UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

"DESARROLLO DE SOFTWARE DE CÓDIGO ABIERTO DE INGENIERÍA PETROLERA"

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO PETROLERO

PRESENTAN:

FERNANDO ATLAHUA LÓPEZ CASTRO

MARCO ANTONIO REYES ATENCIA

ASESOR: ING. ISRAEL CASTRO HERRERA

CIUDAD UNIVERSITARIA 2010

Fernando Atlahua López Castro

A mis padres Consuelo y Fernando. A mi hermana Frida.

Gracias por quererme tanto, por su incondicional apoyo, consejo, respeto, esfuerzo y determinación para formarme un carácter desobediente y autónomo.

A Mamá LLita, Luisa y Josefina por su invaluable ayuda precisa y desinteresada, sin la cual no me hubiera sido posible alcanzar esta meta.

A Pamela, por tu lealtad, tu paciencia, tu tiempo, tu consejo, tu ayuda y tu cariño.

A Camilita, por recibirme siempre con un coleteo hospitalario y noble.

Gracias a todos por darme la certeza de que siempre podemos contar con nuestro mutuo apoyo.

A toda mi familia (López y Castro) y a mis amigos. Que me han honrado con su compañía, sus reconocimientos, sus cuestionamientos y su humilde perdón por mis errores. No quiero cometer la torpeza de olvidar a alguien por lo que prefiero agradecerles personalmente conforme nos encontremos de nuevo.

A la tercera ley de Newton que me ha puesto a prueba una y otra vez. Que me ha permitido experimentar y aprender de las cosas buenas y malas.

A todos los mexicanos y mexicanas que con su trabajo han invertido en mi educación a través de la Universidad Nacional Autónoma de México, y con quienes estaré permanentemente en deuda.

A mi país y a su cultura que me enorgullece sobre cualquier otra.

A los hombres y mujeres libres que han participado en el avance de la humanidad.

Marco Antonio Reyes Atencia

A **Dios**, por haberme colocado con los seres indicados, en el momento indicado y en el lugar indicado para desarrollarme como la persona que soy.

A la **Universidad Nacional Autónoma de México** y a la **Facultad de Ingeniería** por formarme en sus aulas, con sus profesores, para darme las armas necesarias para poder desarrollarme como un profesional, humano y mexicano comprometido con mi país.

A mi **madre** que me dio la vida. Por estar en todos los momentos de mi vida, en los buenos y malos, felices y amargos. Por ser mi apoyo, inspiración y aliento para poder tener los logros que he tenido. Mamá, te amo con todo mi ser. Eres todo para mi.

A mis hermanos **Alexis** y **Carlos** por estar siempre apoyándome, por darme tantas alegrías con sus acciones. Son los mejores que me pudieron haber tocado. Agradezco a mis padres por darme a mis hermanos que amo tanto. ¡Éxito!

A mi **padre** por darme la vida. Por ser mi apoyo a pesar de las distancias y dificultades, siempre has sido importante en mi vida. Te amo papá.

Agradecimientos

A mi socio, **Fernando Atlahua López** por ser mi amigo durante tanto tiempo, empujarme a desarrollar capacidades que no pensé que tuviera, por haber realizado este trabajo conmigo y por creer en mi en todo momento. Gracias socio.

A mi amigo, el **Ingeniero Juan Ocariz**, por apoyarme en uno de los momentos más difíciles en mi vida y por recordarme cual es el cometido de la ingeniería.

A mi **abuela Graciela** y demás familiares en Colombia que siempre ha estado al pendiente de mi crecimiento.

A mis amigos de la Facultad, **Memeto, Alexi, Smith, Yaniak, Aarón, Oscar, Chiapas, Memibi, Talib, Pablo, Vero, Adán, Juliana, Mary, Tiki, Sergio, Pau, Eder, Choco, Fredy, Chavo, Isi, Dano, Eliana, Aliskair, Carlos, Azu, Jet** y, en fin, a todos los que durante mi estancia en esta gloriosa facultad me dieron tantos momentos de alegría.

Al crew **L-5 Ingeniero José Luis, Diego, Uziel, Inge Cayo y Peter.** Estos últimos meses hemos hecho una amistad muy fuerte que espero perdure por siempre.

A mis brps **Luis, Alheli, Eugenio, David, Will, Alán, Andrés y América.** Saben lo mucho que los quiero y se que siempre contaré con ustedes.

A **Berenise, Karla, Ori, Gabriela y Atenea**. Todo el tiempo han creído en mi. Las admiro y las quiero mucho.

A los **Escorpiones Rojos** y a los **Pumas** por haberme formado un carácter, por darme alegrías y ayudarme a superar dificultades a través de la práctica del futbol americano, deporte que tanto amo.

A **Rodrigo Antonio Teja.** Que Dios te tenga en su gloria.

Ser cultos para ser libres.

-José Martí

ÍNDICE

CONTENIDO

FIGURAS

TABLAS

ECUACIONES

INTRODUCCIÓN

OBJETIVO

Sentar las bases y la primera versión de una colección de software de Ingeniería Petrolera de código abierto (Apantli), que ofrezca una alternativa para estimular el interés y las habilidades de los alumnos de la Carrera de Ingeniería Petrolera, por aplicar sus conocimientos de Ingeniería y programación, a la solución de problemas de la industria, mediante la utilización y desarrollo de nuevos módulos adaptados a la plataforma.

Se entiende como plataforma, a esta colección de software que podrá ser consultada y mejorada bajo los principios del software libre [5], por cualquier persona interesada.

RESUMEN

A pesar del gran avance que el Software de Ingeniería Petrolera ha tenido en las últimas décadas, acompañado por el avance tecnológico en herramientas y procedimientos, planteamos algunos aspectos que podrían mejorar la eficiencia y el beneficio de dicho software. Principalmente sobre la capacidad de adaptar el software a las necesidades del usuario y no al revés como en la mayoría de los casos sucede. Así mismo, la tendencia global de poner el software a disposición de los usuarios a través de internet, por todos los beneficios que eso representa.

Posteriormente, comentamos sobre el desarrollo de software en la Carrera de Ingeniería Petrolera, desde nuestro punto de vista como alumnos y con fundamento en “Un nuevo modelo educativo basado en el aprendizaje” (Libro escrito por nuestro profesor el Dr. Rafael Rodríguez Nieto y M.A. Mabel Rodríguez de la Torre, publicado en 2008 por la Facultad de Ingeniería, a través del proyecto PAPIME PE101707).

El Capítulo 2 está dedicado al aspecto técnico del desarrollo de la plataforma. Desde la instalación, configuración y puesta en marcha del ambiente de desarrollo, pasando por algunos fundamentos de ingeniería de software hasta llegar a las notas sobre los éxitos, dificultades y sus respectivas soluciones que enfrentamos durante el desarrollo de la plataforma.

El Capítulo 3 comienza con la parte teórica de las pruebas de presión, haciendo especial énfasis en las pruebas de interferencia y su interpretación por el método del Match Point. Lo que da pie a la explicación del algoritmo numérico que de manera alternativa, ajusta pares de puntos de una prueba real a una Curva Tipo.

En la primera versión de este módulo, el ajuste está enfocado a yacimientos homogéneos, infinitos, con flujo radial e isótropo. El ajuste está llevado a través del método del Match Point, por medio de la curva tipo desarrollada por Theis en 1935, dónde se utilizan variables adimensionales para la variación en la presión, el incremento en el tiempo y la distancia entre pozos.

Inmediatamente después se aborda la tarea de ampliar las capacidades de la plataforma, integrando nuevos módulos. Para esta primera versión, se desarrolló un módulo de suavizado de datos, que se utilizará previo al método de ajuste.

El código de toda la plataforma está incluido y comentado, para facilitar la comprensión y mantenimiento del mismo.

En el Capítulo 4 se describen y comparan los resultados de diferentes pruebas realizadas por varios métodos como el gráfico con papel, el gráfico con software comercial y el numérico que se presenta en este trabajo.

Cabe mencionar que los autores hemos colocado material adicional como instructivos, videos y un foro de discusión dedicado a este trabajo, que estará permanentemente disponible en el sitio http://www.atlahua.com/tesis.

Capítulo 1 PANORAMA ACTUAL DEL SOFTWARE DE INGENIERÍA PETROLERA

1.1. SOFTWARE ACTUAL DE INGENIERÍA PETROLERA Y ALGUNAS DE SUS CARACTERÍSTICAS

Actualmente gran parte del software de Ingeniería Petrolera es desarrollado y proporcionado por compañías operadoras o de servicios. Algunos ejemplos software para el área de simulación numérica de yacimientos son:

- *ECLIPSE©, Frontsim™ (Schlumberger)*
- *IMEX™, STARS™, GEM™ (CMG)*
- *VIP™, Nexus© (Landmark)*
- *MoReS© (Shell)*
- *CHEARS© (Chevron)*
- *PSIM© (ConocoPhilips)*
- *3DSL® (StreamSim Technologies)*
- *IPOS©, MEPO© (SPT Group)*

Específicamente para la caracterización dinámica de yacimientos figuran:

- Weatherofrd (Pan System) [1]
- Fekete (F.A.S.T. WellTest) [2]
- Fekete (F.A.S.T. RTA) [2]
- HydroSOLVE, Inc (AQTESOLV v4.5) [3]
- KAPPA ENG (TOPAZE)
- KAPPA ENG (SAPHIR)

La mayoría del software de estas empresas tienen algunas características en común, tales como utilizar consideraciones en las ecuaciones originales para posibilitar y facilitar la solución de un problema, pero lamentablemente el usuario no siempre las conoce y peor aún: no está facultado para modificar dichos procedimientos.

Algunas de estas consideraciones pueden ser: suponer procesos isotérmicos, desestimar la presencia o cambio de fase de algún fluido, forzar el ajuste de una correlación que no corresponde al fenómeno que intenta representar, requerir ciertos datos difíciles de obtener o aquellos que son medidos con mucha incertidumbre.

Si fuera necesaria una adecuación o extensión del funcionamiento del programa, los usuarios tendrían que comunicarlo al fabricante y esperar una respuesta favorable del fabricante para realizar la modificación, esperar que la modificación funcione y por supuesto, pagar por dicha modificación.

1.2. DISPONIBILIDAD Y COMPATIBILIDAD DEL SOFTWARE DE INGENIERÍA PETROLERA

Por otra parte existe el problema de la compatibilidad con cada nueva versión de los sistemas operativos, por lo que es difícil que globalmente se maneje la misma versión (o la más reciente) de un software.

Al ver las especificaciones técnicas de los productos, se puede observar que algunos de estos sistemas están diseñados para funcionar en plataformas *UNIX*, *LINUX* o *Windows*.

¿Cómo debería el cliente decidirse al respecto del sistema operativo que utilizará para usar el software?

En base a la experiencia de utilizar *Linux*, *Mac Os X* y *Windows* simultáneamente para tareas cotidianas y escolares, es común que para hacer una misma tarea exista un programa favorito para una plataforma que no esté disponible para todas las demás. Un ejemplo claro es "*Microsoft Office*" (MSO). MSO está disponible para *Windows* y *Mac*, pero no para *Linux*. Para *Linux* existe "*OpenOffice.org*" (OO.org), que en contraste, también está disponible para *Windows* y *Mac* [4]. Pero hay un pequeño detalle que ocasiona un dilema sobre qué sistema operativo usar y con cuál suite de *Office*.

Trabajar en *Linux* es muy cómodo. La probabilidad de que el equipo se congele (o que aparezca una pantalla de error) es mínima. Así mismo, la probabilidad de que el equipo se infecte con un virus es aún menor y la seguridad contra el *hacking* de información está muy por encima que la de *Windows*. Pero existe un problema y es que a pesar de que OO.org puede abrir un documento de MSO, hasta la versión 3 de OO.org no existe soporte para el editor de ecuaciones de MSO.

El dilema entonces es: ¿Por el hecho de que no puedo ver ecuaciones de MSO en OO.org, tendremos que renunciar a la estabilidad de *Linux* para ir a *Windows* y utilizar MSO?.

El objetivo de citar el ejemplo anterior es porque existe una analogía en lo que al software de Ingeniería Petrolera se refiere. Si el mejor software para "Análisis de pruebas de presión" sólo funcionara en *Linux*, el cliente tendría dos alternativas:

1. Capacitar o contratar a alguien que tenga conocimientos del software y del sistema operativo.
2. Renunciar a utilizar el mejor software del mercado y conformarse a utilizar alguno compatible con *Windows* y ahorrar en la capacitación o contratación de personal.

Pero ¿Sería posible que el mejor software para "Análisis de pruebas de presión" fuera compatible con todos los sistemas operativos? Quizá sí, pero no hay que olvidar que una empresa existe para hacer dinero y al igual que pasa con las diferentes versiones de MSO, se vende por separado la licencia que es para *Mac* y la que es para *Windows*.

¿Sería costeable para el cliente adquirir dos licencias diferentes? sin olvidar que una versión pueda ser compatible con *Windows XP*, otra para *Windows Vista*, otra para *Windows 7* etc. Entonces ¿Será posible que el mejor software para "análisis de pruebas de presión" sea compatible con todos los sistemas operativos y con sus respectivas versiones sin tener que pagar por licencias diferentes?

La respuesta es sí y a lo largo de este trabajo se explica cómo.

Una evolución muy importante del software en la Industria Petrolera fue la capacidad de monitoreo y operación de equipos a control remoto (con o sin

cable) en tiempo real. Por ejemplo, una compañía es capaz de tener a especialistas en un centro de control supervisando simultáneamente 2 o más operaciones en tiempo real, ya sea manipulando los equipos o teniendo comunicación con el personal que se encuentra físicamente en la locación.

En lo que respecta a algunas herramientas de medición, éstas almacenan información en una memoria interna, para que una vez recuperada la herramienta, los datos sean descargados a otro dispositivo donde se pudieran analizar. Pero la tendencia ahora está enfocada a que la medición y la transmisión de datos sea simultánea con el objetivo de tener la información de manera más oportuna y previniendo pérdidas de información si ocurre un imprevisto y la herramienta no puede ser recuperada.

Cualquier esfuerzo encaminado a reducir los costos asociados a la deficiente recuperación de datos debe considerarse en el desarrollo de las nuevas tecnologías.

1.3. SOFTWARE DE INGENIERÍA PETROLERA Y SU EVOLUCIÓN

Hablando de desarrollo tecnológico, podemos identificar claramente 3 áreas con grandes necesidades:

1. En herramientas físicas (Adquisición de datos).
2. En herramientas virtuales (Manipulación de datos).
3. En procedimientos (Aplicación de los datos).

Estas áreas no pueden avanzar unilateralmente, dependen y se benefician unas de otras como se ilustra en el siguiente ejemplo:

1. Adquisición de datos:

Imaginemos que inicialmente, utilizamos una herramienta para medir pulsos de presión con poca resolución y poca sensibilidad, que además tiene un sistema de almacenamiento limitado. Esta herramienta entregará una lectura cada hora y tiene capacidad para almacenar 72 lecturas.

2. Manipulación de datos:

Al tener menos de 72 datos que manipular, es posible que no se requiera de un software avanzado y que los datos sean alimentados manualmente por el operador de software.

3. Aplicación de los datos:

Al tener una herramienta de baja resolución y baja sensibilidad, quizá los datos sirvan más para hacer una descripción cualitativa que una descripción cuantitativa confiable.

Proceso de innovación:

A raíz de esta situación, surge la necesidad de mejorar la herramienta, de modo que ahora tiene una resolución 100 veces mayor, es capaz de tomar una lectura por segundo y almacenar cien mil datos. Esto obligará a que el software sea más potente al requerir más memoria y velocidad para procesar los datos, será necesario igualmente ingresar los datos de manera automática para finalmente obtener un diagnóstico más acertado de los parámetros que se estudien, lo cual podría derivar en mejoras para el modelo matemático que se utiliza y la inclusión de otros parámetros que una nueva herramienta pueda medir.

Es evidente que las herramientas actuales mejoran constantemente en muchos aspectos, pero consideramos que las otras dos áreas, están rezagadas y no aprovechan todo el potencial de los datos recopilados.

Un claro ejemplo de esta situación está en la simulación numérica de yacimientos donde el escalamiento de datos geológicos tiene como objetivo reducir el número de datos en un modelo geológico para acelerar su procesamiento, conservando una descripción lo más fiel posible del modelo original.

El escalamiento se realiza debido a que unos de los objetivos de la simulación numérica de yacimientos es tomar decisiones oportunas, por lo que no se debe permitir que los simuladores requieran más potencia y más tiempo de procesamiento del que actualmente disponen.

¿Cómo podríamos reducir el nivel de escalamiento de un modelo geológico, para aprovechar todos los datos que las herramientas registran?

De inmediato se tienen dos propuestas:

1. Utilizando equipos cada vez más sofisticados, obviando los costos asociados.

2. Mejorando la eficiencia del simulador de modo que para obtener los mismos resultados, realice menos procesos con menos recursos en menos tiempo. Eso permitiría ampliar el rango de datos utilizados y eventualmente la calidad de la interpretación.

Se considera que algunos métodos para desarrollar y utilizar software de ingeniería petrolera no han avanzando a la par de las herramientas y de la necesidad de tener la información al instante y siempre disponible. Por otra parte, mucho del manejo y procesamiento de información en la industria petrolera, se hace a través de hojas de cálculo con algunas serias desventajas asociadas:

1. Se requiere de un editor de hojas de cálculo, que a su vez requiere un sistema operativo el cual también tiene requerimientos de hardware. Es decir: *Excel 2007* + *Windows* (*Xp, Vista, 7*) = 1 gb de *ram*, 1 *Ghz* de procesador como mínimo.

2. Se requiere de un nivel de conocimiento o familiaridad por encima de los usuarios comunes, para manipular y en algunos casos siquiera para consultar la información contenida en la hoja de cálculo (como el uso de *Macros*).

3. Pese a que las computadoras estén en red, si el archivo que el "Usuario A" requiere sólo está en la máquina del "Usuario B", podría el "Usuario A" quedar inhabilitado para usar el archivo si el equipo del "Usuario B" no está disponible o si el archivo ha sufrido algún daño o modificación inconveniente e irreparable.

La confidencialidad de la información manejada en archivos de hojas de cálculo o similares, está comprometida debido a que no hay restricciones acerca de quién tiene la información, quién la transfiere, quién la modifica etcétera.

Ésta deficiencia en el manejo de la información, podría inclusive ocasionar la pérdida total de algunos archivos que no estén a salvo de ser borrados o modificados.

¿Cómo se podría entonces tener la información disponible, controlada y a salvo?

Una manera de hacerlo es utilizar bases de datos, tal y como operan en las instituciones bancarias.

1.4. SOLUCIÓN PROPUESTA

Hasta ahora hemos desarrollado este capítulo planteando problemas y propuestas de solución de manera aislada. La propuesta de solución global que se desarrolla en este trabajo consiste en que los datos obtenidos por una sonda de presión (para pruebas de presión) o por otros medios, sean almacenados en una base de datos que pueda ser consultada por el nuevo software.

Ante la creciente demanda y consumo de *internet*, cada vez son más los servicios que se ofrecen a través de éste debido a una simple razón: Los procesos que tienen ciertos requerimientos de *hardware*, se realizan en un servidor (remoto y potente), mientras que el usuario sólo ingresa datos a través de un navegador de *internet* (*Mozilla FireFox, Internet Explorer, Opera, Safari, Google Chrome*, etc.), ya sea con una PC, una *netbook*, un teléfono celular, una agenda electrónica etc.

El principio de operación de los programas ejecutados del lado del servidor (como el que se presenta en esta Tesis), sigue esta lógica de la siguiente manera:

1. El usuario envía al servidor un archivo o variables como datos de entrada.
2. El servidor procesa la información y genera un archivo con resultados.
3. El usuario descarga el archivo de resultados que podrá utilizar para otros propósitos.

Cabe mencionar que para este trabajo, los tres pasos se realizan con una interfaz intuitiva y amigable para el usuario que no obstante, siempre estaría sujeta a mejorías.

APANTLI

"Apantli" es el nombre de la plataforma de software de código abierto de Ingeniería Petrolera la cuál surge como una solución para los problemas planteados en esta Tesis. ***Apantli*** (del Náhuatl, *por donde fluye el agua, tubería*).

Apantli opera bajo los principios descritos anteriormente, así por ejemplo, si un ingeniero que trabaja en alguna oficina del Distrito Federal necesitara los datos de una prueba de interferencia realizada a un pozo de la Región Marina Suroeste, sólo tendría que ingresar a través de un navegador, con un nombre de usuario y contraseña. Utilizaría la interfaz para cargar datos desde una hoja de cálculo o

bien seleccionarlos desde una base de datos (donde dicha información no tiene que ser forzosamente descargada al equipo del usuario y además existiría una constancia de quién, cuándo y desde dónde se hizo la petición de la consulta).

Si se emplea el acceso a la información por medio de la base de datos, el usuario puede prescindir de una hoja de cálculo, ya que podría cargar los datos directamente del sistema de información. El usuario tampoco estaría sujeto a un sistema operativo en particular y no tendrá que instalar software adicional. En otras palabras, no habría necesidad de adquirir nuevos equipos o licencias de software en un tiempo considerable.

Otros beneficios de utilizar Apantli corriendo del lado del servidor es que una vez que la máquina del usuario ha enviado información al servidor para ser procesada, la máquina del usuario, no tendrá más que esperar los resultados, de forma que aunque exista la posibilidad de que la máquina del usuario falle (corte de suministro de luz, error del sistema operativo, error del software, etcétera), los procesos en el servidor remoto no serán interrumpidos y no habrá que empezar de nuevo con una simulación que pudiera durar varias horas.

Cuando el equipo de un usuario reciba mantenimiento, Apantli seguiría disponible para otros usuarios y en caso de que el software se mejore con nuevas capacidades o reparación de *bugs* (fallos conocidos), el usuario utilizará siempre la versión más reciente del software. Lo que a su vez mejorará la calidad en la ayuda que un usuario pueda recibir de otro u otros usuarios, por el hecho de que trabajarían con la misma versión.

En la primera etapa de desarrollo de Apantli, se trabajaría como software de código abierto, lo que permitiría que los usuarios que así lo deseen, ejerzan las libertades (con todos sus beneficios) que el software de código abierto ofrece [5] y que se describen más adelante.

Por último para la parte del procesamiento de la información, Apantli perseguirá la meta de utilizar de manera más eficiente los recursos y procedimientos. En este caso, con el ajuste numérico de la curva tipo.

Esta Tesis, no cuestiona la efectividad teórica de los métodos gráficos, pero si la conveniencia de seguir utilizándolos tolerando la incertidumbre asociada a la precisión del usuario con la lectura e interpretación de gráficas.

En el caso de las pruebas de presión que utilizan la solución fuente lineal y la construcción de curvas tipo, es posible realizar el empate de dichas curvas de manera numérica con una precisión tal que se minimice el error humano. Además de reducir el tiempo que toma realizar la interpretación.

El programa que aquí se presenta logró calcular con éxito varios ejemplos de pruebas de interferencia por el método del *Match Point*, ajustando numéricamente las curvas asociadas al procedimiento.

Una vez tratados los puntos anteriores, surge la pregunta de ¿quién y por qué? podría participar en el desarrollo y aplicación del nuevo software. Consideramos que el potencial necesario está en los propios estudiantes mexicanos.

1.5. DESARROLLO DE SOFTWARE EN LA CARRERA DE INGENIERÍA PETROLERA

Los autores de este trabajo deseamos ofrecer un panorama alternativo de la importancia y oportunidad que tiene el desarrollo de software de Ingeniería Petrolera, así como la oportunidad de que se involucren de manera activa en la utilización y ampliación de un catálogo de software hecho principalmente por y para los alumnos de la carrera.

Basados en nuestra experiencia como alumnos, hablaremos del uso y desarrollo de software aplicado en la carrera de Ingeniería Petrolera.

Actualmente, el plan de estudios de la carrera de Ingeniería Petrolera, contempla dos asignaturas explícitamente aplicadas a la programación. La primera es la materia de “Computación para ingenieros” cuyo objetivo es:

> *El alumno conocerá la importancia de la computación e informática como herramienta para su desempeño académico y profesional de ingeniería.*
>
> *Empleará el software básico que le permita generar productos que resuelvan problemas matemáticos y de ingeniería.*

La segunda asignatura es “Programación Avanzada” (la cual sirve como antecedente obligatorio para la materia de “Simulación numérica de yacimientos“), el objetivo es:

> *El alumno programará y aplicará software comercial para analizar el desempeño de algoritmos matemáticos aplicables a la ingeniería petrolera.*

Es claro entonces que al tener dos materias de programación en el plan de estudios, el ejercicio de plantear un diagrama de flujo para resolver un problema y programarlo no debe ser ajeno a ningún estudiante de Ingeniería Petrolera.

Con mucho agrado, constatamos el interés de varios profesores por promover el ejercicio de las habilidades de programación, aplicadas a la solución de los problemas que se expusieron en varias de las asignaturas de la carrera. No obstante percibimos algunos aspectos que no estimulan de la mejor manera, el interés y el aprendizaje de los alumnos.

El primer obstáculo radica en que algunos alumnos no desarrollaron suficientes habilidades en el curso de "Computación para ingenieros", lo que deriva en que la materia de "Programación Avanzada" se tiene prácticamente el primer contacto con la programación y es necesario invertir tiempo de la asignatura para cubrir dichas deficiencias en lugar de comenzar inmediatamente con la aplicación de los conocimientos de programación a la solución de problemas como lo especifica el temario.

En el temario de "Programación Avanzada", se indica que los lenguajes de programación deben ser *Fortran 95* y *MathLab*. Asumimos que la capacidad y popularidad de Fortran en el ámbito científico, puede ser una de las razones por las que se incluye en el temario. No cuestionamos la efectividad de *Fortran* como herramienta de programación en el ámbito científico, de hecho *FORTRAN* también podría ejecutarse en un navegador como *scripts ".CGI"* y con algunas librerías, puede acceder a bases de datos (de manera limitada) y archivos de *Excel*, pero si consideramos bajo los argumentos que a continuación detallamos, que al igual que "*C/C++*", *Fortran* no es la herramienta más didáctica para el aprendizaje de los temas descritos en los temarios de las asignaturas citadas.

Algunos de estos inconvenientes (que nosotros mismos padecimos) son:

a) Existen varios compiladores de *Fortran* tales como: *Absoft, Lahey Comupter Systems, NAGWare, G95, GNU Fortran, PathScale, The Portland Group, f2c, IBM XL, Sun Studio Compiler y Visual Fortran*, por mencionar algunos [6]. Cada uno utiliza su propia interfaz y algunas particularidades que ocasionalmente conducen a que un código que funciona en un compilador como Lahey, no lo

haga correctamente en Visual. Por ejemplo, está la manera en que Lahey facilita la escritura de módulos, pero que no lo hace bajo el sistema de "Proyectos" de Visual, por lo que en este último no se compilará correctamente.

En *Fortran* se requiere forzosamente compilar una sección principal con todo y sus módulos, aún si alguna modificación sólo se hizo en la sección principal y se mantuvieron los módulos intactos. Esto es una consecuencia lógica de generar un archivo "ejecutable" (.exe para el caso de *Windows*). Lo que nos conduce nuevamente al problema de los sistemas operativos. ¿Se correrá el programa en *Windows*, se correrá en *LINUX*? ¿Habrá que crear un ejecutable para diferentes sistemas operativos?

b) Si bien existen algunos libros sobre programación en *Fortran* e información en la red, la mayoría tratan estrictamente del lenguaje y no de la interfaz. Dándose el caso de alumnos que quizá entienden un problema, lo pueden plasmar en un diagrama de flujo, entienden la sintaxis que deben utilizar para escribir el código, pero no consuman la solución del ejercicio debido a que no saben utilizar la interfaz del compilador.

c) Si en las referencias bibliográficas no se encuentra la respuesta de algún procedimiento, la probabilidad de encontrar dicha respuesta en la red también es escasa, ya que no existen demasiados foros o comunidades virtuales, donde los usuarios puedan solicitar ayuda y responder de forma clara y didáctica a preguntas de otros usuarios, este tipo de ayuda es fundamental, cuando se quiere solucionar detalles que en muchas ocasiones no se tratan en los manuales del propio fabricante.
   Aunado a los puntos anteriores, la mayoría de estas fuentes están disponibles únicamente en inglés, por lo que los alumnos que no tienen los antecedentes suficientes de comprensión de dicho idioma, tiene una desventaja aún mayor de fuentes para consultar.

d) Las licencias de los compiladores más estables, son poco accesibles (de USD $249 hasta USD $795 para la versión académica de *Lahey* [7]) para la mayoría de los estudiantes que quisieran adquirir el compilador y usarlo fuera de algún laboratorio escolar.

