizar en una mini central, para ciertas condiciones de entrada: como la altura y el caudal. Se requiere que este sistema razone y aprenda basándose en analogías: RAZONAMIENTO BASADO EN CASOS. Se ha utilizado información de diferentes turbinas, seleccionados y calculadas de acuerdo al rango de valores característicos de las turbinas hidráulicas señalados en la Tabla 3.6 y en el gráfico de la figura 3.2. Los pasos a seguir en el NeuroShell 2 son los siguientes:

1. **Datagrid del NeuroShell 2**

Los datos proporcionados por los expertos se tabulan en una hoja del cálculo con formato Excel o utilizamos el Datagrid del NeuroShell 2 como el mecanismo de introducción de datos; tal como se observa en la figura 3.15.



	A	B	C	D	E	F	G	H	I
1	H_m	Q_m3_seg	Vel_Rot.n_rpm	Ef_Pelton	Ef_Francis	Ef_kaplan	Ef_Helice	Ef_Michell	P_Pelton_k
2	300	0.045	900	70.00	80.00	90.00	85.00	65.00	124.342
3	300	0.045	1200	70.00	80.00	90.00	85.00	65.00	124.342
4	300	0.045	1800	70.00	80.00	90.00	85.00	65.00	124.342
5	300	0.07	720	70.00	80.00	90.00	85.00	65.00	193.421
6	300	0.07	1200	70.00	80.00	90.00	85.00	65.00	193.421
7	300	0.07	1800	70.00	80.00	90.00	85.00	65.00	193.421
8	300	0.1	720	70.00	80.00	90.00	85.00	65.00	276.316
9	300	0.1	900	70.00	80.00	90.00	85.00	65.00	276.316
10	300	0.1	1200	70.00	80.00	90.00	85.00	65.00	276.316
11	300	0.2	514	70.00	80.00	90.00	85.00	65.00	552.632
12	300	0.2	600	70.00	80.00	90.00	85.00	65.00	552.632
13	300	0.2	720	70.00	80.00	90.00	85.00	65.00	552.632
14	300	0.2	900	70.00	80.00	90.00	85.00	65.00	552.632
15	300	0.2	1200	70.00	80.00	90.00	85.00	65.00	552.632

Fig. 3.14 Tabulación de datos.

2. Archivo DSC

Crear el archivo **DSC**, como se observa en la figura 3.15.

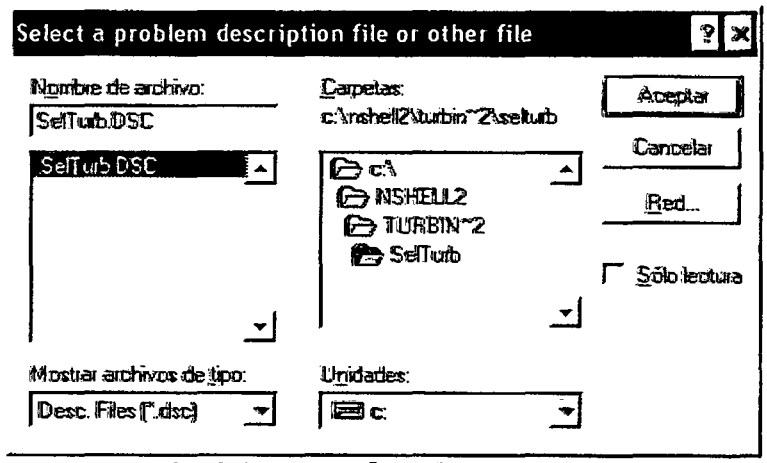


Fig. 3.15 Creación del archivo dsc.

3. File Import.

Seleccionar de ***"file import"*** el archivo ***SelTurb.wk1***, mostrados en las figuras 3.16 y 3.17.

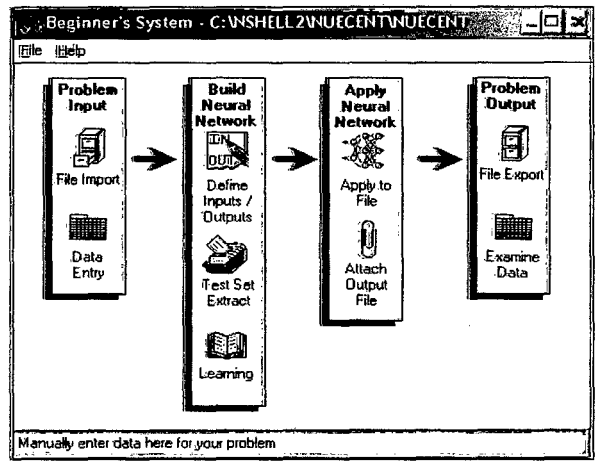


Fig. 3.16 Seleccionar File Import.

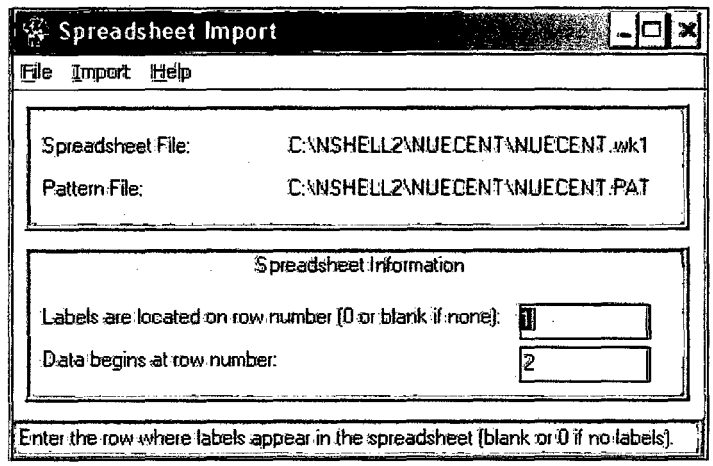


Fig. 3.17 Seleccionar el archivo nuecent.wk1.

4. Data entry

Seleccionar en el menú del NeuroShell2, ***"Data entry"*** mostrada en la figura 3.18 y generamos el archivo nuecent.pat, mostrado en la figura 3.19.

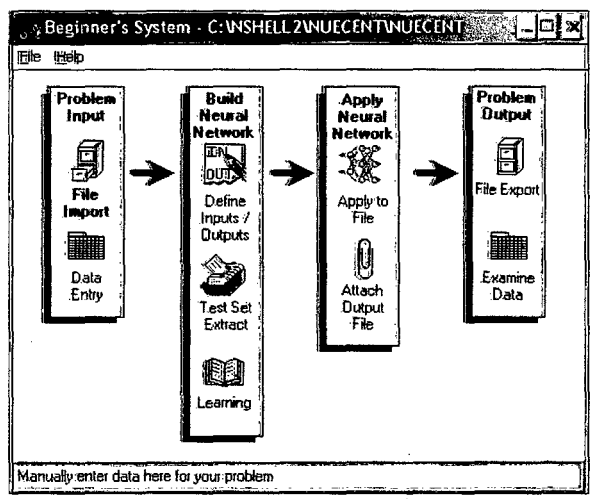
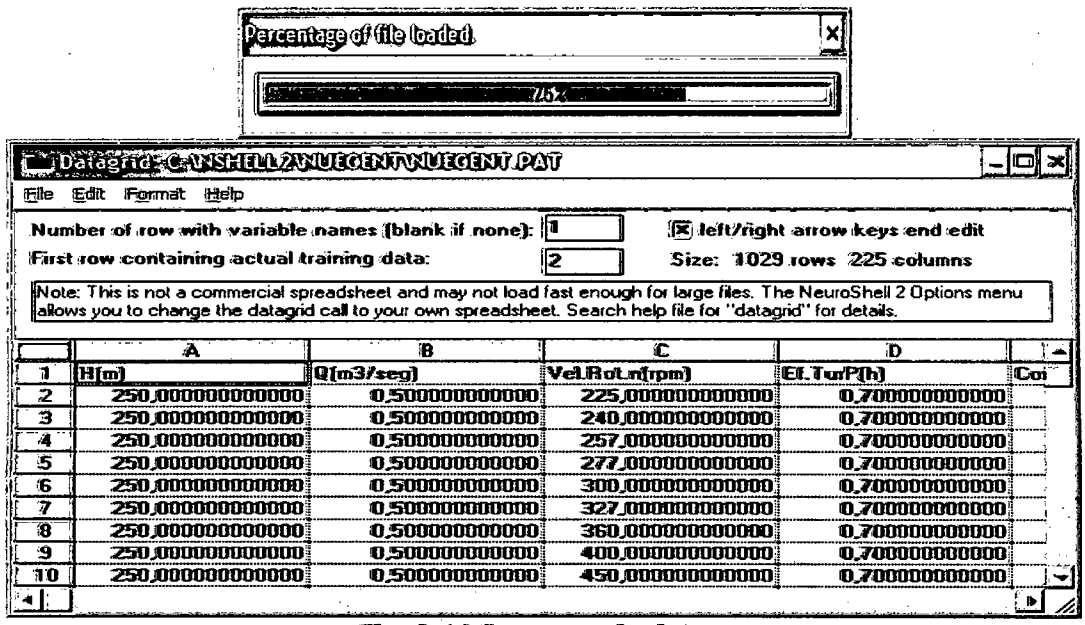


Fig. 3.18 Seleccionar Data Entry.

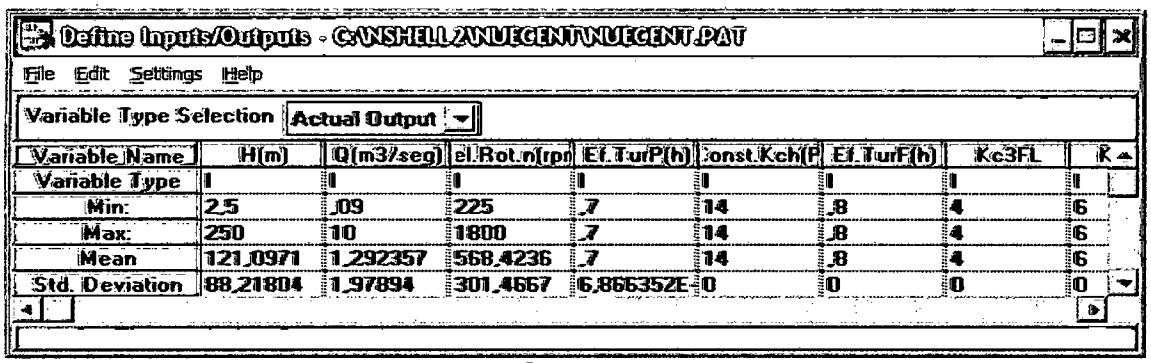


	A	B	C	D	
1	H(m)	Q(m3/seg)	Vel.Rot.n(rpm)	Ef.TurP(h)	Cor
2	250,0000000000000	0,5000000000000	225,0000000000000	0,7000000000000	
3	250,0000000000000	0,5000000000000	240,0000000000000	0,7000000000000	
4	250,0000000000000	0,5000000000000	257,0000000000000	0,7000000000000	
5	250,0000000000000	0,5000000000000	277,0000000000000	0,7000000000000	
6	250,0000000000000	0,5000000000000	300,0000000000000	0,7000000000000	
7	250,0000000000000	0,5000000000000	327,0000000000000	0,7000000000000	
8	250,0000000000000	0,5000000000000	360,0000000000000	0,7000000000000	
9	250,0000000000000	0,5000000000000	400,0000000000000	0,7000000000000	
10	250,0000000000000	0,5000000000000	450,0000000000000	0,7000000000000	

Fig. 3.19 Ingresos de datos.

5. Input y Output

Seleccionar las entradas (***input***) y las salidas (***output***), como se observa en las figuras **3.20** y **3.21**.



Variable Name	H(m)	Q(m3/seg)	el.Rot.n(rpi	Ef.TurP(h)	onst.Kch(P	Ef.TurF(h)	Kc3FL	K
Variable Type	I	I	I	I	I	I	I	I
Min:	2,5	,09	225	,7	14	,8	4	6
Max:	250	10	1800	,7	14	,8	4	6
Mean	121,0971	1,292357	568,4236	,7	14	,8	4	6
Std. Deviation	88,21804	1,97894	301,4667	6,866352E-	0	0	0	0

Fig. 3.20 Selección de entradas y salidas.

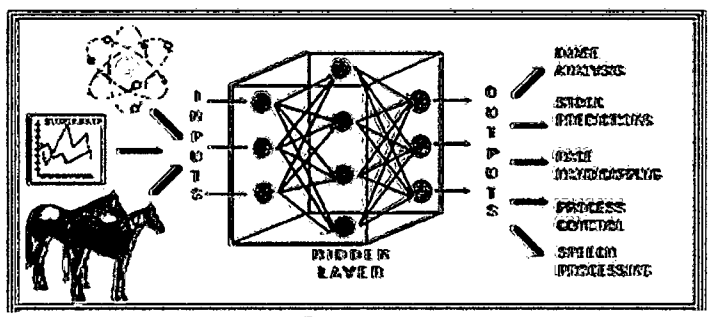


Fig. 3.21 Selección de entradas y salidas.

6. Test Set Extraction

Realizar la importación de los datos, mostrado en las figuras 3.22 y 3.23.

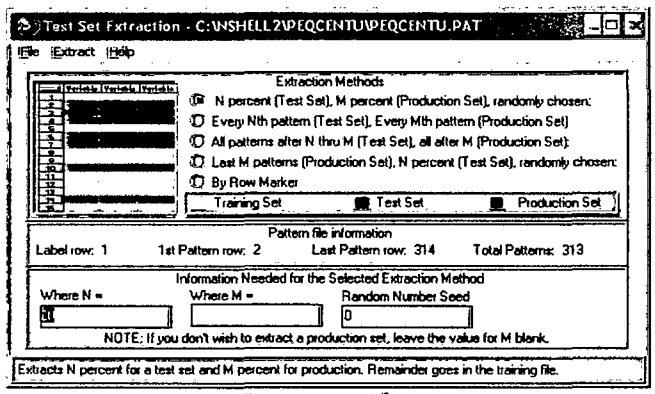


Fig. 3.22 Importación de datos.

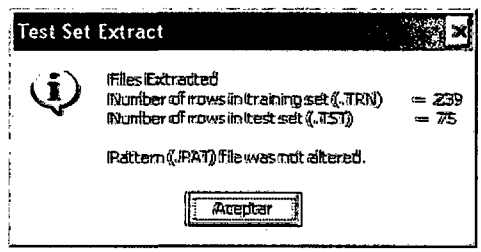


Fig. 3.23 Archivos extraídos.

7. Train

Realizar el entrenamiento utilizando 30 neuronas, tal como se observan en las figuras 3.24, 3.25 y 3.26; durante 10 minutos.

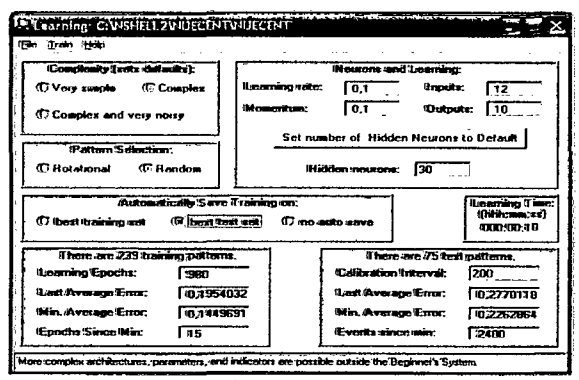


Fig. 3.24 Inicio del entrenamiento.

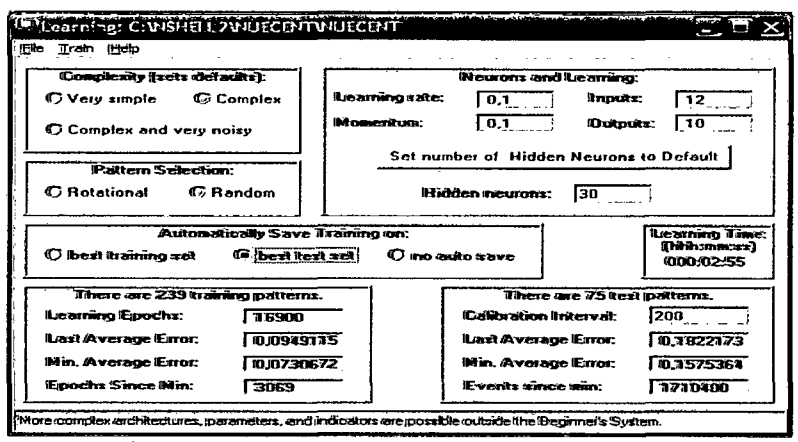


Fig. 3.25 Entrenamiento a los 5 minutos.

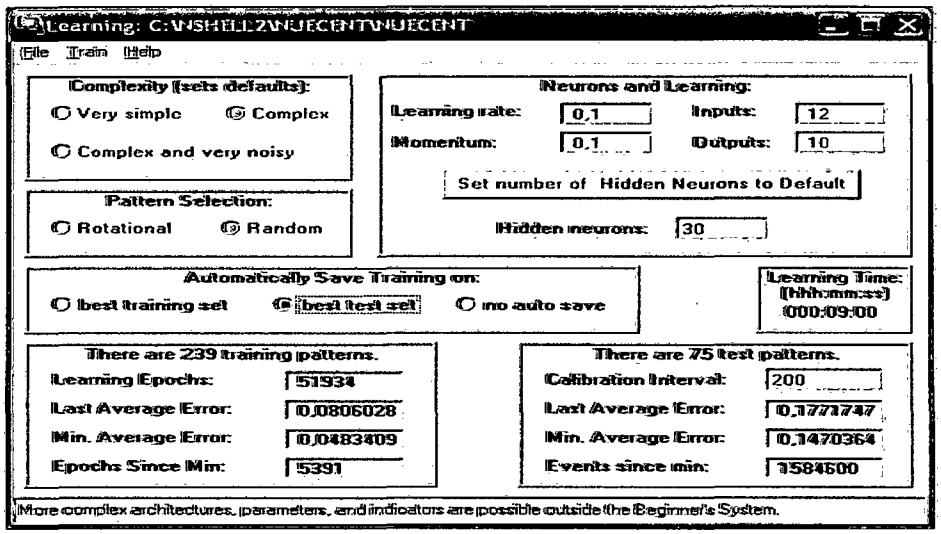


Fig. 3.26 Entrenamiento a los 9 minutos.

8. Run

Luego iniciamos la salida del proceso mostrado en la figura 3.27.

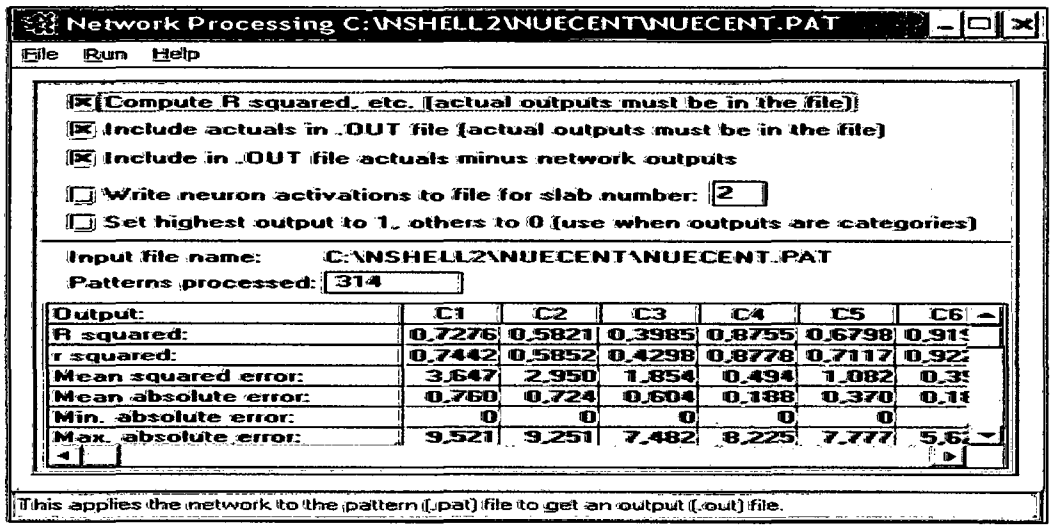


Fig. 3.27 Correr el proceso.

9. Attach

Adjuntar los archivos de lado a lado, mostrado en la figura 3.28.

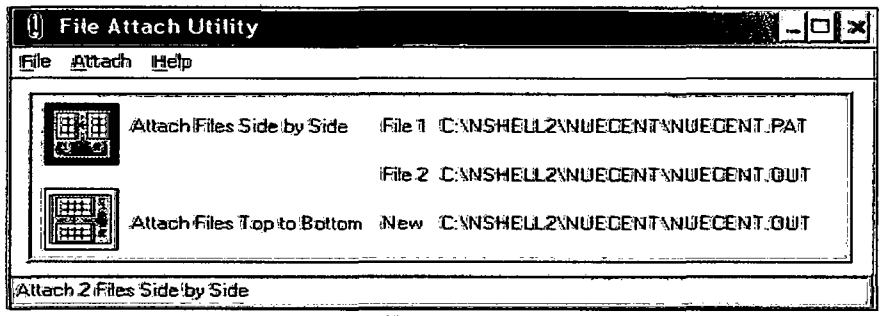


Fig. 3.28 Adjuntar los archivos.

10. Spreadsheet Export

Realizar la exportación de los archivos, mostrado en la figura 3.29.

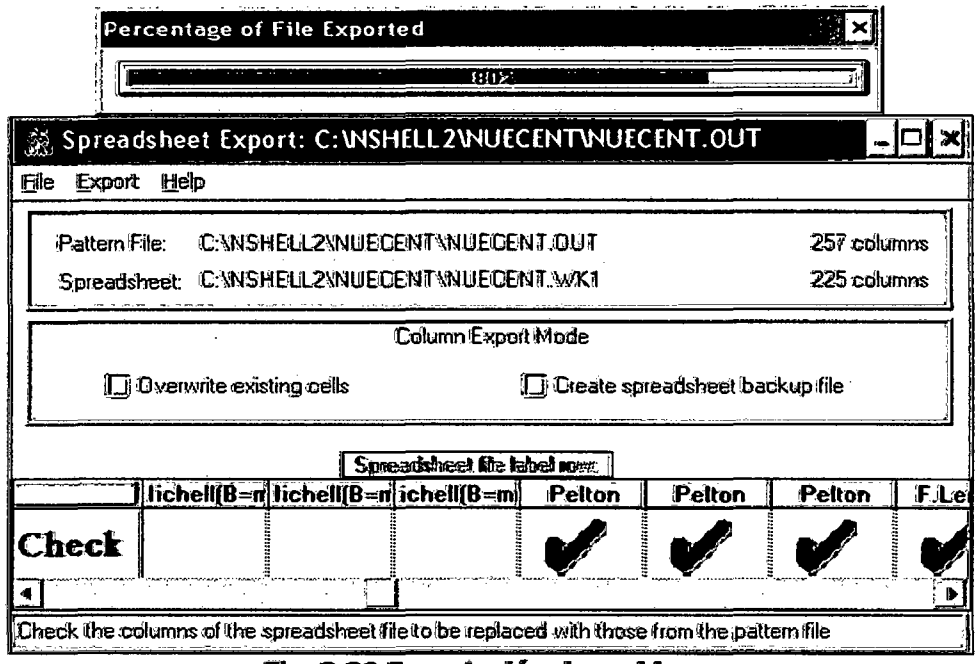


Fig. 3.29 Exportación de archivos.

11. Datagrid

Examinar los archivos de la data, mostrado en la figura 3.30.

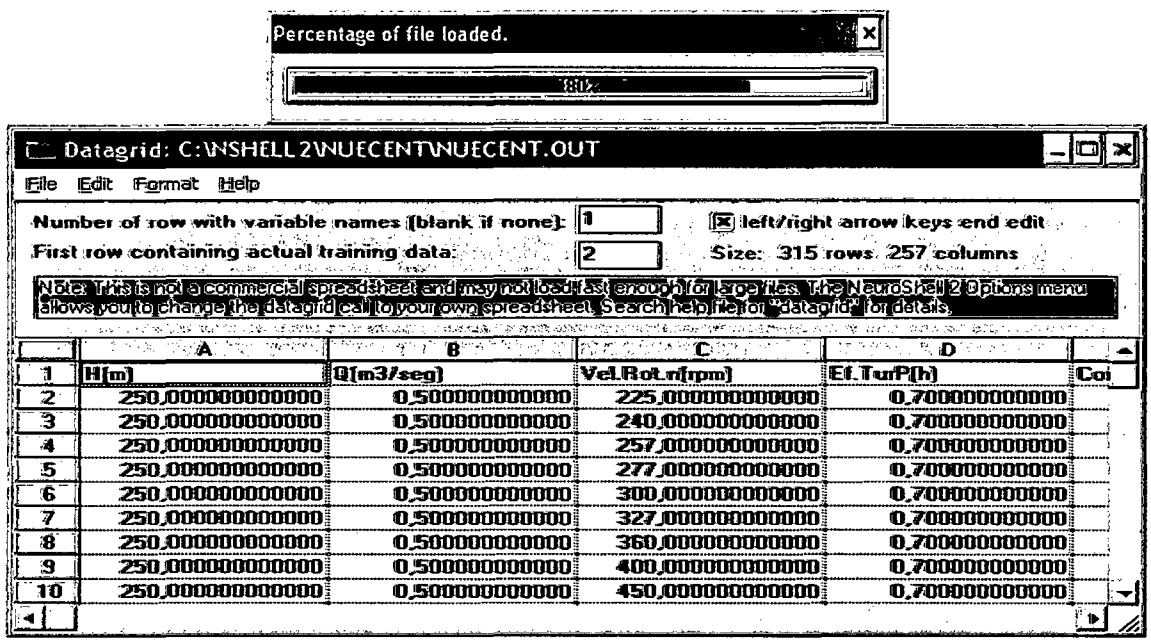


Fig. 3.30 Examinar los archivos.

12. Generate Runtime Systems

Finalmente crear las funciones especializadas, mostrado en la figura 3.31.

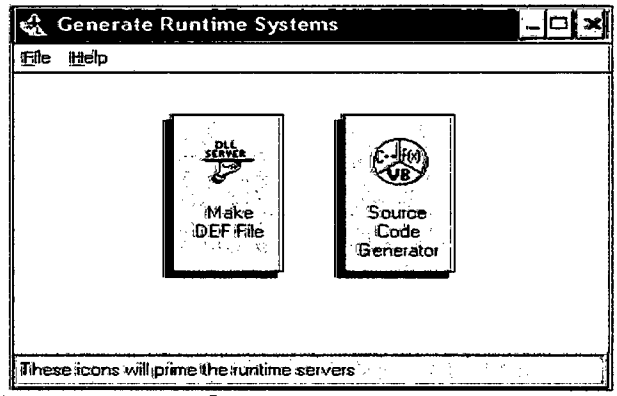


Fig. 3.31 Creación de funciones especializadas.

3.2.4 Pantalla del menú de ingreso de datos: MERNA

Al tener la función generada por NeuroShell 2 utilizamos un software vigente, este puede ser C#, Net Bean 5.5 y en base a la ayuda de estos programas de programación; se desarrolla un formulario de ingresos de datos y obtención de resultados, la cual mostramos en la figura 3.32 y en la figura 3.33 el diagrama de flujo del proceso realizado.

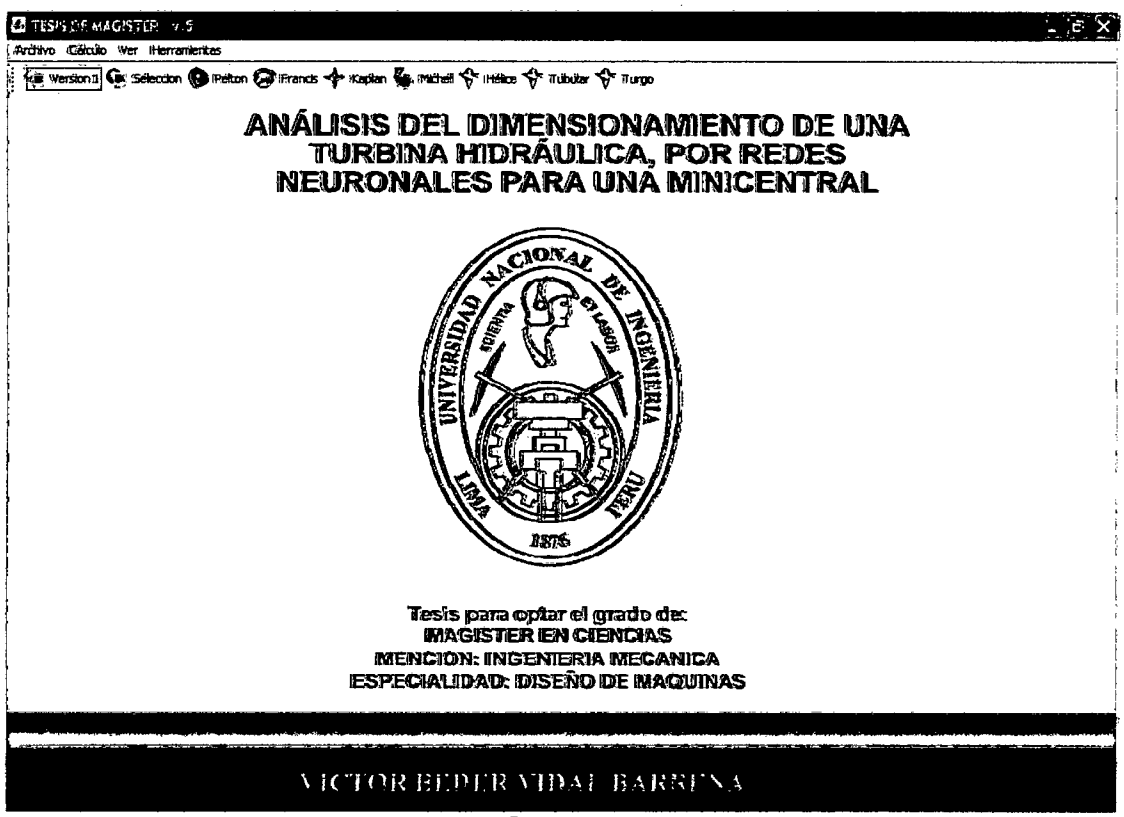


Fig. 3.32 Menú de ingreso de datos.

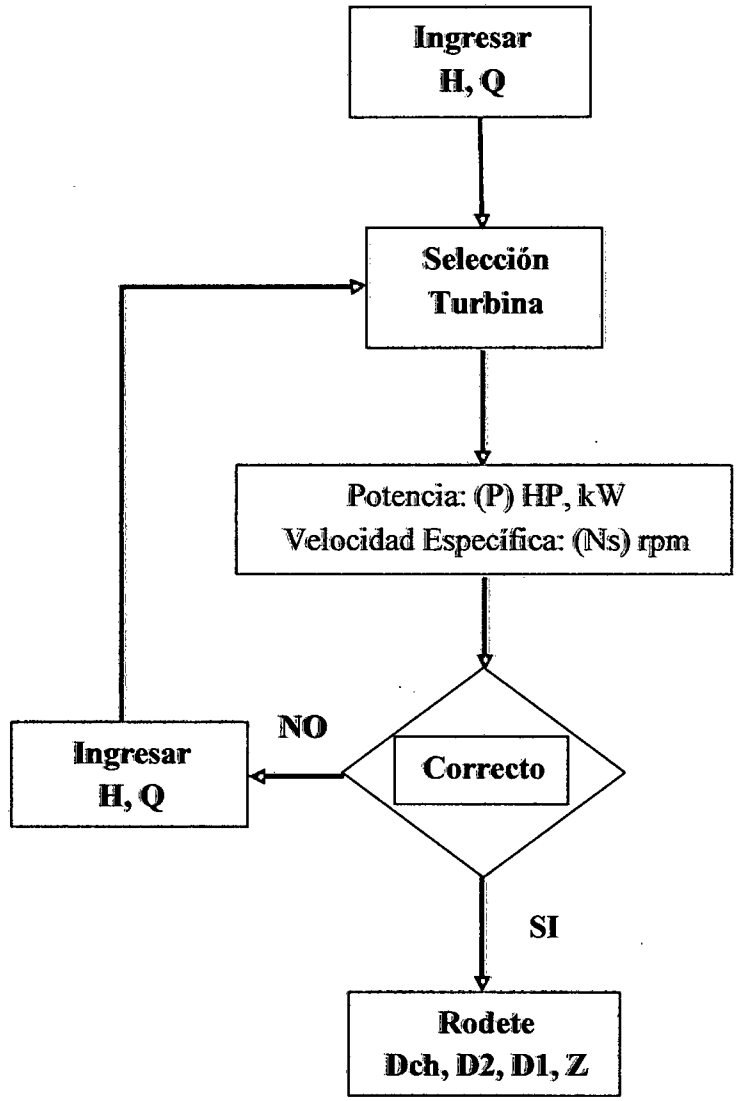


Fig. 3.33 Diagrama de flujo del proceso realizado.

3.2.4.1 Selección y diseño de una turbina Francis Normal

Hacer doble clic en el ítem 1028, ingresa como datos **H = 20m, Q= 1.2m^3/seg**, el programa **MERNA** selecciona Turbina Francis Normal, como se observa en la figura 3.34. La función utilizada por el NeuroShell 2 es la siguiente: FN=SI(Y((Altura)H=20;(Caudal)Q>=1,2;Q<=1,4;(Potencia)P>=150;P<=750;(Velocidad Específica)Ns>=125;Ns<=225);10;0)

En el menú de turbina Francis y en el ítem 1028, haciendo doble clic, el programa MERNA muestra las dimensiones de la turbina Francis Normal, seleccionada en el paso anterior; tal como se observa en la figura 3.35.

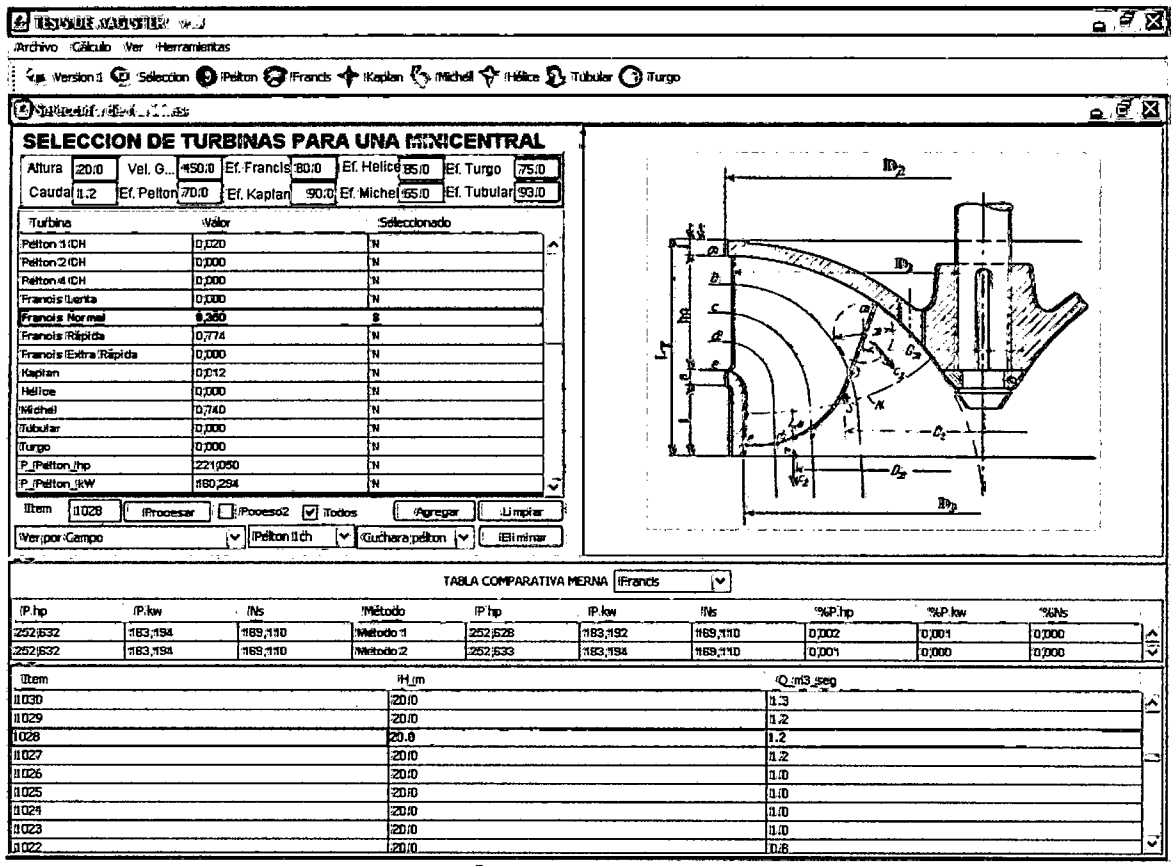


Fig. 3.34 Selección de una turbina Francis normal.

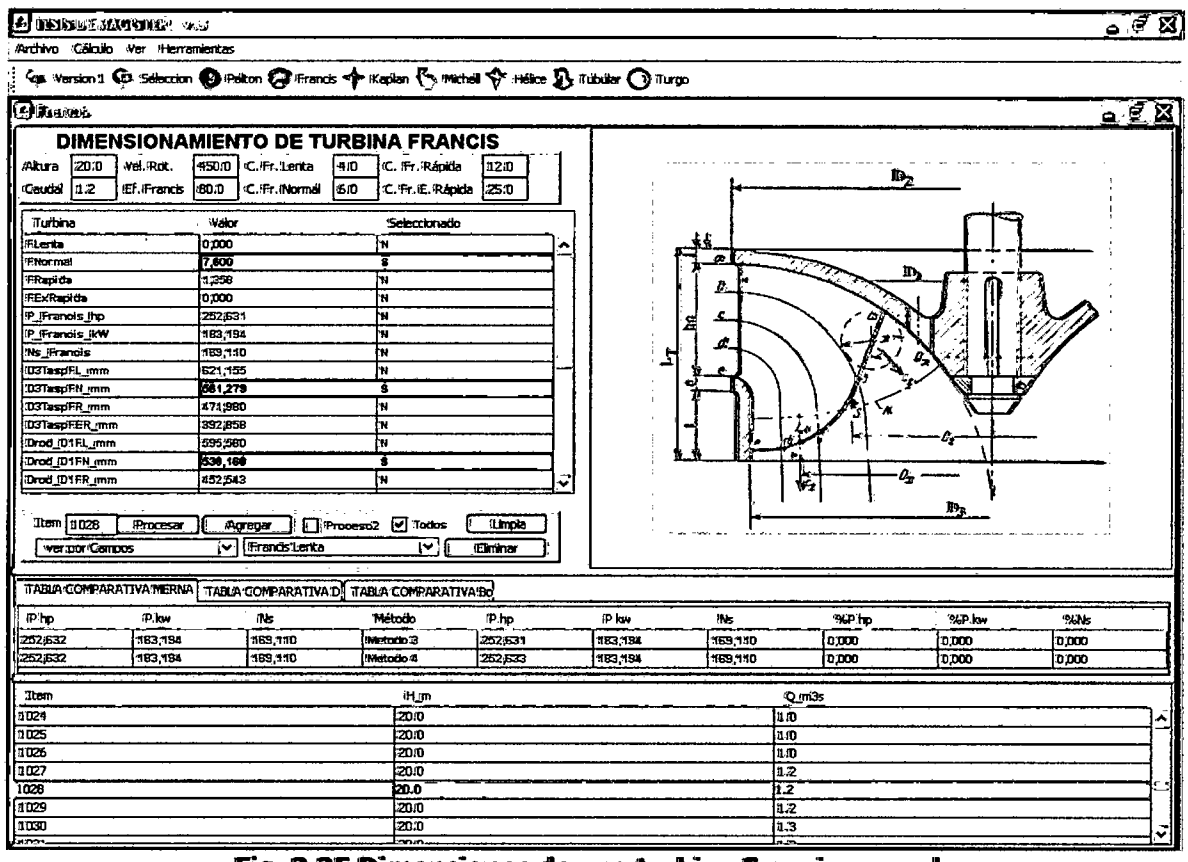


Fig. 3.35 Dimensiones de una turbina Francis normal.

3.2.4.2 Selección y diseño de una turbina Michell

Ingresar los datos del ítem 484, haciendo doble clic, **H = 50 m, Q= 0.180 m^3/seg**, el programa **MERNA** selecciona Turbina Michell, como se observa en la figura 3.36; el Manual de Mini y Micro centrales Hidráulicas **[32]**, en la página 146 selecciona la misma turbina. La función utilizada por el NeuroShell 2 es la siguiente:

M=SI(Y((altura)H=50;(Caudal)Q>=0,12;Q<=0,4;(Potencia)P>=30;P<=150;
(Velocidad Específica)Ns>=40; Ns<=160);10;0)

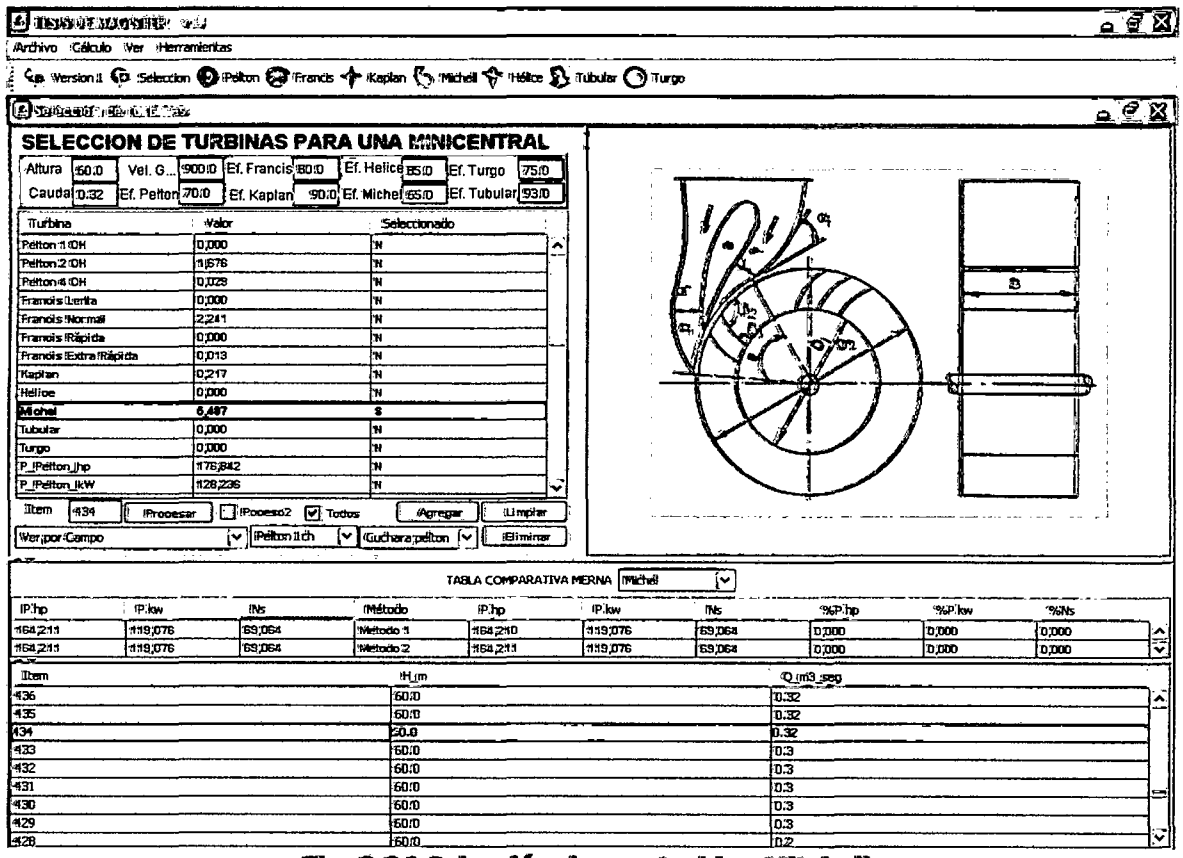


Fig. 3.36 Selección de una turbina Mitchell.

En el menú de turbina Michell y en el ítem 484, haciendo doble clic, el programa MERNA muestra las dimensiones de la turbina Michell, seleccionada en el paso anterior; tal como se observa en la figura 3.37. Las funciones utilizadas son las siguientes:

Vsal_C1_ms = E342*RAIZ (2*9,8*A342)

Dchorro_mm = RAIZ((4*Q)/(C1*PI()))*(POTENCIA(10;3))

D2_Rd_mm = (1000*Ns*RAIZ(H))/K

D1_Rd_mm = 0,66*D2

Dmed_mm = (D2+D1)/2

Arod_B_mm_teta_60 = (98,8*Q*POTENCIA(1000;2))/(D2*RAIZ(H)*θ1)

Arod_B_mm_teta_90 = (98,8*Q*POTENCIA(1000;2))/(D2*RAIZ(H)*θ2)

Arod_B_mm_teta_120 = (98,8*Q*POTENCIA(1000;2))/(D2*RAIZ(H)*θ3)

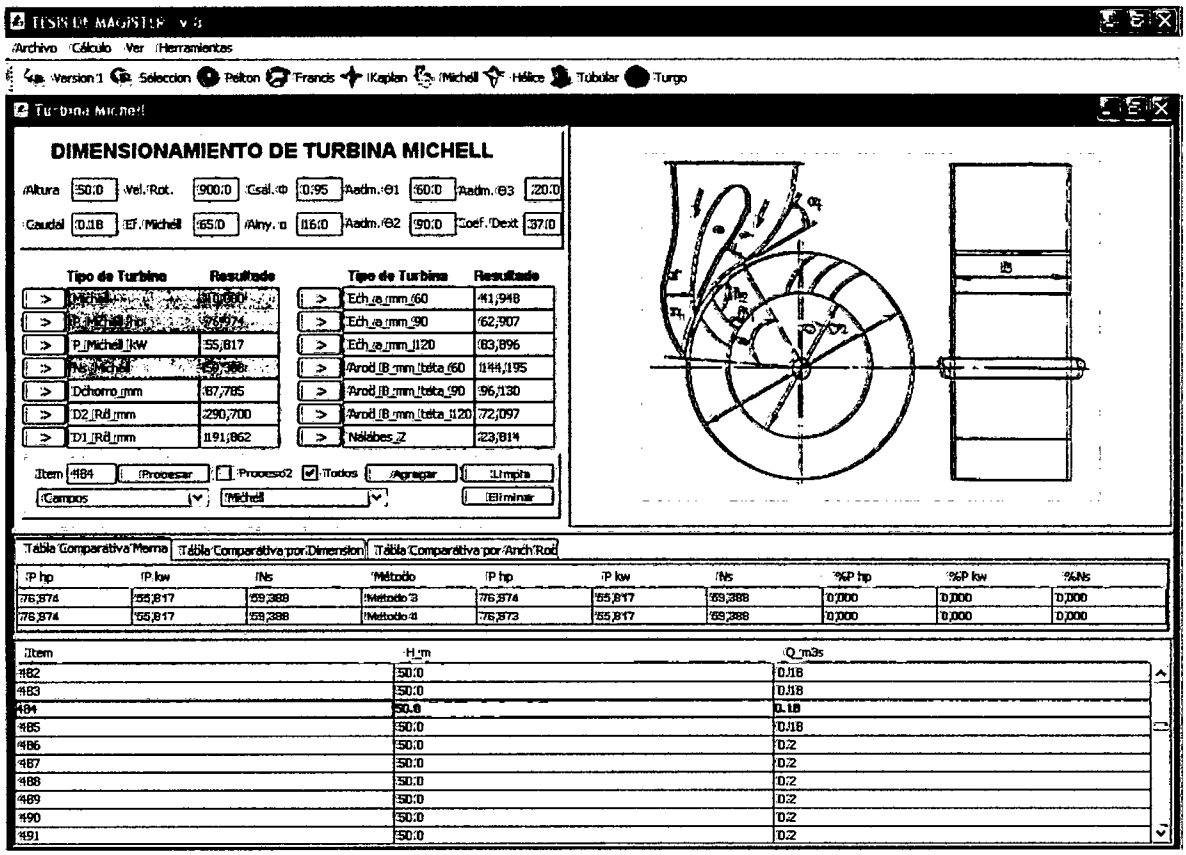


Fig. 3.37 Dimensiones de una turbina Michell.

3.2.4.3 Selección y diseño de una turbina Pelton 1ch

Haciendo doble clic en el ítem 93, ingresamos los datos **H= 200 m, Q= 0.1 m^3/seg**, el programa **MERNA** selecciona Turbina Pelton 1ch, como se observa en la figura 3.38. La función utilizada por el NeuroShell 2 es la siguiente:

Pelton_1ch = SI (Y (H = 200; Q>= 0,03; Q<= 0,24; P>=35; P<=390; Ns>=10; Ns<=30) ;10;0)

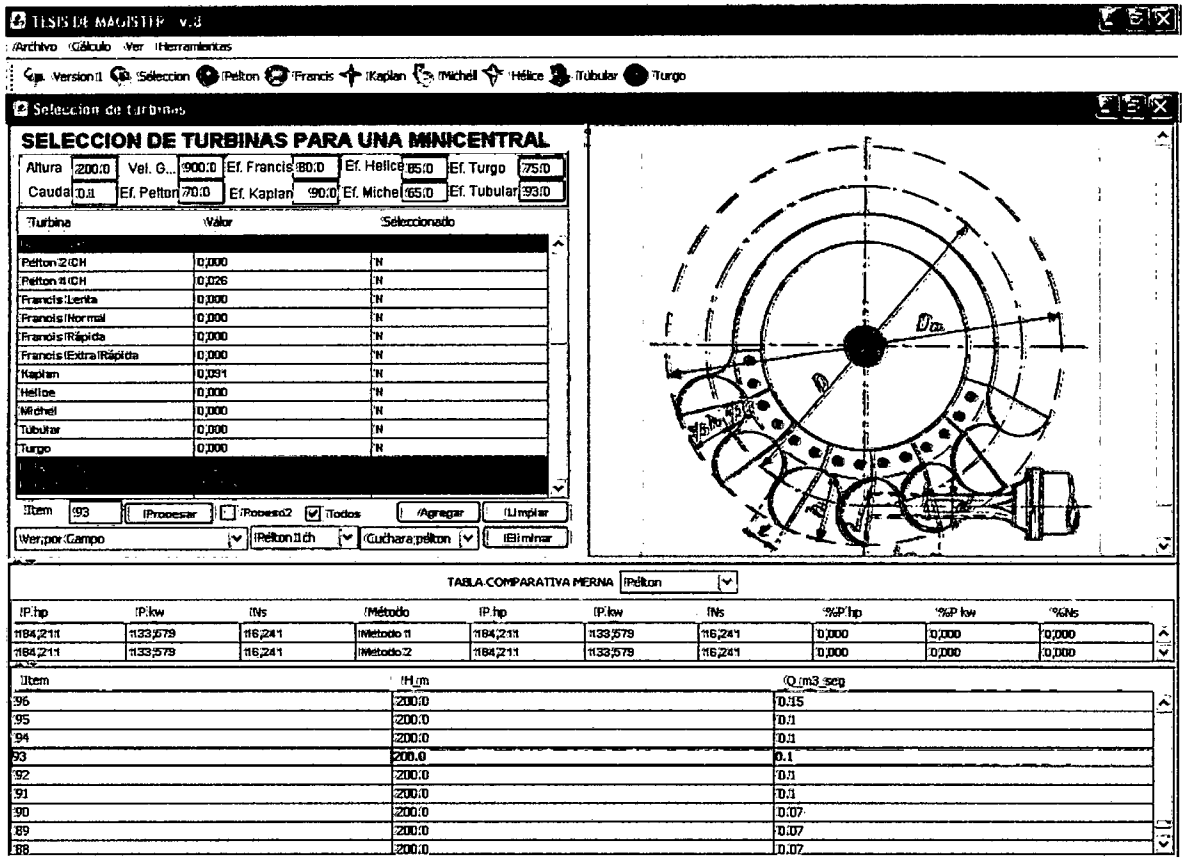


Figura 3.38 Selección de una turbina Pelton 1ch.

En el menú de turbina Pelton y en el ítem 93, haciendo doble clic, el programa MERNA muestra las dimensiones de la turbina Pelton 1ch, seleccionada en el paso anterior; tal como se observa en la figura 3.39. Las funciones utilizadas son las siguientes:

dch_Pelton_1ch = (550)*RAIZ((Q/(RAIZ(H))))

Dr_Pelton_mm = ((K*1000)*RAIZ(H))/N

Vr_Pelton_rpm=1000*(60*Vt)/(Dr*PI())

Ancho_b_P1ch=3,75*dch

Alto_h_P1ch =3,5*dch

Espesor_t_P1ch=1,5*dch

D3_P1Chmm=Dr+2*((3*h)/5)

Zc_Pelton1Ch=0,5*(Dr/dch)+k

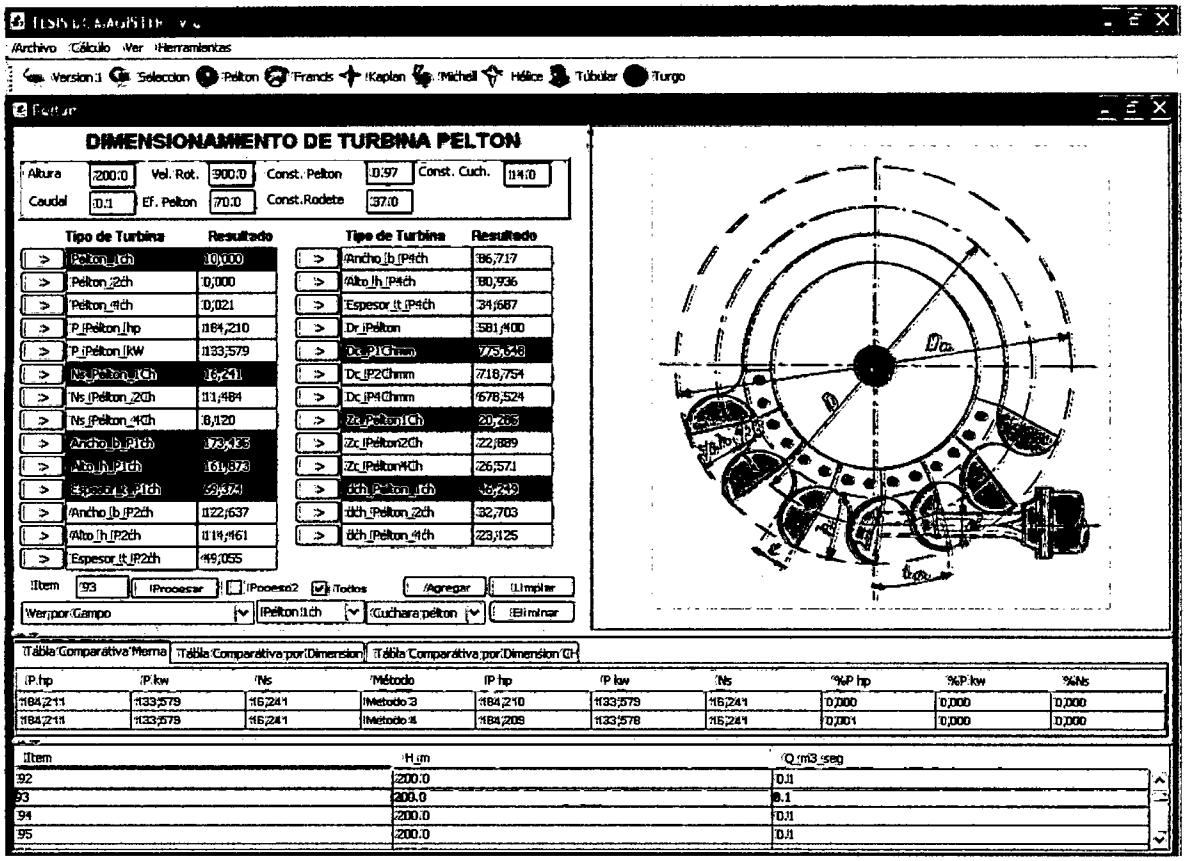


Fig. 3.39 Dimensiones de una turbina Pelton 1ch.

3.2.4.4 Selección y diseño de una turbina Kaplan

Haciendo doble clic en el ítem 984, ingresamos los datos **H= 25m, Q= 10 m^3/seg**, el programa **MERNA** selecciona Turbina Kaplan, como se observa en la figura 3.40. La función utilizada por el NeuroShell 2 es la siguiente:

KAPLAN =SI (Y (H = 25; Q >= 6,5; Q <= 20; P >=1100; P<=4000; Ns>=300; Ns<=600); 10;0)

En el menú de turbina Kaplan y en el ítem 984, haciendo doble clic, el programa MERNA muestra las dimensiones de la turbina Kaplan, seleccionada en el paso anterior; tal como se observa en la figura 3.41. Las funciones utilizadas son las siguientes:

Tas_C3_m/s = RAIZ (2*9,8*0,3*H)
Dcho_D3_mm = 1000*RAIZ ((4*Q)/(PI()*C3))
Drod_D2_mm=0, 98*D3
Dcu_Dn_mm=D3*((0,25+(94,64/D3)))
Dme_D1 _mm=D2+(D2-Dc)/2
Arod_Bo_mm=(0,8*Q)/(0,9*D2*PI()*Cmo)*10^6

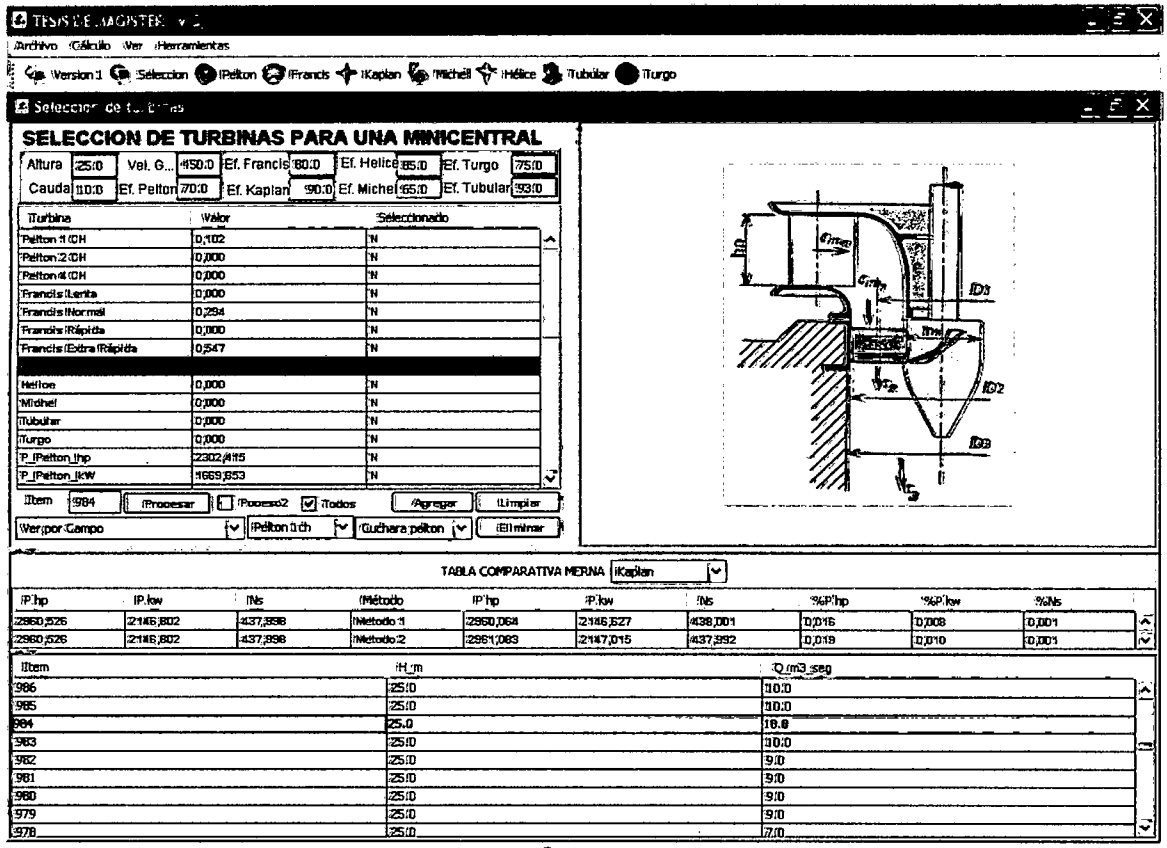


Figura 3.40 Selección de una turbina Kaplan.

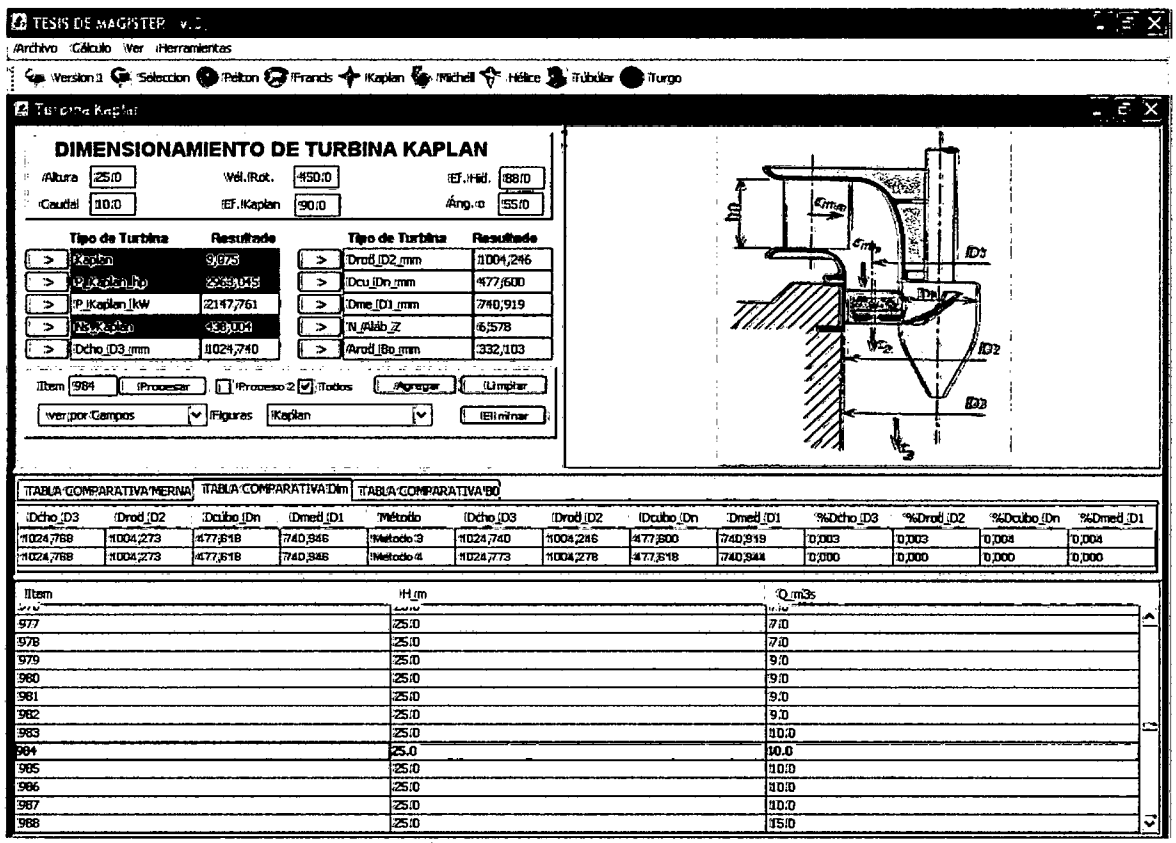


Fig. 3.41 Dimensiones de una turbina Kaplan.

CAPÍTULO 4

RESULTADOS

4.0 RESULTADOS

Los resultados obtenidos con la configuración que presenta el NeuroShell2 se pueden considerar como aceptables, posteriormente se probaron con diferentes funciones de activación y tiempo de aprendizaje (modificando los tiempos de parada); en el entrenamiento se observó que los valores que el NeuroShell2 presenta por defecto son los mejores. En la tabla de datos que se le proporcionaron a la red, hay valores que cumplen los requisitos de selección, como también hay otros que no cumplen estas condiciones; se procedió de esta forma para ver si la red era capaz de generalizar más y de ofrecer mejores resultados durante el proceso de entrenamiento. Al culminar el estudio tenemos los siguientes resultados:

4.1 RESULTADOS EN SELECCIÓN Y DISEÑO DE UNA TURBINA MICHELL.

Ingresar los datos de **H = 50 m, Q = 0,180 m^3/seg, V_r = 900 rpm** y eficiencia de la turbina **η = 65%.**

4.1.1 Método Tradicional

Utilizando las ecuaciones (2.1), (2.2) y (2.3) y comparando con los valores de la tabla 1.1 y en el gráfico de la figura 1.2, seleccionamos la turbina Michell.

Tabla 4.1 Selección de la Turbina Michell.

Altura	Caudal	Vel. Rot.	Efic. Turb.	Tipo Turbina		Pot. Turb.	Pot. Turb.	Vel. Espcif.	Turb. Selec.
H	Q	n	η	Nombre	Clase	P	P	Ns	
m	m³/seg	rpm				HP	KW	rpm	
50	0,180	900	0.70	Pelton	1 Ch	82,895	60,110	61,33	0
50	0,180	900	0,70	Pelton	2 Ch	82,895	60,110	43,58	0
50	0,180	900	0,70	Pelton	4 Ch	82,895	60,110	30,82	0
50	0,180	900	0,65	Michell		76,974	55,817	59,39	10
50	0,180	900	0,80	Francis	Lenta	94,737	68,698	65,89	0
50	0,180	900	0,80	Francis	Normal	94,737	68,698	65,89	0
50	0,180	900	0,80	Francis	Rápida	94,737	68,698	65,89	0
50	0,180	900	0,82	Kaplan		97,105	70,415	67,70	0
50	0,180	900	0,85	Hélice		100,658	72,991	67,91	0

4.1.2 Método MERNA

Ingresamos los datos indicados y el **MERNA** selecciona la **Turbina Michell**, como se observan en las figuras **4.2** y **4.3**; el Manual de Mini y Micro centrales Hidráulicas **[32]**, en la página 146 selecciona la misma turbina.

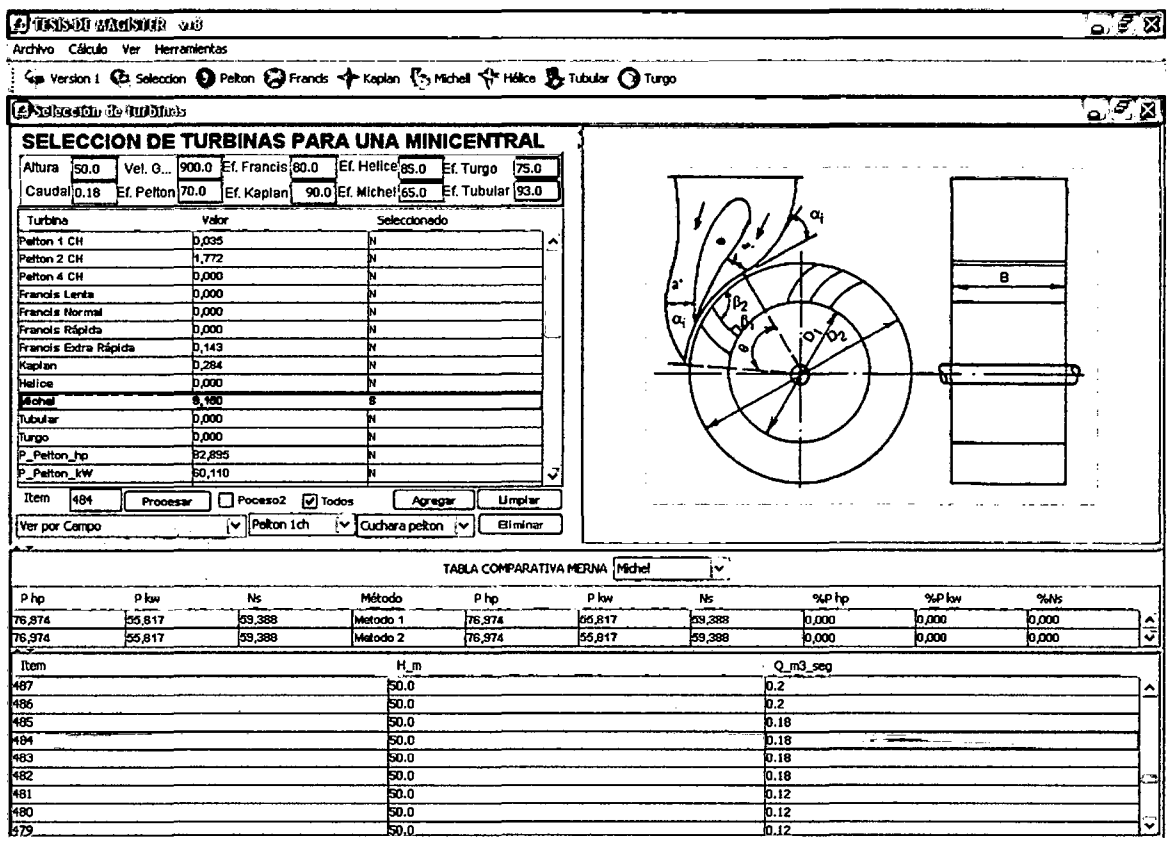


Fig. 4.1 Selección de una turbina Mitchell. Método 1.

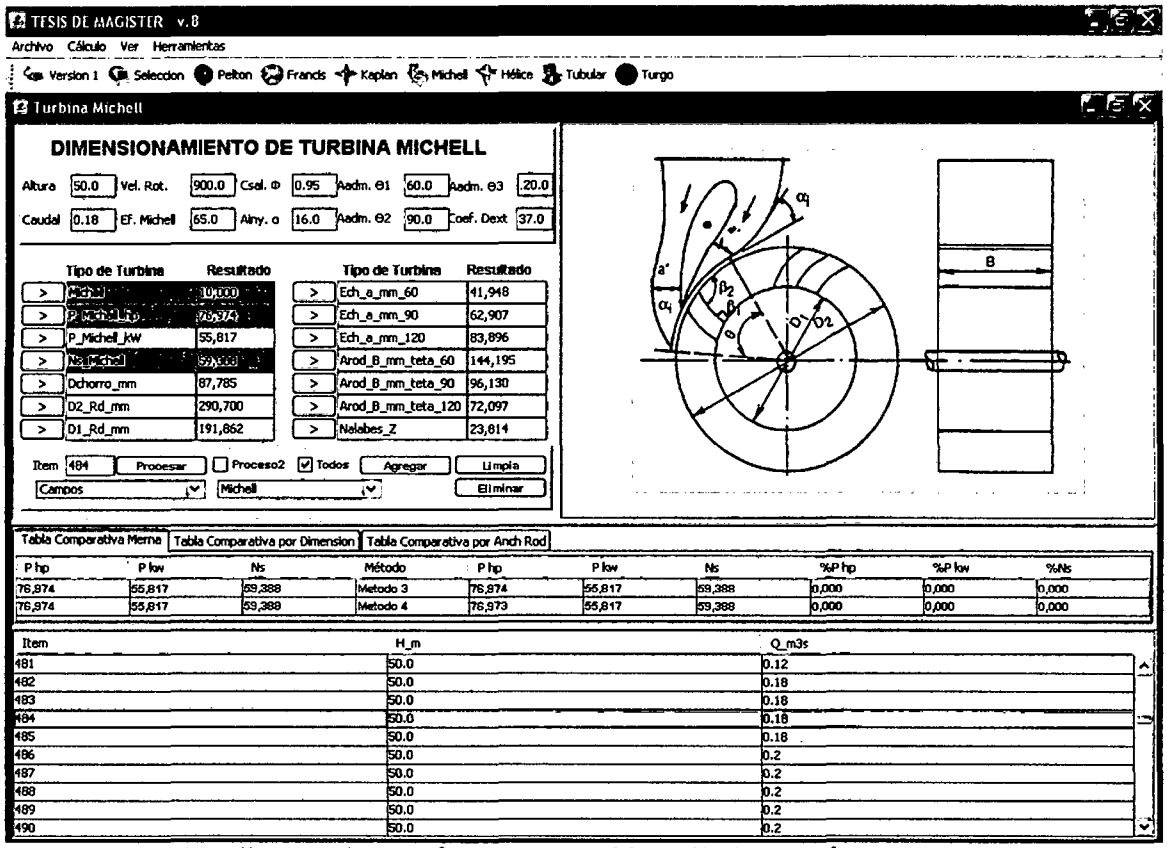


Fig. 4.2 Selección de una turbina Mitchell. Método 2.

En la tabla **4.2** y **4.3** mostramos una comparación en los resultados, en la selección y dimensiones principales de la turbina; por el método tradicional y por el método **MERNA** (métodos 1,2 y 3,4) y la desviación en los resultados que se presentan. Al Ingresar los datos indicados el MERNA calcula las dimensiones principales de la **Turbina Michell**, como se observan en las figuras **4.3** y **4.4.**

Tabla 4.2 Comparación de resultados: Selección de la Turbina Michell.

Método Tradicional			Método	Valores de Salida			[(Mt - SM)/Mt]*100 %		
P hp	P kW	Ns	MERNA	P hp	P kW	Ns	P hp	P kW	Ns
76.974	55.817	59.39	Método 1	76.974	55.817	59.338	0.000	0.000	0.000
76.974	55.817	59.39	Método 2	76.974	55.817	59.338	0.000	0.000	0.000
76.974	55.817	59.39	Método 3	76.974	55.817	59.338	0.000	0.000	0.000
76.974	55.817	59.39	Método 4	76.974	55.817	59.338	0.000	0.000	0.000

Tabla 4.3 Comparación de resultados: Dimensiones de la Turbina Michell.

Método Tradicional			Método	Valores de Salida			[(Mt - SM)/Mt]*100 %		
Dchorro	D2_Rd	D1_Rd	MERNA	Dchorro	D2_Rd	D1_Rd	Dchorro	D2_Rd	D1_Rd
87.786	290.699	191.861	Método 3	76.974	55.817	59.338	0.000	0.000	0.000
87.786	290.699	191.861	Método 4	76.974	55.817	59.338	0.000	0.000	0.000

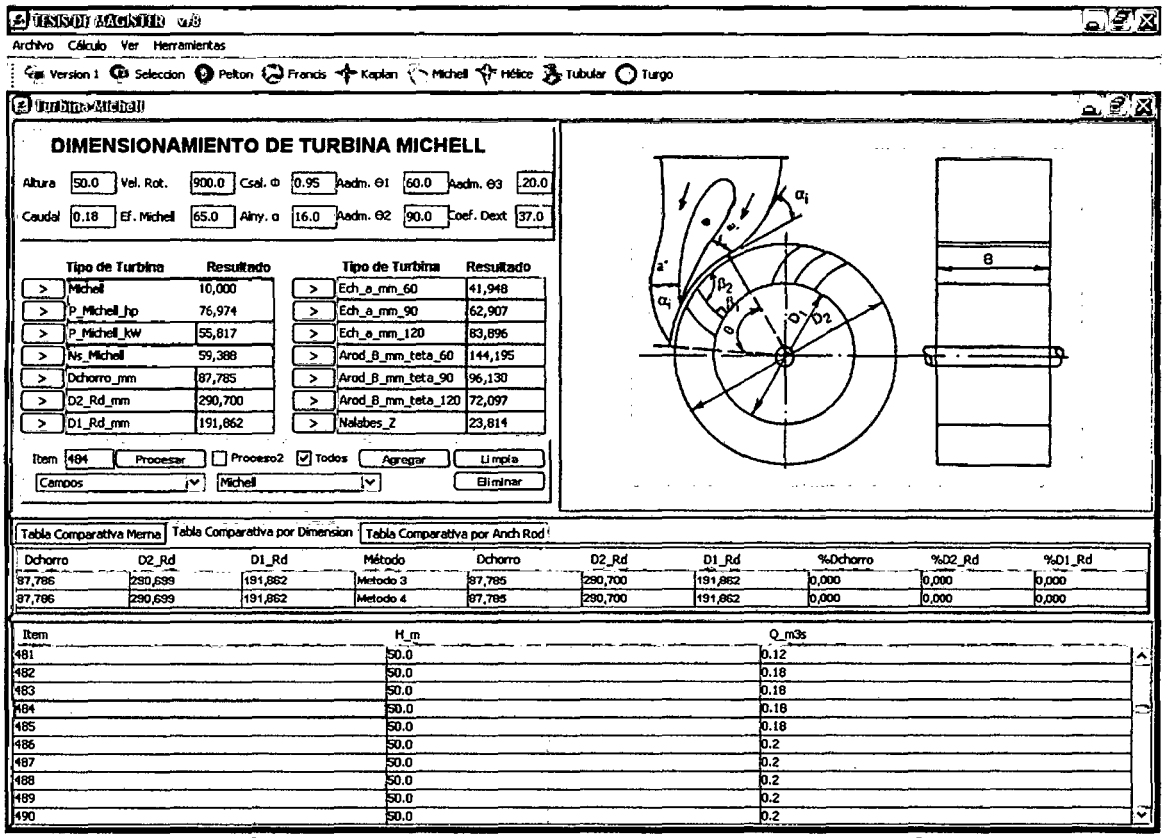


Fig. 4.3 Dimensiones principales de la turbina Mitchell. Método 3.

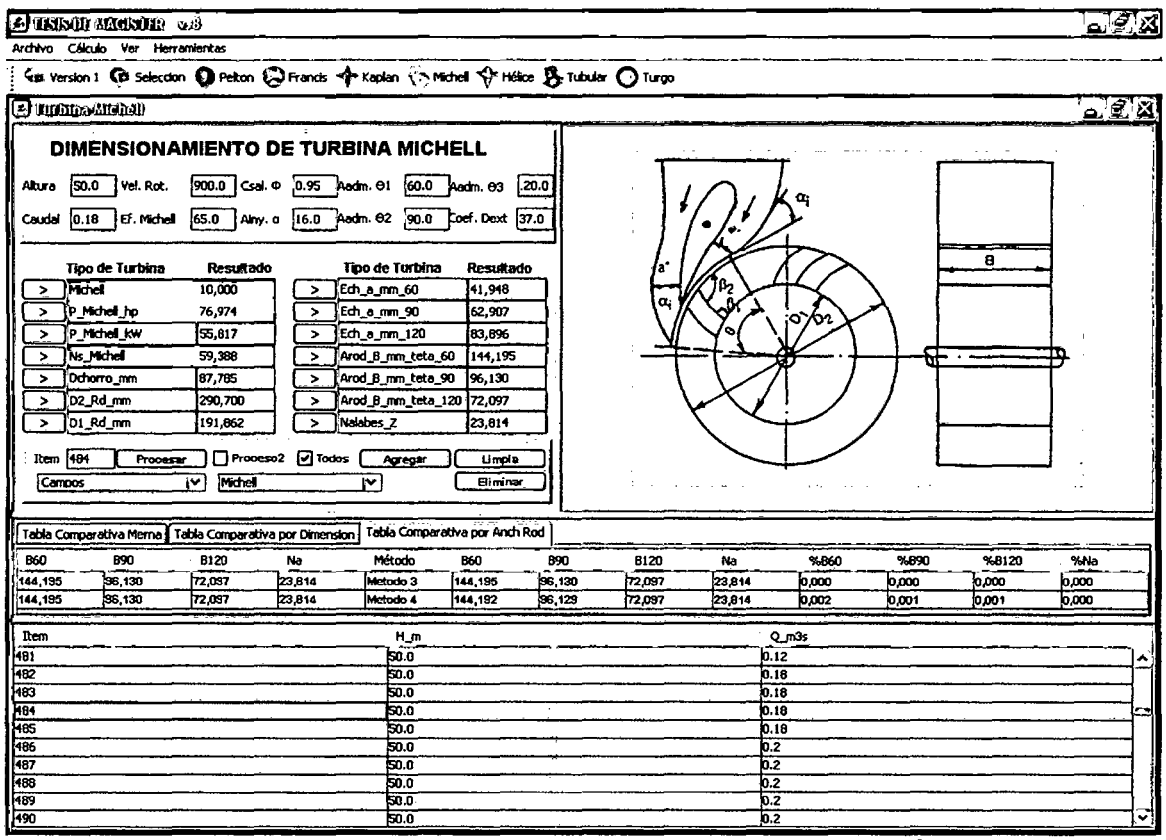


Fig. 4.4 Dimensiones principales de la turbina Mitchell. Método 4.

4.2 RESULTADOS EN SELECCIÓN Y DISEÑO DE UNA TURBINA HÉLICE.

Ingresar los datos de **H = 3.50 m, Q= 6 m^3/seg, Vr = 257 rpm** y eficiencia de la turbina **η = 85%;**

4.2.1 Método Tradicional

Utilizando las ecuaciones (2.1), (2.2) y (2.3) y comparando con los valores de la tabla 1.1 y en el gráfico de la figura 1.2, seleccionamos la ***turbina Hélice***.

Tabla 4.4 Selección de la Turbina Hélice.

Altura	Caudal	Vel. Rot.	Efic. Turb.	Tipo Turbina		Pot. Turb.	Pot. Turb.	Vel. Espcif.	Turb. Selec.
H	Q	n	η	Nombre	Clase	P	P	Ns	
m	m^3/seg	rpm				HP	kW	rpm	
3,50	6,0	257	0.70	Pelton	1 Ch	165,789	120,221	838,13	0
3,50	6,0	257	0,70	Pelton	2 Ch	165,789	120,221	592,65	0
3,50	6,0	257	0,70	Pelton	4 Ch	165,789	120,221	419,06	0
3,50	6,0	257	0,65	Michell		153,947	111,634	807,64	0
3,50	6,0	257	0,80	Francis	Lenta	189,474	137,395	896,00	0
3,50	6,0	257	0,80	Francis	Normal	189,474	137,395	896,00	0
3,50	6,0	257	0,80	Francis	Rápida	189,474	137,395	896,00	0
3,50	6,0	257	0,82	Kaplan		194,211	140,830	907,13	0
3,50	6,0	257	0,85	Hélice		234,868	170,313	822,74	10

4.2.2 Método MERNA

Ingresamos los datos indicados y el MERNA selecciona la **Turbina Hélice**, como se observan en las figuras **4.5** y **4.6**; el texto de Motores Hidráulicos de Quantz, en la página 180 selecciona la misma turbina.

En las tablas **4.5** y **4.6** mostramos una comparación en los resultados, en la selección y dimensiones principales de la turbina; por el método tradicional y por el método **MERNA** (métodos 1, 2 y 3,4) y la desviación en los resultados que se presentan.

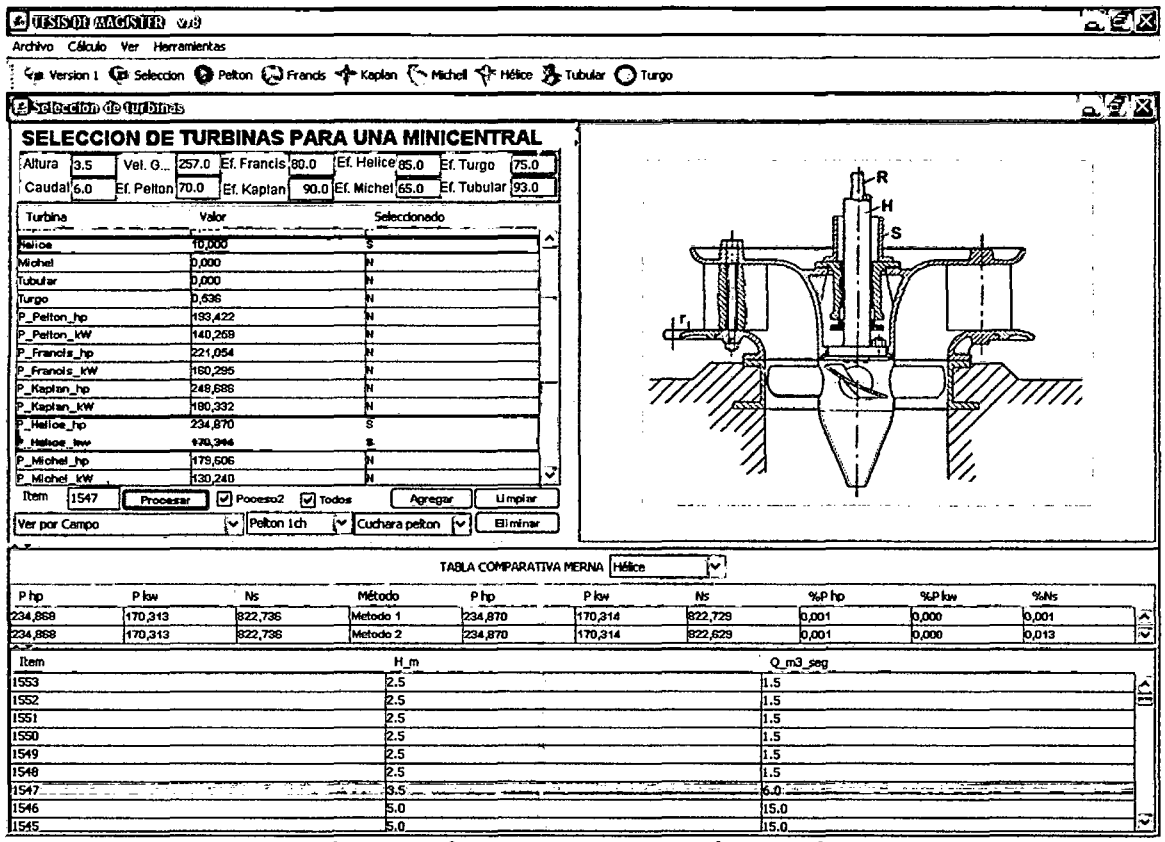


Fig. 4.5 Selección de una turbina Hélice. Método 1.

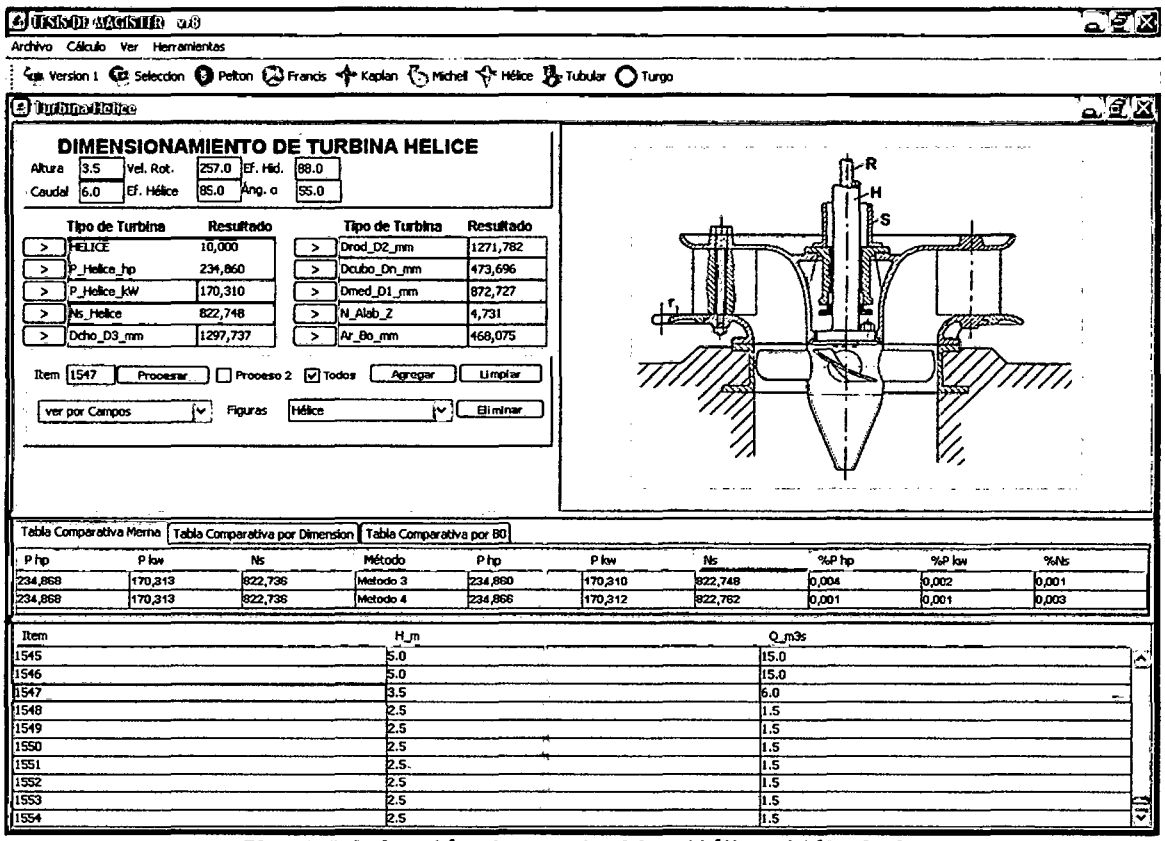


Fig. 4.6 Selección de una turbina Hélice. Método 3.

Tabla 4.5 Comparación de resultados: Selección de la Turbina Hélice.

Método Tradicional			Método	Variables de Salida			[(Mt - SM)/Mt]*100 %		
P hp	P kw	Ns	MERNA	P hp	P kw	Ns	P hp	P kw	Ns
234.868	170.313	822.74	Método 1	234.870	170.314	822.729	0.001	0.000	0.001
234.868	170.313	822.74	Método 2	234.870	170.314	822.629	0.001	0.000	0.013
234.868	170.313	822.74	Método 3	234.860	170.310	822.748	0.004	0.002	0.001
234.868	170.313	822.74	Método 4	234.866	170.312	822.762	0.001	0.001	0.003

Tabla 4.6 Comparación de resultados: Dimensiones de la Turbina Hélice.

Método Tradicional			Método	Variables de Salida			[(Mt - SM)/Mt]*100 %		
Dchorro	D2_Rd	D1_Rd	MERNA	Dchorro	D2_Rd	D1_Rd	Dchorro	D2_Rd	D1_Rd
1297.685	1271.731	872.713	Método 1	1297.757	1271.782	875.727	0.004	0.004	0.002
1297.685	1271.731	872.713	Método 2	1297.716	1271.759	872.726	0.002	0.002	0.001

Al Ingresar los datos indicados el MERNA calcula las dimensiones principales de la **Turbina Hélice**, como se observan en las figuras **4.7** y **4.8.**

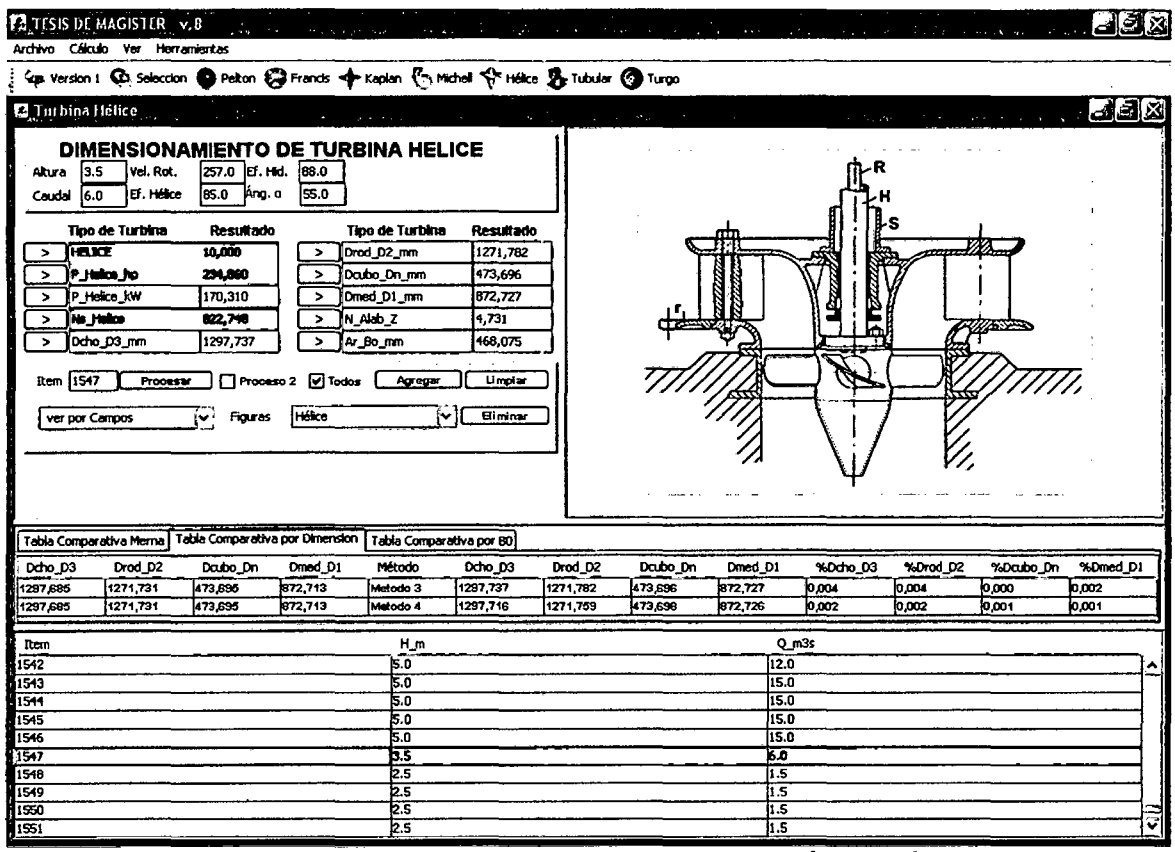


Fig. 4.7 Dimensiones principales de la turbina Hélice. Método 3.

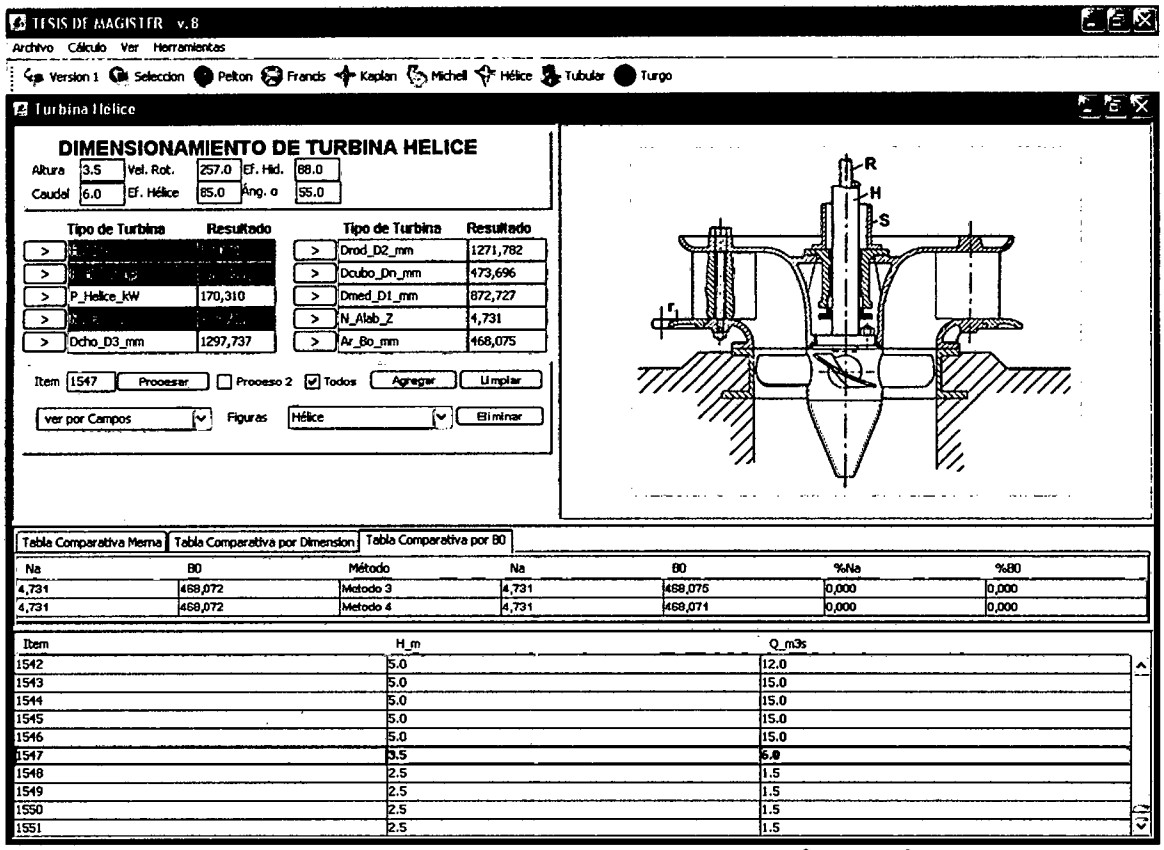


Fig. 4.8 Dimensiones principales de la turbina Hélice. Método 4.

4.3 RESULTADOS EN SELECCIÓN Y DISEÑO DE UNA TURBINA PELTON 2CH.

Ingresar los datos de **H = 120 m, Q = 0.150 m^3/seg, Vr = 1800 rpm** y eficiencia de la turbina **η = 85%.**

4.3.1 Método Tradicional

Utilizando las ecuaciones (2.1), (2.2) y (2.3) y comparando con los valores de la tabla 1.1 y en el gráfico de la figura 1.2, seleccionamos la ***turbina Pelton 2ch***.

Tabla 4.7 Selección de la Turbina Pelton 2ch.

Altura	Caudal	Vel. Rot.	Efic. Turb.	Tipo Turbina		Pot. Turb.	Pot. Turb.	Vel. Espcif.	Turb. Selec.
H	Q	n	η	Nombre	Clase	P	P	Ns	
m	m^3/seg	rpm				HP	KW	rpm	
120	0,150	1800	0.85	Pelton	1 Ch	201.316	145.983	64.30	0
120	0,150	1800	0,85	Pelton	2 Ch	201.316	145.983	45.47	10
120	0,150	1800	0,85	Pelton	4 Ch	201.316	145.983	32.15	0

Tabla 4.7 Selección de la Turbina Pelton 2ch (cont).

Altura	Caudal	Vel. Rot.	Efic. Turb.	Tipo Turbina	Pot. Turb.	Pot. Turb.	Vel. Espcif.	Turb. Selec.	Turb. Selec
H	Q	n	η	Nombre	Clase	P	P	Ns	
m	m³/seg	rpm				HP	KW	rpm	
120	0,150	1800	0,65	Michell		153.947	111.634	56.23	0
120	0,150	1800	0,80	Francis	Lenta	189.474	137.395	62.38	0
120	0,150	1800	0,80	Francis	Normal	189.474	137.395	62.38	0
120	0,150	1800	0,80	Francis	Rápida	189.474	137.395	62.38	0
120	0,150	1800	0,82	Kaplan		194.211	140.830	63.16	0
120	0,150	1800	0,85	Hélice		201.316	145.983	64.30	0

4.3.2 Método MERNA

Ingresamos los datos indicados y el MERNA selecciona la **Turbina Pelton 2CH**, como se observan en las figuras **4.9** y **4.10**; el texto de Motores Hidráulicos de Quantz **[31]**, en la página 203 selecciona la misma turbina.

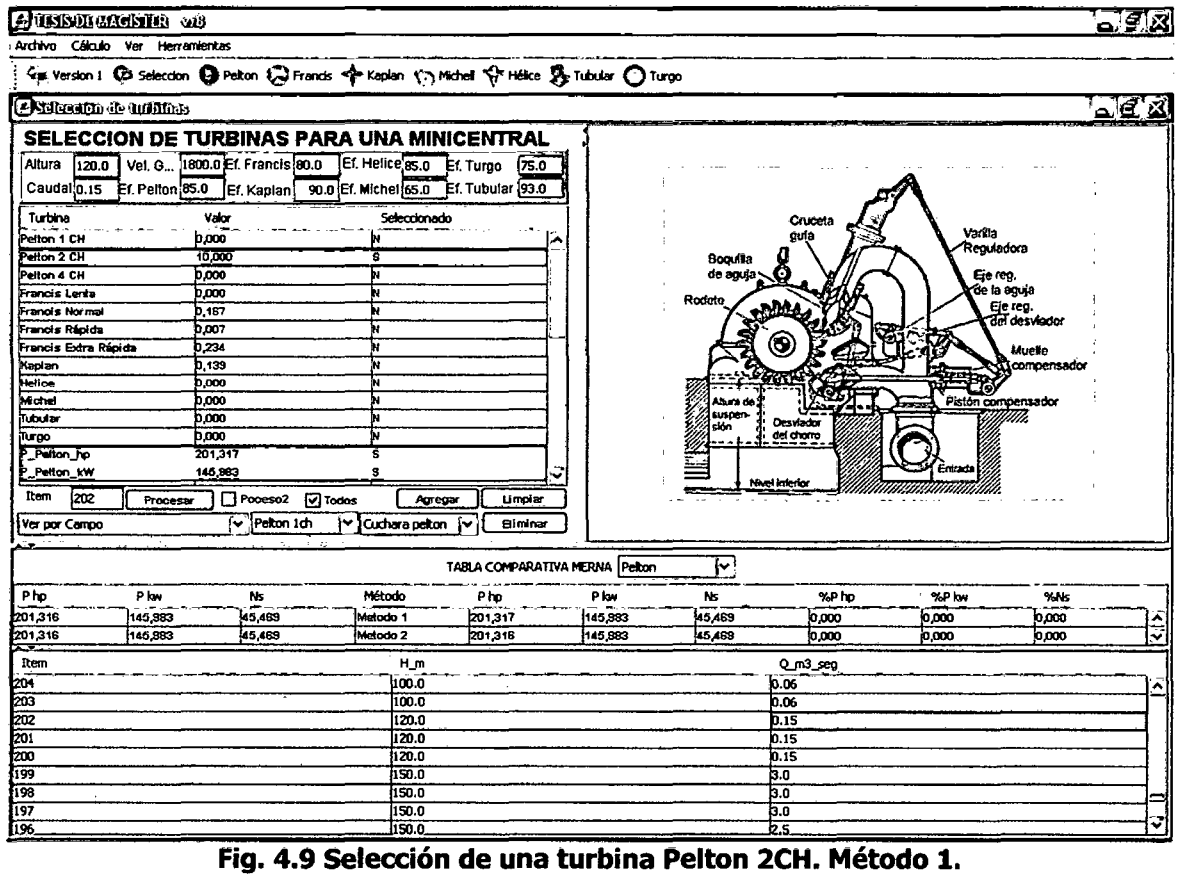


Fig. 4.9 Selección de una turbina Pelton 2CH. Método 1.

En la tabla **4.8** y **4.9** mostramos una comparación en los resultados, en la selección y dimensiones principales de la turbina; por el método tradicional y por

el método **MERNA** (métodos 1,2 y 3,4) y la desviación en los resultados que se presentan.

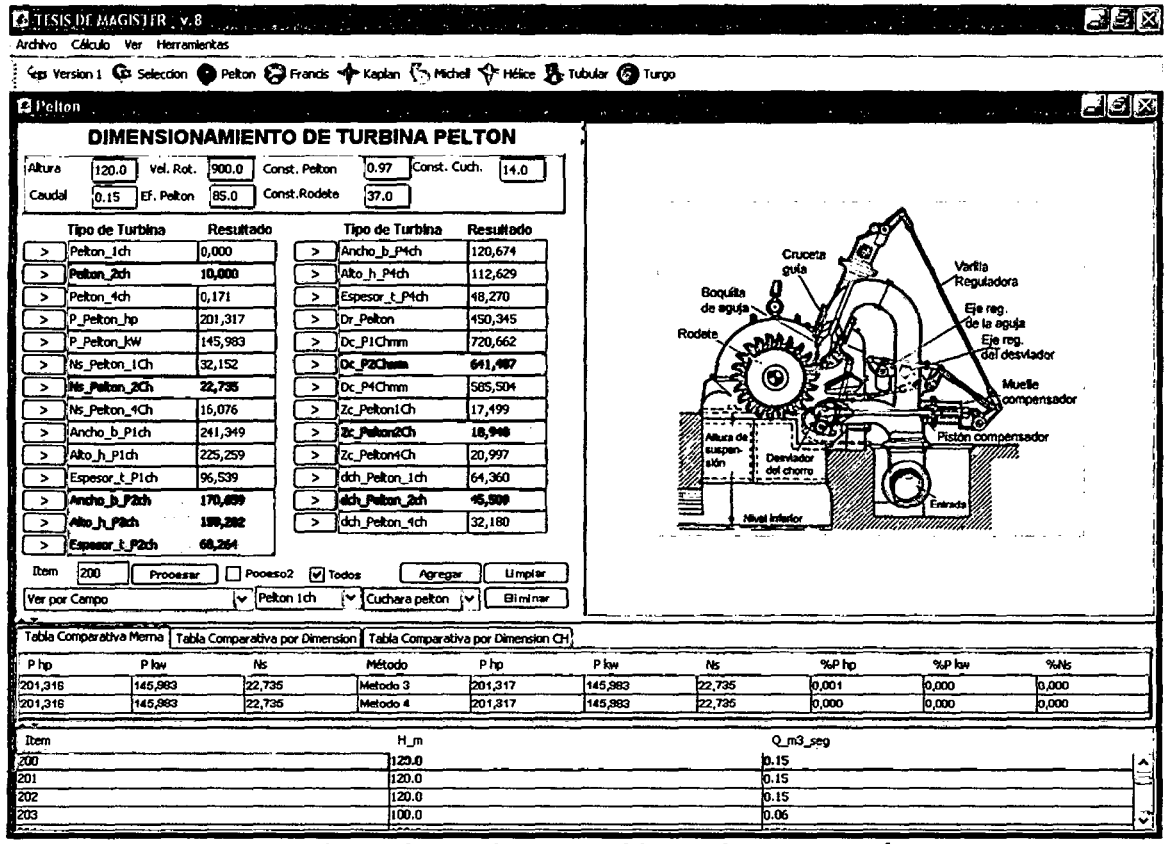


Fig. 4.10 Dimensiones de una turbina Pelton 2CH. Método 2.

Al Ingresar los datos indicados el MERNA calcula las dimensiones principales de la **Turbina Pelton 2ch**, como se observan en las figuras **4.11** y **4.12.**

Tabla R.8 Comparación de resultados: Selección de la Turbina Pelton 2ch.

Método Tradicional			Método	Variables de Salida			[(Mt - SM)/Mt]*100 %		
P hp	P kw	Ns	MERNA	P hp	P kw	Ns	P hp	P kw	Ns
201.316	145.983	45.469	Método 1	201.317	145.983	45.469	0.000	0.000	0.000
201.316	145.983	45.469	Método 2	201.317	145.983	45.469	0.000	0.000	0.000
201.316	145.983	22.735	Método 3	201.317	145.983	22.735	0.001	0.000	0.000
201.316	145.983	22.735	Método 4	201.317	145.983	22.735	0.000	0.000	0.000

Tabla R.9 Comparación de resultados: Dimensiones de la Turbina Pelton 2ch.

Método Tradicional			Método	Variables de Salida			[(Mt - SM)/Mt]*100 %		
Dchorro	D_Rd	Dc	MERNA	Dchorro	D_Rd	Dc	Dchorro	D2_Rd	D1_Rd
45.509	450.350	641.488	Método 3	45.509	450.345	641.487	0.000	0.001	0.000
45.509	450.350	641.488	Método 4	45.509	450.342	641.487	0.000	0.002	0.000

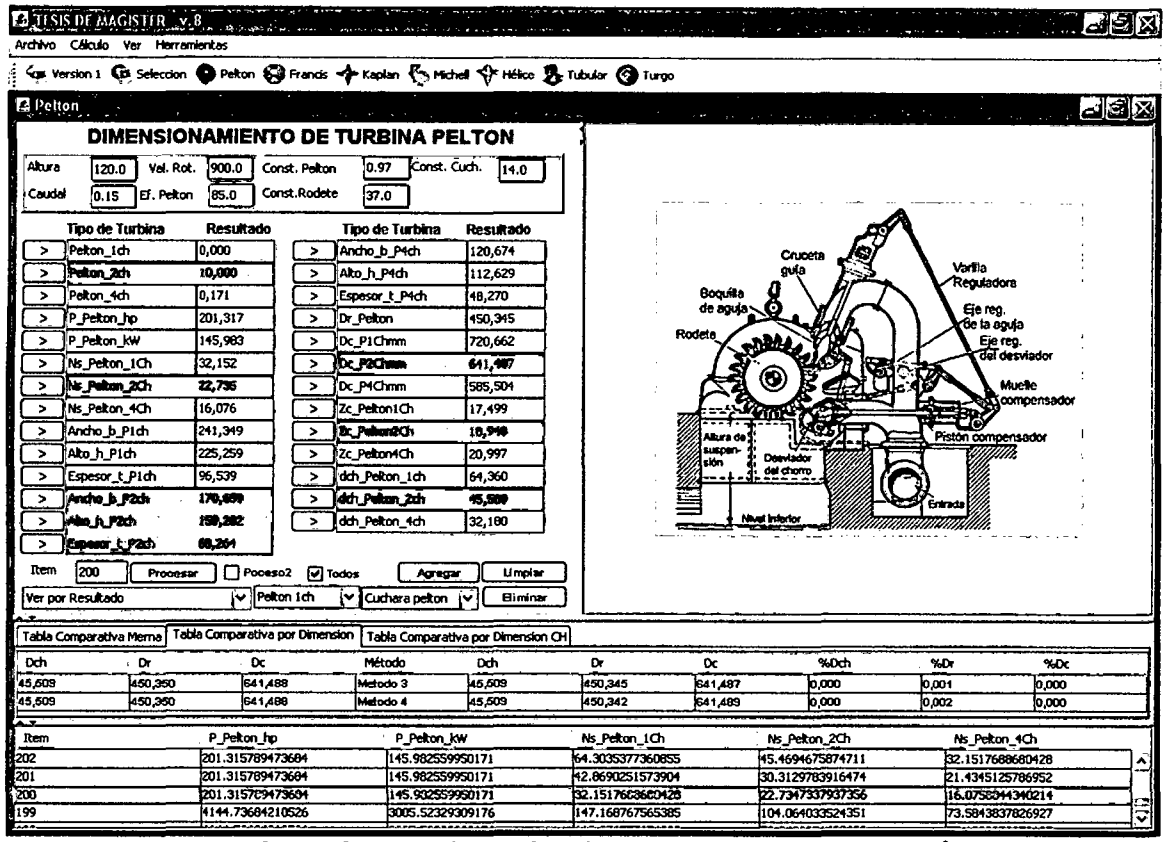


Fig. 4.11 Dimensiones principales de la turbina Pelton 2ch. Método 3.

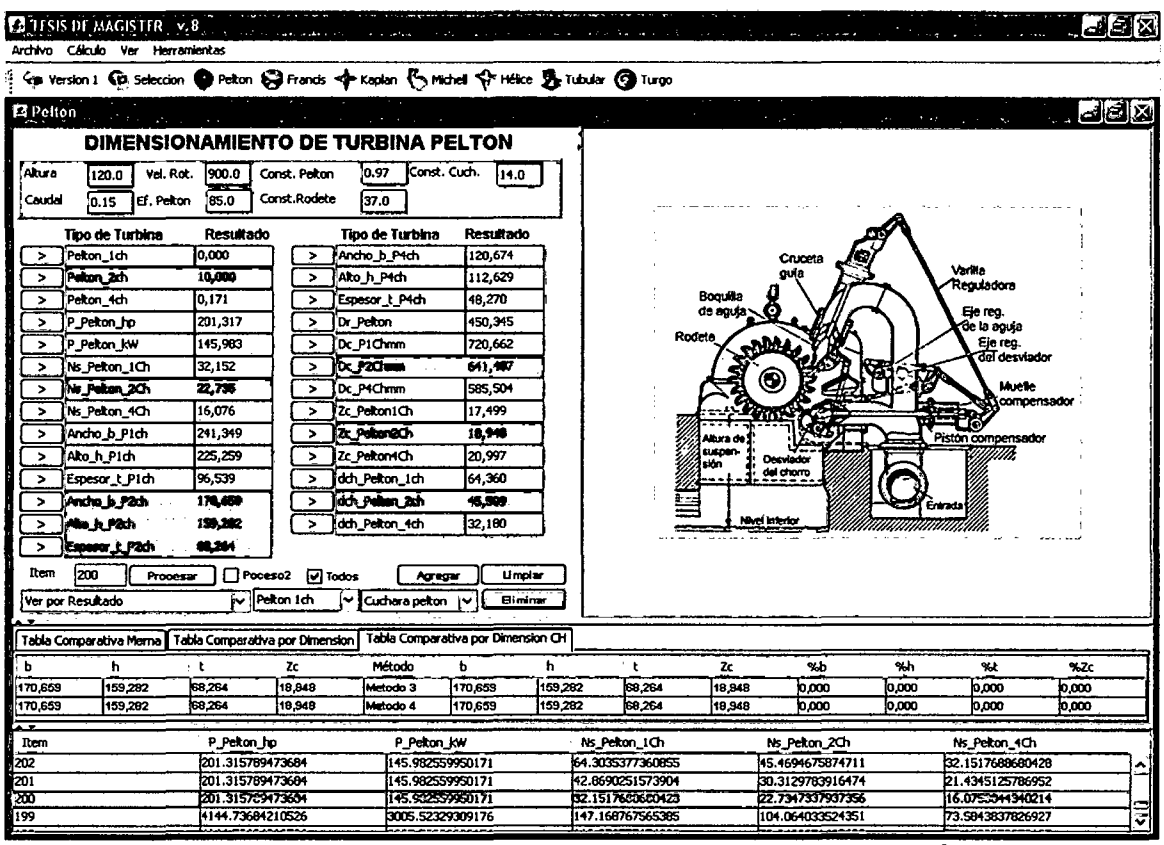


Fig. 4.12 Dimensiones principales de la turbina Pelton 2ch. Método 4.

4.4 RESULTADOS EN SELECCIÓN Y DISEÑO DE UNA TURBINA KAPLAN.

Ingresar los datos de **H = 9 m, Q = 10 m^3/seg, Vr = 277 rpm** y eficiencia de la turbina **η = 90%;**

4.4.1 Método Tradicional

Utilizando las ecuaciones (2.1), (2.2) y (2.3) y comparando con los valores de la tabla 1.1 y en el gráfico de la figura 1.2, seleccionamos la ***turbina Kaplan***.

Tabla 4.10 Selección de la Turbina Kaplan.

Altura	Caudal	Vel. Rot.	Efic. Turb.	Tipo Turbina		Pot. Turb.	Pot. Turb.	Vel. Espcif.	Turb. Selec.
H	Q	n	η	Nombre	Clase	P	P	Ns	
m	m^3/seg	rpm				HP	KW	rpm	
9	9	277	0.70	Pelton	1 Ch	828.947	601.105	474.672	0
9	9	277	0,70	Pelton	2 Ch	828.947	601.105	335.644	0
9	9	277	0,70	Pelton	4 Ch	828.947	601.105	237.336	0
9	9	277	0,65	Michell Banki		769.737	502.352	457.405	0
9	9	277	0,80	Francis	Lenta	947.368	686.917	507.446	0
9	9	277	0,80	Francis	Normal	947.368	686.917	507.446	0
9	9	277	0,80	Francis	Rápida	947.368	686.917	507.446	0
9	9	277	0,90	Kaplan		1065.789	772.849	538.527	10
9	9	277	0,85	Hélice		1006.579	729.913	523.063	0

4.4.2 Método MERNA

Ingresamos los datos indicados y el MERNA selecciona la **Turbina Kaplan**, como se observan en las figuras **4.13** y **4.14**.

En las tablas **4.11** y **4.12** mostramos una comparación en los resultados, en la selección y dimensiones principales de la turbina; por el método tradicional y por el método **MERNA** (métodos 1,2 y 3,4) y la desviación en los resultados que se presentan.

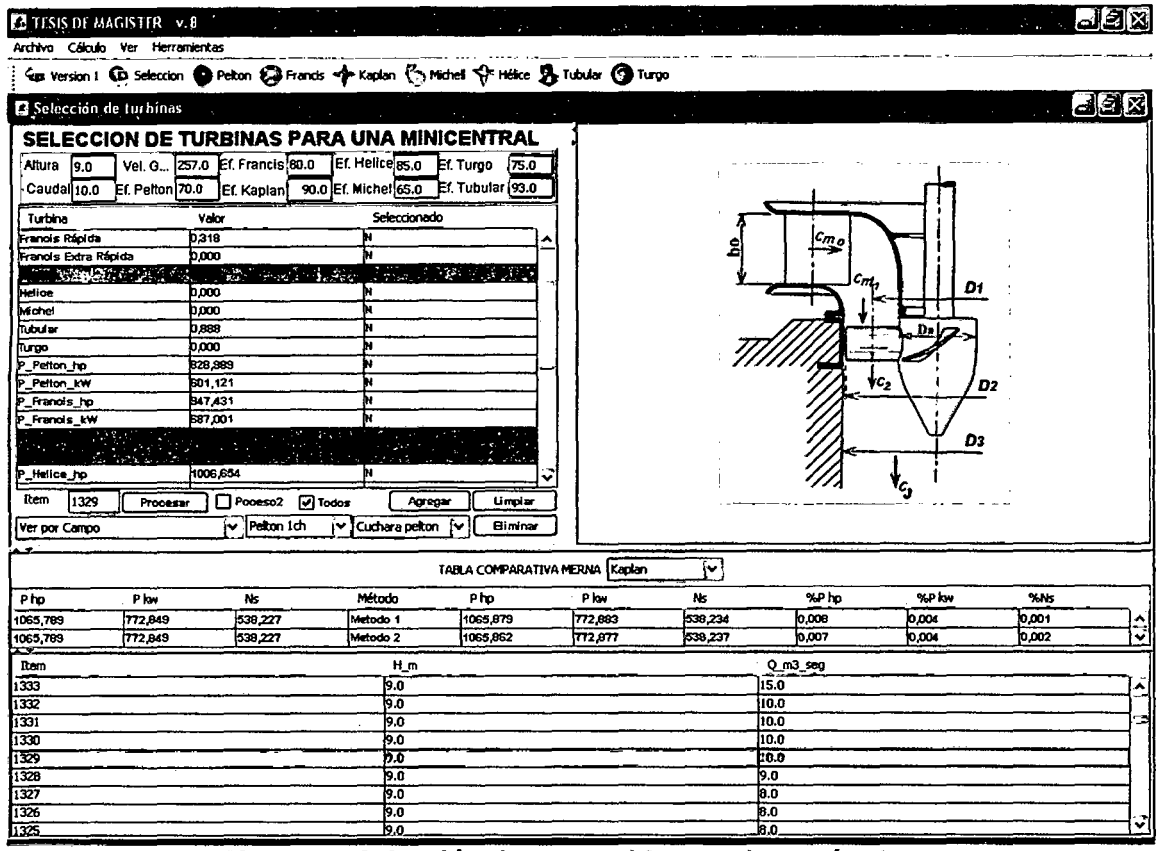


Fig. 4.1 3 Selección de una turbina Kaplan. Método 1.

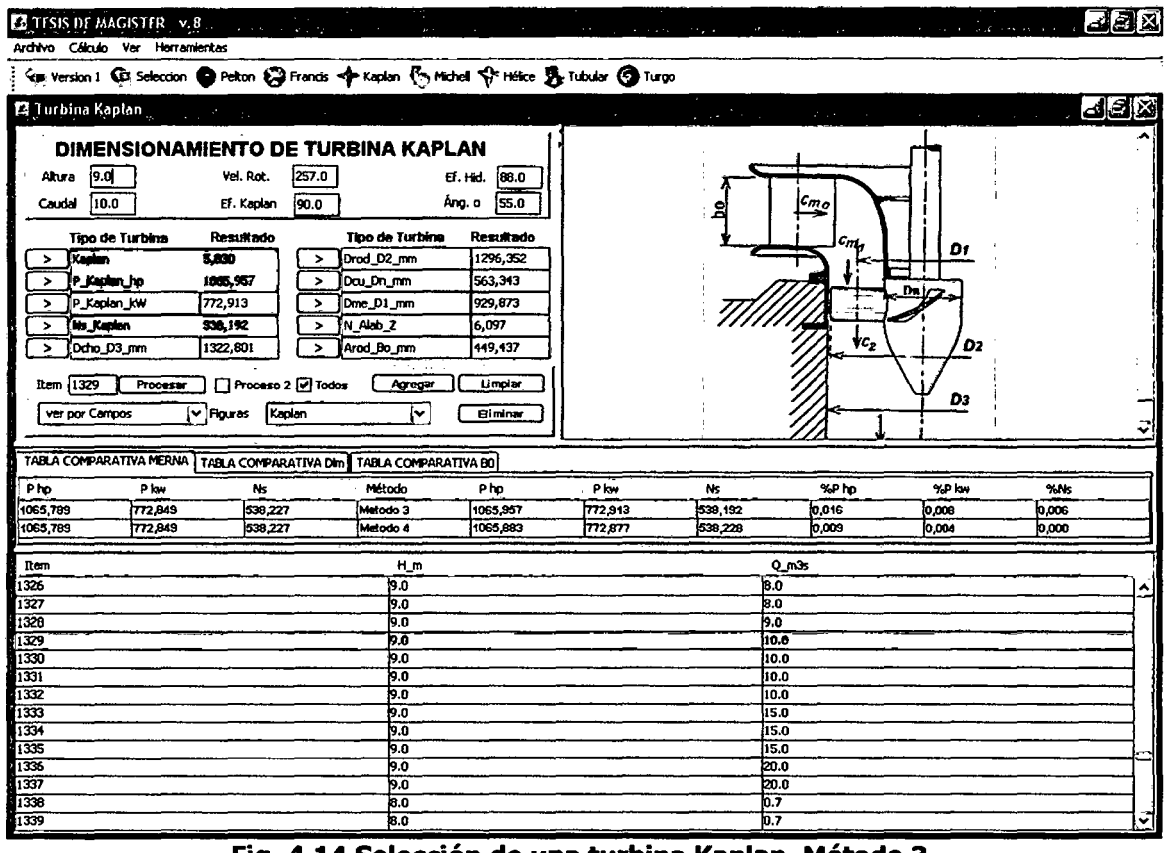


Fig. 4.14 Selección de una turbina Kaplan. Método 3.

Tabla 4.11 Comparación de resultados: Selección de la Turbina Kaplan.

Método Tradicional			Método	Variables de Salida			[(Mt - SM)/Mt]*100 %		
P hp	P kw	Ns	MERNA	P hp	P kw	Ns	P hp	P kw	Ns
1065.789	772.849	538.227	Método 1	1065.879	772.883	538.234	0.008	0.004	0.001
1065.789	772.849	538.227	Método 2	1065.862	772.877	538.237	0.007	0.004	0.002
1065.789	772.849	538.227	Método 3	1065.957	772.913	638.192	0.016	0.008	0.006
1065.789	772.849	538.227	Método 4	1065.883	772.877	538.228	0.009	0.004	0.0000

Tabla 4.12 Comparación de resultados: Dimensiones de la Turbina Kaplan.

Método Tradicional			Método	Variables de Salida			[(Mt - SM)/Mt]*100 %		
Dchorro	D_Rd	Dc	MERNA	Dchorro	D_Rd	Dc	Dchorro	D2_Rd	D1_Rd
1322.970	1296.511	563.369	Método 3	1322.801	1296.352	563.343	0.013	0.012	0.005
1322.970	1296.511	563.369	Método 4	1322.922	1296.466	563.365	0.004	0.004	0.001

Al Ingresar los datos indicados el MERNA calcula las dimensiones principales de la **Turbina Kaplan**, como se observa en las figuras **4.15** y **4.16.**

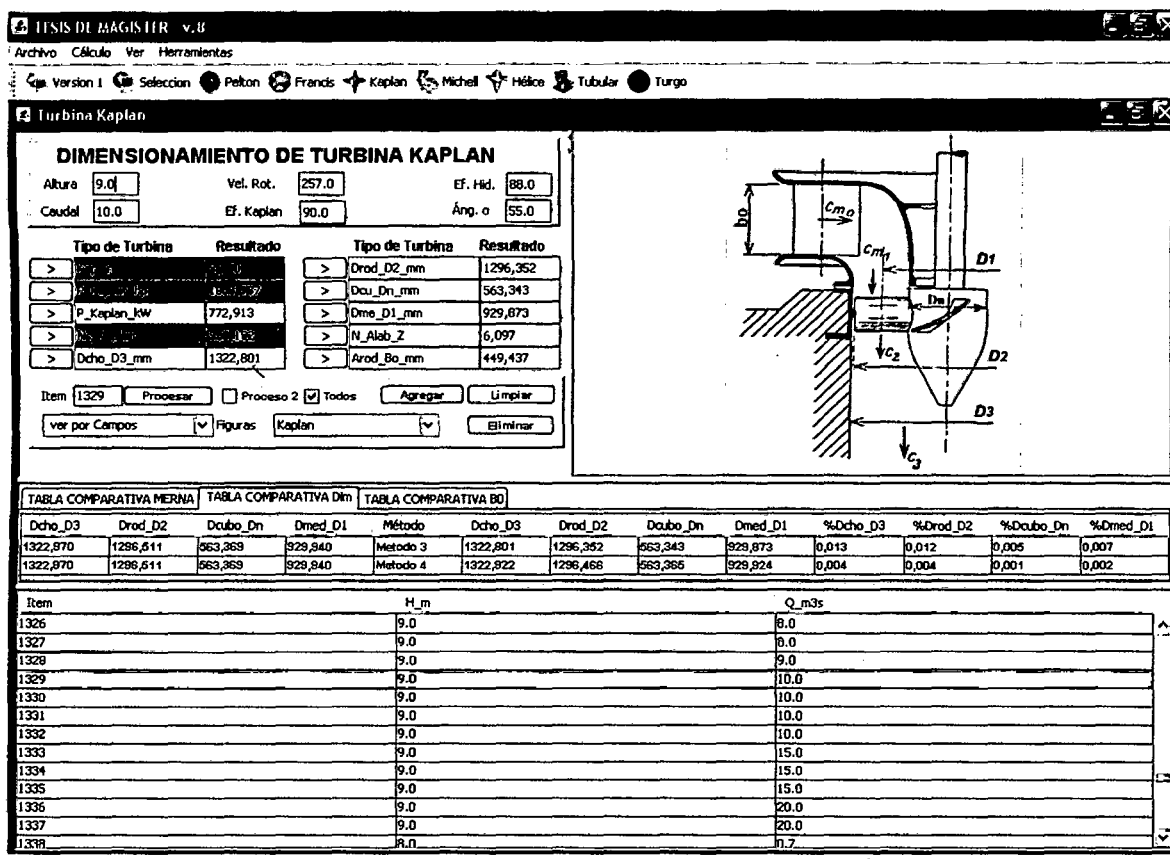


Fig. 4.15 Dimensiones de una turbina Kaplan. Método 3.

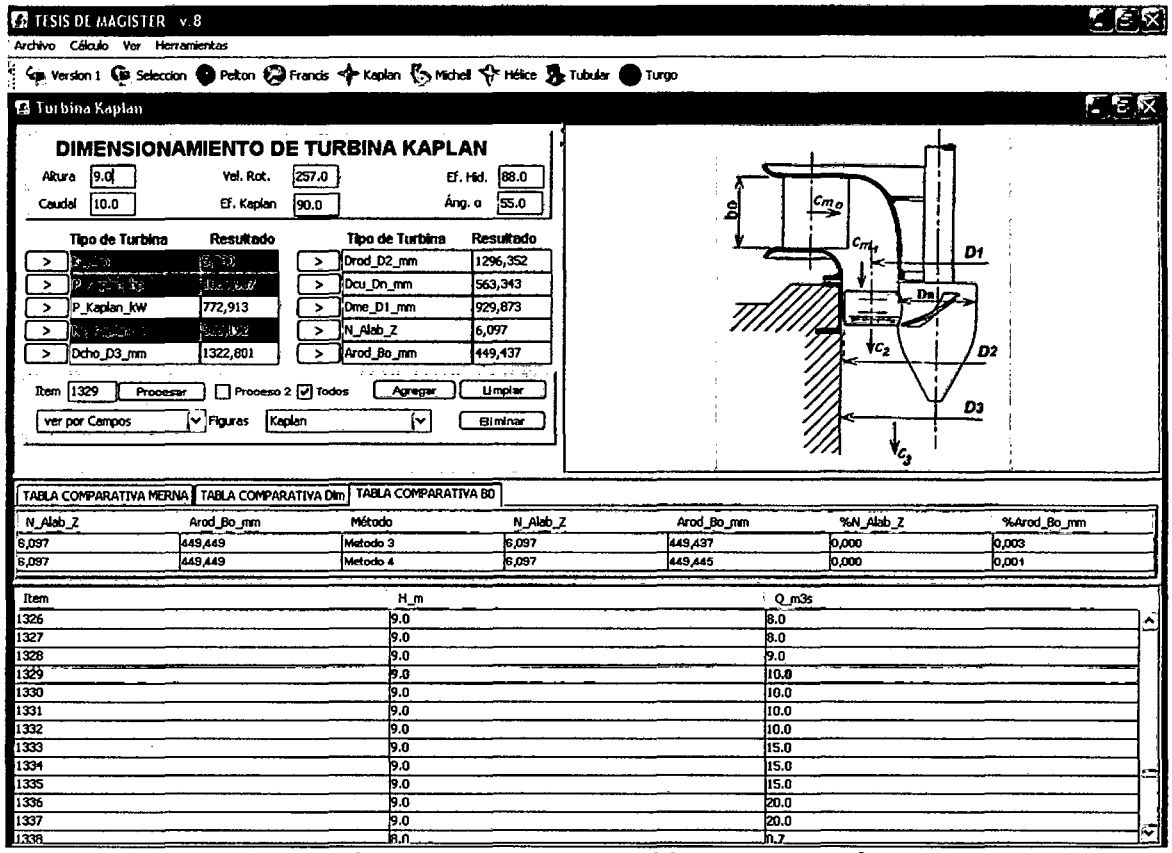


Fig. 4.16 Dimensiones de una turbina Kaplan. Método 4.

4.5 RESULTADOS EN SELECCIÓN Y DISEÑO DE UNA TURBINA FRANCIS LENTA.

Ingresar los datos de **H = 100 m, Q = 2.50 m^3/seg, Vr = 720 rpm** y eficiencia de la turbina **η = 80%.**

4.5.1 Método Tradicional

Utilizando las ecuaciones (2.1), (2.2) y (2.3) y comparando con los valores de la tabla 1.1 y en el gráfico de la figura 1.2, seleccionamos la ***turbina Francis Lenta***, como se observan en las figuras **4.17** y **4.18**.

En las tablas **4.14** y **4.15** mostramos una comparación en los resultados, en la selección y dimensiones principales de la turbina; por el método tradicional y por el método **MERNA** (métodos 1,2 y 3,4) y la desviación en los resultados que se presentan. Al Ingresar los datos indicados el MERNA calcula las dimensiones principales de la **Turbina Francis Lenta**, como se observan en las figuras **4.19** y **4.20.**

Tabla R.13 Selección de la Turbina Francis Lenta.

Altura	Caudal	Vel. Rot.	Efic. Turb.	Tipo Turbina		Pot. Turb.	Pot. Turb.	Vel. Espcif.	Turb. Selec.
H	Q	n	η	Nombre	Clase	P	P	Ns	
m	m³/seg	rpm				HP	KW	rpm	
100	2,50	720	0.70	Pelton	1 Ch	2302.63	1717.76	109.26	0
100	2,50	720	0,70	Pelton	2 Ch	2302.63	1717.76	77.26	0
100	2,50	720	0,70	Pelton	4 Ch	2302.63	1717.76	54.63	0
100	2,50	720	0,65	Michell		2136.98	1594.19	105.25	0
100	2,50	720	0,80	Francis	Lenta	2631.58	1908.27	116.80	10
100	2,50	720	0,80	Francis	Normal	2631.58	1963.16	116.80	0
100	2,50	720	0,80	Francis	Rápida	2631.58	1963.16	116.80	0
100	2,50	720	0,90	Kaplan		2960.53	2208.55	123.88	0
100	2,50	720	0,85	Hélice		2796.05	2085.86	120.39	0

4.5.2 Método MERNA

Ingresamos los datos indicados y el MERNA selecciona la **Turbina Francis Lenta**, como se observan en las figuras **4.17** y **4.18**.

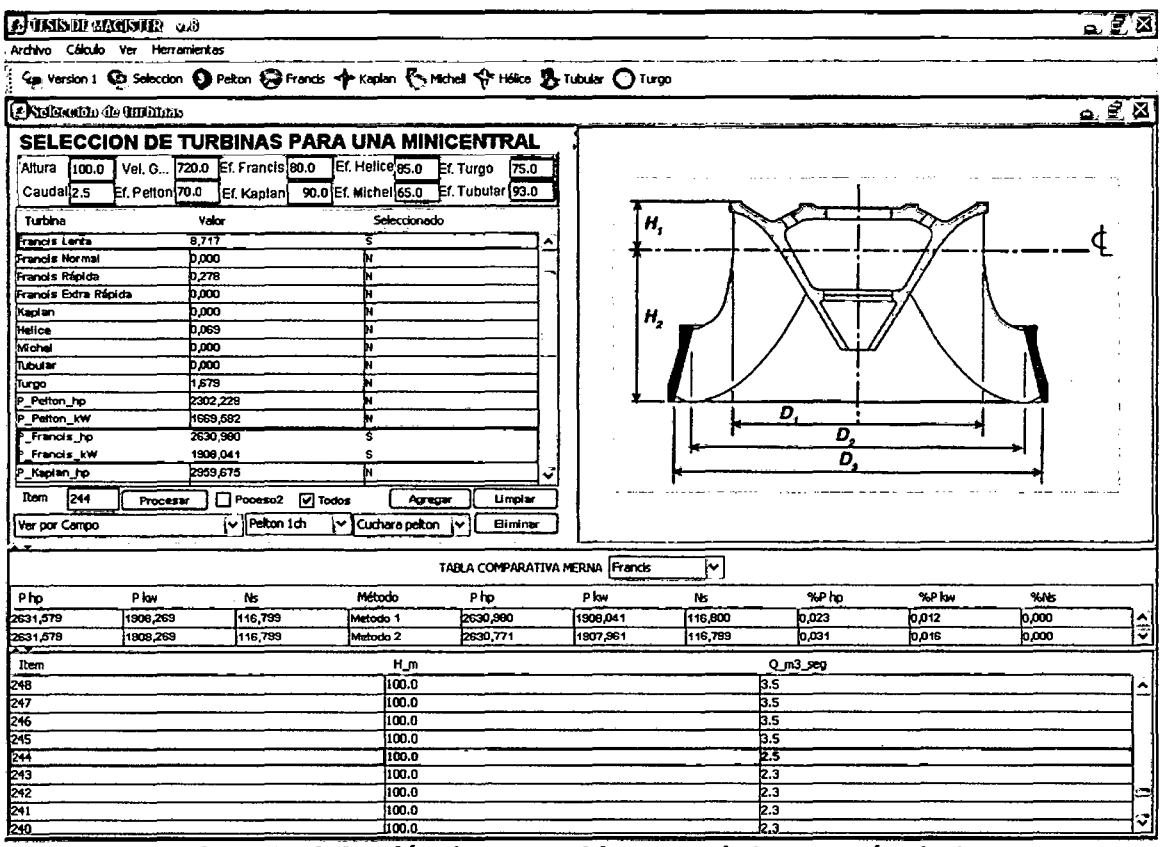


Fig. 4.17 Selección de una turbina Francis Lenta. Método 1.

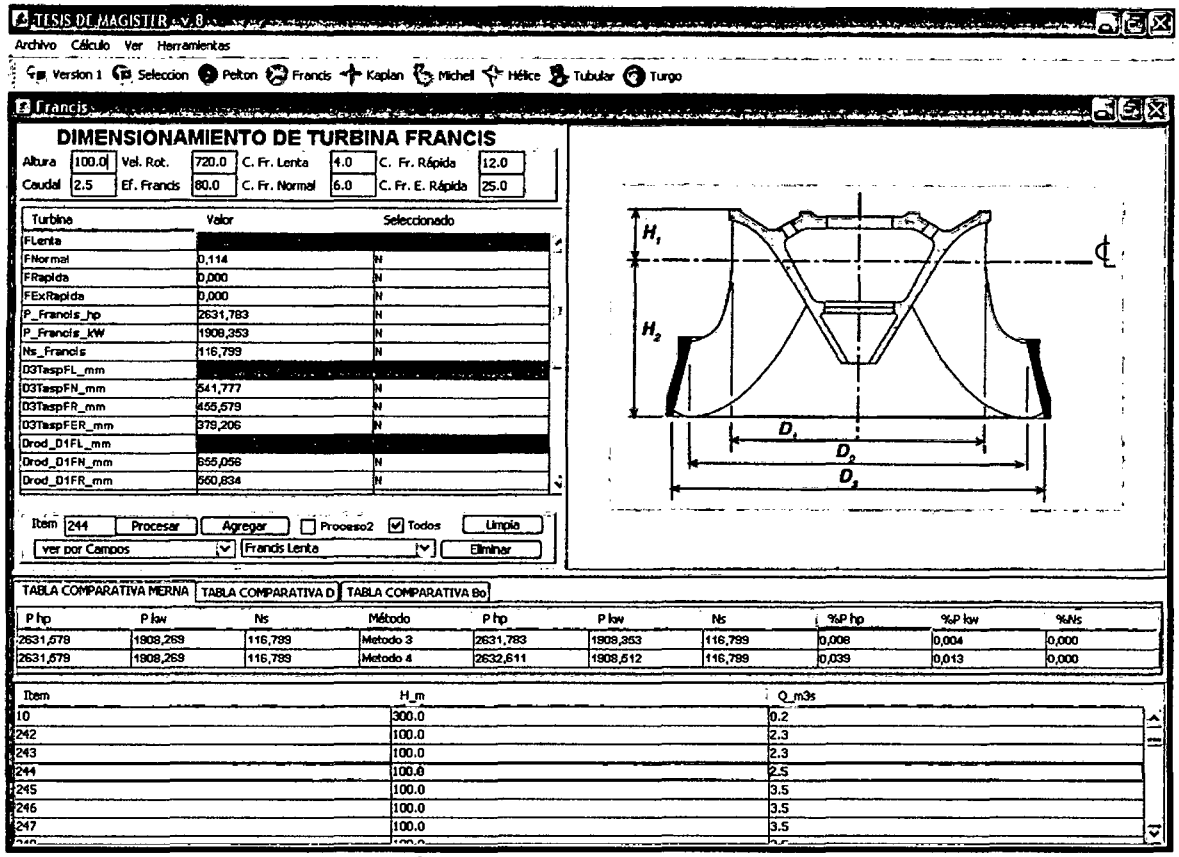


Fig. 4.18 Selección de una turbina Francis Lenta. Método 2.

Tabla R.14 Comparación de resultados: Selección de la Turbina Francis Lenta.

Método Tradicional			Método	Variables de Salida			[(Mt - SM)/Mt]*100 %		
P hp	P kw	Ns	MERNA	P hp	P kw	Ns	P hp	P kw	Ns
2631.579	1908.158	116.799	Método 1	2631.783	1908.353	116.800	0.023	0.012	0.000
2631.579	1908.158	116.799	Método 2	2630.771	1907.961	116.779	0.031	0.016	0.000
2631.579	1908.158	116.799	Método 3	2631.783	1908.353	116.779	0.008	0.004	0.000
2631.579	1908.158	116.799	Método 4	2432.611	1908.512	116.799	0.039	0.013	0.000

Tabla R.15 Comparación de resultados: Dimensiones de la Turbina Francis Lenta.