   Si bien existen opciones gratuitas como *Silver Frost*, *G95* o *GNU Fortran*, estas versiones son desarrolladas una comunidad que como lo aclaran en su sitio,

incorporan la mayoría (pero no todas) de las funciones de *Fortran 95*, 2003 y 2008 [8].

Por tal motivo se llegó a la conclusión que en muchos casos, los alumnos dedican mucho tiempo aprendiendo a utilizar la interfaz y a resolver problemas no documentados de ésta, que en el ejercicio como tal de programar para resolver problemas.

e) En muchas ocasiones, la instalación de estos compiladores requiere de permisos de "administrador" de los equipos, lo que dificulta que sean instalados en laboratorios escolares y espacios públicos.

La solución a todos los inconvenientes que hemos descrito anteriormente, se enfocan hacia:

- Utilizar un lenguaje de programación cuya sintaxis sea diseñada y proporcionada por un único proveedor.
- Que dicho lenguaje tenga una amplia documentación y presencia en foros de ayuda y discusión.
- Que el lenguaje sea permanentemente revisado y mejorado por usuarios y proveedor de manera conjunta.
- Que sea gratuito y de código abierto.
- Que sea multiplataforma (*Windows, Linux, UNIX, Mac*, etcétera).
- Que no requiera de muchos recursos (*hardware*) para operar.
- Que puedan ser utilizados sin requerir instalaciones o permisos de administrador para su ejecución.
- Que tenga una interfaz simple, para que los alumnos puedan comenzar a utilizarlo de inmediato y enfoquen la mayor parte de su tiempo y atención a cumplir con el objetivo de las asignaturas mencionadas, para que posteriormente tengan más herramientas y confianza para utilizar *Fortran* si así tuvieran que hacerlo.

El software propuesto es: *PHP*.

1.6. LENGUAJE PHP

PHP es un leguaje muy popular, multipropósito, especialmente adecuado para el desarrollo en la red y puede estar contenido en documentos HTML. Apareció en 1995 y para octubre de 2008 ya se encontraba en el 38% de los sitios de internet [9].

Muchos de los sitios de la UNAM están diseñados con *PHP*, por lo que si el lector ha revisado alguna vez su historial académico, entonces habrá utilizado un sistema hecho en *PHP*.

PHP incluye cientos de funciones y librerías en su distribución más reciente, pero por su naturaleza de código abierto y su popularidad, diariamente miles de usuarios alrededor del mundo mejoran y fabrican nuevas funciones y librerías de las cuales algunas se llegan a incorporar al núcleo del lenguaje y otras tantas están disponibles para quien quiera utilizarlas. Es un lenguaje que también está bajo revisión de sus propios usuarios, lo que acelera la detección de errores de programación y su correspondiente arreglo, en contraste con la mayoría de los lenguajes comerciales como *Fortran*.

PHP es un lenguaje derivado del lenguaje *C* y desde la versión 5 (liberada en 2004), incorporó la capacidad de utilizar Programación Orientada a Objetos (POO) [10].

En cuanto a capacidad de *PHP* para interactuar con otros programas o lenguajes, podemos mencionar que tiene un sistema muy completo de comunicación con los principales servidores de bases de datos (*MySQL, SQL Server, Oracle, PostgreSQL*).

Recientemente, Microsoft hospeda un proyecto de código abierto llamado *CODE PLEX* [11]. Ese proyecto, hospeda a su vez al proyecto *PHP EXCEL*, el cual es un conjunto de clases (grupo de funciones) que permiten a *Excel* leer y crear archivos de *EXCEL 2003 - 2007, PDF y HTML*, esto es muy importante, desde dos puntos de vista:

1. Mucha de la información en la industria se registra, se interpreta y se distribuye en una hoja de cálculo. Como se ha mencionado, este no es un obstáculo para *PHP*.

2. Otra parte muy importante de la información, sobre todo a una escala más allá de una computadora, se actualiza y consulta las 24 horas del día, desde diferentes ubicaciones. Por lo que la capacidad de tener la información ordenada, actualizada y siempre disponible, ha sido la tarea de las bases de datos con las que *PHP* trabaja perfectamente.

Quizá una interrogante para quienes no han programado en *PHP* sea si este es capaz de hacer cálculos matemáticos complejos.

Para responder la pregunta habría en principio que definir ¿Qué es un cálculo complejo? Por ejemplo, en el trabajo desarrollado en esta Tesis, en el capítulo 3 demuestra que *PHP* permitió hacer un programa de Ingeniería Petrolera, utilizando métodos numéricos directos e iterativos a niveles muy aceptables de precisión.

PHP soluciona el problema de los sistemas operativos y sus respectivas versiones por una simple cuestión: Se ejecuta en un navegador de *internet*.

Algunas ventajas y razones más fuertes para programar en *PHP* son:

1. Todos los sistemas operativos con interfaz gráfica cuentan con un navegador. Inclusive muchos de los nuevos teléfonos móviles conocidos como "*Smart Phones*".
2. En equipos con restricciones administrativas, un usuario visitante rara vez tendrá restringido el uso de un navegador.
3. Si en un lugar no hay acceso a *internet* como tal, fácilmente se puede emular un entorno de *internet* local con el que se pueda programar y ejecutar *PHP*.
4. Un navegador de *internet* también es gratuito.
5. Es posible ejecutar desde PHP programas compilados en otros lenguajes como Fortran, C, C++, etcétera.

PHP 5 incluye los conceptos de clase, objeto, método, herencia y muchos otros propios de la programación orientada a objetos. Aquellos que no estén familiarizados con estos conceptos, deberían consultar la documentación oficial de PHP disponible en [12].

Capítulo 2 CONSTRUCCIÓN DE LA PLATAFORMA

2.1. ANTECEDENTES

Dado que la enseñanza de *PHP* en este trabajo está enfocada hacia la aplicación, haremos comentarios solo sobre aspectos de configuración y operación de la plataforma. Es responsabilidad del usuario aprender por su cuenta la sintaxis de *PHP*.

Al final de este capítulo, citaremos diversas fuentes con cursos en línea (en inglés y en español) donde el usuario puede aprender de manera didáctica y veloz la sintaxis de *PHP*. En [13] hay documentación y manuales sobre el uso de los formularios en *HTML*.

Debido al objetivo que perseguimos, algunos de esos recursos serán provistos por los autores de este trabajo, de manera continua y permanentemente están disponibles en [14].

2.2. REQUISITOS DE SOFTWARE

Sistema Operativo	*Windows 98, NT, 2000, XP, Vista, 7*	*Linux* (Probado con: *SuSE, RedHat, Mandrake y Debian*)	*Mac Os X* (10.4 ó superior)
Entorno Web	*XAMPP*	*XAMPP*	*XAMPP*
IDE	*NetBeans 6.9* *Alternativa a Net Beans: JEdit*	*NetBeans 6.9* *Alternativa a Net Beans: JEdit*	*NetBeans 6.9* Alternativa a *Net Beans: JEdit*

2.3. EL AMBIENTE DE PROGRAMACIÓN

A lo largo del capítulo uno, mencionamos que una de las ventajas del desarrollo de aplicaciones *web* es la posibilidad de programar y ejecutar nuestros programas sin la necesidad de instalar software especializado, prueba de ello es que basta con un navegador (*Internet Explorer, Mozilla Firefox*, *Google Chrome, Opera, Safari* etcétera.) y un editor de texto (bloc de notas, *vi*, etc.).

A continuación explicaremos brevemente cómo funciona el ciclo de transmisión y procesamiento de datos usando software del lado del servidor, este proceso se puede aprecia en la figura 2.1

Fase 1.- Algún dispositivo (*PC, laptop, smart phone, pda*) hace una petición a través de un navegador. Esta información es recibida por el intérprete de *PHP*, quien le hará llegar la petición al servidor (*Apache* en este caso).

Fase 2.- Una vez que el servidor recibe la petición, podría consultar una base de datos para consultar información de otras fuentes o bien, si la información está contenida dentro de la misma petición, empezará a procesarla, para generar un archivo de salida.

Fase 3.- Este archivo de salida, se genera al instante, dependiendo de la petición que cada dispositivo haya realizado y nuevamente *PHP* es el puente entre el servidor, que cruza *internet* y despliega los resultados en el navegador del usuario.

Si queremos facilitar aún más esas tareas o si debemos trabajar sin una conexión a internet, podemos instalar algunos programas que emulen una conexión en nuestra propia máquina y otros cuantos programas que simplifiquen escribir código. Este capítulo comenzará con la descripción de cómo preparar un ambiente de desarrollo local (en nuestro equipo, emulando una conexión a internet). Para tal efecto, hemos preparado una serie de videos disponibles en el sitio dedicado a este trabajo: *www.atlahua.com/tesis*, donde se explica paso por paso.

Si de momento el lector no está en posibilidades de ver los videos o si solo requiere una guía sin mucho detalle para recordar una instalación previa, ofrecemos la siguiente descripción del proceso:

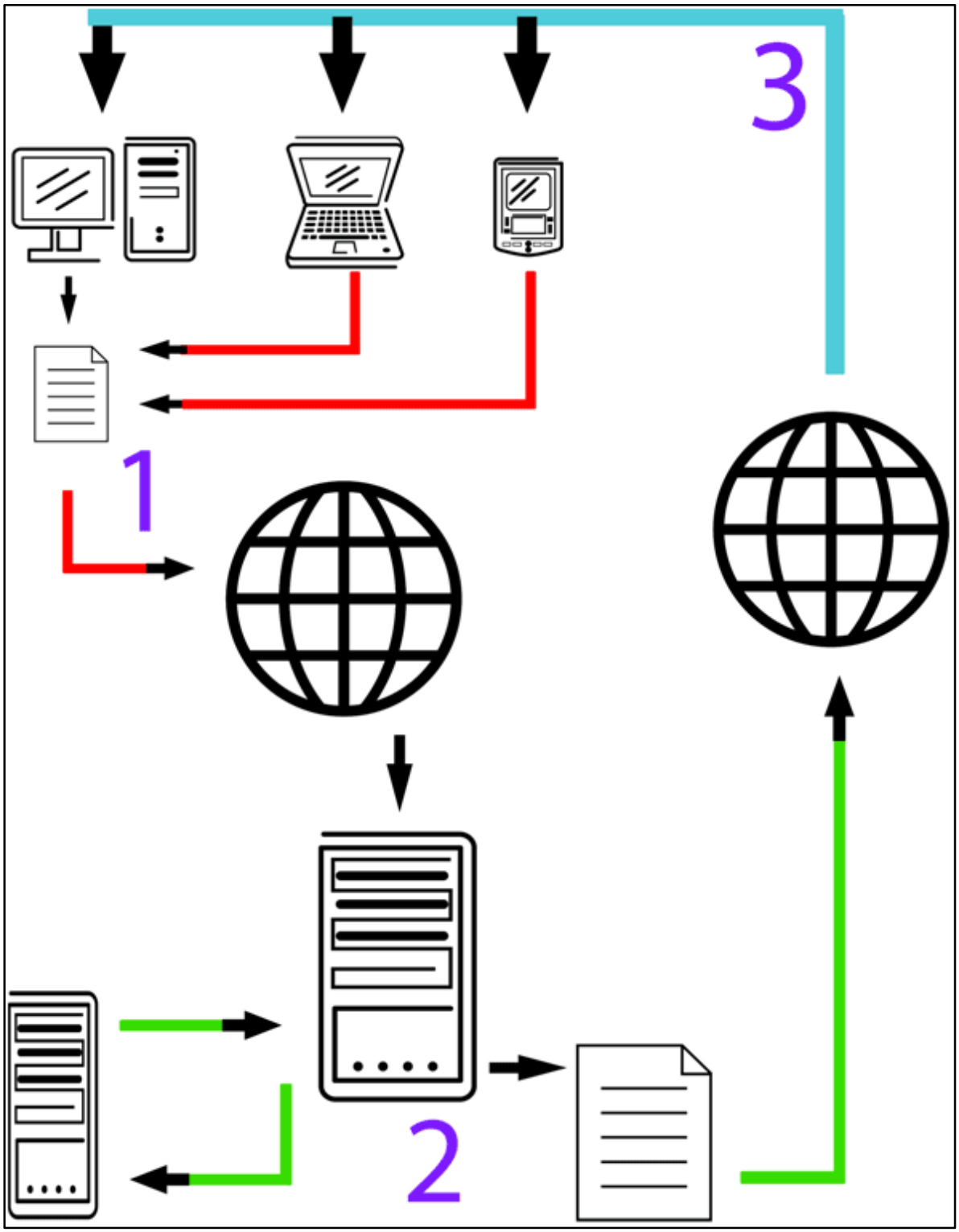


FIGURA 2.1PROCESO DE INTERCAMBIO DE INFORMACIÓN USANDO SOFTWARE DEL LADO DEL SERVIDOR

Hay dos ambientes para programar en *PHP*.

2.3.1. AMBIENTE REMOTO

Si tenemos una cuenta de *FTP* con algún servidor que tenga funcionando *PHP* (casi todos lo hacen). Podemos subir nuestros archivos .php y verlos funcionar de inmediato, sin configurar nada. Se puede abrir una cuenta en algún servicio de hospedaje (hosting) gratuito con *PHP* 5 y *FTP*. Uno de tantos está disponible en: *http://www.000webhost.com/*. Los usuarios interesados en unirse al equipo de Apantli, recibirán gratuitamente una cuenta en el servidor de pruebas del proyecto, para incorporarse a este proyecto, pueden ponerse en contacto al correo electrónico: apantli@atlahua.com

Si ya se tiene una cuenta para uno de estos servicios, ir al inciso 2.3.4.

2.3.2. AMBIENTE LOCAL

Está dirigido a los usuarios que no cuentan con una cuenta en un servidor remoto, que no cuentan con acceso a internet o que quieren hacer pruebas en sus equipos antes de publicar el trabajo en la red.

2.3.3. INSTALACIÓN DE XAMPP

Después de descargar los archivos necesarios (*XAMPP*, *NetBeans* y/o Jedit), procederemos a instalar *XAMPP*. Aceptando todos los valores por default, *XAMPP* se instalará en *c:/xampp/* (figura 2.2) con la siguiente ruta de archivos:

FIGURA 2.2 INSTALACIÓN DE XAMPP

Identifique el directorio llamado *htdocs* (que dependiendo de algunos entornos, puede llamarse '*www*' o '*public_html*'). Hablaremos de la importancia de dicho directorio más adelante.

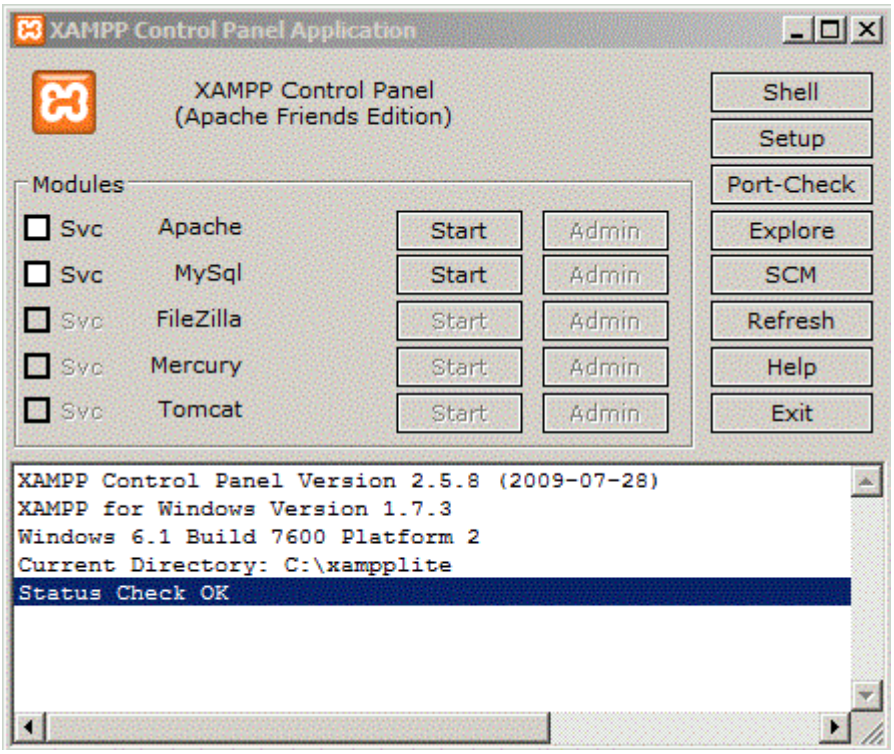


FIGURA 2.3 PANEL DE CONTROL XAMPP

Ejecutamos el programa *xampp-control.exe* que es el panel de control del sistema e iniciamos el servicio del servidor apache (figura 2.3 Panel de control xampp)

Para verificar la instalación, hacemos *clic* en el botón de *admin*, el cual abrirá el navegador predeterminado del sistema e intentara acceder a '*localhost*'. Si vemos alguna de las siguientes pantallas (figura 2.4 Arranque inicial de xamppy figura 2.5 arranque regular de xampp), sabremos que el entorno está trabajando.

FIGURA 2.4 ARRANQUE INICIAL DE XAMPP

Si es la primera vez que hacemos esta operación, debería aparecer la siguiente pantalla, donde podemos seleccionar el idioma en que se presenta la información del entorno:

Las demás ocasiones aparecerá una pantalla así:

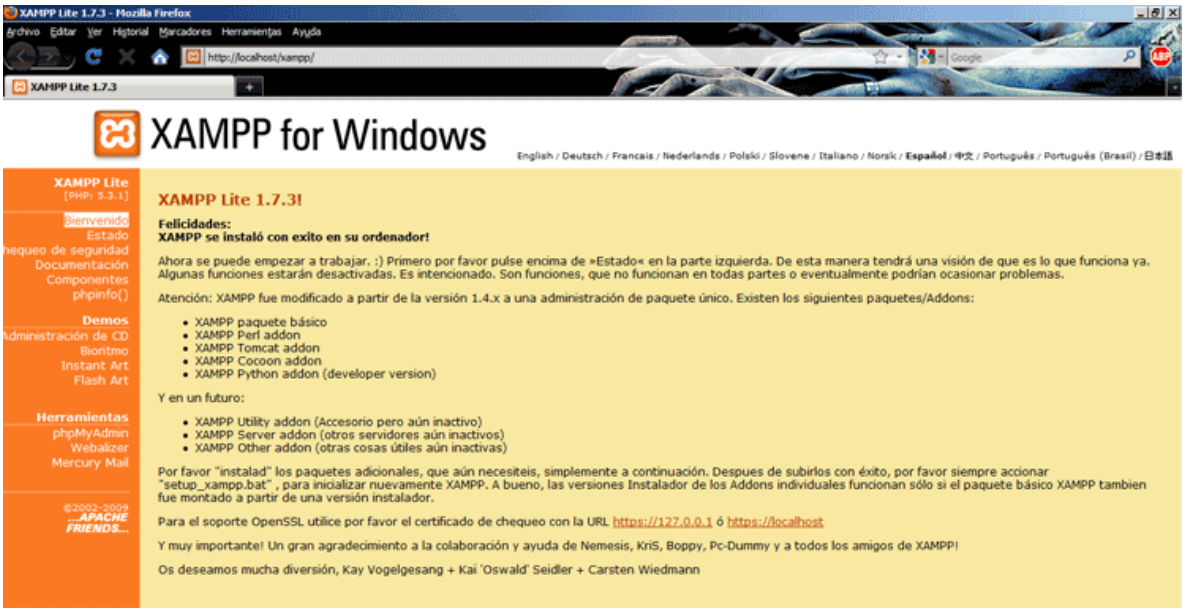


FIGURA 2.5 ARRANQUE REGULAR DE XAMPP

Si se observa la barra de direcciones, se puede apreciar: *http://localhost/xampp/* esto quiere decir que *localhost* será la ruta principal y de ahí partirán los proyectos que desarrollemos (figura 2.6 dirección de localhost).

FIGURA 2.6 DIRECCIÓN DE LOCALHOST

2.3.4. EJEMPLO DE UN PROGRAMA SENCILLO EN PHP

Vamos a hacer nuestro primer programa en *PHP*. Para esto podemos utilizar cualquier editor de texto, pero recomendamos un programa como *Jedit* o *NetBeans* para facilitar la escritura de código, a largo plazo. En este ejemplo usaremos *Jedit*.

Abrimos un archivo nuevo. Y tecleamos el siguiente código (figura 2.7 código php, un breve programa):

```
<?PHP
$nombre1='Atlahua';
$nombre2='Marco';
echo 'Software de '.$nombre.' y '.$nombre2.' para el mundo';
?>
```

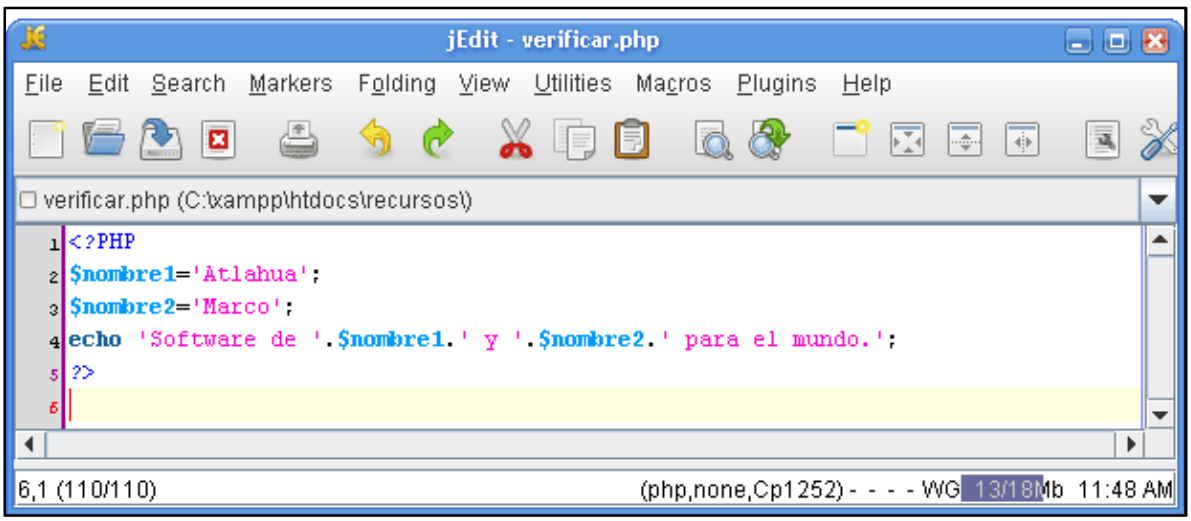


FIGURA 2.7 CÓDIGO PHP, UN BREVE PROGRAMA

Quizá alguien con experiencia programando en *FORTRAN* haya notado que la declaración de variables no requiere un tipo específico como: número, cadena, booleana etcétera. Ya que *PHP* tiene la capacidad (pero no la limitante) de detectar el tipo de variable de acuerdo al contexto en que se utiliza. Esto es que *PHP* no es tan estricto como otros lenguajes para manejar las variables. Un caso práctico es el manejo de arreglos ya que a diferencia de FORTRAN, *PHP* no requiere determinar el tipo de variables que contendrá un arreglo que pueden ser de diferentes tipos a la vez. El programador tampoco debe reservar y desalojar la memoria que un arreglo requiere para funcionar, lo que desde un punto de vista práctico y didáctico es muy útil en el aprendizaje permitiendo sentar las bases para después utilizar lenguajes , reglas y técnica más complejas.

2.3.5. UBICACIÓN DE ARCHIVOS PHP

Los archivos deben colocarse dentro del directorio *htdocs* (figura 2.2). La razón es porque el servidor está configurado para interpretar únicamente los archivos *PHP* que se encuentren en dicho directorio. *Para más información sobre esta condición, consulta la documentación de Apache y PHP.*

Así pues, para guardar el archivo que escribió en el paso 3 con el nombre de verificar.php, creamos una carpeta llamada “recursos”, dentro del directorio *htdocs*, quedando la estructura de archivos de la siguiente manera (figura 2.8 estructura de archivos xampp).

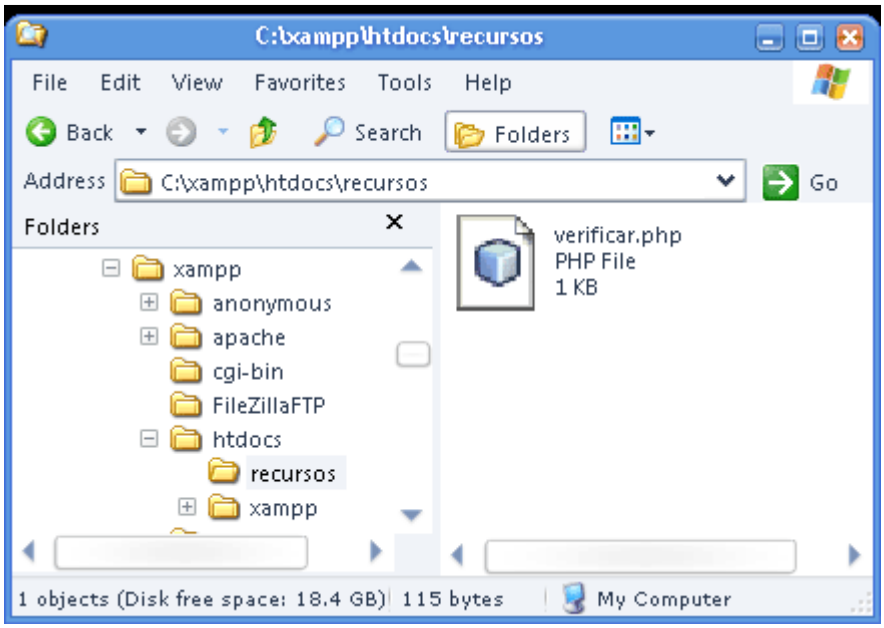


FIGURA 2.8 ESTRUCTURA DE ARCHIVOS XAMPP

(Los iconos pueden variar dependiendo de los programas predeterminados para abrirlos).

Finalmente, para saber si lo hicimos bien, abriremos un navegador e ingresaremos la siguiente dirección: *http://localhost/recursos/verificar.php*. Se debería ver en la barra de direcciones del navegador algo similar a la figura 2.9 ejecución de un programa en php.

FIGURA 2.9 EJECUCIÓN DE UN PROGRAMA EN PHP

2.3.6. TRANSFERENCIA DE ARCHIVOS AL SERVIDOR

Para guardar los archivos en el servidor remoto y que todos puedan utilizarlos a través de internet, necesitamos un gestor de *FTP* (Protocolo de transferencia de archivos). Nosotros utilizamos *Filezilla* por ser una herramienta gratuita y multiplataforma. Una vez instalado *Filezilla*, ejecutamos el programa y veremos

la interfaz (figura 2.10). En la parte superior hay campos que deben ser llenados en orden (de izquierda a derecha), de la siguiente manera:

En "servidor" va la dirección ftp que nos asignaron los proveedores del hospedaje o en caso de tener una cuenta para este proyecto, proporcionada por los autores, sería algo parecido a:

Servidor: ftp.atlahua.com	Nombre de usuario: marco@atlahua.com
Contraseña: **********	Puerto: 21

Posteriormente hay que hacer clic en "Conexión rápida"

Si la conexión fue realizada exitosamente, podremos ver dos espacios con estructuras de archivo similares a las de la figura 2.10 transferencia de archivos al servidor.

Uno de esos espacios está identificado como "Sitio Local" que contiene los archivos de nuestro equipo. El otro se identifica como "Sitio remoto" que contiene los archivos en el servidor que está en internet. Para enviar un archivo o una carpeta de nuestro sitio local a nuestro sitio remoto, tan solo hay que arrastrar y soltar (*drag & drop*) los documentos deseados, de un espacio al otro (justo como lo hacemos para mover archivos entre ventanas).

¡Listo! Ya se un entorno de desarrollo local funcionando, ya sabemos dónde guardar nuestros archivos .php y también cómo verlos funcionando.

Ahora podemos entrar de lleno a explicar la estructura de archivos y cómo se añaden módulos a la plataforma.