Método Tradicional			Método	Variables de Salida			[(Mt - SM)/Mt]*100 %		
D3Tasp	D_Rd	D2cor	MERNA	D3Tasp	D_Rd	D2cor	D3Tasp	D_Rd	D2cor
599.578	724.938	602.207	Método 3	599.574	724.939	602.204	0.001	0.000	0.000
599.578	724.938	602.207	Método 4	599.579	724.939	602.210	0.000	0.000	0.000

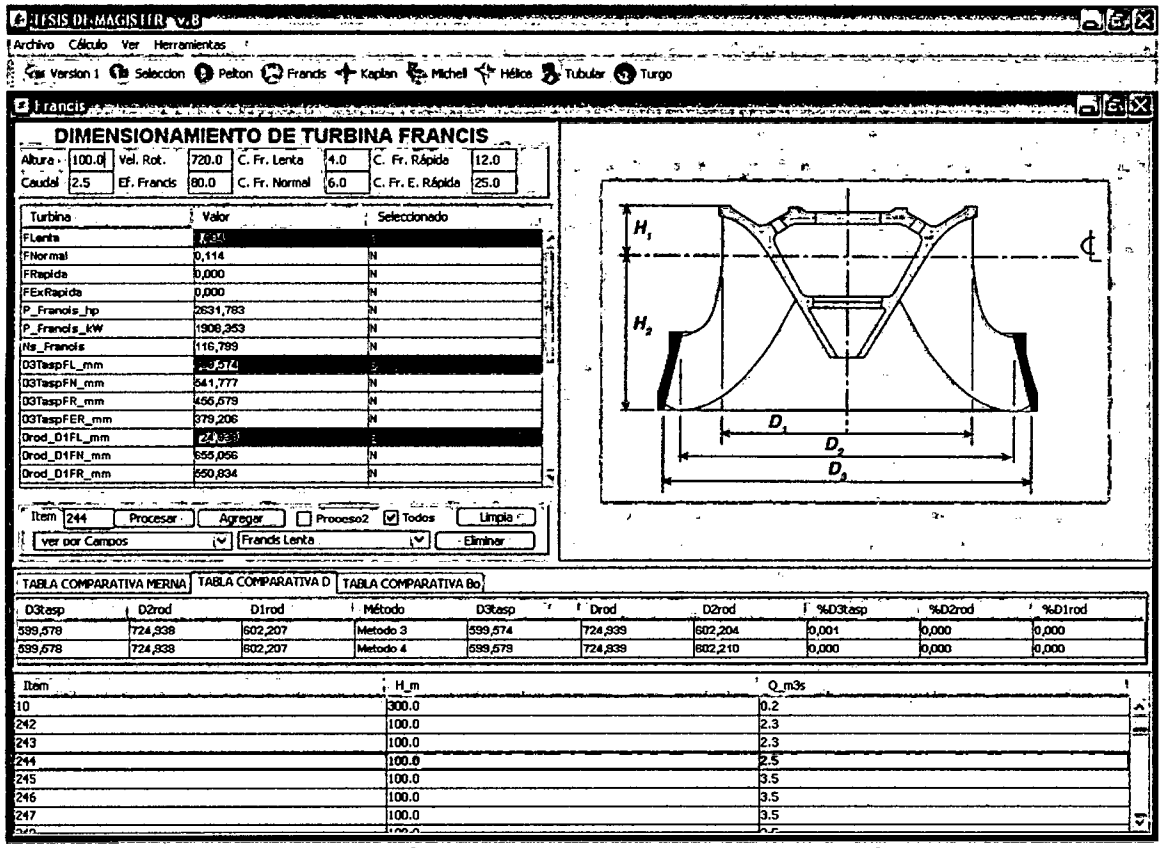


Fig. 4.19 Selección de una turbina Francis Lenta. Método 3.

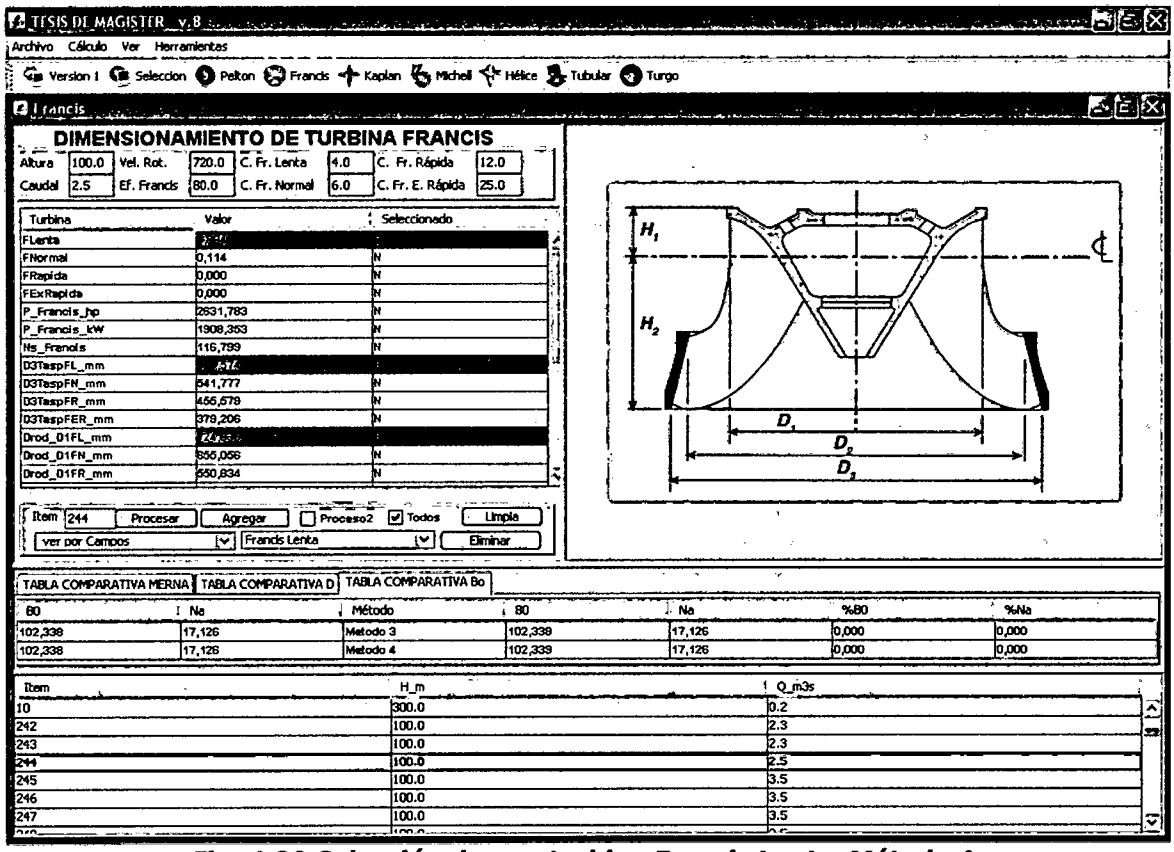


Fig. 4.20 Selección de una turbina Francis Lenta. Método 4.

CONCLUSIONES

En la evaluación de la selección de las turbinas y cálculo de las dimensiones principales, se han utilizado dos métodos: el tradicional y la utilización del MERNA que utiliza cuatro métodos para la selección y dos métodos para el cálculo de sus dimensiones principales.

1. El sistema desarrollado está especialmente dirigido a resolver el problema de la selección de turbinas hidráulicas, en aquellos casos en que no se encuentre el experto o de simplificar los pasos previos de cálculos para la selección de estas turbinas hidráulicas. La experiencia demuestra que ésta suele ser la situación más recomendable en los casos reales de selección de turbinas hidráulicas para diferentes casos complejos.

2. Se presenta una alternativa viable para la generación de energía eléctrica en base a fuentes renovables de energía en nuestro país, en especifico se analizó las mini centrales.

3. Se presenta una metodología general para la selección y diseño de las dimensiones principales de una turbina para una mini-hidráulica, tomando como base estudios previos realizados, teniendo como variables de entrada el caudal y la altura.

4. La investigación y la dificultad en la recopilación en archivos del número suficiente de proyectos reales, para abastecer al MERNA una base de datos fiable (almacenada en formato Excel), amplia y homogénea; esta base de datos permitirá al MERNA realizar estimaciones ajustadas y disponer de información real acerca de la selección de turbinas y diseño de estas.

5. El entrenamiento del MERNA se realizó en el sistema de principiante y el sistema avanzado. En el sistema avanzado, la red backpropagation genera un archivo en C que ocupa 180kb, la red GRNN genera un archivo en C que ocupa 1.9 MB, este archivo no es posible compilarlos directamente sin sucesivas divisiones del código fuente.

6. Al ingresar los datos de **H = 20 m, Q = 2 m^3/seg, Vr = 720 rpm** y eficiencia de la turbina **η = 80%;** el programa **MERNA** selecciona **Turbina Francis Rápida**, como se observan en las figuras 5.1, 5.2, 5.3 y 5.4.

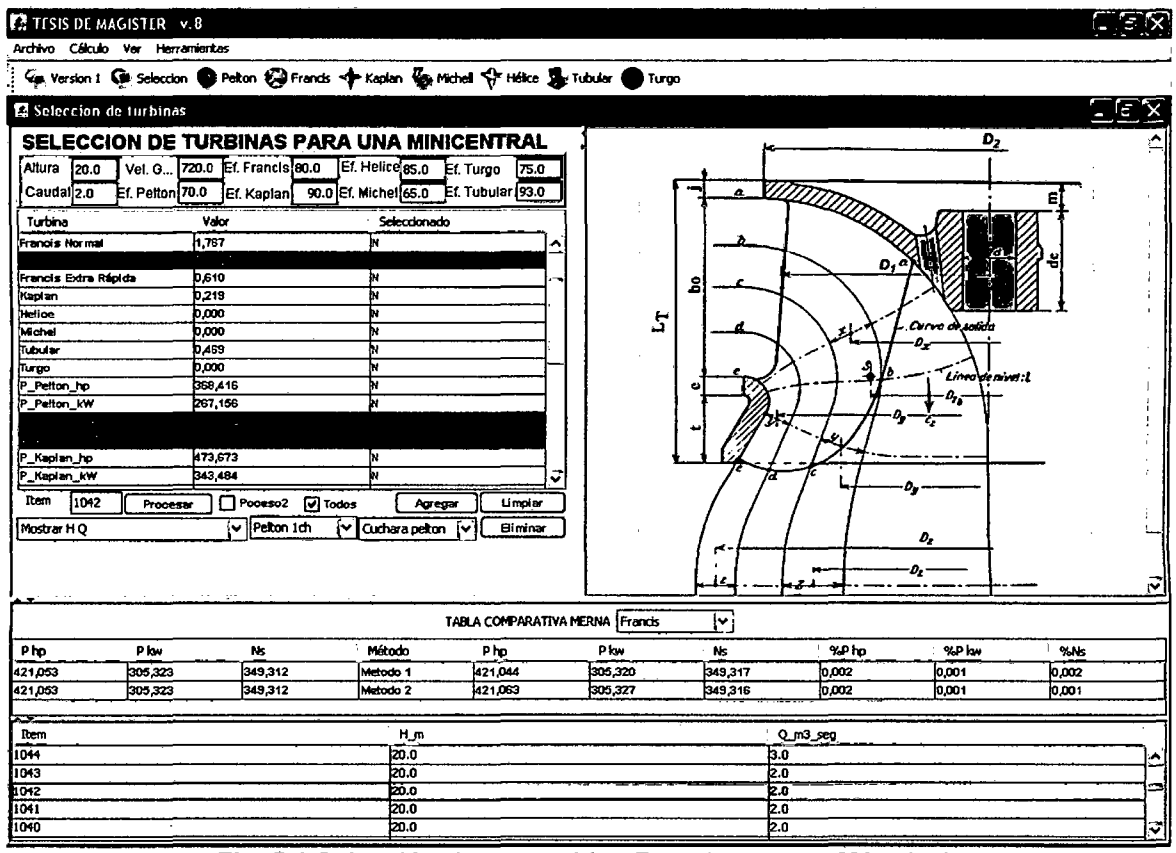


Fig. 5.1 Selección de una turbina Francis Rápida. Método 1.

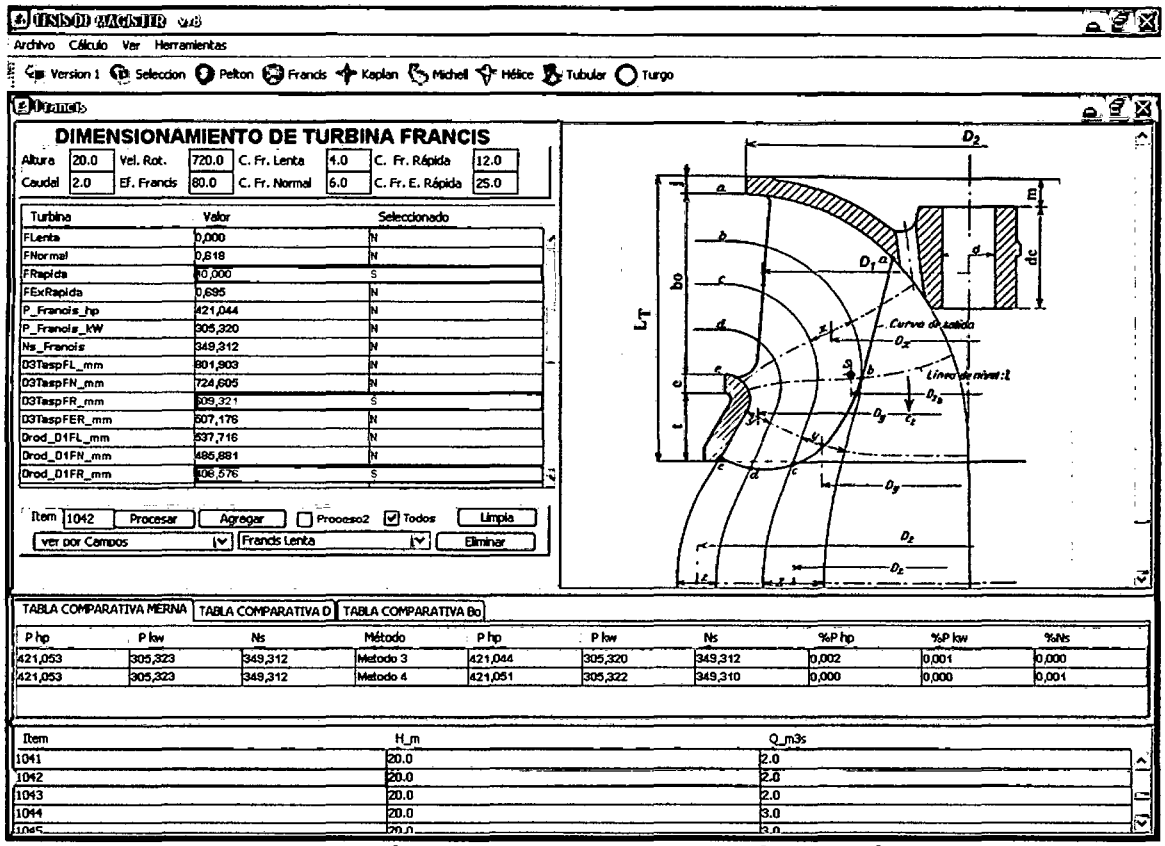


Fig. 5.2 Selección de una turbina Francis Rápida. Método 3.

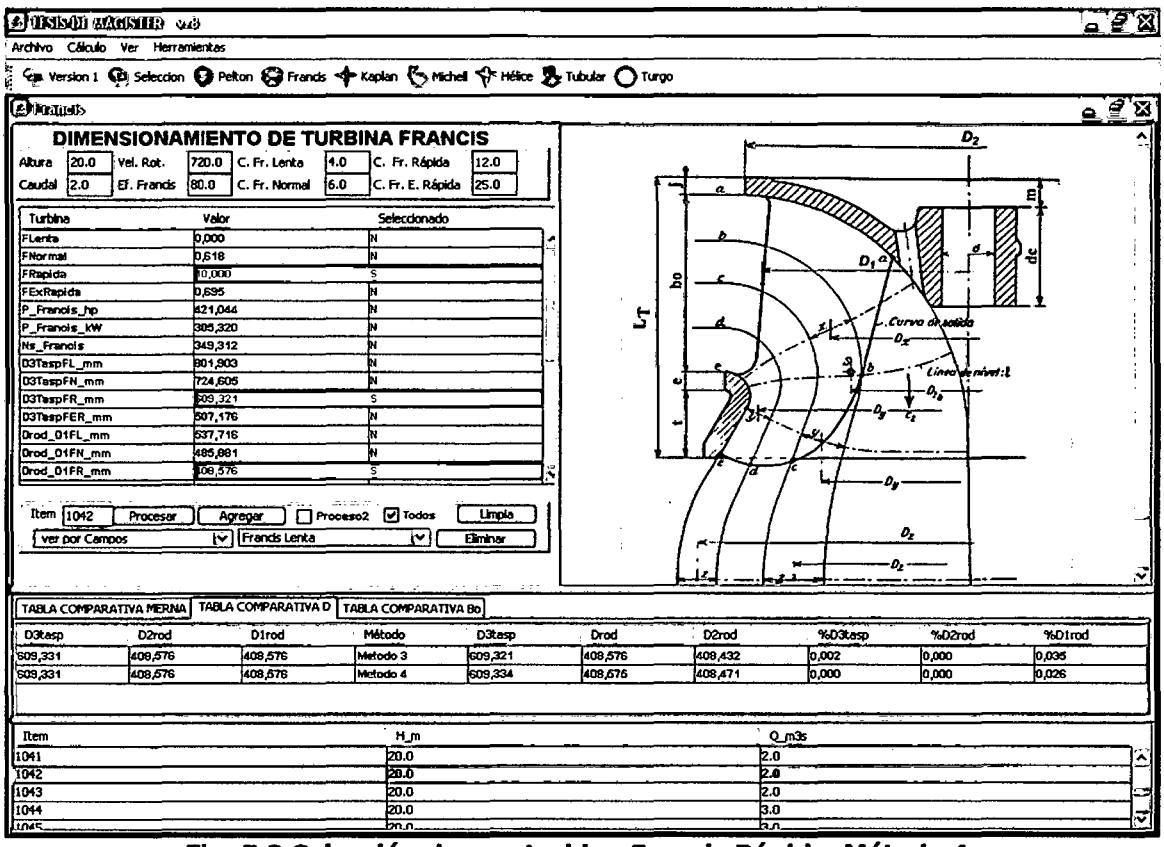


Fig. 5.3 Selección de una turbina Francis Rápida. Método 4.

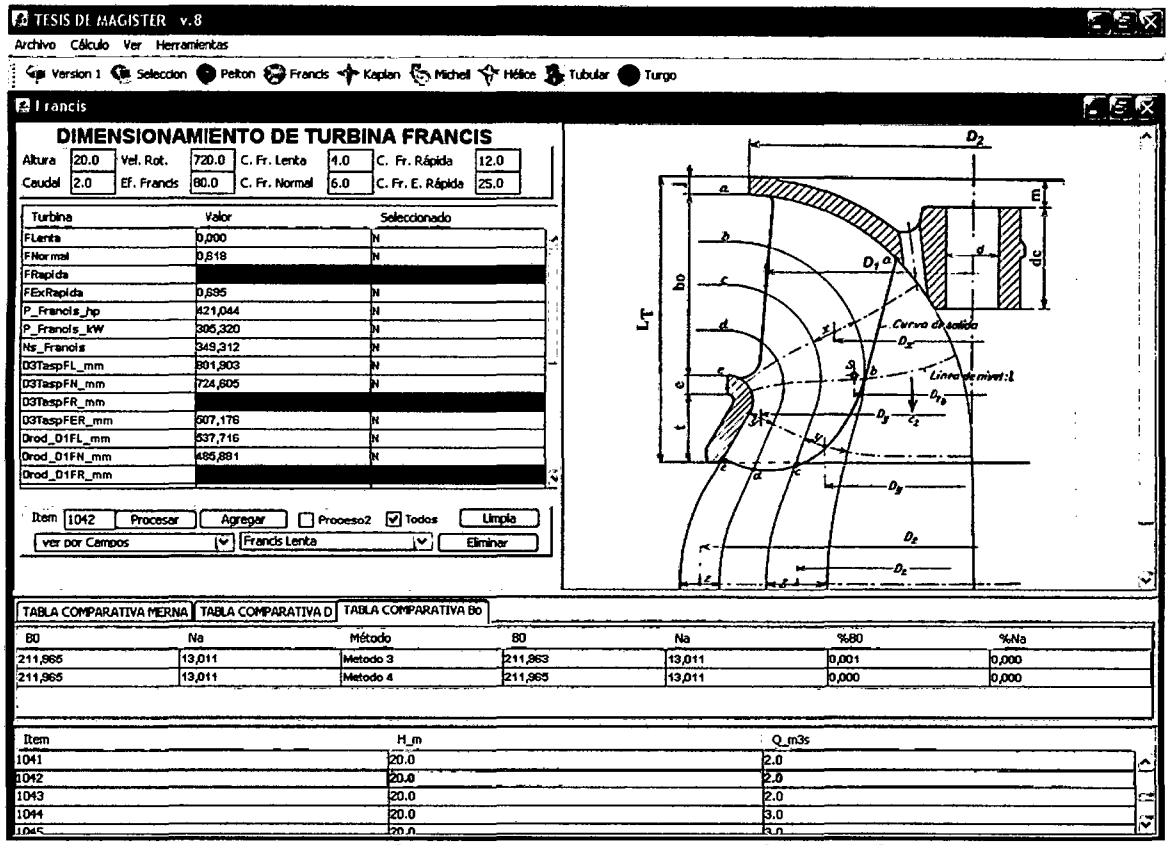


Fig. C.4 Selección de una turbina Francis Rápida. Método 4.

Tabla 5.1 Selección: Comparación de resultados.

Variable	Tradicional	MÉTODO			
		MERNA			
		Método 1	Método 2	Método 3	Método 4
P hp	421.053	421.044	421.063	421.044	421.051
P kw	305.323	305.320	305.327	305.320	305.322
Ns	349.312	349.317	349.316	349.312	349.310

Tabla 5.2 Desviación de los resultados.

Variable	DESVIACIÓN (%)			
	(M1) - Tradicional	(M2) - Tradicional	(M3) - Tradicional	(M4) - tradicional
P hp	0.002	0.002	0.002	0.000
P kw	0.001	0.001	0.001	0.000
Ns	0.002	0.002	0.000	0.001

En la tabla **5.1** mostramos una comparación en los resultados obtenidos por el método tradicional y por el método **MERNA** (métodos 1,2 y 3,4) y en la Tabla **5.2** la desviación en los resultados que se presentan.

En la tabla **5.1** observamos que en el cálculo del número específico de revoluciones (Ns), los cuatros métodos dan valores iguales al método tradicional, tanto en selección como en el dimensionamiento de la turbina Francis, las desviaciones se observan en la Tabla **5.2**.

En la Potencia en HP y en kW los cuatro métodos realizados, tanto en selección como en el dimensionamiento de la turbina Francis, las desviaciones son del 0.001%.

En la tabla **5.3** mostramos una comparación en los resultados en las dimensiones obtenidos por el método tradicional y por el método **MERNA** (métodos 3 y4) y en la Tabla **5.4** la desviación en los resultados que se presentan.

Tabla 5.3 Comparación de resultados.

	Francis Rápida		
Variable	Tradicional	MERNA	
		Método 3	Método 4
D3tasp	609.331	609.321	609.334
$D1_{rod}$	408.576	408.576	408.575
$D2_{cor}$	665.839	665.822	665.844

Tabla 5.4 Desviación de los resultados.

	DESVIACIÓN (%)	
Variable	Método 3 - Tradicional	Método 4 - tradicional
D3tasp	0.002	0.000
$D1_{rod}$	0.000	0.010
$D2_{cor}$	0.001	0.010

7. Al ingresar los datos de **H = 200 m, Q = 1 m^3/seg, Vr = 1200 rpm** y eficiencia de la turbina **η = 70%;** el programa **MERNA** selecciona **Turbina Pelton 4ch**, como se observan en las figuras **5.5**, **5.6**, **5.7** y **5.8**.

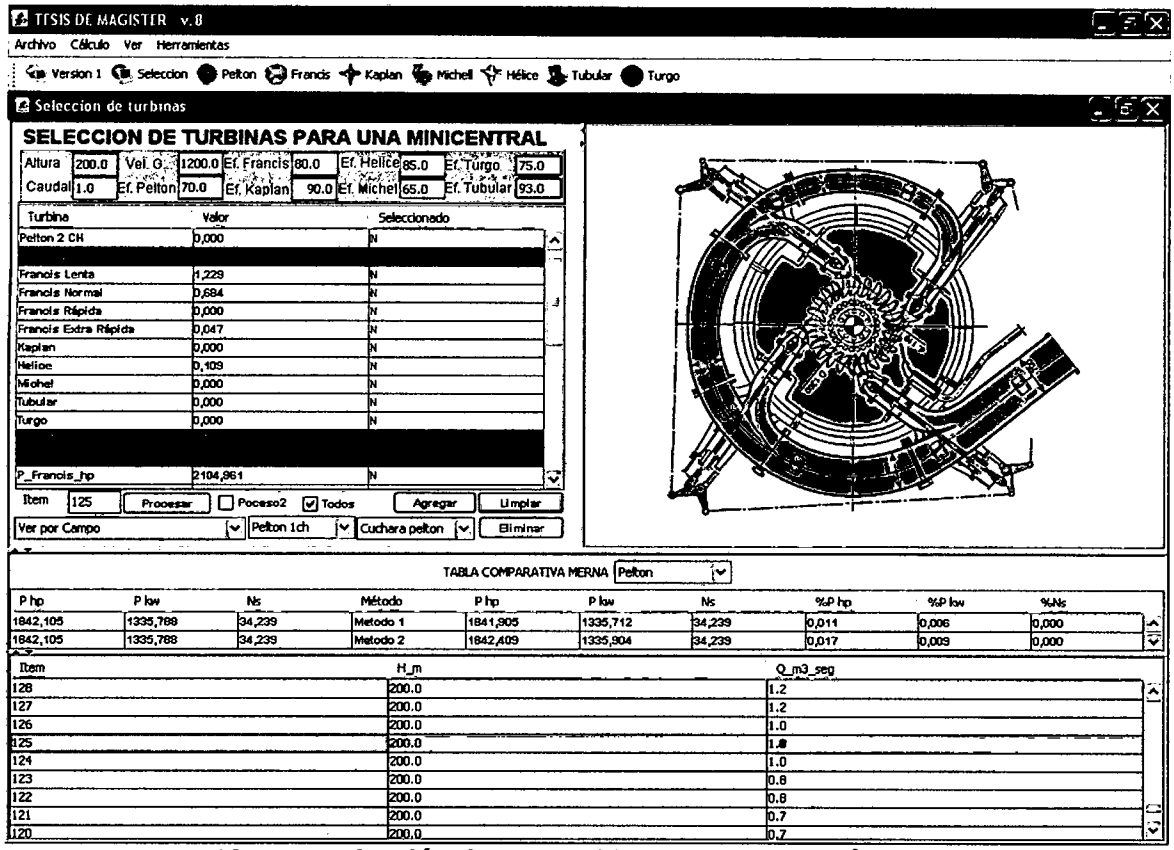


Fig. 5.5 Selección de una turbina Pelton 4ch. Método 1.

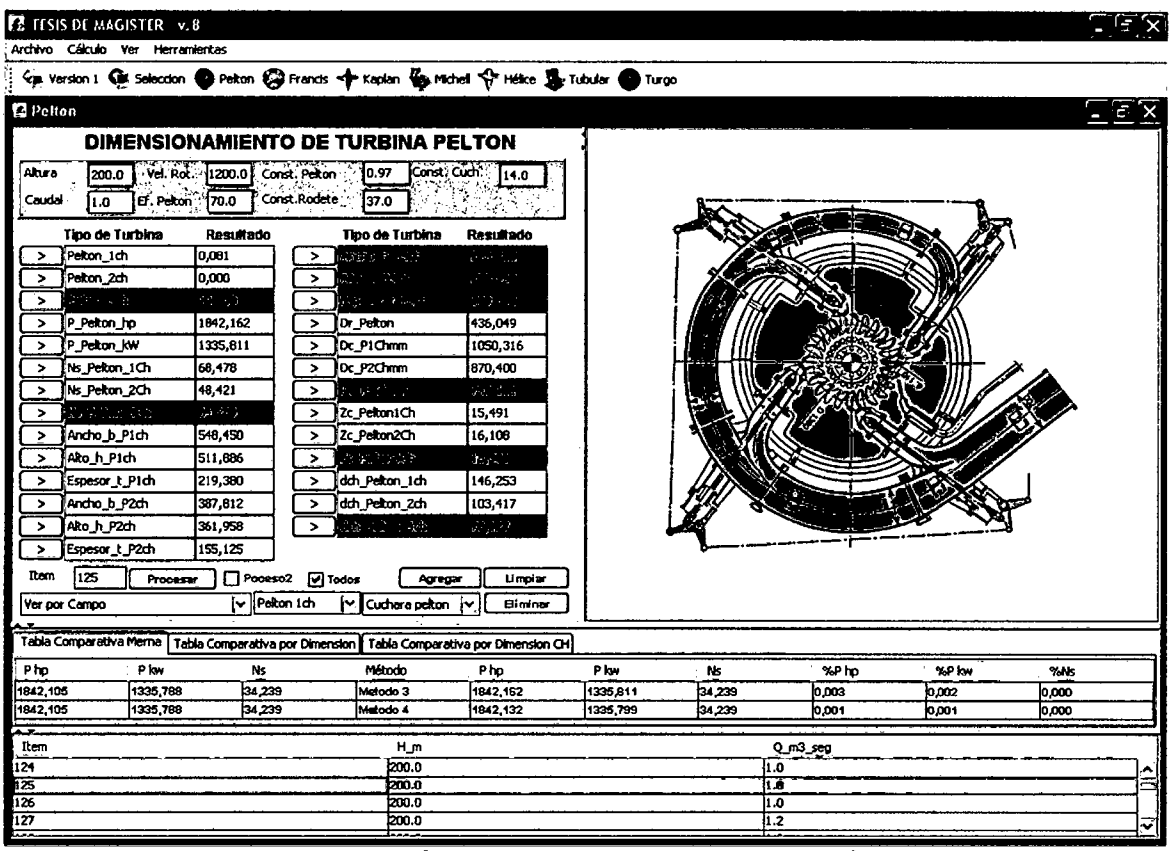


Fig. 5.6 Selección de una turbina Pelton 4ch. Método 2.

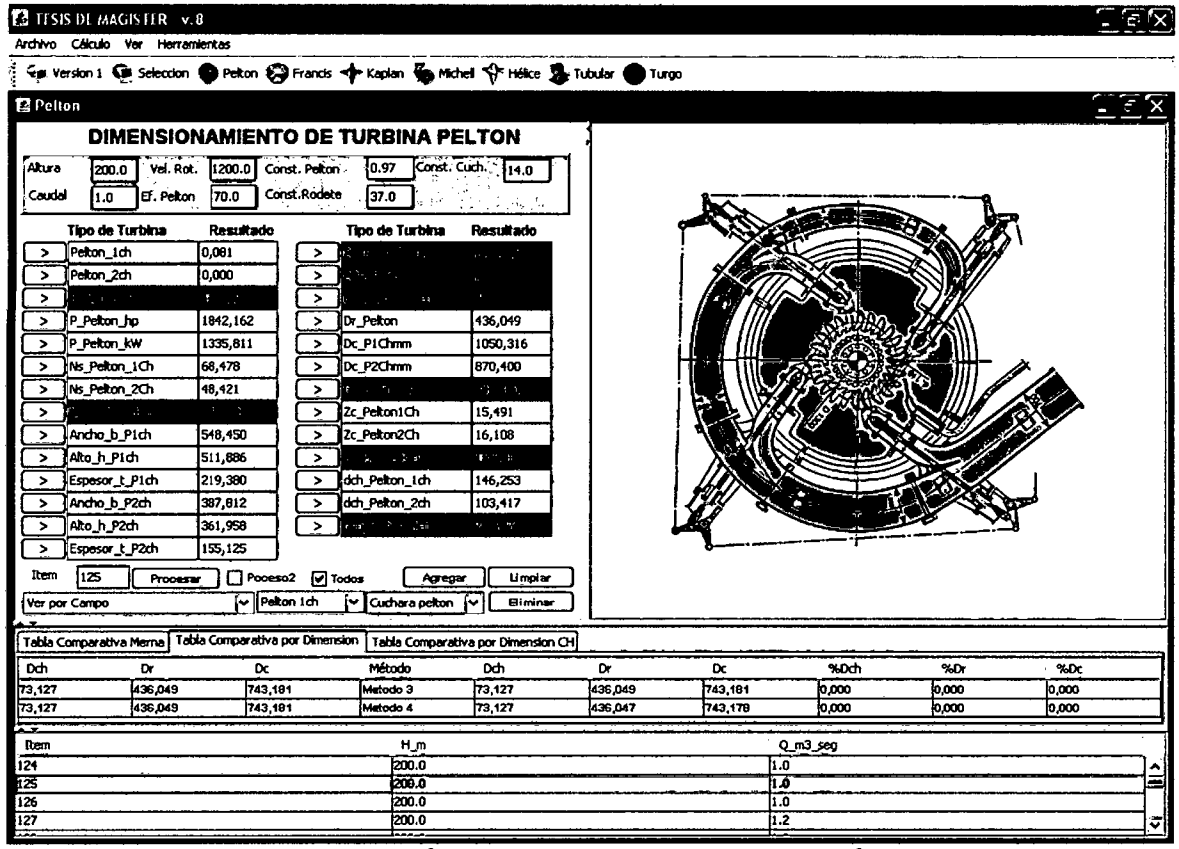


Fig. 5.7 Selección de una turbina Pelton 4ch. Método 3.

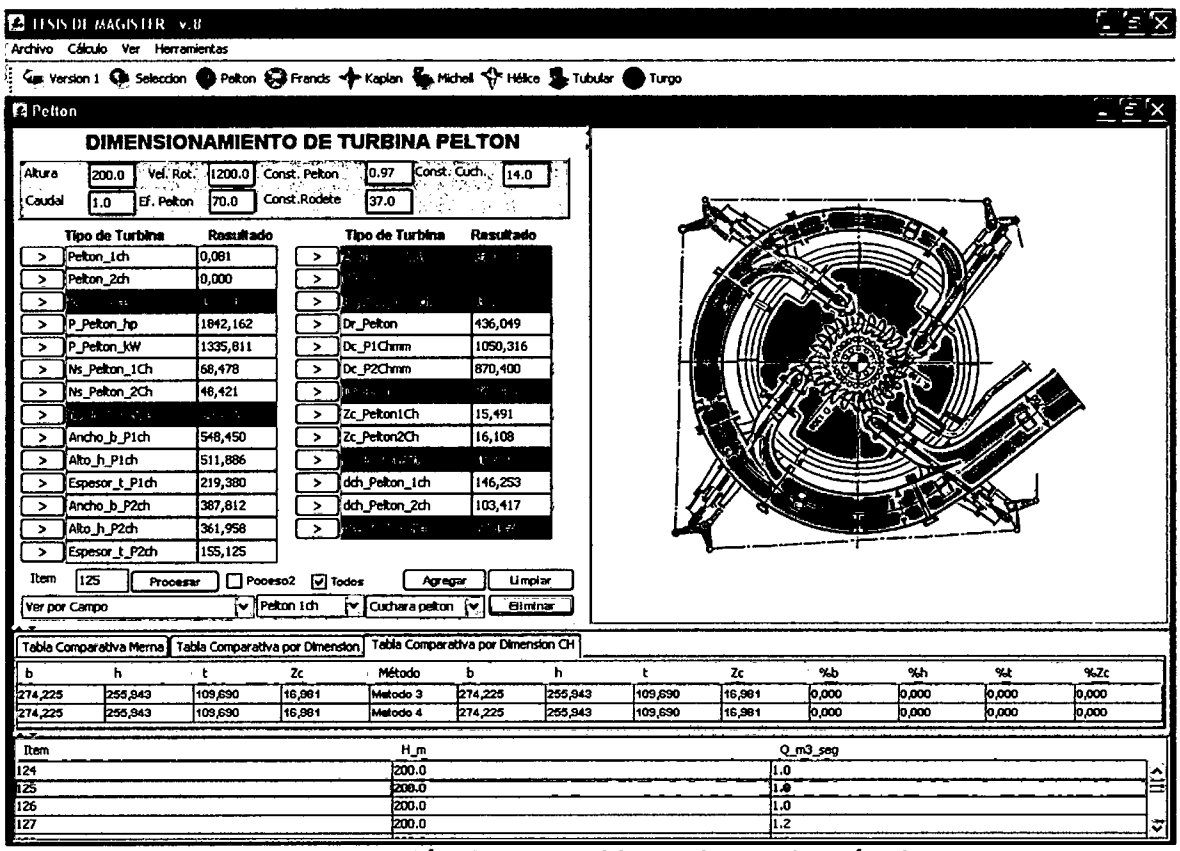


Fig. 5.8 Selección de una turbina Pelton 4ch. Método 4.

En la tabla **5.5** mostramos una comparación en los resultados obtenidos por el método tradicional y por el método **MERNA** (métodos 1,2 y 3,4) y en la Tabla **5.6** la desviación en los resultados que se presentan.

Tabla 5.5 Comparación de resultados.

	MÉTODO				
	Tradicional	MERNA			
Variable		Método1	Método 2	Método 3	Método 4
P hp	1842.105	1841.905	1842.409	1842.105	1842.105
P kw	1335.788	1335.712	1335.904	1335.788	1335.788
Ns	34.239	34.239	34.239	34.239	34.239

Tabla 5.6 Desviación de los resultados.

	DESVIACIÓN (%)			
Variable	M1 - Tradicional	M2 - Tradicional	M3 - Tradicional	M4 – Tradicional
P hp	0.011	0.017	0.003	0.001
P kw	0.006	0.009	0.002	0.001
Ns	0.000	0.000	0.000	0.000

En la tabla **5.6** observamos la desviación de los resultados, en el cálculo del número específico de revoluciones (**Ns**), los cuatro métodos dan los mismos resultados que el método tradicional, cuyas desviaciones son del 0.000%. En la Potencia en HP y en kW los métodos 3 y 4 del dimensionamiento de la turbina Pelton 4ch tienen una desviación del 0.003% y 0.001% respectivamente.

En la tabla **5.7** mostramos una comparación de los resultados en las dimensiones obtenidos por el método tradicional y por el método **MERNA** (métodos 3 y 4) y en la Tabla **5.8** la desviación de los resultados que se presentan, observamos que los método 3 y 4 se obtienen las mismas desviaciones.

Tabla 5.7 Comparación de resultados.

Variable	Tradicional	Pelton 4ch	
		MERNA	
		Método 3	Método 4
D_{ch}	73.127	73.127	73.127
D_r	436.059	436.049	436.047
D_c	743.181	743.181	743.178

Tabla 5.8 Desviación de los resultados.

	Pelton 4ch	
	DESVIACIÓN (%)	
Variable	Método 3 - Tradicional	Método 4 - tradicional
Dch	0.000	0.000
Dr	0.000	0.000
Dc	0.000	0.000

8. Al ingresar los datos de **H = 100 m, Q = 0.5 m^3/seg, Vr = 1800 rpm** y eficiencia de la turbina **η = 80%;** el programa **MERNA** selecciona **Turbina Francis Normal**, como se observan en las figuras 5.9, 5.10, 5.11 y 5.12.

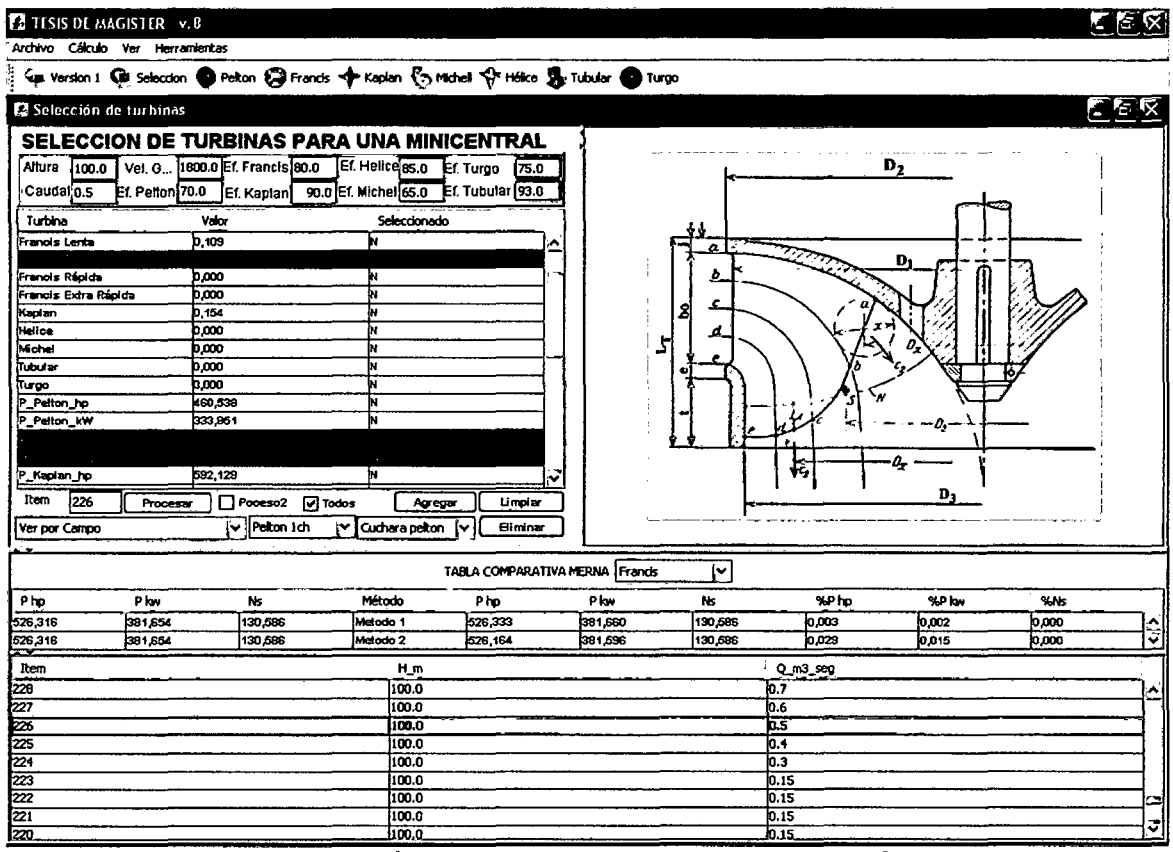


Fig. 5.9 Selección de una turbina Francis Normal. Método 1.

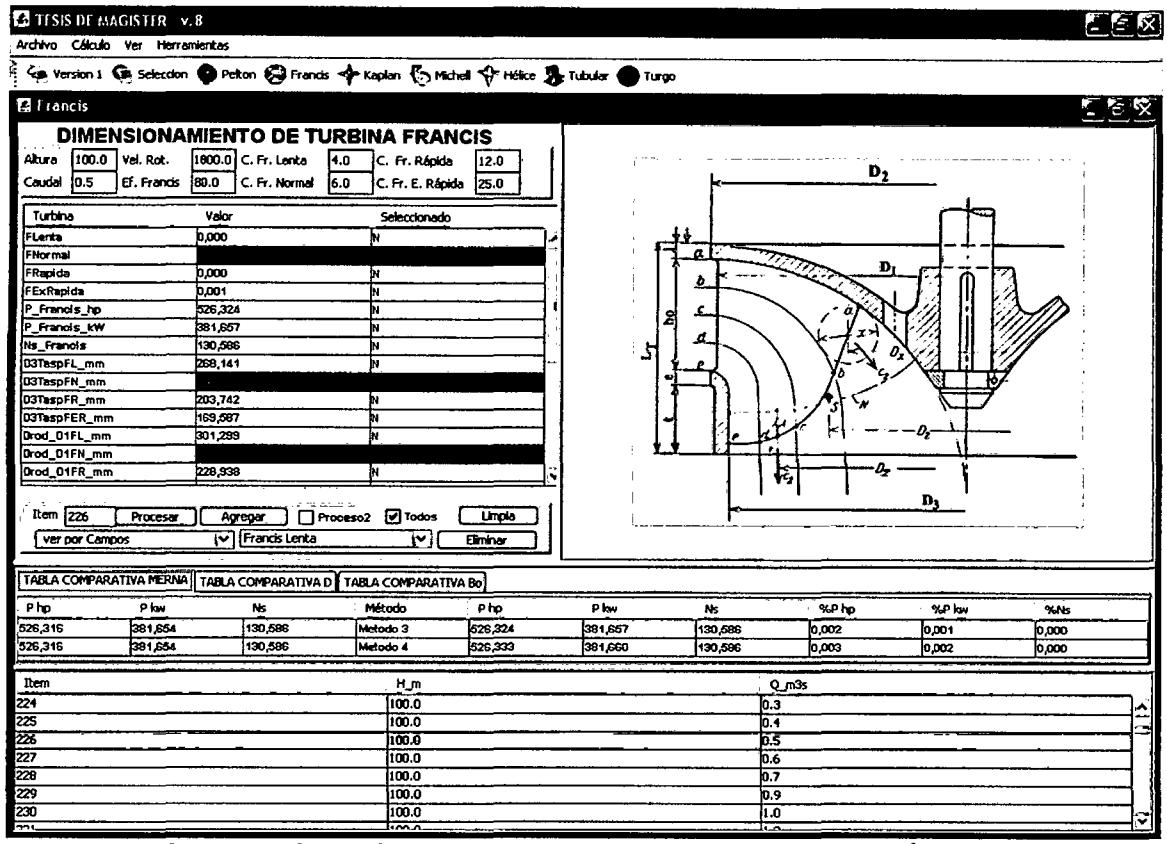


Fig. 5.10 Dimensiones de una turbina Francis Normal. Método 3.

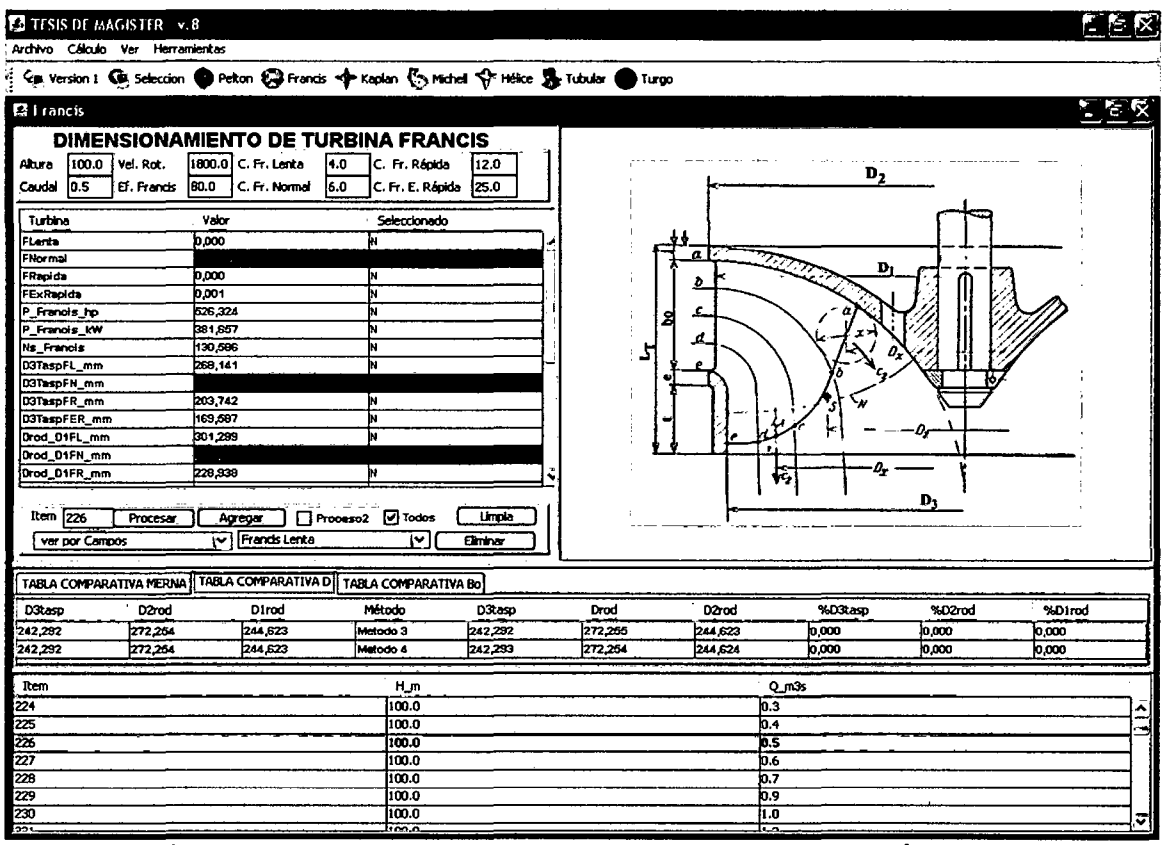


Fig.5.11 Dimensiones de una turbina Francis Normal. Método 4.

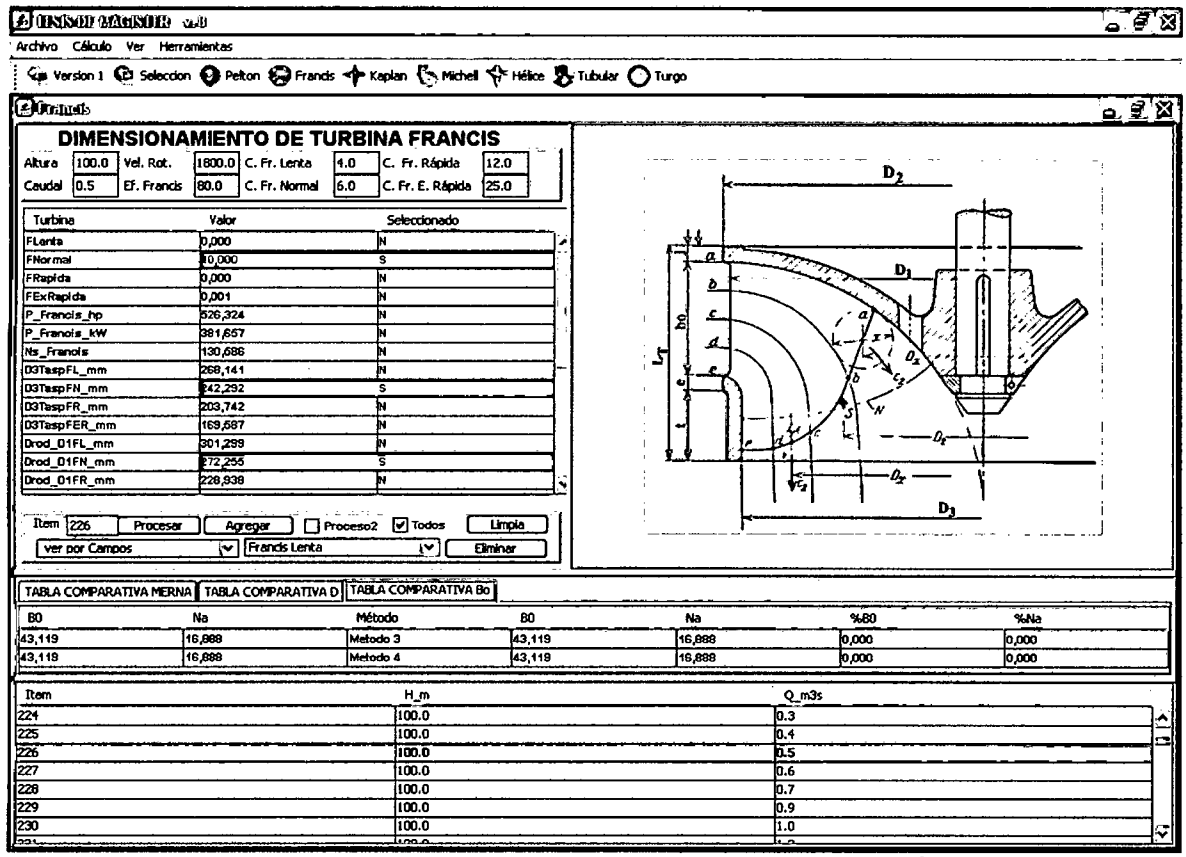


Fig. 5.12 Dimensiones de una turbina Francis Normal. Método 4.

En la tabla **5.9** mostramos una comparación en los resultados obtenidos por el método tradicional y por el método **MERNA** (métodos 1,2 y 3,4) y en la tabla **5.10** la desviación que se presentan en los resultados obtenidos.

Tabla 5.9 Comparación de resultados.

	MÉTODO				
	Tradicional	MERNA			
Variable		Método1	Método 2	Método 3	Método 4
P hp	526.316	526.333	526.164	526.324	526.333
P kw	381.654	381.660	381.596	381.657	381.660
Ns	130.586	130.586	130.586	130.586	130.586

Tabla 5.10 Desviación de los resultados

	Francis Normal			
	DESVIACIÓN (%)			
Variable	M1 - Tradicional	M2 - Tradicional	M3 - Tradicional	M4 - Tradicional
P hp	0.003	0.029	0.002	0.000
P kw	0.002	0.015	0.001	0.002
Ns	0.000	0.000	0.000	0.000

En la tabla **5.9** observamos que en el cálculo del número específico de revoluciones (Ns), los cuatro métodos dan el mismo valor que el valor calculado en el método tradicional, sus desviaciones son del 0.000%. En la Tabla **5.10** la Potencia en HP y en kW los métodos 3 y 4 tiene una desviación del 0.002% y 0.001% respectivamente.

En la tabla **5.11** mostramos una comparación en los resultados de las dimensiones obtenidos por el método tradicional y por el método **MERNA** (métodos 3 y 4) y en la Tabla **5.12** la desviación en los resultados que se presentan; asimismo observamos que con el método 3 se obtienen las desviaciones más bajas.

Tabla 5.11 Comparación de resultados.

	Francis Normal		
Variable	Tradicional	MERNA	
		Método 3	Método 4
D_{3tasp}	242.292	242.292	242.293
D_{rod}	272.254	272.255	272.254
D_{2cor}	244.623	244.623	244.624

Tabla 5.12 Desviación de los resultados.

	Francis Normal	
	DESVIACIÓN (%)	
Variable	Método 3 – Tradicional	Método 4 - tradicional
D_{3tasp}	0.000	0.000
D_{rod}	0.000	0.000
D_{2cor}	0.000	0.000

9. Al ingresar los datos de **H = 50 m, Q = 0.2 m^3/seg, Vr = 1200 rpm** y eficiencia de la turbina **η = 65%;** el programa **MERNA** selecciona **Turbina Michell**, como se observan en las figuras 5.13, 5.14, 5.15 y 5.16.

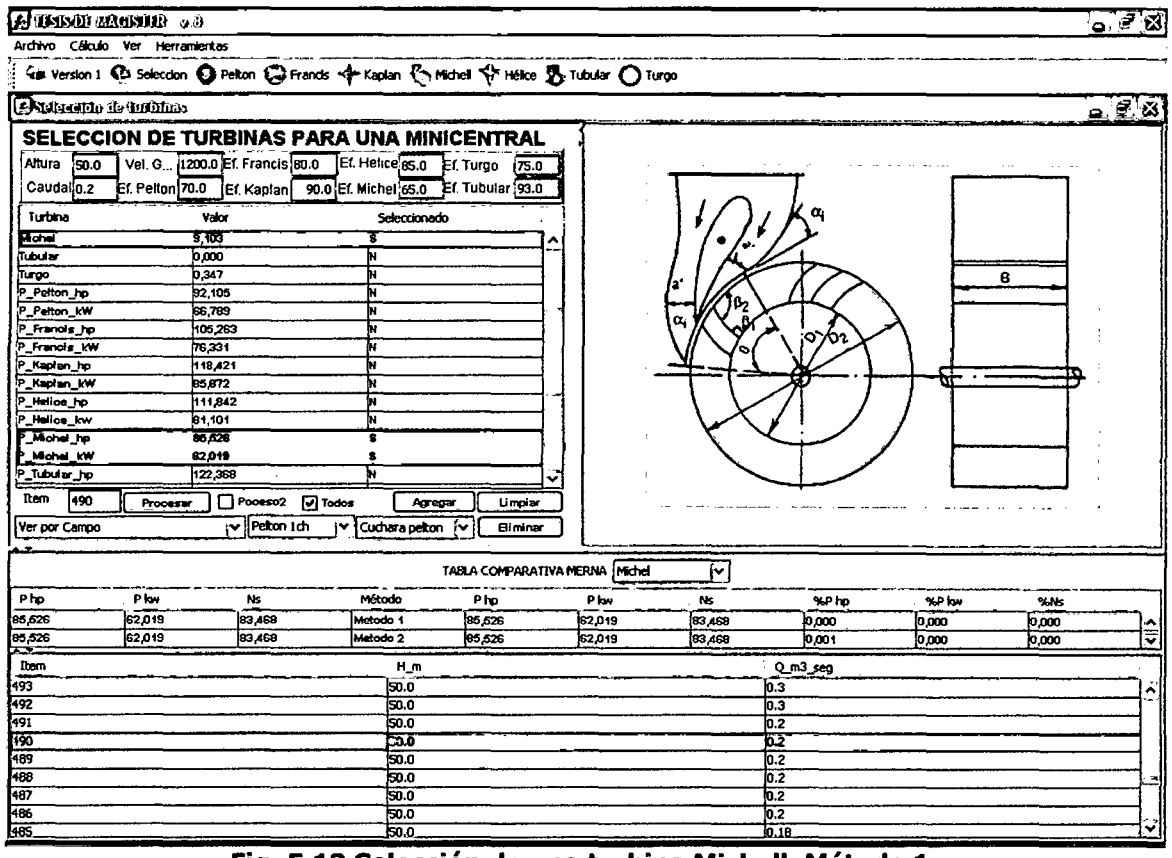


Fig. 5.13 Selección de una turbina Michell. Método 1.

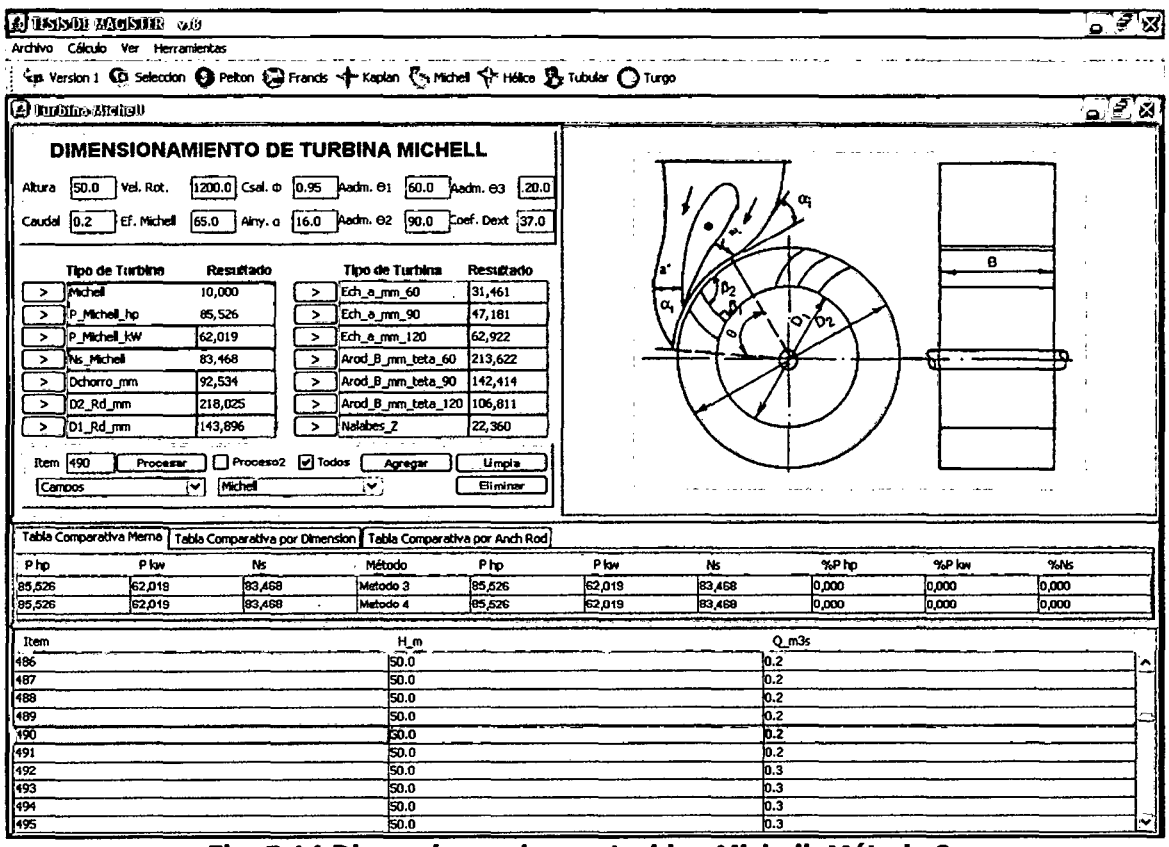


Fig. 5.14 Dimensiones de una turbina Michell. Método 2.

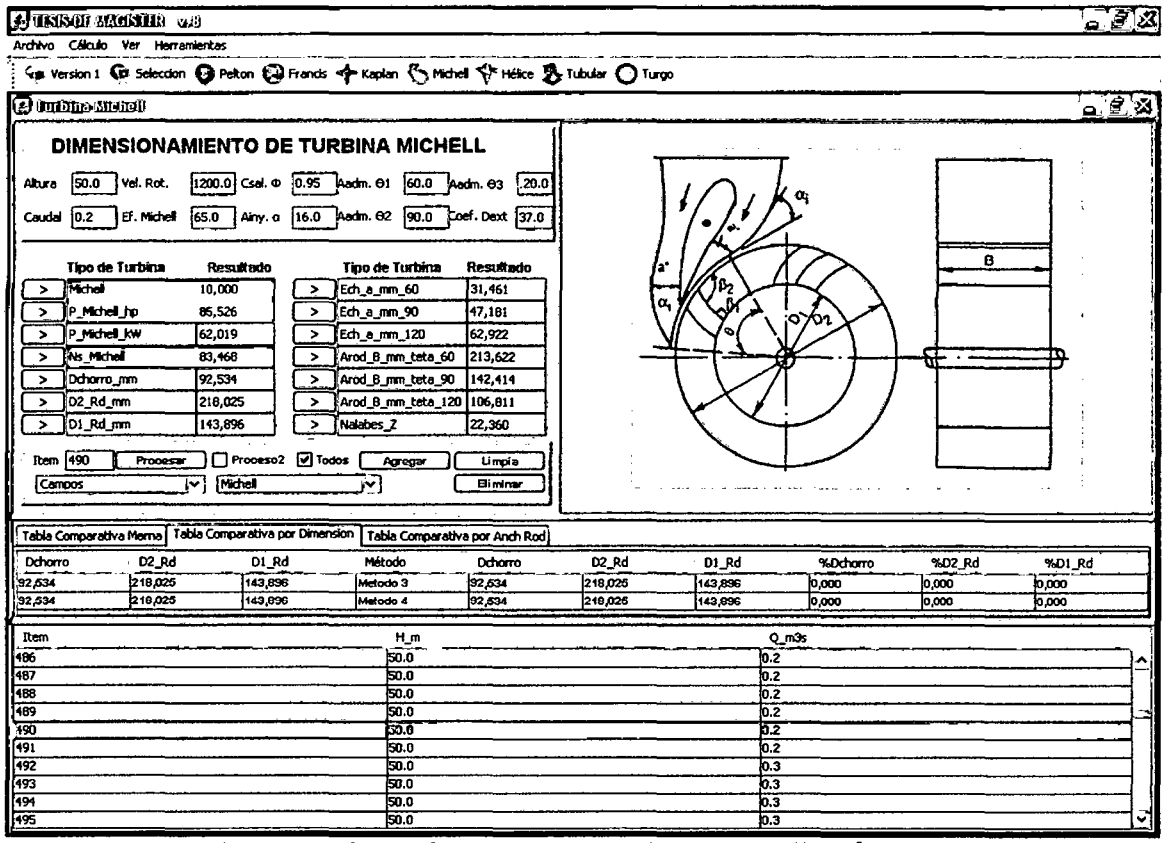


Fig. 5.15 Dimensiones de una turbina Michell. Método 3.

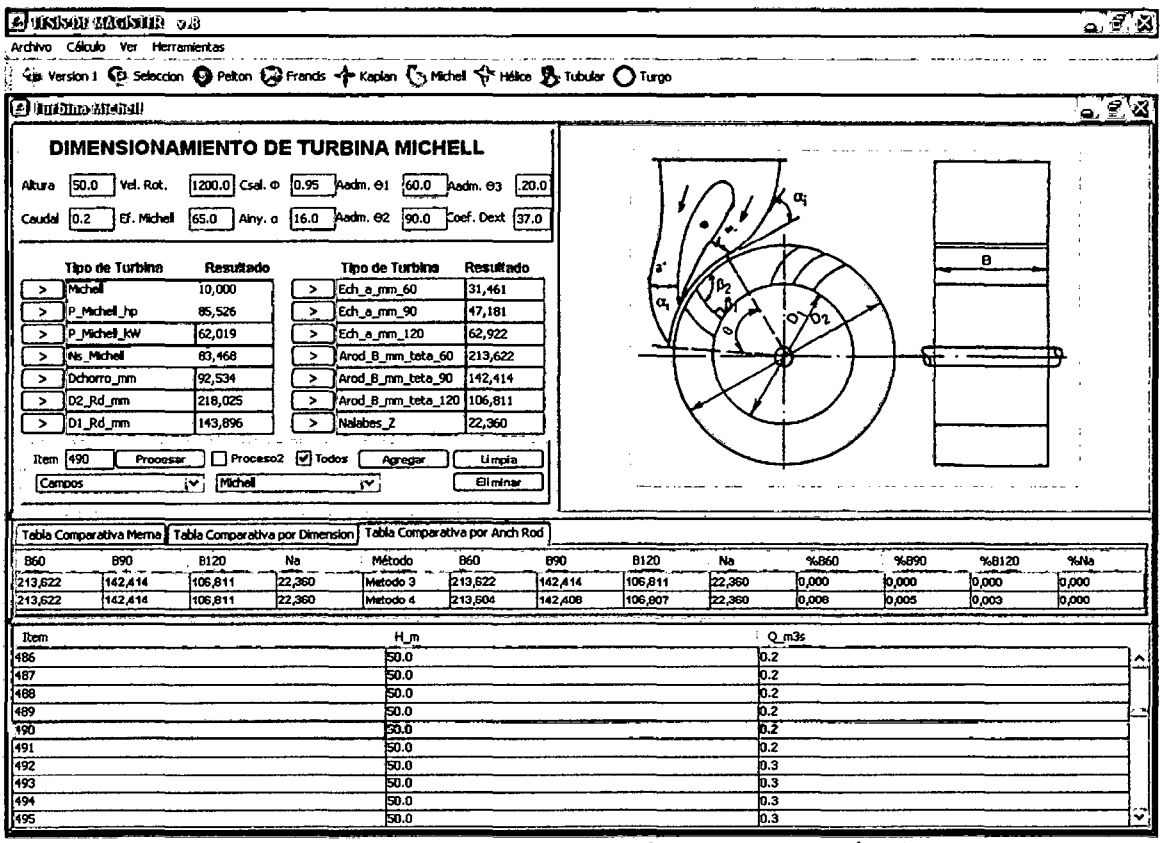


Fig. 5.16 Dimensiones de una turbina Michell. Método 4.

En la tabla **5.13** mostramos una comparación en los resultados obtenidos por el método tradicional y por el método **MERNA** (métodos 1,2 y 3,4) y la desviación en los resultados que se presentan.

Tabla 5.13 Comparación de resultados.

	MÉTODO				
	Tradicional	MERNA			
Variable		Método 1	Método 2	Método 3	Método 4
P hp	85.526	85.526	85.526	85.526	85.526
P kw	62.019	62.019	62.019	62.019	62.019
Ns	83.468	83.468	83.468	83.468	83.468

Tabla 5.14 Desviación de los resultados

	Michell			
	DESVIACIÓN (%)			
Variable	M1 - Tradicional	M2 - Tradicional	M3 - Tradicional	M4 - tradicional
P hp	0.000	0.000	0.000	0.000
P kw	0.000	0.000	0.000	0.000
Ns	0.000	0.000	0.000	0.000

En las tablas **5.13** y **5.14** observamos que en el cálculo del número específico de revoluciones (Ns), los cuatro métodos dan el mismo valor que el calculado por el método tradicional, sus desviaciones son del 0.000%. En la Potencia en HP y en kW los cuatro métodos dan el mismo resultado al valor calculado por el método tradicional, sus desviaciones son del 0.000%.

En la tabla **5.15** mostramos una comparación de los resultados de las dimensiones obtenidos por el método tradicional y por el método **MERNA** (métodos 3 y 4) y en la Tabla **5.16** la desviación en los resultados que se presentan; observamos que las desviaciones son del 0.000%.

Tabla 5.15 Comparación de resultados.

Variable	Tradicional	Michell	
		MERNA	
		Método 3	Método 4
D_{chorro}	92.534	92.534	92.534
D_{2Rd}	218.025	218.025	218.025
D_{1Rd}	143.896	143.896	143.896

Tabla 5.16 Desviación de los resultados.

	Michell	
	DESVIACIÓN (%)	
Variable	Método 3 - Tradicional	Método 4 - tradicional
D_{chorro}	0.000	0.000
D_{2Rd}	0.000	0.000
D_{1Rd}	0.000	0.000

10. Al ingresar los datos de **H = 8 m, Q = 4.0 m^3/seg, Vr = 600 rpm** y eficiencia de la turbina **η = 85%;** el programa **MERNA** selecciona **Turbina Hélice**, como se observan en las figuras 5.17, 5.18, 5.19 y 5.20.