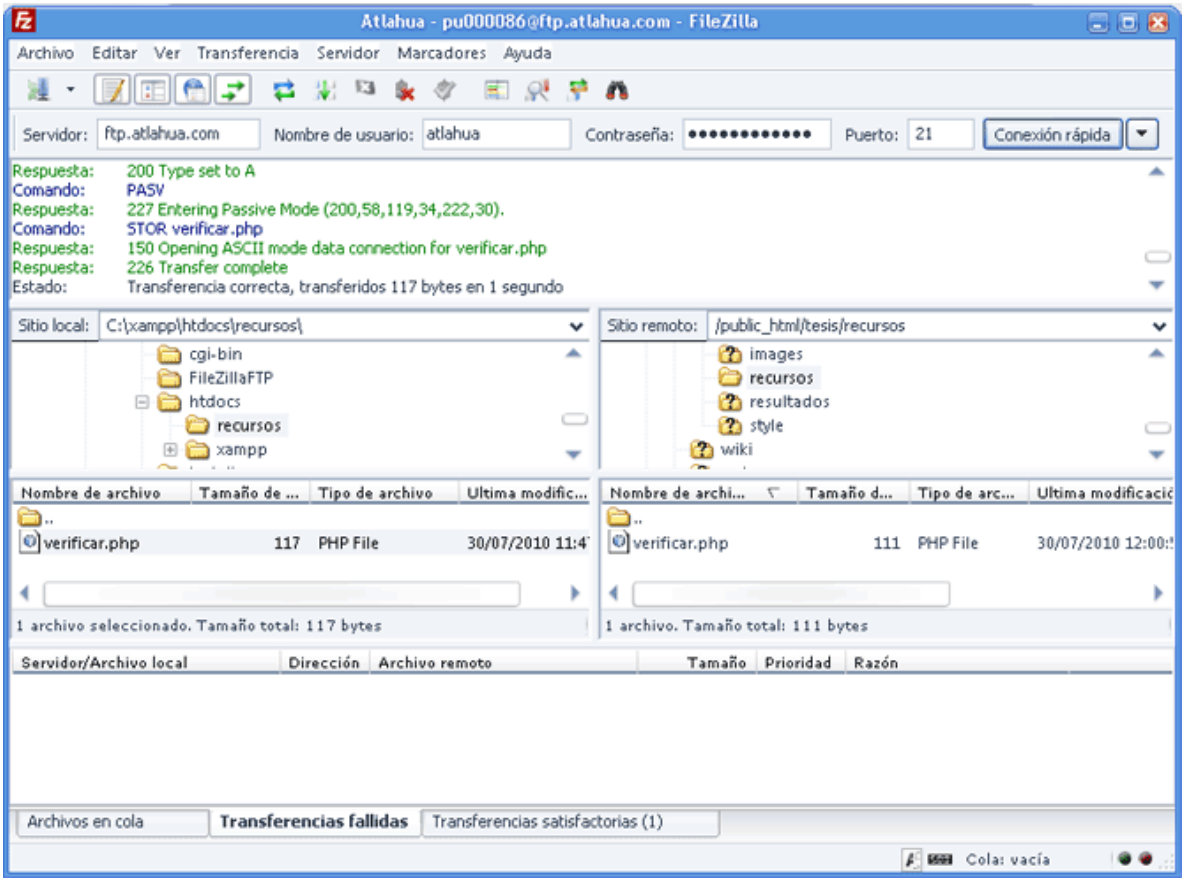


FIGURA 2.10 TRANSFERENCIA DE ARCHIVOS AL SERVIDOR

2.4. PROGRAMACIÓN ORIENTADA A OBJETOS

De acuerdo con la empresa de desarrollo de software *Sun Microsystems*, la idea de la Programación Orientada a Objetos (POO) es que una aplicación se puede considerar como una colección de unidades individuales, llamadas objetos, que interactúan entre sí. Los programas tradicionales pueden considerarse como una colección de funciones o como una lista de instrucciones de programación. Una explicación alternativa de la POO aplicada a la Ingeniería Petrolera, puede facilitarse con el siguiente ejemplo:

Se propone que los alumnos de la carrera de Ingeniería Petrolera tienen el objetivo de crear una colección de software que resuelva los problemas expuestos en las diferentes asignaturas del plan de estudios, independientemente, del perfil que tengan, cada unos de los alumnos (perforación, producción, yacimientos o evaluación de proyectos), habrán ciertos

procedimientos que pueden ser utilizados por dos o más áreas como por ejemplo el cálculo de un índice de productividad, el cálculo de algunas propiedades de los fluidos, el cálculo de presión de fondo, etc.

Se parte de la necesidad de conocer el factor de volumen del aceite (Bo) utilizando las correlaciones conocidas, para posteriormente resolver otro problema específico del área de cada equipo. Algunas formas de hacer el programa son:

a) Cada equipo hace su propio programa para determinar el Bo, dentro del mismo archivo con el resto del código fuente.
b) Cada equipo hace su propio programa para determinar el Bo, en un archivo separado del resto de su código fuente.
c) Uno de los equipos hace un programa para determinar el Bo, en un archivo separado del resto de su código fuente utilizando POO y lo comparte con el resto de los equipos.

Si el lector ha participado en algún proyecto de programación en equipo, quizá mientras estuvo leyendo los incisos anteriores, se fue imaginando una serie de problemas con sus respectivas soluciones para cada uno de ellos. Para aquellas personas que no encuentren mucha diferencia entre la eficiencia de los tres métodos, hacemos los siguientes comentarios:

a) La función o subrutina que realice el cálculo de Bo, usará variables que no podrán utilizarse en el resto del programa (o de lo contrario podrían fácilmente conducir a un resultado no deseado).

   Si se hace alguna modificación a la parte del Bo o al complemento del programa, deberán compilarse ambas partes, aunque sólo una de ellas haya sufrido una modificación.

   Dependiendo de la complejidad del problema y la técnica de programación, la cantidad de líneas de código será mayor y si no tiene el código comentado con una buena descripción, se dificultará el mantenimiento del código para un programador que no haya sido el autor original.

b) Al estar el cálculo del Bo en un archivo separado, se elimina la necesidad de compilar ambas partes (por lo menos en *PHP* ya que en Fortran se debe

compilar junto de cualquier forma) y se facilita dar mantenimiento sólo a la parte del código del Bo.

Persiste el problema de las variables, ya que pese a que el código está en un archivo separado es una práctica común hacer un '*include*' (en *Fortran, C*, *PHP*, o su equivalente para otros lenguajes) para incorporar un archivo a otro, con todo y sus variables.

c) Al estar el cálculo del Bo en un archivo separado, se elimina la necesidad de compilar ambas partes. Al utilizar POO, los equipos únicamente tienen que crear un nuevo "Objeto", a partir de la "Clase Bo" y no tienen que preocuparse (ni siquiera enterarse) del procedimiento o del nombre que se utilizó para las variables en la "Clase Bo" figura 2.11.

El hecho de utilizar POO, lleva implícito algunas directivas de programación que facilitarían a otro programador, entender y mantener el código. En otras palabras: se programa con un formato "universal". Si se hace alguna mejora o corrección a la "Clase Bo", el resto del programa de cada equipo se verá beneficiado automáticamente sin tener que modificarse o compilarse nuevamente.

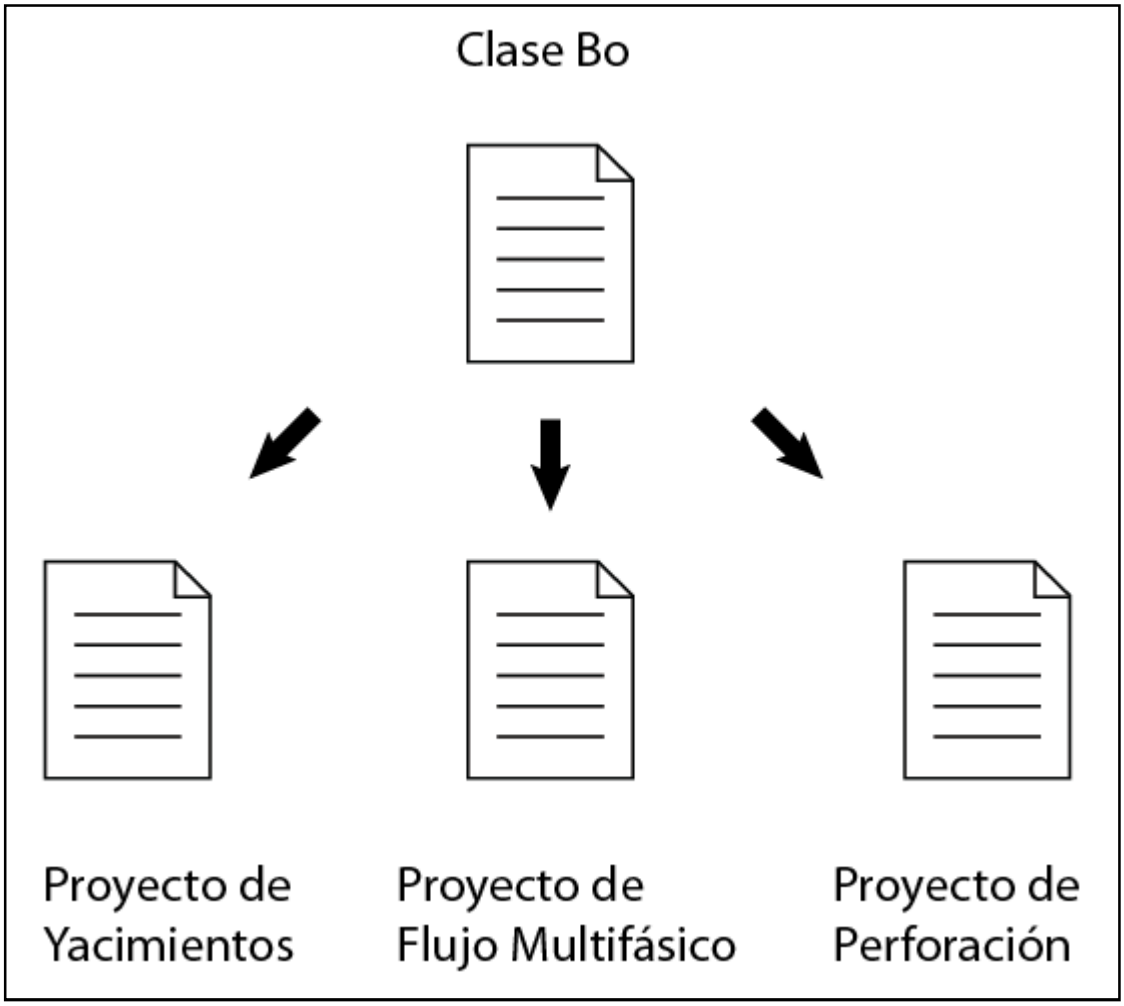


FIGURA 2.11 APLICACIÒN DE LA PROGRAMACIÒN ORIENTDA A OBJETOS

claseBo.php *(el archivo externo)*

```
<?php
class claseBo {
var $Bo; //La variable para Bo
var $var1; //Cualquier variable involucrada en la correlación
var $var2; //Otra variable involucrada en la correlación
function set_Bo($var1,$var2) {
        $Bo= ($var1+$var2)*0.5; //Un cálculo simplificado de Bo
        $this->Bo = $Bo; //Asignación de Bo en modo POO
}
```

```
function get_Bo() {
return $this->Bo; //Recupera el valor del Bo
}
}
?>
```

El código anterior representa un programa simplificado para calcular Bo, hecho en POO. A continuación se ejemplifica qué tendría que hacer cada uno de los equipos para utilizar este programa:

unprograma.html

```
<html>
<head>
<title>Ejemplo</title>
</head>
<body>
        <?php
        //Se incluye la clase de Bo
        include('Classes/claseBo.php');
        //Se inicializan las variables particulares de cada equipo
        $var1=3.0;
        $var2=5.0;
        //Se crea un nuevo objeto llamado "$miPrueba"
        $miCalculoDeBo = new claseBo ();
        $miCalculoDeBo->set_Bo($var1,$var2);
        //A la función set_Bo, contenida en la clase "claseBo"
                        //recibe $var1 y $var2 como parámetros y calcula $Bo
        echo 'Bo = '.$miPrueba->get_Bo().' [scf@c.s./bl@c.y.]';
        //Se imprime el valor calculado, utilizando la función
        //get_Bo, contenida en la clase "claseBo".
        ?>
</body>
</html>
```

El código anterior es el código completo de un archivo HTML combinado con PHP. Quizá parezcan muchas líneas para quienes no estén familiarizados con este

lenguaje, por lo cual hacernos énfasis en las instrucciones sustanciales que cada equipo tendría que hacer para utilizar una única clase "claseBo" hecha por alguien más.

```
<?php
include('Classes/claseBo.php');
$miPrueba = new claseBo ();
$miPrueba->set_Bo($var1,$var2);
echo 'Bo = '.$miPrueba->get_Bo().' [scf@c.s./bl@c.y.]';
?>
```

Así, queda demostrado que el usuario final de una clase, se puede limitar a entender cómo utilizar la clase y no tanto en cómo funciona, lo que en algunos casos es conveniente para no distraerse del objetivo inicial de resolver un problema, utilizando pequeños programas (las clases) que alguien más ya ha programado y en la mayoría de los casos, mantiene y mejora.

¿Por qué orientado a objetos?

Menos código significa menos errores, que a su vez significa menos tiempo para depurar y desarrollar un programa. La programación orientada a objetos no sólo brinda una gran flexibilidad en la reutilización de código, también facilita el mantenimiento del mismo y cuenta con lineamientos estandarizados para su correcta escritura.

Permite que varias personas trabajen en un mismo proyecto, sin tener que trabajar en el mismo archivo y por ende, sin interferir con variables utilizadas por otros programadores.

Existe una amplia documentación al respecto y los lenguajes de programación tienden cada vez más a funcionar bajo esa dinámica. Un notable ejemplo es la versión 2003 del compilador Fortran, diseñada para trabajar orientada a objetos o la misma plataforma de *Visual .Net de Microsoft*.

En los próximos capítulos se abordará la teoría necesaria para comprender el método del Match Point aplicado a la interpretación de las pruebas de interferencia. En seguida se explicará el método que soluciona numéricamente el

Método Gráfico del Match Point. Posteriormente, se describirá el funcionamiento del programa, realizando simultáneamente una interpretación de una prueba, para finalmente mencionar la necesidad de ampliar y mejorar las capacidades del programa.

2.5. ARQUITECTURA DE LA PLATAFORMA

Mencionamos en el capítulo 1, un ejemplo de cómo varios equipos de ingeniería petrolera podrían trabajar para resolver problemas de sus áreas de especialización. En ese capítulo se habló de los problemas asociados a los diferentes escenarios y se propuso una solución: Utilizar programas (clases de POO) que alguien ya ha desarrollado y que son de uso común para todas las áreas. Ahora describiremos cómo llegar a ese escenario.

ARQUITECTURA MODELO VISTA CONTROLADOR

Con la evolución de las capacidades de internet y la demanda de aplicaciones, surgió una nueva forma de desarrollar software en *PHP* basada en el patrón de diseño MVC [15], que por sus siglas en inglés significa: Model, View, Controller.

Este modelo de desarrollo propone separar el funcionamiento de un programa en 3 capas (modelo, vista y controlador). Se abordará el concepto con un ejemplo (utilizando pseudocódigo para facilitar la comprensión del concepto) de un programa escrito en programación estructurada.

Cabecera:
1.- Código PHP y HTML con configuraciones iniciales
Cuerpo:
//Ajustar opciones de visualización
2.- Código HTML, CSS para la parte visual (imágenes de fondo, color, tipo y tamaño de letra, distribución de espacios).
3.- Código HTML para seleccionar el modo de cargar los datos
//Obtener los datos de la prueba desde la base de datos
4.- Código PHP para definir instrucciones de conexión a la base de datos
5.- Código PHP para conectar a la base de datos
6.- Código PHP y HTML para navegar por la base de datos y mostrar la información recuperada.
7.- Código PHP para cerrar conexión a la base de datos
//Obtener datos de un archivo de hoja de cálculo

8.- Código HTML para mostrar formulario para seleccionar archivo
9.- Código PHP para leer datos del archivo seleccionado
//*Preparación de datos*
10.- Código PHP para suavizar los datos de la prueba
//*Inicio del método*
11.- Código PHP que realiza el método del Match Point
//*Presentación de resultados*
12.- Código PHP y HTML para mostrar los resultados en la página.
13.- Código PHP y HTML para crear un archivo de hoja de cálculo descargable
//*fin del programa.*

De manera similar sería un programa para perforación o para producción donde se repetiría el código para la parte visual, para las conexiones a las bases de datos y para mostrar los resultados. Así que cada que alguien iniciara un nuevo programa, tendría que escribir estos elementos comunes.

Es entonces cuando existe la oportunidad para simplificar esa parte del desarrollo, separando el código.

Para empezar, vamos a colocar la parte visual, en un archivo a donde solo se vacíen los resultados. Este archivo se llamará "la vista":

Cabecera:
1.- Código HTML con configuraciones iniciales
Cuerpo:
2.- Código HTML, CSS para la parte visual (imágenes de fondo, color, tipo y tamaño de letra, distribución de espacios).
3.- Código HTML para seleccionar el modo de cargar los datos
4.- Código HTML para navegar por la base de datos y mostrar la información recuperada.
5.- Código HTML para mostrar formulario para seleccionar archivo
6.- Código HTML para mostrar los resultados en la página.
7.- Código HTML para ofrecer un archivo de hoja de cálculo descargable.

En otro archivo vamos a colocar el código referente a la obtención de información desde la base de datos, al cual llamaremos "el modelo 1":

Cuerpo:
1.- Código *PHP* para insertar o heredar clases
2.- Código *PHP* para seleccionar el modo de cargar los datos
//Obtener los datos de la prueba desde la base de datos
3.- Código *PHP* para definir instrucciones de conexión a la base de datos
4.- Código *PHP* para conectar a la base de datos
5.- Código *PHP* para navegar por la base de datos y recuperar la información.
6.- Código *PHP* para cerrar conexión a la base de datos

En otro archivo vamos a colocar el código referente a la obtención de información desde una hoja de cálculo, al cuál llamaremos "el modelo 2":

1.- Código *PHP* para insertar o heredar clases
//Obtener datos de un archivo de hoja de cálculo
2.- Código *PHP* para leer datos del archivo seleccionado
//Preparación de datos

En otro archivo vamos a colocar el código referente a la manipulación de información, al cual llamaremos "el modelo 3":

1.- Código *PHP* para suavizar los datos de la prueba.
//Inicio del método
2.- Código *PHP* que realiza el método del Match Point.
//Presentación de resultados
3.- Código *PHP* para crear un archivo de hoja de cálculo descargable.
//fin del programa.

Se hace una pausa para señalar algunos hechos:

En el archivo original están mezclados en exceso código *PHP* y HTML, lo que puede resultar difícil de entender y mantener. Al separar el archivo original, el código HTML quedó solo en el archivo de la vista "La Vista" y el código *PHP* en el "El Modelo", pero la consecuencia de esta acción es que no hay nada que vincule "al Modelo" con "la Vista".

Es entonces cuando creamos un tercer archivo llamado "el Controlador", que se encargará de la interacción con los otros dos archivos. El controlador quedaría de la siguiente manera:

1.- Código *PHP* para recibir la opción de dónde cargar los datos (BD u hoja de cálculo)
2.- Código *PHP* para ejecutar la carga de datos
3.- Código *PHP* ordenar el suavizado de datos
4.- Código *PHP* que ordena la ejecución del el método del Match Point
5.- Código *PHP* que recibe los resultados "del método".
6.- Código *PHP* que envía los resultados a "la vista" para ser publicados.

Un esquema de la separación se aprecia en la figura 2.12.

Como puede apreciarse, dividir un programa en estar tres capas, facilita enormemente la tarea del desarrollo.

Digamos que un programador de HTML podría dedicarse a hacer la parte visual, sin tener que conocer o entender mucho de dónde vienen los datos que se van a presentar.

Un programador de *PHP*, podrá concentrarse en hacer algoritmos y no prestar mucha atención en la parte de la interacción del usuario y de cómo presentar visualmente los resultados.

Y finalmente un programador interesado en resolver un problema de Ingeniería Petrolera (en este caso), podrá utilizar a su conveniencia, una colección de algoritmos previamente hechos, para no empezar desde cero con todos los procedimientos, y no preocuparse por la parte visual que muestre los resultados de su programa.

El mantenimiento de los programas será mucho más sencillo ya que cada especialista sabrá en cuál de los archivos se encuentra el código que debe revisar o mejorar.

Una vez comprendido este ejemplo, describamos la arquitectura MVC desde el punto de vista del usuario. En la figura 2.13 se ilustra el procedimiento de la arquitectura (Modelo Vista Controlador).

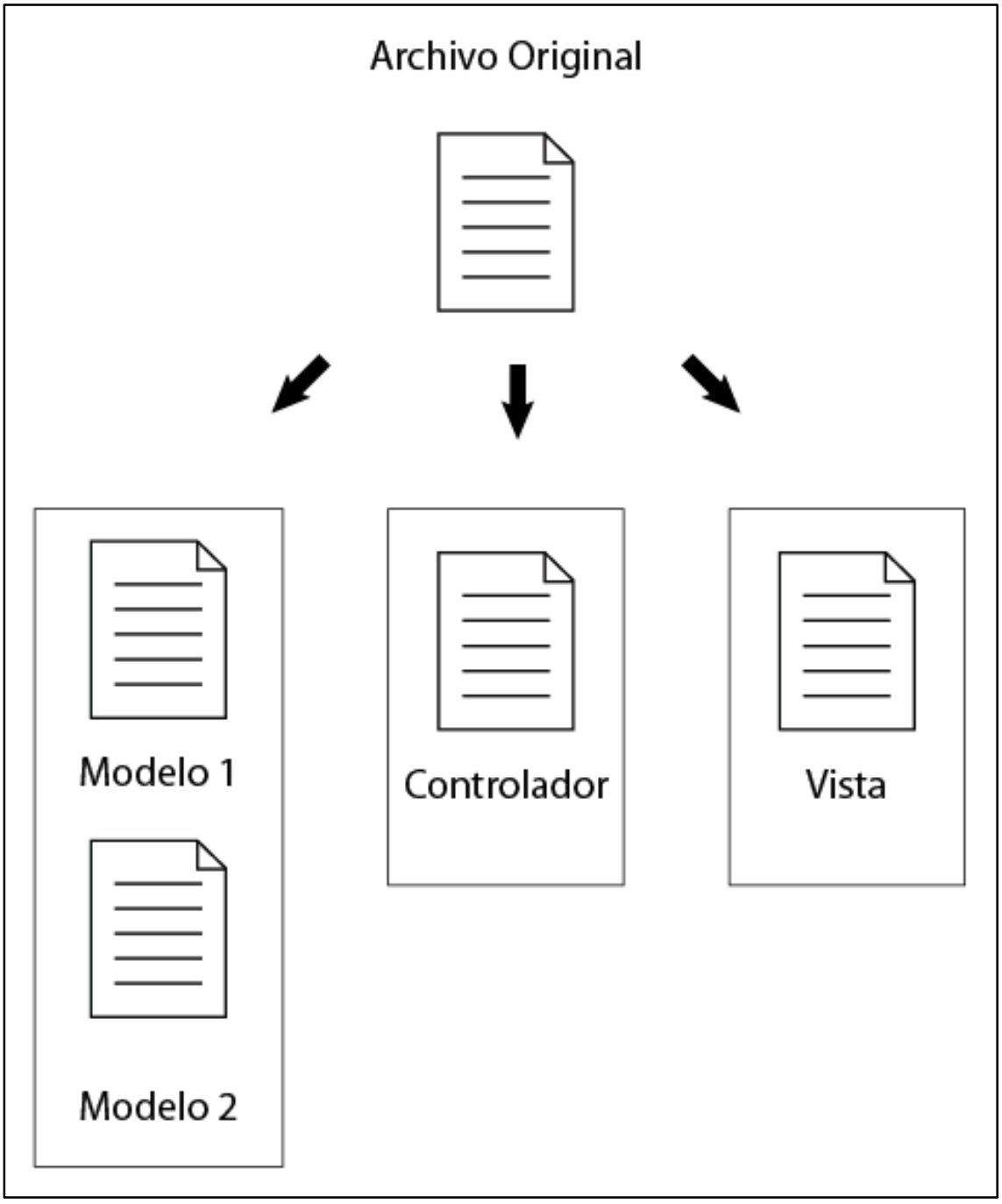


FIGURA 2.12

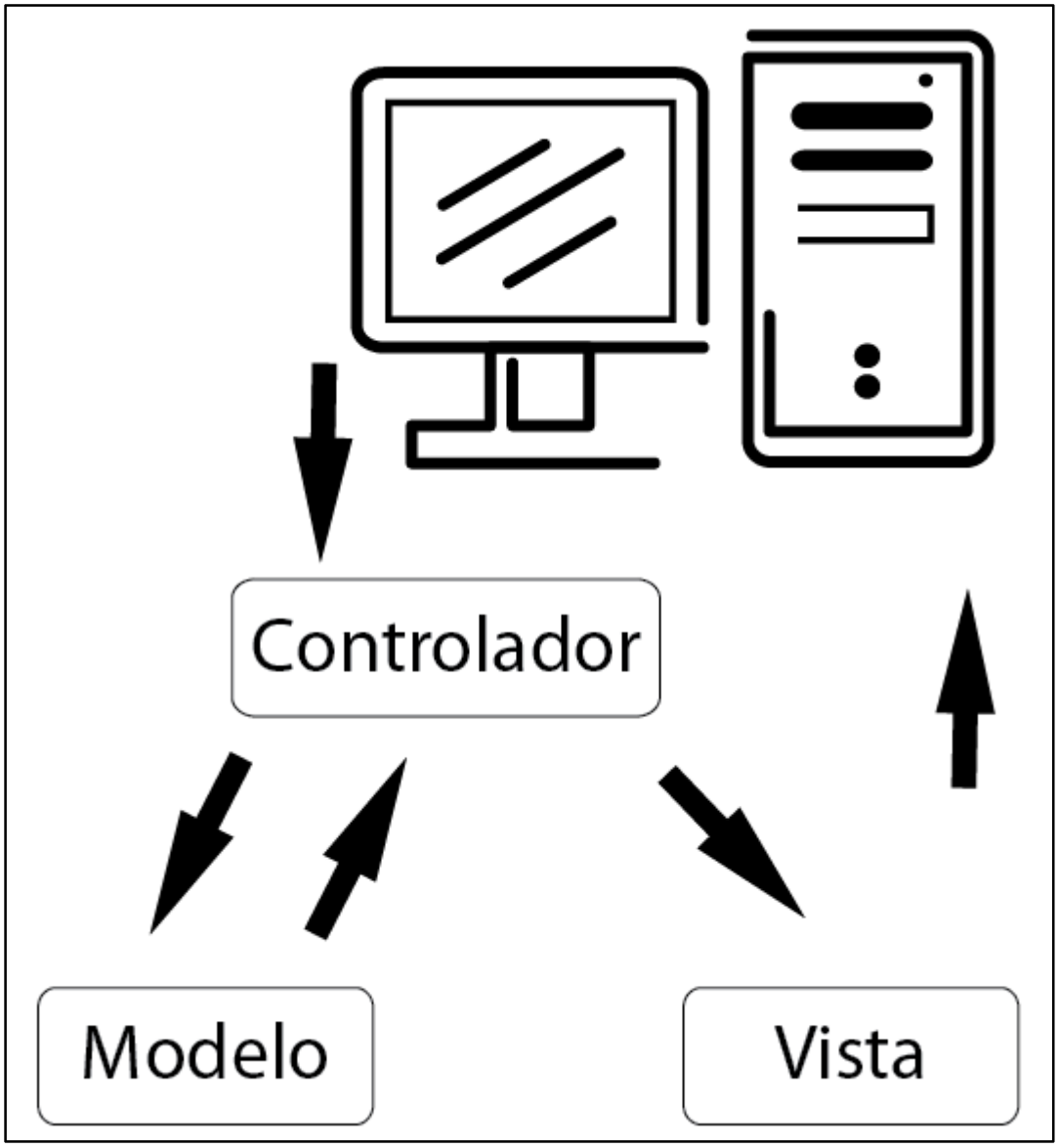


FIGURA 2.13

1.- El usuario hace la petición de acceder a la aplicación.

2.- El controlador recibe dicha petición y le ordena a la vista, que le muestre el sitio al usuario.

3.- El usuario ve las opciones que tiene para utilizar la aplicación y hace una nueva petición al controlador.

4.- El controlador recibe dicha petición y le ordena al modelo que obtenga la información necesaria de algún medio, la procese y se la devuelva procesada.