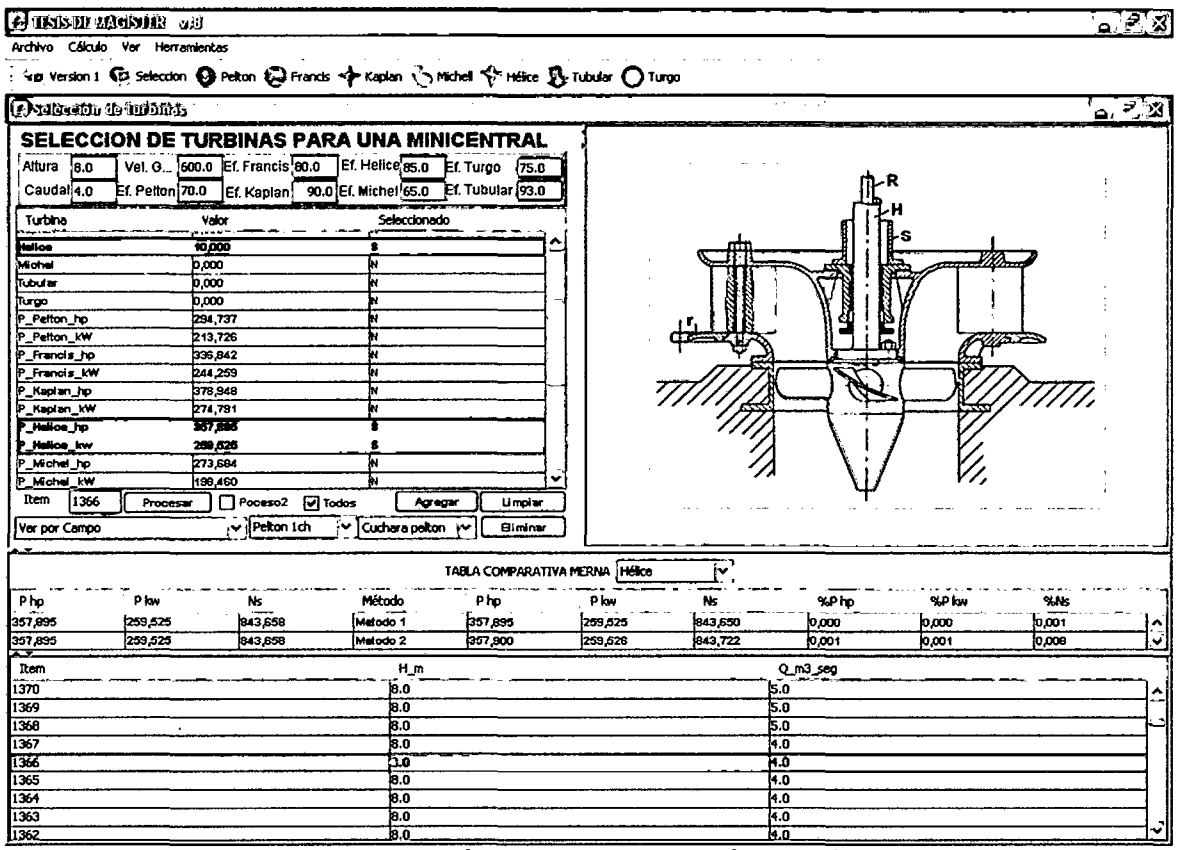


Fig. 5.17 Selección de una turbina Hélice. Método 1.

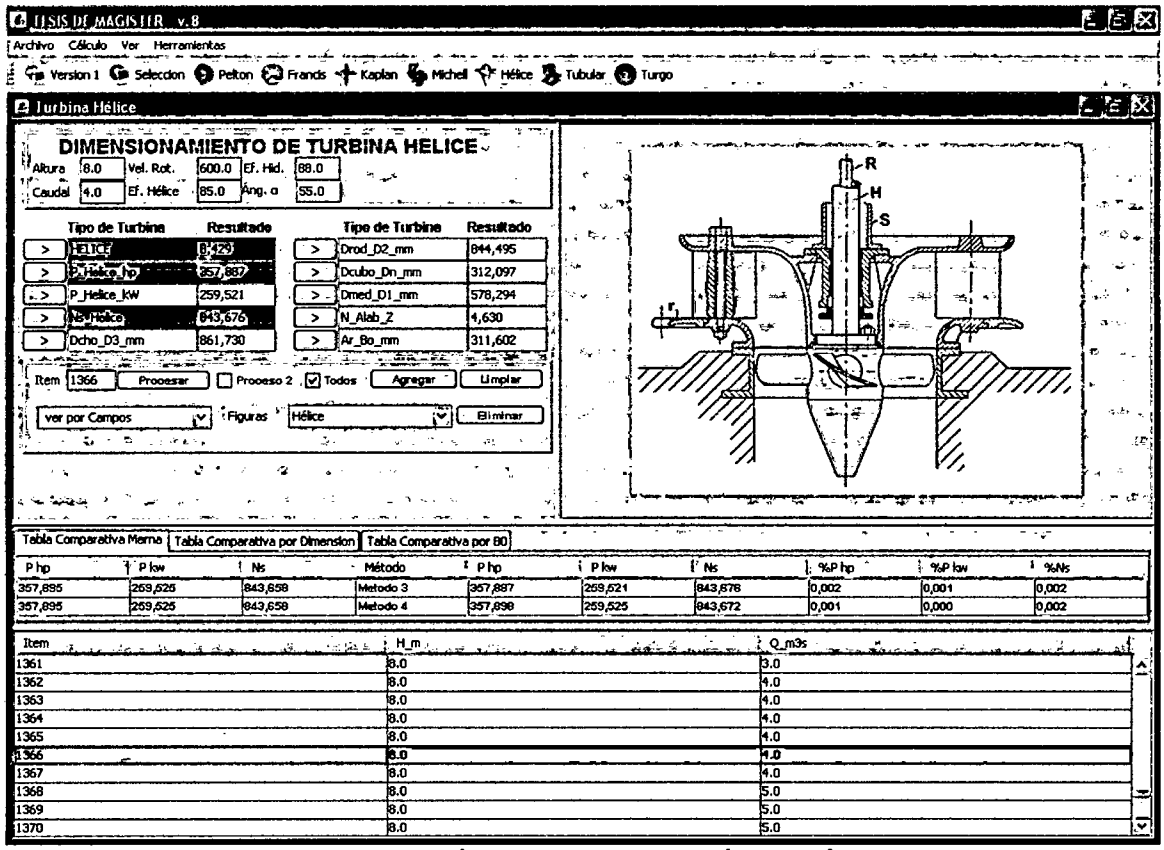


Fig. 5.18 Selección de una turbina Hélice. Método 3.

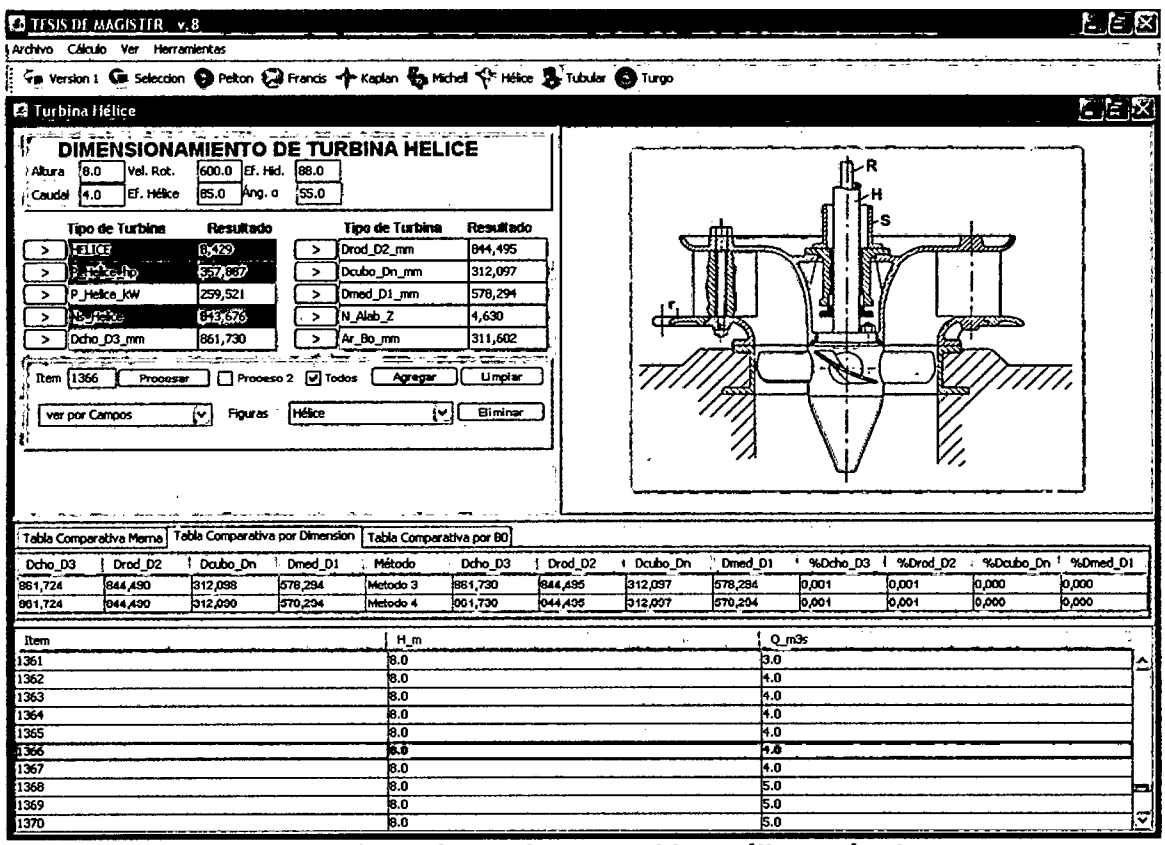


Fig. 5.19 Dimensiones de una turbina Hélice. Método 3.

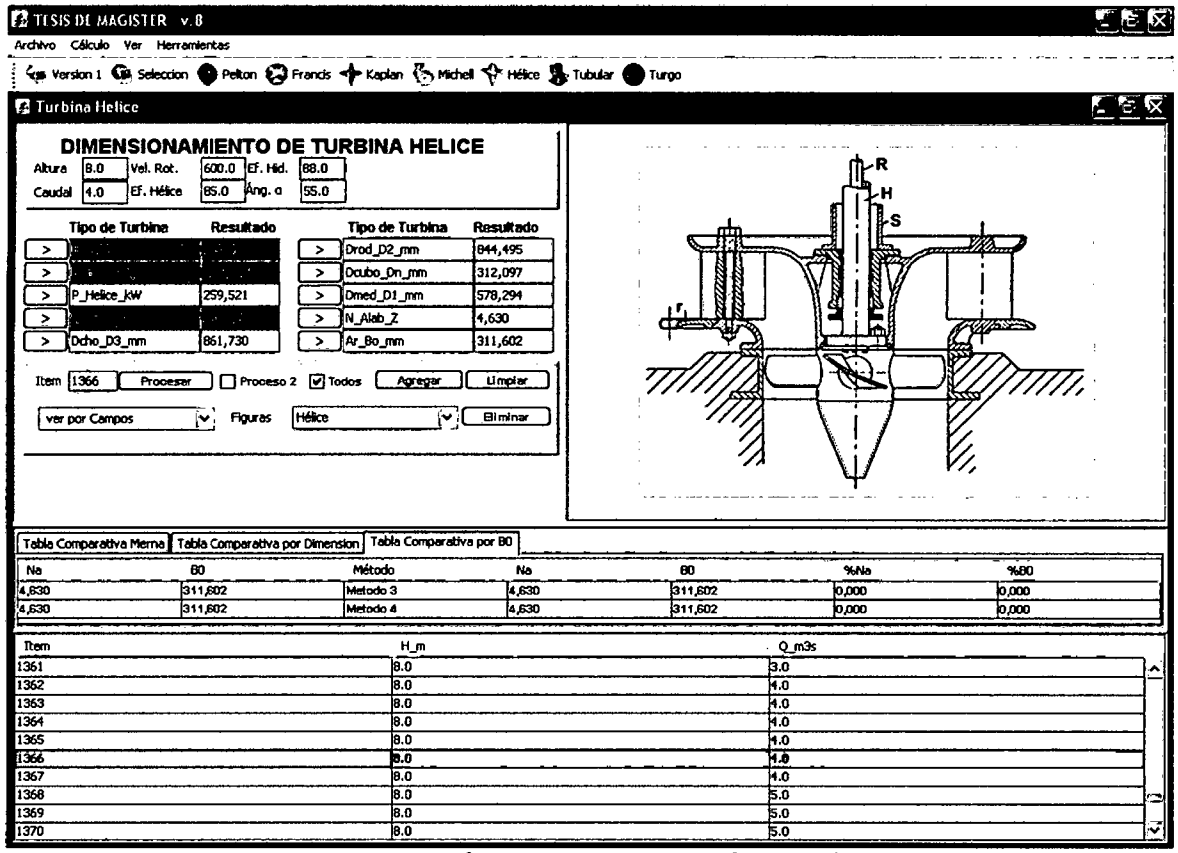


Fig. 5.20 Selección de una turbina Hélice. Método 4.

En la tabla **5.17** mostramos una comparación en los resultados obtenidos por el método tradicional y por el método **MERNA** (métodos 1,2 y 3,4) y en la Tabla **5.18** se muestran la desviación en los resultados que se obtienen.

Tabla 5.17 Comparación de resultados.

	Hélice				
	Tradicional	MERNA			
Variable		Método 1	Método 2	Método 3	Método 4
P hp	357.895	357.895	357.900	357.887	357.898
P kw	259.525	259.525	259.526	259.521	259.525
Ns	843.658	843.650	843.722	843.676	843.672

En la tabla **5.17** observamos que en el cálculo del número específico de revoluciones (Ns), en los cuatro métodos se han calculado valores casi idénticos al método tradicional; y en la Tabla **5.18** las desviaciones son del 0.001% y 0.008% en los métodos 1 y 2 (MERNA).

En el cálculo de la Potencia en HP y en kW, los cuatro métodos del MERNA dan valores casi idénticos al valor calculado por el método tradicional, siendo la desviación del 0.001%, como se observan en las Tablas **5.17** y **5.18**.

Tabla 5.18 Desviación de los resultados

	Hélice			
	DESVIACIÓN (%)			
Variable	M1 - Tradicional	M2 - Tradicional	M3 - Tradicional	M4 - tradicional
P hp	0.000	0.001	0.002	0.001
P kw	0.000	0.001	0.001	0.000
Ns	0.001	0.008	0.002	0.002

En la tabla **5.19** mostramos una comparación en los resultados de las dimensiones obtenidas por el método tradicional y por el método **MERNA** (métodos 3 y 4), observamos que las dimensiones halladas por los métodos 3 y 4, son casi idénticas a los valores calculados por el método tradicional.

Tabla 5.19 Comparación de resultados.

	Hélice		
Variable	Tradicional	MERNA	
		Método 3	Método 4
D_{chorro}	861.724	861.730	861.730
D_{rod}	844.490	844.495	844.495
D_{cub}	312.098	312.097	312.097
D_{med}	578.294	578.294	578.294

En la Tabla **5.20** se muestran la desviación de las dimensiones calculadas de la turbina Hélice por los métodos 3 y 4 es del 0.001%.

Tabla 5.20 Desviación de los resultados.

	Hélice	
	DESVIACIÓN (%)	
Variable	Método 3 - Tradicional	Método 4 - tradicional
D_{chorro}	0.001	0.001
D_{rod}	0.001	0.001
D_{cub}	0.000	0.000
D_{med}	0.000	0.000

11. Al ingresar los datos de **H = 15 m, Q = 15.0 m^3/seg, Vr = 257 rpm** y eficiencia de la turbina **η = 90%;** el programa **MERNA** selecciona **Turbina Kaplan**, como se observan en las figuras 5.21, 5.22, 5.23 y 5.24.

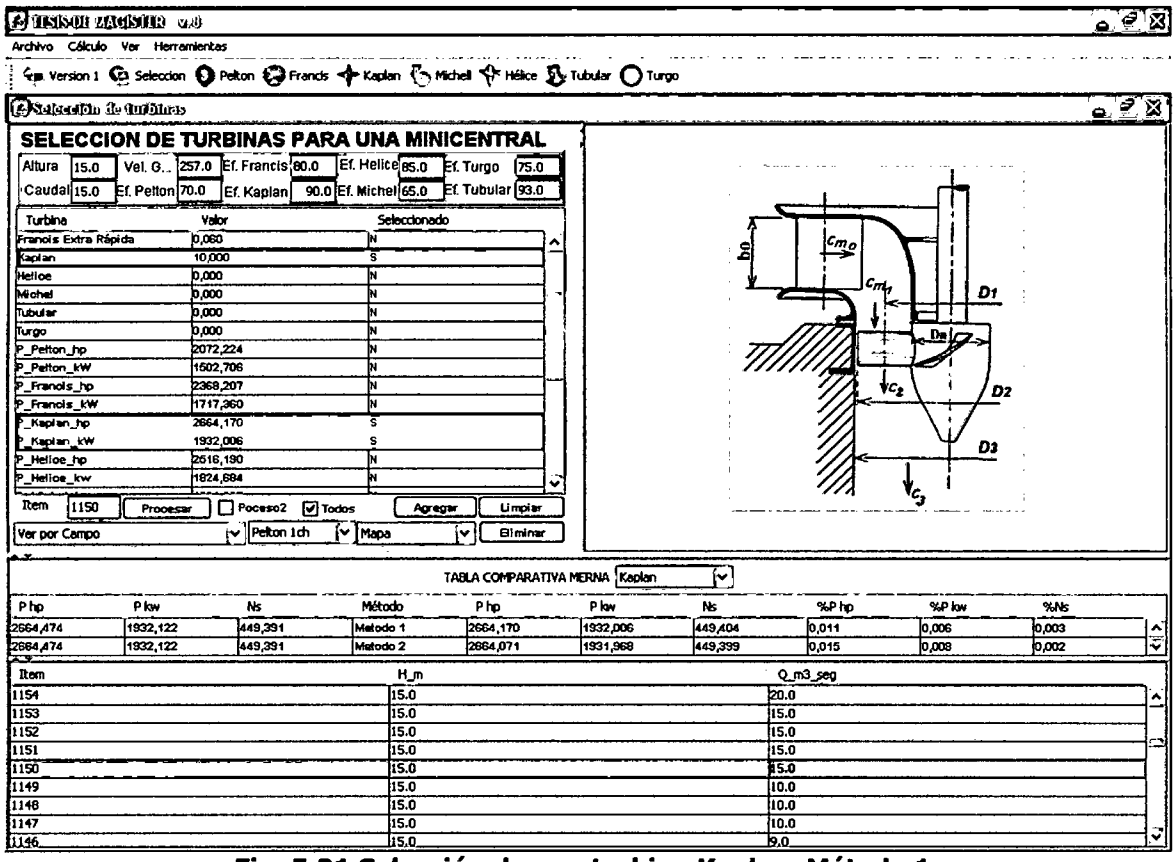


Fig. 5.21 Selección de una turbina Kaplan. Método 1.

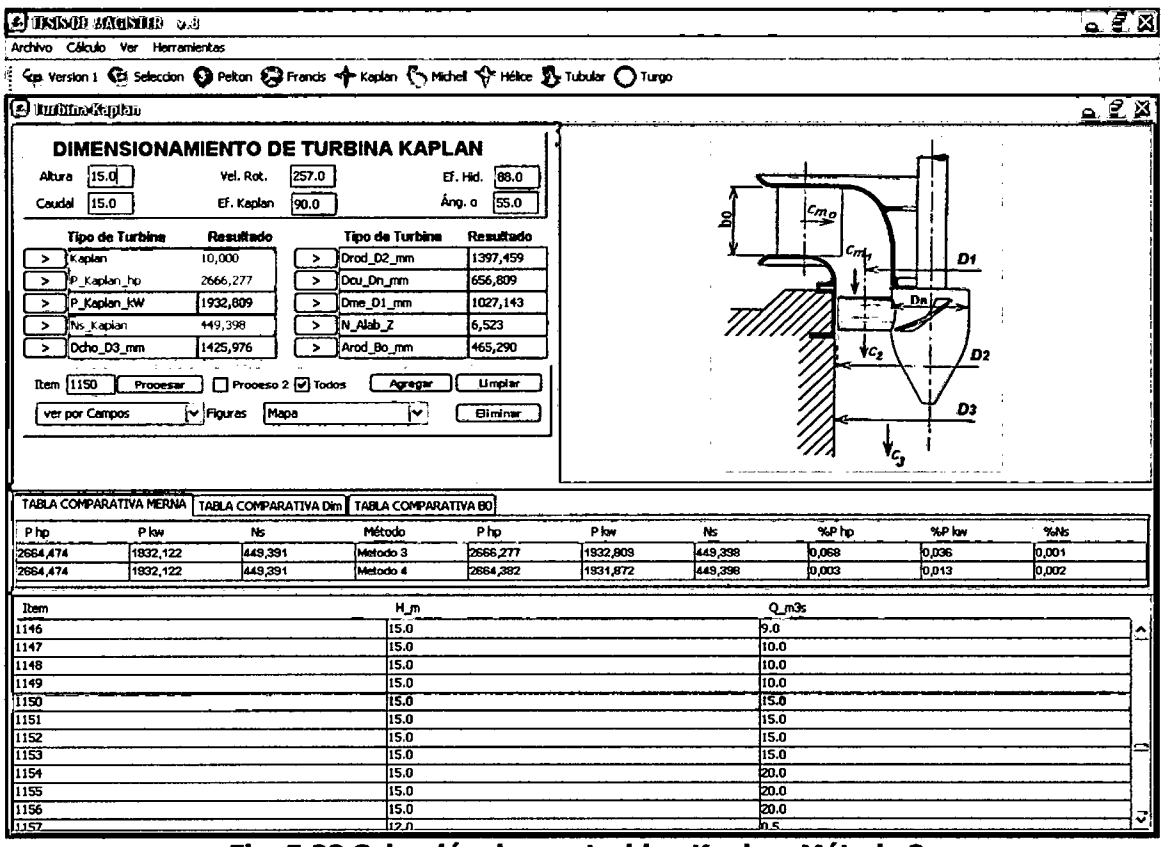


Fig. 5.22 Selección de una turbina Kaplan. Método 3.

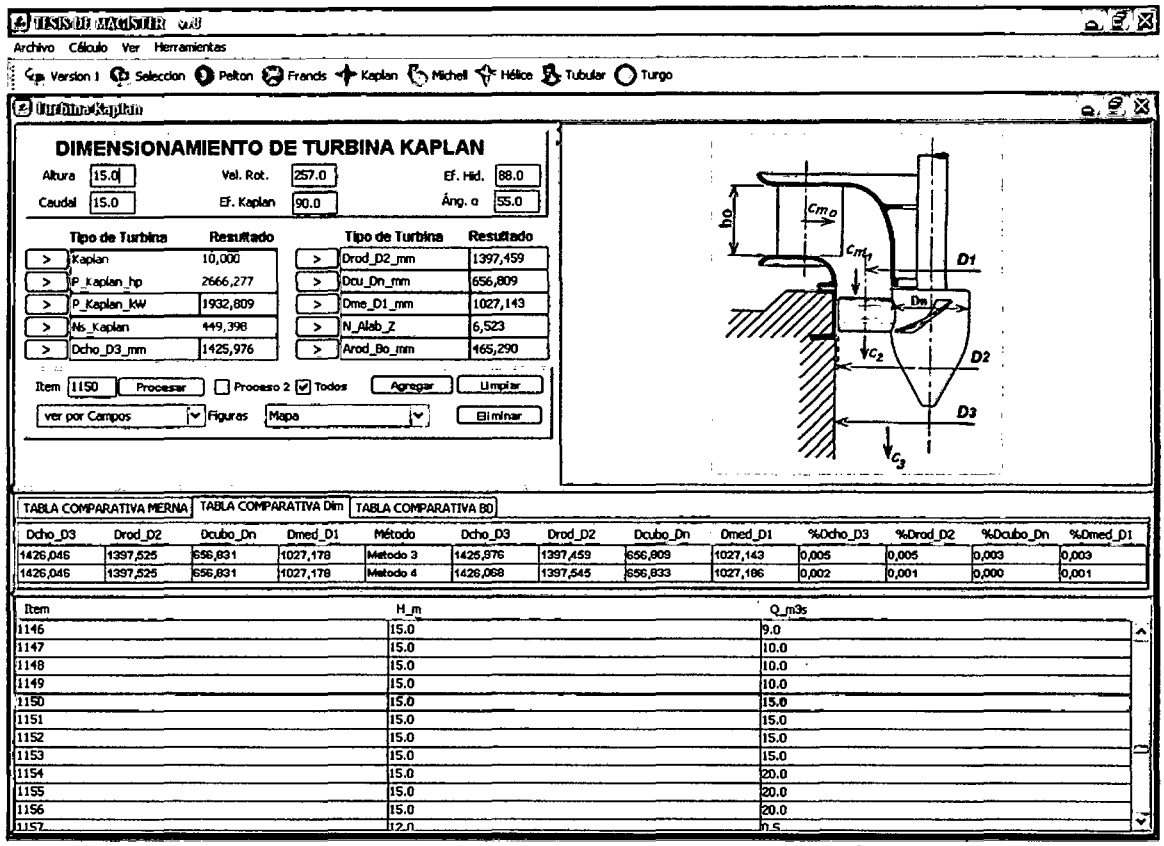


Fig. 5.23 Dimensiones de una turbina Kaplan. Método 3.

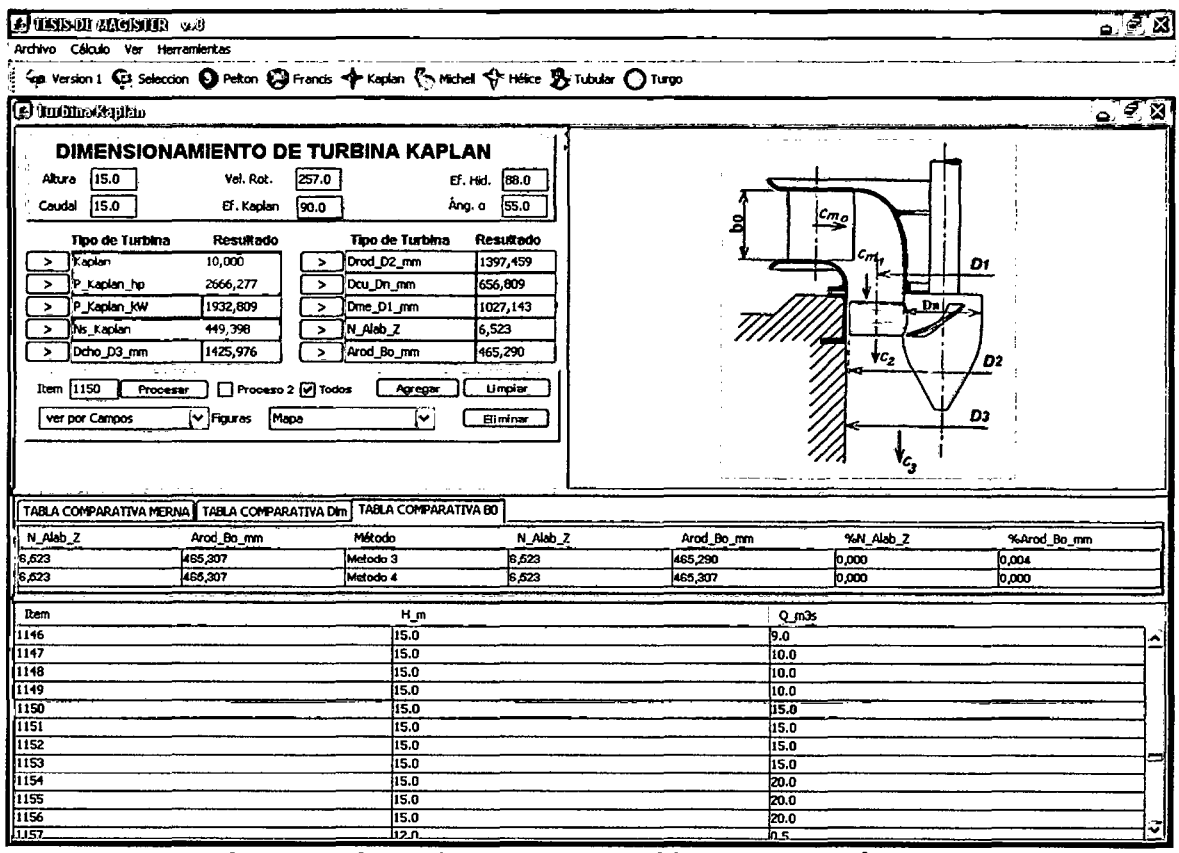


Fig. 5.24 Dimensiones de una turbina Kaplan. Método 4.

Tabla 5.21 Comparación de resultados.

	Kaplan				
	Tradicional	MERNA			
Variable		Método 1	Método 2	Método 3	Método 4
P hp	2664.474	2664.170	2664.071	2666.277	2664.382
P kw	1932.122	1932.006	1931.968	1932.809	1931.872
Ns	449.391	449.404	449.399	449.398	449.398

Tabla 5.22 Desviación de los resultados

	Kaplan			
	DESVIACIÓN (%)			
Variable	M1 - Tradicional	M2 - Tradicional	M3 - Tradicional	M4 - tradicional
P hp	0.011	0.015	0.068	0.003
P kw	0.006	0.008	0.036	0.013
Ns	0.003	0.002	0.001	0.002

Tabla 5.23 Comparación de resultados.

	Kaplan		
Variable	Tradicional	MERNA	
		Método 3	Método 4
D_{chorro} D_3	1426.046	1425.976	1426.068
D_{rod} D_2	1397.525	1397.459	1397.545
D_{cubo} D_n	656.831	656.809	656.833
D_{med} D_1	1027.178	1027.143	1027.186

Tabla 5.24 Desviación de los resultados.

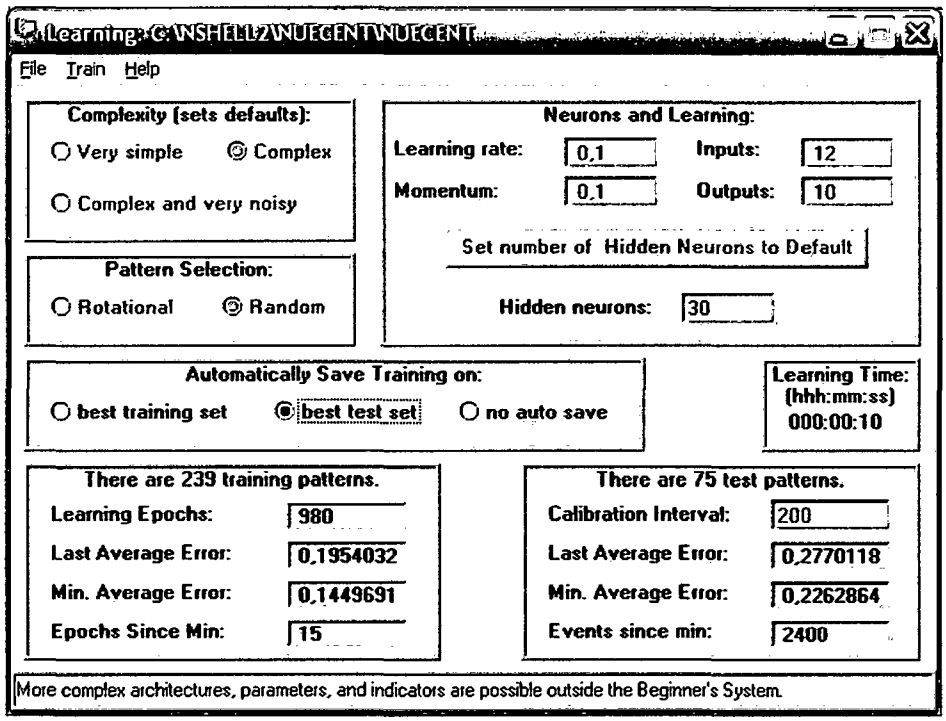
	Kaplan	
	DESVIACIÓN (%)	
Variable	Método 3 - Tradicional	Método 4 - tradicional
D_{chorro} D_3	0.005	0.002
D_{rod} D_2	0.005	0.001
D_{cubo} D_n	0.003	0.000
D_{med} D_1	0.003	0.001

En la tabla **5.23** mostramos una comparación en los resultados de las dimensiones obtenidas por el método tradicional y por el método **MERNA** (métodos 3 y 4). En la Tabla **5.24** se muestran la desviación de las dimensiones calculadas de la turbina Kaplan por los métodos 3 y 4.

RECOMENDACIONES

1.- Incrementar la base de cálculos reales, a fin de que la red neuronal realice estimaciones más confiables.

2.- Realizar el entrenamiento utilizando mayor número de neuronas, superiores a las 30 neuronas que se han utilizado, e incrementar el tiempo de entrenamiento de la red.



3.- El entrenamiento de la red debe seguir procesos específicos ajustados al problema en particular, lo que incluye examinar y determinar un factor clave como lo es la condición de parada adecuada a cada caso.

4.- Mejorar el diseño de la red neuronal y la definición de la topología radica en la habilidad que posteriormente tenga para clasificar patrones en la(s) capa(s) intermedia(s), donde reside la verdadera "cocina" donde los datos que ingresan en la capa de entrada se relacionan entre sí, de modo que las salidas numéricas resultantes en la capa de salida sean una representación de conocimiento adquirido por la Red Neuronal.

5.- Realizar afinamientos para que el rendimiento de la red neuronal dependa de los datos, arquitectura y tipo de aprendizaje. Por lo que se debe desarrollar heurísticamente el análisis para cada una de las etapas de diseño de la red.

REFERENCIAS BIBLIOGRÁFICAS

[1] **D. ACKLEY, G. HINTON, T. SEJNOWSKI.** "Un algoritmo de aprendizaje para la máquina de Boltzmann" Cognitive Science, vol.9, pp.147-169, 1985

[2] **L.B. ALMEIDA.** "Una regla de aprendizaje para perceptrón asincrónica con comentarios en un ambiente de combinatoria". IEEE 1st Int. Conf. on Neural Networks, vol.2, pp.609-618, San Diego, CA, 1987

[3] **L.B. ALMEIDA.** "Backpropagation en perceptrón con comentarios" *Neural Computers (R. Eckmiller, C. von dear Malsburg, eds) NATO ASI Ser., pp. 199-208, New York: Springer Verlag, 198.*

[4] **M. AYOUBI, R. ISERMANN.** "Modelo basado en la detección de fallas y diagnóstico con redes neuronales y la aplicación de un turbocompresor". *IFAC Artificial Intelligence in Real-Time Control, AIRTC'94, Valencia, Spain, 1994*

[5] **A.R. BARRON.** "Propiedades estadísticas de la redes neuronales artificiales" *Proc. of the 28th Conference on Decision and Control, pp.280-285, 1989*

[6] **A.R. BARRON**. "La regularización de la complejidad con la aplicación de redes neuronales artificiales" *Nonparametric Functional Estimation and Related Topics (G. Rousses, ed.), pp.561-576, 1991*

[7] **A.R. BARRON.** *"Las R*edes Neuronales" *Proc. of the Seventh Yale Workshop on Adaptive and Learning Systems, pp. 69-72. New Haven, CT: Yale University, 1992.*

[8] **D. BARSCHDORFF.** *"Comparación* entre las redes neuronales y la clásica Decisión de Algoritmos" *Fault Detection, Supervision and Safety for Technical Processes. IFAC Symposia Series, no.6, 1992*

[9] **E.B. BAUM, D. HAUSSLER**. *"T*amaño de red de generalización válida" *Neural Computation 1, pp. 151 - 160, 1989.*

[10] **BAREI, J.** *(1989). "Desarrollo de la Turbina tipo Banki para micro centrales por parte de EDELCA".* Departamento de Micro centrales de EDELCA. Caracas – Venezuela.

[11] ***BRACAMONTE, E.** (1996). "Diseño de un sistema de Energía Eléctrica para las Unidades Fronterizas de las Fuerzas Armadas Nacionales".* Trabajo de Investigación no publicado. Maracay - Venezuela.

[12] **DIMITRI P. BERTSEKAS.** "Notas sobre la programación no lineal y Discreta -Momento Control Óptimo" *MIT, CA, July 1979.*

[13] **M.BIANCHINI, P. FRASCONI**. "Aprender sin los mínimos locales en función de Redes de Base Radial" *IEEE Trans on Neural Networks, vol.6, n0.3, 1995*

[14] **S.A.BILLINGS, C.F.FUNG**. *"Recurrent Radial Basis Function Networks for Adaptive Noise Cancellation" Neural Networks, vol.8, no.2, pp.273-290, 1995*

[15] **A. BLUMER, A. EHRENFEUCHT, D. HAUSSLER, M.K.WARMUTH.** *"Learn ability and the Vapid - Chervonenkis Dimension" Journal of the Association for Computing Machinery 36, pp.929-965, 1989*

[16] **D.S. BROOMHEAD, D. LOWE.** *"Multivariate functional interpolation and adaptive networks Complex Systems 2, pp.321-355, 1988*

[17] **E.BURATTINI, G.TAMBURRINI.** *"A neural knowledge representation for diagnostic expert systems" Artificial Neural Networks, T. Kohonen, K. Makisara, O. Simula, J. Kangas (Editors) Elsevier Science Publisher B.V.(North Holland), 1991*

[18] **P. BURRASCANO.** *"Learning vector quantization for the probabilistic neural network" IEEE Trans. on Neural Networks, vol.2, pp.458-461, July 1991*

[19] **T.L. BURROWS, M.NIRANJAN**. "The use of feed-forward and recurrent neural networks for system identification CUED / F-INFENG / TR-158, December 1993

[20] **T. CACOULLOS.** *"Estimation of a multivariate density". Inst. Statist. Math., vol.18, no.2, pp.179-189, 1966*

[21] **G.A. CARPENTER, S. GROSSBERG** *"A massively parallel architecture for a self organizing neural pattern recognition machine" Computer Vision, Graphics and image Processing 37,pp. 54-115, 1987*

[22] **F. DE CUADRA GARCÍA** *[Cuadra, 1990. "El problema general de la optimización de diseño por ordenador: aplicación de técnicas de ingeniería de conocimiento" Tesis Doctoral. E.T.S. de Ingenieros Industriales. Universidad Pontificia Comillas. 1990.*

[23] **S. CHEN, S. BILLINGS.** *[Chen & Billings, 1989]* "Estimador Recursiva del error de predicción de parámetros para los modelos no lineales". *Int. J. Control 49 (2), pp. 569-594.*

[24] **S.CHEN, S.BILLINGS, P.GRANT** *[Chen et al., 1990] "Nonlinear system identification using neural networks". Int. J. Control, vol.51, no.6, pp.1191-1214*

[25] **S.CHEN, C.F.N.COWAN, P.M.GRANT** *[Chen et al., 1991]"Orthogonal least squares learning algorithm for Radial Basis" Function Networks IEEE Trans. on Neural Networks, vol.2, no.2, pp.302-309, 1991*

[26] **M. Chow, R. N. Sharpe, J. C. Hung.** *[Chow et al., 1993].* "Sobre la aplicación y diseño de redes neuronales artificiales para la detección de la falla de un motor" *IEEE Trans. on Industrial Electronics, vol.40, no.2, pp.181-196, April 1993.*

[27] **G. CYBENCO** *[Cybenko, 1989].* "Aproximación de superposición de una función sigmoidea Matemáticas de Control, Señales y Sistemas 2", *pp.303-314 Springer Verlag, New York Inc., 1989*

[28] **DAMITHA, K. RANAWEERA, G.KARADY** *[Damitha et al., 1993] "Power System Static Security Analysis using Radial Basis Function Neural Network"ESAP'93, pp.272-274*

[29] **GEORGE E.P. BOX, GWILYM M. JENKINS.** "Análisis de series temporales: la previsión y control" *Holden - Day Inc., CA, (Revised Edition) 1976.*

[30] **Quantz L.** (1992) *"Motores Hidráulicos" Construcción y cálculo de las instalaciones modernas de fuerza hidráulica.* L. Quantz. Editorial Gustavo Gili, SA

[31] **ITDG** Perú (1995) "Manual de Mini y Micro centrales Hidráulicas" Una guía para el desarrollo de proyectos. Intermediate Technology Development Group, ITDG-PERÜ

[32] **Juan Cevallos Ampuero [33]**, "Aplicación de Redes Neuronales para Optimizar Problemas Multi respuesta en mejora de la Calidad"

[33] **Gustavo Ovando, Mónica Bocco y Silvina Sayago [34]**, "Redes Neuronales para Modelar Predicción de Heladas"

[34] **Patricia González Serrano**, "Simulación Técnico- Económica Del Mercado Eléctrico Español"

[35] **Hernández López Leonor,** "Desarrollo de una metodología para la predicción y optimización de emisiones contaminantes y consumo en motores Diesel de automoción mediante redes neuronales artificiales"

[36] **Rodríguez R., Jorge Enrique,** "Redes Neuronales Artificiales para la clasificación de Imágenes Satelitales"

[37] **González García, Ismael,** "Control Neuronal De Un Generador De Inducción Para Generación Eólica"

[38] **Martínez Estudillo, F. J. y Hervás Martínez, C.** "Modelo no lineal basado en redes neuronales de unidades producto para clasificación. Una aplicación a la determinación del riesgo en tarjetas de crédito".

[39] **González Serrano, Patricia.** "Departamento De Sistemas Energéticos, Simulación Técnico- Económica Del Mercado Eléctrico Español"

[40] **Antonio Muñoz, San Roque.** "Aplicación De Técnicas De Redes Neuronales Artificiales Al Diagnóstico De Procesos Industriales"

[41] **MATAIX, C.,** (1975), Turbo máquinas hidráulicas: turbinas hidráulicas, bombas y ventiladores, ICAI.

[42] **Wilfredo Jara T.** (1998). Máquinas Hidráulicas" Fondo Editorial INIFIM. Instituto de Investigación de la Facultad de Ingeniería Mecánica.

[43] **ESA, A., HOLLMÉN, J., SIMULLA, O., VESANTO, J.,** (1999), Process Monitoring and Modeling using the Self-Organizing Map, *Integrated Computer-Aided Engineering, Vol. 6, Nro 1, pp. 3 – 14.*

[44] **TSUGUO NOZAKI**, "Guía para la Elaboración de Proyectos de Pequeñas Centrales Hidroeléctricas destinadas a la Electrificación Rural del Perú" Julio de 1968.

[45] **TSUGUO NOZAKI,** "Guía para la Elaboración de Estudios Preliminares de Factibilidad de Proyectos de Pequeñas Centrales de Mediana o Grande Capacidad en el Perú" Enero de 1969.

[46] **IIVARINEN, J., RAUHAMAA, J., VISA, A.,** (1999), "Unsupervised Segmentation of Surface Defects, *Workshop of Texture Analysis in Machine Vision", Oulu, Finland, June 14-15, pp. 53 – 58.*

[47] **KOHONEN, T.**, (1997), Self-Organizing Maps, Springer-Verlag

[48] **SARRATE LANA I., ALBRECHT K.:** La escuela del técnico Mecánico. Hidráulica. Motores hidráulicos y bombas. Editorial Labor, S. A. Barcelona. Madrid. 1951.

ANEXO A

NEUROSHELL 2

A.1 NEUROSHELL 2

NeuroShell 2 es un programa del software que imita la habilidad del cerebro humano para clasificar los modelos o tomar predicciones o decisiones basado en la experiencia del pasado. El cerebro humano confía en los estímulos neurales mientras la red neural usa los juegos de los datos. NeuroShell 2 le permite que construya problema personalizado sofisticado, que resuelve las aplicaciones sin programar.

NeuroShell 2 y el cerebro pueden resolver problemas que no pueden resolverse por el software de la computadora convencional escritos en un modo paso a paso. Simplemente como el cerebro, sin embargo, no se garantiza que las redes neurales dar siempre una respuesta completamente correcta, sobre todo si los modelos están de alguna manera incompletos o contradictorios.

A.2 INICIO EN EL NEUROSHELL 2.

Ingresar en el menú Todos los Programas y seleccionar NeuroShell 2, hacer clic en el icono "cerebro" de NeuroShell 2 que ha sido colocado cuando se instalo el programa; tal como se observa en la **figura A.1**. El menú Principal de

NeuroShell 2 será mostrado. Note que una barra de menú muy angosta esta sobre y desligada de la pantalla de aviso NeuroShell 2.

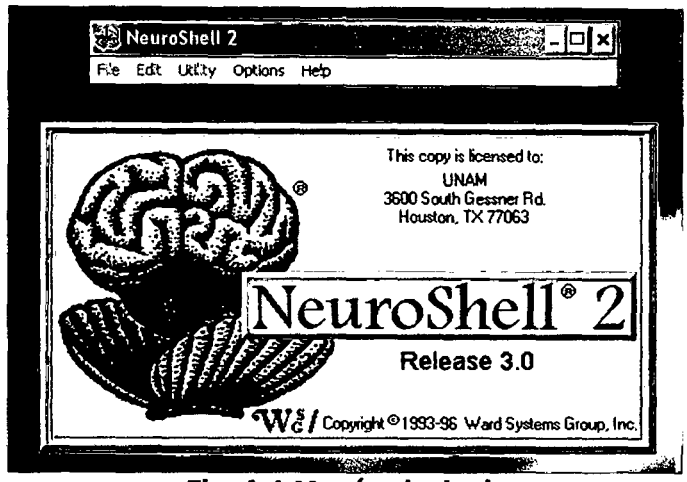


Fig. A.1 Menú principal.

A.3 DESIGNACIÓN DEL PROBLEMA.

Cada problema se refiere a una aplicación neuronal. Durante varias etapas del procesamiento del problema, NeuroShell 2 construirá diferentes archivos asociados con la aplicación. Los nombres de los todos los archivos empiezan con el mismo nombre del problema (máximo de 8 caracteres) y terminan con diferentes apropiadas extensiones.

A.3.1 MENÚ DE ARCHIVO (file):

Seleccione "file" (archivo) en la barra de Menú y aparece el menú mostrado en la **figura A.2**, en donde puede seleccionar crear un nuevo problema (New Problem) o abrir un problema existente (Open Problem).

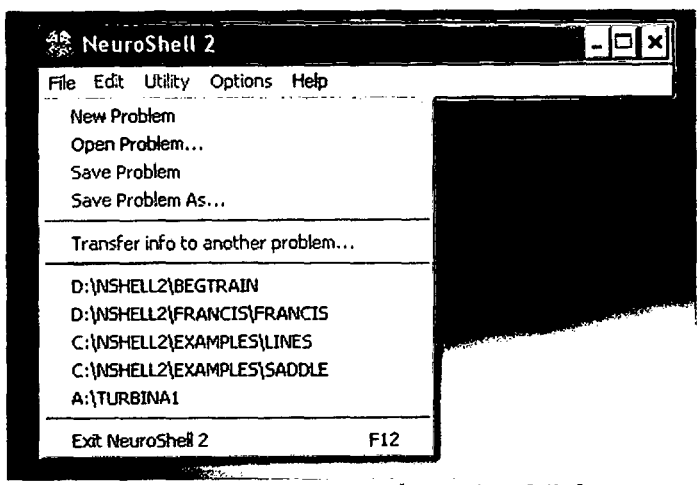


Fig. A.2 Barra de menú archivo (file).

A.3.1.1 Nuevo Problema.

Si se selecciona la opción ***New Problem***, aparece el menú mostrado en la **figura A.3**; este menú le permite empezar un nuevo problema en NeuroShell 2, escribiendo el nombre del problema o el nombre del archivo de descripción del problema. (DSC).

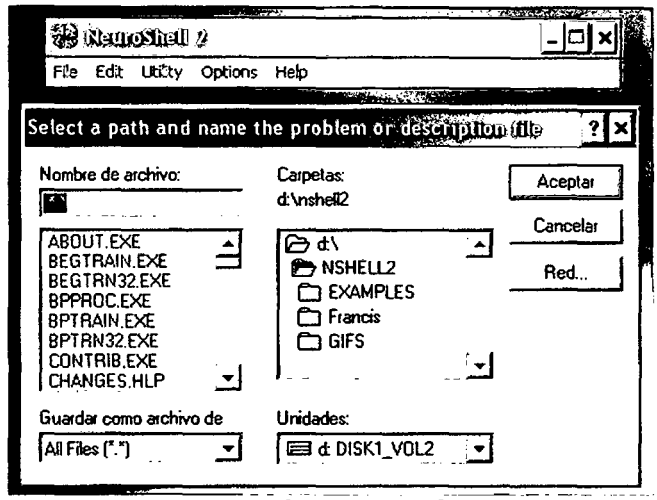


Fig. A.3 Barra de menú de la opción: New Problem.

Seleccionar el directorio ***examples***, luego ingresar el nombre del archivo a trabajar, por ejemplo Francis.dsc: Presionar Aceptar; tal como se observa en la **figura A.4**.

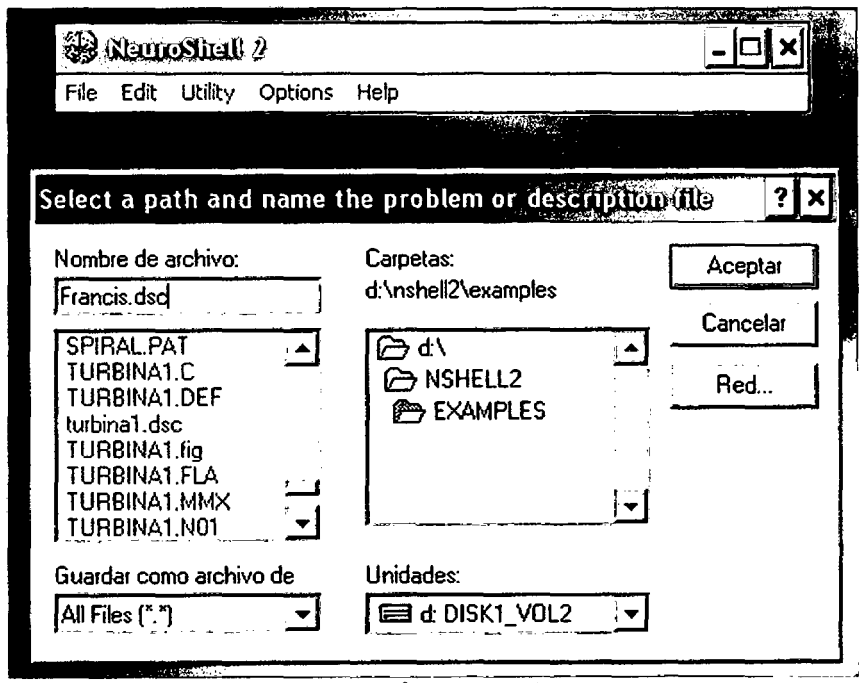


Fig. A.4 Barra de menú al seleccionar: examples.

A.3.1.2 Abrir Problema

Si selecciona la opción ***Open Problem***, aparece el menú mostrado en la **figura A.5**; le permite seleccionar un problema existente en NeuroShell 2, por la selección asociada al archivo de descripción previamente gravado.

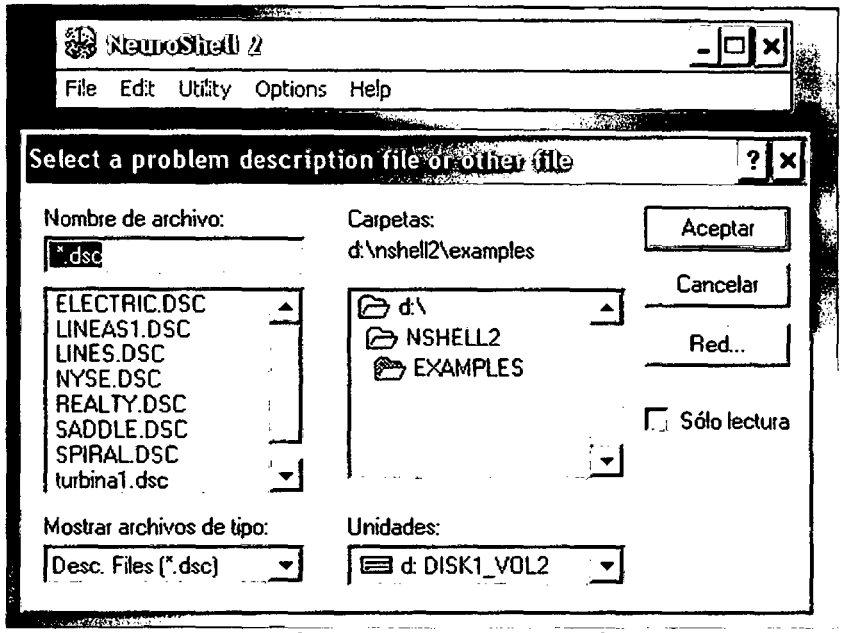


Fig. A.5 Barra de menú al seleccionar la opción: Open Problem.

Puede seleccionar marcando el archivo o tipeando su nombre, no es necesario tipiar su extensión; por ejemplo **turbina1**, tal como se observa en la **figura A.6**.

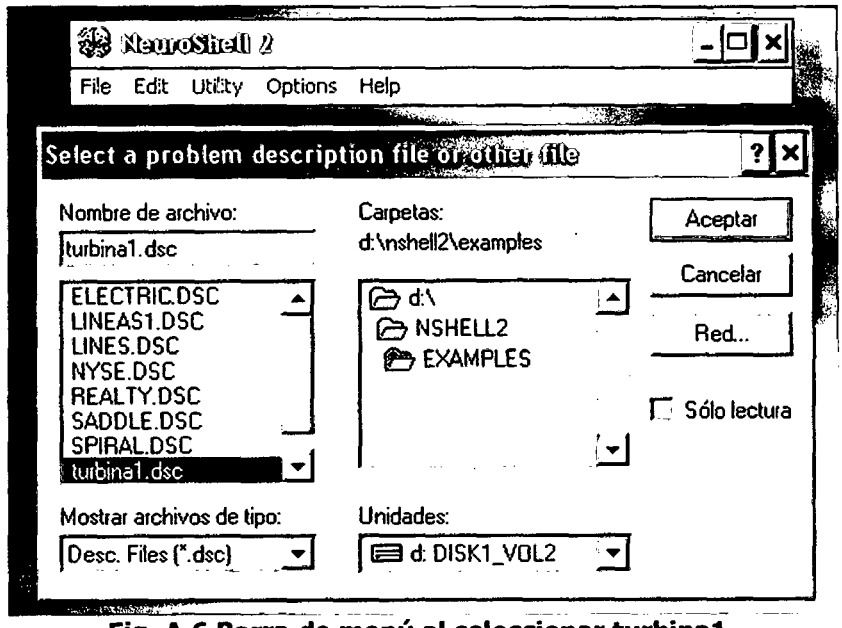


Fig. A.6 Barra de menú al seleccionar turbina1.

Al seleccionar el archivo ***turbina1***, presionar ***Aceptar***; y aparece el menú mostrado en la **figura A.7**. Puede añadir o cambiar la descripción del problema en cualquier momento que la caja de descripción del problema este mostrada.

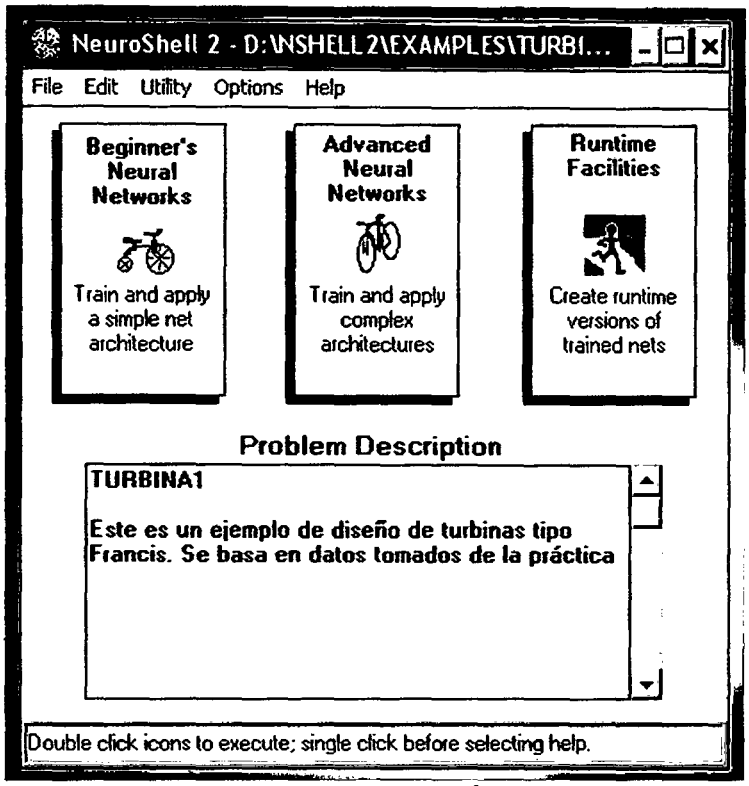


Fig. A.7 Ventana del menú principal.

A.3.2 VENTANA DEL MENÚ PRINCIPAL.

La ventana del Menú Principal brinda la opción de usar el NeuroShell 2 en tres diferentes maneras: ***el Sistema de Principiantes, el Sistema Avanzado, y el Sistema de Tiempo de Ejecución***; tal como se observa en la **figura A.7**. Estos tres sistemas son construidos de subprogramas llamados "módulos" y cada módulo es representado por un icono. La mayoría de usuarios deberán empezar primero con el Sistema de Principiantes.

Para usar un módulo, **doble clic del botón izquierdo del ratón sobre el icono** que representa ese módulo. Para recibir ayuda sobre cómo usar un módulo, ***clic simple sobre el icono*** y luego seleccione Contexto Actual (current context) desde el Menú de Ayuda (Help); tal como se observa en la **figura A.8**.

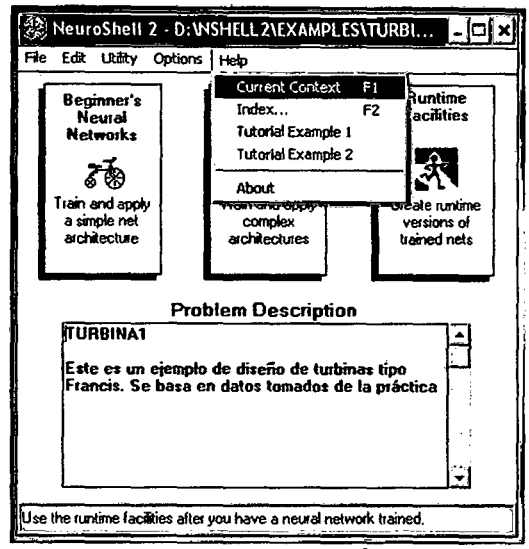


Fig. A.8 Ventana del menú de ayuda.

A.3.3 EL SISTEMA DE PRINCIPIANTES.

Para usar el módulo el sistema de principiantes, doble clic del botón izquierdo del ratón sobre el icono del ***"triciclo"*** que representa ese módulo y que se observa en la **figura A.7**; y aparece el menú mostrado en la **figura A.9**. Cuando un grupo de iconos aparece en la pantalla, el orden de operaciones a trabajar es desde la izquierda a la derecha. Si los iconos aparecen en una columna, trabaje desde arriba hacia abajo. Note que puede no tener que usar todos los iconos que aparecen en la pantalla de forma de crear una aplicación red neuronal. Mucho de los módulos son opcionales dependiendo sobre el tipo de aplicación que este creando.

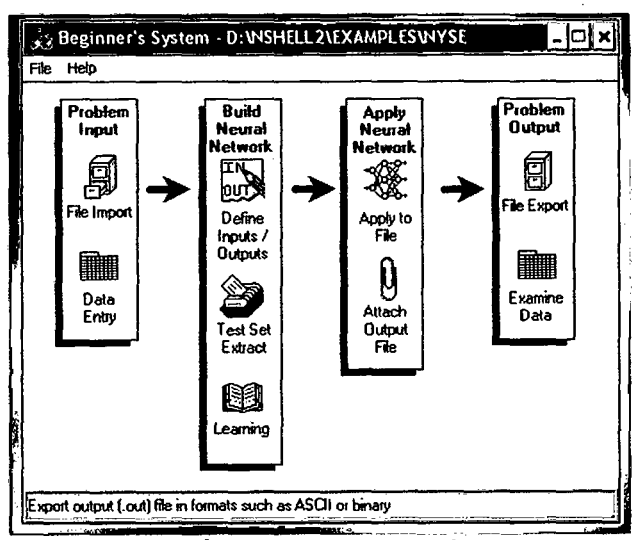


Fig. A.9 Menú del sistema de principiantes.

A.3.4 EL SISTEMA DE AVANZADO.

Para usar el módulo el sistema de avanzado, doble clic del botón izquierdo del ratón sobre el icono de la ***"bicicleta"*** que representa ese módulo y que se observa en la **figura A.7**, y aparece el menú mostrado en la **figura A.10**. Cuando un grupo de iconos aparece en la pantalla, el orden de operaciones a trabajar es desde la izquierda a la derecha. Si los iconos aparecen en una columna, trabaje desde arriba hacia abajo.

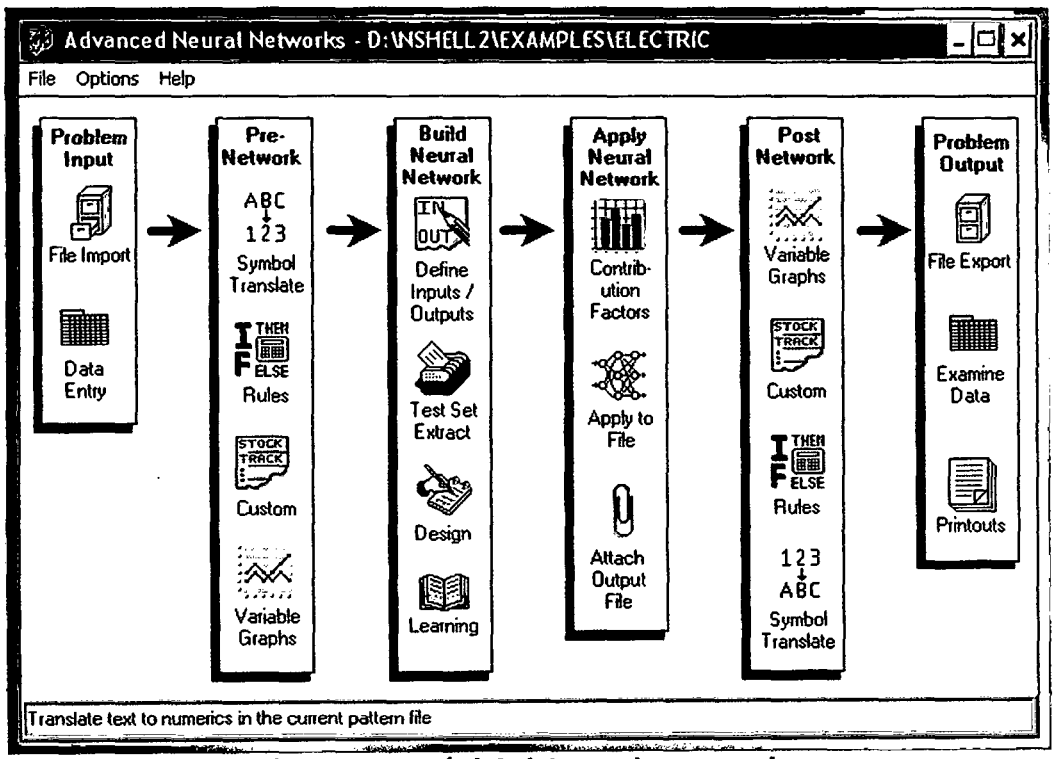


Fig. A.10 Menú del sistema de avanzado.

A.3.5 SISTEMA DE TIEMPO DE EJECUCIÓN.

Para usar el módulo el sistema de tiempo de ejecución, doble clic del botón izquierdo del ratón sobre el icono del ***"atleta"*** que representa ese módulo y que se observa en la **figura A.7**, y aparece el menú mostrado en la **figura A.11**. Cuando un grupo de iconos aparece en la pantalla, el orden de operaciones a trabajar es desde la izquierda a la derecha. Si los iconos aparecen en una columna, trabaje desde arriba hacia abajo.

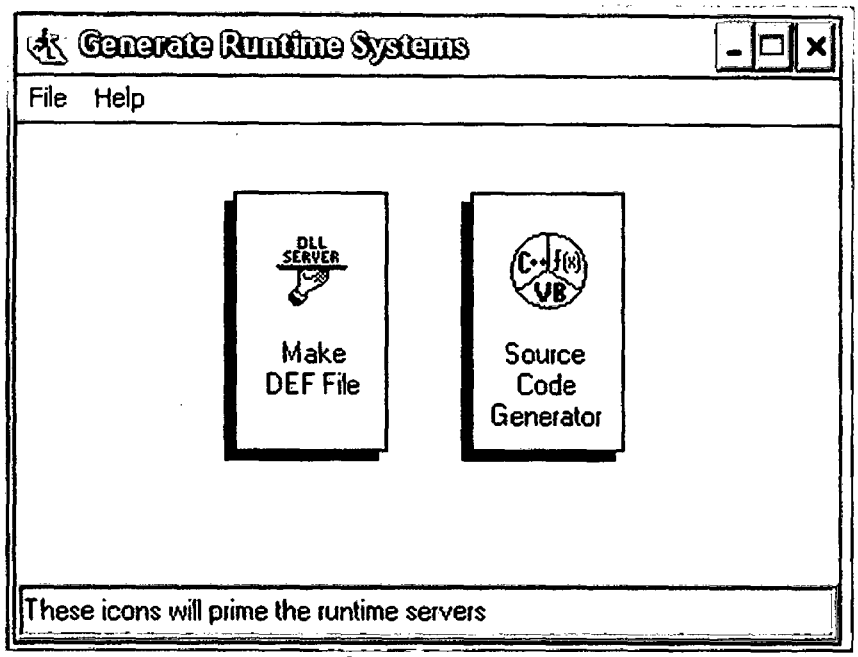


Fig. A.11 Menú del sistema de avanzado.

Si seleccionamos haciendo doble clic del botón izquierdo del ratón en la opción "hacer el archivo en def" (Make DEF File) aparece el menú mostrado en la **figura A.12**; doble clic del botón izquierdo del ratón en la opción "generador del código fuente" (Source code generador) aparece el menú mostrado en la **figura A.13**.

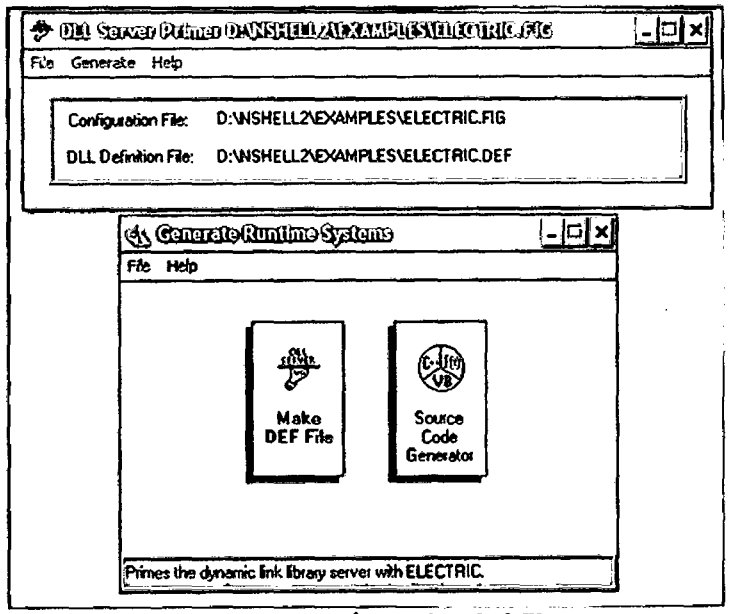


Fig. A.12 Opción Make def file.

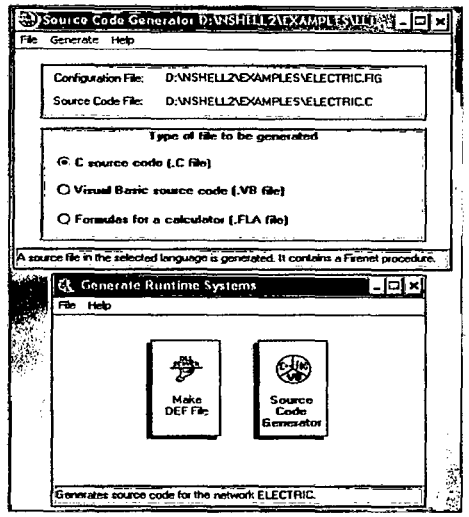


Fig. A.13 Opción Source code Generator.

A.4 Opciones del Menú Principal

NeuroShell 2 ofrece varias opciones del menú principal las cuales permite trabajar con archivos dentro de NeuroShell 2 y con programas externos.

A.4.1 Menú de Archivo (File)

Esta opción permite nuevo, abrir, gravar, gravar como y transferir información a otro problema, dentro de la caja de edición de texto que muestra la descripción del problema, (File mostrado en la **figura A.14**).

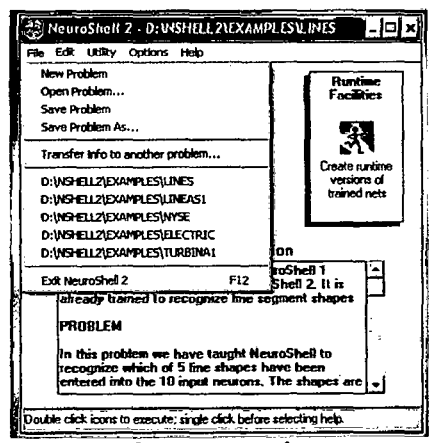


Fig. A.14 Opción del menú principal: File.