5.- El modelo trabajará con los datos y le devolverá la información requerida al controlador.

6.- El controlador enviará dicha información a la vista, para que esta la presente del modo predeterminado al usuario.

Esto puede convertirse en un ciclo de peticiones y entregas de información, siendo el controlador el único intermediario entre "el Modelo" y "la vista".

2.6. DESARROLLO DE LA PLATAFORMA

2.6.1. LA PRIMERA VERSIÓN

Inicialmente, el programa para resolver el método del Match Point, fue escrito en Fortran, como parte de un trabajo final en la materia de Caracterización Dinámica de Yacimientos. La razón por la cual se hizo en ese lenguaje fue porque **Fortran** fue el lenguaje que todos los miembros del equipo conocíamos en común, aquel programa se desarrolló en un solo archivo de aproximadamente 250 líneas del código en programación estructurada y con apenas algunos comentarios dirigidos a los desarrolladores y no al usuario final (prácticamente incomprensible e inútil para cualquiera ajeno a su escritura). En el código original (disponible en http://atlahua.com/tesis). Se puede apreciar que algunos procedimientos están repetidos. Esto es consecuencia de haber trabajado, sin utilizar POO, ya que existen variables y funciones diferentes que realizan la misma labor, pero programadas por uno u otro miembro del equipo.

La versión original lee los datos de un archivo .txt con un formato establecido, pero también se debe especificar la cantidad de pares de puntos empleados.

Además de imprimir los resultados en pantalla, el programa guarda 3 columnas en un archivo .txt que posteriormente tiene que importarse desde la hoja de cálculo para poder observar la gráfica.

2.6.2. LA SEGUNDA VERSIÓN

Este trabajo de Tesis comenzó con la tarea de transcribir el código de manera mas clara pero manteniendo el método de programación estructurada ahora en lenguaje *PHP*.

Una vez terminado, nos propusimos a realizar la gráfica sin tener que exportar el archivo de datos a la hoja de cálculo, por lo que, después de una exhaustiva

investigación, probamos diferentes librerías de *PHP* para graficar los resultados que guardamos en un arreglo de (3 columnas por "n" filas).

La librería que mejor funcionó de forma sencilla para gráficas cartesianas fue *PHPlot*. Lamentablemente aún no tiene la potencia necesaria para trazar correctamente en escala (log – log) los valores que le suministramos. En la documentación oficial [16], se menciona esta limitante.

A pesar del fracaso prematuro de la posibilidad de graficar los valores en la misma interfaz, observamos lo fácil que resultaba utilizar la librería *PHPlot*, la cual está hecha con POO. Fue a partir de esa situación que visualizamos el potencial que puede tener un software de código abierto utilizando POO, para facilitar el uso del programa a usuarios con conocimientos básicos de programación. Prueba de esto es que fuimos capaces de hacer otro tipo de gráficos con *PHPlot* hasta el día de hoy, sin tener idea de cómo está programada la librería ya que tan solo es cuestión de escribir unas 10 líneas de código para tener una gráfica cartesiana con varios elementos estilizados como título, colores, leyendas etcétera.

Así pues comenzamos a escribir el código con POO y apareció la primera versión de la clase CurvaTipo, la cual también eliminó redundancias del algoritmo original y redujo considerablemente el número de líneas de código a aproximadamente 190 líneas para el algoritmo, pero tan solo 8 para utilizar la clase .

Durante este replanteamiento, investigamos también sobre directivas para hacer el código más fácil de mantener a lo cuál surgió de inmediato prácticas de buena programación entre las que descubrimos el uso de un *Framework*.

Un *Framework* es un conjunto de herramientas para desarrollar proyectos de programación de manera más eficiente. Una definición más completa de *Framework*, se puede leer en la documentación de Symfony.

Symfony es un *Framework* gratuito y de código abierto, desarrollado inicialmente por Fabien Potencier a través de Sensio Labs. [16]

Además de ofrecer un *Framework*, Symfony ofrece suficiente documentación y material didáctico para desarrollar software con prácticas de buena programación, utilizando la arquitectura Modelo Vista Controlador (MVC).

2.6.3. LA TERCERA VERSIÓN (ACTUAL)

A pesar de ser de gran utilidad combinar la POO con la arquitectura MVC. Puede resultar complicado introducir la idea a usuarios principantes de acuerdo al objetivo de este trabajo, por lo que para la primera versión de la plataforma (tercera del algoritmo) decidimos presentar una arquitectura híbrida (figura 2.14) basada en la filosofía de MVC, pero utilizando varios controladores y limitando la obtención de datos desde la hoja de cálculo (esto con el fin de no involucrar de momento las conexiones a bases de datos). Así, tendremos más de un controlador o vista y utilizaremos variables de sesión de *PHP* para comunicar variables de un archivo a otro.

La idea de tener varios archivos es también para demostrar la utilidad de la POO y de que el "programador final" deberá escribir muy poco código para utilizar esta plataforma y crear módulos nuevos.

Iniciamos con una descripción del árbol de directorios.

Directorio "/" (raíz): es aquí donde se encuentran los archivos (vista y controladores): Index.html, subir.html, upload.php, leer.php, ejemplo.php, grafica.php.

Directorio "classes": aquí se localizan las clases (módulos): CurvaTipo.php, PHPExcel.php, excel_reader.php.

Directorio "archivos": aquí se suben los archivos de hoja de cálculo también donde se generan los archivos de resultados, para descargarse.

Directorio "style e images": Aquí se localizan archivos propios de la parte visual.

A continuación, se publica el código de cada uno de los controladores y vistas. En el capítulo 3, se expone el código comentado de cada uno de los módulos.

2.6.4. ACCESO A LA APLICACIÓN

Para probar el software, hay que acceder a www.atlahua.com/tesis y hacer clic en la liga de "Acceso" (si la dirección del sitio no está disponible, por favor reportarlo a: a: tesis@atlahua.com)

FIGURA 2.14 ACCESO A LA PLATAFORMA APANTLI DESDE: WWW.ATLAHUA.COM/TESIS

2.6.5. INGRESO DE DATOS

Como se ha mencionado previamente para esta versión, los datos deben ingresarse a través de un archivo de hoja de cálculo. Dicha hoja lleva un formato especial y riguroso (figura 2.15 Preparación de datos), de la siguiente forma:

De la celda 3 a la 7 de la columna B, se colocan los valores de gasto, factor de volumen del aceite, viscosidad, espesor y radio.

De la celda 3 hasta la celda que corresponda al número de datos empleados en la columna E y F, se colocan los pares de puntos de tiempo y delta de presión respectivamente.

En el sitio de Apantli, se encuentra un archivo de muestra con datos ordenados según la explicación.

Una vez que se tiene lista la hoja, utilice el botón de “Seleccionar archivo” o “Examinar” para seleccionar el archivo de datos desde su equipo.

Posteriormente, haga clic en “Subir archivo” para enviarlo al servidor

	A	B	C	D	E	F
1	Datos roca/fluido				T[hr]	Delta P
2						
3	Gasto	427	stb/d		1	3.44
4	Bo	1.12			1.5	2.91
5	Viscosidad	0.8	cp		2	4.46
6	Espesor	23	ft		3	9.36
7	Distancia entre pozos	427	ft		5	21.72
8					10	29.82
9					13	40.48
10					16	38.53
11					18	44.49
12					20	44.25

FIGURA 2.15 PREPARACIÓN DE DATOS

El código empleado en esta parte es el siguiente:

Nota: las líneas antecedidas por "//" (doble diagonal) o entre bloques como "/* ... */" (Diagonal, asterisco – asterisco, diagonal), son comentarios explicando el código que les precede.

acceso.html

```
<form action="upload.php" method="post" enctype="multipart/form-data">
                    <input name="archivo" type="file"  size="40" value="Seleccionar" />
                    <input name="enviar" type="submit" value="Subir archivo" />
                    <input name="action" type="hidden" value="upload" />
                </form>
```

El archivo "upload.php" contiene el siguiente código:

```
<?php
//Se declara el inicio de sesión para recibir las variables del archivo anterior
    session_start();
    //Declaración de error de la clase excel_reader2.php
    error_reporting(E_ALL ^ E_NOTICE);
      //Se incluye la clase excel_reader2.php, para poder leer los datos de la hoja de cálculo
    require_once 'Classes/excelreader/excel_reader2.php';
    //se declara un arreglo vacío para alojar los datos de tiempo vs presión
    $CR = array();
    //se declara un arreglo vacío para alojar los datos de las propiedades roca/fluidos
    $props = array();
     //se declara una variable de sesión con el nombre del archivo subido (el nombre de la prueba)
```

```
15    $hoja = $_SESSION['nombre_prueba'];
16    //Se declara una variable para vaciar el contenido de la hoja de cálculo
17    $data = new Spreadsheet_Excel_Reader('archivos/' . $hoja);  18
20    //se cuenta el número de filas contenidas en la hoja 1 del archivo subido
21    $filas = $data->rowcount($sheet_index = 0);
22    //se cuenta el número de columnas contenidas en la hoja 1 del archivo subido
23    $columnas = $data->colcount($sheet_index = 0);
24    //SE va guardando en el archivo para tiempo vs presión, los pares de puntos a partir de
la celda 3 de la
25    //columna E y F respectivamente
26    for ($i = 3; $i < ($filas + 1); $i++) {
27          //El empleo de $i-3, se debe nuevamente a la convención de que en PHP los
arreglos comienzan con el elemento 0 (cero)
28          //así con $CR[$i - 3][0]  y  $CR[$i - 3][1], como $i vale 3 al inicio del for,
obtenemos: $CR[0][0] y $CR[0][1]
29       // (que son el primer par de puntos)
30       $CR[$i - 3][0] = $data->val($i, 'E');
31       $CR[$i - 3][1] = $data->val($i, 'F');
32    }
33     //Se guardan los valores de roca/fluidos en un arreglo, para no tener que utilizar más
variables de sesión.
34     //Evidentemente las propiedades corresponden a las coordenadas de la celda en la
hoja de cálculo donde está el valor deseado
35    $props[qbls] = $data->val(3, 'B');
36    $props[Bo] = $data->val(4, 'B');
37    $props[mu] = $data->val(5, 'B');
38    $props[h] = $data->val(6, 'B');
39    $props[radio] = $data->val(7, 'B');
40    //se crean nuevas variables de sesión para almacenar los valores
41    $_SESSION['CR'] = $CR;
42    $_SESSION['props'] = $props;
43 ?>
44    <!--Escrito en HTML, está el enlace para continuar-->
45    <a href="ejemplo.php">Click aquí para continuar</a>
```

Después de subir el archivo, se muestra un menú con las opciones de:

- Suavizar los datos sin interpretar la prueba
- Interpretar la prueba sin suavizar los datos
- Suavizar los datos e interpretar la prueba

2.6.6. EJECUCIÓN DEL PROGRAMA

Tras seleccionar una opción, se llega al controlador (archivo) “ejemplo.php” donde se utiliza la clase CurvaTipo.php para resolver el método de Match Point.

Se muestra el código más completo, correspondiente al caso de suavizar los datos e interpretar la prueba:

```
<?php
session_start();
//Se incluye la clase de CurvaTipo
include('Classes/CurvaTipo.php');
include('Classes/PromedioMovil.php');
$CR = array();
//Se recibe el arreglo con los datos de la prueba y las propiedades roca/fluidos
$CR = $_SESSION['CR'];
$props = $_SESSION['props'];
$CT = array();
//Se ejecuta el módulo de suavizado, utilizando una ventana de 3 datos para el promedio móvil y el arreglo con los datos de la prueba como parámetro.
                $suavizado = new PromedioMovil(3,$CR);
//Se accede a los resultados por medio de la función get_suavizado();
                $CR = $suavizado->get_suavizado();
//Utilizando el nuevo arreglo, con los dato suavizados, se  ejecuta el módulo de CurvaTipo pasando el arreglo con las propiedades roca / fluido y los datos de la prueba como parámetro
                $prueba = new CurvaTipo($props, $CR);
                $CT = $prueba->get_ajustado();
                $CR2 = $prueba->get_original();
//Se accede al arreglo con los datos ajustados  por medio de la función get_ajustado (), al arreglo original, por medio de la función get_original() y se imprimen las variables de permeabilidad y el producto porosidad por compresiblidad usando las funciones respectivas
                echo '<br><b>Permeabilidad = ' . $prueba->get_perm() . ' [mD]<br>  Producto (phi * ct) = ' . $prueba->get_fict() . ' [psi^-1]</b>';
                $_SESSION["CR"] = $CR;
                $_SESSION["CT"] = $CT;
                $_SESSION["CR2"] = $CR2;

?>
```

2.6.7 Generación dinámica de resultados

Finalmente, el código para graficar y crear el archivo desargable es:

grafica.php (parte1)

```
<?php
   //Nuevamente se abre la sesión para recibir los arreglos que se van a graficary a guardar en la hoja de cálculo
    session_start();
// Se reciben los arreglos
    $hoja = $_SESSION['nombre_prueba'];
    $CT = $_SESSION["CT"];
    $CR2 = $_SESSION["CR2"];
//Se cuenta el número de elementos para definir el límite del ciclo for
    $numCT = count($CT);
//Se manda imprimir la declaración de archivos y el Script de "JavaScript" necesario para graficar con JQPLot
       echo '<script type="text/javascript" language="javascript"> td= new Array(); pd= new Array();  pr= new Array(); td2= new Array(); pd2= new Array();  pr2= new Array();</script>';
//Este es un artificio para mandar el arreglo de PHP a un arreglo en JavaScript
    for ($i = 0; $i < count($CT); $i++) {
         echo '<script type="text/javascript" language="javascript">td[' . $i . ']=' . $CT[$i][0] . ';</script>';
         echo '<script type="text/javascript" language="javascript">pr[' . $i . ']=' . $CT[$i][1] . ';</script>';
         echo '<script type="text/javascript" language="javascript">pd[' . $i . ']=' . $CT[$i][2] . ';</script>';
        echo '<script type="text/javascript" language="javascript">td2[' . $i . ']=' . $CR2[$i][0] . ';</script>';
        echo '<script type="text/javascript" language="javascript">pr2[' . $i . ']=' . $CR2[$i][1] . ';</script>';
       echo '<script type="text/javascript" language="javascript">pd2[' . $i . ']=' . $CR2[$i][2] . ';</script>';
    }
    ?>
    <!-- Ahora ingresamos el HTML-->
    <html>
       <head>
          <title>Tesis Atlahua y Marco</title>
             <!--Esta es una cabecera para indicar la codificación de caracteres, garantiza la impresión de letras con acentos y la ñ-->
          <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
          <!--Este es parte del script para guardar los datos en la hoja de Excel-->
          <link href="style/style.css" rel="stylesheet" type="text/css"><style>
             table.excel {
                border-style:ridge;
                border-width:1;
                border-collapse:collapse;
                font-family:sans-serif;
                font-size:12px;
             }
             table.excel thead th, table.excel tbody th {
                background:#CCCCCC;
                border-style:ridge;
                border-width:1;
                text-align: center;
                vertical-align:bottom;
```

```
 46              }
 47              table.excel tbody th {
 48                 text-align:center;
 49                 width:20px;
 50              }
 51              table.excel tbody td {
 52                 vertical-align:bottom;
 53              }
 54              table.excel tbody td {
 55                 padding: 0 3px;
 56                 border: 1px solid #EEEEEE;
 57              }
 58           </style>
 59           <!-- Aquí se incluye a la librería de jquery para graficar -->
 60                        <script language="javascript" type="text/javascript" src="dist/jquery-
1.4.2.min.js"></script>
 61           <!-- Aquí se incluye a la librería de jqplot para graficar -->
 62                                    <script language="javascript" type="text/javascript"
src="dist/jquery.jqplot.min.js"></script>
 63                                    <script language="javascript" type="text/javascript"
src="dist/plugins/jqplot.logAxisRenderer.min.js"></script>
 64           <!-- Agregar el log plug in, responsable de la escala log-log en los ejes -->
 65                                                <script type="text/javascript"
src="dist/plugins/jqplot.logAxisRenderer.min.js"></script>
 66           <!-- Aquí se incluye el archivo CSS (estilos) para graficar -->
 67           <link rel="stylesheet" type="text/css" href="dist/jquery.jqplot.css" />
68          <!--Este es en realidad la continuación del script que transporta el arreglo de PHP a
JavaScript-->
 69           <script type="text/javascript" language="javascript">
 70              //Se declaran los arreglos que serán la fuente del gráfico
 71              line1 =new Array();
 72              line1[0]=new Array();
 73              line1[1]=new Array();
 74              line2 =new Array();
 75              line2[0]=new Array();
 76              line2[1]=new Array();
 77              line3 =new Array();
 78              line3[0]=new Array();
 79              line3[1]=new Array();
 80              line4 =new Array();
 81              line4[0]=new Array();
 82              line4[1]=new Array();
 83              //se pasa el valor de $numCT a JavaScript
 84              numCR=<?PHP echo $numCT; ?>;
 85              //La ultima parte del artificio para pasar el arreglo de PHP a JavaScript en
formato "bidimensional"
 86              //(par de puntos) que requiere JavaScript para hacer la gráfica
 87              for(j=0;j<numCR;j++){
 88                 line2.push([td[j],pr[j]]);
 89                 line1.push([td[j],pd[j]]);
 90                 line3.push([td2[j],pr2[j]]);
 91                 line4.push([td2[j],pd2[j]]);
 92              }
 93           </script>
 94
```

```
 95      </head>
<body>
 97          <div class="first-section">
 98              <!--se define una zona para contener el gráfico-->
 99              <div id="chartdiv" style="height:400px;width:800px; "></div>
100              <script id="source" language="javascript" type="text/javascript">
101                  //se manda llamar a la función de graficar con los parámetros de: la zona
objetivo y los arreglos a graficar
102                      $.jqplot('chartdiv', [line1,line2,line3,line4], {
103                          //Esto es código de formato
104                          legend:{show:true, location:'nw'},
105                          title:'Match Point',
106                          grid: {background:'#f3f3f3', gridLineColor:'#accf9b'},
107                          series:[
108                              {label:'Curva Tipo', showMarker:false, color:'#336699'},
109                              {label:'Curva Ajustada', markerOptions:{style:'circle'},color:'#ccff00'},
110                              {label:'Curva Original', markerOptions:{style:'circle'},color:'#ff9900'},
111                              {label:'Curva Tipo', showMarker:false, color:'#336699'}],
112                          axes:{
113                              //Aquí se utilizó un artificio que no está docuemtnado en JQPLOT para
114                              //visualizar correctamente la escala de ambos ejes la cual cociste en
establecer
115                              //de forma manual los ciclos en la escala log - log
116                              xaxis:{renderer:$.jqplot.LogAxisRenderer,ticks:[0.01,1,
10,100,1000],label:'TD'},
117                              yaxis:{renderer:$.jqplot.LogAxisRenderer,ticks:[0.01,1,
10,100,1000],label:'PD - PR'}
118                          }
119                      });
120              </script>
121              <?php
122              //Esta es la continuación del código para generar el archivo de excel y hacerlo
descargable
123              error_reporting(E_ALL);
124              /** Se manda llamar a la clase PHPExcel */
125              require_once 'Classes/PHPExcel.php';
126              /** Se manda llamar a la clase PHPExcel_IOFactory */
127              require_once 'Classes/PHPExcel/IOFactory.php';
128              // Se crea un nuevo objeto de PHPExcel
129              $objPHPExcel = new PHPExcel();
130             // Se añaden los datos utilizando el sistema de coordenadas comentado en el
archivo de leer.php
131              for ($i = 2; $i < count($CT) + 1; $i++) {
132                  $objPHPExcel->setActiveSheetIndex(0)
133                          ->setCellValue('A1', 'TD')
134                          ->setCellValue('B1', 'PR')
135                          ->setCellValue('C1', 'PD')
136                          ->setCellValue('A' . $i, $CT[$i - 2][0])
137                          ->setCellValue('B' . $i, $CT[$i - 2][1])
138                          ->setCellValue('C' . $i, $CT[$i - 2][2]);
139              }
```

2.7. NOTAS SOBRE EL DESARROLLO DE LA PLATAFORMA

Mucho del código que se omitió anteriormente es código para utilizar CSS (hojas de estilo) que pertenece a la parte visual de la interfaz. No es objetivo de este trabajo ahondar en el tema y por lo tanto invitamos a quienes desconozcan CSS a:

1.- Copiar y pegar el código respectivo, para concentrarse en la parte de la programación.

2.- Dedicar un par de horas al breve manual en línea que se cita en las referencias.

Nuevamente hacemos énfasis de que los elementos de un arreglo en *PHP* se empiezan a contar desde cero y no desde uno:

Arreglo "campos":

Elemento	0	1	2	3
Valor	Cantarell	Chicontepec	Maloob	Zaap

En *PHP* se lee un elemento como: campos[0]=Cantarell;

En el archivo upload.php se requiere profundizar en el manejo de los "tipos" de archivo a fin de evitar vulnerabilidades de seguridad. La restricción en el uso de los tipos: application/vnd.ms-excel o application/octet-stream puede ocasionar que algunos archivos no se identifiquen correctamente a pesar de tratarse de hojas de cálculo.

El uso de un prefijo en los archivos que se envían al servidor es solo una medida de no duplicar las pruebas y evitar resultados no deseados si un archivo que contiene datos diferentes lleva el mismo nombre en la máquina de diferentes usuarios que estén utilizando Apantli simultáneamente.

La carpeta "archivos" tiene permisos especiales para permitir la escritura de los archivos que se envían y los que se generan para descargarse. Si en el desarrollo de un módulo similar al de este trabajo se presentan problemas con el envío o generación de archivos del lado del servidor, la razón puede ser que la carpeta objetivo no cuenta con los permisos debidos.

Para los comandos que se señalan como parte de la clase *PHP*reader y *PHP*Excel, hacemos la misma sugerencia en el sentido de copiar y pegar el código que se indica como parte de estas clases, a menos que el usuario tenga interés en comprender cómo funcionan dichas clases.

En la parte gráfica, se presentó el problema de enviar los valores de un arreglo en *PHP* a un arreglo en JavaScript.

A la fecha, en JavaScript no existe la figura de arreglo bidimensional como se maneja en *PHP* u otros lenguajes a donde se puede acceder a un elemento con "coordenadas" de los arreglos.

Algunos programadores han resuelto esta situación, haciendo "arreglos" de "arreglos" en JavaScript, pero para efecto de utilizar la librería de JqPlot, resolvimos la situación primero haciendo vectores unidimensionales en JavaScript a partir de cada columna de los vectores multidimensionales de *PHP*. Posteriormente utilizamos el comando "Push" para ir formando arreglos en JavaScript de la forma "pares de puntos". Como se muestra en el siguiente ejemplo:

```
arrgeloPHP=[[a,1],[b,2],[c,3]];
arregloJS1=[[a],[b],[c]];
arregloJS2=[[1],[2],[3]];
para: j desde cero hasta 2
arregloJSfinal=push([arregloJS1[j],arregloJS2[j]]);
arregloJSfinal=[[a,1],[b,2],[c,3]];
```

Es así como finalmente el arreglo en JavaScript "arregloJSfinal" adquiere la forma necesaria para utilizarse con JqPlot.

La gráfica que aparece actualmente contiene la curva original y la curva ajustada, cada una con su respectiva curva tipo.

Por otra parte, el archivo de Excel contiene solo la curva ajustada con la curva tipo.

Capítulo 3 EJEMPLO DE DESARROLLO DE UN MÓDULO DE APLICACIÓN

3.1. INTRODUCCIÓN

Apantli es una plataforma que busca la integración de programas de diferentes áreas de la Ingeniería Petrolera que puedan ayudar a resolver los problemas concernientes en el estudio y ejercicio de la profesión. Ya sea en:

- Ingeniería yacimientos.
- Ingeniería de perforación, terminación y mantenimiento de pozos.
- Producción y su manejo en superficie.
- Comercialización.
- Planeación, administración y evaluación de proyectos.
- Algún otra área relacionada con la Industria Petrolera.

Para la primera versión de Apantli trabajamos en la parte de la Caracterización Dinámica de Yacimientos. Enfocándonos en las pruebas de interferencia.

CARACTERIZACIÓN DINÁMICA DE YACIMIENTOS

Con el objetivo de predecir diferentes escenarios de producción, es necesario conocer información confiable del comportamiento del yacimiento [18].

La Caracterización Dinámica es un procedimiento mediante el cual se identifican y evalúan los elementos que afectan la explotación de un yacimiento a través del

análisis de variables que indican el comportamiento del sistema roca-fluidos como son:

- Presión
- Temperatura
- Flujo de fluidos

Al identificar estos elementos, también podemos cotejar y ajustar la información geológica y geofísica existente, para identificar:

- Presencia y ubicación de fallas geológicas.
- Acuñamientos
- Estratificación
- Discordancias
- Doble porosidad
- Doble permeabilidad

La Caracterización Dinámica se apoya en:

- Datos de producción de los fluidos del sistema (agua, aceite y gas).
- Registros de presión de fondo fluyendo y cerrado.
- Registros de molinete hidráulico y de temperatura
- Pruebas de trazadores.
- Pruebas de variación de presión.

3.2. TEORÍA SOBRE LAS PRUEBAS DE PRESIÓN

La Caracterización Dinámica de Yacimientos se basa principalmente en las pruebas de variación de presión debido a que son más sencillas y rápidas de tomar e interpretar (en comparación con otras técnicas como el análisis de producción) [19]. A través de ellas se pueden evaluar diferentes propiedades [18][19][20] como:

- Presión inicial (Pi) y presión promedio del yacimiento en el área de drene (P*)
- Permeabilidad vertical y horizontal (k)
- Grado de daño durante la perforación y terminación del pozo (S)
- Índice de productividad
- Geometría del pozo
- Límites del yacimiento
- Efectividad y eficiencia de una estimulación
- Grado de conectividad entre pozos
- Estructuras geológicas (sistema de fracturas, estratos, etc.)

3.2.1. DEFINICIÓN DE UNA PRUEBA DE PRESIÓN

De acuerdo con Bourdet (1977), Durante una prueba de pozo, una respuesta de presión es creada por el cambio temporal del gasto de producción (figura 3.1 respuesta de presión es creada por el cambio temporal del gasto de producción). Esto incluye la condición de que el gasto pueda ser cero y que dichas modificaciones puedan ser casi inmediatas o graduales.

¿Cuánto tiempo toma hacer una prueba de presión?
La respuesta del pozo es generalmente observada por un tiempo corto en comparación con la vida del yacimiento. EL tiempo que dura una prueba depende del objetivo de la misma.

Para tal motivo, debe haber una planeación cuidadosa que garantice proporcionar datos suficientes para hacer la interpretación de la prueba, según los objetivos de la misma.

Economides, (1994); Establece que desde la perforación hasta los tratamientos a los pozos, tienen importancia para tomar diferentes tipos de pruebas.

Se hace énfasis en la conveniencia de tomar una prueba de presión y de PVT, al término de la perforación, para aprovechar las condiciones de bajo saturación, tanto para hacer una buena caracterización del fluido como para tener el fluido en una sola fase durante la realización de la prueba.
Las pruebas de presión se le pueden aplicar a pozos del tipo: exploratorios, delimitadores y de desarrollo. [20]

Earlougher, (1977); establece que algunos impedimentos para obtener beneficios de una prueba de presión son:

- Datos insuficientes

Un claro ejemplo de esta condición es cuando se propone determinar la existencia y ubicación de una falla. Para este tipo de pruebas comúnmente se requerirá de más tiempo de medición para alcanzar a detectar la posible presencia de la falla que actúa como frontera.

- Selección del método de análisis inadecuado

Cuando el ingeniero no toma en cuenta si las consideraciones de un modelo, permiten hacer una buena representación del fenómeno físico que estudia.

- Incapacidad de integrar a la interpretación, información importante disponible

Cualquier otra información de estudios y mediciones que afecten el comportamiento de la prueba.

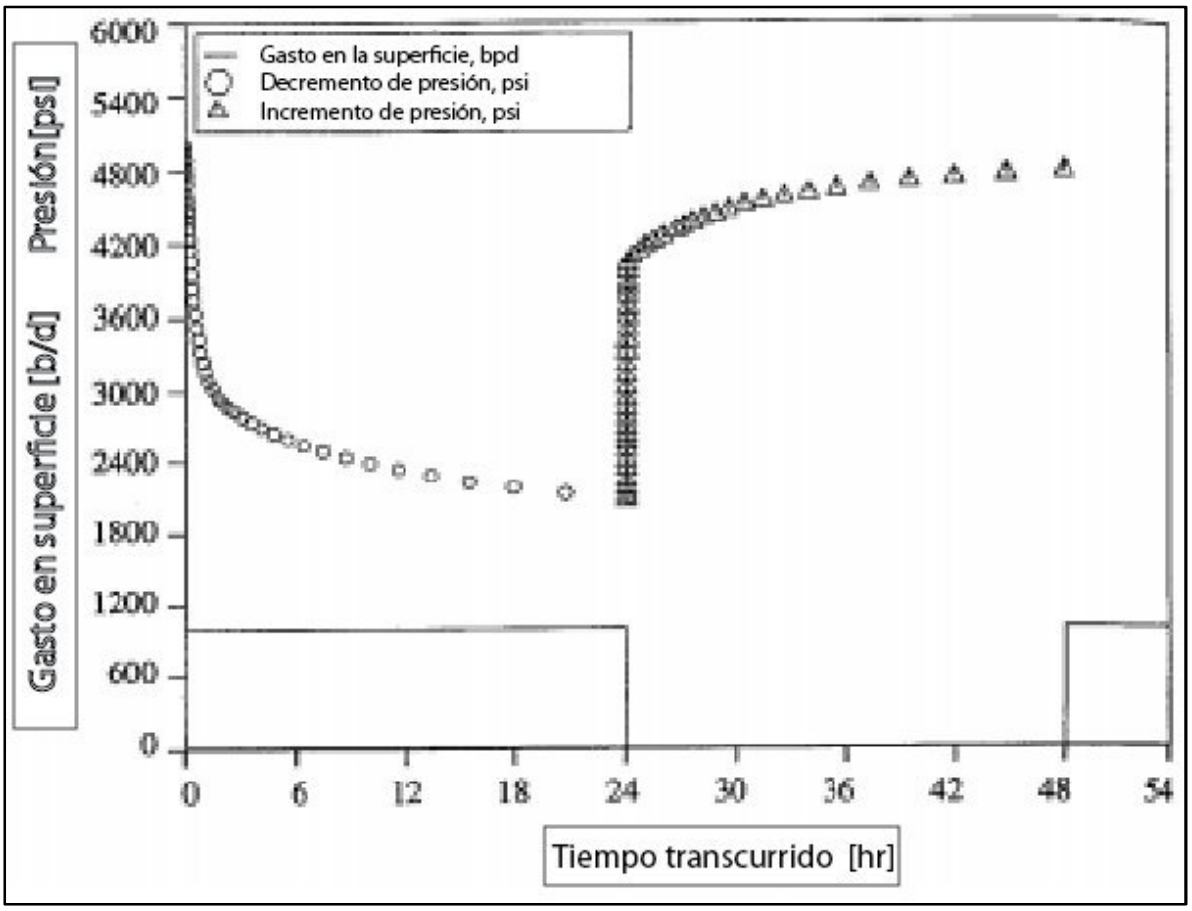


FIGURA 3.1 RESPUESTA DE PRESIÓN ES CREADA POR EL CAMBIO TEMPORAL DEL GASTO DE PRODUCCIÓN

En ocasiones, a pesar de que la prueba tenga una duración adecuada, se presentan otros factores que impiden a la herramienta registrar las mediciones correctamente. Tal es el caso del efecto de almacenamiento y descarga.

Brevemente comentamos que dicho fenómeno es consecuencia de la compresibilidad del aceite. Cuando el pozo es cerrado en superficie, la formación no deja de aportar fluido instantáneamente y tendrá que pasar un

lapso de tiempo hasta que la presión en la cabeza del pozo cerrado, se estabilice con la presión en la cara del pozo.

De manera análoga, cuando el pozo se abre nuevamente en superficie, comenzará el fenómeno de descarga caracterizado por que la producción del pozo es aportada por el aceite que se almacenó y comprimió en el pozo.

Y no será hasta un tiempo después que se genere nuevamente un desequilibrio de presión que permita a la formación volver a suministrar fluidos al pozo.

En resumen, el coeficiente de almacenamiento (o descarga) puede ser calculado como:

$$C = V[bl] * c[psi^{-1}]$$

ECUACIÓN 3-1

Donde ***V*** es el volumen del fluido en una sola fase en el fondo del pozo (en barriles o pies cúbicos) y ***c*** es la compresibilidad del fluido en el fondo del pozo a esas condiciones de presión y temperatura [18].

El cambio en el gasto que aporta la formación tiene un comportamiento diferente al de la superficie, debido al efecto de almacenamiento (figura 3.2 Efecto del almacenamiento. Gasto en la superficie y de la formación al pozo.).

El almacenamiento y daño se manifiestan en la figura 3.3 Efectos de almacenamiento y daño. Esto es de relevante importancia porque aquí se debe aplicar criterio y experiencia a la hora de validar los datos de la prueba y de hacer el análisis. Si bien a primera versión del módulo de pruebas de interferencia de Apantli, se limitó a interpretar los datos tal cual eran suministrados por el usuario, una mejora significativa fue la incorporación del módulo de suavizado de datos que permite corregir perturbaciones en la lectura de la herramienta. No obstante en ocasiones será necesario hacer un ajuste manual de los datos de entrada para conseguir una interpretación más confiable.

Durante el periodo de tiempo que toma el almacenamiento, la caída de presión en la cara de la formación es menor que en un pozo que no presenta efecto de almacenamiento. Cuando la influencia del pozo activo alcanza al pozo de observación, las lecturas de presión en tiempos breves después de iniciada la prueba, podrían no seguir el comportamiento de la curva exponencial integral.

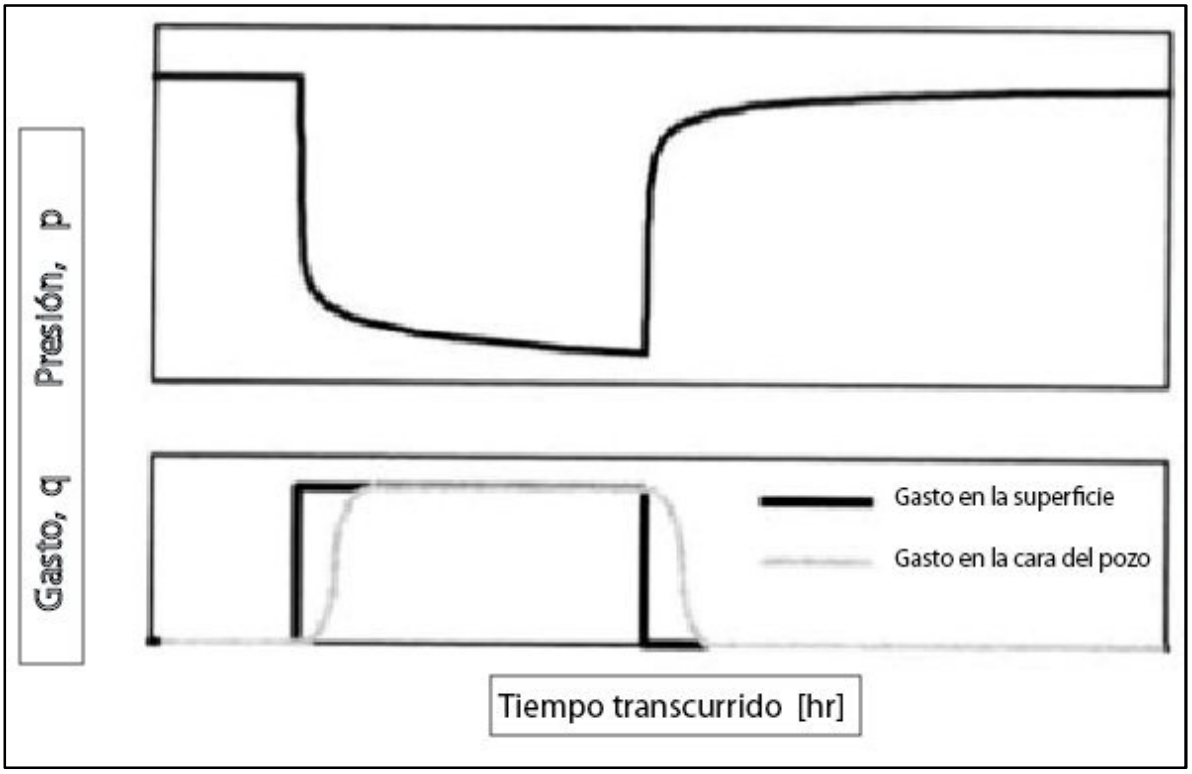


FIGURA 3.2 EFECTO DEL ALMACENAMIENTO. GASTO EN LA SUPERFICIE Y DE LA FORMACIÓN AL POZO.

Para interpretar pruebas bajo esas condiciones, Ramey [22] propuso un análisis utilizando otras curvas tipo.

Para mermar dichos efectos, diferentes autores sugieren utilizar una herramienta de fondo para abrir y cerrar el pozo a una profundidad más cercana a la zona productora.

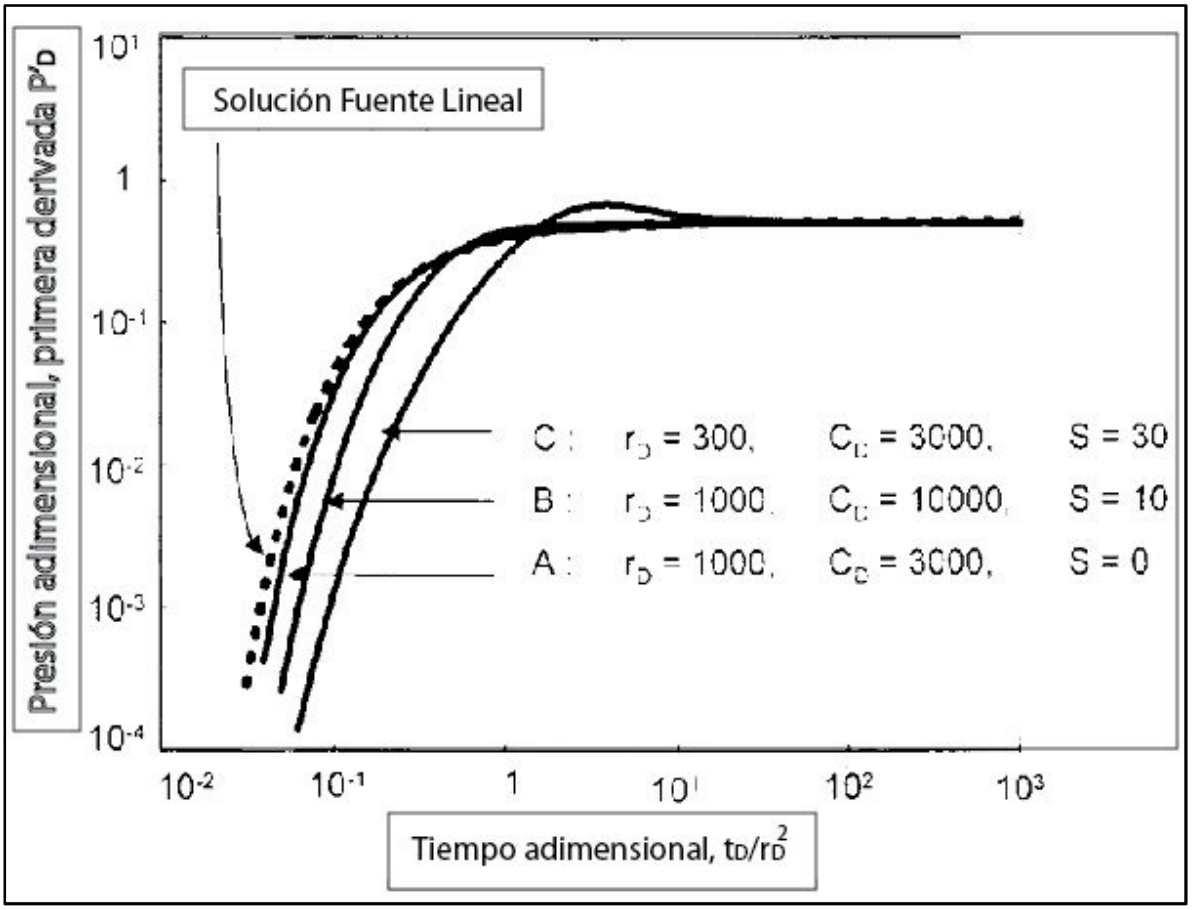


FIGURA 3.3 EFECTOS DE ALMACENAMIENTO Y DAÑO

3.2.2. CLASIFICACIÓN DE LAS PRUEBAS DE PRESIÓN:

De acuerdo al número de pozos involucrados en la prueba tenemos

- Un pozo
- Varios pozos (multipozo):

De acuerdo al tipo de pozo

- Productor
- Inyector

Haciendo combinaciones entre estas 4 principales características y dependiendo de otros factores como: el objetivo del estudio, las herramientas, condiciones, duración, requerimientos de hardware [20], concebimos una clasificación abreviada (figura 3.4)

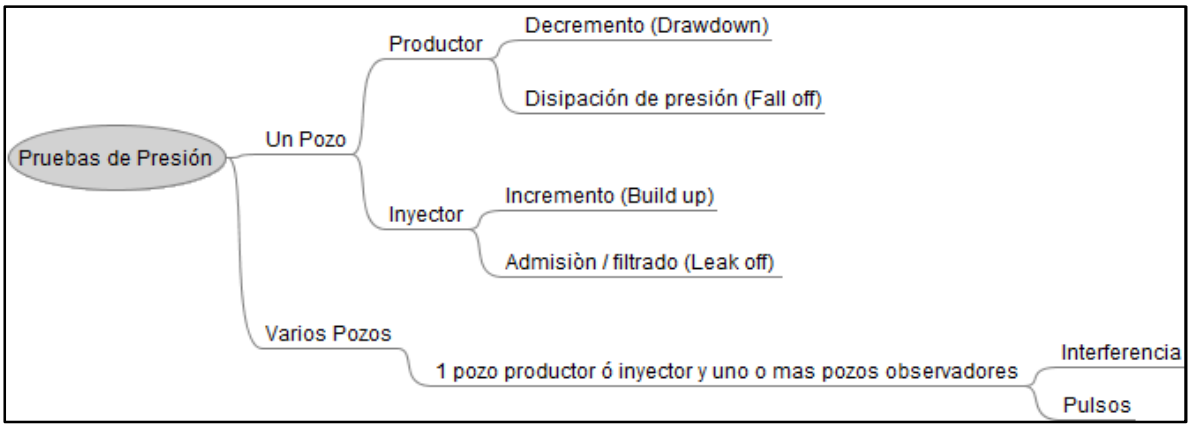


FIGURA 3.4 CLASIFICACIÓN ABREVIADA DE LAS PRUEBAS DE PRESIÓN

- **Prueba de decremento:** Previo a la prueba, se cierra el pozo por un tiempo suficiente para que la presión en el área de drene se estabilice y se aproxime a la presión inicial. Posteriormente se abre a producción a un gasto constante, lo que en la práctica resulta difícil, por lo que frecuentemente, es una prueba poco precisa. [20]

- **Prueba de incremento:** Después de producir por cierto tiempo que permita alcanzar un gasto estabilizado, el pozo se cierra para observar cómo se recupera la presión. A diferencia de la prueba de decremento, esta tiene la característica de que el gasto al ser cero, se encuentra perfectamente controlado [20].

- **Prueba de disipación de presión (fall off):** Se inyecta un fluido por cierto tiempo y se cierra el pozo, para medir la disipación de la presión de inyección. [20]

- **Prueba de admisión / filtrado (leak off)**: Se incrementa gradualmente la presión de inyección, para determinar a qué presión, la formación comienza a admitir fluido.

- **Prueba de interferencia:** Un pozo activo (inyector o productor), genera una variación de presión de fondo que es medida desde un pozo cerrado llamado observador. Esto proporciona información sobre la conectividad entre el pozo activo y el pozo observador [20] La prueba se puede realizar con mínimo un pozo activo y con uno o varios pozos observadores. Esto último para una mejor estimación del grado de homogeneidad y la dirección en que varían las propiedades del yacimiento. En la figura 3.5, se aprecia la respuesta del pozo activo y del pozo observador en una prueba de incremento de presión. La diferencia es evidente debido a que cuando el pozo activo se cierra, la presión comienza a recuperarse casi de inmediato. No así en el pozo observador que de acuerdo con la gráfica, empieza a manifestar diferencias de presión 7 horas

después del cierre del pozo.

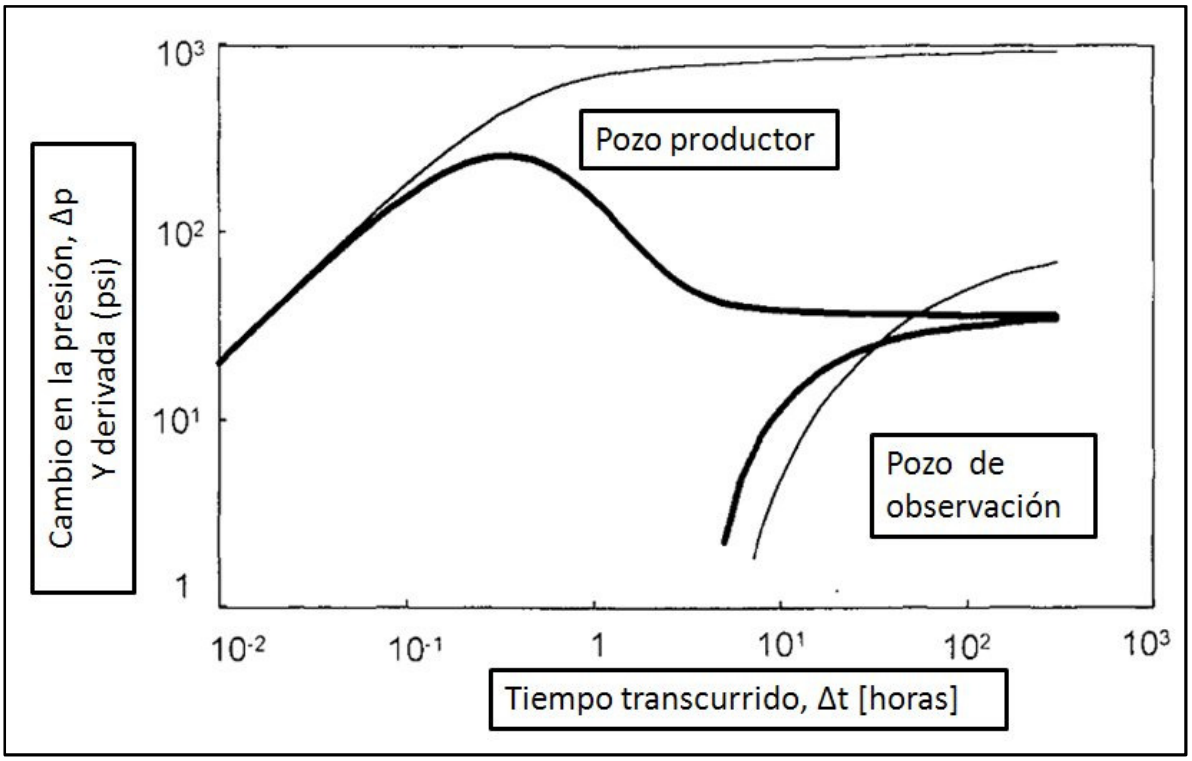


FIGURA 3.5 EFECTOS DE ALMACENAMIENTO Y DAÑO

3.2.3. REALIZACIÓN OPERACIONAL DE LA PRUEBA.

Para realizar el muestreo de la presión se realizan los siguientes pasos:

- La herramienta se baja con empacadores para aislar el intervalo a probar.
- Durante el descenso se mide la presión hidrostática hasta llegar a la profundidad deseada.
- Al llegar a la profundidad deseada se asientan los empacadores aislando la formación objetivo, del resto del pozo.
- Se abre una válvula y se permite el ingreso de fluido a presión atmosférica (si no existe colchón de agua) a la herramienta. En esta etapa se consigue un gasto estabilizado y una muestra del fluido
- Se cierra la válvula y comienza la recuperación de presión
- Se repiten los dos puntos anteriores y se registra la presión en función del tiempo

3.2.4. INTERPRETACIÓN DE UNA PRUEBA DE INTERFERENCIA POR EL MÉTODO DEL MATCH POINT

Queda fuera del alcance de este trabajo hacer una demostración exhaustiva del desarrollo matemático que se utiliza en la interpretación de las pruebas de presión, por lo que solo citaremos los aspectos más relevantes. Para ahondar en este tema, recomendamos revisar la bibliografía citada.

FUNDAMENTOS MATEMÁTICOS

La interpretación de las pruebas de presión utiliza como principio la ecuación de difusión expresada en coordenadas cilíndricas,

$\frac{\partial^2 p}{\partial r^2} + \frac{1}{r}\frac{\partial p}{\partial r} = \frac{1}{0.0002637}\frac{\emptyset \mu c_t}{k}\frac{\partial p}{\partial t}$ que lleva implícita las siguientes consideraciones [21]:

ECUACIÓN 3-2

- Flujo Radial
- Porosidad constante
- Medio isotrópico
- Condiciones isotérmicas
- Efectos gravitacionales despreciables
- Fluidos de compresibilidad pequeña y constante
- Gradientes de presión pequeños $c(\frac{\partial p}{\partial r})^2$
- Medio saturado al 100% por un solo fluido
- Yacimiento actuando con comportamiento infinito

Esta ecuación está expresada en variables dimensionales, por lo que resulta poco práctico, observar el comportamiento de la presión en función de cada una de las variables involucradas.

Por tal motivo, se agruparán ciertas variables para formar a partir de esos grupos, unas nuevas variables adimensionales. Estas variables son proporcionales a las variables reales.

La adimesionalización de las variables de hace en función de la geometría de flujo y del tipo de fluido, por lo que respetando la consideración de flujo radial y fluido de compresibilidad pequeña y constante, las nuevas variables adimensionales son:

Para la presión: $P_D = \frac{khΔP}{\alpha q_o B_o \mu_o}$ a=141,2

ECUACIÓN 3-3

Para el tiempo: $t_D = \frac{ßkt}{\phi \mu_o c_t r_w^2}$ $ß = 2.637\text{x}10^{-4}$

ECUACIÓN 3-4

Para la distancia: $r_D = \frac{r}{r_w}$

ECUACIÓN 3-5

Existen diferentes soluciones a la ecuación de difusión. Dependiendo de las condiciones iniciales y de frontera empleadas para resolverla [22].

En este trabajo presentamos la solución para flujo radial transitorio, gasto de producción constante, considerando que el pozo produce como una línea concéntrica, sin considerar daño y almacenamiento. Por lo que establecemos las siguientes condiciones iniciales y de frontera:

Condición inicial: $P_D(r_D, t_D) = 0; t_D \geq 0$

ECUACIÓN 3-6

Condición de frontera interna: $r_D \frac{\partial P_D}{\partial r_D} = -1$, pozo que produce a gasto constante.

ECUACIÓN 3-7

Condición de frontera externa: $\lim_{r_D \to \infty}(r_D, t_D) = 0$, yacimiento actuando como infinito (sin efectos de frontera).

ECUACIÓN 3-8

Utilizando además la transformada de Laplace y las funciones Bessel, podemos obtener la solución fuente lineal:

$P_D = -\frac{1}{2} E_i(\frac{-r_D^2}{4t_D})$ Donde E_i es la función Integral Exponencial.

ECUACIÓN 3-9

Con esta expresión, se puede construir la curva tipo.

1.- Suponer valores para $\frac{t_D}{{r_D}^2}$

2.- Calcular $x = \frac{{r_D}^2}{4t_D}$

ECUACIÓN 3-10

3.- Evaluar $E_i(x)$mediante uno de los siguientes polinomios:

Para $0 \leq x \leq 1$ $$E_i = -0.57721566 + 0.99999193x + -0.24991055x^2 + 0.05519968 x^3 \pm 0.00976004^4 + 0.00107857x^5 - \ln(x)$$

ECUACIÓN 3-11

Para $1 \leq x \leq \infty$: $E_i = \frac{0.2677737343+8.6347608925x+18.0590169730x^2+8.5733287401x^3+x^4}{3.9584969228+21.0996530827x+25.6329561486x^2+9.5733223454x^3+x^4} \left(\frac{1}{xe^x}\right)$

ECUACIÓN 3-12

4.- Graficar en papel (log - log) $P_D vs \frac{t_D}{r_D^2}$

El método gráfico del Match Point consiste en:

5.- Graficar en papel transparente (log - log) los datos de la prueba, encimar las gráficas y desplazar vertical y horizontalmente la curva de la prueba hasta que coincida con la curva tipo.

6.- Se puede trazar una línea sobre la curva para indicar el grado de ajuste que se alcanzó.

7.- Se selecciona el punto de ajuste, contenido en el área bajo las dos curvas de modo que a partir de ese punto se tomen lecturas a los ejes correspondientes de cada curva.

8.- Se toman las lecturas en la curva real de: (p vs t) y en la cuerva tipo de (pd vs $\frac{td}{r_D^2}$)

9.- Se emplean las siguientes expresiones para calcular las propiedades de interés:

$$kh = \frac{\alpha q B \mu (pD)_M}{(\Delta p)_M}$$

ECUACIÓN 3-13

$$\phi c_t = \frac{\beta k(t)_M}{\mu r^2 \left[\frac{t_D}{r^2{}_D}\right]_M}$$

ECUACIÓN 3-14

En donde el subíndice M indica que esos datos fueron tomados del punto de ajuste. pD es la presión adimensional, Δp es la caída de presión, (t)M es el tiempo de la prueba, tD/rD2 el tiempo dividido por la distancia radial al cuadrado y α y ß son factores de conversión de unidades. Los cálculos efectuados en las ecuaciones de kh y ΦCt permiten determinar las propiedades de la roca en el yacimiento correspondiente a la zona indicada.

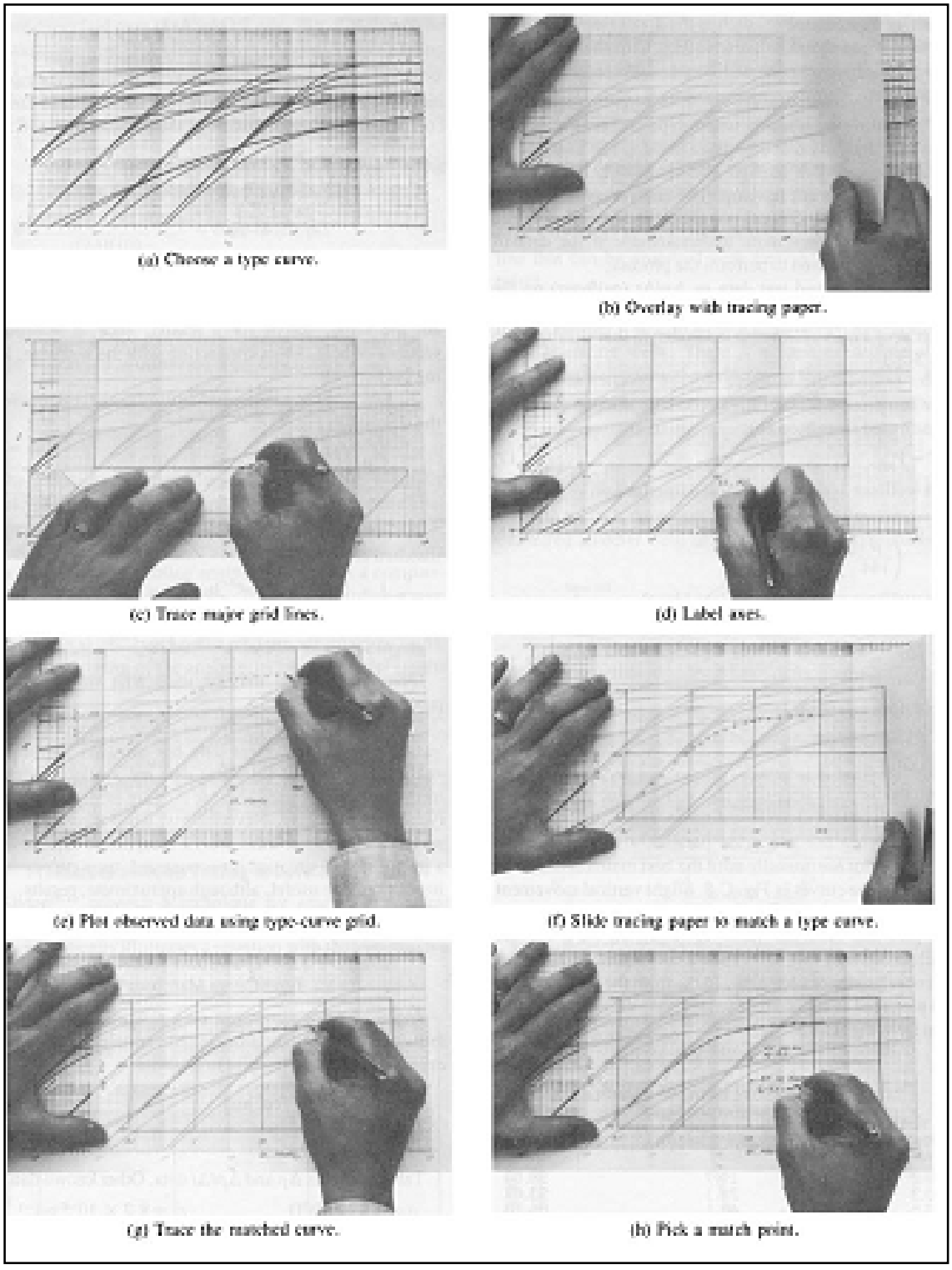


FIGURA 3.6 MÉTODO GRÁFICO DEL MATCH POINT

3.3. ALGORITMO NUMÉRICO DEL MATCH POINT

Utilizaremos parte de un ejemplo documentado en la referencia para explicar el funcionamiento del algoritmo.