1. **Nuevo Problema**: Permite empezar un nuevo problema en NeuroShell 2 por el tipeo del nombre del problema ó los archivos de descripción de los problemas (.DSC).

2. Abre Problema: Permite seleccionar un problema existente por la selección del archivo de descripción asociado. Puede seleccionar el archivo ó tipea sus nombres. Si tipeo esto, la extensión DSC no es requerida.

3. Grabar Problema: Graba el archivo de descripción para el problema actual bajo el mismo nombre que entro cuando creo el problema.

4. Grabar Problema Como: Graba el archivo de descripción para el problema actual bajo un nuevo nombre entrado. Tendrá la oportunidad de copiar todos los otros archivos en el problema existente dando a ellos el nuevo nombre.

5. Transfiere información a otro problema: La opción transfiere permite seleccionar alguno ó todos los archivos desde el actual problema de NeuroShell 2 y copia ellos a otro problema. Esta opción es útil cuando desea crear una aplicación similar al problema actual y desea hacer uso de los archivos existentes, pero con otro nombre de problema. Cuando selecciona la opción Transfiere, una pantalla mostrará una lista de los archivos disponibles de NeuroShell 2 para el problema existente.

A.4.2 Menú de Edición (Edit)

Esta opción permite cortar, copiar, y pegar datos dentro del la caja de edición de texto que muestra la descripción del problema; (edit mostrado en la **figura A.15**).

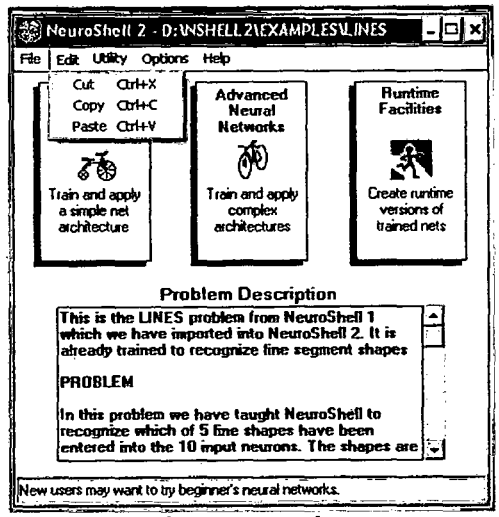


Fig. A.15 Opción del menú principal: Edit.

A.4.3 Menú de Utilidad (Utility)

Esta opción permite ejecutar un programa externo, editar un archivo externo con Excel, editar un archivo de texto (opción mostrada en la **figura A.15**).

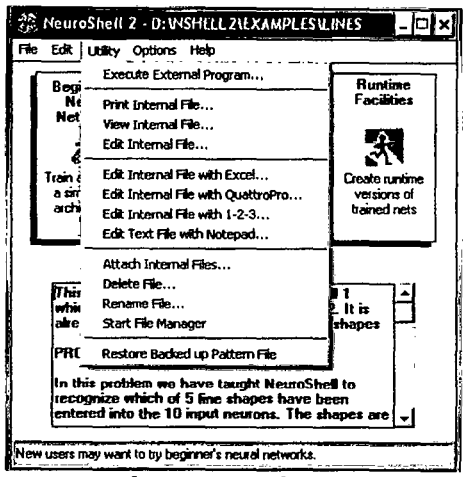


Fig. A.15 Opción del menú principal: Utility.

1. **Ejecuta Programa Externo**: Si se desea usar un programa externo que genere data para el uso con el NeuroShell 2, por ejemplo. Esta opción "shell out" a otro programa que se escoja: Imprimir, ver, ó editar los archivos internos de NeuroShell 2. Estos 2 módulos para hacer la impresión, visión ó edición.

2. **Edita Archivo Interno con Excel, Quattro Pro, ó 1-2-3:** Seleccionado estas opciones empezara la ejecución del programa seleccionado y carga del archivo NeuroShell 2 dentro de ese programa. Los archivos de patrones de NeuroShell 2 (y algunos archivos internos) están en formato de hoja de cálculo que puede ser cargado por cualquier programa de hoja de cálculo convencional.

3. **Editar un Archivo de Texto con Notepad**: Este es una rápida manera de visualizar un archivo de texto. Use esto para ver archivos ASCII si desea importar ó tiene para exportar archivos ASCII por ejemplo.

4. **Añadir Archivos Internos**: Use esta opción para combinar ó "engomar" dos archivos juntos. Puede desear usar esta opción para crear un simple archivo que contenga los patrones de la rede de entrenamiento (.archivo PAT) y la predicciones o clasificaciones de la red (archivo OUT).Esta opción puede también ser usada para añadir nuevos patrones al conjunto de datos de entrenamiento.

5. **Elimina Archivo**: Use esta opción para borrar un archivo.

6. **Renombrar Archivo**: Use esta opción para cambiar el nombre de un archivo.

7. **Administrador de Inicio de Archivos** Use esta opción para ejecutar el Administrador de Archivo del Windows.

8. **Restaura desde Archivo de Patrones de Respaldo**: Antes de que 'los módulos NeuroShell 2 Traslado de Símbolos y reglas de proceso un archivo de respaldo copia es hecho con la extensión OLD Esta opción restaure el antiguo archivo OLD que el caso que no quiera el proceso que resulte de Traslado de Símbolo ó Reglas

A.4.4 Menú de Opciones (Options):

Esta opción permite minimizar sobre un nuevo módulo, colocar color de fondo, seleccionar su propia hoja de cálculo (opción mostrada en la **figura A.16**).

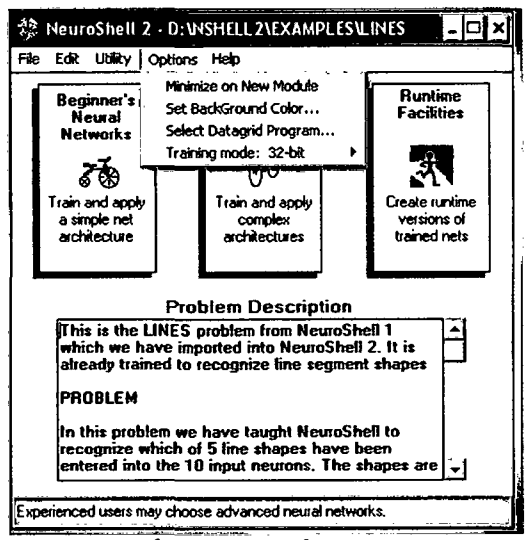


Fig. A.16 Opción del menú principal: Options.

1. **Minimiza sobre Nuevo Modulo**: esta opción automáticamente minimizará las ventanas existentes cuando haga doble clic sobre un icono para abrir un nuevo módulo.

2. **Pone Color de Fondo**: Esta opción permite seleccionar un diferente color de fondo.

3. **Selecciona Datagrid**: Esta opción permite seleccionar su propia hoja de cálculo.

ANEXO B

CÓDIGO GENERADOR POR NEUROSHELL 2

B.1 PROGRAMA C GENERADO POR NEUROSHELL2

B.1.1 SELECCIÓN DE TURBINAS: MÉTODO 1.

```
/* Insert this code into your C program to fire the C:\NSHELL2\EXAMPLES\TURBINA\SELTURB\SELTURB
network */
/* This code is designed to be simple and fast for porting to any machine */
/* Therefore all code and weights are inline without looping or data storage */
/*  which might be harder to port between compilers. */

#include <math.h>
void Fire_SELTURB(double *inarray, double *outarray)
{
 double netsum;
 double feature2[78];

/* inarray [0] is H_m */
/* inarray [1] is Q_m3_seg */
/* inarray [2] is Vel_Rot.n_rpm */
/* inarray [3] is Ef_Pelton */
/* inarray [4] is Ef_Francis */
/* inarray [5] is Ef_kaplan */
/* inarray [6] is Ef_Helice */
/* inarray [7] is Ef_Michell */
/* outarray [0] is P_Pelton_hp */
/* outarray [1] is P_Pelton_kW */
/* outarray [2] is P_Francis_hp */
/* outarray [3] is P_Francis_kW */
/* outarray [4] is P_Kaplan_hp */
/* outarray [5] is P_Kaplan_kW */
/* outarray [6] is P_Hélice_hp */
/* outarray [7] is P_Helice_kw */
/* outarray [8] is P_Michel_hp */
/* outarray [9] is P_Michel_kW */
/* outarray [10] is Ns_Pelton_1Ch */
/* outarray [11] is Ns_Pelton_2Ch */
/* outarray [12] is Ns_Pelton_4Ch */
/* outarray [13] is Ns_Francis */
/* outarray [14] is Ns_Kaplan */
/* outarray [15] is Ns_Helice */
/* outarray [16] is Ns_Michell */
/* outarray [17] is Pelton_1ch */
/* outarray [18] is Pelton_2ch */
/* outarray [19] is Pelton_4ch */
/* outarray [20] is FLenta */
/* outarray [21] is FNormal */
/* outarray [22] is FRapida */
/* outarray [23] is FExRapida */
/* outarray [24] is KAPLAN */
/* outarray [25] is HELICE */
/* outarray [26] is Michell */
```

```
if (inarray[0]< 3) inarray[0] =  3;
if (inarray[0]> 300) inarray[0] =  300;
inarray[0] = (inarray[0] -  3) / 297;

if (inarray[1]< .03) inarray[1] =  .03;
if (inarray[1]> 20) inarray[1] =  20;
inarray[1] = (inarray[1] -  .03) / 19.97;

if (inarray[2]< 225) inarray[2] =  225;
if (inarray[2]> 2500) inarray[2] =  2500;
inarray[2] = (inarray[2] -  225) / 2275;

if (inarray[3]< 70) inarray[3] =  70;
if (inarray[3]> 71) inarray[3] =  71;
inarray[3] = (inarray[3] -  70);

if (inarray[4]< 80) inarray[4] =  80;
if (inarray[4]> 81) inarray[4] =  81;
inarray[4] = (inarray[4] -  80);

if (inarray[5]< 80) inarray[5] =  80;
if (inarray[5]> 81) inarray[5] =  81;
inarray[5] = (inarray[5] -  80);

if (inarray[6]< 85) inarray[6] =  85;
if (inarray[6]> 86) inarray[6] =  86;
inarray[6] = (inarray[6] -  85);

if (inarray[7]< 65) inarray[7] =  65;
if (inarray[7]> 66) inarray[7] =  66;
inarray[7] = (inarray[7] -  65);

netsum =  4.012454;
netsum += inarray[0] * -6.006174;
netsum += inarray[1] * -43.81947;
netsum += inarray[2] * -.1946824;
netsum += inarray[3] *  .0976806;
netsum += inarray[4] *  .237138;
netsum += inarray[5] *  .2687429;
netsum += inarray[6] *  .2696768;
netsum += inarray[7] * -.1725181;
feature2[0] = 1 / (1 + exp(-netsum));

netsum = -4.58809;
netsum += inarray[0] *  1.996068;
netsum += inarray[1] *  124.5476;
netsum += inarray[2] *  3.064328;
netsum += inarray[3] * -.2312784;
netsum += inarray[4] *  .1737816;
netsum += inarray[5] * -.2647328;
netsum += inarray[6] *  .225602;
netsum += inarray[7] * -.1068545;
feature2[1] = 1 / (1 + exp(-netsum));

netsum =  6.691545;
netsum += inarray[0] *  1.241722;
netsum += inarray[1] *  1.124249;
netsum += inarray[2] *  1.409892;
netsum += inarray[3] * -4.477066E-03;
netsum += inarray[4] *  .1462783;
netsum += inarray[5] * -.1350169;
netsum += inarray[6] * -6.497696E-02;
netsum += inarray[7] * -9.201331E-03;
feature2[2] = 1 / (1 + exp(-netsum));

netsum =  6.803522;
netsum += inarray[0] *  .4219669;
netsum += inarray[1] * -.1689109;
netsum += inarray[2] *  2.105579;
netsum += inarray[3] * -7.252114E-02;
netsum += inarray[4] *  6.913358E-02;
netsum += inarray[5] * -.2128941;
netsum += inarray[6] *  .2924742;
netsum += inarray[7] * -.2945067;
feature2[3] = 1 / (1 + exp(-netsum));

netsum =  6.753985;
netsum += inarray[0] *  1.885692;
netsum += inarray[1] *  1.842216;
netsum += inarray[2] *  1.223961;
netsum += inarray[3] * -1.913511E-03;
netsum += inarray[4] * -8.332469E-02;
netsum += inarray[5] *  .2890683;
netsum += inarray[6] * -.188815;
netsum += inarray[7] * -3.608204E-02;
feature2[4] = 1 / (1 + exp(-netsum));

netsum =  12.53918;
netsum += inarray[0] * -10.48725;
netsum += inarray[1] * -11.54977;
netsum += inarray[2] *  1.307204;
netsum += inarray[3] *  .1760155;
netsum += inarray[4] *  .2139927;
netsum += inarray[5] *  7.199012E-02;
netsum += inarray[6] *  .237431;
netsum += inarray[7] *  4.71511E-03;
feature2[5] = 1 / (1 + exp(-netsum));

netsum += feature2[70] * -.1061427;
netsum += feature2[71] * -5.131286E-02;
netsum += feature2[72] * -.1983506;
netsum += feature2[73] * -.1868874;
netsum += feature2[74] * -.4420827;
netsum += feature2[75] *  7.672975E-03;
netsum += feature2[76] * -.3880841;
netsum += feature2[77] *  .0171786;
outarray[13] = 1 / (1 + exp(-netsum));

netsum = -.1555729;
netsum += feature2[0] * -.2053683;
netsum += feature2[1] *  .2608078;
netsum += feature2[2] *  .2242703;
netsum += feature2[3] *  .4026024;
```

```
if (inarray[0]< 3) inarray[0] =  3;
if (inarray[0]> 300) inarray[0] =  300;
inarray[0] = (inarray[0] -  3) / 297;

if (inarray[1]< .03) inarray[1] =  .03;
if (inarray[1]> 20) inarray[1] =  20;
inarray[1] = (inarray[1] -  .03) / 19.97;

if (inarray[2]< 225) inarray[2] =  225;
if (inarray[2]> 2500) inarray[2] =  2500;
inarray[2] = (inarray[2] -  225) / 2275;

if (inarray[3]< 70) inarray[3] =  70;
if (inarray[3]> 71) inarray[3] =  71;
inarray[3] = (inarray[3] -  70);

if (inarray[4]< 80) inarray[4] =  80;
if (inarray[4]> 81) inarray[4] =  81;
inarray[4] = (inarray[4] -  80);

if (inarray[5]< 80) inarray[5] =  80;
if (inarray[5]> 81) inarray[5] =  81;
inarray[5] = (inarray[5] -  80);

if (inarray[6]< 85) inarray[6] =  85;
if (inarray[6]> 86) inarray[6] =  86;
inarray[6] = (inarray[6] -  85);

if (inarray[7]< 65) inarray[7] =  65;
if (inarray[7]> 66) inarray[7] =  66;
inarray[7] = (inarray[7] -  65);

netsum =  4.012454;
netsum += inarray[0] * -6.006174;
netsum += inarray[1] * -43.81947;
netsum += inarray[2] * -.1946824;
netsum += inarray[3] *  .0976806;
netsum += inarray[4] *  .237138;
netsum += inarray[5] *  .2687429;
netsum += inarray[6] *  .2696768;
netsum += inarray[7] * -.1725181;
feature2[0] = 1 / (1 + exp(-netsum));

netsum = -4.58809;
netsum += inarray[0] *  1.996068;
netsum += inarray[1] *  124.5476;
netsum += inarray[2] *  3.064328;
netsum += inarray[3] * -.2312784;
netsum += inarray[4] *  .1737816;
netsum += inarray[5] * -.2647328;
netsum += inarray[6] *  .225602;
netsum += inarray[7] * -.1068545;
feature2[1] = 1 / (1 + exp(-netsum));

netsum =  6.691545;
netsum += inarray[0] *  1.241722;
netsum += inarray[1] *  1.124249;
netsum += inarray[2] *  1.409892;
netsum += inarray[3] * -4.477066E-03;
netsum += inarray[4] *  .1462783;
netsum += inarray[5] * -.1350169;
netsum += inarray[6] * -6.497696E-02;
netsum += inarray[7] * -9.201331E-03;
feature2[2] = 1 / (1 + exp(-netsum));

netsum =  6.803522;
netsum += inarray[0] *  .4219669;
netsum += inarray[1] * -.1689109;
netsum += inarray[2] *  2.105579;
netsum += inarray[3] * -7.252114E-02;
netsum += inarray[4] *  6.913358E-02;
netsum += inarray[5] * -.2128941;
netsum += inarray[6] *  .2924742;
netsum += inarray[7] * -.2945067;
feature2[3] = 1 / (1 + exp(-netsum));

netsum =  6.753985;
netsum += inarray[0] *  1.885692;
netsum += inarray[1] *  1.842216;
netsum += inarray[2] *  1.223961;
netsum += inarray[3] * -1.913511E-03;
netsum += inarray[4] * -8.332469E-02;
netsum += inarray[5] *  .2890683;
netsum += inarray[6] * -.188815;
netsum += inarray[7] * -3.608204E-02;
feature2[4] = 1 / (1 + exp(-netsum));

netsum =  12.53918;
netsum += inarray[0] * -10.48725;
netsum += inarray[1] * -11.54977;
netsum += inarray[2] *  1.307204;
netsum += inarray[3] *  .1760155;
netsum += inarray[4] *  .2139927;
netsum += inarray[5] *  7.199012E-02;
netsum += inarray[6] *  .237431;
netsum += inarray[7] *  4.71511E-03;
feature2[5] = 1 / (1 + exp(-netsum));

netsum += feature2[70] * -.1061427;
netsum += feature2[71] * -5.131286E-02;
netsum += feature2[72] * -.1983506;
netsum += feature2[73] * -.1868874;
netsum += feature2[74] * -.4420827;
netsum += feature2[75] *  7.672975E-03;
netsum += feature2[76] * -.3880841;
netsum += feature2[77] *  .0171786;
outarray[13] = 1 / (1 + exp(-netsum));

netsum = -.1555729;
netsum += feature2[0] * -.2053683;
netsum += feature2[1] *  .2608078;
netsum += feature2[2] *  .2242703;
netsum += feature2[3] *  .4026024;
```

```
netsum += feature2[4] * -.1994831;
netsum += feature2[5] * -.0486236;
netsum += feature2[6] * .2696616;
netsum += feature2[7] * -.2318429;
netsum += feature2[8] * .2153651;
netsum += feature2[9] * -.2137569;
netsum += feature2[10] * .1337688;
netsum += feature2[11] * .5015489;
netsum += feature2[12] * -.2311162;
netsum += feature2[13] * -1.231483;
netsum += feature2[14] * .0879225;
netsum += feature2[15] * .3611635;
netsum += feature2[16] * .1297449;
netsum += feature2[17] * .2806726;
netsum += feature2[18] * .2323795;
netsum += feature2[19] * .4311188;
netsum += feature2[20] * .3078803;
netsum += feature2[21] * .1831245;
netsum += feature2[22] * .4989658;
netsum += feature2[23] * -.3631028;
netsum += feature2[24] * -3.055629E-02;
netsum += feature2[25] * 6.689677E-02;
netsum += feature2[26] * -.5139582;
netsum += feature2[27] * .1299174;
netsum += feature2[28] * .1252134;
netsum += feature2[29] * .3670477;
netsum += feature2[30] * -.4100025;
netsum += feature2[31] * -.4820666;
netsum += feature2[32] * 3.317787E-02;
netsum += feature2[33] * -.3644724;
netsum += feature2[34] * -.0832167;
netsum += feature2[35] * -2.446911;
netsum += feature2[36] * -.1232537;
netsum += feature2[37] * .1542605;
netsum += feature2[38] * -.2874018;
netsum += feature2[39] * -.1492803;
netsum += feature2[40] * -.0681086;
netsum += feature2[41] * -.3346819;
netsum += feature2[42] * -1.317804E-03;
netsum += feature2[43] * .1386337;
netsum += feature2[44] * .4574402;
netsum += feature2[45] * -6.674092E-02;
netsum += feature2[46] * -.2284631;
netsum += feature2[47] * -5.233391E-02;
netsum += feature2[48] * .0721999;
netsum += feature2[49] * -6.423534E-02;
netsum += feature2[50] * -.3508618;
netsum += feature2[51] * .1724207;
netsum += feature2[52] * -.2348868;
netsum += feature2[53] * 8.358204E-02;
netsum += feature2[54] * -.2854364;
netsum += feature2[55] * -.1852949;
netsum += feature2[56] * -9.729192E-02;
netsum += feature2[57] * 3.33165;
netsum += feature2[58] * -4.549921;
netsum += feature2[59] * .475064;
netsum += feature2[60] * -.1053541;
netsum += feature2[61] * 4.834981E-02;
netsum += feature2[62] * 3.131916;
netsum += feature2[63] * .1450535;
netsum += feature2[64] * 4.637285E-03;
netsum += feature2[65] * 3.128135E-03;
netsum += feature2[66] * .2952096;
netsum += feature2[67] * 1.010071;
netsum += feature2[68] * .1034905;
netsum += feature2[69] * .314106;
netsum += feature2[70] * -.1386881;
netsum += feature2[71] * -.1696001;
netsum += feature2[72] * -.1984208;
netsum += feature2[73] * -1.750844E-02;
netsum += feature2[74] * .1416057;
netsum += feature2[75] * 6.660113E-02;
netsum += feature2[76] * -.2768271;
netsum += feature2[77] * -2.748005E-02;
outarray[14] = 1 / (1 + exp(-netsum));

netsum = -5.550731E-02;
netsum += feature2[0] * -4.636625;
netsum += feature2[1] * -6.37167;
netsum += feature2[2] * -.1827714;
netsum += feature2[3] * .1026804;
netsum += feature2[4] * -.9555315;
netsum += feature2[5] * -.4341631;
netsum += feature2[6] * -1.234118;
netsum += feature2[7] * .3693337;
netsum += feature2[8] * -.8373896;
netsum += feature2[9] * .6586234;
netsum += feature2[10] * -9.694033E-02;
netsum += feature2[11] * 1.551679;
netsum += feature2[12] * .1391298;
netsum += feature2[13] * 1.045563;
netsum += feature2[14] * .2442916;
netsum += feature2[15] * -.6620581;
netsum += feature2[16] * .1611816;
netsum += feature2[17] * .8191512;
netsum += feature2[18] * 6.292721;
netsum += feature2[19] * 1.32685;
netsum += feature2[20] * -.805654;
netsum += feature2[21] * -.3656001;
netsum += feature2[22] * 2.119214E-03;
netsum += feature2[23] * .5620463;
netsum += feature2[24] * .2066217;
netsum += feature2[25] * .2378432;
netsum += feature2[26] * .5564146;
netsum += feature2[27] * .4734852;
netsum += feature2[28] * -.8577573;
netsum += feature2[29] * 7.188172;
netsum += feature2[30] * .689226;
netsum += feature2[31] * 3.624709E-02;
netsum += feature2[32] * -2.749696E-02;
netsum += feature2[33] * .2428257;
netsum += feature2[34] * 2.180475E-02;
netsum += feature2[35] * -2.921005;
```

```
netsum += feature2[36] * .1493925;
netsum += feature2[37] * -3.669127E-02;
netsum += feature2[38] * -.6756941;
netsum += feature2[39] * .3548143;
netsum += feature2[40] * -.5055785;
netsum += feature2[41] * .9162754;
netsum += feature2[42] * -.1983868;
netsum += feature2[43] * .3392236;
netsum += feature2[44] * 2.872041E-02;
netsum += feature2[45] * -6.022549E-02;
netsum += feature2[46] * -.9783021;
netsum += feature2[47] * .4780843;
netsum += feature2[48] * -.3050171;
netsum += feature2[49] * 8.886951E-02;
netsum += feature2[50] * .7122769;
netsum += feature2[51] * -.9858402;
netsum += feature2[52] * -.5086016;
netsum += feature2[53] * 4.510923;
netsum += feature2[54] * .5051236;
netsum += feature2[55] * .9749169;
netsum += feature2[56] * -.7045599;
netsum += feature2[57] * -2.779886;
netsum += feature2[58] * 2.155971;
netsum += feature2[59] * -1.1749;
netsum += feature2[60] * .204671;
netsum += feature2[61] * .3816415;
netsum += feature2[62] * 2.812952;
netsum += feature2[63] * .2510594;
netsum += feature2[64] * -3.901409;
netsum += feature2[65] * -2.839547;
netsum += feature2[66] * -.9092724;
netsum += feature2[67] * 1.929407;
netsum += feature2[68] * -.8316443;
netsum += feature2[69] * -.3394692;
netsum += feature2[70] * .2312422;
netsum += feature2[71] * .2223419;
netsum += feature2[72] * 2.239259;
netsum += feature2[73] * -.2155613;
netsum += feature2[74] * .4760993;
netsum += feature2[75] * 6.282055E-04;
netsum += feature2[76] * -1.235251;
netsum += feature2[77] * 2.217366E-02;
outarray[26] = 1 / (1 + exp(-netsum));


outarray[0] = 4794.079 * (outarray[0] - .1) /
.8 + 41.44737;
if (outarray[0]< 41.44737) outarray[0] =
41.44737;
if (outarray[0]> 4835.526) outarray[0] =
4835.526;

outarray[1] = 3476.389 * (outarray[1] - .1) /
.8 + 30.05523;
if (outarray[1]< 30.05523) outarray[1] =
30.05523;
if (outarray[1]> 3506.444) outarray[1] =
3506.444;

outarray[2] = 5478.947 * (outarray[2] - .1) /
.8 + 47.36842;
if (outarray[2]< 47.36842) outarray[2] =
47.36842;
if (outarray[2]> 5526.316) outarray[2] =
5526.316;

outarray[3] = 3973.016 * (outarray[3] - .1) /
.8 + 34.34884;
if (outarray[3]< 34.34884) outarray[3] =
34.34884;
if (outarray[3]> 4007.365) outarray[3] =
4007.365;

outarray[4] = 5478.947 * (outarray[4] - .1) /
.8 + 47.36842;
if (outarray[4]< 47.36842) outarray[4] =
47.36842;
if (outarray[4]> 5526.316) outarray[4] =
5526.316;

outarray[5] = 3973.016 * (outarray[5] - .1) /
.8 + 34.34884;
if (outarray[5]< 34.34884) outarray[5] =
34.34884;
if (outarray[5]> 4007.365) outarray[5] =
4007.365;

outarray[6] = 5821.381 * (outarray[6] - .1) /
.8 + 50.32895;
if (outarray[6]< 50.32895) outarray[6] =
50.32895;
if (outarray[6]> 5871.71) outarray[6] =
5871.71;

outarray[7] = 4221.33 * (outarray[7] - .1) / .8
+ 36.49564;
if (outarray[7]< 36.49564) outarray[7] =
36.49564;
if (outarray[7]> 4257.825) outarray[7] =
4257.825;

outarray[8] = 4451.644 * (outarray[8] - .1) /
.8 + 38.48684;
if (outarray[8]< 38.48684) outarray[8] =
38.48684;
if (outarray[8]> 4490.131) outarray[8] =
4490.131;

outarray[9] = 3228.075 * (outarray[9] - .1) /
.8 + 27.90843;
if (outarray[9]< 27.90843) outarray[9] =
27.90843;
```

```
if (outarray[9]> 3255.984) outarray[9] =
3255.984;

outarray[10] = 1886.564 * (outarray[10] - .1)
/ .8 + 8.020156;
if (outarray[10]< 8.020156) outarray[10] =
8.020156;
if (outarray[10]> 1894.584) outarray[10] =
1894.584;

outarray[11] = 1334.002 * (outarray[11] - .1)
/ .8 + 5.671106;
if (outarray[11]< 5.671106) outarray[11] =
5.671106;
if (outarray[11]> 1339.673) outarray[11] =
1339.673;

outarray[12] = 943.2818 * (outarray[12] - .1)
/ .8 + 4.010078;
if (outarray[12]< 4.010078) outarray[12] =
4.010078;
if (outarray[12]> 947.2919) outarray[12] =
947.2919;

outarray[13] = 2016.821 * (outarray[13] - .1)
/ .8 + 8.573907;
if (outarray[13]< 8.573907) outarray[13] =
8.573907;
if (outarray[13]> 2025.395) outarray[13] =
2025.395;

outarray[14] = 2016.821 * (outarray[14] - .1)
/ .8 + 8.573907;
if (outarray[14]< 8.573907) outarray[14] =
8.573907;
if (outarray[14]> 2025.395) outarray[14] =
2025.395;

outarray[15] = 2078.891 * (outarray[15] - .1)
/ .8 + 8.837781;
if (outarray[15]< 8.837781) outarray[15] =
8.837781;
if (outarray[15]> 2087.729) outarray[15] =
2087.729;

outarray[16] = 1817.939 * (outarray[16] - .1)
/ .8 + 7.728415;
if (outarray[16]< 7.728415) outarray[16] =
7.728415;
if (outarray[16]> 1825.667) outarray[16] =
1825.667;

outarray[17] = 10 * (outarray[17] - .1) / .8 ;
if (outarray[17]< 0) outarray[17] = 0;
if (outarray[17]> 10) outarray[17] = 10;

outarray[18] = 10 * (outarray[18] - .1) / .8 ;
if (outarray[18]< 0) outarray[18] = 0;
if (outarray[18]> 10) outarray[18] = 10;

outarray[19] = 10 * (outarray[19] - .1) / .8 ;
if (outarray[19]< 0) outarray[19] = 0;
if (outarray[19]> 10) outarray[19] = 10;

outarray[20] = 10 * (outarray[20] - .1) / .8 ;
if (outarray[20]< 0) outarray[20] = 0;
if (outarray[20]> 10) outarray[20] = 10;

outarray[21] = 10 * (outarray[21] - .1) / .8 ;
if (outarray[21]< 0) outarray[21] = 0;
if (outarray[21]> 10) outarray[21] = 10;

outarray[22] = 10 * (outarray[22] - .1) / .8 ;
if (outarray[22]< 0) outarray[22] = 0;
if (outarray[22]> 10) outarray[22] = 10;

outarray[23] = 10 * (outarray[23] - .1) / .8 ;
if (outarray[23]< 0) outarray[23] = 0;
if (outarray[23]> 10) outarray[23] = 10;

outarray[24] = 10 * (outarray[24] - .1) / .8 ;
if (outarray[24]< 0) outarray[24] = 0;
if (outarray[24]> 10) outarray[24] = 10;

outarray[25] = 10 * (outarray[25] - .1) / .8 ;
if (outarray[25]< 0) outarray[25] = 0;
if (outarray[25]> 10) outarray[25] = 10;

outarray[26] = 10 * (outarray[26] - .1) / .8 ;
if (outarray[26]< 0) outarray[26] = 0;
if (outarray[26]> 10) outarray[26] = 10;

 }

}
```