Se realizó una prueba de interferencia entre el pozo "Apantli 217 y el Apantli 410" separados por 340 [ft]. Se expresaron las lecturas de tiempo y presión como:

TABLA 3-1 EJEMPLO DE UNA PRUEBA DE INTERFERENCIA

Tiempo [hrs]	Δ Presión (Pi - Pwf) [Psi]
1	2
1.5	5
2	7
3	12
5	21
10	33
18	41
24	49
36	58
50	68
90	75
120	81
150	86
180	89

Las propiedades roca / fluido son:

r=	340	[ft]	h=	23	[ft]
q=	427	[stb/d]	Ø=	0.12	fracc
μ=	0.8	[cp]	Ct=	8.30E-006	[Psi^{-1}]
Bo=	1.12				

Se considera el último par de puntos del arreglo que contiene los datos de tiempo vs delta P. En este caso sería: 180 hr vs 89 psi.

Utilizamos el valor de 180 hr como tiempo propuesto de "td": $\frac{td}{rd^2}=\frac{180}{1}$

Definimos $x=\frac{1}{4(\frac{rd^2}{td})}=\frac{1}{4*180}=0.001388889$

Como $x\leq 1$ se utiliza el polinomio (Ecuación 3-10) para calcular:

$Ei(x)=6.003423847$ Por lo que $Pd=\frac{1}{2}(Ei(x))=3.00171197388$

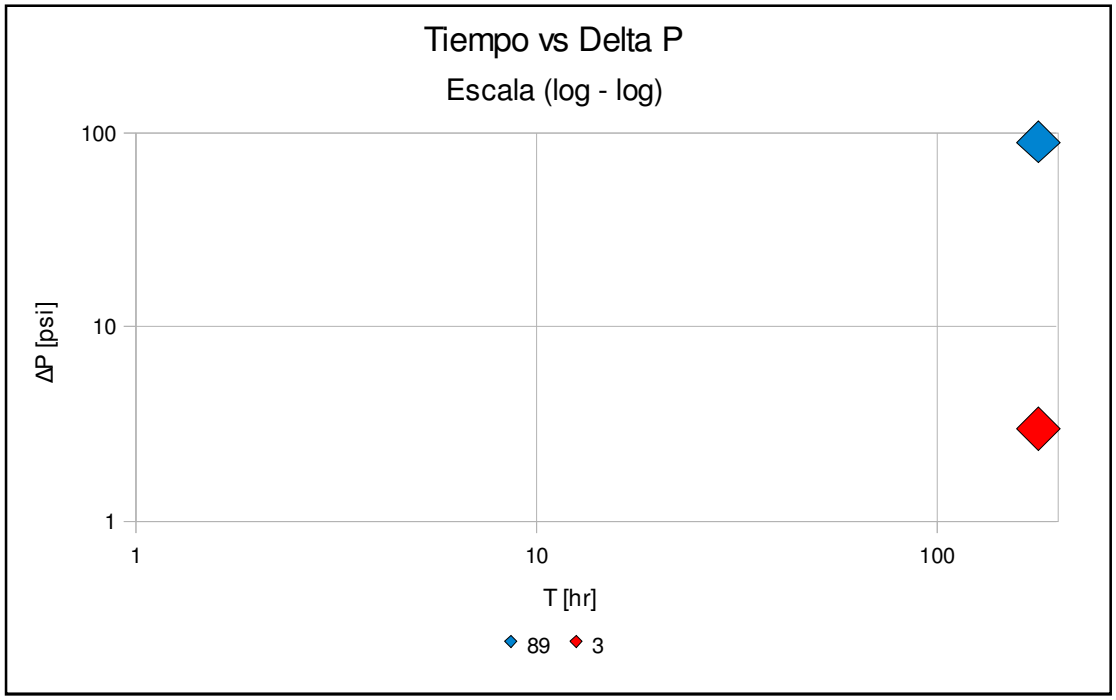


FIGURA 3.7 AJUSTE NUMERICO, PASO 1

Hasta este momento tenemos graficados un punto de la prueba real PR (89) y un valor de la Curva Tipo para flujo radial Pd (3.0017), correspondiente a un tiempo de (160 [hr]).

Si se calculan todas las Pd para cada uno de los tiempos, tendremos una gráfica como la siguiente:

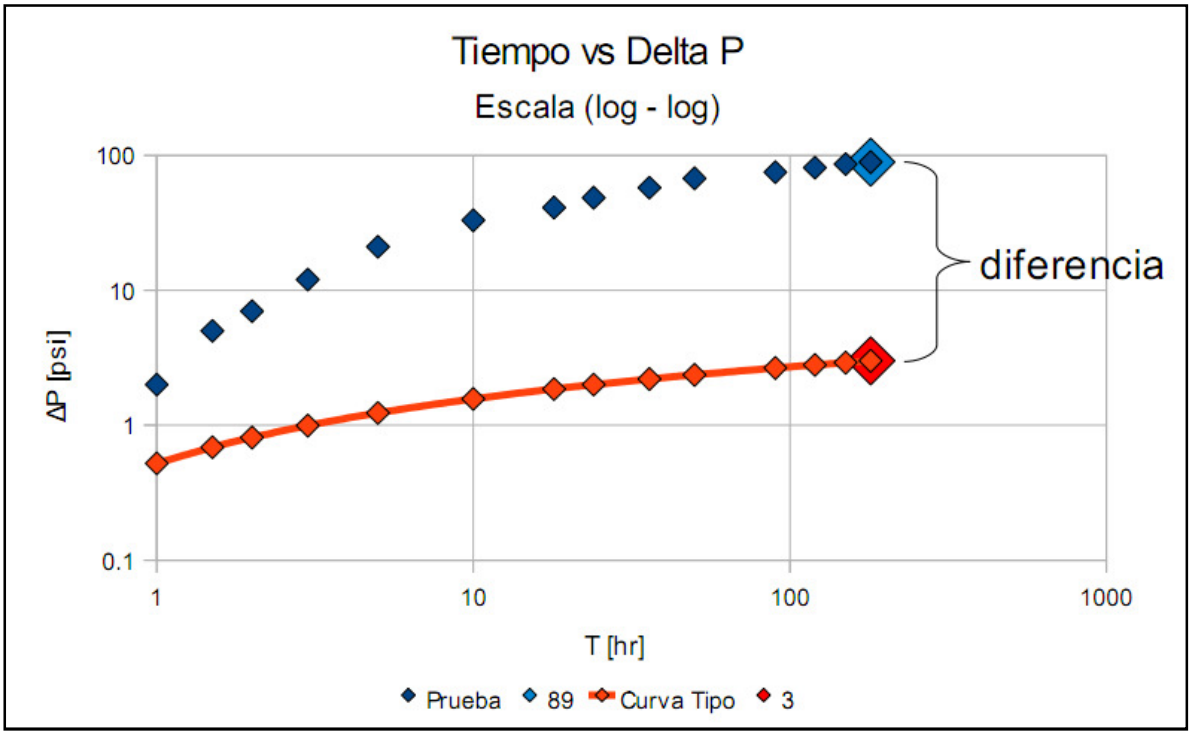


FIGURA 3.8 AJUSTE NUMÉRICO PASO 2

En la gráfica se puede apreciar que para 160 [hr], existe una diferencia entre la presión de la ΔP de la prueba (89 [psi]) y la Pd de la Curva Tipo (3.0017 [psi]).

¡Aquí es donde nuestro método de ajuste numérico comienza a actuar!

Lo primero es dejar en claro que esta gráfica está compuesta por datos en escala "normal", pero graficados en escala (log – log).

Por tal motivo, nos interesa es conocer esa distancia en escala (log – log) para lo cual utilizamos una variable llamada "diferencia" definida de la siguiente forma:

$$diferencia = \log b10(Pd) - logb10(P\,real) = \log b10(3.0017) - \log b10(89) = -1.472020989$$

ECUACIÓN 3-15

El signo con el que resulta la operación anterior nos dice mucho acerca de la situación del ajuste ya que por convención establecimos que un signo negativo será la forma de que el programa interprete que el punto de la curva real se encuentra por encima del punto calculado de la curva tipo.

Así pues, el siguiente paso es hacer coincidir el punto de la curva real (89 [psi]), con el punto de la curva tipo (3.0017 [psi]) referidos ambos al mismo tiempo (160 [hr]).

Esta operación se consigue multiplicando a 89 [psi] por un factor equivalente a la distancia en escala normal. De acuerdo al ejemplo será:

$$factor = 10^{diferencia} = 10^{-1.472020989} = 0.033727101$$

ECUACIÓN 3-16

De este modo ya podemos trasladar la presión de la curva real de la siguiente manera:

$$Presion\ real\ desplazada = Preal * factor = 89 * 0.033727101 = 3.001711974$$

ECUACIÓN 3-17

Lo cual demuestra que: $Preal\ desplzada = Pd\ (para\ td = 160)$ y por lo tanto se ha empatado correctamente ese punto.

Si multiplicamos por el factor a todos los valores de ΔP, gráficamente, el primer empate vertical queda de la siguiente forma:

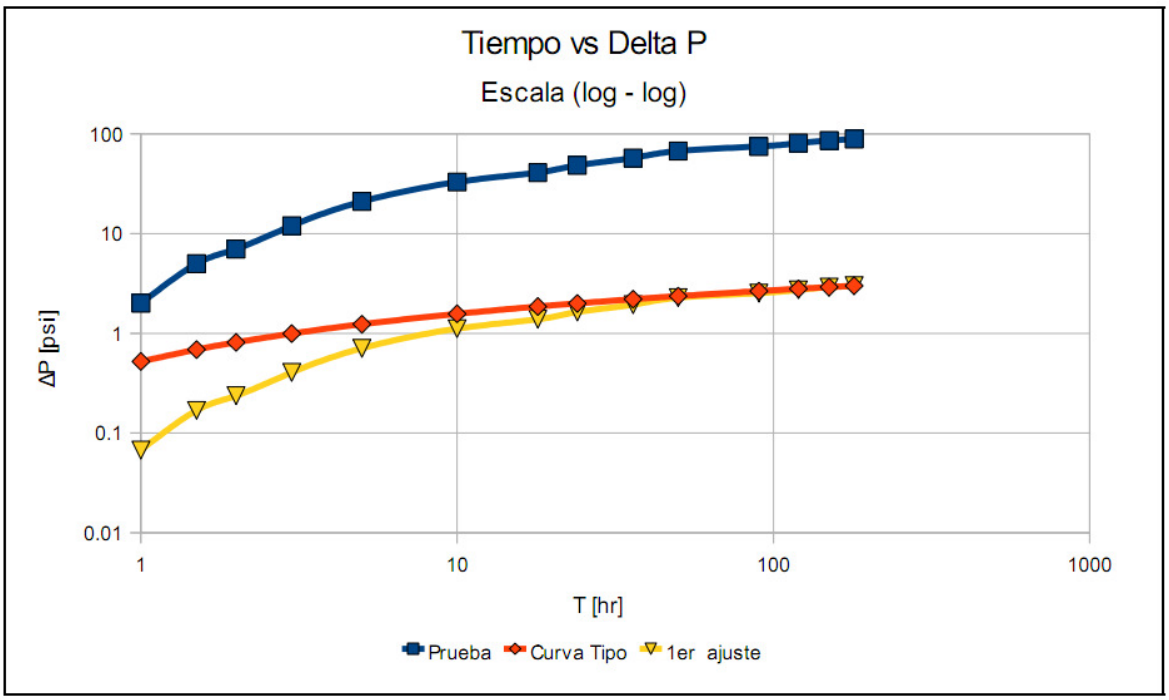


FIGURA 3.9 AJUSTE NUMÉRICO, PASO 3

Hasta el momento hemos estudiado las gráficas utilizando sólo los valores de la prueba (de (1 a 180 [hr] y de 2 a 89 [psi]).

La siguiente gráfica contiene la curva tipo para algunos valores de tiempo desde 0.01 hasta 180 [hr], la curva de la prueba original y la curva de la prueba desplazada con el primer empate vertical.

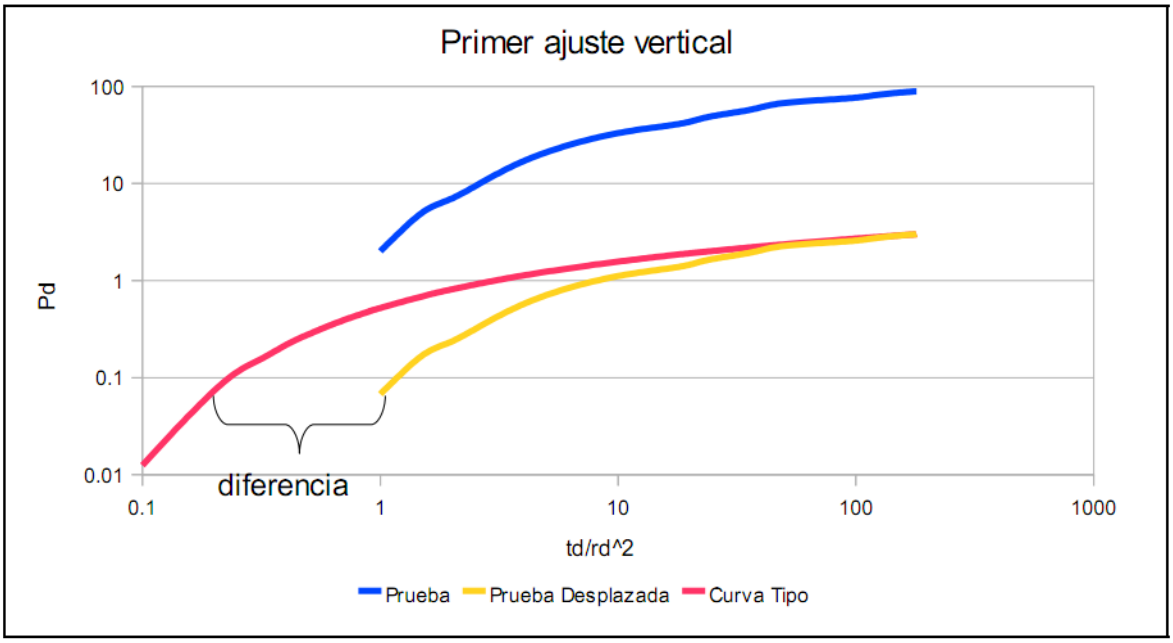


FIGURA 3.10 AJUSTE NUMÉRICO, PASO 4

La tabla de datos de esta gráfica es:

TABLA 3-2

td	Pd	Prueba	Prueba Desplazada	td	Pd	Prueba	Prueba Desplazada
0.01	2.67E-013			0.8	4.38E-001		
0.02	1.39E-007			0.9	4.82E-001		
0.03	1.30E-005			1	5.22E-001	2	0.07
0.04	1.35E-004			1.5	6.87E-001	5	0.17
0.05	5.74E-004			2	8.12E-001	7	0.24
0.06	1.55E-003			3	9.95E-001	12	0.4
0.07	3.19E-003			5	1.23E+000	21	0.71
0.08	5.57E-003			10	1.57E+000	33	1.11
0.09	8.67E-003			18	1.86E+000	41	1.38
0.1	1.25E-002			24	2.00E+000	48.5	1.64
0.2	7.32E-002			36	2.20E+000	57.5	1.94
0.3	1.46E-001			50	2.36E+000	67.5	2.28
0.4	2.16E-001			90	2.66E+000	75	2.53
0.5	2.80E-001			120	2.80E+000	81	2.73
0.6	3.38E-001			150	2.91E+000	86	2.9
0.7	3.90E-001			180	3.00E+000	89	3

En este caso no podemos seguir un procedimiento análogo simple, porque la máquina no tendría forma de saber qué valor de td proporcionar para obtener la

distancia en un solo paso, por lo que debemos entonces aproximar la solución, nuevamente utilizando solo un par de puntos que ahora serán los correspondientes al primer par (1 [hr], 2 [psi]).

La manera de que el programa pueda determinar que el punto (1 [hr], 2 [psi]) de la prueba ha si empatado horizontalmente con la curva tipo es utilizando una segunda variable de diferencia de tal modo que la primera variable de diferencia será en un plano vertical y la segunda en un plano horizontal (que más adelante se asociará como la variable "paso") como de muestra a continuación:

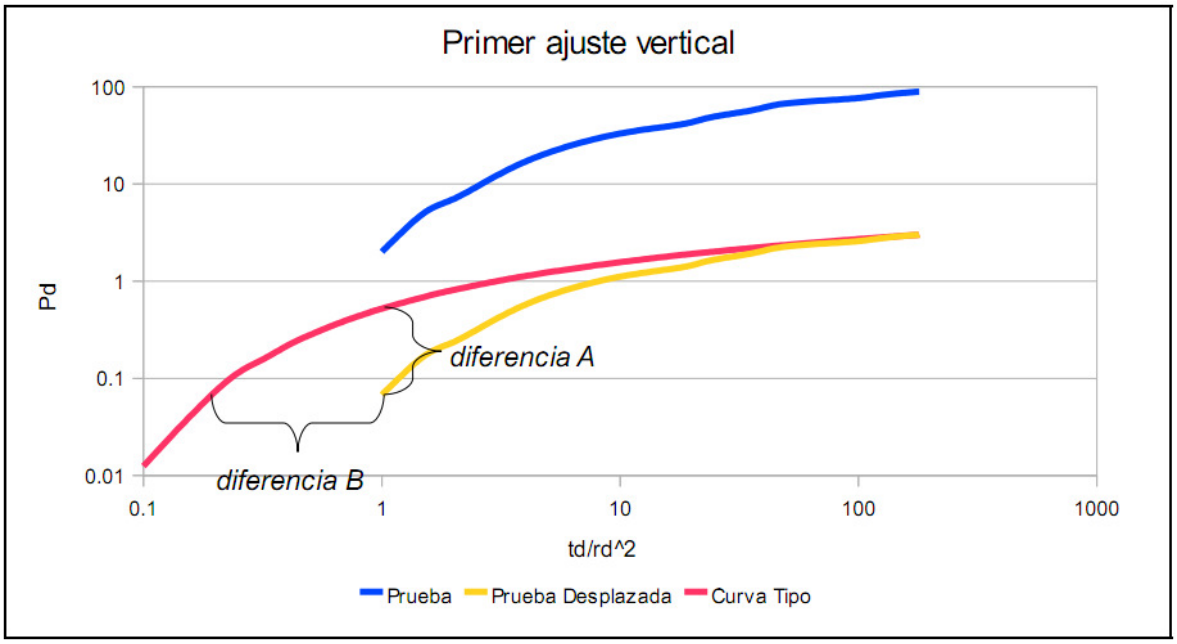


FIGURA 3.11 AJUSTE NUMÉRICO, PASO 5

Entonces vamos a proponer un valor inicial para aproximar horizontalmente el punto (1 [hr], 2 [psi]) a la curva tipo. No sin antes recordar que el punto (1 [hr], 2 [psi]) ha sido trasladado multiplicando por el factor que calculamos y ahora equivale a:

(1 , 0.0674542)

Enfatizamos que ya no estamos usando las unidades de tiempo y presión debido a que el ajuste se empatará con la Curva Tipo para flujo radial, que está expresada

en variables adimensionales. Esta es la misma razón por la cuál el eje de las gráficas cambia a Pd para el eje "y" a la vez que usaremos $\frac{td}{rd^2}$ para el eje "a".

Una vez aclarado el punto anterior, continuamos con el método verificando el valor de la distancia A.

$$diferencia = Pd - P(el\ primer\ dato)$$

ECUACIÓN 3-18

Pd en este momento se debe calcular con el tiempo correspondiente al par de datos propuesto, que recordamos es: (1 , 0.0674542).

Así que siguiendo el procedimiento descrito para el ajuste vertical, con un td = 1, resulta una Pd= 0.52214134869. Por lo que:

$$diferencia = 0.52214134869 - 0.0674542 = 0.45468714703$$

Nuevamente, el signo del resultado es de vital importancia ya que por convención establecimos que un signo positivo denota que la curva desplazada se encuentra a la derecha de la curva tipo tal y como se ha observado en las gráficas anteriores.

El objetivo es por tanto: proponer un nuevo valor de td menor al anterior usando la variable "paso". Como se puede intuir, "paso" está altamente relacionado con la precisión del ajuste ya que entre menor sea, más fina será la aproximación, pero también requerirá mayor tiempo de cómputo aproximarse a una solución suficientemente precisa.

nuevo td =td viejo−paso

ECUACIÓN 3-19

Debemos agregar que la fórmula anterior será siempre una resta ya que es una forma de garantizar que si "paso" fuera negativo y por tanto la curva desplazada estuviera a la izquierda de la curva tipo, por regla de los signos el valor del nuevo td se incrementaría y comenzaría a avanzar hacia la derecha.

Continuando con el ejemplo, aclaramos que el valor de “paso” es igual a 0.001 ya que demostró ser un valor que da una buena precisión al ajuste y en combinación con nuestro método, no consume mucho tiempo de cómputo.

nuevo td =1−0.001=.999

Es evidente que con el nuevo td no habrá un cambio significativo en el valor de diferencia, el cual deberá tender a cero, para poder asumir que se ha completado el ajuste horizontal.

Por lo que después de “n” aproximaciones de td, se encuentra que el valor que empata es td=0.194. Obteniendo el factor de manera análoga al desplazamiento vertical, calculamos que: factor =0.194

Ahora podemos multiplicar a cada uno de los valores de tiempo de la prueba original, para desplazarla horizontalmente como se muestra en la siguiente figura:

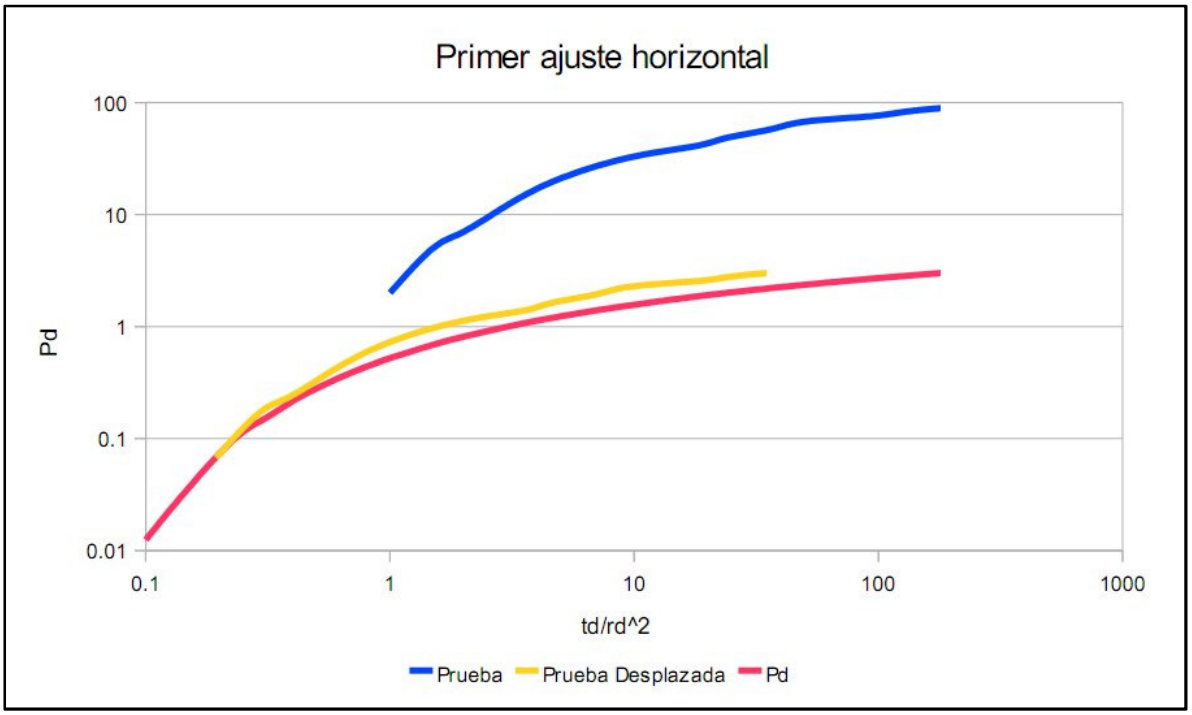


FIGURA 3.12 AJUSTE NUMÉRICO, PASO 6

La tabla de la gráfica anterior es:

TABLA 3-3

td	Pd	Prueba	td desplazado	Prueba Desplazada
0.01	2.67E-013			
0.02	1.39E-007			
0.03	1.30E-005			
0.04	1.35E-004			
0.05	5.74E-004			
0.06	1.55E-003			
0.07	3.19E-003			
0.08	5.57E-003			
0.09	8.67E-003			
0.1	1.25E-002			
0.2	7.32E-002			
0.3	1.46E-001			
0.4	2.16E-001			
0.5	2.80E-001			
0.6	3.38E-001			
0.7	3.90E-001			
0.8	4.38E-001			
0.9	4.82E-001			
1	5.22E-001	2	0.19	0.07
1.5	6.87E-001	5	0.29	0.17
2	8.12E-001	7	0.39	0.24
3	9.95E-001	12	0.58	0.4
5	1.23	21	0.97	0.71
10	1.57	33	1.94	1.11
18	1.86	41	3.49	1.38
24	2	48.5	4.66	1.64
36	2.2	57.5	6.98	1.94
50	2.36	67.5	9.7	2.28
90	2.66	75	17.46	2.53
120	2.8	81	23.28	2.73
150	2.91	86	29.1	2.9
180	3	89	34.92	3

Durante cada iteración de este proceso, existen un par de variables que registran los desplazamientos verticales y horizontales totales, cuya utilidad la comentaremos más adelante.

Habiendo realizado la quinta iteración, el ajuste de la curva es:

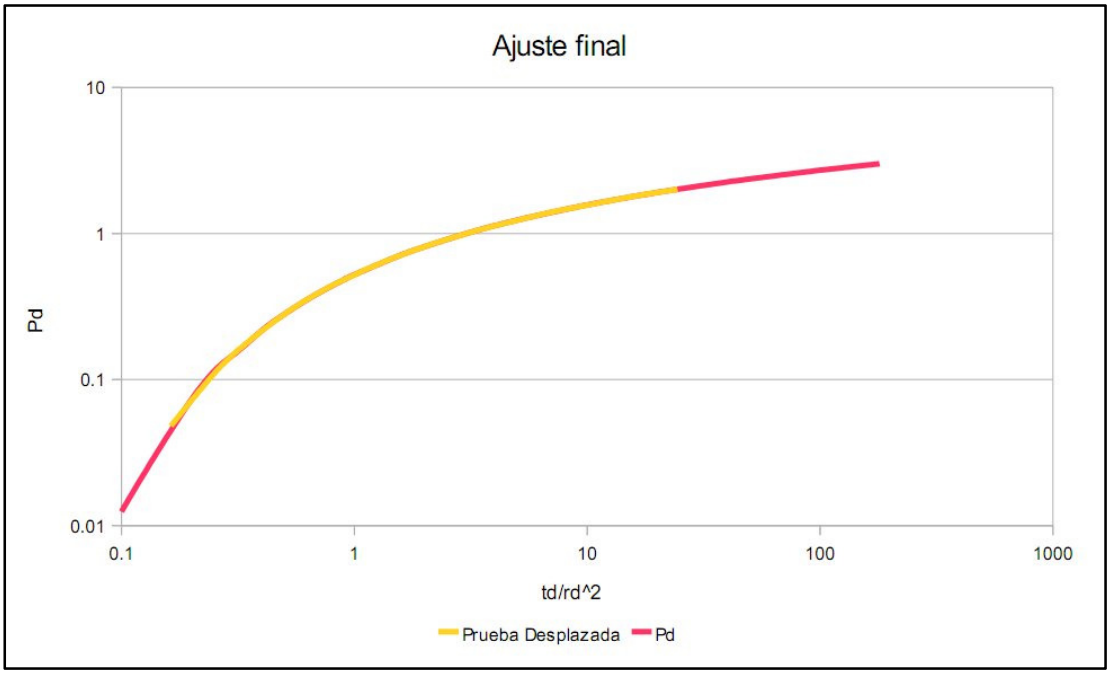


FIGURA 3.13 AJUSTE NUMÉRICO, PASO 7

La tabla de la gráfica anterior es:

TABLA 3-4

TD	Prueba Ajustada	Curva Tipo	TD	Prueba Ajustada	Curva Tipo
0.163	0.047	0.048	3.912	1.163	1.118
0.245	0.106	0.106	5.868	1.369	1.31
0.326	0.201	0.165	8.15	1.564	1.469
0.489	0.331	0.273	14.67	1.753	1.756
0.815	0.531	0.445	19.56	1.901	1.898
1.63	0.732	0.723	24.45	2.007	2.008
2.934	0.962	0.984			

Una vez ajustada la gráfica, debemos proceder a calcular el Match Point y continuar con el proceso de interpretación.