B.1.2 SELECCIÓN DE TURBINAS: MÉTODO 2.

```
/* Insert this code into your C program to fire the C:\NSHELL2\TURBINA2\SELTURB\SELTURB network */
/* This code is designed to be simple and fast for porting to any machine */
/* Therefore all code and weights are inline without looping or data storage */
/*  which might be harder to port between compilers. */

#include <math.h>
```

```
void Fire_SELTURB(double *inarray, double *outarray)
{
 double netsum;
 double feature2[17];
 double feature3[17];
 double feature4[17];

/* inarray[0] is H_m */
/* inarray[1] is Q_m3_seg */
/* inarray[2] is Vel_Rot.n_rpm */
/* inarray[3] is Ef_Pelton */
/* inarray[4] is Ef_Francis */
/* inarray[5] is Ef_kaplan */
/* inarray[6] is Ef_Helice */
/* inarray[7] is Ef_Michell */
/* outarray[0] is P_Pelton_hp */
/* outarray[1] is P_Pelton_kW */
/* outarray[2] is P_Francis_hp */
/* outarray[3] is P_Francis_kW */
/* outarray[4] is P_Kaplan_hp */
/* outarray[5] is P_Kaplan_kW */
/* outarray[6] is P_Hélice_hp */
/* outarray[7] is P_Helice_kw */
/* outarray[8] is P_Michel_hp */
/* outarray[9] is P_Michel_kW */

/* outarray[10] is Ns_Pelton_1Ch */
/* outarray[11] is Ns_Pelton_2Ch */
/* outarray[12] is Ns_Pelton_4Ch */
/* outarray[13] is Ns_Francis */
/* outarray[14] is Ns_Kaplan */
/* outarray[15] is Ns_Helice */
/* outarray[16] is Ns_Michell */
/* outarray[17] is Pelton_1ch */
/* outarray[18] is Pelton_2ch */
/* outarray[19] is Pelton_4ch */
/* outarray[20] is FLenta */
/* outarray[21] is FNormal */
/* outarray[22] is FRapida */
/* outarray[23] is FExRapida */
/* outarray[24] is KAPLAN */
/* outarray[25] is HELICE */
/* outarray[26] is Michell */

if (inarray[0]< 3) inarray[0] =  3;
if (inarray[0]> 300) inarray[0] =  300;
inarray[0] =  2 * (inarray[0] -  3) / 297 -1;

if (inarray[1]< .03) inarray[1] =  .03;
if (inarray[1]> 25) inarray[1] =  25;
inarray[1] =  2 * (inarray[1] -  .03) / 24.97 -1;

if (inarray[2]< 225) inarray[2] =  225;
if (inarray[2]> 2500) inarray[2] =  2500;
inarray[2] =  2 * (inarray[2] -  225) / 2275 -1;

if (inarray[3]< 70) inarray[3] =  70;
if (inarray[3]> 71) inarray[3] =  71;
inarray[3] =  2 * (inarray[3] -  70) -1;

if (inarray[4]< 80) inarray[4] =  80;
if (inarray[4]> 81) inarray[4] =  81;
inarray[4] =  2 * (inarray[4] -  80) -1;

if (inarray[5]< 90) inarray[5] =  90;
if (inarray[5]> 91) inarray[5] =  91;
inarray[5] =  2 * (inarray[5] -  90) -1;

if (inarray[6]< 85) inarray[6] =  85;
if (inarray[6]> 86) inarray[6] =  86;
inarray[6] =  2 * (inarray[6] -  85) -1;

if (inarray[7]< 65) inarray[7] =  65;
if (inarray[7]> 66) inarray[7] =  66;
inarray[7] =  2 * (inarray[7] -  65) -1;

netsum = -9.278794;
netsum += inarray[0] *  2.130964;
netsum += inarray[1] * -55.63792;
netsum += inarray[2] * -.9635499;
netsum += inarray[3] *  9.032338;
netsum += inarray[4] *  8.896361;
netsum += inarray[5] *  8.764627;
netsum += inarray[6] *  8.981965;
netsum += inarray[7] *  9.249156;
feature2[0] = exp(-netsum * netsum);

netsum = -.3356453;
netsum += inarray[0] * -13.81011;
netsum += inarray[1] *  6.950732;
netsum += inarray[2] *  5.568862;
netsum += inarray[3] *  .488485;
netsum += inarray[4] *  .1647717;
netsum += inarray[5] *  .6168813;
netsum += inarray[6] *  .5130173;
netsum += inarray[7] *  .6266463;
feature2[1] = exp(-netsum * netsum);

netsum =  9.219331;
netsum += inarray[0] * -3.278037;
netsum += inarray[1] *  58.98454;
netsum += inarray[2] *  .2089272;
netsum += inarray[3] * -9.200826;
netsum += inarray[4] * -9.568002;
netsum += inarray[5] * -9.287702;
netsum += inarray[6] * -9.459937;
```

```
netsum += inarray[7] * -9.373599;
feature2[2] = exp(-netsum * netsum);

netsum =  .1123975;
netsum += inarray[0] * -6.405803E-06;
netsum += inarray[1] * -2.665343E-06;
netsum += inarray[2] *  8.860962E-06;
netsum += inarray[3] *  .2087619;
netsum += inarray[4] *  2.793559E-02;
netsum += inarray[5] * -4.532864E-02;
netsum += inarray[6] * -5.941014E-02;
netsum += inarray[7] * -1.956329E-02;
feature2[3] = exp(-netsum * netsum);

netsum = -.526773;
netsum += inarray[0] * -10.89583;
netsum += inarray[1] *  2.047482;
netsum += inarray[2] *  4.889545;
netsum += inarray[3] *  .5687325;
netsum += inarray[4] *  .7594013;
netsum += inarray[5] *  .5337099;
netsum += inarray[6] *  .6190051;
netsum += inarray[7] *  .5608106;
feature2[4] = exp(-netsum * netsum);

netsum = -4.651912;
netsum += inarray[0] * -38.8163;
netsum += inarray[1] *  4.058908;
netsum += inarray[2] *  7.97945;
netsum += inarray[3] *  4.463066;
netsum += inarray[4] *  4.650092;
netsum += inarray[5] *  4.382041;
netsum += inarray[6] *  4.749206;
netsum += inarray[7] *  4.564811;
feature2[5] = exp(-netsum * netsum);

netsum = -8.031471;
netsum += inarray[0] * -50.96907;
netsum += inarray[1] *  1.349757;
netsum += inarray[2] *  .9734847;
netsum += inarray[3] *  8.087894;
netsum += inarray[4] *  8.236326;
netsum += inarray[5] *  8.2223;
netsum += inarray[6] *  8.0952;
netsum += inarray[7] *  8.15027;
feature2[6] = exp(-netsum * netsum);

netsum = -2.913704;
netsum += inarray[0] * -33.56334;
netsum += inarray[1] *  10.05718;
netsum += inarray[2] *  4.7988;
netsum += inarray[3] *  3.132398;
netsum += inarray[4] *  3.302194;
netsum += inarray[5] *  3.261202;
netsum += inarray[6] *  3.28441;
netsum += inarray[7] *  3.041435;
feature2[7] = exp(-netsum * netsum);

netsum =  3.438163;
netsum += inarray[0] *  5.135822;
netsum += inarray[1] *  17.4249;
netsum += inarray[2] *  1.359593;
netsum += inarray[3] * -3.269688;
netsum += inarray[4] * -3.337133;
netsum += inarray[5] * -3.452521;
netsum += inarray[6] * -3.259282;
netsum += inarray[7] * -3.270801;
feature2[8] = exp(-netsum * netsum);

netsum =  .1490578;
netsum += inarray[0] *  .2590984;
netsum += inarray[1] *  8.684754E-02;
netsum += inarray[2] *  5.173325E-03;
netsum += inarray[3] *  .2684744;
netsum += inarray[4] * -.1996794;
netsum += inarray[5] *  .2253343;
netsum += inarray[6] * -.2272567;
netsum += inarray[7] * -.1379637;
feature2[9] = exp(-netsum * netsum);

netsum = -.1414636;
netsum += inarray[0] * -10.45879;
netsum += inarray[1] *  9.712525;
netsum += inarray[2] *  1.198444;
netsum += inarray[3] *  .0925781;
netsum += inarray[4] *  4.029226E-02;
netsum += inarray[5] *  .1389309;
netsum += inarray[6] * -.1334005;
netsum += inarray[7] *  .2357203;
feature2[10] = exp(-netsum * netsum);

netsum = -.9075661;
netsum += inarray[0] *  6.72969;
netsum += inarray[1] * -10.79017;
netsum += inarray[2] * -3.983727;
netsum += inarray[3] *  .8878998;
netsum += inarray[4] *  1.118063;
netsum += inarray[5] *  1.172194;
netsum += inarray[6] *  1.323785;
netsum += inarray[7] *  1.001575;
feature2[11] = exp(-netsum * netsum);

netsum = -2.070767;
netsum += inarray[0] *  4.011036;
netsum += inarray[1] * -16.54174;
netsum += inarray[2] * -1.735263;
netsum += inarray[3] *  2.256442;
netsum += inarray[4] *  1.935186;
netsum += inarray[5] *  2.011906;
netsum += inarray[6] *  1.918106;
netsum += inarray[7] *  2.340504;
feature2[12] = exp(-netsum * netsum);
```

```
netsum = -9.889485E-02;
netsum += inarray[0] * -13.41075;
netsum += inarray[1] *  10.21087;
netsum += inarray[2] *  2.568665;
netsum += inarray[3] *  3.812202E-02;
netsum += inarray[4] *  .2268677;
netsum += inarray[5] * -8.617245E-02;
netsum += inarray[6] *  .11723;
netsum += inarray[7] *  .1473739;
feature2[13] = exp(-netsum * netsum);

netsum =  7.889425;
netsum += inarray[0] * -1.025794;
netsum += inarray[1] *  51.23505;
netsum += inarray[2] *  2.417069;
netsum += inarray[3] * -7.866017;
netsum += inarray[4] * -8.004015;
netsum += inarray[5] * -7.816721;
netsum += inarray[6] * -7.909219;
netsum += inarray[7] * -8.186836;
feature2[14] = exp(-netsum * netsum);

netsum = -1.587871;
netsum += inarray[0] * -10.33541;
netsum += inarray[1] * -1.428747;
netsum += inarray[2] *  1.660502;
netsum += inarray[3] *  1.362235;
netsum += inarray[4] *  1.08542;
netsum += inarray[5] *  1.282021;
netsum += inarray[6] *  1.252591;
netsum += inarray[7] *  1.453643;
feature2[15] = exp(-netsum * netsum);

netsum = -.193417;
netsum += inarray[0] *  2.251587E-03;
netsum += inarray[1] * -1.502392E-03;
netsum += inarray[2] * -8.304471E-03;
netsum += inarray[3] * -8.312319E-03;
netsum += inarray[4] *  .2180088;
netsum += inarray[5] * -.2585248;
netsum += inarray[6] * -.2890495;
netsum += inarray[7] *  .151924;
feature2[16] = exp(-netsum * netsum);

netsum = -.5446295;
netsum += inarray[0] *  .6374514;
netsum += inarray[1] *  .6920365;
netsum += inarray[2] *  .3369457;
netsum += inarray[3] *  .739545;
netsum += inarray[4] *  .4213045;
netsum += inarray[5] *  .4851378;
netsum += inarray[6] *  .4581105;
netsum += inarray[7] *  .6767899;
feature3[0] = tanh(netsum);

netsum =  1.193857;
netsum += inarray[0] *  1.641722;
netsum += inarray[1] *  8.064013;
netsum += inarray[2] * -2.175391E-02;
netsum += inarray[3] * -1.160581;
netsum += inarray[4] * -1.152956;
netsum += inarray[5] * -.8810059;
netsum += inarray[6] * -.756901;
netsum += inarray[7] * -.8113878;
feature3[1] = tanh(netsum);

netsum = -.6663667;
netsum += inarray[0] *  .4558362;
netsum += inarray[1] *  .1165736;
netsum += inarray[2] *  .3560349;
netsum += inarray[3] *  .7350841;
netsum += inarray[4] *  .8810424;
netsum += inarray[5] *  .8241688;
netsum += inarray[6] *  .685376;
netsum += inarray[7] *  .3365965;
feature3[2] = tanh(netsum);

netsum = -.8741248;
netsum += inarray[0] * -12.54293;
netsum += inarray[1] * -.9860775;
netsum += inarray[2] *  4.973096;
netsum += inarray[3] *  1.054954;
netsum += inarray[4] *  .6266273;
netsum += inarray[5] *  .7790931;
netsum += inarray[6] *  .9061326;
netsum += inarray[7] *  .5114728;
feature3[3] = tanh(netsum);

netsum =  .43859;
netsum += inarray[0] * -.2645646;
netsum += inarray[1] * -.7133281;
netsum += inarray[2] * -.3505348;
netsum += inarray[3] * -.8587748;
netsum += inarray[4] * -.8542516;
netsum += inarray[5] * -.5260545;
netsum += inarray[6] * -.370869;
netsum += inarray[7] * -.6366007;
feature3[4] = tanh(netsum);

netsum = -.1514026;
netsum += inarray[0] * -5.266235;
netsum += inarray[1] *  5.89672;
netsum += inarray[2] *  .7242751;
netsum += inarray[3] * -4.399207E-02;
netsum += inarray[4] *  2.897635E-02;
netsum += inarray[5] * -.4002639;
netsum += inarray[6] *  2.682449E-02;
netsum += inarray[7] * -.022238;
feature3[5] = tanh(netsum);

netsum = -3.995313;
netsum += inarray[0] * -3.0327;
netsum += inarray[1] * -22.62037;
```

```
netsum += inarray[2] *  2.595698;
netsum += inarray[3] *  4.176915;
netsum += inarray[4] *  3.853969;
netsum += inarray[5] *  4.041949;
netsum += inarray[6] *  4.285523;
netsum += inarray[7] *  4.246883;
feature3[6] = tanh(netsum);

netsum =  .5269387;
netsum += inarray[0] * -9.609886E-02;
netsum += inarray[1] * -.6702701;
netsum += inarray[2] * -.3882791;
netsum += inarray[3] * -.7878019;
netsum += inarray[4] * -.8641025;
netsum += inarray[5] * -.7362946;
netsum += inarray[6] * -.3472958;
netsum += inarray[7] * -.7998279;
feature3[7] = tanh(netsum);

netsum =  .9592593;
netsum += inarray[0] *  .6862443;
netsum += inarray[1] *  9.207538;
netsum += inarray[2] * -.1581033;
netsum += inarray[3] * -1.504784;
netsum += inarray[4] * -1.542509;
netsum += inarray[5] * -1.291986;
netsum += inarray[6] * -1.00055;
netsum += inarray[7] * -1.462338;
feature3[8] = tanh(netsum);

netsum =  .4774549;
netsum += inarray[0] *  4.897983;
netsum += inarray[1] *  4.841372;
netsum += inarray[2] * -.3125938;
netsum += inarray[3] * -.8938413;
netsum += inarray[4] * -.8496583;
netsum += inarray[5] * -.5408561;
netsum += inarray[6] * -.7457334;
netsum += inarray[7] * -1.029791;
feature3[9] = tanh(netsum);

netsum =  7.919343;
netsum += inarray[0] *  9.217402;
netsum += inarray[1] *  41.68781;
netsum += inarray[2] *  1.930244;
netsum += inarray[3] * -7.840027;
netsum += inarray[4] * -7.859685;
netsum += inarray[5] * -7.954023;
netsum += inarray[6] * -7.981675;
netsum += inarray[7] * -8.037846;
feature3[10] = tanh(netsum);

netsum =  .7268276;
netsum += inarray[0] * -.426102;
netsum += inarray[1] * -.584038;
netsum += inarray[2] * -.1223907;
netsum += inarray[3] * -.3087343;
netsum += inarray[4] * -.6061106;
netsum += inarray[5] * -.7596001;
netsum += inarray[6] * -.7474791;
netsum += inarray[7] * -.8009327;
feature3[11] = tanh(netsum);

netsum = -1.634588;
netsum += inarray[0] * -17.37486;
netsum += inarray[1] *  3.547589;
netsum += inarray[2] *  2.052678;
netsum += inarray[3] *  1.930658;
netsum += inarray[4] *  1.845392;
netsum += inarray[5] *  1.936152;
netsum += inarray[6] *  1.809057;
netsum += inarray[7] *  2.183632;
feature3[12] = tanh(netsum);

netsum = -9.631815E-02;
netsum += inarray[0] *  .3820803;
netsum += inarray[1] * -3.812609;
netsum += inarray[2] * -5.872171;
netsum += inarray[3] * -3.980106E-02;
netsum += inarray[4] * -1.262406E-02;
netsum += inarray[5] *  .1630763;
netsum += inarray[6] *  .231212;
netsum += inarray[7] *  .126016;
feature3[13] = tanh(netsum);

netsum =  1.853561;
netsum += inarray[0] *  5.926773;
netsum += inarray[1] *  11.27109;
netsum += inarray[2] * -1.580979;
netsum += inarray[3] * -1.549403;
netsum += inarray[4] * -1.962914;
netsum += inarray[5] * -1.550027;
netsum += inarray[6] * -1.760502;
netsum += inarray[7] * -1.985259;
feature3[14] = tanh(netsum);

netsum =  3.581756;
netsum += inarray[0] * -.7929824;
netsum += inarray[1] *  21.23032;
netsum += inarray[2] *  4.738927E-02;
netsum += inarray[3] * -3.675248;
netsum += inarray[4] * -3.487715;
netsum += inarray[5] * -3.29634;
netsum += inarray[6] * -3.712868;
netsum += inarray[7] * -3.266392;
feature3[15] = tanh(netsum);

netsum = -2.656711E-02;
netsum += inarray[0] *  .2916313;
netsum += inarray[1] *  1.117193;
netsum += inarray[2] *  .3886688;
netsum += inarray[3] *  .2945068;
netsum += inarray[4] *  .3311472;
```

```
netsum += inarray[5] *  .421938;
netsum += inarray[6] * -8.800162E-02;
netsum += inarray[7] *  .2940512;
feature3[16] = tanh(netsum);

netsum =  .3890214;
netsum += inarray[0] *  3.036166;
netsum += inarray[1] *  .9331115;
netsum += inarray[2] * -.209792;
netsum += inarray[3] * -.2165886;
netsum += inarray[4] * -.4282739;
netsum += inarray[5] * -.1086271;
netsum += inarray[6] * -.1493222;
netsum += inarray[7] * -.4112692;
feature4[0] = 1 - exp(-netsum * netsum);

netsum =  5.366681;
netsum += inarray[0] *  40.16782;
netsum += inarray[1] * -.0494918;
netsum += inarray[2] * -5.302598;
netsum += inarray[3] * -5.653204;
netsum += inarray[4] * -5.613995;
netsum += inarray[5] * -5.096655;
netsum += inarray[6] * -5.572027;
netsum += inarray[7] * -5.56393;
feature4[1] = 1 - exp(-netsum * netsum);

netsum =  .7021881;
netsum += inarray[0] *  1.341746;
netsum += inarray[1] * -.6468292;
netsum += inarray[2] *  4.117766;
netsum += inarray[3] * -.9128084;
netsum += inarray[4] * -1.002733;
netsum += inarray[5] * -.7523251;
netsum += inarray[6] * -1.101477;
netsum += inarray[7] * -1.07348;
feature4[2] = 1 - exp(-netsum * netsum);

netsum =  1.481583;
netsum += inarray[0] * -1.357019;
netsum += inarray[1] *  6.29689;
netsum += inarray[2] *  .3162687;
netsum += inarray[3] * -.9382887;
netsum += inarray[4] * -1.1741;
netsum += inarray[5] * -1.140302;
netsum += inarray[6] * -1.148869;
netsum += inarray[7] * -1.447849;
feature4[3] = 1 - exp(-netsum * netsum);

netsum = -2.897003;
netsum += inarray[0] *  2.300158;
netsum += inarray[1] * -19.50001;
netsum += inarray[2] * -.4796322;
netsum += inarray[3] *  3.4021;
netsum += inarray[4] *  2.993273;
netsum += inarray[5] *  3.24317;
netsum += inarray[6] *  3.098715;
netsum += inarray[7] *  3.311838;
feature4[4] = 1 - exp(-netsum * netsum);

netsum =  2.346058;
netsum += inarray[0] *  52.16183;
netsum += inarray[1] * -34.7388;
netsum += inarray[2] * -5.455485;
netsum += inarray[3] * -2.306767;
netsum += inarray[4] * -2.129963;
netsum += inarray[5] * -2.153414;
netsum += inarray[6] * -2.292517;
netsum += inarray[7] * -2.003141;
feature4[5] = 1 - exp(-netsum * netsum);

netsum =  5.319527;
netsum += inarray[0] * -.6159033;
netsum += inarray[1] *  31.75972;
netsum += inarray[2] *  .884829;
netsum += inarray[3] * -5.417443;
netsum += inarray[4] * -5.498423;
netsum += inarray[5] * -5.446029;
netsum += inarray[6] * -5.16186;
netsum += inarray[7] * -5.159617;
feature4[6] = 1 - exp(-netsum * netsum);

netsum = -.4315916;
netsum += inarray[0] *  8.665919E-02;
netsum += inarray[1] *  .3110738;
netsum += inarray[2] *  .5757611;
netsum += inarray[3] *  .1681135;
netsum += inarray[4] *  .2672513;
netsum += inarray[5] *  .5854143;
netsum += inarray[6] *  .5837115;
netsum += inarray[7] *  .4895708;
feature4[7] = 1 - exp(-netsum * netsum);

netsum =  1.374799;
netsum += inarray[0] *  13.65605;
netsum += inarray[1] * -1.652376;
netsum += inarray[2] * -5.266515;
netsum += inarray[3] * -1.864814;
netsum += inarray[4] * -1.440267;
netsum += inarray[5] * -1.756514;
netsum += inarray[6] * -1.761236;
netsum += inarray[7] * -1.891923;
feature4[8] = 1 - exp(-netsum * netsum);

netsum =  1.610957;
netsum += inarray[0] *  11.32206;
netsum += inarray[1] * -1.670251;
netsum += inarray[2] * -.9508961;
netsum += inarray[3] * -1.496588;
netsum += inarray[4] * -1.891908;
netsum += inarray[5] * -2.066816;
netsum += inarray[6] * -1.554744;
netsum += inarray[7] * -1.797325;
feature4[9] = 1 - exp(-netsum * netsum);
```

```
netsum =  .3727364;
netsum += inarray[0] * -.1080832;
netsum += inarray[1] * -.3784903;
netsum += inarray[2] * -.5236476;
netsum += inarray[3] * -.512477;
netsum += inarray[4] * -.5814909;
netsum += inarray[5] * -.4689965;
netsum += inarray[6] * -.4042488;
netsum += inarray[7] * -.2496439;
feature4[10] = 1 - exp(-netsum * netsum);

netsum = -.5800024;
netsum += inarray[0] * -15.35357;
netsum += inarray[1] * -16.49053;
netsum += inarray[2] * -.2846282;
netsum += inarray[3] *  .3137432;
netsum += inarray[4] *  .4418969;
netsum += inarray[5] *  .4254741;
netsum += inarray[6] *  .8445359;
netsum += inarray[7] *  .8336233;
feature4[11] = 1 - exp(-netsum * netsum);

netsum =  1.078498;
netsum += inarray[0] * -14.58774;
netsum += inarray[1] *  16.51487;
netsum += inarray[2] * -.351595;
netsum += inarray[3] * -1.332972;
netsum += inarray[4] * -1.048703;
netsum += inarray[5] * -1.221178;
netsum += inarray[6] * -1.066221;
netsum += inarray[7] * -1.290772;
feature4[12] = 1 - exp(-netsum * netsum);

netsum = -1.358942;
netsum += inarray[0] * -3.852341;
netsum += inarray[1] * -3.067793;
netsum += inarray[2] * -5.765327;
netsum += inarray[3] *  1.077273;
netsum += inarray[4] *  1.420844;
netsum += inarray[5] *  1.373874;
netsum += inarray[6] *  1.017596;
netsum += inarray[7] *  1.260324;
feature4[13] = 1 - exp(-netsum * netsum);

netsum = -.5035008;
netsum += inarray[0] * -5.019427E-02;
netsum += inarray[1] *  .4097015;
netsum += inarray[2] *  .2812212;
netsum += inarray[3] *  .664559;
netsum += inarray[4] *  .5329266;
netsum += inarray[5] *  .3536009;
netsum += inarray[6] *  .3595358;
netsum += inarray[7] *  .6405866;
feature4[14] = 1 - exp(-netsum * netsum);

netsum =  2.45554;
netsum += inarray[0] *  18.59668;
netsum += inarray[1] * -.6207746;
netsum += inarray[2] * -2.762791;
netsum += inarray[3] * -2.250782;
netsum += inarray[4] * -2.119813;
netsum += inarray[5] * -2.100881;
netsum += inarray[6] * -2.427184;
netsum += inarray[7] * -2.116592;
feature4[15] = 1 - exp(-netsum * netsum);

netsum =  .7315203;
netsum += inarray[0] * -7.771874E-02;
netsum += inarray[1] *  9.509701E-02;
netsum += inarray[2] * -.1093585;
netsum += inarray[3] * -.7124208;
netsum += inarray[4] * -.7139043;
netsum += inarray[5] * -.5349607;
netsum += inarray[6] * -.4433709;
netsum += inarray[7] * -.2151675;
feature4[16] = 1 - exp(-netsum * netsum);

netsum = -.1594291;
netsum += feature2[0] *  9.982102E-02;
netsum += feature2[1] *  9.853359E-02;
netsum += feature2[2] *  1.160011E-02;
netsum += feature2[3] * -7.884867E-03;
netsum += feature2[4] *  5.041399E-02;
netsum += feature2[5] * -7.010787E-02;
netsum += feature2[6] * -1.703526E-03;
netsum += feature2[7] * -4.624376E-02;
netsum += feature2[8] *  3.095055E-02;
netsum += feature2[9] *  .1534905;
netsum += feature2[10] *  7.450251E-02;
netsum += feature2[11] *  2.583472E-02;
netsum += feature2[12] *  9.738775E-02;
netsum += feature2[13] * -5.756174E-02;
netsum += feature2[14] * -6.894302E-02;
netsum += feature2[15] * -.0865636;
netsum += feature2[16] *  4.068426E-02;
netsum += -.1389937;
netsum += feature3[0] *  .275925;
netsum += feature3[1] *  .7789719;
netsum += feature3[2] * -9.393725E-02;
netsum += feature3[3] * -.1160613;
netsum += feature3[4] * -.2673727;
netsum += feature3[5] * -.3569014;
netsum += feature3[6] *  6.152876E-02;
netsum += feature3[7] * -.2406653;
netsum += feature3[8] *  .6653302;
netsum += feature3[9] *  .8707212;
netsum += feature3[10] * -2.595287E-02;
netsum += feature3[11] * -7.001323E-02;
netsum += feature3[12] * -.1434766;
netsum += feature3[13] *  2.249397E-02;
netsum += feature3[14] * -.2505375;
netsum += feature3[15] *  1.133528;
netsum += feature3[16] *  .284638;
```

```
netsum +=  .1000453;
netsum += feature4[0] * -.5823139;
netsum += feature4[1] *  .0685712;
netsum += feature4[2] *  .3089656;
netsum += feature4[3] *  .1658631;
netsum += feature4[4] *  .1620492;
netsum += feature4[5] *  .0302265;
netsum += feature4[6] * -9.756247E-02;
netsum += feature4[7] * -3.317256E-02;
netsum += feature4[8] * -.1874768;
netsum += feature4[9] *  1.631775;
netsum += feature4[10] * -3.060556E-02;
netsum += feature4[11] * -1.299931;
netsum += feature4[12] * -1.186932E-02;
netsum += feature4[13] * -6.463331E-02;
netsum += feature4[14] * -.156519;
netsum += feature4[15] *  .106286;
netsum += feature4[16] * -.2376102;
outarray[0] = 1 / (1 + exp(-netsum));

netsum = -6.428884E-02;
netsum += feature2[0] *  .1000523;
netsum += feature2[1] *  9.855926E-02;
netsum += feature2[2] *  1.133594E-02;
netsum += feature2[3] *  8.363542E-02;
netsum += feature2[4] *  4.975718E-02;
netsum += feature2[5] * -7.038063E-02;
netsum += feature2[6] * -2.697068E-03;
netsum += feature2[7] * -4.650452E-02;
netsum += feature2[8] *  3.121076E-02;
netsum += feature2[9] *  .1075884;
netsum += feature2[10] *  7.464791E-02;
netsum += feature2[11] *  2.642348E-02;
netsum += feature2[12] *  9.723803E-02;
netsum += feature2[13] * -5.719874E-02;
netsum += feature2[14] * -6.903163E-02;
netsum += feature2[15] * -8.605916E-02;
netsum += feature2[16] * -4.550292E-02;
netsum += -.352198;
netsum += feature3[0] *  .2160453;
netsum += feature3[1] *  .7776144;
netsum += feature3[2] *  .1321614;
netsum += feature3[3] * -.1147544;
netsum += feature3[4] *  .205361;
netsum += feature3[5] * -.3554764;
netsum += feature3[6] *  6.118999E-02;
netsum += feature3[7] *  .1375634;
netsum += feature3[8] *  .664855;
netsum += feature3[9] *  .8721535;
netsum += feature3[10] * -2.623136E-02;
netsum += feature3[11] * -.4207377;
netsum += feature3[12] * -.1436497;
netsum += feature3[13] *  2.318728E-02;
netsum += feature3[14] * -.2498246;
netsum += feature3[15] *  1.134554;
netsum += feature3[16] *  .2000191;
netsum += -1.270332E-03;

netsum += feature4[0] * -.5865219;
netsum += feature4[1] *  6.878597E-02;
netsum += feature4[2] *  .3076231;
netsum += feature4[3] *  .1683333;
netsum += feature4[4] *  .1612958;
netsum += feature4[5] *  3.025419E-02;
netsum += feature4[6] * -9.807137E-02;
netsum += feature4[7] * -.2508459;
netsum += feature4[8] * -.19026;
netsum += feature4[9] *  1.629992;
netsum += feature4[10] * -.3105591;
netsum += feature4[11] * -1.302134;
netsum += feature4[12] * -1.178028E-02;
netsum += feature4[13] * -6.479353E-02;
netsum += feature4[14] *  2.258279E-02;
netsum += feature4[15] *  .1061629;
netsum += feature4[16] * -6.916831E-02;
outarray[1] = 1 / (1 + exp(-netsum));

netsum =  .1985629;
netsum += feature2[0] *  9.914069E-02;
netsum += feature2[1] *  9.673495E-02;
netsum += feature2[2] *  9.303743E-03;
netsum += feature2[3] * -.2547946;
netsum += feature2[4] *  4.962596E-02;
netsum += feature2[5] * -6.961165E-02;
netsum += feature2[6] * -6.797703E-04;
netsum += feature2[7] * -.0459341;
netsum += feature2[8] *  3.090797E-02;
netsum += feature2[9] * -8.292437E-02;
netsum += feature2[10] *  7.472249E-02;
netsum += feature2[11] *  2.802641E-02;
netsum += feature2[12] *  9.646337E-02;
netsum += feature2[13] * -5.711888E-02;
netsum += feature2[14] * -6.988299E-02;
netsum += feature2[15] * -8.478034E-02;
netsum += feature2[16] *  7.623917E-02;
netsum += -.18998;
netsum += feature3[0] *  .3716987;
netsum += feature3[1] *  .7722573;
netsum += feature3[2] * -.2041326;
netsum += feature3[3] * -.1159883;
netsum += feature3[4] * -.2287805;
netsum += feature3[5] * -.3508427;
netsum += feature3[6] *  6.650011E-02;
netsum += feature3[7] *  .1780408;
netsum += feature3[8] *  .6623566;
netsum += feature3[9] *  .8658233;
netsum += feature3[10] * -2.626068E-02;
netsum += feature3[11] *  1.342634E-02;
netsum += feature3[12] * -.1453275;
netsum += feature3[13] *  2.584079E-02;
netsum += feature3[14] * -.2459468;
netsum += feature3[15] *  1.141873;
netsum += feature3[16] *  .4249638;
netsum += -.1526113;
```

```
netsum += feature4[0] * -.5860531;
netsum += feature4[1] *  6.803294E-02;
netsum += feature4[2] *  .3086981;
netsum += feature4[3] *  .1825872;
netsum += feature4[4] *  .1589908;
netsum += feature4[5] *  .0297163;
netsum += feature4[6] * -.1064095;
netsum += feature4[7] * -.1084175;
netsum += feature4[8] * -.184821;
netsum += feature4[9] *  1.633342;
netsum += feature4[10] * -5.600537E-02;
netsum += feature4[11] * -1.294336;
netsum += feature4[12] * -1.171703E-02;
netsum += feature4[13] * -6.570626E-02;
netsum += feature4[14] * -.1294171;
netsum += feature4[15] *  .1063571;
netsum += feature4[16] * -.2105135;
outarray[2] = 1 / (1 + exp(-netsum));

netsum = -.3908116;
netsum += feature2[0] *  .1001915;
netsum += feature2[1] *  9.762753E-02;
netsum += feature2[2] *  9.700065E-03;
netsum += feature2[3] *  4.165382E-02;
netsum += feature2[4] *  4.816473E-02;
netsum += feature2[5] * -7.062761E-02;
netsum += feature2[6] * -4.033762E-03;
netsum += feature2[7] * -4.681948E-02;
netsum += feature2[8] *  3.176474E-02;
netsum += feature2[9] * -.1045747;
netsum += feature2[10] *  7.504913E-02;
netsum += feature2[11] *  2.865907E-02;
netsum += feature2[12] *  9.644988E-02;
netsum += feature2[13] * -5.629505E-02;
netsum += feature2[14] * -6.967079E-02;
netsum += feature2[15] * -8.418839E-02;
netsum += feature2[16] *  1.503857E-02;
netsum += -.1721451;
netsum += feature3[0] *  .2748244;
netsum += feature3[1] *  .7715641;
netsum += feature3[2] *  .0468402;
netsum += feature3[3] * -.1124184;
netsum += feature3[4] * -.1858345;
netsum += feature3[5] * -.3495999;
netsum += feature3[6] *  6.304035E-02;
netsum += feature3[7] *  .1234029;
netsum += feature3[8] *  .6624152;
netsum += feature3[9] *  .8721729;
netsum += feature3[10] * -2.696152E-02;
netsum += feature3[11] *  .1084971;
netsum += feature3[12] * -.1449545;
netsum += feature3[13] *  2.403444E-02;
netsum += feature3[14] * -.2461066;
netsum += feature3[15] *  1.140569;
netsum += feature3[16] *  .1114481;
netsum +=  6.324437E-02;
netsum += feature4[0] * -.5961475;
netsum += feature4[1] *  6.891977E-02;
netsum += feature4[2] *  .3050192;
netsum += feature4[3] *  .1815865;
netsum += feature4[4] *  .1582266;
netsum += feature4[5] *  3.004046E-02;
netsum += feature4[6] * -.1031753;
netsum += feature4[7] * -.1259796;
netsum += feature4[8] * -.1935712;
netsum += feature4[9] *  1.627391;
netsum += feature4[10] * -.3509837;
netsum += feature4[11] * -1.303012;
netsum += feature4[12] * -1.154152E-02;
netsum += feature4[13] * -6.571095E-02;
netsum += feature4[14] * -.1654664;
netsum += feature4[15] *  .1059589;
netsum += feature4[16] *  .0751171;
outarray[3] = 1 / (1 + exp(-netsum));

netsum =  .2015459;
netsum += feature2[0] *  .0987739;
netsum += feature2[1] *  9.593422E-02;
netsum += feature2[2] *  8.385834E-03;
netsum += feature2[3] *  .2482599;
netsum += feature2[4] *  4.948243E-02;
netsum += feature2[5] * -6.930164E-02;
netsum += feature2[6] *  9.54523E-05;
netsum += feature2[7] * -4.571648E-02;
netsum += feature2[8] *  3.083443E-02;
netsum += feature2[9] * -.1761675;
netsum += feature2[10] *  7.476703E-02;
netsum += feature2[11] *  2.880708E-02;
netsum += feature2[12] *  9.609863E-02;
netsum += feature2[13] * -5.704242E-02;
netsum += feature2[14] * -7.027869E-02;
netsum += feature2[15] * -8.413721E-02;
netsum += feature2[16] * -.1651714;
netsum += -2.817038E-03;
netsum += feature3[0] *  4.778834E-03;
netsum += feature3[1] *  .7697378;
netsum += feature3[2] *  2.340295E-02;
netsum += feature3[3] * -.1163804;
netsum += feature3[4] * -.2041107;
netsum += feature3[5] * -.348613;
netsum += feature3[6] *  6.873964E-02;
netsum += feature3[7] *  .1158195;
netsum += feature3[8] *  .6612103;
netsum += feature3[9] *  .8631743;
netsum += feature3[10] * -2.633168E-02;
netsum += feature3[11] * -.1492229;
netsum += feature3[12] * -.146102;
netsum += feature3[13] *  .0270249;
netsum += feature3[14] * -.2441584;
netsum += feature3[15] *  1.145192;
netsum += feature3[16] *  .5164756;
netsum += -.3051459;
netsum += feature4[0] * -.5864208;
netsum += feature4[1] *  6.772359E-02;
```

```
netsum += feature4[2] * .309004;
netsum += feature4[3] * .1890782;
netsum += feature4[4] * .1578233;
netsum += feature4[5] * 2.947736E-02;
netsum += feature4[6] * -.1099848;
netsum += feature4[7] * .0488277;
netsum += feature4[8] * -.1828259;
netsum += feature4[9] * 1.634616;
netsum += feature4[10] * -.2380027;
netsum += feature4[11] * -1.291197;
netsum += feature4[12] * -1.168022E-02;
netsum += feature4[13] * -.0661381;
netsum += feature4[14] * -.1960984;
netsum += feature4[15] * .1064254;
netsum += feature4[16] * -.1863879;
outarray[4] = 1 / (1 + exp(-netsum));

netsum = -.1052504;
netsum += feature2[0] * .1003509;
netsum += feature2[1] * 9.811052E-02;
netsum += feature2[2] * 1.038229E-02;
netsum += feature2[3] * -.2122504;
netsum += feature2[4] * 4.850223E-02;
netsum += feature2[5] * -7.071573E-02;
netsum += feature2[6] * -4.137163E-03;
netsum += feature2[7] * -4.686007E-02;
netsum += feature2[8] * 3.176082E-02;
netsum += feature2[9] * -3.145891E-02;
netsum += feature2[10] * 7.496937E-02;
netsum += feature2[11] * 2.796077E-02;
netsum += feature2[12] * 9.672735E-02;
netsum += feature2[13] * -5.647707E-02;
netsum += feature2[14] * -6.939258E-02;
netsum += feature2[15] * -8.476541E-02;
netsum += feature2[16] * -.2022875;
netsum += 2.032227E-02;
netsum += feature3[0] * .1985782;
netsum += feature3[1] * .7736316;
netsum += feature3[2] * .2309931;
netsum += feature3[3] * -.1126861;
netsum += feature3[4] * -.3369415;
netsum += feature3[5] * -.3514898;
netsum += feature3[6] * 6.168573E-02;
netsum += feature3[7] * -.0410716;
netsum += feature3[8] * .6633248;
netsum += feature3[9] * .8732511;
netsum += feature3[10] * -2.685198E-02;
netsum += feature3[11] * -7.138541E-02;
netsum += feature3[12] * -.1444165;
netsum += feature3[13] * 2.253906E-02;
netsum += feature3[14] * -.2474901;
netsum += feature3[15] * 1.138046;
netsum += feature3[16] * 8.818797E-02;
netsum += -.1232352;
netsum += feature4[0] * -.5943624;
netsum += feature4[1] * .0690288;
netsum += feature4[2] * .3053317;
netsum += feature4[3] * .1765742;
netsum += feature4[4] * .1591887;
netsum += feature4[5] * .0301742;
netsum += feature4[6] * -.1005835;
netsum += feature4[7] * 1.822072E-02;
netsum += feature4[8] * -.1937737;
netsum += feature4[9] * 1.627272;
netsum += feature4[10] * -.1067787;
netsum += feature4[11] * -1.304154;
netsum += feature4[12] * -1.160129E-02;
netsum += feature4[13] * -6.539606E-02;
netsum += feature4[14] * 8.239657E-02;
netsum += feature4[15] * .105958;
netsum += feature4[16] * .1311692;
outarray[5] = 1 / (1 + exp(-netsum));

netsum = -.3260403;
netsum += feature2[0] * .1007293;
netsum += feature2[1] * 9.954499E-02;
netsum += feature2[2] * 1.239564E-02;
netsum += feature2[3] * -8.844779E-03;
netsum += feature2[4] * 4.955441E-02;
netsum += feature2[5] * -7.092898E-02;
netsum += feature2[6] * -4.27072E-03;
netsum += feature2[7] * -.0469407;
netsum += feature2[8] * 3.159972E-02;
netsum += feature2[9] * .1862136;
netsum += feature2[10] * 7.467406E-02;
netsum += feature2[11] * 2.581532E-02;
netsum += feature2[12] * 9.756291E-02;
netsum += feature2[13] * -5.708216E-02;
netsum += feature2[14] * -6.858984E-02;
netsum += feature2[15] * -8.652391E-02;
netsum += feature2[16] * -.3306651;
netsum += -.3426494;
netsum += feature3[0] * 2.163002E-02;
netsum += feature3[1] * .7799175;
netsum += feature3[2] * -1.346784E-02;
netsum += feature3[3] * -.113593;
netsum += feature3[4] * -.2756725;
netsum += feature3[5] * -.3573031;
netsum += feature3[6] * 5.790848E-02;
netsum += feature3[7] * -.3678144;
netsum += feature3[8] * .6660095;
netsum += feature3[9] * .8761922;
netsum += feature3[10] * -2.640462E-02;
netsum += feature3[11] * -.2817113;
netsum += feature3[12] * -.1428319;
netsum += feature3[13] * 1.988129E-02;
netsum += feature3[14] * -.2516357;
netsum += feature3[15] * 1.130661;
netsum += feature3[16] * 3.666928E-02;
netsum += 5.705023E-02;
netsum += feature4[0] * -.5886109;
netsum += feature4[1] * 6.930493E-02;
netsum += feature4[2] * .3064503;
netsum += feature4[3] * .1615161;
```

```
netsum += feature4[4] * .1620871;
netsum += feature4[5] * 3.055244E-02;
netsum += feature4[6] * -9.308289E-02;
netsum += feature4[7] * 2.466139E-02;
netsum += feature4[8] * -.1942007;
netsum += feature4[9] * 1.627284;
netsum += feature4[10] * 9.149173E-02;
netsum += feature4[11] * -1.307202;
netsum += feature4[12] * -1.178822E-02;
netsum += feature4[13] * -6.441652E-02;
netsum += feature4[14] * 6.120576E-02;
netsum += feature4[15] * .1059878;
netsum += feature4[16] * 7.528453E-02;
outarray[6] = 1 / (1 + exp(-netsum));

netsum = -.118895;
netsum += feature2[0] * .1002182;
netsum += feature2[1] * 9.804522E-02;
netsum += feature2[2] * 1.037404E-02;
netsum += feature2[3] * -.2052928;
netsum += feature2[4] * 4.868957E-02;
netsum += feature2[5] * -7.060495E-02;
netsum += feature2[6] * -3.749672E-03;
netsum += feature2[7] * -4.675575E-02;
netsum += feature2[8] * 3.161895E-02;
netsum += feature2[9] * -2.559258E-02;
netsum += feature2[10] * 7.491428E-02;
netsum += feature2[11] * 2.782566E-02;
netsum += feature2[12] * .0967501;
netsum += feature2[13] * -5.659066E-02;
netsum += feature2[14] * -6.940717E-02;
netsum += feature2[15] * -8.487576E-02;
netsum += feature2[16] * .0575963;
netsum += 5.793406E-02;
netsum += feature3[0] * .2107144;
netsum += feature3[1] * .7738645;
netsum += feature3[2] * -.21908;
netsum += feature3[3] * -.1131044;
netsum += feature3[4] * -.396398;
netsum += feature3[5] * -.3517993;
netsum += feature3[6] * 6.206824E-02;
netsum += feature3[7] * -.319727;
netsum += feature3[8] * .6633617;
netsum += feature3[9] * .8725704;
netsum += feature3[10] * -2.672952E-02;
netsum += feature3[11] * 5.406847E-03;
netsum += feature3[12] * -.144426;
netsum += feature3[13] * 2.340317E-02;
netsum += feature3[14] * -.2475646;
netsum += feature3[15] * 1.138089;
netsum += feature3[16] * .1225536;
netsum += -.2009033;
netsum += feature4[0] * -.5930761;
netsum += feature4[1] * 6.893262E-02;
netsum += feature4[2] * .3057766;
netsum += feature4[3] * .1763407;
netsum += feature4[4] * .1593465;
netsum += feature4[5] * 3.014747E-02;
netsum += feature4[6] * -.1008639;
netsum += feature4[7] * -.1755624;
netsum += feature4[8] * -.1928001;
netsum += feature4[9] * 1.627993;
netsum += feature4[10] * -.225705;
netsum += feature4[11] * -1.303181;
netsum += feature4[12] * -1.162348E-02;
netsum += feature4[13] * -6.536502E-02;
netsum += feature4[14] * .0529286;
netsum += feature4[15] * .1060135;
netsum += feature4[16] * .1073704;
outarray[7] = 1 / (1 + exp(-netsum));

netsum = -.2938153;
netsum += feature2[0] * 9.912894E-02;
netsum += feature2[1] * 9.685824E-02;
netsum += feature2[2] * .0095728;
netsum += feature2[3] * -.2825736;
netsum += feature2[4] * 4.989893E-02;
netsum += feature2[5] * -6.955232E-02;
netsum += feature2[6] * -4.255117E-04;
netsum += feature2[7] * -.0458745;
netsum += feature2[8] * 3.085213E-02;
netsum += feature2[9] * -5.105687E-02;
netsum += feature2[10] * 7.465762E-02;
netsum += feature2[11] * 2.768002E-02;
netsum += feature2[12] * 9.657522E-02;
netsum += feature2[13] * -.0572731;
netsum += feature2[14] * -6.978238E-02;
netsum += feature2[15] * -8.506852E-02;
netsum += feature2[16] * -.29689;
netsum += -.327196;
netsum += feature3[0] * .2830801;
netsum += feature3[1] * .7731898;
netsum += feature3[2] * -4.565427E-02;
netsum += feature3[3] * -.1164308;
netsum += feature3[4] * .0761206;
netsum += feature3[5] * -.3517344;
netsum += feature3[6] * 6.616295E-02;
netsum += feature3[7] * -.2690493;
netsum += feature3[8] * .6627409;
netsum += feature3[9] * .8657095;
netsum += feature3[10] * -2.616689E-02;
netsum += feature3[11] * -7.106339E-02;
netsum += feature3[12] * -.1451423;
netsum += feature3[13] * .0249773;
netsum += feature3[14] * -.2465157;
netsum += feature3[15] * 1.140831;
netsum += feature3[16] * .4444143;
netsum += .1857142;
netsum += feature4[0] * -.5844128;
netsum += feature4[1] * 6.799798E-02;
netsum += feature4[2] * .3091715;
netsum += feature4[3] * .1805023;
netsum += feature4[4] * .1594302;
netsum += feature4[5] * 2.973893E-02;
```

```
netsum += feature4[6] * -.1054388;
netsum += feature4[7] *  6.904215E-02;
netsum += feature4[8] * -.1841333;
netsum += feature4[9] *  1.633796;
netsum += feature4[10] *  .101162;
netsum += feature4[11] * -1.294101;
netsum += feature4[12] * -1.176156E-02;
netsum += feature4[13] * -6.558147E-02;
netsum += feature4[14] *  .0798706;
netsum += feature4[15] *  .1063841;
netsum += feature4[16] *  .2213631;
outarray[8] = 1 / (1 + exp(-netsum));

netsum =  .133593;
netsum += feature2[0] *  .1001166;
netsum += feature2[1] *  9.827905E-02;
netsum += feature2[2] *  1.085068E-02;
netsum += feature2[3] * -.2988594;
netsum += feature2[4] *  4.921663E-02;
netsum += feature2[5] * -7.044494E-02;
netsum += feature2[6] * -3.151041E-03;
netsum += feature2[7] * -4.665584E-02;
netsum += feature2[8] *  3.143203E-02;
netsum += feature2[9] *  3.432573E-02;
netsum += feature2[10] *  7.476057E-02;
netsum += feature2[11] *  2.712989E-02;
netsum += feature2[12] *  9.697919E-02;
netsum += feature2[13] * -5.692467E-02;
netsum += feature2[14] * -6.920125E-02;
netsum += feature2[15] * -8.543494E-02;
netsum += feature2[16] * -.339539;
netsum +=  8.342457E-03;
netsum += feature3[0] *  3.083527E-02;
netsum += feature3[1] *  .7756486;
netsum += feature3[2] * -1.526052E-02;
netsum += feature3[3] * -.1139742;
netsum += feature3[4] *  4.378567E-02;
netsum += feature3[5] * -.3535835;
netsum += feature3[6] *  6.163523E-02;
netsum += feature3[7] * -.1937318;
netsum += feature3[8] *  .6640502;
netsum += feature3[9] *  .8722742;
netsum += feature3[10] * -2.650845E-02;
netsum += feature3[11] * -8.772483E-02;
netsum += feature3[12] * -.1440252;
netsum += feature3[13] *  2.262983E-02;
netsum += feature3[14] * -.2486685;
netsum += feature3[15] *  1.13624;
netsum += feature3[16] *  .1679158;
netsum += -.3111052;
netsum += feature4[0] * -.5898148;
netsum += feature4[1] *  6.877413E-02;
netsum += feature4[2] *  .3067056;
netsum += feature4[3] *  .1725509;
netsum += feature4[4] *  .1600997;
netsum += feature4[5] *  3.022915E-02;
netsum += feature4[6] * -.0993307;
netsum += feature4[7] * -.1093916;
netsum += feature4[8] * -.1913845;
netsum += feature4[9] *  1.629062;
netsum += feature4[10] *  0;
netsum += feature4[11] * -1.302502;
netsum += feature4[12] * -1.173952E-02;
netsum += feature4[13] * -6.511384E-02;
netsum += feature4[14] * -.3133606;
netsum += feature4[15] *  .1060979;
netsum += feature4[16] *  .114139;
outarray[9] = 1 / (1 + exp(-netsum));

netsum = -6.522733E-02;
netsum += feature2[0] * -3.054574E-02;
netsum += feature2[1] *  4.341216E-02;
netsum += feature2[2] * -1.114519E-02;
netsum += feature2[3] *  3.359599E-02;
netsum += feature2[4] * -2.340689E-02;
netsum += feature2[5] * -4.073283E-02;
netsum += feature2[6] *  .2724735;
netsum += feature2[7] * -8.708645E-02;
netsum += feature2[8] * -4.942606E-02;
netsum += feature2[9] * -6.169086E-02;
netsum += feature2[10] * -2.728949E-03;
netsum += feature2[11] *  4.851996E-02;
netsum += feature2[12] * -2.083569E-02;
netsum += feature2[13] * -3.611483E-02;
netsum += feature2[14] * -2.211943E-03;
netsum += feature2[15] * -.1132458;
netsum += feature2[16] * -.2355234;
netsum += -.1110575;
netsum += feature3[0] *  6.538279E-02;
netsum += feature3[1] *  .06424;
netsum += feature3[2] * -.1495663;
netsum += feature3[3] *  .0426062;
netsum += feature3[4] *  .1237102;
netsum += feature3[5] *  .1970609;
netsum += feature3[6] * -.1840101;
netsum += feature3[7] * -.1481063;
netsum += feature3[8] * -3.639089E-02;
netsum += feature3[9] * -9.426732E-02;
netsum += feature3[10] *  .1039559;
netsum += feature3[11] * -.2439626;
netsum += feature3[12] *  .2122959;
netsum += feature3[13] * -.1179142;
netsum += feature3[14] * -4.235541E-02;
netsum += feature3[15] *  9.513943E-02;
netsum += feature3[16] *  8.088798E-02;
netsum += -.1943247;
netsum += feature4[0] *  .2025577;
netsum += feature4[1] *  9.187154E-02;
netsum += feature4[2] *  .6591138;
netsum += feature4[3] *  .026295;
netsum += feature4[4] *  7.186238E-02;
netsum += feature4[5] * -3.288485E-02;
netsum += feature4[6] *  .1597732;
netsum += feature4[7] * -.2655662;
```

```
netsum += feature4[8] * -.4842804;
netsum += feature4[9] * -1.006997;
netsum += feature4[10] * -.1318907;
netsum += feature4[11] * .3510543;
netsum += feature4[12] * 1.301063E-02;
netsum += feature4[13] * -.0535349;
netsum += feature4[14] * -.2203403;
netsum += feature4[15] * .0334351;
netsum += feature4[16] * -3.663665E-02;
outarray[10] = 1 / (1 + exp(-netsum));

netsum = -.0501876;
netsum += feature2[0] * -2.959599E-02;
netsum += feature2[1] * 4.415477E-02;
netsum += feature2[2] * -1.062738E-02;
netsum += feature2[3] * 5.975494E-02;
netsum += feature2[4] * -2.330016E-02;
netsum += feature2[5] * -4.088946E-02;
netsum += feature2[6] * .2720853;
netsum += feature2[7] * -.0873724;
netsum += feature2[8] * -4.928346E-02;
netsum += feature2[9] * 6.376716E-02;
netsum += feature2[10] * -2.843303E-03;
netsum += feature2[11] * 4.790636E-02;
netsum += feature2[12] * -2.048941E-02;
netsum += feature2[13] * -3.622447E-02;
netsum += feature2[14] * -1.90576E-03;
netsum += feature2[15] * -.1140309;
netsum += feature2[16] * 3.804721E-02;
netsum += .1453397;
netsum += feature3[0] * 5.234855E-02;
netsum += feature3[1] * 6.675693E-02;
netsum += feature3[2] * .2729555;
netsum += feature3[3] * 4.316342E-02;
netsum += feature3[4] * -.4069374;
netsum += feature3[5] * .1948416;
netsum += feature3[6] * -.18615;
netsum += feature3[7] * -.3101534;
netsum += feature3[8] * -.0352412;
netsum += feature3[9] * -9.057976E-02;
netsum += feature3[10] * .1040554;
netsum += feature3[11] * 9.583195E-02;
netsum += feature3[12] * .2131041;
netsum += feature3[13] * -.119564;
netsum += feature3[14] * -4.424088E-02;
netsum += feature3[15] * .0914415;
netsum += feature3[16] * -2.256233E-02;
netsum += -.4283122;
netsum += feature4[0] * .2033653;
netsum += feature4[1] * 9.208061E-02;
netsum += feature4[2] * .6587797;
netsum += feature4[3] * 1.923892E-02;
netsum += feature4[4] * 7.329198E-02;
netsum += feature4[5] * -3.267801E-02;
netsum += feature4[6] * .1634086;
netsum += feature4[7] * -.2252316;
netsum += feature4[8] * -.4858644;
netsum += feature4[9] * -1.007305;
netsum += feature4[10] * -.1994335;
netsum += feature4[11] * .3474331;
netsum += feature4[12] * 1.320062E-02;
netsum += feature4[13] * -5.284429E-02;
netsum += feature4[14] * .1767307;
netsum += feature4[15] * 3.333941E-02;
netsum += feature4[16] * -.1960846;
outarray[11] = 1 / (1 + exp(-netsum));

netsum = -.2572922;
netsum += feature2[0] * -2.868118E-02;
netsum += feature2[1] * 4.491434E-02;
netsum += feature2[2] * -1.002627E-02;
netsum += feature2[3] * 1.981604E-02;
netsum += feature2[4] * -.0222554;
netsum += feature2[5] * -4.069107E-02;
netsum += feature2[6] * .2730828;
netsum += feature2[7] * -.0874341;
netsum += feature2[8] * -4.945682E-02;
netsum += feature2[9] * .2771727;
netsum += feature2[10] * -3.061155E-03;
netsum += feature2[11] * 4.666978E-02;
netsum += feature2[12] * -1.990342E-02;
netsum += feature2[13] * -3.711128E-02;
netsum += feature2[14] * -1.416306E-03;
netsum += feature2[15] * -.115778;
netsum += feature2[16] * -.2325987;
netsum += -9.789654E-02;
netsum += feature3[0] * .276757;
netsum += feature3[1] * 7.125701E-02;
netsum += feature3[2] * .1630376;
netsum += feature3[3] * 4.146578E-02;
netsum += feature3[4] * -7.072899E-02;
netsum += feature3[5] * .1906045;
netsum += feature3[6] * -.1878683;
netsum += feature3[7] * .1815547;
netsum += feature3[8] * -3.319318E-02;
netsum += feature3[9] * -8.940519E-02;
netsum += feature3[10] * .1044195;
netsum += feature3[11] * -7.148815E-02;
netsum += feature3[12] * .2140089;
netsum += feature3[13] * -.1209525;
netsum += feature3[14] * -4.709728E-02;
netsum += feature3[15] * 8.592208E-02;
netsum += feature3[16] * 2.605695E-02;
netsum += 0;
netsum += feature4[0] * .2114567;
netsum += feature4[1] * 9.210794E-02;
netsum += feature4[2] * .660675;
netsum += feature4[3] * 7.924065E-03;
netsum += feature4[4] * 7.582548E-02;
netsum += feature4[5] * -3.243656E-02;
netsum += feature4[6] * .1679284;
netsum += feature4[7] * -.3359252;
netsum += feature4[8] * -.4843538;
netsum += feature4[9] * -1.005204;
```

```
netsum += feature4[10] * -8.875046E-02;
netsum += feature4[11] *  .3469748;
netsum += feature4[12] *  1.331447E-02;
netsum += feature4[13] * -5.168577E-02;
netsum += feature4[14] * -.1515724;
netsum += feature4[15] *  .033594;
netsum += feature4[16] * -.2460766;
outarray[12] = 1 / (1 + exp(-netsum));

netsum =  .1022643;
netsum += feature2[0] * -3.038034E-02;
netsum += feature2[1] *  4.353132E-02;
netsum += feature2[2] * -1.102302E-02;
netsum += feature2[3] * -.288521;
netsum += feature2[4] * -2.311801E-02;
netsum += feature2[5] * -4.067944E-02;
netsum += feature2[6] *  .2727984;
netsum += feature2[7] * -8.705833E-02;
netsum += feature2[8] * -4.944885E-02;
netsum += feature2[9] * -.0159888;
netsum += feature2[10] * -2.764774E-03;
netsum += feature2[11] *  4.827134E-02;
netsum += feature2[12] * -2.071976E-02;
netsum += feature2[13] * -3.633041E-02;
netsum += feature2[14] * -2.111247E-03;
netsum += feature2[15] * -.1136392;
netsum += feature2[16] * -.1916261;
netsum += -.2741456;
netsum += feature3[0] *  .1117928;
netsum += feature3[1] *  6.520297E-02;
netsum += feature3[2] *  .2673859;
netsum += feature3[3] *  .0420403;
netsum += feature3[4] * -9.890612E-02;
netsum += feature3[5] *  .1961351;
netsum += feature3[6] * -.1843368;
netsum += feature3[7] *  7.309108E-03;
netsum += feature3[8] * -3.593513E-02;
netsum += feature3[9] * -9.439149E-02;
netsum += feature3[10] *  .1040307;
netsum += feature3[11] *  .0852882;
netsum += feature3[12] *  .2124608;
netsum += feature3[13] * -.1193822;
netsum += feature3[14] * -4.292914E-02;
netsum += feature3[15] *  9.395628E-02;
netsum += feature3[16] *  .1070056;
netsum += -.2476786;
netsum += feature4[0] *  .2048221;
netsum += feature4[1] *  9.185454E-02;
netsum += feature4[2] *  .6596648;
netsum += feature4[3] *  2.399607E-02;
netsum += feature4[4] *  7.238343E-02;
netsum += feature4[5] * -3.283779E-02;
netsum += feature4[6] *  .1607245;
netsum += feature4[7] *  .1499967;
netsum += feature4[8] * -.4835904;
netsum += feature4[9] * -1.006257;
netsum += feature4[10] * -.0272184;
netsum += feature4[11] *  .3513575;
netsum += feature4[12] *  1.303612E-02;
netsum += feature4[13] * -5.332446E-02;
netsum += feature4[14] * -.1401169;
netsum += feature4[15] *  3.353744E-02;
netsum += feature4[16] * -.12891;
outarray[13] = 1 / (1 + exp(-netsum));

netsum = -.2893554;
netsum += feature2[0] * -2.932112E-02;
netsum += feature2[1] *  4.450821E-02;
netsum += feature2[2] * -1.025935E-02;
netsum += feature2[3] * -.172492;
netsum += feature2[4] * -2.130971E-02;
netsum += feature2[5] * -4.037698E-02;
netsum += feature2[6] *  .274863;
netsum += feature2[7] * -.0869224;
netsum += feature2[8] * -4.975092E-02;
netsum += feature2[9] *  .2969054;
netsum += feature2[10] * -3.098844E-03;
netsum += feature2[11] *  4.642057E-02;
netsum += feature2[12] * -.0198223;
netsum += feature2[13] * -3.779371E-02;
netsum += feature2[14] * -1.392715E-03;
netsum += feature2[15] * -.1163471;
netsum += feature2[16] * -6.353115E-02;
netsum += -.1829565;
netsum += feature3[0] *  .1943101;
netsum += feature3[1] *  7.196278E-02;
netsum += feature3[2] *  .131328;
netsum += feature3[3] *  3.869554E-02;
netsum += feature3[4] *  .1208717;
netsum += feature3[5] *  .1894858;
netsum += feature3[6] * -.1863716;
netsum += feature3[7] * -4.461884E-02;
netsum += feature3[8] * -3.280183E-02;
netsum += feature3[9] * -9.454666E-02;
netsum += feature3[10] *  .1046807;
netsum += feature3[11] * -.1417543;
netsum += feature3[12] *  .2136522;
netsum += feature3[13] * -.1209876;
netsum += feature3[14] * -4.699887E-02;
netsum += feature3[15] *  8.613072E-02;
netsum += feature3[16] *  .2557569;
netsum += -.2619454;
netsum += feature4[0] *  .2193024;
netsum += feature4[1] *  9.176635E-02;
netsum += feature4[2] *  .6630647;
netsum += feature4[3] *  7.545715E-03;
netsum += feature4[4] *  7.610845E-02;
netsum += feature4[5] * -3.251156E-02;
netsum += feature4[6] *  .1668152;
netsum += feature4[7] *  .083841;
netsum += feature4[8] * -.4799366;
netsum += feature4[9] * -1.001264;
netsum += feature4[10] * -.128107;
netsum += feature4[11] *  .3525438;
```

```
netsum += feature4[12] *  1.317331E-02;
netsum += feature4[13] * -5.168484E-02;
netsum += feature4[14] * -.2783349;
netsum += feature4[15] *  3.422769E-02;
netsum += feature4[16] *  7.981069E-02;
outarray[14] = 1 / (1 + exp(-netsum));

netsum = -.2741528;
netsum += feature2[0] * -3.116153E-02;
netsum += feature2[1] *  4.296619E-02;
netsum += feature2[2] * -1.143191E-02;
netsum += feature2[3] * -.3139931;
netsum += feature2[4] * -2.291622E-02;
netsum += feature2[5] * -4.047732E-02;
netsum += feature2[6] *  .2735124;
netsum += feature2[7] * -8.674938E-02;
netsum += feature2[8] * -4.964071E-02;
netsum += feature2[9] * -8.544825E-02;
netsum += feature2[10] * -2.692203E-03;
netsum += feature2[11] *  4.855045E-02;
netsum += feature2[12] * -2.090456E-02;
netsum += feature2[13] * -3.645285E-02;
netsum += feature2[14] * -2.295979E-03;
netsum += feature2[15] * -.1133207;
netsum += feature2[16] * -.3226393;
netsum +=  .129195;
netsum += feature3[0] *  .0815168;
netsum += feature3[1] *  6.385499E-02;
netsum += feature3[2] *  .1155264;
netsum += feature3[3] *  4.094482E-02;
netsum += feature3[4] * -5.578401E-02;
netsum += feature3[5] *  .1971979;
netsum += feature3[6] * -.1825034;
netsum += feature3[7] *  .2118505;
netsum += feature3[8] * -.0365431;
netsum += feature3[9] * -9.800705E-02;
netsum += feature3[10] *  .1040418;
netsum += feature3[11] * -.3071151;
netsum += feature3[12] *  .2118848;
netsum += feature3[13] * -.1173377;
netsum += feature3[14] * -4.176278E-02;
netsum += feature3[15] *  9.643685E-02;
netsum += feature3[16] *  .2291704;
netsum += -.2029265;
netsum += feature4[0] *  .2064255;
netsum += feature4[1] *  9.164485E-02;
netsum += feature4[2] *  .6605132;
netsum += feature4[3] *  2.828936E-02;
netsum += feature4[4] *  7.169623E-02;
netsum += feature4[5] * -3.297894E-02;
netsum += feature4[6] *  .1578951;
netsum += feature4[7] * -.1263719;
netsum += feature4[8] * -.4814304;
netsum += feature4[9] * -1.00513;
netsum += feature4[10] *  4.222365E-02;
netsum += feature4[11] *  .3551157;
netsum += feature4[12] *  1.288048E-02;
netsum += feature4[13] * -5.372618E-02;
netsum += feature4[14] * -9.951999E-02;
netsum += feature4[15] *  3.375071E-02;
netsum += feature4[16] *  .2604431;
outarray[15] = 1 / (1 + exp(-netsum));

netsum =  6.613023E-02;
netsum += feature2[0] * -3.006203E-02;
netsum += feature2[1] *  4.393447E-02;
netsum += feature2[2] * -1.067366E-02;
netsum += feature2[3] * -.3501116;
netsum += feature2[4] * -2.157933E-02;
netsum += feature2[5] * -4.031204E-02;
netsum += feature2[6] *  .2748731;
netsum += feature2[7] * -8.675449E-02;
netsum += feature2[8] * -4.980562E-02;
netsum += feature2[9] *  .1812844;
netsum += feature2[10] * -2.984559E-03;
netsum += feature2[11] *  4.699773E-02;
netsum += feature2[12] * -2.012871E-02;
netsum += feature2[13] * -3.756076E-02;
netsum += feature2[14] * -1.67387E-03;
netsum += feature2[15] * -.1155424;
netsum += feature2[16] * -9.621643E-02;
netsum += -.1742156;
netsum += feature3[0] * -2.188879E-02;
netsum += feature3[1] *  6.964681E-02;
netsum += feature3[2] *  1.961952E-02;
netsum += feature3[3] *  3.873118E-02;
netsum += feature3[4] * -.0572844;
netsum += feature3[5] *  .1915997;
netsum += feature3[6] * -.1847646;
netsum += feature3[7] * -4.933616E-02;
netsum += feature3[8] * -3.385823E-02;
netsum += feature3[9] * -9.683992E-02;
netsum += feature3[10] *  .1045436;
netsum += feature3[11] * -8.491701E-02;
netsum += feature3[12] *  .2130154;
netsum += feature3[13] * -.1191296;
netsum += feature3[14] * -4.537604E-02;
netsum += feature3[15] *  8.942993E-02;
netsum += feature3[16] *  .3017158;
netsum += -.2663296;
netsum += feature4[0] *  .2171185;
netsum += feature4[1] *  9.164679E-02;
netsum += feature4[2] *  .6628636;
netsum += feature4[3] *  1.374077E-02;
netsum += feature4[4] *  7.481817E-02;
netsum += feature4[5] * -3.267303E-02;
netsum += feature4[6] *  .163733;
netsum += feature4[7] * -.1813052;
netsum += feature4[8] * -.4793789;
netsum += feature4[9] * -1.001629;
netsum += feature4[10] * -5.337683E-03;
netsum += feature4[11] *  .3546557;
netsum += feature4[12] *  .0130378;
netsum += feature4[13] * -5.230563E-02;
```

```
netsum += feature4[14] * -.224341;
netsum += feature4[15] *  3.420901E-02;
netsum += feature4[16] * -2.997049E-02;
outarray[16] = 1 / (1 + exp(-netsum));

netsum = -.0871721;
netsum += feature2[0] * -2.144521;
netsum += feature2[1] * -4.448839E-02;
netsum += feature2[2] * -.4005091;
netsum += feature2[3] *  .2491008;
netsum += feature2[4] *  .103228;
netsum += feature2[5] *  1.425065E-02;
netsum += feature2[6] * -8.274832E-02;
netsum += feature2[7] * -2.775137E-02;
netsum += feature2[8] *  .1452291;
netsum += feature2[9] *  .9049762;
netsum += feature2[10] *  .123446;
netsum += feature2[11] *  .0872754;
netsum += feature2[12] *  .2531824;
netsum += feature2[13] * -7.545377E-02;
netsum += feature2[14] *  5.535702E-02;
netsum += feature2[15] *  .1114788;
netsum += feature2[16] *  3.641256E-02;
netsum +=  1.273109E-02;
netsum += feature3[0] *  .1176406;
netsum += feature3[1] *  .1472907;
netsum += feature3[2] *  9.772182E-02;
netsum += feature3[3] * -.5695692;
netsum += feature3[4] *  .2763528;
netsum += feature3[5] *  .3965719;
netsum += feature3[6] *  .1523217;
netsum += feature3[7] *  .2915269;
netsum += feature3[8] * -3.761017E-03;
netsum += feature3[9] * -9.394163E-02;
netsum += feature3[10] *  5.442842E-02;
netsum += feature3[11] *  9.694701E-02;
netsum += feature3[12] * -1.407259E-03;
netsum += feature3[13] * -1.367245E-02;
netsum += feature3[14] * -.2875669;
netsum += feature3[15] * -1.588063;
netsum += feature3[16] * -.3411622;
netsum +=  .2417312;
netsum += feature4[0] *  .1546022;
netsum += feature4[1] *  2.580408E-02;
netsum += feature4[2] *  .031376;
netsum += feature4[3] *  .2905543;
netsum += feature4[4] * -.5831627;
netsum += feature4[5] * -.1149004;
netsum += feature4[6] * -2.733244;
netsum += feature4[7] *  4.435084E-02;
netsum += feature4[8] *  .3192606;
netsum += feature4[9] * -.3230639;
netsum += feature4[10] *  8.709101E-02;
netsum += feature4[11] *  .0749438;
netsum += feature4[12] *  .027183;
netsum += feature4[13] * -.102262;
netsum += feature4[14] *  .3980757;
netsum += feature4[15] * -4.982418E-02;
netsum += feature4[16] * -5.902603E-02;
outarray[17] = 1 / (1 + exp(-netsum));

netsum = -2.805811E-02;
netsum += feature2[0] *  3.957041;
netsum += feature2[1] *  .1062071;
netsum += feature2[2] * -2.364446;
netsum += feature2[3] * -.2364501;
netsum += feature2[4] *  .3139521;
netsum += feature2[5] * -4.669798E-02;
netsum += feature2[6] * -9.554782E-02;
netsum += feature2[7] * -1.011353E-02;
netsum += feature2[8] *  .1370771;
netsum += feature2[9] * -.3019075;
netsum += feature2[10] * -.1480832;
netsum += feature2[11] * -.1493765;
netsum += feature2[12] * -8.906589E-02;
netsum += feature2[13] *  .1797263;
netsum += feature2[14] * -8.091728E-02;
netsum += feature2[15] * -.2303406;
netsum += feature2[16] * -.2141423;
netsum += -.2279016;
netsum += feature3[0] *  .4686097;
netsum += feature3[1] * -.3230553;
netsum += feature3[2] *  .4287883;
netsum += feature3[3] *  .6006863;
netsum += feature3[4] * -.3818886;
netsum += feature3[5] * -.5317657;
netsum += feature3[6] *  1.215249;
netsum += feature3[7] *  8.081295E-02;
netsum += feature3[8] *  .0744914;
netsum += feature3[9] *  6.291211E-02;
netsum += feature3[10] *  9.514976E-02;
netsum += feature3[11] * -.3108238;
netsum += feature3[12] *  .3403691;
netsum += feature3[13] *  .6897222;
netsum += feature3[14] *  .5131958;
netsum += feature3[15] * -.3925945;
netsum += feature3[16] * -1.650232E-02;
netsum += -.1346832;
netsum += feature4[0] * -.7183999;
netsum += feature4[1] *  .1204374;
netsum += feature4[2] *  .3305629;
netsum += feature4[3] * -1.736894;
netsum += feature4[4] * -.6994048;
netsum += feature4[5] * -.0298544;
netsum += feature4[6] *  2.201084;
netsum += feature4[7] * -.3277493;
netsum += feature4[8] *  .5822;
netsum += feature4[9] * -.2156827;
netsum += feature4[10] * -5.861459E-02;
netsum += feature4[11] *  .5371926;
netsum += feature4[12] *  .5810279;
netsum += feature4[13] *  .362216;
netsum += feature4[14] * -.2359412;
netsum += feature4[15] *  8.054603E-02;
```

```
netsum += feature4[16] * -6.078743E-03;
outarray[18] = 1 / (1 + exp(-netsum));

netsum = -1.37511E-03;
netsum += feature2[0] * -1.397079;
netsum += feature2[1] * 4.353316E-02;
netsum += feature2[2] * 1.354434;
netsum += feature2[3] * .112401;
netsum += feature2[4] * -.0386434;
netsum += feature2[5] * -.0525167;
netsum += feature2[6] * .2819443;
netsum += feature2[7] * -4.181996E-02;
netsum += feature2[8] * 4.713617E-03;
netsum += feature2[9] * -.3199296;
netsum += feature2[10] * 8.967161E-03;
netsum += feature2[11] * -.1367796;
netsum += feature2[12] * .1250004;
netsum += feature2[13] * -1.055236E-03;
netsum += feature2[14] * .2258358;
netsum += feature2[15] * 3.306491E-02;
netsum += feature2[16] * .2901937;
netsum += -.1115558;
netsum += feature3[0] * .225426;
netsum += feature3[1] * 5.378804E-03;
netsum += feature3[2] * .1883165;
netsum += feature3[3] * 6.398771E-02;
netsum += feature3[4] * .1564406;
netsum += feature3[5] * -.1312269;
netsum += feature3[6] * -.5034187;
netsum += feature3[7] * .4080826;
netsum += feature3[8] * .16485;
netsum += feature3[9] * 9.232281E-02;
netsum += feature3[10] * -6.225928E-02;
netsum += feature3[11] * .0561798;
netsum += feature3[12] * 2.485161E-03;
netsum += feature3[13] * .4059957;
netsum += feature3[14] * .1164118;
netsum += feature3[15] * .4244484;
netsum += feature3[16] * -5.425857E-02;
netsum += -.1444986;
netsum += feature4[0] * .4156628;
netsum += feature4[1] * -.1344273;
netsum += feature4[2] * 3.217099E-02;
netsum += feature4[3] * -1.153962;
netsum += feature4[4] * -2.161218;
netsum += feature4[5] * .0654761;
netsum += feature4[6] * .884622;
netsum += feature4[7] * -.1459991;
netsum += feature4[8] * -.2948339;
netsum += feature4[9] * .7667853;
netsum += feature4[10] * -.0823068;
netsum += feature4[11] * -2.133677;
netsum += feature4[12] * .1566546;
netsum += feature4[13] * .1336686;
netsum += feature4[14] * -1.465682E-02;
netsum += feature4[15] * 3.405773E-02;
netsum += feature4[16] * .1771962;
outarray[19] = 1 / (1 + exp(-netsum));

netsum = -.681951;
netsum += feature2[0] * .4808471;
netsum += feature2[1] * -.1439364;
netsum += feature2[2] * -.1097401;
netsum += feature2[3] * -.6345702;
netsum += feature2[4] * -.1102759;
netsum += feature2[5] * 2.553549E-02;
netsum += feature2[6] * -.1441805;
netsum += feature2[7] * .1315049;
netsum += feature2[8] * 7.518549E-02;
netsum += feature2[9] * -.4540344;
netsum += feature2[10] * 5.622924E-02;
netsum += feature2[11] * .1960231;
netsum += feature2[12] * -.452196;
netsum += feature2[13] * .16326;
netsum += feature2[14] * -.1224413;
netsum += feature2[15] * -.1299815;
netsum += feature2[16] * -.6250322;
netsum += -.2930877;
netsum += feature3[0] * .3686935;
netsum += feature3[1] * -.4172523;
netsum += feature3[2] * .5096961;
netsum += feature3[3] * -1.218814;
netsum += feature3[4] * -.7585832;
netsum += feature3[5] * -1.282396;
netsum += feature3[6] * .7123765;
netsum += feature3[7] * -.2891739;
netsum += feature3[8] * .7622097;
netsum += feature3[9] * -6.598943E-02;
netsum += feature3[10] * -.0955903;
netsum += feature3[11] * -.8718266;
netsum += feature3[12] * .3169701;
netsum += feature3[13] * 4.59656E-03;
netsum += feature3[14] * .5634782;
netsum += feature3[15] * .7750067;
netsum += feature3[16] * .7139654;
netsum += -.5938213;
netsum += feature4[0] * -.1386441;
netsum += feature4[1] * .2336703;
netsum += feature4[2] * .170028;
netsum += feature4[3] * 3.42494;
netsum += feature4[4] * 2.15602;
netsum += feature4[5] * .2979617;
netsum += feature4[6] * -.4061868;
netsum += feature4[7] * -.3143424;
netsum += feature4[8] * .6261958;
netsum += feature4[9] * -.3730845;
netsum += feature4[10] * -.8561757;
netsum += feature4[11] * 2.166692;
netsum += feature4[12] * -.1522277;
netsum += feature4[13] * 5.447038E-02;
netsum += feature4[14] * -.408326;
netsum += feature4[15] * 9.023207E-02;
netsum += feature4[16] * -.5734635;
outarray[20] = 1 / (1 + exp(-netsum));
```

```
netsum = -.5055426;
netsum += feature2[0] * -.1817589;
netsum += feature2[1] * -.4879159;
netsum += feature2[2] * -.4768394;
netsum += feature2[3] * -.369628;
netsum += feature2[4] * .1058696;
netsum += feature2[5] * .2322319;
netsum += feature2[6] * -.1376621;
netsum += feature2[7] * -.1550036;
netsum += feature2[8] * -.3524822;
netsum += feature2[9] * -.4478379;
netsum += feature2[10] * -1.470395;
netsum += feature2[11] * .5487649;
netsum += feature2[12] * .8321803;
netsum += feature2[13] * 1.068443;
netsum += feature2[14] * 1.769008;
netsum += feature2[15] * .7999319;
netsum += feature2[16] * -.389156;
netsum += -.1361846;
netsum += feature3[0] * .424141;
netsum += feature3[1] * .5840187;
netsum += feature3[2] * .2895997;
netsum += feature3[3] * .678273;
netsum += feature3[4] * -.1292808;
netsum += feature3[5] * -.5418659;
netsum += feature3[6] * -1.77958;
netsum += feature3[7] * -.1901126;
netsum += feature3[8] * -.6237158;
netsum += feature3[9] * 5.385555E-02;
netsum += feature3[10] * 1.847559;
netsum += feature3[11] * -.3912142;
netsum += feature3[12] * -.4315381;
netsum += feature3[13] * -2.112628;
netsum += feature3[14] * -.6609244;
netsum += feature3[15] * 1.607197;
netsum += feature3[16] * .5736569;
netsum += -.5560333;
netsum += feature4[0] * .156609;
netsum += feature4[1] * .3055184;
netsum += feature4[2] * -9.773186E-02;
netsum += feature4[3] * -3.257928E-02;
netsum += feature4[4] * 1.349141;
netsum += feature4[5] * .8965985;
netsum += feature4[6] * -2.154003;
netsum += feature4[7] * -.6649521;
netsum += feature4[8] * .5775474;
netsum += feature4[9] * -.2540984;
netsum += feature4[10] * -.6735958;
netsum += feature4[11] * -.4604834;
netsum += feature4[12] * .4558707;
netsum += feature4[13] * .1931967;
netsum += feature4[14] * -.1337029;
netsum += feature4[15] * .4780812;
netsum += feature4[16] * -.2782983;
outarray[21] = 1 / (1 + exp(-netsum));

netsum = .3075594;
netsum += feature2[0] * -4.148044E-02;
netsum += feature2[1] * 1.467233;
netsum += feature2[2] * -5.765689E-02;
netsum += feature2[3] * .1510086;
netsum += feature2[4] * .7953346;
netsum += feature2[5] * -1.466597;
netsum += feature2[6] * -.3568816;
netsum += feature2[7] * 2.229546;
netsum += feature2[8] * -6.180795E-02;
netsum += feature2[9] * .1961231;
netsum += feature2[10] * .7758421;
netsum += feature2[11] * -.1587666;
netsum += feature2[12] * .1055093;
netsum += feature2[13] * -1.182917;
netsum += feature2[14] * -8.521164E-02;
netsum += feature2[15] * -.5445319;
netsum += feature2[16] * .4054246;
netsum += -4.177612E-02;
netsum += feature3[0] * -6.439663E-02;
netsum += feature3[1] * -.4106916;
netsum += feature3[2] * -.5439832;
netsum += feature3[3] * .1977865;
netsum += feature3[4] * .4768214;
netsum += feature3[5] * 1.29112;
netsum += feature3[6] * -.3362019;
netsum += feature3[7] * 5.095915E-02;
netsum += feature3[8] * .6036866;
netsum += feature3[9] * .1402611;
netsum += feature3[10] * -1.759524E-02;
netsum += feature3[11] * .2405592;
netsum += feature3[12] * -1.308243;
netsum += feature3[13] * .0476276;
netsum += feature3[14] * -7.494947E-02;
netsum += feature3[15] * -.1586238;
netsum += feature3[16] * -.5840786;
netsum += .4444339;
netsum += feature4[0] * .2649181;
netsum += feature4[1] * -1.063338;
netsum += feature4[2] * -1.173332;
netsum += feature4[3] * -.1687374;
netsum += feature4[4] * -.476624;
netsum += feature4[5] * -4.052002;
netsum += feature4[6] * -7.380632E-02;
netsum += feature4[7] * .1521398;
netsum += feature4[8] * -1.630774;
netsum += feature4[9] * 1.61272;
netsum += feature4[10] * -4.269277E-02;
netsum += feature4[11] * -.1458386;
netsum += feature4[12] * 4.282573E-02;
netsum += feature4[13] * -.2015104;
netsum += feature4[14] * .3126151;
netsum += feature4[15] * .171232;
netsum += feature4[16] * .4980499;
outarray[22] = 1 / (1 + exp(-netsum));

netsum = -.1764495;
```

```
netsum += feature2[0] * -2.315889E-03;
netsum += feature2[1] *  .511911;
netsum += feature2[2] * -.281034;
netsum += feature2[3] * -.4831658;
netsum += feature2[4] *  .1952543;
netsum += feature2[5] *  .6939362;
netsum += feature2[6] * -2.475118;
netsum += feature2[7] * -.4134085;
netsum += feature2[8] *  .6121315;
netsum += feature2[9] * -.120023;
netsum += feature2[10] *  1.61212;
netsum += feature2[11] *  .1620891;
netsum += feature2[12] *  .4765576;
netsum += feature2[13] * -.726401;
netsum += feature2[14] *  .3176222;
netsum += feature2[15] * -.5044598;
netsum += feature2[16] * -.105497;
netsum += -.4033448;
netsum += feature3[0] *  .3576689;
netsum += feature3[1] * -6.330287E-02;
netsum += feature3[2] *  .5950831;
netsum += feature3[3] * -2.814957E-02;
netsum += feature3[4] * -.1106676;
netsum += feature3[5] * -.0173098;
netsum += feature3[6] *  2.116163E-02;
netsum += feature3[7] * -.4018261;
netsum += feature3[8] *  3.343648E-02;
netsum += feature3[9] * -.2178237;
netsum += feature3[10] * -.6468236;
netsum += feature3[11] * -.1000675;
netsum += feature3[12] *  .4594381;
netsum += feature3[13] * -.1677051;
netsum += feature3[14] *  .3968353;
netsum += feature3[15] * -.2591742;
netsum += feature3[16] *  .3980177;
netsum += -.6262275;
netsum += feature4[0] *  .2318635;
netsum += feature4[1] *  1.554979;
netsum += feature4[2] * -.2575847;
netsum += feature4[3] *  .1001018;
netsum += feature4[4] * -.3627397;
netsum += feature4[5] *  3.032024;
netsum += feature4[6] *  .1275648;
netsum += feature4[7] * -.5528494;
netsum += feature4[8] * -.2861452;
netsum += feature4[9] * -.9421886;
netsum += feature4[10] * -.5175532;
netsum += feature4[11] *  .8692129;
netsum += feature4[12] *  7.942677E-03;
netsum += feature4[13] *  .1521529;
netsum += feature4[14] * -.4630744;
netsum += feature4[15] *  .5146495;
netsum += feature4[16] * -.4543906;
outarray[23] = 1 / (1 + exp(-netsum));

netsum =  .2985819;
netsum += feature2[0] *  7.546861E-02;
netsum += feature2[1] * -.9057129;
netsum += feature2[2] *  3.876311E-03;
netsum += feature2[3] *  .2811382;
netsum += feature2[4] * -.6783946;
netsum += feature2[5] *  1.360962;
netsum += feature2[6] * -1.008376;
netsum += feature2[7] * -1.373084;
netsum += feature2[8] *  .1773863;
netsum += feature2[9] *  5.579031E-02;
netsum += feature2[10] *  .4605159;
netsum += feature2[11] *  .210614;
netsum += feature2[12] *  .2915196;
netsum += feature2[13] * -.9950874;
netsum += feature2[14] *  4.295753E-02;
netsum += feature2[15] *  1.386924;
netsum += feature2[16] *  .2423063;
netsum +=  .3007702;
netsum += feature3[0] *  4.357998E-02;
netsum += feature3[1] *  .2830358;
netsum += feature3[2] * -8.805215E-02;
netsum += feature3[3] *  .400731;
netsum += feature3[4] * -.1420603;
netsum += feature3[5] *  .3751267;
netsum += feature3[6] * -.3823563;
netsum += feature3[7] * -4.137364E-02;
netsum += feature3[8] * -.7014194;
netsum += feature3[9] *  .3337325;
netsum += feature3[10] *  1.677879E-02;
netsum += feature3[11] *  5.432534E-02;
netsum += feature3[12] *  .6310396;
netsum += feature3[13] * -1.630619E-02;
netsum += feature3[14] *  .7130215;
netsum += feature3[15] * -.1527234;
netsum += feature3[16] *  .4473297;
netsum +=  .2130702;
netsum += feature4[0] * -1.575439;
netsum += feature4[1] * -1.607305;
netsum += feature4[2] *  .6066403;
netsum += feature4[3] *  7.200585E-02;
netsum += feature4[4] * -.2672594;
netsum += feature4[5] * -7.430886E-02;
netsum += feature4[6] *  .1167742;
netsum += feature4[7] *  .3657437;
netsum += feature4[8] *  .3037682;
netsum += feature4[9] *  .4942872;
netsum += feature4[10] *  5.389816E-02;
netsum += feature4[11] * -.9378299;
netsum += feature4[12] *  1.085573E-02;
netsum += feature4[13] *  .1983645;
netsum += feature4[14] *  .1856362;
netsum += feature4[15] * -1.774012;
netsum += feature4[16] *  .5004287;
outarray[24] = 1 / (1 + exp(-netsum));

netsum = -.4102101;
netsum += feature2[0] * -.1192021;
netsum += feature2[1] * -.5048786;
```

```
netsum += feature2[2] *  .1432994;
netsum += feature2[3] * -.8456954;
netsum += feature2[4] * -.1182771;
netsum += feature2[5] * -.3290169;
netsum += feature2[6] *  2.946723;
netsum += feature2[7] * -.5806878;
netsum += feature2[8] * -.3442121;
netsum += feature2[9] * -.476873;
netsum += feature2[10] * -1.094144;
netsum += feature2[11] * -.2876037;
netsum += feature2[12] * -.4320113;
netsum += feature2[13] *  .8707688;
netsum += feature2[14] * -.2656275;
netsum += feature2[15] * -.9086761;
netsum += feature2[16] * -.1496554;
netsum += -.1711361;
netsum += feature3[0] *  .3388088;
netsum += feature3[1] * -.2208434;
netsum += feature3[2] *  .3091679;
netsum += feature3[3] * -.427543;
netsum += feature3[4] * -.3852794;
netsum += feature3[5] *  .8147128;
netsum += feature3[6] *  .801157;
netsum += feature3[7] * -.5965335;
netsum += feature3[8] *  .1456866;
netsum += feature3[9] * -.1861839;
netsum += feature3[10] *  .339537;
netsum += feature3[11] * -.1625948;
netsum += feature3[12] *  .5502238;
netsum += feature3[13] *  .2549001;
netsum += feature3[14] * -.9382777;
netsum += feature3[15] *  .5397156;
netsum += feature3[16] *  .2440248;
netsum += -.1928512;
netsum += feature4[0] *  1.303436;
netsum += feature4[1] *  1.625236;
netsum += feature4[2] *  .4646503;
netsum += feature4[3] * -.2744274;
netsum += feature4[4] *  .8675063;
netsum += feature4[5] * -.5449609;
netsum += feature4[6] * -.1387964;
netsum += feature4[7] * -.3350293;
netsum += feature4[8] *  1.76642;
netsum += feature4[9] * -.2384665;
netsum += feature4[10] * -.4533813;
netsum += feature4[11] *  .1591108;
netsum += feature4[12] *  .0488232;
netsum += feature4[13] * -.2083472;
netsum += feature4[14] * -.5085511;
netsum += feature4[15] *  1.092933;
netsum += feature4[16] * -.4151816;
outarray[25] = 1 / (1 + exp(-netsum));

netsum = -9.454906E-02;
netsum += feature2[0] * -1.223662;
netsum += feature2[1] * -.2192892;
netsum += feature2[2] *  2.508044;
netsum += feature2[3] * -.2739756;
netsum += feature2[4] * -.7211137;
netsum += feature2[5] * -.2452639;
netsum += feature2[6] *  .2772861;
netsum += feature2[7] * -.3912483;
netsum += feature2[8] * -.1186267;
netsum += feature2[9] * -.1491327;
netsum += feature2[10] * -.2547779;
netsum += feature2[11] * -.8408157;
netsum += feature2[12] * -1.028467;
netsum += feature2[13] *  .5868462;
netsum += feature2[14] * -1.696559;
netsum += feature2[15] * -.2052399;
netsum += feature2[16] * -.489409;
netsum += -7.758802E-02;
netsum += feature3[0] * -6.904944E-03;
netsum += feature3[1] *  .3012756;
netsum += feature3[2] * -8.863187E-02;
netsum += feature3[3] *  .1905311;
netsum += feature3[4] * -4.198126E-02;
netsum += feature3[5] *  .1286711;
netsum += feature3[6] *  .358032;
netsum += feature3[7] *  .1572804;
netsum += feature3[8] * -.602511;
netsum += feature3[9] * -.2589206;
netsum += feature3[10] * -1.476484;
netsum += feature3[11] * -.1961623;
netsum += feature3[12] * -.7567646;
netsum += feature3[13] *  1.287455;
netsum += feature3[14] * -.4404136;
netsum += feature3[15] * -.6440281;
netsum += feature3[16] * -6.342192E-02;
netsum += -.1985789;
netsum += feature4[0] *  .1878645;
netsum += feature4[1] * -.7406157;
netsum += feature4[2] * -.9226816;
netsum += feature4[3] * -.1504055;
netsum += feature4[4] *  .8869594;
netsum += feature4[5] *  1.51085;
netsum += feature4[6] *  1.651823;
netsum += feature4[7] *  9.588923E-02;
netsum += feature4[8] * -.1107868;
netsum += feature4[9] *  .7057498;
netsum += feature4[10] *  8.329659E-04;
netsum += feature4[11] * -.2957856;
netsum += feature4[12] * -.813566;
netsum += feature4[13] * -.5973057;
netsum += feature4[14] *  4.651726E-02;
netsum += feature4[15] * -.5298412;
netsum += feature4[16] * -.1787528;
outarray[26] = 1 / (1 + exp(-netsum));

outarray[0] =  4563.816 *  (outarray[0] - .1) /
.8  +  41.44737;
if (outarray[0]< 41.44737) outarray[0] =
41.44737;
```

```
if (outarray[0]> 4605.263) outarray[0] =
4605.263;

outarray[1] = 3404.606 * (outarray[1] - .1) /
.8 + 30.91974;
if (outarray[1]< 30.91974) outarray[1] =
30.91974;
if (outarray[1]> 3435.526) outarray[1] =
3435.526;

outarray[2] = 5215.79 * (outarray[2] - .1) / .8
+ 47.36842;
if (outarray[2]< 47.36842) outarray[2] =
47.36842;
if (outarray[2]> 5263.158) outarray[2] =
5263.158;

outarray[3] = 3890.979 * (outarray[3] - .1) /
.8 + 35.33684;
if (outarray[3]< 35.33684) outarray[3] =
35.33684;
if (outarray[3]> 3926.316) outarray[3] =
3926.316;

outarray[4] = 5867.764 * (outarray[4] - .1) /
.8 + 53.28947;
if (outarray[4]< 53.28947) outarray[4] =
53.28947;
if (outarray[4]> 5921.053) outarray[4] =
5921.053;

outarray[5] = 4377.351 * (outarray[5] - .1) /
.8 + 39.75395;
if (outarray[5]< 39.75395) outarray[5] =
39.75395;
if (outarray[5]> 4417.105) outarray[5] =
4417.105;

outarray[6] = 5541.776 * (outarray[6] - .1) /
.8 + 50.32895;
if (outarray[6]< 50.32895) outarray[6] =
50.32895;
if (outarray[6]> 5592.105) outarray[6] =
5592.105;

outarray[7] = 4134.165 * (outarray[7] - .1) /
.8 + 37.54539;
if (outarray[7]< 37.54539) outarray[7] =
37.54539;
if (outarray[7]> 4171.71) outarray[7] =
4171.71;

outarray[8] = 4235.501 * (outarray[8] - .1) /
.8 + 38.4657;
if (outarray[8]< 38.4657) outarray[8] =
38.4657;
if (outarray[8]> 4273.967) outarray[8] =
4273.967;

outarray[9] = 3159.684 * (outarray[9] - .1) /
.8 + 28.69541;
if (outarray[9]< 28.69541) outarray[9] =
28.69541;
if (outarray[9]> 3188.379) outarray[9] =
3188.379;

outarray[10] = 2154.011 * (outarray[10] - .1)
/ .8 + 10.02519;
if (outarray[10]< 10.02519) outarray[10] =
10.02519;
if (outarray[10]> 2164.036) outarray[10] =
2164.036;

outarray[11] = 1523.115 * (outarray[11] - .1)
/ .8 + 7.088883;
if (outarray[11]< 7.088883) outarray[11] =
7.088883;
if (outarray[11]> 1530.204) outarray[11] =
1530.204;

outarray[12] = 1077.005 * (outarray[12] - .1)
/ .8 + 5.012597;
if (outarray[12]< 5.012597) outarray[12] =
5.012597;
if (outarray[12]> 1082.018) outarray[12] =
1082.018;

outarray[13] = 2302.734 * (outarray[13] - .1)
/ .8 + 10.71738;
if (outarray[13]< 10.71738) outarray[13] =
10.71738;
if (outarray[13]> 2313.451) outarray[13] =
2313.451;

outarray[14] = 2442.418 * (outarray[14] - .1)
/ .8 + 11.3675;
if (outarray[14]< 11.3675) outarray[14] =
11.3675;
if (outarray[14]> 2453.786) outarray[14] =
2453.786;

outarray[15] = 2373.604 * (outarray[15] - .1)
/ .8 + 11.04723;
if (outarray[15]< 11.04723) outarray[15] =
11.04723;
if (outarray[15]> 2384.651) outarray[15] =
2384.651;

outarray[16] = 2075.086 * (outarray[16] - .1)
/ .8 + 9.657866;
if (outarray[16]< 9.657866) outarray[16] =
9.657866;
```

```
if (outarray[16]> 2084.744) outarray[16] =  2084.744;

outarray[17] =  10 *  (outarray[17] - .1) / .8 ;
if (outarray[17]< 0) outarray[17] =  0;
if (outarray[17]> 10) outarray[17] =  10;

outarray[18] =  10 *  (outarray[18] - .1) / .8 ;
if (outarray[18]< 0) outarray[18] =  0;
if (outarray[18]> 10) outarray[18] =  10;

outarray[19] =  10 *  (outarray[19] - .1) / .8 ;
if (outarray[19]< 0) outarray[19] =  0;
if (outarray[19]> 10) outarray[19] =  10;

outarray[20] =  10 *  (outarray[20] - .1) / .8 ;
if (outarray[20]< 0) outarray[20] =  0;
if (outarray[20]> 10) outarray[20] =  10;

outarray[21] =  10 *  (outarray[21] - .1) / .8 ;
if (outarray[21]< 0) outarray[21] =  0;
if (outarray[21]> 10) outarray[21] =  10;

outarray[22] =  10 *  (outarray[22] - .1) / .8 ;
if (outarray[22]< 0) outarray[22] =  0;
if (outarray[22]> 10) outarray[22] =  10;

outarray[23] =  10 *  (outarray[23] - .1) / .8 ;
if (outarray[23]< 0) outarray[23] =  0;
if (outarray[23]> 10) outarray[23] =  10;

outarray[24] =  10 *  (outarray[24] - .1) / .8 ;
if (outarray[24]< 0) outarray[24] =  0;
if (outarray[24]> 10) outarray[24] =  10;

outarray[25] =  10 *  (outarray[25] - .1) / .8 ;
if (outarray[25]< 0) outarray[25] =  0;
if (outarray[25]> 10) outarray[25] =  10;

outarray[26] =  10 *  (outarray[26] - .1) / .8 ;
if (outarray[26]< 0) outarray[26] =  0;
if (outarray[26]> 10) outarray[26] =  10;
}
```

B.1.1 SELECCIÓN DE TURBINAS: MÉTODO 3 Y 4.

```
/* Insert this code into your C program to fire the C:\NSHELL2\TURBINA2\FRANCIS\FRANCIS network */
/* This code is designed to be simple and fast for porting to any machine */
/* Therefore all code and weights are inline without looping or data storage */
/*   which might be harder to port between compilers. */

#include <math.h>
void Fire_FRANCIS(double *inarray, double *outarray)
{
 double netsum;
 double feature2[19];
 double feature3[19];
 double feature4[19];
```

```
/* inarray[0] is H_m */
/* inarray[1] is Q_m3_seg */
/* inarray[2] is Vel_Rot.n_rpm */
/* inarray[3] is Ef_Francis */
/* inarray[4] is Kc3FL */
/* inarray[5] is Kc3FN */
/* inarray[6] is Kc3FR */
/* inarray[7] is Kc3FER */
/* outarray[0] is P_Francis_hp */
/* outarray[1] is P_Francis_kW */
/* outarray[2] is Ns_Francis */
/* outarray[3] is Nº_Polos */
/* outarray[4] is Vsal_FLC3_m/s */
/* outarray[5] is Vsal_FNC3_m/s */
/* outarray[6] is Vsal_FRC3_m/s */
/* outarray[7] is Vsal_FERC3_m/s */
/* outarray[8] is D3TaspFL_mm */
/* outarray[9] is D3TaspFN_mm */
/* outarray[10] is D3TaspFR_mm */
/* outarray[11] is D3TaspFER_mm */
/* outarray[12] is Vtg_FL_m/s */
/* outarray[13] is Vtg_FN_m/s */
/* outarray[14] is Vtg_FR_m/s */
/* outarray[15] is Vtg_FER_m/s */
/* outarray[16] is Drod_D1FL_mm */
/* outarray[17] is Drod_D1FN_mm */
/* outarray[18] is Drod_D1FR_mm */
/* outarray[19] is Drod_D1FER_mm */
/* outarray[20] is Nrod_FL_rpm */
/* outarray[21] is Nrod_FN_rpm */
/* outarray[22] is Nrod_FR_rpm */
/* outarray[23] is Nrod_FER_rpm */
/* outarray[24] is D2cor_FL_mm */
/* outarray[25] is D2cor_FN_mm */
/* outarray[26] is D2cor_FR_mm */
/* outarray[27] is D2cor_FER_mm */
/* outarray[28] is Nal_ZL */
/* outarray[29] is Nal_ZN */
/* outarray[30] is Nal_ZR */
/* outarray[31] is Nal_ZER */
/* outarray[32] is AnCdFL_Bo_mm */
/* outarray[33] is AnCdFN_Bo_mm */
/* outarray[34] is AnCdFR_Bo_mm */
/* outarray[35] is AnCdFER_Bo_mm */
/* outarray[36] is FLenta */
/* outarray[37] is FNormal */
/* outarray[38] is FRapida */
/* outarray[39] is FExRapida */

if (inarray[0]< 3) inarray[0] = 3;
if (inarray[0]> 300) inarray[0] = 300;
inarray[0] = 2 * (inarray[0] - 3) / 297 -1;

if (inarray[1]< .03) inarray[1] = .03;
if (inarray[1]> 25) inarray[1] = 25;
inarray[1] = 2 * (inarray[1] - .03) / 24.97 -1;

if (inarray[2]< 225) inarray[2] = 225;
if (inarray[2]> 2500) inarray[2] = 2500;
inarray[2] = 2 * (inarray[2] - 225) / 2275 -1;

if (inarray[3]< 80) inarray[3] = 80;
if (inarray[3]> 81) inarray[3] = 81;
inarray[3] = 2 * (inarray[3] - 80) -1;

if (inarray[4]< 4) inarray[4] = 4;
if (inarray[4]> 5) inarray[4] = 5;
inarray[4] = 2 * (inarray[4] - 4) -1;

if (inarray[5]< 6) inarray[5] = 6;
if (inarray[5]> 7) inarray[5] = 7;
inarray[5] = 2 * (inarray[5] - 6) -1;

if (inarray[6]< 12) inarray[6] = 12;
if (inarray[6]> 13) inarray[6] = 13;
inarray[6] = 2 * (inarray[6] - 12) -1;

if (inarray[7]< 25) inarray[7] = 25;
if (inarray[7]> 26) inarray[7] = 26;
inarray[7] = 2 * (inarray[7] - 25) -1;

netsum = -5.478104;
netsum += inarray[0] * -13.79763;
netsum += inarray[1] * -21.7597;
netsum += inarray[2] * 1.22125;
netsum += inarray[3] * 5.153677;
netsum += inarray[4] * 5.273217;
netsum += inarray[5] * 4.993727;
netsum += inarray[6] * 5.276625;
netsum += inarray[7] * 5.382453;
feature2[0] = exp(-netsum * netsum);

netsum = -3.787917;
netsum += inarray[0] * -2.645108;
netsum += inarray[1] * -22.67536;
netsum += inarray[2] * -.5008544;
netsum += inarray[3] * 4.095648;
netsum += inarray[4] * 3.821211;
netsum += inarray[5] * 3.632778;
netsum += inarray[6] * 3.781802;
netsum += inarray[7] * 3.76464;
feature2[1] = exp(-netsum * netsum);

netsum = -1.909063;
netsum += inarray[0] * -29.20437;
netsum += inarray[1] * 15.10171;
netsum += inarray[2] * 4.505999;
netsum += inarray[3] * 1.871705;
netsum += inarray[4] * 1.462958;
netsum += inarray[5] * 1.915365;
netsum += inarray[6] * 1.487649;
netsum += inarray[7] * 1.725582;
feature2[2] = exp(-netsum * netsum);
```

```
netsum = -.1122053;
netsum += inarray[0] * -10.61923;
netsum += inarray[1] *  5.953619;
netsum += inarray[2] *  1.288514;
netsum += inarray[3] *  6.236722E-02;
netsum += inarray[4] *  .4172336;
netsum += inarray[5] *  .2826306;
netsum += inarray[6] *  .5669273;
netsum += inarray[7] *  .5314485;
feature2[3] = exp(-netsum * netsum);

netsum =  .9627185;
netsum += inarray[0] * -2.840932;
netsum += inarray[1] *  8.102728;
netsum += inarray[2] *  .8753229;
netsum += inarray[3] * -.8522743;
netsum += inarray[4] * -.9088507;
netsum += inarray[5] * -1.118157;
netsum += inarray[6] * -.6588864;
netsum += inarray[7] * -1.194974;
feature2[4] = exp(-netsum * netsum);

netsum = -4.088692;
netsum += inarray[0] * -44.44529;
netsum += inarray[1] *  15.8206;
netsum += inarray[2] *  5.968985;
netsum += inarray[3] *  4.007732;
netsum += inarray[4] *  3.865179;
netsum += inarray[5] *  4.079013;
netsum += inarray[6] *  4.106398;
netsum += inarray[7] *  4.37394;
feature2[5] = exp(-netsum * netsum);

netsum =  1.812512E-02;
netsum += inarray[0] *  .1334502;
netsum += inarray[1] * -1.440658E-02;
netsum += inarray[2] * -8.605251E-02;
netsum += inarray[3] *  3.312322E-02;
netsum += inarray[4] *  .1620957;
netsum += inarray[5] *  3.132826E-02;
netsum += inarray[6] * -.2000631;
netsum += inarray[7] * -6.432273E-02;
feature2[6] = exp(-netsum * netsum);

netsum =  2.185077E-02;
netsum += inarray[0] *  6.084363;
netsum += inarray[1] * -3.080244;
netsum += inarray[2] * -4.166409;
netsum += inarray[3] *  .2810207;
netsum += inarray[4] *  6.904722E-04;
netsum += inarray[5] *  .1590608;
netsum += inarray[6] *  .1986919;
netsum += inarray[7] * -5.396901E-02;
feature2[7] = exp(-netsum * netsum);

netsum =  .1776088;
netsum += inarray[0] * -8.766216E-04;
netsum += inarray[1] * -1.238065E-04;
netsum += inarray[2] *  1.029209E-04;
netsum += inarray[3] *  5.955117E-02;
netsum += inarray[4] *  .2590413;
netsum += inarray[5] * -.1161412;
netsum += inarray[6] * -5.890157E-02;
netsum += inarray[7] *  3.490452E-02;
feature2[8] = exp(-netsum * netsum);

netsum = -.4072317;
netsum += inarray[0] *  1.763618;
netsum += inarray[1] * -3.511988;
netsum += inarray[2] * -2.245776;
netsum += inarray[3] *  .4720059;
netsum += inarray[4] *  .6710939;
netsum += inarray[5] *  .5965686;
netsum += inarray[6] *  .7635241;
netsum += inarray[7] *  .3716368;
feature2[9] = exp(-netsum * netsum);

netsum =  .1665241;
netsum += inarray[0] *  2.232694;
netsum += inarray[1] * -1.581779;
netsum += inarray[2] *  1.456446;
netsum += inarray[3] * -.5711281;
netsum += inarray[4] * -.4802418;
netsum += inarray[5] * -.2527362;
netsum += inarray[6] * -.4719571;
netsum += inarray[7] * -.1562285;
feature2[10] = exp(-netsum * netsum);

netsum = -1.569384;
netsum += inarray[0] * -48.89923;
netsum += inarray[1] *  34.98304;
netsum += inarray[2] *  4.571926;
netsum += inarray[3] *  1.619826;
netsum += inarray[4] *  1.740726;
netsum += inarray[5] *  1.938305;
netsum += inarray[6] *  1.758449;
netsum += inarray[7] *  1.860844;
feature2[11] = exp(-netsum * netsum);

netsum =  .6666138;
netsum += inarray[0] *  4.12354;
netsum += inarray[1] *  .4287938;
netsum += inarray[2] * -.3275603;
netsum += inarray[3] * -.5855017;
netsum += inarray[4] * -.7980617;
netsum += inarray[5] * -.4074121;
netsum += inarray[6] * -.6772136;
netsum += inarray[7] * -.4459397;
feature2[12] = exp(-netsum * netsum);

netsum =  .1949701;
netsum += inarray[0] *  .531792;
netsum += inarray[1] *  8.330692E-02;
netsum += inarray[2] * -5.105321E-04;
netsum += inarray[3] * -5.122166E-03;
netsum += inarray[4] *  9.313379E-02;
```

```
netsum += inarray[5] * -.2016728;
netsum += inarray[6] *  .1029482;
netsum += inarray[7] * -.368086;
feature2[13] = exp(-netsum * netsum);

netsum =  .1721914;
netsum += inarray[0] * -5.493276;
netsum += inarray[1] *  7.592573;
netsum += inarray[2] * -.2441396;
netsum += inarray[3] * -.2796408;
netsum += inarray[4] * -.1798892;
netsum += inarray[5] * -.5162085;
netsum += inarray[6] * -.4113888;
netsum += inarray[7] * -8.455894E-02;
feature2[14] = exp(-netsum * netsum);

netsum =  .6720135;
netsum += inarray[0] *  1.03775;
netsum += inarray[1] *  7.080067;
netsum += inarray[2] * -.2626191;
netsum += inarray[3] * -.6690389;
netsum += inarray[4] * -1.17105;
netsum += inarray[5] * -1.042261;
netsum += inarray[6] * -.8195415;
netsum += inarray[7] * -1.188978;
feature2[15] = exp(-netsum * netsum);

netsum = -1.328141;
netsum += inarray[0] * -18.80286;
netsum += inarray[1] *  7.036247;
netsum += inarray[2] *  5.918089;
netsum += inarray[3] *  1.234059;
netsum += inarray[4] *  1.309373;
netsum += inarray[5] *  1.283815;
netsum += inarray[6] *  1.12374;
netsum += inarray[7] *  .8494778;
feature2[16] = exp(-netsum * netsum);

netsum =  1.121457E-03;
netsum += inarray[0] *  1.01565;
netsum += inarray[1] * -.7234788;
netsum += inarray[2] *  .2359582;
netsum += inarray[3] * -.1751456;
netsum += inarray[4] *  .1105588;
netsum += inarray[5] * -.3614156;
netsum += inarray[6] * -.2648728;
netsum += inarray[7] *  6.342434E-02;
feature2[17] = exp(-netsum * netsum);

netsum = -.2849136;
netsum += inarray[0] * -1.482265;
netsum += inarray[1] *  .6539246;
netsum += inarray[2] *  .5185043;
netsum += inarray[3] * -.1253357;
netsum += inarray[4] *  .1224783;
netsum += inarray[5] *  8.346537E-02;
netsum += inarray[6] *  .2301015;
netsum += inarray[7] * -3.830885E-02;
feature2[18] = exp(-netsum * netsum);

netsum = -.3034792;
netsum += inarray[0] *  1.116319E-02;
netsum += inarray[1] *  1.200362;
netsum += inarray[2] *  .5281821;
netsum += inarray[3] *  .4205438;
netsum += inarray[4] *  .3979477;
netsum += inarray[5] *  .3963362;
netsum += inarray[6] *  .3777692;
netsum += inarray[7] *  .3801495;
feature3[0] = tanh(netsum);

netsum =  .8738331;
netsum += inarray[0] * -.4889644;
netsum += inarray[1] * -.1850564;
netsum += inarray[2] *  5.455552;
netsum += inarray[3] * -1.217658;
netsum += inarray[4] * -1.03809;
netsum += inarray[5] * -.700538;
netsum += inarray[6] * -.7932075;
netsum += inarray[7] * -.8489037;
feature3[1] = tanh(netsum);

netsum =  2.378962;
netsum += inarray[0] *  8.759796;
netsum += inarray[1] *  5.951619;
netsum += inarray[2] * -1.154179;
netsum += inarray[3] * -2.226026;
netsum += inarray[4] * -2.366457;
netsum += inarray[5] * -2.422383;
netsum += inarray[6] * -2.607682;
netsum += inarray[7] * -2.331521;
feature3[2] = tanh(netsum);

netsum = -4.676708;
netsum += inarray[0] * -28.51575;
netsum += inarray[1] *  1.370674;
netsum += inarray[2] *  2.287251;
netsum += inarray[3] *  4.378609;
netsum += inarray[4] *  4.703241;
netsum += inarray[5] *  4.242495;
netsum += inarray[6] *  4.41921;
netsum += inarray[7] *  4.267383;
feature3[3] = tanh(netsum);

netsum = -4.16978;
netsum += inarray[0] * -22.65795;
netsum += inarray[1] * -1.286685;
netsum += inarray[2] * -1.812115;
netsum += inarray[3] *  3.965956;
netsum += inarray[4] *  4.478365;
netsum += inarray[5] *  4.339801;
netsum += inarray[6] *  3.973463;
netsum += inarray[7] *  4.46337;
feature3[4] = tanh(netsum);
```

```
netsum = -1.056862;
netsum += inarray[0] * -1.10875;
netsum += inarray[1] * -5.972617;
netsum += inarray[2] * -.1079557;
netsum += inarray[3] *  .594629;
netsum += inarray[4] *  .8455369;
netsum += inarray[5] *  .7934816;
netsum += inarray[6] *  .4615737;
netsum += inarray[7] *  .7334737;
feature3[5] = tanh(netsum);

netsum = -8.837726E-02;
netsum += inarray[0] * -5.728523;
netsum += inarray[1] * -5.791149;
netsum += inarray[2] *  9.669379E-02;
netsum += inarray[3] *  .200502;
netsum += inarray[4] *  8.616171E-02;
netsum += inarray[5] *  .3239483;
netsum += inarray[6] *  .39864;
netsum += inarray[7] *  .3985489;
feature3[6] = tanh(netsum);

netsum =  1.223111;
netsum += inarray[0] *  1.821194;
netsum += inarray[1] * -.2163793;
netsum += inarray[2] *  4.604093;
netsum += inarray[3] * -1.275443;
netsum += inarray[4] * -.8806974;
netsum += inarray[5] * -1.115154;
netsum += inarray[6] * -1.169112;
netsum += inarray[7] * -1.15845;
feature3[7] = tanh(netsum);

netsum = -.7213756;
netsum += inarray[0] * -2.827211;
netsum += inarray[1] * -6.014158;
netsum += inarray[2] * -3.996637E-02;
netsum += inarray[3] *  .8046387;
netsum += inarray[4] *  .4759661;
netsum += inarray[5] *  .2632383;
netsum += inarray[6] *  .794503;
netsum += inarray[7] *  .2620489;
feature3[8] = tanh(netsum);

netsum = -2.252481;
netsum += inarray[0] * -9.394224;
netsum += inarray[1] * -2.647905;
netsum += inarray[2] *  9.234291E-02;
netsum += inarray[3] *  2.045324;
netsum += inarray[4] *  1.964067;
netsum += inarray[5] *  2.263806;
netsum += inarray[6] *  2.021521;
netsum += inarray[7] *  2.030105;
feature3[9] = tanh(netsum);

netsum = -4.631982;
netsum += inarray[0] * -4.033898;
netsum += inarray[1] * -22.65215;
netsum += inarray[2] *  4.37736;
netsum += inarray[3] *  4.214149;
netsum += inarray[4] *  4.091879;
netsum += inarray[5] *  4.153566;
netsum += inarray[6] *  4.527131;
netsum += inarray[7] *  4.442386;
feature3[10] = tanh(netsum);

netsum = -.4483488;
netsum += inarray[0] * -6.363194;
netsum += inarray[1] *  4.810481;
netsum += inarray[2] *  2.551845;
netsum += inarray[3] *  .7033749;
netsum += inarray[4] *  .5874906;
netsum += inarray[5] *  .4111943;
netsum += inarray[6] *  .2810183;
netsum += inarray[7] *  .4930439;
feature3[11] = tanh(netsum);

netsum = -3.219392;
netsum += inarray[0] * -.1541415;
netsum += inarray[1] * -17.32107;
netsum += inarray[2] *  1.266204;
netsum += inarray[3] *  3.182918;
netsum += inarray[4] *  2.878112;
netsum += inarray[5] *  2.814409;
netsum += inarray[6] *  3.017715;
netsum += inarray[7] *  2.869546;
feature3[12] = tanh(netsum);

netsum =  .2932296;
netsum += inarray[0] * -.7260727;
netsum += inarray[1] * -.4058585;
netsum += inarray[2] * -.0879497;
netsum += inarray[3] * -.5574953;
netsum += inarray[4] * -.613674;
netsum += inarray[5] * -.2766995;
netsum += inarray[6] * -.671043;
netsum += inarray[7] * -.668974;
feature3[13] = tanh(netsum);

netsum =  .7528782;
netsum += inarray[0] * -.9045894;
netsum += inarray[1] *  .8821045;
netsum += inarray[2] * -3.941496;
netsum += inarray[3] * -.9149609;
netsum += inarray[4] * -.7391471;
netsum += inarray[5] * -.5961426;
netsum += inarray[6] * -.5230899;
netsum += inarray[7] * -.7608724;
feature3[14] = tanh(netsum);

netsum = -.2354967;
netsum += inarray[0] *  .154537;
netsum += inarray[1] *  .5740142;
netsum += inarray[2] *  .7307581;
```

```
netsum += inarray[3] *  .5772225;
netsum += inarray[4] *  .6539126;
netsum += inarray[5] *  .6897922;
netsum += inarray[6] *  .6959427;
netsum += inarray[7] *  .4725034;
feature3[15] = tanh(netsum);

netsum =  1.81432;
netsum += inarray[0] * -1.96891;
netsum += inarray[1] *  12.02656;
netsum += inarray[2] * -.2083613;
netsum += inarray[3] * -2.015613;
netsum += inarray[4] * -2.031635;
netsum += inarray[5] * -1.788168;
netsum += inarray[6] * -1.852357;
netsum += inarray[7] * -1.621921;
feature3[16] = tanh(netsum);

netsum =  9.458772E-02;
netsum += inarray[0] * -2.286869;
netsum += inarray[1] *  3.926624;
netsum += inarray[2] *  6.674441;
netsum += inarray[3] *  .2280252;
netsum += inarray[4] *  .4592978;
netsum += inarray[5] *  .3678898;
netsum += inarray[6] *  .3185799;
netsum += inarray[7] *  .1530698;
feature3[17] = tanh(netsum);

netsum =  1.31436;
netsum += inarray[0] * -.9401179;
netsum += inarray[1] *  6.66729;
netsum += inarray[2] *  3.916973E-02;
netsum += inarray[3] * -1.049552;
netsum += inarray[4] * -1.457971;
netsum += inarray[5] * -1.279358;
netsum += inarray[6] * -1.373488;
netsum += inarray[7] * -.9733554;
feature3[18] = tanh(netsum);

netsum = -8.882126;
netsum += inarray[0] * -56.7633;
netsum += inarray[1] *  .4994507;
netsum += inarray[2] *  1.700129;
netsum += inarray[3] *  9.041796;
netsum += inarray[4] *  8.973028;
netsum += inarray[5] *  8.903322;
netsum += inarray[6] *  8.632603;
netsum += inarray[7] *  8.837315;
feature4[0] = 1 - exp(-netsum * netsum);

netsum =  .2946113;
netsum += inarray[0] * -2.669188E-02;
netsum += inarray[1] * -.1392849;
netsum += inarray[2] * -2.396717;
netsum += inarray[3] * -.1488648;
netsum += inarray[4] *  5.854492E-02;
netsum += inarray[5] *  3.869787E-02;
netsum += inarray[6] * -.1380638;
netsum += inarray[7] * -.3303229;
feature4[1] = 1 - exp(-netsum * netsum);

netsum = -2.988592;
netsum += inarray[0] * -36.06646;
netsum += inarray[1] *  6.935266;
netsum += inarray[2] *  11.59262;
netsum += inarray[3] *  3.352005;
netsum += inarray[4] *  3.31984;
netsum += inarray[5] *  3.260385;
netsum += inarray[6] *  3.332677;
netsum += inarray[7] *  3.292908;
feature4[2] = 1 - exp(-netsum * netsum);

netsum =  1.046242;
netsum += inarray[0] *  .4921586;
netsum += inarray[1] *  7.810551;
netsum += inarray[2] * -.6721357;
netsum += inarray[3] * -.8029283;
netsum += inarray[4] * -.9597003;
netsum += inarray[5] * -.7063922;
netsum += inarray[6] * -.9885621;
netsum += inarray[7] * -.6535472;
feature4[3] = 1 - exp(-netsum * netsum);

netsum =  .8776177;
netsum += inarray[0] *  2.251366;
netsum += inarray[1] * -1.385321;
netsum += inarray[2] *  .608431;
netsum += inarray[3] * -.8546925;
netsum += inarray[4] * -.8754066;
netsum += inarray[5] * -.7128801;
netsum += inarray[6] * -.4557205;
netsum += inarray[7] * -.5036122;
feature4[4] = 1 - exp(-netsum * netsum);

netsum = -1.872471;
netsum += inarray[0] * -5.656478;
netsum += inarray[1] * -8.008894;
netsum += inarray[2] *  .8030125;
netsum += inarray[3] *  1.849906;
netsum += inarray[4] *  1.434005;
netsum += inarray[5] *  1.640865;
netsum += inarray[6] *  1.685682;
netsum += inarray[7] *  1.649121;
feature4[5] = 1 - exp(-netsum * netsum);

netsum = -.1802559;
netsum += inarray[0] * -2.421047;
netsum += inarray[1] * -1.114509;
netsum += inarray[2] *  6.102552E-02;
netsum += inarray[3] *  .1614573;
netsum += inarray[4] * -7.837715E-03;
netsum += inarray[5] *  .4052194;
netsum += inarray[6] *  .3671643;
```

```
netsum += inarray[7] * -.1518567;
feature4[6] = 1 - exp(-netsum * netsum);

netsum =  2.880243;
netsum += inarray[0] *  .873866;
netsum += inarray[1] *  16.29805;
netsum += inarray[2] *  .2244682;
netsum += inarray[3] * -2.960358;
netsum += inarray[4] * -2.549545;
netsum += inarray[5] * -2.741243;
netsum += inarray[6] * -3.056852;
netsum += inarray[7] * -3.005763;
feature4[7] = 1 - exp(-netsum * netsum);

netsum =  5.581188;
netsum += inarray[0] * -3.799807;
netsum += inarray[1] *  35.61201;
netsum += inarray[2] *  3.603816;
netsum += inarray[3] * -5.783441;
netsum += inarray[4] * -5.3826;
netsum += inarray[5] * -5.691954;
netsum += inarray[6] * -5.736175;
netsum += inarray[7] * -5.293347;
feature4[8] = 1 - exp(-netsum * netsum);

netsum =  3.157714;
netsum += inarray[0] *  6.550279;
netsum += inarray[1] * -.2096181;
netsum += inarray[2] *  13.65161;
netsum += inarray[3] * -3.111802;
netsum += inarray[4] * -3.336902;
netsum += inarray[5] * -3.228772;
netsum += inarray[6] * -3.188584;
netsum += inarray[7] * -3.230474;
feature4[9] = 1 - exp(-netsum * netsum);

netsum =  .4420536;
netsum += inarray[0] *  8.406163;
netsum += inarray[1] * -1.705918;
netsum += inarray[2] * -2.675127;
netsum += inarray[3] * -8.393306E-02;
netsum += inarray[4] * -.4554291;
netsum += inarray[5] * -.20817;
netsum += inarray[6] * -.1638989;
netsum += inarray[7] * -.1985271;
feature4[10] = 1 - exp(-netsum * netsum);

netsum =  .2059372;
netsum += inarray[0] *  1.914523;
netsum += inarray[1] *  1.535249;
netsum += inarray[2] *  5.900069E-02;
netsum += inarray[3] * -.056491;
netsum += inarray[4] *  6.104679E-02;
netsum += inarray[5] * -.0412524;
netsum += inarray[6] * -3.617917E-02;
netsum += inarray[7] * -2.578711E-02;
feature4[11] = 1 - exp(-netsum * netsum);

netsum =  6.459284;
netsum += inarray[0] * -2.055845;
netsum += inarray[1] *  42.89668;
netsum += inarray[2] *  2.51774;
netsum += inarray[3] * -6.342625;
netsum += inarray[4] * -6.599291;
netsum += inarray[5] * -6.563766;
netsum += inarray[6] * -6.792032;
netsum += inarray[7] * -6.31735;
feature4[12] = 1 - exp(-netsum * netsum);

netsum = -.1497506;
netsum += inarray[0] * -21.55146;
netsum += inarray[1] *  20.83467;
netsum += inarray[2] *  .9428095;
netsum += inarray[3] *  .1687066;
netsum += inarray[4] *  2.308896E-02;
netsum += inarray[5] *  .3005248;
netsum += inarray[6] *  .1227963;
netsum += inarray[7] *  .4052385;
feature4[13] = 1 - exp(-netsum * netsum);

netsum = -.3201815;
netsum += inarray[0] *  1.128974;
netsum += inarray[1] *  1.275685;
netsum += inarray[2] *  3.872861;
netsum += inarray[3] *  .5263622;
netsum += inarray[4] *  .4938102;
netsum += inarray[5] *  9.721208E-02;
netsum += inarray[6] *  .2253827;
netsum += inarray[7] *  .4857824;
feature4[14] = 1 - exp(-netsum * netsum);

netsum = -.1042724;
netsum += inarray[0] *  1.296381;
netsum += inarray[1] *  .5646659;
netsum += inarray[2] * -1.175975;
netsum += inarray[3] *  .3500463;
netsum += inarray[4] *  .4059474;
netsum += inarray[5] *  1.695132E-02;
netsum += inarray[6] *  .5305222;
netsum += inarray[7] *  .2439102;
feature4[15] = 1 - exp(-netsum * netsum);

netsum =  .6527948;
netsum += inarray[0] *  6.507547;
netsum += inarray[1] * -.6952026;
netsum += inarray[2] * -.6596867;
netsum += inarray[3] * -.7786314;
netsum += inarray[4] * -.9697689;
netsum += inarray[5] * -1.16457;
netsum += inarray[6] * -1.144388;
netsum += inarray[7] * -.9160156;
feature4[16] = 1 - exp(-netsum * netsum);
```

```
netsum = -.6046912;
netsum += inarray[0] *  1.787036;
netsum += inarray[1] * -.5156695;
netsum += inarray[2] * -2.436541;
netsum += inarray[3] *  9.686866E-02;
netsum += inarray[4] *  .2867557;
netsum += inarray[5] *  .1457754;
netsum += inarray[6] *  .3346111;
netsum += inarray[7] *  .4986471;
feature4[17] = 1 - exp(-netsum * netsum);

netsum =  .5395787;
netsum += inarray[0] * -.1309323;
netsum += inarray[1] * -.2037924;
netsum += inarray[2] *  1.932976;
netsum += inarray[3] * -5.163424E-02;
netsum += inarray[4] * -.2064637;
netsum += inarray[5] * -.2688338;
netsum += inarray[6] * -.4853292;
netsum += inarray[7] * -8.659288E-02;
feature4[18] = 1 - exp(-netsum * netsum);

netsum =  8.718774E-02;
netsum += feature2[0] *  .016577;
netsum += feature2[1] * -8.873417E-02;
netsum += feature2[2] * -1.876435E-02;
netsum += feature2[3] * -.0473732;
netsum += feature2[4] * -1.905886E-03;
netsum += feature2[5] * -7.590414E-02;
netsum += feature2[6] * -.5210372;
netsum += feature2[7] * -3.099672E-02;
netsum += feature2[8] * -.316933;
netsum += feature2[9] *  .2880816;
netsum += feature2[10] *  2.030445E-02;
netsum += feature2[11] * -2.129001E-02;
netsum += feature2[12] * -.5636273;
netsum += feature2[13] * -.3761285;
netsum += feature2[14] * -1.680289E-02;
netsum += feature2[15] * -.1948952;
netsum += feature2[16] *  4.076173E-03;
netsum += feature2[17] *  1.293544;
netsum += feature2[18] *  .4369362;
netsum += -.2476546;
netsum += feature3[0] *  .4522427;
netsum += feature3[1] * -.1500977;
netsum += feature3[2] * -.4142584;
netsum += feature3[3] * -1.053195;
netsum += feature3[4] * -.1539914;
netsum += feature3[5] * -.7941819;
netsum += feature3[6] * -.5460317;
netsum += feature3[7] *  .8565906;
netsum += feature3[8] * -.5832472;
netsum += feature3[9] *  .8842039;
netsum += feature3[10] *  .1870124;
netsum += feature3[11] * -.0729821;
netsum += feature3[12] * -.2796381;
netsum += feature3[13] * -.2261133;
netsum += feature3[14] * -.1054694;
netsum += feature3[15] *  .145014;
netsum += feature3[16] *  .3320754;
netsum += feature3[17] * -.4054737;
netsum += feature3[18] *  3.391928;
netsum += -.2470685;
netsum += feature4[0] *  .0778499;
netsum += feature4[1] * -.2969437;
netsum += feature4[2] * -9.406544E-03;
netsum += feature4[3] * -.341611;
netsum += feature4[4] *  2.510133;
netsum += feature4[5] * -.1079716;
netsum += feature4[6] * -1.074124;
netsum += feature4[7] *  .3115947;
netsum += feature4[8] *  8.104191E-02;
netsum += feature4[9] * -.0440037;
netsum += feature4[10] *  8.186349E-02;
netsum += feature4[11] * -1.826585;
netsum += feature4[12] *  7.565524E-02;
netsum += feature4[13] * -3.143007E-02;
netsum += feature4[14] * -.9910228;
netsum += feature4[15] * -2.218725;
netsum += feature4[16] *  .9347152;
netsum += feature4[17] * -.4702787;
netsum += feature4[18] *  .5353323;
outarray[0] = 1 / (1 + exp(-netsum));

netsum = -.3346092;
netsum += feature2[0] *  .0162934;
netsum += feature2[1] * -8.818297E-02;
netsum += feature2[2] * -1.896266E-02;
netsum += feature2[3] * -4.723549E-02;
netsum += feature2[4] * -1.834355E-03;
netsum += feature2[5] * -7.583345E-02;
netsum += feature2[6] * -.2388382;
netsum += feature2[7] * -3.142047E-02;
netsum += feature2[8] * -.2424299;
netsum += feature2[9] *  .2866326;
netsum += feature2[10] *  .0188608;
netsum += feature2[11] * -2.148598E-02;
netsum += feature2[12] * -.5648733;
netsum += feature2[13] * -.4127173;
netsum += feature2[14] * -1.679739E-02;
netsum += feature2[15] * -.1955786;
netsum += feature2[16] *  4.205631E-03;
netsum += feature2[17] *  1.29769;
netsum += feature2[18] *  .4399984;
netsum += -.1864281;
netsum += feature3[0] *  .364581;
netsum += feature3[1] * -.1513544;
netsum += feature3[2] * -.4113649;
netsum += feature3[3] * -1.053693;
netsum += feature3[4] * -.1549695;
netsum += feature3[5] * -.793064;
netsum += feature3[6] * -.546452;
netsum += feature3[7] *  .8558335;
netsum += feature3[8] * -.5818372;
```

```
netsum += feature3[9] * .8879998;
netsum += feature3[10] * .1887136;
netsum += feature3[11] * -7.139468E-02;
netsum += feature3[12] * -.2795635;
netsum += feature3[13] * -.3441044;
netsum += feature3[14] * -.102743;
netsum += feature3[15] * .2640757;
netsum += feature3[16] * .3327664;
netsum += feature3[17] * -.4049702;
netsum += feature3[18] * 3.393776;
netsum += -5.952555E-02;
netsum += feature4[0] * 7.809013E-02;
netsum += feature4[1] * -.2944781;
netsum += feature4[2] * -9.549645E-03;
netsum += feature4[3] * -.3421195;
netsum += feature4[4] * 2.502638;
netsum += feature4[5] * -.108158;
netsum += feature4[6] * -1.074739;
netsum += feature4[7] * .3107743;
netsum += feature4[8] * .0810126;
netsum += feature4[9] * -4.368422E-02;
netsum += feature4[10] * 8.195668E-02;
netsum += feature4[11] * -1.825214;
netsum += feature4[12] * 7.572651E-02;
netsum += feature4[13] * -3.159206E-02;
netsum += feature4[14] * -.9886568;
netsum += feature4[15] * -2.219986;
netsum += feature4[16] * .9354509;
netsum += feature4[17] * -.4683259;
netsum += feature4[18] * .5327024;
outarray[1] = 1 / (1 + exp(-netsum));

netsum = -5.875017E-02;
netsum += feature2[0] * -3.166651E-02;
netsum += feature2[1] * -5.040236E-03;
netsum += feature2[2] * -1.325287E-03;
netsum += feature2[3] * 3.359487E-02;
netsum += feature2[4] * -3.434391E-02;
netsum += feature2[5] * 3.47659E-03;
netsum += feature2[6] * 5.479329E-02;
netsum += feature2[7] * -2.733007E-02;
netsum += feature2[8] * -.2455989;
netsum += feature2[9] * -.1900701;
netsum += feature2[10] * 7.756468E-02;
netsum += feature2[11] * 2.725795E-02;
netsum += feature2[12] * 1.830292E-02;
netsum += feature2[13] * 7.44888E-03;
netsum += feature2[14] * 3.267475E-03;
netsum += feature2[15] * -3.946541E-03;
netsum += feature2[16] * -5.86097E-03;
netsum += feature2[17] * .4065373;
netsum += feature2[18] * -.1502019;
netsum += -.2545307;
netsum += feature3[0] * -.2750452;
netsum += feature3[1] * .1402813;
netsum += feature3[2] * .7680174;
netsum += feature3[3] * 3.205167;
netsum += feature3[4] * .3845983;
netsum += feature3[5] * -4.843915E-02;
netsum += feature3[6] * 2.182159E-02;
netsum += feature3[7] * .8283022;
netsum += feature3[8] * -3.872889E-02;
netsum += feature3[9] * -.2427877;
netsum += feature3[10] * 3.113943E-02;
netsum += feature3[11] * 2.897749E-02;
netsum += feature3[12] * -.2942756;
netsum += feature3[13] * .3631359;
netsum += feature3[14] * .2022129;
netsum += feature3[15] * -.1131312;
netsum += feature3[16] * 5.528154E-02;
netsum += feature3[17] * .2482427;
netsum += feature3[18] * .1014222;
netsum += -9.616262E-02;
netsum += feature4[0] * -4.154647E-02;
netsum += feature4[1] * -7.639001E-02;
netsum += feature4[2] * 2.083706E-02;
netsum += feature4[3] * -5.308967E-02;
netsum += feature4[4] * -.3329391;
netsum += feature4[5] * -2.775598E-03;
netsum += feature4[6] * 2.710117E-03;
netsum += feature4[7] * .2471321;
netsum += feature4[8] * -6.56596E-04;
netsum += feature4[9] * -5.322411E-02;
netsum += feature4[10] * 2.302301E-02;
netsum += feature4[11] * 3.701547E-03;
netsum += feature4[12] * 9.288241E-03;
netsum += feature4[13] * 5.876423E-03;
netsum += feature4[14] * -.1294795;
netsum += feature4[15] * .1787323;
netsum += feature4[16] * -.6905922;
netsum += feature4[17] * .2179588;
netsum += feature4[18] * .2099681;
outarray[2] = 1 / (1 + exp(-netsum));

netsum = .1804229;
netsum += feature2[0] * -8.506813E-04;
netsum += feature2[1] * 9.23208E-04;
netsum += feature2[2] * -1.022537E-02;
netsum += feature2[3] * -7.963174E-03;
netsum += feature2[4] * .0146363;
netsum += feature2[5] * 1.545909E-02;
netsum += feature2[6] * .5107836;
netsum += feature2[7] * -4.521234E-03;
netsum += feature2[8] * 4.878249E-02;
netsum += feature2[9] * 6.160371E-03;
netsum += feature2[10] * .1077227;
netsum += feature2[11] * 1.783691E-02;
netsum += feature2[12] * .0222999;
netsum += feature2[13] * .4656031;
netsum += feature2[14] * -1.313233E-02;
netsum += feature2[15] * 2.607921E-03;
netsum += feature2[16] * -4.949759E-03;
netsum += feature2[17] * .5555099;
netsum += feature2[18] * -.4287879;
```

```
netsum +=  .680896;
netsum += feature3[0] * -.2567368;
netsum += feature3[1] * -2.155335;
netsum += feature3[2] *  .1160899;
netsum += feature3[3] *  5.461229E-03;
netsum += feature3[4] * -.1099943;
netsum += feature3[5] *  3.436682E-02;
netsum += feature3[6] * -1.264658E-03;
netsum += feature3[7] * -2.82851;
netsum += feature3[8] * -2.65205E-03;
netsum += feature3[9] *  .4060166;
netsum += feature3[10] * -2.759733E-02;
netsum += feature3[11] *  2.656027E-02;
netsum += feature3[12] * -3.967452E-02;
netsum += feature3[13] *  .5912968;
netsum += feature3[14] * -.1288924;
netsum += feature3[15] * -.5593992;
netsum += feature3[16] * -.0190982;
netsum += feature3[17] * -.2077267;
netsum += feature3[18] * -8.158438E-03;
netsum +=  .314011;
netsum += feature4[0] *  1.178027E-02;
netsum += feature4[1] *  .2284925;
netsum += feature4[2] *  9.611134E-03;
netsum += feature4[3] *  .0192547;
netsum += feature4[4] *  .6921598;
netsum += feature4[5] * -5.340625E-03;
netsum += feature4[6] *  .0298273;
netsum += feature4[7] * -.0330142;
netsum += feature4[8] * -1.567961E-03;
netsum += feature4[9] *  .1403046;
netsum += feature4[10] * -2.193901E-02;
netsum += feature4[11] * -3.834914E-02;
netsum += feature4[12] * -6.305316E-03;
netsum += feature4[13] *  2.929683E-03;
netsum += feature4[14] *  5.768358E-02;
netsum += feature4[15] * -.2288839;
netsum += feature4[16] *  7.486493E-02;
netsum += feature4[17] * -.124349;
netsum += feature4[18] * -.8698367;
outarray[3] = 1 / (1 + exp(-netsum));

netsum = -1.140142E-02;
netsum += feature2[0] * -7.838357E-03;
netsum += feature2[1] * -1.716437E-02;
netsum += feature2[2] * -2.236265E-04;
netsum += feature2[3] * -3.709573E-02;
netsum += feature2[4] * -1.166194E-02;
netsum += feature2[5] *  7.685617E-03;
netsum += feature2[6] * -.1242829;
netsum += feature2[7] * -5.33105E-03;
netsum += feature2[8] * -.3726002;
netsum += feature2[9] *  5.058989E-02;
netsum += feature2[10] * -2.762204E-02;
netsum += feature2[11] * -1.439426E-02;
netsum += feature2[12] * -.126828;
netsum += feature2[13] * -.3397655;
netsum += feature2[14] * -1.784105E-02;
netsum += feature2[15] *  .1220517;
netsum += feature2[16] * -1.905494E-02;
netsum += feature2[17] * -.3979155;
netsum += feature2[18] * -.5736315;
netsum +=  .0678502;
netsum += feature3[0] * -.2191589;
netsum += feature3[1] *  5.086157E-03;
netsum += feature3[2] * -5.358848E-02;
netsum += feature3[3] * -1.366936;
netsum += feature3[4] * -.3844801;
netsum += feature3[5] * -.1429262;
netsum += feature3[6] * -.1479802;
netsum += feature3[7] *  .4547073;
netsum += feature3[8] *  1.706924E-02;
netsum += feature3[9] * -3.950949E-02;
netsum += feature3[10] *  2.481355E-03;
netsum += feature3[11] *  7.372723E-02;
netsum += feature3[12] * -5.580768E-02;
netsum += feature3[13] * -.4474029;
netsum += feature3[14] * -.4087104;
netsum += feature3[15] *  .2381571;
netsum += feature3[16] * -5.183303E-02;
netsum += feature3[17] * -.2171978;
netsum += feature3[18] * -.3729102;
netsum += -.4325876;
netsum += feature4[0] *  5.539147E-02;
netsum += feature4[1] * -.1710119;
netsum += feature4[2] * -1.815156E-02;
netsum += feature4[3] *  6.133429E-02;
netsum += feature4[4] *  .8661048;
netsum += feature4[5] * -7.830662E-03;
netsum += feature4[6] *  .0110472;
netsum += feature4[7] *  .104885;
netsum += feature4[8] *  5.136887E-03;
netsum += feature4[9] * -2.501908E-02;
netsum += feature4[10] * -2.916389E-02;
netsum += feature4[11] *  1.176684E-02;
netsum += feature4[12] *  4.703646E-03;
netsum += feature4[13] * -4.332963E-03;
netsum += feature4[14] * -.2037196;
netsum += feature4[15] * -.7795016;
netsum += feature4[16] *  .8424653;
netsum += feature4[17] * -.1512669;
netsum += feature4[18] *  .261848;
outarray[4] = 1 / (1 + exp(-netsum));

netsum = -4.722031E-02;
netsum += feature2[0] * -8.000262E-03;
netsum += feature2[1] * -1.680207E-02;
netsum += feature2[2] * -3.243294E-04;
netsum += feature2[3] * -.0369595;
netsum += feature2[4] * -1.186541E-02;
netsum += feature2[5] *  7.525295E-03;
netsum += feature2[6] * -9.797968E-02;
netsum += feature2[7] * -5.410881E-03;
netsum += feature2[8] * -.1368091;
```

```
netsum += feature2[9] *  5.033706E-02;
netsum += feature2[10] * -2.877823E-02;
netsum += feature2[11] * -1.439525E-02;
netsum += feature2[12] * -.127058;
netsum += feature2[13] * -.3770014;
netsum += feature2[14] * -1.794307E-02;
netsum += feature2[15] *  .1209394;
netsum += feature2[16] * -1.890023E-02;
netsum += feature2[17] * -.3931586;
netsum += feature2[18] * -.5720466;
netsum += -.2855176;
netsum += feature3[0] *  .1943337;
netsum += feature3[1] *  5.368457E-03;
netsum += feature3[2] * -4.763287E-02;
netsum += feature3[3] * -1.366314;
netsum += feature3[4] * -.3864007;
netsum += feature3[5] * -.142164;
netsum += feature3[6] * -.1450251;
netsum += feature3[7] *  .4502307;
netsum += feature3[8] *  1.816689E-02;
netsum += feature3[9] * -2.950455E-02;
netsum += feature3[10] *  2.978366E-03;
netsum += feature3[11] *  7.511196E-02;
netsum += feature3[12] * -5.523267E-02;
netsum += feature3[13] * -.466932;
netsum += feature3[14] * -.4090543;
netsum += feature3[15] *  4.579091E-02;
netsum += feature3[16] * -5.112208E-02;
netsum += feature3[17] * -.2180664;
netsum += feature3[18] * -.3693007;
netsum += -5.463653E-02;
netsum += feature4[0] *  5.535346E-02;
netsum += feature4[1] * -.1701238;
netsum += feature4[2] * -1.807726E-02;
netsum += feature4[3] *  .0606728;
netsum += feature4[4] *  .8792794;
netsum += feature4[5] * -8.105821E-03;
netsum += feature4[6] *  1.164849E-02;
netsum += feature4[7] *  .1039171;
netsum += feature4[8] *  5.215293E-03;
netsum += feature4[9] * -2.434635E-02;
netsum += feature4[10] * -2.912143E-02;
netsum += feature4[11] *  1.730887E-02;
netsum += feature4[12] *  4.798151E-03;
netsum += feature4[13] * -4.405526E-03;
netsum += feature4[14] * -.2025119;
netsum += feature4[15] * -.7789724;
netsum += feature4[16] *  .842842;
netsum += feature4[17] * -.1508292;
netsum += feature4[18] *  .2613397;
outarray[5] = 1 / (1 + exp(-netsum));

netsum = -.1906302;
netsum += feature2[0] * -8.213125E-03;
netsum += feature2[1] * -1.625005E-02;
netsum += feature2[2] * -7.886151E-04;
netsum += feature2[3] * -.0370568;
netsum += feature2[4] * -1.222747E-02;
netsum += feature2[5] *  7.570244E-03;
netsum += feature2[6] * -7.861973E-02;
netsum += feature2[7] * -5.739116E-03;
netsum += feature2[8] * -.3296;
netsum += feature2[9] *  4.936224E-02;
netsum += feature2[10] * -3.011088E-02;
netsum += feature2[11] * -.0145221;
netsum += feature2[12] * -.1286291;
netsum += feature2[13] * -.4332907;
netsum += feature2[14] * -1.838263E-02;
netsum += feature2[15] *  .1185957;
netsum += feature2[16] * -1.885975E-02;
netsum += feature2[17] * -.3871869;
netsum += feature2[18] * -.5645671;
netsum += -.1956375;
netsum += feature3[0] * -3.669748E-02;
netsum += feature3[1] *  3.678397E-03;
netsum += feature3[2] * -5.043067E-02;
netsum += feature3[3] * -1.365451;
netsum += feature3[4] * -.3871159;
netsum += feature3[5] * -.1406406;
netsum += feature3[6] * -.1444825;
netsum += feature3[7] *  .4504517;
netsum += feature3[8] *  1.983386E-02;
netsum += feature3[9] * -3.311089E-02;
netsum += feature3[10] *  3.768059E-03;
netsum += feature3[11] *  7.854988E-02;
netsum += feature3[12] * -5.402923E-02;
netsum += feature3[13] * -.2571496;
netsum += feature3[14] * -.4052454;
netsum += feature3[15] *  .1047454;
netsum += feature3[16] * -5.019277E-02;
netsum += feature3[17] * -.2167054;
netsum += feature3[18] * -.3626605;
netsum += -.1606783;
netsum += feature4[0] *  5.559727E-02;
netsum += feature4[1] * -.1679029;
netsum += feature4[2] * -1.833118E-02;
netsum += feature4[3] *  5.938511E-02;
netsum += feature4[4] *  .855292;
netsum += feature4[5] * -8.521466E-03;
netsum += feature4[6] *  1.042634E-02;
netsum += feature4[7] *  .1029989;
netsum += feature4[8] *  5.263951E-03;
netsum += feature4[9] * -2.419138E-02;
netsum += feature4[10] * -.0288771;
netsum += feature4[11] *  2.108153E-02;
netsum += feature4[12] *  4.965364E-03;
netsum += feature4[13] * -4.730272E-03;
netsum += feature4[14] * -.2000619;
netsum += feature4[15] * -.7695271;
netsum += feature4[16] *  .8439519;
netsum += feature4[17] * -.1489539;
netsum += feature4[18] *  .2586853;
outarray[6] = 1 / (1 + exp(-netsum));
```

```
netsum = -.0936944;
netsum += feature2[0] * -8.434274E-03;
netsum += feature2[1] * -1.588754E-02;
netsum += feature2[2] * -1.276662E-03;
netsum += feature2[3] * -3.724976E-02;
netsum += feature2[4] * -1.270376E-02;
netsum += feature2[5] *  7.527395E-03;
netsum += feature2[6] * -.3622355;
netsum += feature2[7] * -5.976677E-03;
netsum += feature2[8] * -6.346072E-02;
netsum += feature2[9] *  4.929508E-02;
netsum += feature2[10] * -3.072543E-02;
netsum += feature2[11] * -1.447999E-02;
netsum += feature2[12] * -.1296972;
netsum += feature2[13] * -.4769722;
netsum += feature2[14] * -.0190792;
netsum += feature2[15] *  .1160555;
netsum += feature2[16] * -1.892473E-02;
netsum += feature2[17] * -.3802107;
netsum += feature2[18] * -.5576386;
netsum += -.2540948;
netsum += feature3[0] * -.1581596;
netsum += feature3[1] *  3.403907E-03;
netsum += feature3[2] * -.0517299;
netsum += feature3[3] * -1.364262;
netsum += feature3[4] * -.3882559;
netsum += feature3[5] * -.1393557;
netsum += feature3[6] * -.1426372;
netsum += feature3[7] *  .4483981;
netsum += feature3[8] *  2.147656E-02;
netsum += feature3[9] * -3.397377E-02;
netsum += feature3[10] *  3.27942E-03;
netsum += feature3[11] *  8.231948E-02;
netsum += feature3[12] * -5.186557E-02;
netsum += feature3[13] * -.2958901;
netsum += feature3[14] * -.4041604;
netsum += feature3[15] *  .1474643;
netsum += feature3[16] * -4.947887E-02;
netsum += feature3[17] * -.2164469;
netsum += feature3[18] * -.354253;
netsum += -.1930426;
netsum += feature4[0] *  5.580377E-02;
netsum += feature4[1] * -.1671444;
netsum += feature4[2] * -1.851363E-02;
netsum += feature4[3] *  5.829655E-02;
netsum += feature4[4] *  .8352511;
netsum += feature4[5] * -9.114374E-03;
netsum += feature4[6] *  8.735992E-03;
netsum += feature4[7] *  .1020404;
netsum += feature4[8] *  5.405982E-03;
netsum += feature4[9] * -2.405058E-02;
netsum += feature4[10] * -2.876697E-02;
netsum += feature4[11] *  2.502191E-02;
netsum += feature4[12] *  5.003968E-03;
netsum += feature4[13] * -5.019467E-03;
netsum += feature4[14] * -.1982296;
netsum += feature4[15] * -.7551309;
netsum += feature4[16] *  .8450029;
netsum += feature4[17] * -.1475994;
netsum += feature4[18] *  .2574629;
outarray[7] = 1 / (1 + exp(-netsum));

netsum =  .2035794;
netsum += feature2[0] * -3.618084E-02;
netsum += feature2[1] * -5.953116E-02;
netsum += feature2[2] *  1.007071E-02;
netsum += feature2[3] *  1.917552E-02;
netsum += feature2[4] * -9.087718E-04;
netsum += feature2[5] * -2.098803E-03;
netsum += feature2[6] *  .1434139;
netsum += feature2[7] * -6.297688E-03;
netsum += feature2[8] *  .1421541;
netsum += feature2[9] * -.0377967;
netsum += feature2[10] * -2.870324E-02;
netsum += feature2[11] *  1.810617E-02;
netsum += feature2[12] *  4.875324E-02;
netsum += feature2[13] *  .1256413;
netsum += feature2[14] *  1.483328E-02;
netsum += feature2[15] *  .185903;
netsum += feature2[16] *  1.450381E-02;
netsum += feature2[17] *  .1013691;
netsum += feature2[18] *  .2716945;
netsum +=  .1166575;
netsum += feature3[0] * -.1123899;
netsum += feature3[1] *  4.087102E-02;
netsum += feature3[2] *  6.971446E-03;
netsum += feature3[3] *  1.892978;
netsum += feature3[4] *  .3200338;
netsum += feature3[5] * -.3830011;
netsum += feature3[6] *  4.876999E-02;
netsum += feature3[7] * -.5845385;
netsum += feature3[8] * -.1344725;
netsum += feature3[9] * -1.448592;
netsum += feature3[10] *  2.870492E-02;
netsum += feature3[11] *  3.546294E-02;
netsum += feature3[12] * -.2144611;
netsum += feature3[13] * -.0542923;
netsum += feature3[14] *  .1845365;
netsum += feature3[15] *  .1922844;
netsum += feature3[16] *  1.786496E-02;
netsum += feature3[17] *  .1366104;
netsum += feature3[18] *  .3269697;
netsum += -.1101582;
netsum += feature4[0] * -4.800826E-02;
netsum += feature4[1] *  .072121;
netsum += feature4[2] *  9.416622E-03;
netsum += feature4[3] * -6.063479E-03;
netsum += feature4[4] * -1.596525;
netsum += feature4[5] *  1.198939E-02;
netsum += feature4[6] * -1.823218E-02;
netsum += feature4[7] *  .6146363;
netsum += feature4[8] * -2.276465E-02;
netsum += feature4[9] *  4.824297E-02;
netsum += feature4[10] * -8.443635E-04;
```

```
netsum += feature4[11] *  .0316442;
netsum += feature4[12] *  2.648742E-02;
netsum += feature4[13] * -1.816337E-03;
netsum += feature4[14] *  8.973592E-02;
netsum += feature4[15] *  .5221522;
netsum += feature4[16] * -.4937865;
netsum += feature4[17] *  .1441088;
netsum += feature4[18] * -.2333872;
outarray[8] = 1 / (1 + exp(-netsum));

netsum = -3.957446E-02;
netsum += feature2[0] * -3.577378E-02;
netsum += feature2[1] * -6.060822E-02;
netsum += feature2[2] *  1.168383E-02;
netsum += feature2[3] *  1.986435E-02;
netsum += feature2[4] * -7.167532E-04;
netsum += feature2[5] *  5.931127E-05;
netsum += feature2[6] *  .2158234;
netsum += feature2[7] * -3.931732E-03;
netsum += feature2[8] *  1.732978E-02;
netsum += feature2[9] * -3.892687E-02;
netsum += feature2[10] * -.024825;
netsum += feature2[11] *  1.701006E-02;
netsum += feature2[12] *  4.746775E-02;
netsum += feature2[13] *  .243999;
netsum += feature2[14] *  1.679112E-02;
netsum += feature2[15] *  .1908165;
netsum += feature2[16] *  1.164326E-02;
netsum += feature2[17] *  9.611306E-02;
netsum += feature2[18] *  .2491213;
netsum +=  .107924;
netsum += feature3[0] * -4.506801E-02;
netsum += feature3[1] *  4.431551E-02;
netsum += feature3[2] *  1.122875E-02;
netsum += feature3[3] *  1.920549;
netsum += feature3[4] *  .3129508;
netsum += feature3[5] * -.3852704;
netsum += feature3[6] *  4.343694E-02;
netsum += feature3[7] * -.5923761;
netsum += feature3[8] * -.1369874;
netsum += feature3[9] * -1.44694;
netsum += feature3[10] *  2.815396E-02;
netsum += feature3[11] *  3.169414E-02;
netsum += feature3[12] * -.2195543;
netsum += feature3[13] *  1.876577E-02;
netsum += feature3[14] *  .1729175;
netsum += feature3[15] * -8.882323E-02;
netsum += feature3[16] *  1.616443E-02;
netsum += feature3[17] *  .1344423;
netsum += feature3[18] *  .3141914;
netsum += -.1417206;
netsum += feature4[0] * -4.803752E-02;
netsum += feature4[1] *  6.692366E-02;
netsum += feature4[2] *  0;
netsum += feature4[3] * -4.27691E-03;
netsum += feature4[4] * -1.563858;
netsum += feature4[5] *  1.246548E-02;
netsum += feature4[6] * -1.430748E-02;
netsum += feature4[7] *  .616016;
netsum += feature4[8] * -2.339608E-02;
netsum += feature4[9] *  .0482779;
netsum += feature4[10] *  3.147637E-04;
netsum += feature4[11] *  2.121534E-02;
netsum += feature4[12] *  2.664296E-02;
netsum += feature4[13] * -9.67805E-04;
netsum += feature4[14] *  8.754973E-02;
netsum += feature4[15] *  .5039162;
netsum += feature4[16] * -.4915009;
netsum += feature4[17] *  .1425989;
netsum += feature4[18] * -.2262473;
outarray[9] = 1 / (1 + exp(-netsum));

netsum = -.1364217;
netsum += feature2[0] * -3.670748E-02;
netsum += feature2[1] * -5.849634E-02;
netsum += feature2[2] *  9.616692E-03;
netsum += feature2[3] *  1.908101E-02;
netsum += feature2[4] * -1.155036E-03;
netsum += feature2[5] * -2.228236E-03;
netsum += feature2[6] *  .2042985;
netsum += feature2[7] * -6.869286E-03;
netsum += feature2[8] *  .1561681;
netsum += feature2[9] * -3.898512E-02;
netsum += feature2[10] * -3.114758E-02;
netsum += feature2[11] *  1.810619E-02;
netsum += feature2[12] *  4.800444E-02;
netsum += feature2[13] *  3.470905E-02;
netsum += feature2[14] *  1.429939E-02;
netsum += feature2[15] *  .1830454;
netsum += feature2[16] *  1.462871E-02;
netsum += feature2[17] *  .1135727;
netsum += feature2[18] *  .2795289;
netsum +=  .1126749;
netsum += feature3[0] *  .1152951;
netsum += feature3[1] *  4.074359E-02;
netsum += feature3[2] *  1.252229E-02;
netsum += feature3[3] *  1.893807;
netsum += feature3[4] *  .3176729;
netsum += feature3[5] * -.3809094;
netsum += feature3[6] *  .052681;
netsum += feature3[7] * -.5894774;
netsum += feature3[8] * -.1318426;
netsum += feature3[9] * -1.439578;
netsum += feature3[10] *  2.985867E-02;
netsum += feature3[11] *  3.937068E-02;
netsum += feature3[12] * -.2118781;
netsum += feature3[13] *  .1053979;
netsum += feature3[14] *  .1839007;
netsum += feature3[15] * -.1592381;
netsum += feature3[16] *  1.983089E-02;
netsum += feature3[17] *  .1355022;
netsum += feature3[18] *  .3360194;
netsum += -6.366671E-02;
netsum += feature4[0] * -4.780142E-02;
```

```
netsum += feature4[1] *  7.448743E-02;
netsum += feature4[2] *  9.273367E-03;
netsum += feature4[3] * -7.417067E-03;
netsum += feature4[4] * -1.605408;
netsum += feature4[5] *  1.114403E-02;
netsum += feature4[6] * -1.996889E-02;
netsum += feature4[7] *  .6124293;
netsum += feature4[8] * -2.258185E-02;
netsum += feature4[9] *  .0487793;
netsum += feature4[10] * -1.188714E-03;
netsum += feature4[11] *  3.969554E-02;
netsum += feature4[12] *  2.661194E-02;
netsum += feature4[13] * -2.083077E-03;
netsum += feature4[14] *  9.430573E-02;
netsum += feature4[15] *  .5294154;
netsum += feature4[16] * -.492447;
netsum += feature4[17] *  .14652;
netsum += feature4[18] * -.2363327;
outarray[10] = 1 / (1 + exp(-netsum));

netsum = -9.331414E-02;
netsum += feature2[0] * -3.527369E-02;
netsum += feature2[1] * -6.143057E-02;
netsum += feature2[2] *  1.108705E-02;
netsum += feature2[3] *  1.961268E-02;
netsum += feature2[4] * -5.24083E-04;
netsum += feature2[5] * -2.107925E-03;
netsum += feature2[6] * -.1280103;
netsum += feature2[7] * -4.760764E-03;
netsum += feature2[8] *  7.491789E-02;
netsum += feature2[9] * -3.456047E-02;
netsum += feature2[10] * -2.424417E-02;
netsum += feature2[11] *  1.823703E-02;
netsum += feature2[12] *  5.176281E-02;
netsum += feature2[13] *  .2974462;
netsum += feature2[14] *  1.602083E-02;
netsum += feature2[15] *  .1921629;
netsum += feature2[16] *  1.438447E-02;
netsum += feature2[17] *  .0833054;
netsum += feature2[18] *  .2488984;
netsum += -7.087833E-02;
netsum += feature3[0] *  .2792824;
netsum += feature3[1] *  4.661345E-02;
netsum += feature3[2] *  8.562759E-03;
netsum += feature3[3] *  1.892727;
netsum += feature3[4] *  .3222882;
netsum += feature3[5] * -.3880146;
netsum += feature3[6] *  4.564139E-02;
netsum += feature3[7] * -.5843915;
netsum += feature3[8] * -.1401253;
netsum += feature3[9] * -1.44609;
netsum += feature3[10] *  2.527098E-02;
netsum += feature3[11] *  2.655809E-02;
netsum += feature3[12] * -.2189262;
netsum += feature3[13] *  .3834065;
netsum += feature3[14] *  .1740427;
netsum += feature3[15] * -6.211202E-02;
netsum += feature3[16] *  1.418614E-02;
netsum += feature3[17] *  .1348812;
netsum += feature3[18] *  .30969;
netsum +=  .2811685;
netsum += feature4[0] * -.0486275;
netsum += feature4[1] *  6.380343E-02;
netsum += feature4[2] *  9.840358E-03;
netsum += feature4[3] * -2.980912E-03;
netsum += feature4[4] * -1.547265;
netsum += feature4[5] *  .0133683;
netsum += feature4[6] * -1.231287E-02;
netsum += feature4[7] *  .6180921;
netsum += feature4[8] * -2.308469E-02;
netsum += feature4[9] *  4.787366E-02;
netsum += feature4[10] * -3.843875E-04;
netsum += feature4[11] *  2.157984E-02;
netsum += feature4[12] *  2.614807E-02;
netsum += feature4[13] * -1.224828E-03;
netsum += feature4[14] *  8.157405E-02;
netsum += feature4[15] *  .5005958;
netsum += feature4[16] * -.4967542;
netsum += feature4[17] *  .1376857;
netsum += feature4[18] * -.2242603;
outarray[11] = 1 / (1 + exp(-netsum));

netsum = -.5401376;
netsum += feature2[0] * -7.119249E-03;
netsum += feature2[1] * -1.893555E-02;
netsum += feature2[2] *  1.158581E-03;
netsum += feature2[3] * -3.691031E-02;
netsum += feature2[4] * -.0103301;
netsum += feature2[5] *  7.818467E-03;
netsum += feature2[6] * -7.64315E-03;
netsum += feature2[7] * -4.407803E-03;
netsum += feature2[8] * -6.023187E-02;
netsum += feature2[9] *  5.281226E-02;
netsum += feature2[10] * -2.340325E-02;
netsum += feature2[11] * -1.414669E-02;
netsum += feature2[12] * -.1226461;
netsum += feature2[13] * -.1599962;
netsum += feature2[14] * -1.637655E-02;
netsum += feature2[15] *  .1298324;
netsum += feature2[16] * -1.927272E-02;
netsum += feature2[17] * -.4184353;
netsum += feature2[18] * -.5956229;
netsum += -.3499072;
netsum += feature3[0] * -.138177;
netsum += feature3[1] *  8.959621E-03;
netsum += feature3[2] * -5.046866E-02;
netsum += feature3[3] * -1.370151;
netsum += feature3[4] * -.3807102;
netsum += feature3[5] * -.1478497;
netsum += feature3[6] * -.1541867;
netsum += feature3[7] *  .45907;
netsum += feature3[8] *  1.139242E-02;
netsum += feature3[9] * -3.829441E-02;
netsum += feature3[10] *  6.403938E-04;
```

```
netsum += feature3[11] *  6.301505E-02;
netsum += feature3[12] * -.0603002;
netsum += feature3[13] * -.3132776;
netsum += feature3[14] * -.4171314;
netsum += feature3[15] *  .2120902;
netsum += feature3[16] * -5.503085E-02;
netsum += feature3[17] * -.2191872;
netsum += feature3[18] * -.3957516;
netsum += -.1164713;
netsum += feature4[0] *  .0548205;
netsum += feature4[1] * -.1773958;
netsum += feature4[2] * -1.760135E-02;
netsum += feature4[3] *  6.549563E-02;
netsum += feature4[4] *  .9138562;
netsum += feature4[5] * -6.360791E-03;
netsum += feature4[6] *  1.379343E-02;
netsum += feature4[7] *  .1083044;
netsum += feature4[8] *  4.882724E-03;
netsum += feature4[9] * -2.599178E-02;
netsum += feature4[10] * -2.983776E-02;
netsum += feature4[11] * -4.610777E-03;
netsum += feature4[12] *  4.224862E-03;
netsum += feature4[13] * -3.386455E-03;
netsum += feature4[14] * -.2117063;
netsum += feature4[15] * -.8089572;
netsum += feature4[16] *  .8392572;
netsum += feature4[17] * -.1564012;
netsum += feature4[18] *  .2689587;
outarray[12] = 1 / (1 + exp(-netsum));

netsum = -5.328769E-03;
netsum += feature2[0] * -7.641044E-03;
netsum += feature2[1] * -1.781503E-02;
netsum += feature2[2] *  3.739938E-04;
netsum += feature2[3] * -3.690528E-02;
netsum += feature2[4] * -1.099938E-02;
netsum += feature2[5] *  7.798839E-03;
netsum += feature2[6] * -3.456036E-02;
netsum += feature2[7] * -4.960972E-03;
netsum += feature2[8] * -.1116045;
netsum += feature2[9] *  5.109677E-02;
netsum += feature2[10] * -2.664245E-02;
netsum += feature2[11] * -1.427221E-02;
netsum += feature2[12] * -.1252562;
netsum += feature2[13] * -.2713664;
netsum += feature2[14] * -1.716406E-02;
netsum += feature2[15] *  .1254527;
netsum += feature2[16] * -1.912536E-02;
netsum += feature2[17] * -.4029512;
netsum += feature2[18] * -.5834187;
netsum += -.3492062;
netsum += feature3[0] * -.1606688;
netsum += feature3[1] *  7.654977E-03;
netsum += feature3[2] * -4.406772E-02;
netsum += feature3[3] * -1.368562;
netsum += feature3[4] * -.384346;
netsum += feature3[5] * -.1449322;
netsum += feature3[6] * -.1503846;
netsum += feature3[7] *  .453287;
netsum += feature3[8] *  1.519967E-02;
netsum += feature3[9] * -2.785788E-02;
netsum += feature3[10] *  1.900792E-03;
netsum += feature3[11] *  6.977382E-02;
netsum += feature3[12] * -5.761409E-02;
netsum += feature3[13] * -.5271611;
netsum += feature3[14] * -.4111195;
netsum += feature3[15] *  .2237377;
netsum += feature3[16] * -.0533006;
netsum += feature3[17] * -.2170717;
netsum += feature3[18] * -.3815797;
netsum += -.2933703;
netsum += feature4[0] *  5.522196E-02;
netsum += feature4[1] * -.1736668;
netsum += feature4[2] * -1.791244E-02;
netsum += feature4[3] *  6.318515E-02;
netsum += feature4[4] *  .8872541;
netsum += feature4[5] * -7.337211E-03;
netsum += feature4[6] *  1.173665E-02;
netsum += feature4[7] *  .1058245;
netsum += feature4[8] *  5.051277E-03;
netsum += feature4[9] * -2.496983E-02;
netsum += feature4[10] * -2.951601E-02;
netsum += feature4[11] *  5.158252E-03;
netsum += feature4[12] *  4.502754E-03;
netsum += feature4[13] * -3.904314E-03;
netsum += feature4[14] * -.2065603;
netsum += feature4[15] * -.7929084;
netsum += feature4[16] *  .8413042;
netsum += feature4[17] * -.1530087;
netsum += feature4[18] *  .2649225;
outarray[13] = 1 / (1 + exp(-netsum));

netsum = -6.536139E-02;
netsum += feature2[0] * -7.732011E-03;
netsum += feature2[1] * -1.726208E-02;
netsum += feature2[2] * -9.617244E-05;
netsum += feature2[3] * -.0370791;
netsum += feature2[4] * -1.167255E-02;
netsum += feature2[5] *  7.561568E-03;
netsum += feature2[6] * -8.213332E-02;
netsum += feature2[7] * -5.238045E-03;
netsum += feature2[8] * -.408641;
netsum += feature2[9] *  5.102497E-02;
netsum += feature2[10] * -2.707265E-02;
netsum += feature2[11] * -1.440181E-02;
netsum += feature2[12] * -.1262564;
netsum += feature2[13] * -.3323465;
netsum += feature2[14] * -1.770651E-02;
netsum += feature2[15] *  .1223978;
netsum += feature2[16] * -1.899209E-02;
netsum += feature2[17] * -.4015131;
netsum += feature2[18] * -.5753138;
netsum += -1.451544E-02;
netsum += feature3[0] *  1.495429E-02;
```

```
netsum += feature3[1] *  4.733582E-03;
netsum += feature3[2] * -5.756713E-02;
netsum += feature3[3] * -1.366831;
netsum += feature3[4] * -.3836626;
netsum += feature3[5] * -.1431985;
netsum += feature3[6] * -.1475793;
netsum += feature3[7] *  .4558662;
netsum += feature3[8] *  1.642628E-02;
netsum += feature3[9] * -.044288;
netsum += feature3[10] *  2.54652E-03;
netsum += feature3[11] *  7.246819E-02;
netsum += feature3[12] * -5.634412E-02;
netsum += feature3[13] * -.1710896;
netsum += feature3[14] * -.4108147;
netsum += feature3[15] *  .1730432;
netsum += feature3[16] * -.0516924;
netsum += feature3[17] * -.2188903;
netsum += feature3[18] * -.3752116;
netsum += -.4240582;
netsum += feature4[0] *  5.521565E-02;
netsum += feature4[1] * -.1711721;
netsum += feature4[2] * -1.804085E-02;
netsum += feature4[3] *  6.139852E-02;
netsum += feature4[4] *  .8803134;
netsum += feature4[5] * -7.639197E-03;
netsum += feature4[6] *  1.235393E-02;
netsum += feature4[7] *  .1052666;
netsum += feature4[8] *  5.117073E-03;
netsum += feature4[9] * -2.515285E-02;
netsum += feature4[10] * -2.915368E-02;
netsum += feature4[11] *  1.253941E-02;
netsum += feature4[12] *  4.712866E-03;
netsum += feature4[13] * -4.274924E-03;
netsum += feature4[14] * -.2046361;
netsum += feature4[15] * -.7833096;
netsum += feature4[16] *  .8420737;
netsum += feature4[17] * -.1520687;
netsum += feature4[18] *  .2622888;
outarray[14] = 1 / (1 + exp(-netsum));

netsum = -.3137464;
netsum += feature2[0] * -8.50567E-03;
netsum += feature2[1] * -1.562646E-02;
netsum += feature2[2] * -1.511976E-03;
netsum += feature2[3] * -.0373284;
netsum += feature2[4] * -1.301902E-02;
netsum += feature2[5] *  7.427377E-03;
netsum += feature2[6] * -.425519;
netsum += feature2[7] * -6.109207E-03;
netsum += feature2[8] *  7.695494E-02;
netsum += feature2[9] *  4.927329E-02;
netsum += feature2[10] * -3.102662E-02;
netsum += feature2[11] * -1.451275E-02;
netsum += feature2[12] * -.1301809;
netsum += feature2[13] * -.5056526;
netsum += feature2[14] * -1.937582E-02;
netsum += feature2[15] *  .1145802;
netsum += feature2[16] * -1.888184E-02;
netsum += feature2[17] * -.3784086;
netsum += feature2[18] * -.5537906;
netsum +=  7.561964E-02;
netsum += feature3[0] * -9.334236E-02;
netsum += feature3[1] *  2.39874E-03;
netsum += feature3[2] * -5.603926E-02;
netsum += feature3[3] * -1.363456;
netsum += feature3[4] * -.3883288;
netsum += feature3[5] * -.1385272;
netsum += feature3[6] * -.1411389;
netsum += feature3[7] *  .4486763;
netsum += feature3[8] *  2.221095E-02;
netsum += feature3[9] * -3.885284E-02;
netsum += feature3[10] *  3.44523E-03;
netsum += feature3[11] *  8.384104E-02;
netsum += feature3[12] * -5.105376E-02;
netsum += feature3[13] * -.2191407;
netsum += feature3[14] * -.403671;
netsum += feature3[15] *  .3762298;
netsum += feature3[16] * -4.879221E-02;
netsum += feature3[17] * -.2172416;
netsum += feature3[18] * -.3506266;
netsum += -.1491444;
netsum += feature4[0] *  5.582782E-02;
netsum += feature4[1] * -.165963;
netsum += feature4[2] * -1.857692E-02;
netsum += feature4[3] *  5.750158E-02;
netsum += feature4[4] *  .8306136;
netsum += feature4[5] * -9.325077E-03;
netsum += feature4[6] *  8.731665E-03;
netsum += feature4[7] *  .1016451;
netsum += feature4[8] *  5.456118E-03;
netsum += feature4[9] * -2.405351E-02;
netsum += feature4[10] * -2.862741E-02;
netsum += feature4[11] *  2.847248E-02;
netsum += feature4[12] *  5.081743E-03;
netsum += feature4[13] * -5.18783E-03;
netsum += feature4[14] * -.1971735;
netsum += feature4[15] * -.7497256;
netsum += feature4[16] *  .8454262;
netsum += feature4[17] * -.1470856;
netsum += feature4[18] *  .2563407;
outarray[15] = 1 / (1 + exp(-netsum));

netsum = -.1480979;
netsum += feature2[0] * -3.719818E-02;
netsum += feature2[1] * -1.265442E-02;
netsum += feature2[2] * -9.348511E-03;
netsum += feature2[3] *  3.579449E-03;
netsum += feature2[4] * -6.274232E-02;
netsum += feature2[5] * -1.423321E-04;
netsum += feature2[6] *  6.063696E-02;
netsum += feature2[7] * -2.762263E-02;
netsum += feature2[8] *  .3399003;
netsum += feature2[9] * -.1271517;
netsum += feature2[10] * -.1234824;
```

```
netsum += feature2[11] *  3.919817E-03;
netsum += feature2[12] * -7.628346E-02;
netsum += feature2[13] *  .1022288;
netsum += feature2[14] * -6.119589E-03;
netsum += feature2[15] *  8.375827E-02;
netsum += feature2[16] * -7.190211E-03;
netsum += feature2[17] *  .3001102;
netsum += feature2[18] * -.3796772;
netsum +=  .3610182;
netsum += feature3[0] * -.1232244;
netsum += feature3[1] * -2.655659;
netsum += feature3[2] *  .2014078;
netsum += feature3[3] *  .6364655;
netsum += feature3[4] *  8.977345E-02;
netsum += feature3[5] * -.2249691;
netsum += feature3[6] *  .125425;
netsum += feature3[7] *  1.519735;
netsum += feature3[8] * -5.998176E-02;
netsum += feature3[9] * -.8958358;
netsum += feature3[10] *  4.889024E-02;
netsum += feature3[11] *  5.446282E-02;
netsum += feature3[12] * -.3256187;