De acuerdo con la teoría expuesta, lo primero garantizar una selección correcta de la ubicación del punto Match.

Esto lo simplificamos al seleccionar uno de los pares ajustados, el primero o el último, ya que así se cumple con lo establecido en la teoría de que el Match Point esté contenido en el área bajo la curva de ambas gráficas.

0.163	0.047

Las lecturas para el eje correspondiente a la prueba original, se toman tal cual aparece en la tabla, es decir que obtuvimos directamente la lectura de las coordenadas del punto match al hacerlo coincidir convenientemente con el punto mencionado.

Para el caso de las coordenadas del Match en los ejes de la Curva Tipo (que teóricamente estarían en otro papel (log – log)), se multiplican los valores del Match por el factor obtenido con el desplazamiento total (mencionado al final de la explicación del ajuste vertical). El cálculo del desplazamiento total está relacionado con todas las pequeñas aproximaciones que se hacen para empatar la curva. Dado que el código contiene la forma en que se acumula dicho valor, decidimos omitir ese cálculo en esta sección y simplemente reportaremos el valor que el programa reporta.

Aplicando estos valores a las ecuaciones 3.13 y 3.14, Obtenemos finalmente el valor de la permeabilidad y el producto de la porosidad por la compresibilidad.

$k = 55.4549794697[mD]$ y $\emptyset * ct = 2.06924609846\text{E-}6$ respectivamente.

3.3.1. CÓDIGO FUENTE DE LA CLASE CURVATIPO

La clase *CurvaTipo* utiliza variables públicas (a las que se pueden acceder totalmente desde fuera de la clase), variables privadas a las que solo se pueden acceder desde la misma clase y por herencia en otras clases y variables protegidas las cuales solo podrán actuar dentro de la clase CurvaTipo. Esto con el objetivo de que los programadores involucrados en la ampliación y mantenimiento de la plataforma, no tengan problemas para trabajar en sus propios módulos si estos heredan una clase ya disponible.

La declaración de dichas variables se hizo al inicio, de la siguiente manera:

```
 1 <?php
 2
 3 class CurvaTipo {
 4
 5    //Declaración de variables globales
 6    //Declaración de variables privadas
 7    privare $arreglo; //Un arreglo utilizado para almacenar los datos de la curva ajustada
 8      privare $numDatos; //variable para almacenar el número de datos contenidos en un
arreglo
 9     private $maxDatos; //$numDatos-1,  por la convención de comenzar los arreglos en el
renglón cero.
10    private $paso; //el tamaño del paso para aproximar el ajuste horizontal de la curva
11    private $tolerancia; //la tolerancia de error en el ajuste
12     private $diferencia; //Variable para determinar si la posición de la cura original está a la
derecha o a la izquierda de la curva tipo
13    private $props; //el arreglo que contiene las propiedades
14    private $Pd; //Presión adimensional
15    private $td; //Tiempo adimensional
16    private $despVtotal; //desplazamiento vertical total
17    private $despHtotal; //desplazamiento horizontal total
18        private $iteraciones; //número de iteraciones del método. Los experimentos han
demostrado que 5 es más que suficiente
19    private $parMatch; //El par de datos seleccionados para ubicar el Match Point
20    private $trMatch; //valor del punto Match para t real
21    private $PrMatch; //valor del punto Match para delta P real
22    private $tdMatch; //valor del punto Match para td
23    private $PdMatch; //valor del punto Match para Pd
24    //Decalaración de variables públicas
25    public $ajustado;
26    public $original;
27    public $perm;
28    public $fict;
```

La función __construct en una clase de PHP sirve para inicializar variables. Aquí se puede modificar si se desea, la calidad de la aproximación ya sea con el tamaño del paso para cada td nuevo en la aproximación horizontal, el número de iteraciones para ajustar horizontal y verticalmente la curva o bien la tolerancia del error que se genera en la diferencia de presiones como parte de la comprobación del ajuste horizontal.

Posteriormente se inicia el programa, ejecutando el conjunto de instrucciones del método, agrupadas en la función

```
$this->calcularMatchPoint($this->arreglo[$this->maxDatos][0])
```

Cuyo argumento es el primer valor de tiempo propuesto para calcular una Pd.

```
30    //Constructor (inicialización de variables)
 31    function __construct($props, $arreglo) {
 32        $this->arreglo = $arreglo;  //Tiempo V.S. Delta P
 33        $this->props = $props; //propiedades roca / fluido
 34        $this->numDatos = count($this->arreglo); //Cuenta los elementos del arreglo
 35        $this->maxDatos = ($this->numDatos - 1); // numDatos-1
 36        $this->iteraciones = 5; //número de iteraciones
 37        $this->paso = 0.001; //tamaño del paso
 38        $this->tolerancia = 0.002; //margen de error en el ajuste
 39        $this->calcularMatchPoint($this->arreglo[$this->maxDatos][0]); //Acciona el primer
cï¿½lculo
 40    }
```

Se observa que el método se ejecuta una vez, para generar algunos valores que se requerirán más adelante, en la etapa de las iteraciones, tales como los arreglos que contienen los datos de la curva desplazada.

En términos generales, la estructura "arreglo" contendrá la siguiente información:

Arreglo [i,0]	Arreglo [i,1]	Arreglo [i,2]	Arreglo [i,3]
Tiempo original en variables reales (tr)	Presión original en variables reales (Pr)	Tiempo desplazado en variables adimensionales (td)	Presión desplazada en variables adimensionales (Pd)

```
42    //Método de curvaTipoRadial
 43    private function calcularMatchPoint($td) {
 44        $this->td = $td;
 45        $this->curvaTipoRadial($this->td);
 46        $this->despVertical($this->Pd);
 47        $this->curvaTipoRadial($this->arreglo[0][0]);
 48        $this->despHorizontal($this->Pd);
 49        for ($i = 1; $i < $this->iteraciones; $i++) {
 50            $this->despVtotal = 0; //Distancia total, desplazada verticalmente
 51            $this->despHtotal = 0; //Distancia total, desplazada horizontalmente
 52            $this->curvaTipoRadial($this->arreglo[$this->maxDatos][2]);
 53            $this->despVertical($this->Pd);
 54            $this->curvaTipoRadial($this->arreglo[0][2]);
```

```
55          $this->despHorizontal($this->Pd);
56      }
57      $this->construirArreglos($this->arreglo);
58      $this->calcularPropiedades($this->arreglo);
59  }
60
```

En la siguiente función está la información necesaria para calcular una Pd de la curva tipo para flujo radial, incluyendo la decisión de utilizar el polinomio más conveniente de acuerdo con la teoría expuesta.

```
//Función calcularPropiedades
62  private function curvaTipoRadial($td) {
63      $this->td = $td;
64      //Coeficientes de la curva tipo
65      //Polinomio uno
66      $a0 = -0.57721566;
67      $a1 = 0.99999193;
68      $a2 = -0.24991055;
69      $a3 = 0.05519968;
70      $a4 = -0.00976004;
71      $a5 = 0.00107857;
72      //Polinomio dos
73      $aa1 = 8.5733287401;
74      $aa2 = 18.0590169730;
75      $aa3 = 8.6347608925;
76      $aa4 = 0.2677737343;
77      $b1 = 9.5733223454;
78      $b2 = 25.6329561486;
79      $b3 = 21.0996530827;
80      $b4 = 3.9584969228;
81      //Selección del polinomio
82      $x = 1 / (4 * ($this->td));
83      if ($x <= 1) {
84          $Eix = $a0 + $a1 * $x + $a2 * pow($x, 2) + $a3 * pow($x, 3) + $a4 * pow($x, 4) + $a5 *
pow($x, 5) - log($x);
85      } else {
86          $Eix = (pow($x, 4) + $aa1 * pow($x, 3) + $aa2 * pow($x, 2) + $aa3 * $x + $aa4);
87          $Eix = $Eix / (pow($x, 4) + $b1 * pow($x, 3) + $b2 * pow($x, 2) + $b3 * $x + $b4);
88          $Eix = $Eix / ($x * exp($x));
89      }

90  //Calcula Pd
91      $this->Pd = 0.5 * ($Eix);
92  }
93
94  //Escalamiento vertical
95  private function despVertical($Pd) {
96      $this->Pd = $Pd;
97      $despVertical = log10($this->Pd) - log10($this->arreglo[$this->maxDatos][1]);
98      for ($i = 0; $i < $this->numDatos; $i++) {
99          $this->arreglo[$i][3] = $this->arreglo[$i][1] * pow(10, $despVertical);
100     }
```

"$this->despVtotal" es la variable que se utilizará más adelante en el cálculo del Match Point ayudando a leer las coordenadas en la "hoja con la curva original" y en la "hoja de la curva tipo". Este valor inicia en cero y se reinicializa con cada iteración.

```
101         $this->despVtotal = $this->despVtotal + $despVertical;
102     }
103
104     //Escalamiento Horizontal
105     private function despHorizontal($Pd) {
106         $this->Pd = $Pd;
107         $this->diferencia = $this->Pd - $this->arreglo[0][3];
```

Aquí se observa que el proceso de verificación se realiza por lo menos una vez contemplando la poco probable situación de que las curvas estén inicialmente empatadas.

```
108         do {
109             if ($this->diferencia >= 0) {
110                 $this->td = $this->td - $this->paso;
111             } else if ($this->diferencia <= 0) {
112                 $this->td = $this->td + $this->paso;
113             }
114             $this->curvaTipoRadial($this->td);
115             $this->diferencia = $this->Pd - $this->arreglo[0][3];
116         } while (abs($this->diferencia) > $this->tolerancia);
```

Aquí se calcula la otra variable para el cálculo del Match Point.

```
117         $despHorizontal = log10($this->td) - log10($this->arreglo[0][0]);
118         $this->despHtotal = $this->despHtotal + $despHorizontal;
```

Aquí se multiplican cada uno de los elementos del tiempo original por el factor, para guardarlos en el arreglo correspondiente al tiempo desplazado

```
119         for ($i = 0; $i < $this->numDatos; $i++) {
120         $this->arreglo[$i][2] = $this->arreglo[$i][0] * pow(10, $despHorizontal);
121         }
122     }
123
124 //Construye los arreglos que se van a entregar a la siguiente sección para mostrar la gráfica y generar el archivo de excel
125     private function construirArreglos($arreglo) {
126         $this->arreglo = $arreglo;
127         for ($i = 0; $i < $this->numDatos; $i++) {
128             $this->ajustado[$i][0] = $this->arreglo[$i][2]; //Tiempo desp
129             $this->ajustado[$i][1] = $this->arreglo[$i][3]; //Presion desp
130             $this->curvaTipoRadial($this->ajustado[$i][0]);
131             $this->ajustado[$i][2] = $this->Pd;
132         }
133         for ($i = 0; $i < $this->numDatos; $i++) {
134             $this->original[$i][0] = $this->arreglo[$i][0]; //Tiempo Original
135             $this->original[$i][1] = $this->arreglo[$i][1]; //Presión Original
136             $this->curvaTipoRadial($this->original[$i][0]);
137             $this->original[$i][2] = $this->Pd;
138         }
139     }
```

```
140
141     private function calcularPropiedades($arreglo) {
142         $this->arreglo = $arreglo;
143         $this->parMatch = ceil(count($this->arreglo) / 2); //Selección del Match Point
144         $this->trMatch = $this->arreglo[$this->parMatch][2]; //valor del  Match Point para t real
145         $this->PrMatch = $this->arreglo[$this->parMatch][3]; //valor del  Match Point para delta
P real
146         $this->tdMatch = $this->trMatch * pow(10, -$this->despHtotal); //valor del Match Point
para td
147         $this->PdMatch = $this->PrMatch * pow(10, -$this->despVtotal); //valor del  Match Point
para Pd
148         //Cálculo de las propiedades
149         $this->perm = (141.2 * $this->props[qbls] * $this->props[Bo] * $this->props[mu] / $this-
>props[h]) * ($this->PrMatch / $this->PdMatch);
150         $this->fict = ((0.0002637 * $this->perm) / ($this->props[mu] * pow($this->props[radio],
2))) * ($this->tdMatch / $this->PrMatch);
151     }
```

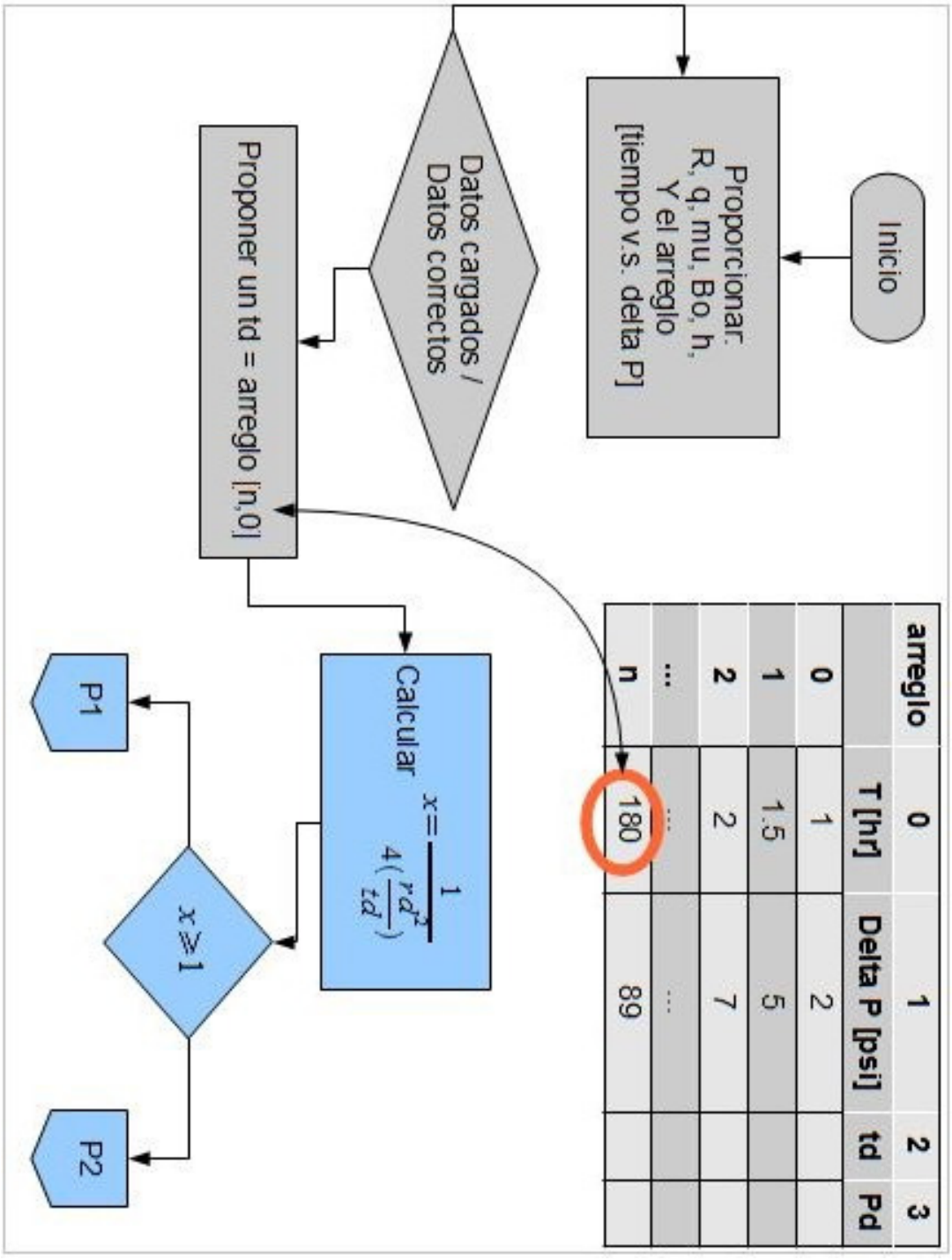
El siguiente conjunto de funciones se utiliza como una buena práctica de programación orientada a objetos.

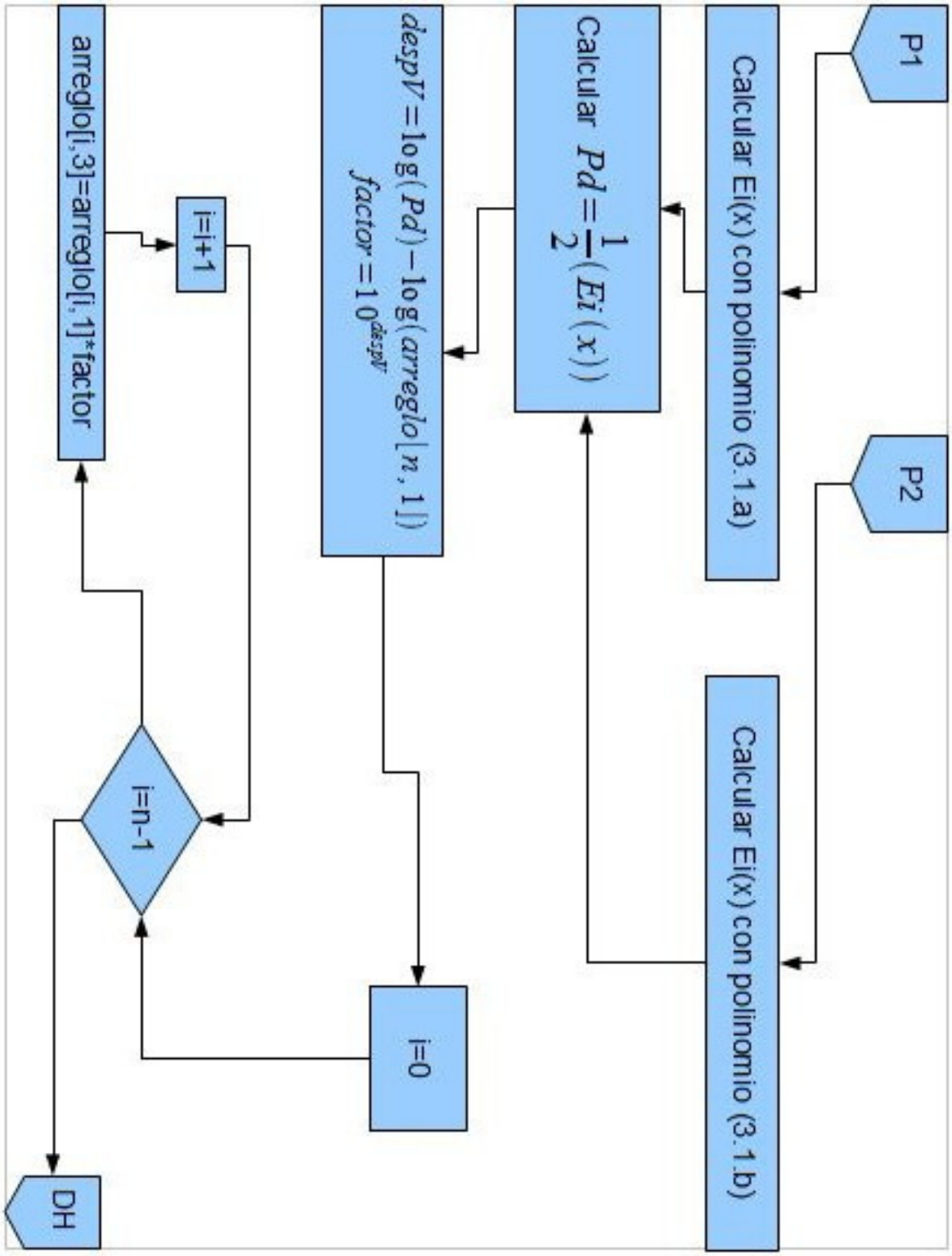
```
152 //Función que regresa la curva ajustada
153     public function get_ajustado() {
154         return $this->ajustado;
155     }
156 //Función que regresa la curva original
157     public function get_original() {
158         return $this->original;
159     }
160 //Regresa permeabilidad
161     public function get_perm() {
162         return $this->perm;
163     }
164 //Regresa elproducto permeabilidad * compresibilidad
165     public function get_fict() {
166         return $this->fict;
167     }
168
169 }
```