netsum += feature3[13] * -.1998053;
netsum += feature3[14] *  6.730945E-02;
netsum += feature3[15] * -.3341132;
netsum += feature3[16] * -3.184267E-02;
netsum += feature3[17] * -.1558098;
netsum += feature3[18] *  2.744199E-02;
netsum +=  .2336751;
netsum += feature4[0] * -1.146289E-02;
netsum += feature4[1] *  .2506219;
netsum += feature4[2] *  7.745825E-03;
netsum += feature4[3] * -2.947339E-02;
netsum += feature4[4] * -1.361738;
netsum += feature4[5] * -3.544433E-03;
netsum += feature4[6] *  5.205269E-03;
netsum += feature4[7] *  .4252599;
netsum += feature4[8] *  8.143663E-04;
netsum += feature4[9] *  1.165031E-02;
netsum += feature4[10] * -3.753376E-03;
netsum += feature4[11] *  7.711013E-02;
netsum += feature4[12] *  2.846257E-02;
netsum += feature4[13] * -3.537115E-03;
netsum += feature4[14] * -.1151537;
netsum += feature4[15] * -.2281758;
netsum += feature4[16] *  1.709158E-02;
netsum += feature4[17] *  6.303384E-02;
netsum += feature4[18] * -.8454255;
outarray[16] = 1 / (1 + exp(-netsum));

netsum =  .1706664;
netsum += feature2[0] * -3.748044E-02;
netsum += feature2[1] * -1.204437E-02;
netsum += feature2[2] * -9.64919E-03;
netsum += feature2[3] *  3.668473E-03;
netsum += feature2[4] * -6.303412E-02;
netsum += feature2[5] * -1.669622E-04;
netsum += feature2[6] *  .2796302;
netsum += feature2[7] * -2.813847E-02;
netsum += feature2[8] *  6.264265E-02;
netsum += feature2[9] * -.1283881;
netsum += feature2[10] * -.1254905;
netsum += feature2[11] *  3.794987E-03;
netsum += feature2[12] * -7.785413E-02;
netsum += feature2[13] *  3.694811E-02;
netsum += feature2[14] * -6.355699E-03;
netsum += feature2[15] *  8.184609E-02;
netsum += feature2[16] * -7.048969E-03;
netsum += feature2[17] *  .3057667;
netsum += feature2[18] * -.3726713;
netsum +=  .204025;
netsum += feature3[0] * -.1165983;
netsum += feature3[1] * -2.657602;
netsum += feature3[2] *  .2016479;
netsum += feature3[3] *  .6355295;
netsum += feature3[4] *  8.892174E-02;
netsum += feature3[5] * -.2232565;
netsum += feature3[6] *  .1265325;
netsum += feature3[7] *  1.519587;
netsum += feature3[8] * -5.819671E-02;
netsum += feature3[9] * -.8952294;
netsum += feature3[10] *  5.079608E-02;
netsum += feature3[11] *  5.700771E-02;
netsum += feature3[12] * -.3248402;
netsum += feature3[13] *  .1033836;
netsum += feature3[14] *  6.975435E-02;
netsum += feature3[15] * -.0392566;
netsum += feature3[16] * -3.067273E-02;
netsum += feature3[17] * -.1564076;
netsum += feature3[18] *  3.185306E-02;
netsum +=  .175902;
netsum += feature4[0] * -.0112413;
netsum += feature4[1] *  .254164;
netsum += feature4[2] *  7.637949E-03;
netsum += feature4[3] * -.030551;
netsum += feature4[4] * -1.372878;
netsum += feature4[5] * -3.893749E-03;
netsum += feature4[6] *  4.601928E-03;
netsum += feature4[7] *  .4242747;
netsum += feature4[8] *  8.55815E-04;
netsum += feature4[9] *  1.192789E-02;
netsum += feature4[10] * -3.527038E-03;
netsum += feature4[11] *  8.217558E-02;
netsum += feature4[12] *  .0285475;
netsum += feature4[13] * -3.74042E-03;
netsum += feature4[14] * -.1117577;
netsum += feature4[15] * -.2245504;
netsum += feature4[16] *  1.802132E-02;
netsum += feature4[17] *  6.514741E-02;
netsum += feature4[18] * -.8489565;
outarray[17] = 1 / (1 + exp(-netsum));

netsum = -.1826196;
netsum += feature2[0] * -3.720774E-02;
```

```
netsum += feature2[1] * -1.278795E-02;
netsum += feature2[2] * -9.266318E-03;
netsum += feature2[3] *  3.522575E-03;
netsum += feature2[4] * -6.260777E-02;
netsum += feature2[5] * -1.473054E-04;
netsum += feature2[6] * -.153879;
netsum += feature2[7] * -2.746058E-02;
netsum += feature2[8] *  .4140619;
netsum += feature2[9] * -.1265206;
netsum += feature2[10] * -.1229954;
netsum += feature2[11] *  4.035754E-03;
netsum += feature2[12] * -7.551268E-02;
netsum += feature2[13] *  .1164246;
netsum += feature2[14] * -6.146102E-03;
netsum += feature2[15] *  8.444608E-02;
netsum += feature2[16] * -7.276238E-03;
netsum += feature2[17] *  .3011138;
netsum += feature2[18] * -.3816631;
netsum +=  .1592132;
netsum += feature3[0] * -.1655201;
netsum += feature3[1] * -2.653964;
netsum += feature3[2] *  .2073672;
netsum += feature3[3] *  .6367522;
netsum += feature3[4] *  8.928083E-02;
netsum += feature3[5] * -.225901;
netsum += feature3[6] *  .1252734;
netsum += feature3[7] *  1.517936;
netsum += feature3[8] * -6.056479E-02;
netsum += feature3[9] * -.8885255;
netsum += feature3[10] *  4.769258E-02;
netsum += feature3[11] *  5.429088E-02;
netsum += feature3[12] * -.3252793;
netsum += feature3[13] *  .3260148;
netsum += feature3[14] *  .0635717;
netsum += feature3[15] * -.3521181;
netsum += feature3[16] * -3.218055E-02;
netsum += feature3[17] * -.1566686;
netsum += feature3[18] *  2.723645E-02;
netsum +=  1.755619E-02;
netsum += feature4[0] * -1.152382E-02;
netsum += feature4[1] *  .2485397;
netsum += feature4[2] *  7.769431E-03;
netsum += feature4[3] * -2.890145E-02;
netsum += feature4[4] * -1.362801;
netsum += feature4[5] * -3.70192E-03;
netsum += feature4[6] *  4.866618E-03;
netsum += feature4[7] *  .4250436;
netsum += feature4[8] *  8.412428E-04;
netsum += feature4[9] *  1.166549E-02;
netsum += feature4[10] * -3.93293E-03;
netsum += feature4[11] *  .076087;
netsum += feature4[12] *  2.836888E-02;
netsum += feature4[13] * -3.49581E-03;
netsum += feature4[14] * -.115839;
netsum += feature4[15] * -.2258672;
netsum += feature4[16] *  1.696067E-02;
netsum += feature4[17] *  6.212657E-02;
netsum += feature4[18] * -.8434866;
outarray[18] = 1 / (1 + exp(-netsum));

netsum = -.3645401;
netsum += feature2[0] * -5.880604E-03;
netsum += feature2[1] *  4.979555E-03;
netsum += feature2[2] * -5.376604E-03;
netsum += feature2[3] * -2.391034E-02;
netsum += feature2[4] * -2.080099E-02;
netsum += feature2[5] *  5.821685E-03;
netsum += feature2[6] * -.1705253;
netsum += feature2[7] * -5.30973E-03;
netsum += feature2[8] * -.2536896;
netsum += feature2[9] * -2.052772E-02;
netsum += feature2[10] * -.106473;
netsum += feature2[11] * -2.440075E-02;
netsum += feature2[12] * -.1098363;
netsum += feature2[13] *  .0543099;
netsum += feature2[14] * -8.930701E-03;
netsum += feature2[15] *  1.694692E-02;
netsum += feature2[16] * -6.075521E-03;
netsum += feature2[17] * -.1317704;
netsum += feature2[18] * -.3916451;
netsum += -.3869346;
netsum += feature3[0] *  .1847824;
netsum += feature3[1] * -3.146054;
netsum += feature3[2] * -.2413964;
netsum += feature3[3] * -1.11833;
netsum += feature3[4] * -.3179062;
netsum += feature3[5] * -1.345856E-02;
netsum += feature3[6] *  1.198195E-02;
netsum += feature3[7] *  1.94397;
netsum += feature3[8] *  6.326617E-02;
netsum += feature3[9] * -.1632584;
netsum += feature3[10] *  4.969986E-02;
netsum += feature3[11] *  4.855357E-02;
netsum += feature3[12] * -5.692903E-02;
netsum += feature3[13] * -1.287376E-02;
netsum += feature3[14] * -2.990626E-02;
netsum += feature3[15] * -.1067872;
netsum += feature3[16] * -3.256503E-02;
netsum += feature3[17] * -.3600588;
netsum += feature3[18] * -2.128281E-02;
netsum += -1.502656E-02;
netsum += feature4[0] *  .0424784;
netsum += feature4[1] *  .2863975;
netsum += feature4[2] * -1.421935E-02;
netsum += feature4[3] * -6.742146E-04;
netsum += feature4[4] *  .3310419;
netsum += feature4[5] *  5.826376E-04;
netsum += feature4[6] *  7.724006E-03;
netsum += feature4[7] *  5.853878E-02;
netsum += feature4[8] *  3.871809E-03;
netsum += feature4[9] * -3.227937E-02;
netsum += feature4[10] * -1.532882E-02;
netsum += feature4[11] *  2.131832E-02;
netsum += feature4[12] *  1.018525E-02;
```

```
netsum += feature4[13] *  3.26338E-03;
netsum += feature4[14] * -.1491566;
netsum += feature4[15] * -.4891309;
netsum += feature4[16] *  .6099927;
netsum += feature4[17] * -4.920115E-02;
netsum += feature4[18] * -.9167876;
outarray[19] = 1 / (1 + exp(-netsum));

netsum = -.1330385;
netsum += feature2[0] *  5.908901E-03;
netsum += feature2[1] * -.0183744;
netsum += feature2[2] * -5.30926E-03;
netsum += feature2[3] * -2.927209E-02;
netsum += feature2[4] *  5.051031E-02;
netsum += feature2[5] * -1.323282E-02;
netsum += feature2[6] * -.3014206;
netsum += feature2[7] *  .0082947;
netsum += feature2[8] *  .1106277;
netsum += feature2[9] *  .1755426;
netsum += feature2[10] * -5.091058E-03;
netsum += feature2[11] * -2.072916E-03;
netsum += feature2[12] * -3.644126E-02;
netsum += feature2[13] * -.151084;
netsum += feature2[14] * -3.245267E-02;
netsum += feature2[15] *  8.13277E-03;
netsum += feature2[16] * -5.310477E-03;
netsum += feature2[17] * -.4587012;
netsum += feature2[18] * -7.953025E-03;
netsum += -.1615407;
netsum += feature3[0] *  .1556157;
netsum += feature3[1] *  .7416637;
netsum += feature3[2] * -3.073458E-02;
netsum += feature3[3] * -.6963644;
netsum += feature3[4] * -.1441081;
netsum += feature3[5] * -2.088196E-02;
netsum += feature3[6] * -7.354046E-02;
netsum += feature3[7] * -.5171103;
netsum += feature3[8] *  1.532974E-02;
netsum += feature3[9] * -.212482;
netsum += feature3[10] * -.2681152;
netsum += feature3[11] * -2.689949E-03;
netsum += feature3[12] *  1.5593;
netsum += feature3[13] * -5.436953E-02;
netsum += feature3[14] * -1.436097;
netsum += feature3[15] * -.1251985;
netsum += feature3[16] * -5.298992E-02;
netsum += feature3[17] * -.5312425;
netsum += feature3[18] * -.1151303;
netsum +=  .1745167;
netsum += feature4[0] *  2.550375E-02;
netsum += feature4[1] * -.4799443;
netsum += feature4[2] * -8.568476E-03;
netsum += feature4[3] *  6.128259E-02;
netsum += feature4[4] *  .4992475;
netsum += feature4[5] * -6.121565E-03;
netsum += feature4[6] *  2.526071E-03;
netsum += feature4[7] * -.1290696;
netsum += feature4[8] *  5.625701E-04;
netsum += feature4[9] *  3.196246E-02;
netsum += feature4[10] * -6.075868E-03;
netsum += feature4[11] * -5.113488E-02;
netsum += feature4[12] * -1.981995E-02;
netsum += feature4[13] *  6.269809E-04;
netsum += feature4[14] *  .2847435;
netsum += feature4[15] *  .1685745;
netsum += feature4[16] *  .4157856;
netsum += feature4[17] * -8.276498E-02;
netsum += feature4[18] *  .7378537;
outarray[20] = 1 / (1 + exp(-netsum));

netsum =  6.464648E-02;
netsum += feature2[0] *  5.293649E-03;
netsum += feature2[1] * -1.614488E-02;
netsum += feature2[2] * -6.579455E-03;
netsum += feature2[3] * -2.927606E-02;
netsum += feature2[4] *  4.886707E-02;
netsum += feature2[5] * -1.321953E-02;
netsum += feature2[6] *  5.629278E-02;
netsum += feature2[7] *  7.217158E-03;
netsum += feature2[8] * -.1252699;
netsum += feature2[9] *  .1719535;
netsum += feature2[10] * -1.060805E-02;
netsum += feature2[11] * -2.271225E-03;
netsum += feature2[12] * -.0420361;
netsum += feature2[13] * -.348491;
netsum += feature2[14] * -3.372176E-02;
netsum += feature2[15] *  5.242061E-04;
netsum += feature2[16] * -4.845611E-03;
netsum += feature2[17] * -.4375074;
netsum += feature2[18] *  1.480331E-02;
netsum += -.2035218;
netsum += feature3[0] * -3.514914E-02;
netsum += feature3[1] *  .7373197;
netsum += feature3[2] * -3.401733E-02;
netsum += feature3[3] * -.6952553;
netsum += feature3[4] * -.1476224;
netsum += feature3[5] * -1.541527E-02;
netsum += feature3[6] * -6.897238E-02;
netsum += feature3[7] * -.5213562;
netsum += feature3[8] *  2.328429E-02;
netsum += feature3[9] * -.2139867;
netsum += feature3[10] * -.2637117;
netsum += feature3[11] *  1.041888E-02;
netsum += feature3[12] *  1.56177;
netsum += feature3[13] * -7.661951E-02;
netsum += feature3[14] * -1.427051;
netsum += feature3[15] * -.1491698;
netsum += feature3[16] * -4.855926E-02;
netsum += feature3[17] * -.5295022;
netsum += feature3[18] * -9.815068E-02;
netsum += -.1456886;
netsum += feature4[0] *  2.596253E-02;
netsum += feature4[1] * -.4718613;
netsum += feature4[2] * -9.141785E-03;
```

```
netsum += feature4[3] *  5.683206E-02;
netsum += feature4[4] *  .456624;
netsum += feature4[5] * -7.519086E-03;
netsum += feature4[6] * -2.715404E-04;
netsum += feature4[7] * -.1328076;
netsum += feature4[8] *  4.789632E-04;
netsum += feature4[9] *  3.301151E-02;
netsum += feature4[10] * -4.851622E-03;
netsum += feature4[11] * -3.619927E-02;
netsum += feature4[12] * -1.926082E-02;
netsum += feature4[13] * -4.332413E-04;
netsum += feature4[14] *  .2932492;
netsum += feature4[15] *  .1872639;
netsum += feature4[16] *  .4199485;
netsum += feature4[17] * -.0762268;
netsum += feature4[18] *  .72846;
outarray[21] = 1 / (1 + exp(-netsum));

netsum =  5.071271E-02;
netsum += feature2[0] *  6.067732E-03;
netsum += feature2[1] * -1.896955E-02;
netsum += feature2[2] * -4.845601E-03;
netsum += feature2[3] * -2.919186E-02;
netsum += feature2[4] *  5.128121E-02;
netsum += feature2[5] * -1.300549E-02;
netsum += feature2[6] * -8.360808E-02;
netsum += feature2[7] *  8.516921E-03;
netsum += feature2[8] * -.1910568;
netsum += feature2[9] *  .1752574;
netsum += feature2[10] * -4.661101E-03;
netsum += feature2[11] * -2.149046E-03;
netsum += feature2[12] * -.0354233;
netsum += feature2[13] * -.1058844;
netsum += feature2[14] * -3.179527E-02;
netsum += feature2[15] *  .0107634;
netsum += feature2[16] * -5.392566E-03;
netsum += feature2[17] * -.4649758;
netsum += feature2[18] * -.0138128;
netsum += -6.182407E-02;
netsum += feature3[0] * -1.443775E-02;
netsum += feature3[1] *  .7413386;
netsum += feature3[2] * -2.861113E-02;
netsum += feature3[3] * -.6975321;
netsum += feature3[4] * -.1429459;
netsum += feature3[5] * -2.214095E-02;
netsum += feature3[6] * -7.687753E-02;
netsum += feature3[7] * -.5142971;
netsum += feature3[8] *  1.302399E-02;
netsum += feature3[9] * -.2118079;
netsum += feature3[10] * -.2679641;
netsum += feature3[11] * -6.837966E-03;
netsum += feature3[12] *  1.557606;
netsum += feature3[13] *  .2046502;
netsum += feature3[14] * -1.435354;
netsum += feature3[15] *  .1509149;
netsum += feature3[16] * -5.435147E-02;
netsum += feature3[17] * -.530643;
netsum += feature3[18] * -.1215217;
netsum += -.1973222;
netsum += feature4[0] *  2.548629E-02;
netsum += feature4[1] * -.4805094;
netsum += feature4[2] * -8.457033E-03;
netsum += feature4[3] *  6.252192E-02;
netsum += feature4[4] *  .5035633;
netsum += feature4[5] * -5.511106E-03;
netsum += feature4[6] *  3.114732E-03;
netsum += feature4[7] * -.1279957;
netsum += feature4[8] *  5.504111E-04;
netsum += feature4[9] *  3.190058E-02;
netsum += feature4[10] * -6.38258E-03;
netsum += feature4[11] * -.0565051;
netsum += feature4[12] * -1.986141E-02;
netsum += feature4[13] *  9.800055E-04;
netsum += feature4[14] *  .282764;
netsum += feature4[15] *  .1583811;
netsum += feature4[16] *  .4147046;
netsum += feature4[17] * -.0834723;
netsum += feature4[18] *  .7387074;
outarray[22] = 1 / (1 + exp(-netsum));

netsum = -1.019337E-02;
netsum += feature2[0] * -2.828362E-03;
netsum += feature2[1] * -2.724678E-03;
netsum += feature2[2] * -6.482209E-04;
netsum += feature2[3] * -4.454184E-03;
netsum += feature2[4] *  9.019101E-03;
netsum += feature2[5] *  9.856169E-07;
netsum += feature2[6] * -.4532567;
netsum += feature2[7] * -9.668206E-03;
netsum += feature2[8] *  4.335722E-02;
netsum += feature2[9] * -3.438002E-02;
netsum += feature2[10] * -2.136722E-02;
netsum += feature2[11] * -3.241172E-03;
netsum += feature2[12] * -5.280835E-02;
netsum += feature2[13] *  7.362878E-02;
netsum += feature2[14] *  5.104686E-04;
netsum += feature2[15] *  2.740593E-02;
netsum += feature2[16] *  6.436678E-03;
netsum += feature2[17] * -8.960912E-02;
netsum += feature2[18] *  .131553;
netsum += -.2882466;
netsum += feature3[0] *  .1040232;
netsum += feature3[1] *  .9738329;
netsum += feature3[2] * -2.331804E-02;
netsum += feature3[3] * -3.006429E-03;
netsum += feature3[4] *  4.392762E-02;
netsum += feature3[5] * -2.120778E-02;
netsum += feature3[6] * -1.645108E-02;
netsum += feature3[7] *  .4937124;
netsum += feature3[8] *  2.722508E-02;
netsum += feature3[9] * -.2029674;
netsum += feature3[10] * -5.477873E-02;
netsum += feature3[11] *  4.566481E-02;
netsum += feature3[12] *  .1028695;
```

```
netsum += feature3[13] *  .0572931;
netsum += feature3[14] * -.2678148;
netsum += feature3[15] *  .3520798;
netsum += feature3[16] * -2.203318E-02;
netsum += feature3[17] *  .9335116;
netsum += feature3[18] *  6.903966E-02;
netsum +=  .1970459;
netsum += feature4[0] * -1.780301E-03;
netsum += feature4[1] * -.8211353;
netsum += feature4[2] * -5.183973E-03;
netsum += feature4[3] *  3.856967E-03;
netsum += feature4[4] *  .1050517;
netsum += feature4[5] *  5.154572E-03;
netsum += feature4[6] * -5.109112E-02;
netsum += feature4[7] *  2.453217E-02;
netsum += feature4[8] * -7.252827E-03;
netsum += feature4[9] * -9.804659E-03;
netsum += feature4[10] *  6.837967E-03;
netsum += feature4[11] * -6.984262E-02;
netsum += feature4[12] * -9.23703E-04;
netsum += feature4[13] * -2.697049E-03;
netsum += feature4[14] * -.1449026;
netsum += feature4[15] * -.1397818;
netsum += feature4[16] *  4.303748E-02;
netsum += feature4[17] *  6.380226E-02;
netsum += feature4[18] *  1.152791;
outarray[23] = 1 / (1 + exp(-netsum));

netsum = -.3122944;
netsum += feature2[0] * -3.789238E-02;
netsum += feature2[1] * -3.504643E-02;
netsum += feature2[2] *  7.219229E-03;
netsum += feature2[3] *  .0175929;
netsum += feature2[4] *  9.624824E-03;
netsum += feature2[5] *  1.046722E-02;
netsum += feature2[6] *  .1171711;
netsum += feature2[7] * -3.752582E-03;
netsum += feature2[8] *  2.420199E-02;
netsum += feature2[9] * -5.605888E-02;
netsum += feature2[10] *  2.064347E-02;
netsum += feature2[11] *  1.700676E-02;
netsum += feature2[12] *  2.281494E-02;
netsum += feature2[13] *  .154284;
netsum += feature2[14] *  5.100555E-03;
netsum += feature2[15] *  .1421454;
netsum += feature2[16] *  8.435206E-03;
netsum += feature2[17] *  .1017169;
netsum += feature2[18] *  .1983804;
netsum +=  .1682727;
netsum += feature3[0] *  .4427628;
netsum += feature3[1] *  9.673925E-03;
netsum += feature3[2] *  .4208697;
netsum += feature3[3] *  2.513176;
netsum += feature3[4] *  .2669837;
netsum += feature3[5] * -.282349;
netsum += feature3[6] * -1.634625E-02;
netsum += feature3[7] * -3.071839E-02;
netsum += feature3[8] * -6.815609E-02;
netsum += feature3[9] * -1.248089;
netsum += feature3[10] *  1.692409E-02;
netsum += feature3[11] *  7.682106E-02;
netsum += feature3[12] * -.1316109;
netsum += feature3[13] * -.1090188;
netsum += feature3[14] *  .2108732;
netsum += feature3[15] *  6.049759E-02;
netsum += feature3[16] *  8.040012E-04;
netsum += feature3[17] *  .1384543;
netsum += feature3[18] *  .1795705;
netsum += -1.164293E-03;
netsum += feature4[0] * -3.173422E-02;
netsum += feature4[1] *  6.851327E-02;
netsum += feature4[2] *  4.846563E-03;
netsum += feature4[3] * -3.705212E-03;
netsum += feature4[4] * -.5719605;
netsum += feature4[5] *  5.037885E-04;
netsum += feature4[6] * -4.702201E-02;
netsum += feature4[7] *  .4041041;
netsum += feature4[8] * -1.276442E-02;
netsum += feature4[9] * -1.496587E-02;
netsum += feature4[10] *  7.056044E-04;
netsum += feature4[11] * -3.150378E-02;
netsum += feature4[12] *  1.618786E-02;
netsum += feature4[13] * -2.304204E-03;
netsum += feature4[14] *  8.502305E-02;
netsum += feature4[15] *  .4848588;
netsum += feature4[16] * -.4873836;
netsum += feature4[17] *  .125178;
netsum += feature4[18] * -.1692997;
outarray[24] = 1 / (1 + exp(-netsum));

netsum = -9.236737E-02;
netsum += feature2[0] * -3.752801E-02;
netsum += feature2[1] * -3.570294E-02;
netsum += feature2[2] *  7.408828E-03;
netsum += feature2[3] *  1.744795E-02;
netsum += feature2[4] *  9.752621E-03;
netsum += feature2[5] *  .0105676;
netsum += feature2[6] * -4.908881E-02;
netsum += feature2[7] * -3.478201E-03;
netsum += feature2[8] * -.3365213;
netsum += feature2[9] * -5.518717E-02;
netsum += feature2[10] *  2.209922E-02;
netsum += feature2[11] *  .017046;
netsum += feature2[12] *  .0233209;
netsum += feature2[13] *  .2049251;
netsum += feature2[14] *  5.254865E-03;
netsum += feature2[15] *  .1436766;
netsum += feature2[16] *  8.260475E-03;
netsum += feature2[17] *  9.621008E-02;
netsum += feature2[18] *  .1950952;
netsum += -.1718964;
netsum += feature3[0] *  .1720541;
netsum += feature3[1] *  1.009643E-02;
netsum += feature3[2] *  .416838;
```

```
netsum += feature3[3] *  2.513202;
netsum += feature3[4] *  .2683341;
netsum += feature3[5] * -.28364;
netsum += feature3[6] * -1.954983E-02;
netsum += feature3[7] * -2.829778E-02;
netsum += feature3[8] * -6.952421E-02;
netsum += feature3[9] * -1.254198;
netsum += feature3[10] *  1.550516E-02;
netsum += feature3[11] *  7.537408E-02;
netsum += feature3[12] * -.1323619;
netsum += feature3[13] *  .1919348;
netsum += feature3[14] *  .212317;
netsum += feature3[15] *  .0247201;
netsum += feature3[16] * -6.161612E-04;
netsum += feature3[17] *  .1404164;
netsum += feature3[18] *  .1758579;
netsum +=  9.551171E-03;
netsum += feature4[0] * -3.186594E-02;
netsum += feature4[1] *  .0663258;
netsum += feature4[2] *  4.839708E-03;
netsum += feature4[3] * -2.813065E-03;
netsum += feature4[4] * -.5743002;
netsum += feature4[5] *  8.612644E-04;
netsum += feature4[6] * -4.696162E-02;
netsum += feature4[7] *  .4054655;
netsum += feature4[8] * -1.287556E-02;
netsum += feature4[9] * -.0152583;
netsum += feature4[10] *  8.121975E-04;
netsum += feature4[11] * -3.678751E-02;
netsum += feature4[12] *  1.609473E-02;
netsum += feature4[13] * -2.160548E-03;
netsum += feature4[14] *  8.194377E-02;
netsum += feature4[15] *  .485461;
netsum += feature4[16] * -.4877326;
netsum += feature4[17] *  .1238581;
netsum += feature4[18] * -.1674188;
outarray[25] = 1 / (1 + exp(-netsum));

netsum = -.2007456;
netsum += feature2[0] * -3.718369E-02;
netsum += feature2[1] * -3.629261E-02;
netsum += feature2[2] *  7.706334E-03;
netsum += feature2[3] *  1.768165E-02;
netsum += feature2[4] *  9.845636E-03;
netsum += feature2[5] *  1.052609E-02;
netsum += feature2[6] *  5.981145E-02;
netsum += feature2[7] * -3.191796E-03;
netsum += feature2[8] * -.1822996;
netsum += feature2[9] * -5.466441E-02;
netsum += feature2[10] *  2.331802E-02;
netsum += feature2[11] *  1.698745E-02;
netsum += feature2[12] *  2.349967E-02;
netsum += feature2[13] *  .239305;
netsum += feature2[14] *  5.708414E-03;
netsum += feature2[15] *  .1449835;
netsum += feature2[16] *  8.342198E-03;
netsum += feature2[17] *  8.719796E-02;
netsum += feature2[18] *  .1909012;
netsum +=  .1470199;
netsum += feature3[0] *  .37385;
netsum += feature3[1] *  8.414033E-03;
netsum += feature3[2] *  .4126568;
netsum += feature3[3] *  2.513378;
netsum += feature3[4] *  .2694483;
netsum += feature3[5] * -.2843841;
netsum += feature3[6] * -1.975059E-02;
netsum += feature3[7] * -2.514856E-02;
netsum += feature3[8] * -7.101224E-02;
netsum += feature3[9] * -1.259662;
netsum += feature3[10] *  1.581826E-02;
netsum += feature3[11] *  7.277916E-02;
netsum += feature3[12] * -.1342235;
netsum += feature3[13] *  .1007254;
netsum += feature3[14] *  .2126826;
netsum += feature3[15] *  .1183599;
netsum += feature3[16] * -1.118577E-03;
netsum += feature3[17] *  .1397562;
netsum += feature3[18] *  .1700161;
netsum += -.1134883;
netsum += feature4[0] * -3.206034E-02;
netsum += feature4[1] *  6.654444E-02;
netsum += feature4[2] *  4.966387E-03;
netsum += feature4[3] * -2.502932E-03;
netsum += feature4[4] * -.555531;
netsum += feature4[5] *  1.520259E-03;
netsum += feature4[6] * -4.407148E-02;
netsum += feature4[7] *  .4067681;
netsum += feature4[8] * -1.294899E-02;
netsum += feature4[9] * -1.534331E-02;
netsum += feature4[10] *  1.126536E-03;
netsum += feature4[11] * -3.740846E-02;
netsum += feature4[12] *  1.606618E-02;
netsum += feature4[13] * -1.958228E-03;
netsum += feature4[14] *  7.989404E-02;
netsum += feature4[15] *  .4789284;
netsum += feature4[16] * -.4884964;
netsum += feature4[17] *  .1228032;
netsum += feature4[18] * -.1665069;
outarray[26] = 1 / (1 + exp(-netsum));

netsum =  8.774314E-02;
netsum += feature2[0] * -3.869719E-02;
netsum += feature2[1] * -3.220917E-02;
netsum += feature2[2] *  8.045707E-03;
netsum += feature2[3] *  1.472424E-02;
netsum += feature2[4] *  8.122072E-03;
netsum += feature2[5] *  .0151474;
netsum += feature2[6] * -.1653514;
netsum += feature2[7] * -3.972239E-03;
netsum += feature2[8] * -.2144084;
netsum += feature2[9] * -5.885524E-02;
netsum += feature2[10] *  1.722161E-02;
netsum += feature2[11] *  1.867886E-02;
netsum += feature2[12] *  2.193341E-02;
```

```
netsum += feature2[13] * -7.765035E-02;
netsum += feature2[14] *  2.213621E-03;
netsum += feature2[15] *  .1318041;
netsum += feature2[16] *  0;
netsum += feature2[17] *  .130434;
netsum += feature2[18] *  .2349476;
netsum +=  .2297904;
netsum += feature3[0] *  .3061219;
netsum += feature3[1] *  3.596778E-03;
netsum += feature3[2] *  .4142311;
netsum += feature3[3] *  2.514005;
netsum += feature3[4] *  .2642949;
netsum += feature3[5] * -.2748903;
netsum += feature3[6] * -9.869166E-03;
netsum += feature3[7] * -.0357837;
netsum += feature3[8] * -6.040112E-02;
netsum += feature3[9] * -1.252432;
netsum += feature3[10] *  .0196349;
netsum += feature3[11] *  8.905173E-02;
netsum += feature3[12] * -.1241778;
netsum += feature3[13] * -.2457282;
netsum += feature3[14] *  .2242621;
netsum += feature3[15] * -9.301877E-03;
netsum += feature3[16] *  6.327328E-03;
netsum += feature3[17] *  .1384088;
netsum += feature3[18] *  .2093181;
netsum +=  .1508813;
netsum += feature4[0] * -3.002922E-02;
netsum += feature4[1] *  7.609858E-02;
netsum += feature4[2] *  3.00997E-03;
netsum += feature4[3] * -8.93079E-03;
netsum += feature4[4] * -.6317115;
netsum += feature4[5] * -1.104471E-03;
netsum += feature4[6] * -5.918034E-02;
netsum += feature4[7] *  .3987071;
netsum += feature4[8] * -1.107423E-02;
netsum += feature4[9] * -1.441954E-02;
netsum += feature4[10] *  1.811142E-04;
netsum += feature4[11] * -1.851733E-02;
netsum += feature4[12] *  1.690402E-02;
netsum += feature4[13] * -2.881045E-03;
netsum += feature4[14] *  8.730357E-02;
netsum += feature4[15] *  .5210128;
netsum += feature4[16] * -.4821889;
netsum += feature4[17] *  .1310595;
netsum += feature4[18] * -.1768522;
outarray[27] = 1 / (1 + exp(-netsum));

netsum = -.1274231;
netsum += feature2[0] *  3.253035E-02;
netsum += feature2[1] *  3.157623E-03;
netsum += feature2[2] *  2.186258E-03;
netsum += feature2[3] * -3.338064E-02;
netsum += feature2[4] *  3.523182E-02;
netsum += feature2[5] * -3.170617E-03;
netsum += feature2[6] * -.1295404;
netsum += feature2[7] *  2.810675E-02;
netsum += feature2[8] * -6.033218E-02;
netsum += feature2[9] *  .1920091;
netsum += feature2[10] * -7.512487E-02;
netsum += feature2[11] * -2.726485E-02;
netsum += feature2[12] * -.0173829;
netsum += feature2[13] *  .1348482;
netsum += feature2[14] * -2.02684E-03;
netsum += feature2[15] *  1.130728E-02;
netsum += feature2[16] *  5.677994E-03;
netsum += feature2[17] * -.420643;
netsum += feature2[18] *  .1352548;
netsum +=  5.142139E-02;
netsum += feature3[0] *  8.186588E-02;
netsum += feature3[1] * -.1375047;
netsum += feature3[2] * -.767574;
netsum += feature3[3] * -3.205522;
netsum += feature3[4] * -.3823555;
netsum += feature3[5] *  4.441877E-02;
netsum += feature3[6] * -2.707238E-02;
netsum += feature3[7] * -.8249736;
netsum += feature3[8] *  3.387453E-02;
netsum += feature3[9] *  .241358;
netsum += feature3[10] * -3.347418E-02;
netsum += feature3[11] * -3.570744E-02;
netsum += feature3[12] *  .2908891;
netsum += feature3[13] * -2.873508E-02;
netsum += feature3[14] * -.2062142;
netsum += feature3[15] *  6.492846E-02;
netsum += feature3[16] * -6.038167E-02;
netsum += feature3[17] * -.2466712;
netsum += feature3[18] * -.1160601;
netsum +=  .1962309;
netsum += feature4[0] *  .0412099;
netsum += feature4[1] *  7.109867E-02;
netsum += feature4[2] * -2.063689E-02;
netsum += feature4[3] *  5.699195E-02;
netsum += feature4[4] *  .3500669;
netsum += feature4[5] *  4.489096E-03;
netsum += feature4[6] *  1.675969E-03;
netsum += feature4[7] * -.2433105;
netsum += feature4[8] *  2.78588E-04;
netsum += feature4[9] *  .0529643;
netsum += feature4[10] * -2.265931E-02;
netsum += feature4[11] * -1.426175E-02;
netsum += feature4[12] * -9.704468E-03;
netsum += feature4[13] * -5.106063E-03;
netsum += feature4[14] *  .1252566;
netsum += feature4[15] * -.1921712;
netsum += feature4[16] *  .6889777;
netsum += feature4[17] * -.221988;
netsum += feature4[18] * -.2048055;
outarray[28] = 1 / (1 + exp(-netsum));

netsum = -8.721392E-02;
netsum += feature2[0] *  3.145462E-02;
netsum += feature2[1] *  5.237772E-03;
netsum += feature2[2] *  1.270802E-03;
```

```
netsum += feature2[3] * -3.371575E-02;
netsum += feature2[4] *  .0345672;
netsum += feature2[5] * -3.324857E-03;
netsum += feature2[6] * -.1145221;
netsum += feature2[7] *  2.733505E-02;
netsum += feature2[8] *  .2135018;
netsum += feature2[9] *  .1900165;
netsum += feature2[10] * -7.841858E-02;
netsum += feature2[11] * -2.714793E-02;
netsum += feature2[12] * -1.827779E-02;
netsum += feature2[13] *  3.980235E-03;
netsum += feature2[14] * -3.391314E-03;
netsum += feature2[15] *  5.058254E-03;
netsum += feature2[16] *  5.763866E-03;
netsum += feature2[17] * -.3963123;
netsum += feature2[18] *  .1462473;
netsum += -.2195139;
netsum += feature3[0] * -2.244889E-03;
netsum += feature3[1] * -.1334475;
netsum += feature3[2] * -.758545;
netsum += feature3[3] * -3.206589;
netsum += feature3[4] * -.3857868;
netsum += feature3[5] *  4.767421E-02;
netsum += feature3[6] * -2.294438E-02;
netsum += feature3[7] * -.8359178;
netsum += feature3[8] *  3.904938E-02;
netsum += feature3[9] *  .2546031;
netsum += feature3[10] * -3.204324E-02;
netsum += feature3[11] * -2.797879E-02;
netsum += feature3[12] *  .2949451;
netsum += feature3[13] * -.2956303;
netsum += feature3[14] * -.2073268;
netsum += feature3[15] * -.2276366;
netsum += feature3[16] * -.0565664;
netsum += feature3[17] * -.2475155;
netsum += feature3[18] * -9.983792E-02;
netsum +=  .1397811;
netsum += feature4[0] *  4.169701E-02;
netsum += feature4[1] *  7.297374E-02;
netsum += feature4[2] * -2.090374E-02;
netsum += feature4[3] *  5.425686E-02;
netsum += feature4[4] *  .3191333;
netsum += feature4[5] *  2.341301E-03;
netsum += feature4[6] * -4.499835E-03;
netsum += feature4[7] * -.247718;
netsum += feature4[8] *  5.920499E-04;
netsum += feature4[9] *  5.346769E-02;
netsum += feature4[10] * -2.303151E-02;
netsum += feature4[11] * -7.001314E-03;
netsum += feature4[12] * -9.40597E-03;
netsum += feature4[13] * -5.903575E-03;
netsum += feature4[14] *  .130534;
netsum += feature4[15] * -.1838397;
netsum += feature4[16] *  .6908771;
netsum += feature4[17] * -.2185304;
netsum += feature4[18] * -.2080275;
outarray[29] = 1 / (1 + exp(-netsum));
```

```
netsum = -.1049099;
netsum += feature2[0] *  3.224792E-02;
netsum += feature2[1] *  3.542572E-03;
netsum += feature2[2] *  2.098637E-03;
netsum += feature2[3] * -3.331335E-02;
netsum += feature2[4] *  3.547854E-02;
netsum += feature2[5] * -3.091403E-03;
netsum += feature2[6] *  6.924054E-02;
netsum += feature2[7] *  2.800102E-02;
netsum += feature2[8] *  .1593615;
netsum += feature2[9] *  .1911495;
netsum += feature2[10] * -7.651635E-02;
netsum += feature2[11] * -2.725445E-02;
netsum += feature2[12] * -1.784223E-02;
netsum += feature2[13] *  .1198315;
netsum += feature2[14] * -2.066668E-03;
netsum += feature2[15] *  1.170835E-02;
netsum += feature2[16] *  5.774672E-03;
netsum += feature2[17] * -.412151;
netsum += feature2[18] *  .1325816;
netsum += -7.680112E-02;
netsum += feature3[0] *  6.229503E-02;
netsum += feature3[1] * -.133112;
netsum += feature3[2] * -.7601357;
netsum += feature3[3] * -3.206724;
netsum += feature3[4] * -.3836863;
netsum += feature3[5] *  .0446044;
netsum += feature3[6] * -2.665149E-02;
netsum += feature3[7] * -.831324;
netsum += feature3[8] *  3.445199E-02;
netsum += feature3[9] *  .2513828;
netsum += feature3[10] * -3.278296E-02;
netsum += feature3[11] * -3.491934E-02;
netsum += feature3[12] *  .2910581;
netsum += feature3[13] * -.1032448;
netsum += feature3[14] * -.209239;
netsum += feature3[15] *  .1118643;
netsum += feature3[16] * -6.068302E-02;
netsum += feature3[17] * -.2470226;
netsum += feature3[18] * -.114824;
netsum +=  9.679858E-03;
netsum += feature4[0] *  4.137009E-02;
netsum += feature4[1] *  7.059075E-02;
netsum += feature4[2] * -2.067519E-02;
netsum += feature4[3] *  5.738097E-02;
netsum += feature4[4] *  .3467701;
netsum += feature4[5] *  4.259491E-03;
netsum += feature4[6] *  1.418748E-03;
netsum += feature4[7] * -.244061;
netsum += feature4[8] *  2.647525E-04;
netsum += feature4[9] *  .053313;
netsum += feature4[10] * -2.256536E-02;
netsum += feature4[11] * -1.384786E-02;
netsum += feature4[12] * -9.701439E-03;
netsum += feature4[13] * -5.15424E-03;
netsum += feature4[14] *  .1274642;
```

```
netsum += feature4[15] * -.2001775;
netsum += feature4[16] *  .6893013;
netsum += feature4[17] * -.2215896;
netsum += feature4[18] * -.2047535;
outarray[30] = 1 / (1 + exp(-netsum));

netsum = -.0600148;
netsum += feature2[0] *  .0320106;
netsum += feature2[1] *  4.266013E-03;
netsum += feature2[2] *  1.530765E-03;
netsum += feature2[3] * -3.355604E-02;
netsum += feature2[4] *  3.493681E-02;
netsum += feature2[5] * -3.214545E-03;
netsum += feature2[6] *  .2538828;
netsum += feature2[7] *  2.726262E-02;
netsum += feature2[8] *  6.799944E-02;
netsum += feature2[9] *  .1892089;
netsum += feature2[10] * -7.709712E-02;
netsum += feature2[11] * -2.742269E-02;
netsum += feature2[12] * -1.929353E-02;
netsum += feature2[13] *  2.089461E-02;
netsum += feature2[14] * -2.818058E-03;
netsum += feature2[15] *  5.670004E-03;
netsum += feature2[16] *  5.861576E-03;
netsum += feature2[17] * -.4166422;
netsum += feature2[18] *  .1502716;
netsum += -7.031105E-02;
netsum += feature3[0] * -.1168927;
netsum += feature3[1] * -.1451836;
netsum += feature3[2] * -.7755423;
netsum += feature3[3] * -3.20503;
netsum += feature3[4] * -.3826703;
netsum += feature3[5] *  4.821804E-02;
netsum += feature3[6] * -2.728586E-02;
netsum += feature3[7] * -.8194131;
netsum += feature3[8] *  3.751343E-02;
netsum += feature3[9] *  .2308932;
netsum += feature3[10] * -.029953;
netsum += feature3[11] * -3.092601E-02;
netsum += feature3[12] *  .2922717;
netsum += feature3[13] * -.1344729;
netsum += feature3[14] * -.194813;
netsum += feature3[15] * -2.289283E-02;
netsum += feature3[16] * -5.667686E-02;
netsum += feature3[17] * -.244749;
netsum += feature3[18] * -.1070429;
netsum += -.3579146;
netsum += feature4[0] *  4.154317E-02;
netsum += feature4[1] *  7.856027E-02;
netsum += feature4[2] * -2.094015E-02;
netsum += feature4[3] *  5.353627E-02;
netsum += feature4[4] *  .3263208;
netsum += feature4[5] *  3.761722E-03;
netsum += feature4[6] * -1.523591E-03;
netsum += feature4[7] * -.2451473;
netsum += feature4[8] *  5.432297E-04;
netsum += feature4[9] *  .0527739;
netsum += feature4[10] * -2.272693E-02;
netsum += feature4[11] * -8.427771E-03;
netsum += feature4[12] * -9.298348E-03;
netsum += feature4[13] * -5.627025E-03;
netsum += feature4[14] *  .1286326;
netsum += feature4[15] * -.1791366;
netsum += feature4[16] *  .6907213;
netsum += feature4[17] * -.2170322;
netsum += feature4[18] * -.21197;
outarray[31] = 1 / (1 + exp(-netsum));

netsum = -9.187103E-02;
netsum += feature2[0] * -3.883305E-02;
netsum += feature2[1] * -7.32707E-03;
netsum += feature2[2] *  9.926917E-03;
netsum += feature2[3] *  1.782686E-02;
netsum += feature2[4] *  3.203652E-02;
netsum += feature2[5] *  1.963002E-02;
netsum += feature2[6] * -.3605091;
netsum += feature2[7] * -1.140917E-02;
netsum += feature2[8] *  .1280287;
netsum += feature2[9] * -.0982013;
netsum += feature2[10] *  7.917552E-02;
netsum += feature2[11] *  2.230869E-02;
netsum += feature2[12] * -1.378734E-03;
netsum += feature2[13] *  9.877175E-02;
netsum += feature2[14] *  3.013075E-03;
netsum += feature2[15] *  7.591786E-02;
netsum += feature2[16] * -5.57133E-03;
netsum += feature2[17] *  5.680544E-02;
netsum += feature2[18] * -2.879705E-02;
netsum +=  .1132207;
netsum += feature3[0] *  .4515386;
netsum += feature3[1] *  2.573948E-03;
netsum += feature3[2] *  .8806961;
netsum += feature3[3] *  3.22475;
netsum += feature3[4] *  .3326634;
netsum += feature3[5] * -.1653738;
netsum += feature3[6] * -7.589861E-03;
netsum += feature3[7] *  .438537;
netsum += feature3[8] * -2.001564E-02;
netsum += feature3[9] * -1.451641;
netsum += feature3[10] * -5.381029E-04;
netsum += feature3[11] *  9.798942E-02;
netsum += feature3[12] * -5.626449E-02;
netsum += feature3[13] *  .1201011;
netsum += feature3[14] *  .1615504;
netsum += feature3[15] *  .3373037;
netsum += feature3[16] *  2.636654E-03;
netsum += feature3[17] *  .1153027;
netsum += feature3[18] * -3.084151E-02;
netsum += -3.197893E-02;
netsum += feature4[0] * -3.238379E-02;
netsum += feature4[1] *  5.923545E-02;
netsum += feature4[2] *  1.029504E-02;
netsum += feature4[3] * -3.282154E-03;
netsum += feature4[4] * -.2670783;
```

```
netsum += feature4[5] * -1.135147E-02;
netsum += feature4[6] * -1.735487E-02;
netsum += feature4[7] *  .1019852;
netsum += feature4[8] * -7.165587E-03;
netsum += feature4[9] * -.0462665;
netsum += feature4[10] * -9.699545E-04;
netsum += feature4[11] * -6.866446E-03;
netsum += feature4[12] *  2.388082E-04;
netsum += feature4[13] * -2.260559E-03;
netsum += feature4[14] *  7.252646E-02;
netsum += feature4[15] *  .403869;
netsum += feature4[16] * -.5219404;
netsum += feature4[17] *  .1437844;
netsum += feature4[18] * -7.749186E-02;
outarray[32] = 1 / (1 + exp(-netsum));

netsum = -8.866863E-02;
netsum += feature2[0] * -3.773298E-02;
netsum += feature2[1] * -6.81766E-03;
netsum += feature2[2] *  1.103521E-02;
netsum += feature2[3] *  1.784109E-02;
netsum += feature2[4] *  3.712293E-02;
netsum += feature2[5] *  2.012131E-02;
netsum += feature2[6] *  4.376305E-02;
netsum += feature2[7] * -1.033936E-02;
netsum += feature2[8] * -.2333352;
netsum += feature2[9] * -9.247474E-02;
netsum += feature2[10] *  8.345732E-02;
netsum += feature2[11] *  2.277656E-02;
netsum += feature2[12] *  2.171639E-03;
netsum += feature2[13] *  8.376247E-02;
netsum += feature2[14] *  3.957349E-03;
netsum += feature2[15] *  7.468638E-02;
netsum += feature2[16] * -5.614753E-03;
netsum += feature2[17] *  3.534267E-02;
netsum += feature2[18] * -8.559051E-03;
netsum += -.3158197;
netsum += feature3[0] *  5.869245E-02;
netsum += feature3[1] *  .1394857;
netsum += feature3[2] *  .8916055;
netsum += feature3[3] *  3.228801;
netsum += feature3[4] *  .3297859;
netsum += feature3[5] * -.1607296;
netsum += feature3[6] * -1.620557E-02;
netsum += feature3[7] *  .3345336;
netsum += feature3[8] * -1.907449E-02;
netsum += feature3[9] * -1.445491;
netsum += feature3[10] * -2.975318E-03;
netsum += feature3[11] *  9.740204E-02;
netsum += feature3[12] * -4.736935E-02;
netsum += feature3[13] * -.2228625;
netsum += feature3[14] *  .1696977;
netsum += feature3[15] *  .1688317;
netsum += feature3[16] *  2.656265E-03;
netsum += feature3[17] *  .1333435;
netsum += feature3[18] * -4.492353E-02;
netsum +=  .1236596;

netsum += feature4[0] * -3.255408E-02;
netsum += feature4[1] *  5.453941E-02;
netsum += feature4[2] *  9.791377E-03;
netsum += feature4[3] * -7.183383E-04;
netsum += feature4[4] * -.2663131;
netsum += feature4[5] * -.0120889;
netsum += feature4[6] * -1.838974E-02;
netsum += feature4[7] *  7.985627E-02;
netsum += feature4[8] * -7.460373E-03;
netsum += feature4[9] * -4.498733E-02;
netsum += feature4[10] * -1.633276E-03;
netsum += feature4[11] * -1.240743E-02;
netsum += feature4[12] * -1.817619E-03;
netsum += feature4[13] * -2.314383E-03;
netsum += feature4[14] *  9.125392E-02;
netsum += feature4[15] *  .4365337;
netsum += feature4[16] * -.5234548;
netsum += feature4[17] *  .14202;
netsum += feature4[18] * -4.766943E-02;
outarray[33] = 1 / (1 + exp(-netsum));

netsum = -.2227241;
netsum += feature2[0] * -3.540196E-02;
netsum += feature2[1] * -7.719271E-03;
netsum += feature2[2] *  1.348421E-02;
netsum += feature2[3] *  1.771083E-02;
netsum += feature2[4] *  4.275906E-02;
netsum += feature2[5] *  2.071923E-02;
netsum += feature2[6] * -.1649103;
netsum += feature2[7] * -7.771479E-03;
netsum += feature2[8] * -.1036992;
netsum += feature2[9] * -8.132064E-02;
netsum += feature2[10] *  9.001525E-02;
netsum += feature2[11] *  2.296692E-02;
netsum += feature2[12] *  7.738755E-03;
netsum += feature2[13] *  .2343035;
netsum += feature2[14] *  6.810531E-03;
netsum += feature2[15] *  8.107398E-02;
netsum += feature2[16] * -5.56872E-03;
netsum += feature2[17] * -5.615581E-05;
netsum += feature2[18] * -1.760276E-02;
netsum += -.1380015;
netsum += feature3[0] *  .3783109;
netsum += feature3[1] *  .2430978;
netsum += feature3[2] *  .8939815;
netsum += feature3[3] *  3.196702;
netsum += feature3[4] *  .3164403;
netsum += feature3[5] * -.1580674;
netsum += feature3[6] * -2.164796E-02;
netsum += feature3[7] *  .2473669;
netsum += feature3[8] * -2.192419E-02;
netsum += feature3[9] * -1.412001;
netsum += feature3[10] * -9.522592E-03;
netsum += feature3[11] *  8.948227E-02;
netsum += feature3[12] * -4.369787E-02;
netsum += feature3[13] *  .2185118;
netsum += feature3[14] *  .1542667;
```

```
netsum += feature3[15] * .3249124;
netsum += feature3[16] * -1.842868E-03;
netsum += feature3[17] * .1378988;
netsum += feature3[18] * -7.313602E-02;
netsum += .094354;
netsum += feature4[0] * -3.234297E-02;
netsum += feature4[1] * 4.015154E-02;
netsum += feature4[2] * 9.379314E-03;
netsum += feature4[3] * 7.036134E-03;
netsum += feature4[4] * -.1962309;
netsum += feature4[5] * -1.097904E-02;
netsum += feature4[6] * -1.227955E-02;
netsum += feature4[7] * 5.833543E-02;
netsum += feature4[8] * -8.06995E-03;
netsum += feature4[9] * -4.357747E-02;
netsum += feature4[10] * -2.480199E-03;
netsum += feature4[11] * -2.198373E-02;
netsum += feature4[12] * -4.659791E-03;
netsum += feature4[13] * -1.121371E-03;
netsum += feature4[14] * 9.618979E-02;
netsum += feature4[15] * .4277457;
netsum += feature4[16] * -.5113068;
netsum += feature4[17] * .1287254;
netsum += feature4[18] * -1.615274E-02;
outarray[34] = 1 / (1 + exp(-netsum));

netsum = 6.950495E-02;
netsum += feature2[0] * -.0403524;
netsum += feature2[1] * -4.309195E-02;
netsum += feature2[2] * 5.409617E-03;
netsum += feature2[3] * 2.046173E-02;
netsum += feature2[4] * -3.704919E-02;
netsum += feature2[5] * -4.588008E-04;
netsum += feature2[6] * -.22891;
netsum += feature2[7] * -1.338271E-02;
netsum += feature2[8] * .2645263;
netsum += feature2[9] * -.1131284;
netsum += feature2[10] * -.0746823;
netsum += feature2[11] * 1.403683E-02;
netsum += feature2[12] * 2.116038E-03;
netsum += feature2[13] * .3227743;
netsum += feature2[14] * 1.596602E-02;
netsum += feature2[15] * .1514209;
netsum += feature2[16] * 1.011615E-02;
netsum += feature2[17] * .2246342;
netsum += feature2[18] * -.0265432;
netsum += .2033064;
netsum += feature3[0] * .215511;
netsum += feature3[1] * -.7150341;
netsum += feature3[2] * -3.066692E-02;
netsum += feature3[3] * 1.47271;
netsum += feature3[4] * .2896477;
netsum += feature3[5] * -.3192789;
netsum += feature3[6] * 8.866049E-02;
netsum += feature3[7] * .0220762;
netsum += feature3[8] * -.1169747;
netsum += feature3[9] * -1.353614;
netsum += feature3[10] * 5.848794E-02;
netsum += feature3[11] * 3.141659E-02;
netsum += feature3[12] * -.3637167;
netsum += feature3[13] * .2087922;
netsum += feature3[14] * .1522779;
netsum += feature3[15] * -.1136518;
netsum += feature3[16] * -.0169659;
netsum += feature3[17] * 4.874049E-02;
netsum += feature3[18] * .2151921;
netsum += 6.659403E-02;
netsum += feature4[0] * -3.796463E-02;
netsum += feature4[1] * .1130186;
netsum += feature4[2] * 1.090023E-02;
netsum += feature4[3] * -1.805075E-02;
netsum += feature4[4] * -1.566883;
netsum += feature4[5] * 1.199658E-02;
netsum += feature4[6] * 6.731355E-03;
netsum += feature4[7] * .5913331;
netsum += feature4[8] * -1.318167E-02;
netsum += feature4[9] * 3.633325E-02;
netsum += feature4[10] * 3.964835E-03;
netsum += feature4[11] * .0552497;
netsum += feature4[12] * 2.909184E-02;
netsum += feature4[13] * 0;
netsum += feature4[14] * -5.029957E-02;
netsum += feature4[15] * .2113835;
netsum += feature4[16] * -.3619273;
netsum += feature4[17] * .157466;
netsum += feature4[18] * -.4578046;
outarray[35] = 1 / (1 + exp(-netsum));

netsum = .371611;
netsum += feature2[0] * -7.459107E-02;
netsum += feature2[1] * .2548573;
netsum += feature2[2] * -2.654725E-03;
netsum += feature2[3] * .4486666;
netsum += feature2[4] * -7.329342E-02;
netsum += feature2[5] * 0;
netsum += feature2[6] * .49707;
netsum += feature2[7] * -.5149158;
netsum += feature2[8] * 4.688228E-02;
netsum += feature2[9] * 1.531357;
netsum += feature2[10] * -.1441816;
netsum += feature2[11] * -.2090809;
netsum += feature2[12] * -1.751352;
netsum += feature2[13] * 4.716155E-02;
netsum += feature2[14] * .3808854;
netsum += feature2[15] * 3.324327;
netsum += feature2[16] * 2.651115E-02;
netsum += feature2[17] * -1.066293;
netsum += feature2[18] * 1.127647;
netsum += .3136841;
netsum += feature3[0] * -.3858018;
netsum += feature3[1] * -.927008;
netsum += feature3[2] * 4.121078;
netsum += feature3[3] * -1.074661;
netsum += feature3[4] * .2331914;
```

```
netsum += feature3[5] * -1.074572;
netsum += feature3[6] *  1.821139;
netsum += feature3[7] *  2.462906;
netsum += feature3[8] *  .6460868;
netsum += feature3[9] * -.3657965;
netsum += feature3[10] * -.1615;
netsum += feature3[11] *  .2754445;
netsum += feature3[12] *  1.444201;
netsum += feature3[13] *  5.678596E-02;
netsum += feature3[14] * -1.543293;
netsum += feature3[15] * -2.532481E-02;
netsum += feature3[16] *  2.518335;
netsum += feature3[17] * -.9775917;
netsum += feature3[18] * -1.828882;
netsum += -9.571709E-02;
netsum += feature4[0] *  .1281435;
netsum += feature4[1] * -.6704976;
netsum += feature4[2] *  1.572031E-02;
netsum += feature4[3] *  1.640785;
netsum += feature4[4] * -4.086291;
netsum += feature4[5] *  .952208;
netsum += feature4[6] * -.2722533;
netsum += feature4[7] * -1.029774;
netsum += feature4[8] * -.1491238;
netsum += feature4[9] *  7.342882E-02;
netsum += feature4[10] * -.3842966;
netsum += feature4[11] *  1.016283;
netsum += feature4[12] * -.3097355;
netsum += feature4[13] * -.239762;
netsum += feature4[14] * -.5526069;
netsum += feature4[15] * -5.071243;
netsum += feature4[16] * -.8528777;
netsum += feature4[17] * -2.003938;
netsum += feature4[18] *  .946243;
outarray[36] = 1 / (1 + exp(-netsum));

netsum = -.1305978;
netsum += feature2[0] * -2.029733;
netsum += feature2[1] *  .9983642;
netsum += feature2[2] * -.6410506;
netsum += feature2[3] *  1.347134;
netsum += feature2[4] *  .876291;
netsum += feature2[5] * -.2141008;
netsum += feature2[6] * -.2639521;
netsum += feature2[7] *  .9694061;
netsum += feature2[8] * -.2879335;
netsum += feature2[9] * -.3518944;
netsum += feature2[10] * -.464126;
netsum += feature2[11] * -.5861252;
netsum += feature2[12] *  .7704247;
netsum += feature2[13] * -.2354016;
netsum += feature2[14] *  .7305539;
netsum += feature2[15] * -.491787;
netsum += feature2[16] * -.4780943;
netsum += feature2[17] * -.2199717;
netsum += feature2[18] * -.1487872;
netsum += -.4094036;

netsum += feature3[0] *  .4902677;
netsum += feature3[1] *  .14398;
netsum += feature3[2] * -1.278632;
netsum += feature3[3] *  6.488828E-02;
netsum += feature3[4] *  .6383151;
netsum += feature3[5] *  9.700023E-02;
netsum += feature3[6] *  .1701944;
netsum += feature3[7] *  .811559;
netsum += feature3[8] *  .1061409;
netsum += feature3[9] * -1.758041;
netsum += feature3[10] * -1.840511;
netsum += feature3[11] *  .1435418;
netsum += feature3[12] * -1.376621;
netsum += feature3[13] * -.2335458;
netsum += feature3[14] * -1.817738;
netsum += feature3[15] *  .5045825;
netsum += feature3[16] *  .666295;
netsum += feature3[17] *  2.915928;
netsum += feature3[18] * -.3117751;
netsum += -.2076916;
netsum += feature4[0] * -.1796421;
netsum += feature4[1] * -1.458866;
netsum += feature4[2] *  6.573024E-02;
netsum += feature4[3] *  .5156263;
netsum += feature4[4] * -.1874862;
netsum += feature4[5] * -1.011646;
netsum += feature4[6] *  .2404576;
netsum += feature4[7] *  .8803746;
netsum += feature4[8] *  1.553041;
netsum += feature4[9] * -4.316342E-02;
netsum += feature4[10] *  .575678;
netsum += feature4[11] * -.2056797;
netsum += feature4[12] * -1.499956;
netsum += feature4[13] *  1.454034;
netsum += feature4[14] *  1.263607;
netsum += feature4[15] * -1.222317;
netsum += feature4[16] *  .691096;
netsum += feature4[17] * -.1269283;
netsum += feature4[18] *  .3964429;
outarray[37] = 1 / (1 + exp(-netsum));

netsum = -.8043524;
netsum += feature2[0] *  .1710643;
netsum += feature2[1] *  .2195557;
netsum += feature2[2] *  2.699094;
netsum += feature2[3] * -.3123901;
netsum += feature2[4] * -.8684938;
netsum += feature2[5] *  .3283499;
netsum += feature2[6] * -.5661883;
netsum += feature2[7] *  .587549;
netsum += feature2[8] * -.8452105;
netsum += feature2[9] * -.2637961;
netsum += feature2[10] *  .53075;
netsum += feature2[11] *  3.355742;
netsum += feature2[12] * -.8761499;
netsum += feature2[13] * -1.021009;
netsum += feature2[14] * -1.062612;
```

```
netsum += feature2[15] * -.1209282;
netsum += feature2[16] *  2.079563;
netsum += feature2[17] *  1.617552;
netsum += feature2[18] *  1.939732;
netsum += -1.244801;
netsum += feature3[0] *  .9992966;
netsum += feature3[1] *  .4089036;
netsum += feature3[2] *  .3039086;
netsum += feature3[3] *  4.478193;
netsum += feature3[4] * -3.835225;
netsum += feature3[5] *  .5116088;
netsum += feature3[6] *  .5070441;
netsum += feature3[7] * -1.905791;
netsum += feature3[8] * -1.794084;
netsum += feature3[9] * -1.348748;
netsum += feature3[10] *  .4001587;
netsum += feature3[11] * -2.19778;
netsum += feature3[12] * -.3959824;
netsum += feature3[13] * -1.011519;
netsum += feature3[14] * -.7277553;
netsum += feature3[15] *  .9399757;
netsum += feature3[16] * -5.630291E-02;
netsum += feature3[17] * -.541714;
netsum += feature3[18] * -.4479994;
netsum += -1.216212;
netsum += feature4[0] *  1.432951;
netsum += feature4[1] *  .3370137;
netsum += feature4[2] *  1.370845;
netsum += feature4[3] * -.4847945;
netsum += feature4[4] * -.4614465;
netsum += feature4[5] *  1.854205E-02;
netsum += feature4[6] * -.4978043;
netsum += feature4[7] *  .5611956;
netsum += feature4[8] *  .1441261;
netsum += feature4[9] * -1.073189;
netsum += feature4[10] *  .5821497;
netsum += feature4[11] * -1.261721;
netsum += feature4[12] *  .3117783;
netsum += feature4[13] *  .5128441;
netsum += feature4[14] *  .6889659;
netsum += feature4[15] *  1.869442;
netsum += feature4[16] *  .9111376;
netsum += feature4[17] *  .40985;
netsum += feature4[18] * -.5085996;
outarray[38] = 1 / (1 + exp(-netsum));

netsum = -.4042704;
netsum += feature2[0] * -.2065078;
netsum += feature2[1] * -.4749881;
netsum += feature2[2] *  4.894628E-02;
netsum += feature2[3] * -2.235207E-02;
netsum += feature2[4] *  .1834634;
netsum += feature2[5] *  2.10359;
netsum += feature2[6] *  .1384059;
netsum += feature2[7] *  .2524251;
netsum += feature2[8] * -.275492;
netsum += feature2[9] * -.273244;
netsum += feature2[10] * -7.854713E-02;
netsum += feature2[11] * -1.69897;
netsum += feature2[12] * -.2012461;
netsum += feature2[13] * -.241412;
netsum += feature2[14] *  .5048268;
netsum += feature2[15] * -.4611538;
netsum += feature2[16] * -.558277;
netsum += feature2[17] * -.7051769;
netsum += feature2[18] * -.1019972;
netsum += -.2500765;
netsum += feature3[0] * -4.298972E-02;
netsum += feature3[1] *  .5108634;
netsum += feature3[2] * -.5888803;
netsum += feature3[3] * -.2995635;
netsum += feature3[4] * -.128062;
netsum += feature3[5] *  .206438;
netsum += feature3[6] * -.7055873;
netsum += feature3[7] * -.9682887;
netsum += feature3[8] *  .7191542;
netsum += feature3[9] * -.8501001;
netsum += feature3[10] * -.1277455;
netsum += feature3[11] *  1.120425;
netsum += feature3[12] *  .0516791;
netsum += feature3[13] * -.1189754;
netsum += feature3[14] *  1.241842;
netsum += feature3[15] *  .3554744;
netsum += feature3[16] * -.0283204;
netsum += feature3[17] *  .3215312;
netsum += feature3[18] *  .5853519;
netsum += -9.201432E-04;
netsum += feature4[0] * -1.598734;
netsum += feature4[1] *  .1837678;
netsum += feature4[2] * -.8303027;
netsum += feature4[3] * -.2188892;
netsum += feature4[4] *  1.291907;
netsum += feature4[5] *  .4072145;
netsum += feature4[6] * -5.380255E-03;
netsum += feature4[7] * -.108004;
netsum += feature4[8] *  .1172905;
netsum += feature4[9] *  .5242521;
netsum += feature4[10] *  .067476;
netsum += feature4[11] *  .1654843;
netsum += feature4[12] * -.222025;
netsum += feature4[13] * -1.343958;
netsum += feature4[14] * -.6369489;
netsum += feature4[15] *  .5855124;
netsum += feature4[16] *  .9344859;
netsum += feature4[17] *  .2788305;
netsum += feature4[18] * -.2015109;
outarray[39] = 1 / (1 + exp(-netsum));


outarray[0] =  5215.79 *  (outarray[0] - .1) / .8
+  47.36842;
if (outarray[0]< 47.36842) outarray[0] =
47.36842;
```

```
if (outarray[0]> 5263.158) outarray[0] =
5263.158;

outarray[1] = 3890.979 * (outarray[1] - .1) /
.8 + 35.33684;
if (outarray[1]< 35.33684) outarray[1] =
35.33684;
if (outarray[1]> 3926.316) outarray[1] =
3926.316;

outarray[2] = 2302.734 * (outarray[2] - .1) /
.8 + 10.71738;
if (outarray[2]< 10.71738) outarray[2] =
10.71738;
if (outarray[2]> 2313.451) outarray[2] =
2313.451;

outarray[3] = 29.12 * (outarray[3] - .1) / .8 +
2.88;
if (outarray[3]< 2.88) outarray[3] = 2.88;
if (outarray[3]> 32) outarray[3] = 32;

outarray[4] = 13.80261 * (outarray[4] - .1) /
.8 + 1.533623;
if (outarray[4]< 1.533623) outarray[4] =
1.533623;
if (outarray[4]> 15.33623) outarray[4] =
15.33623;

outarray[5] = 16.90467 * (outarray[5] - .1) /
.8 + 1.878297;
if (outarray[5]< 1.878297) outarray[5] =
1.878297;
if (outarray[5]> 18.78297) outarray[5] =
18.78297;

outarray[6] = 23.90682 * (outarray[6] - .1) /
.8 + 2.656313;
if (outarray[6]< 2.656313) outarray[6] =
2.656313;
if (outarray[6]> 26.56313) outarray[6] =
26.56313;

outarray[7] = 34.50652 * (outarray[7] - .1) /
.8 + 3.834058;
if (outarray[7]< 3.834058) outarray[7] =
3.834058;
if (outarray[7]> 38.34058) outarray[7] =
38.34058;

outarray[8] = 2994.421 * (outarray[8] - .1) /
.8 + 52.23381;
if (outarray[8]< 52.23381) outarray[8] =
52.23381;
if (outarray[8]> 3046.655) outarray[8] =
3046.655;

outarray[9] = 2705.764 * (outarray[9] - .1) /
.8 + 47.19857;
if (outarray[9]< 47.19857) outarray[9] =
47.19857;
if (outarray[9]> 2752.963) outarray[9] =
2752.963;

outarray[10] = 2275.268 * (outarray[10] - .1)
/ .8 + 39.68911;
if (outarray[10]< 39.68911) outarray[10] =
39.68911;
if (outarray[10]> 2314.957) outarray[10] =
2314.957;

outarray[11] = 1893.839 * (outarray[11] - .1)
/ .8 + 33.03556;
if (outarray[11]< 33.03556) outarray[11] =
33.03556;
if (outarray[11]> 1926.874) outarray[11] =
1926.874;

outarray[12] = 32.73576 * (outarray[12] - .1)
/ .8 + 3.637307;
if (outarray[12]< 3.637307) outarray[12] =
3.637307;
if (outarray[12]> 36.37307) outarray[12] =
36.37307;

outarray[13] = 45.83006 * (outarray[13] - .1)
/ .8 + 5.092229;
if (outarray[13]< 5.092229) outarray[13] =
5.092229;
if (outarray[13]> 50.92229) outarray[13] =
50.92229;

outarray[14] = 49.88306 * (outarray[14] - .1)
/ .8 + 5.542562;
if (outarray[14]< 5.542562) outarray[14] =
5.542562;
if (outarray[14]> 55.42562) outarray[14] =
55.42562;

outarray[15] = 54.5596 * (outarray[15] - .1) /
.8 + 6.062178;
if (outarray[15]< 6.062178) outarray[15] =
6.062178;
if (outarray[15]> 60.62178) outarray[15] =
60.62178;

outarray[16] = 1588.626 * (outarray[16] - .1)
/ .8 + 193.6794;
if (outarray[16]< 193.6794) outarray[16] =
193.6794;
if (outarray[16]> 1782.305) outarray[16] =
1782.305;
```

```
outarray[17] = 1435.485 * (outarray[17] - .1)
/ .8 + 175.0091;
if (outarray[17]< 175.0091) outarray[17] =
175.0091;
if (outarray[17]> 1610.494) outarray[17] =
1610.494;

outarray[18] = 1207.094 * (outarray[18] - .1)
/ .8 + 147.1645;
if (outarray[18]< 147.1645) outarray[18] =
147.1645;
if (outarray[18]> 1354.259) outarray[18] =
1354.259;

outarray[19] = 2686.88 * (outarray[19] - .1) /
.8 + 207.5974;
if (outarray[19]< 207.5974) outarray[19] =
207.5974;
if (outarray[19]> 2894.477) outarray[19] =
2894.477;

outarray[20] = 1884.143 * (outarray[20] - .1)
/ .8 + 50.99396;
if (outarray[20]< 50.99396) outarray[20] =
50.99396;
if (outarray[20]> 1935.137) outarray[20] =
1935.137;

outarray[21] = 2919.206 * (outarray[21] - .1)
/ .8 + 79.00773;
if (outarray[21]< 79.00773) outarray[21] =
79.00773;
if (outarray[21]> 2998.214) outarray[21] =
2998.214;

outarray[22] = 3778.547 * (outarray[22] - .1)
/ .8 + 102.2656;
if (outarray[22]< 102.2656) outarray[22] =
102.2656;
if (outarray[22]> 3880.813) outarray[22] =
3880.813;

outarray[23] = 2275 * (outarray[23] - .1) / .8
+ 225;
if (outarray[23]< 225) outarray[23] = 225;
if (outarray[23]> 2500) outarray[23] = 2500;

outarray[24] = 5248.764 * (outarray[24] - .1)
/ .8 + 50.35735;
if (outarray[24]< 50.35735) outarray[24] =
50.35735;
if (outarray[24]> 5299.121) outarray[24] =
5299.121;

outarray[25] = 4742.793 * (outarray[25] - .1)
/ .8 + 45.503;
if (outarray[25]< 45.503) outarray[25] =
45.503;
if (outarray[25]> 4788.296) outarray[25] =
4788.296;

outarray[26] = 3988.198 * (outarray[26] - .1)
/ .8 + 38.26331;
if (outarray[26]< 38.26331) outarray[26] =
38.26331;
if (outarray[26]> 4026.461) outarray[26] =
4026.461;

outarray[27] = 3319.609 * (outarray[27] - .1)
/ .8 + 31.84878;
if (outarray[27]< 31.84878) outarray[27] =
31.84878;
if (outarray[27]> 3351.458) outarray[27] =
3351.458;

outarray[28] = 35.42667 * (outarray[28] - .1)
/ .8 + -16.66848;
if (outarray[28]<-16.66848) outarray[28] = -
16.66848;
if (outarray[28]> 18.75819) outarray[28] =
18.75819;

outarray[29] = 46.05468 * (outarray[29] - .1)
/ .8 + -26.76903;
if (outarray[29]<-26.76903) outarray[29] = -
26.76903;
if (outarray[29]> 19.28565) outarray[29] =
19.28565;

outarray[30] = 36.84374 * (outarray[30] - .1)
/ .8 + -18.41522;
if (outarray[30]<-18.41522) outarray[30] = -
18.41522;
if (outarray[30]> 18.42852) outarray[30] =
18.42852;

outarray[31] = 23.02734 * (outarray[31] - .1)
/ .8 + -6.634514;
if (outarray[31]<-6.634514) outarray[31] = -
6.634514;
if (outarray[31]> 16.39283) outarray[31] =
16.39283;

outarray[32] = 3173.586 * (outarray[32] - .1)
/ .8 + 10.63496;
if (outarray[32]< 10.63496) outarray[32] =
10.63496;
if (outarray[32]> 3184.221) outarray[32] =
3184.221;

outarray[33] = 3969.239 * (outarray[33] - .1)
/ .8 + -29.25188;
```

```
if (outarray[33]<-29.25188) outarray[33] = -29.25188;
if (outarray[33]> 3939.987) outarray[33] =  3939.987;

outarray[34] =  3928.705 *  (outarray[34] - .1) / .8  + -91.49915;
if (outarray[34]<-91.49915) outarray[34] = -91.49915;
if (outarray[34]> 3837.206) outarray[34] =  3837.206;

outarray[35] =  1081.068 *  (outarray[35] - .1) / .8  +  65.61992;
if (outarray[35]< 65.61992) outarray[35] =  65.61992;
if (outarray[35]> 1146.688) outarray[35] =  1146.688;

outarray[36] =  10 *  (outarray[36] - .1) / .8 ;
if (outarray[36]< 0) outarray[36] =  0;
if (outarray[36]> 10) outarray[36] =  10;

outarray[37] =  10 *  (outarray[37] - .1) / .8 ;
if (outarray[37]< 0) outarray[37] =  0;
if (outarray[37]> 10) outarray[37] =  10;

outarray[38] =  10 *  (outarray[38] - .1) / .8 ;
if (outarray[38]< 0) outarray[38] =  0;
if (outarray[38]> 10) outarray[38] =  10;

outarray[39] =  10 *  (outarray[39] - .1) / .8 ;
if (outarray[39]< 0) outarray[39] =  0;
if (outarray[39]> 10) outarray[39] =  10;

}
```