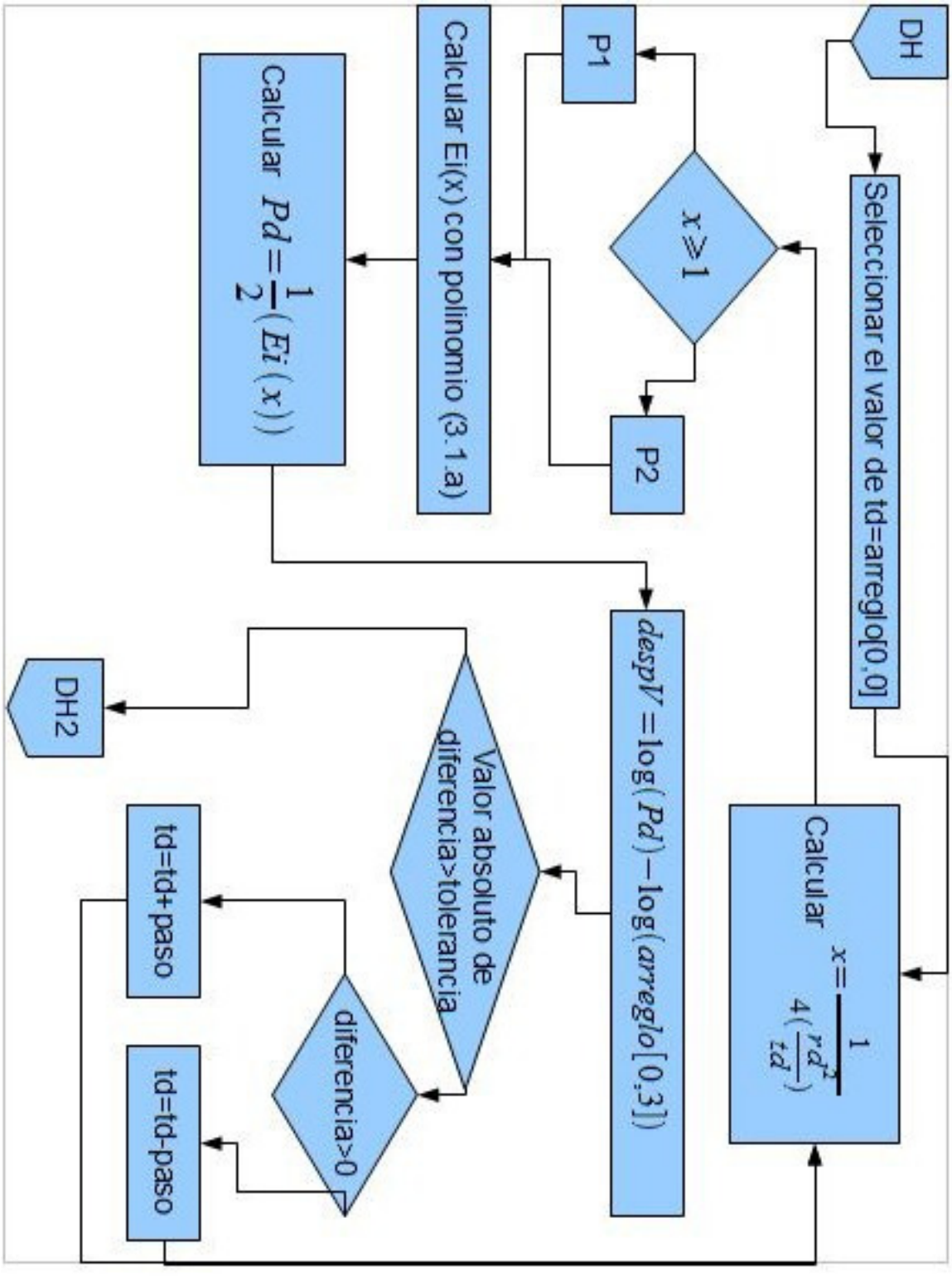
Termina la clase CurvaTipo

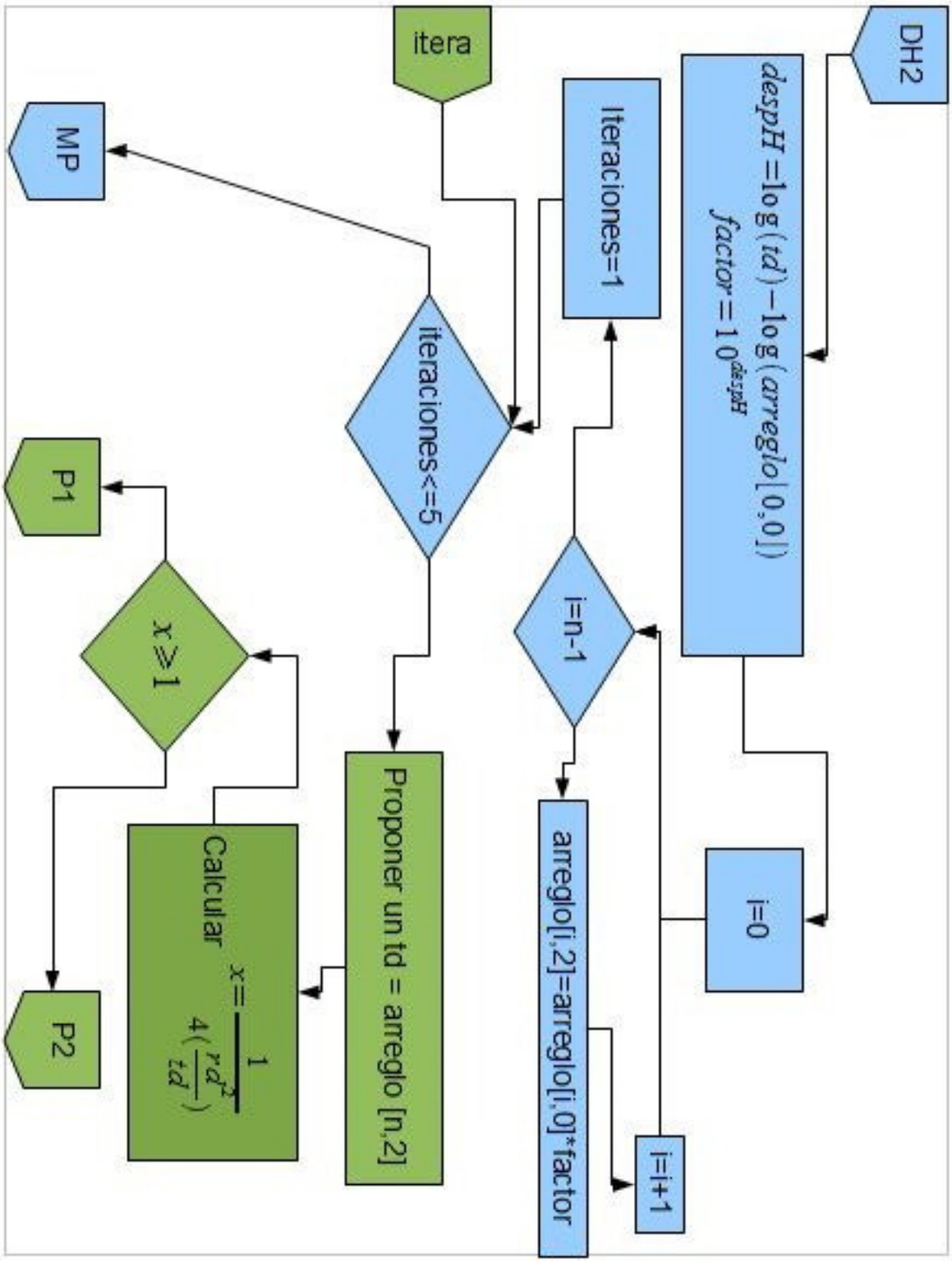
```
170
171 ?>
```

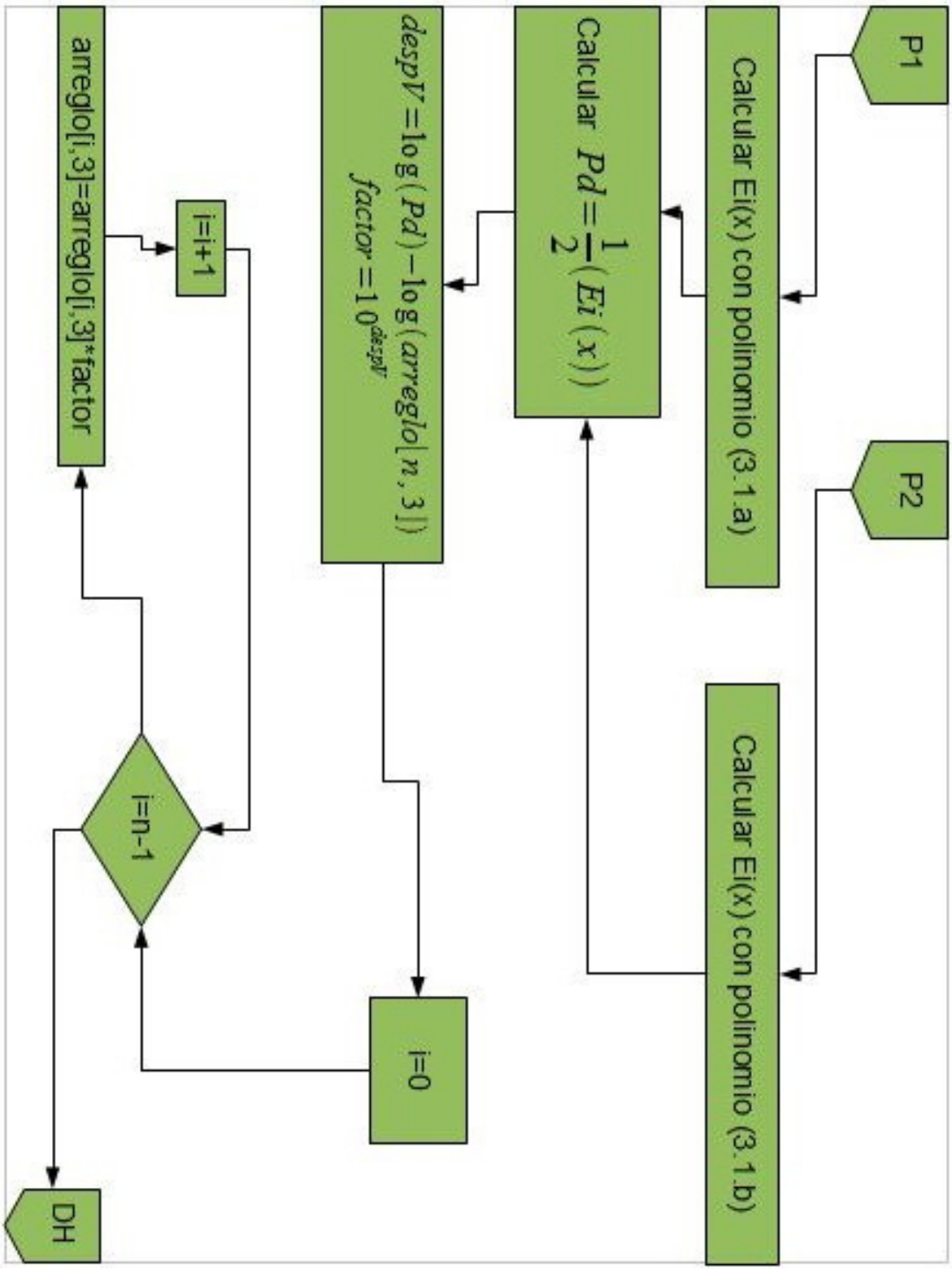
3.3.2. DIAGRAMA DE FLUJO DE LA CLASE CURVATIPO

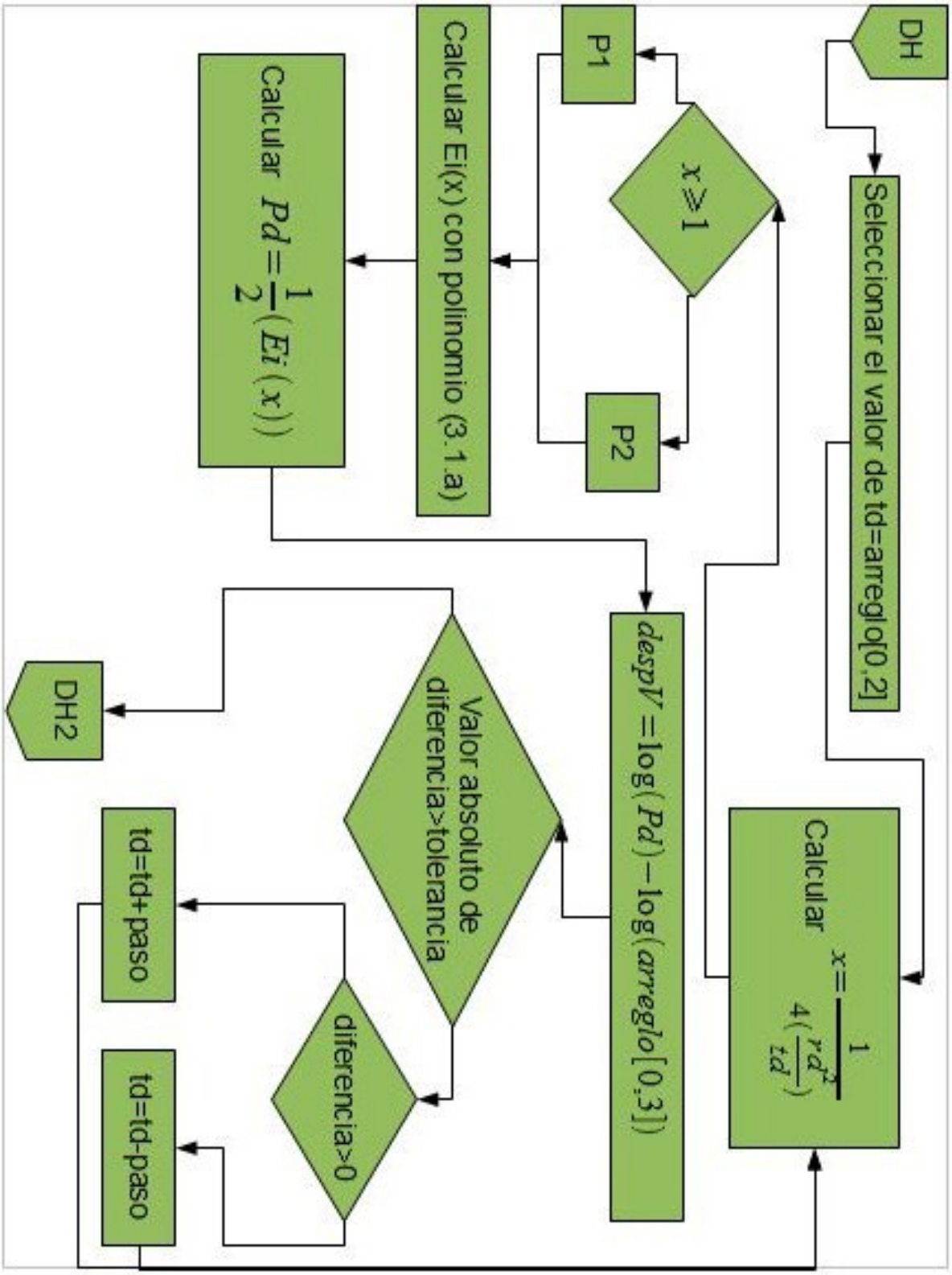

Inicio
Proporcionar:
R, q, mu, Bo, h,
Y el arreglo
[tiempo v.s. delta P]
Datos cargados /
Datos correctos
Proponer un td = arreglo [n,0]
Calcular $x = \frac{1}{4(\frac{r_d^2}{t_d})}$
$x \geq 1$
P1
P2
arreglo 0 1 2 3
T [hr] Delta P [psi] td Pd
0 1 2
1 1.5 5
2 2 7
... ...
n 180 89


P1
P2
Calcular Ei(x) con polinomio (3.1.a)
Calcular Ei(x) con polinomio (3.1.b)
Calcular $Pd = \frac{1}{2}(Ei(x))$
$despV = \log(Pd) - \log(arreglo[n,1])$
$factor = 10^{despV}$
i=0
i=n-1
i=i+1
arreglo[i,3]=arreglo[i,1]*factor
DH


DH
Seleccionar el valor de td=arreglo[0,0]
Calcular $x=\frac{1}{4(\frac{rd^2}{td})}$
$x \geq 1$
P1
P2
Calcular Ei(x) con polinomio (3.1.a)
Calcular $Pd=\frac{1}{2}(Ei(x))$
$despV=\log(Pd)-\log(arreglo[0,3])$
Valor absoluto de diferencia>tolerancia
DH2
diferencia>0
td=td+paso
td=td-paso


DH2
$despH = \log(td) - \log(arreglo[0,0])$
$factor = 10^{despH}$
Iteraciones=1
itera
MP
iteraciones<=5
P1
$x \geq 1$
P2
i=n-1
i=0
arreglo[i,2]=arreglo[i,0]*factor
i=i+1
Proponer un td = arreglo [n,2]
Calcular $x = \frac{1}{4(\frac{rd^x}{td})}$


P1
P2
Calcular Ei(x) con polinomio (3.1.a)
Calcular Ei(x) con polinomio (3.1.b)
Calcular $Pd = \frac{1}{2}(Ei(x))$
$despV = \log(Pd) - \log(arreglo[n,3])$
$factor = 10^{despV}$
i=0
i=n-1
i=i+1
arreglo[i,3]=arreglo[i,3]*factor
DH


DH
Seleccionar el valor de td=arreglo[0,2]
Calcular $x=\frac{1}{4(\frac{rd^2}{td})}$
$despV=\log(Pd)-\log(arreglo[0,3])$
Valor absoluto de diferencia>tolerancia
DH2
diferencia>0
td=td+paso
td=td-paso
$x \geq 1$
P1
P2
Calcular Ei(x) con polinomio (3.1.a)
Calcular $Pd=\frac{1}{2}(Ei(x))$

3.4. MÓDULO DE SUAVIZADO DE DATOS

En un conjunto de datos donde la variable independiente es el tiempo, suelen existir fluctuaciones entre los datos medidos y los que en realidad se manifiestan, por lo que es necesario corregir dichas series de datos mediante métodos estadísticos como el del *Promedio Móvil*.

La fórmula para corregir es la siguiente:

$$\frac{i_1+i_2+\ldots+i_N}{N}, \frac{i_2+i_3+\ldots+i_{N+1}}{N}, \frac{i_3+i_4+\ldots+i_{N+2}}{N}$$

Donde *N* es el número de datos de la ventana de suavizado, *i* es cada uno de los datos por corregir [25].

Las sumas en el numerador de la sucesión se llaman totales móviles de orden *N*. La ventana de suavizado es el rango en el que el método se apoya para corregir los datos.

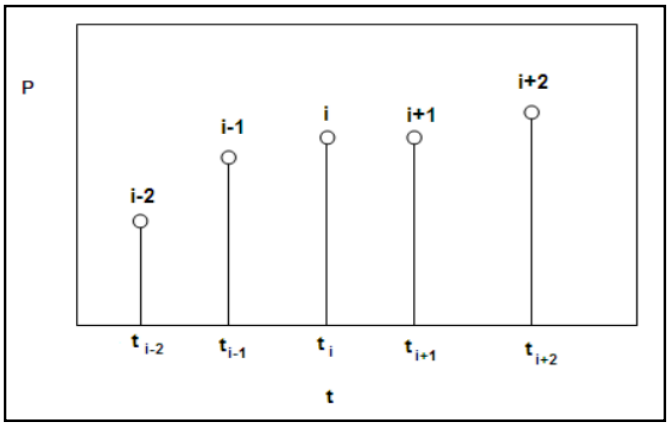
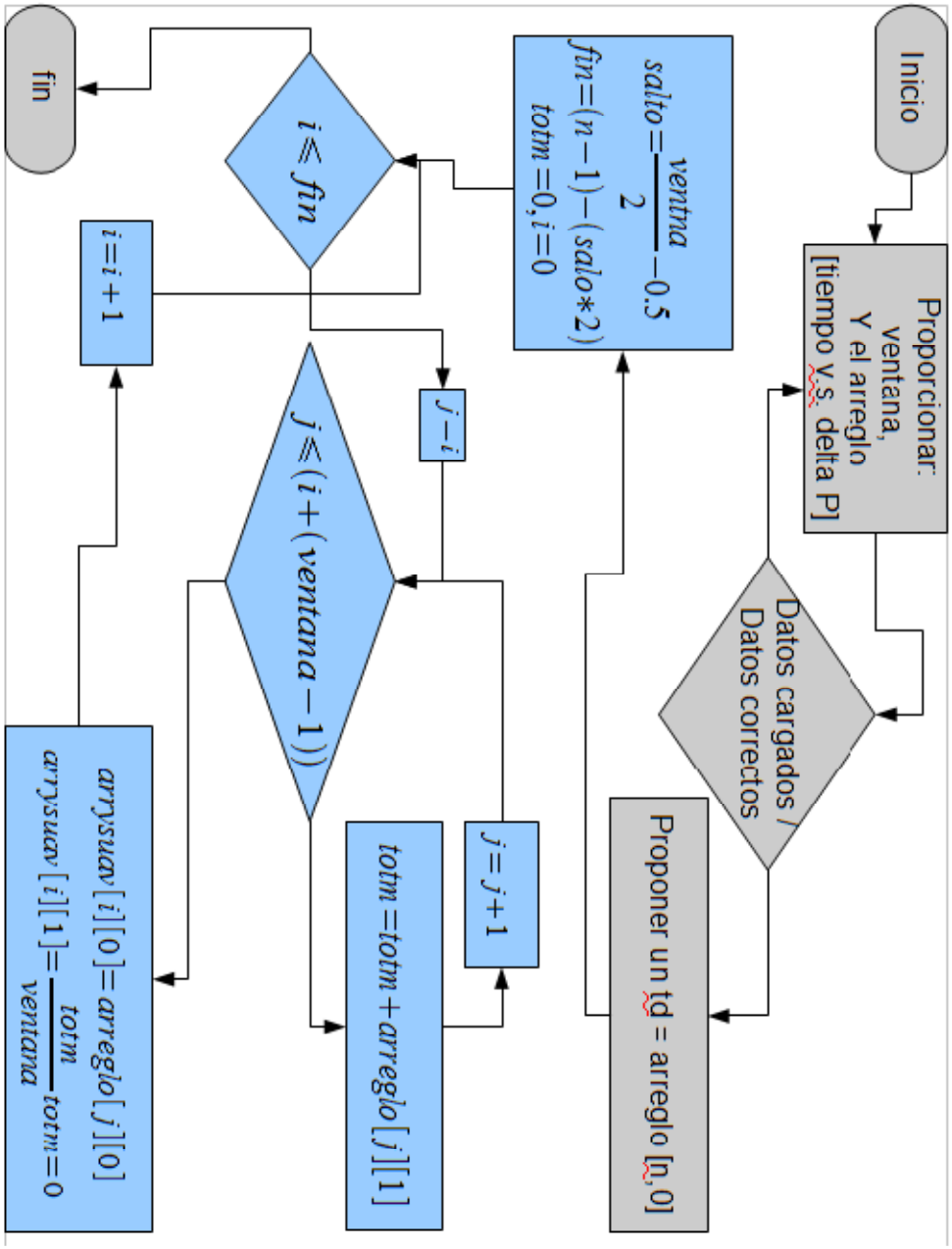


FIGURA 3.14 VENTANA DE SUAVIZADO DE DATOS (NPS)

3.4.1. CÓDIGO FUENTE DEL MÓDULO DE SUAVIZADO DE DATOS

```
<?php

class PromedioMovil {

   function __construct($ventana, $arreglo) {
      $this->arreglo = $arreglo;
      $this->numDatos = count($this->arreglo);
      $this->maxDatos = ($this->numDatos - 1);
      $this->ventana = $ventana;
      $this->promedio($this->ventana);
   }
   function promedio($ventana) {
      for($i=0;$i<=count( $this->arreglo);$i++){
        // echo '<br>'. $this->arreglo[$i][0].'    -->    '. $this->arreglo[$i][1];
}
      $this->ventana = $ventana;
      $salto = ($this->ventana / 2) - .5;
      $fin = $this->maxDatos - ($salto * 2);
      $totm = 0;
      for ($i = 0; $i <= $fin; $i++) {
         for ($j = $i; $j <= ($i+($ventana - 1)); $j++) {
            $totm = $totm + $this->arreglo[$j][1];
           // echo '<br> suma='.$totm;
         }
         // echo '<br> $totm=' . ($totm) / $this->ventana;
         $this->arrysuav[$i][0] = $this->arreglo[$i][0];
         $this->arrysuav[$i][1] = $totm / $this->ventana;
         $totm = 0;
      }
   }

   function get_suavizado(){
      return $this->arrysuav;
   }

}

?>
```

3.4.2. DIAGRAMA DE FLUJO DEL MÓDULO DE SUAVIZADO


Inicio
Proporcionar:
ventana,
Y el arreglo
[tiempo v.s. delta P]
Datos cargados /
Datos correctos
Proponer un td = arreglo [n, 0]
salto = ventana / 2 − 0.5
fin = (n − 1) − (salo * 2)
totm = 0, i = 0
i ⩽ fin
fin
i = i + 1
j − i
j ⩽ (i + (ventana − 1))
totm = totm + arreglo [j] [1]
j = j + 1
arrsuav [i] [0] = arreglo [j] [0]
arrsuav [i] [1] = totm / ventana
totm = 0

Capítulo 4 RESULTADOS

En la referencia consultada [7] se proporciona el ejemplo utilizado en el capítulo anterior y sobre el cuál haremos una comparación entre su solución con el método gráfico y con Apantli.

El autor hace el siguiente ajuste:

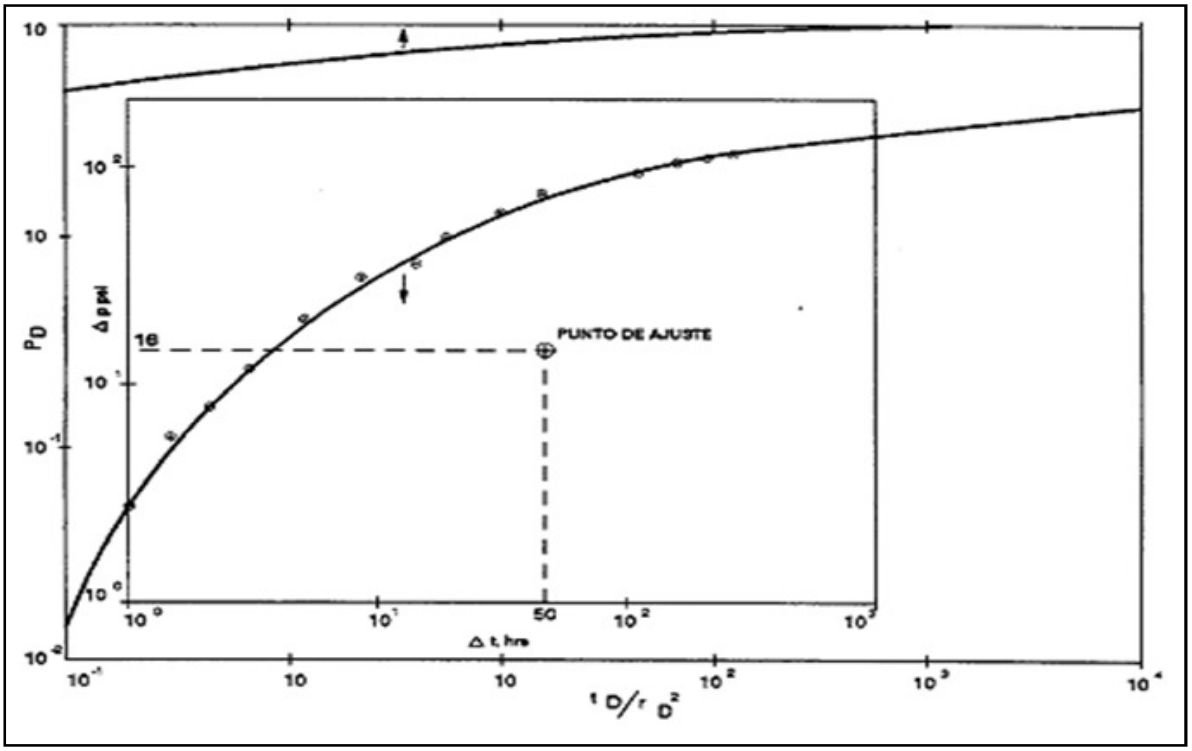


FIGURA 4.1 EJEMPLO DE INTERPRETACIÓN CON EL MÉTODO GRÁFICO DELMATCH POINT

A partir del Match point seleccionado, se obtienen los siguientes puntos de ajuste:

Δt = 50

ΔP = 16

$t_D/r_D^2 = 8$

PD = 0.37

Se estima la permeabilidad a partir de los puntos de ajuste:

$$k = 141.2\frac{q\mu B}{h}\left(\frac{P_D}{\Delta P}\right)_M = 54.32[mD]$$

y el factor ΦCt se obtiene a partir de la ecuación:

$$\phi c_t = \frac{0.000264}{r^2}\frac{k}{\mu}\frac{\Delta t_M}{\left(\frac{t_D}{r_D^2}\right)_M} = 9.69\text{x}10^{-7}$$

Se hizo la comparación de los resultados obtenidos con los dos métodos expuestos en este trabajo:

TABLA 4-1

Parámetro	Método Gráfico	Apantli
Permeabilidad [mD]	54.32	55.45
Producto Ø*Ct[]	9.69E-007	2.12E-6

La calidad del ajuste se observa en la siguiente gráfica:

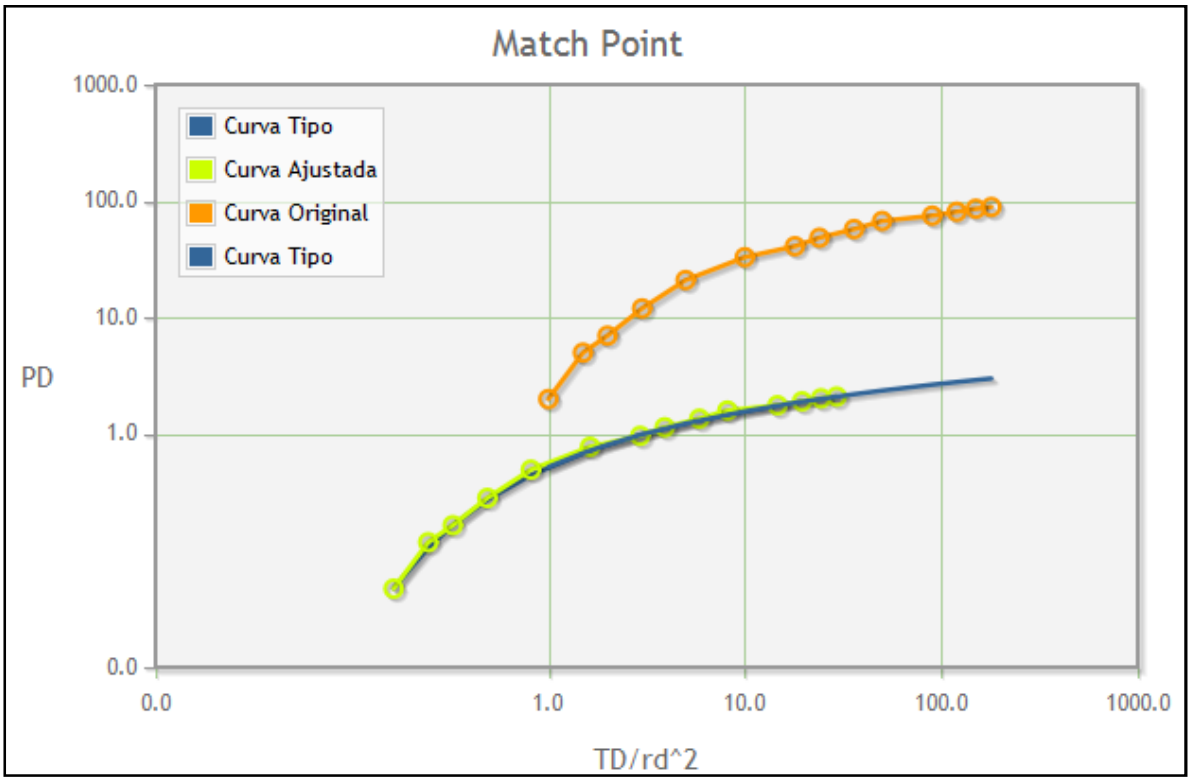


FIGURA 4.2 EFICIENCIA DE AJUSTE

En este caso los resultados varían muy poco, pero sostenemos que un ajuste numérico tendrá mucho menos incertidumbre que un ajuste gráfico, considerando la agudeza visual para determinar cuando las curvas están empatadas y posteriormente, tomando las lecturas en los ejes correspondientes, que no tienen una resolución muy alta.

A continuación haremos un análisis detallado de algunos cálculos que el módulo "Suavizado" y del módulo "Match Point" de la plataforma Apantli realizan.

Comenzamos con el módulo de suavizado.

Dado que consideramos que los datos proporcionados en el ejemplo son pocos, decidimos interpolar datos (pasando de 14 a 63 pares de puntos) utilizando una regresión logarítmica cuya precisión fue de (R^2=0.998), además de respetar los valores medidos en la prueba original.

Hicimos un ajuste con Apantli para la prueba tal cual aparece en la literatura y otra para la prueba con datos interpolados, obteniendo los siguientes resultados:

TABLA 4-2

Parámetro	Prueba original	Prueba interpolada	% Error
Permeabilidad [mD]	55.45	55.45	0
Producto Ø*Ct[]	2.07E-006	2.12E-6	Diferencia: 0.05E-6

Por lo que calificamos a la curva interpolada como confiable y procedemos a distorsionar los valores de manera aleatoria con la siguiente fórmula en la hoja de cálculo:

$$Valor\ con\ ruido = Valor\ original + ((-1^{a})) * factor\ de\ ruido$$

ECUACIÓN 4-1

Donde:
a=un número aleatorio redondeado hacia el entero superior, entre 1 y 4factor de ruido = un número entre 1 y 10 multiplicado por un número aleatorio entre cero y uno.
Utilizando un factor de ruido de 9, obtuvimos los siguientes datos:

TABLA 4-3

T	ΔP	T	ΔP	T	ΔP	T	ΔP
1	3.44	43	59.30	90	76.90	139	82.95
1.5	2.91	46	64.42	94	79.68	142	87.19
2	4.46	49	62.31	97	74.48	145	82.79
3	9.36	50	63.37	100	75.75	148	89.31
5	21.72	55	67.46	103	75.69	150	88.29
10	29.82	58	67.31	106	76.40	154	83.97
13	40.48	61	69.04	109	82.65	157	85.25
16	38.53	64	73.67	112	75.30	160	82.50
18	44.49	67	70.51	115	84.56	163	82.77
20	44.25	70	70.31	118	78.81	166	84.98
24	48.88	73	68.33	120	78.39	169	88.69
28	50.49	76	76.00	124	83.47	172	85.68
31	53.44	79	68.56	127	85.57	175	85.62
34	59.85	82	72.87	130	80.30	178	85.18
36	55.88	85	78.58	133	78.90	180	85.72
40	62.52	88	70.82	136	84.13		

La prueba con ruido respecto a al prueba original se ve en la figura siguiente:

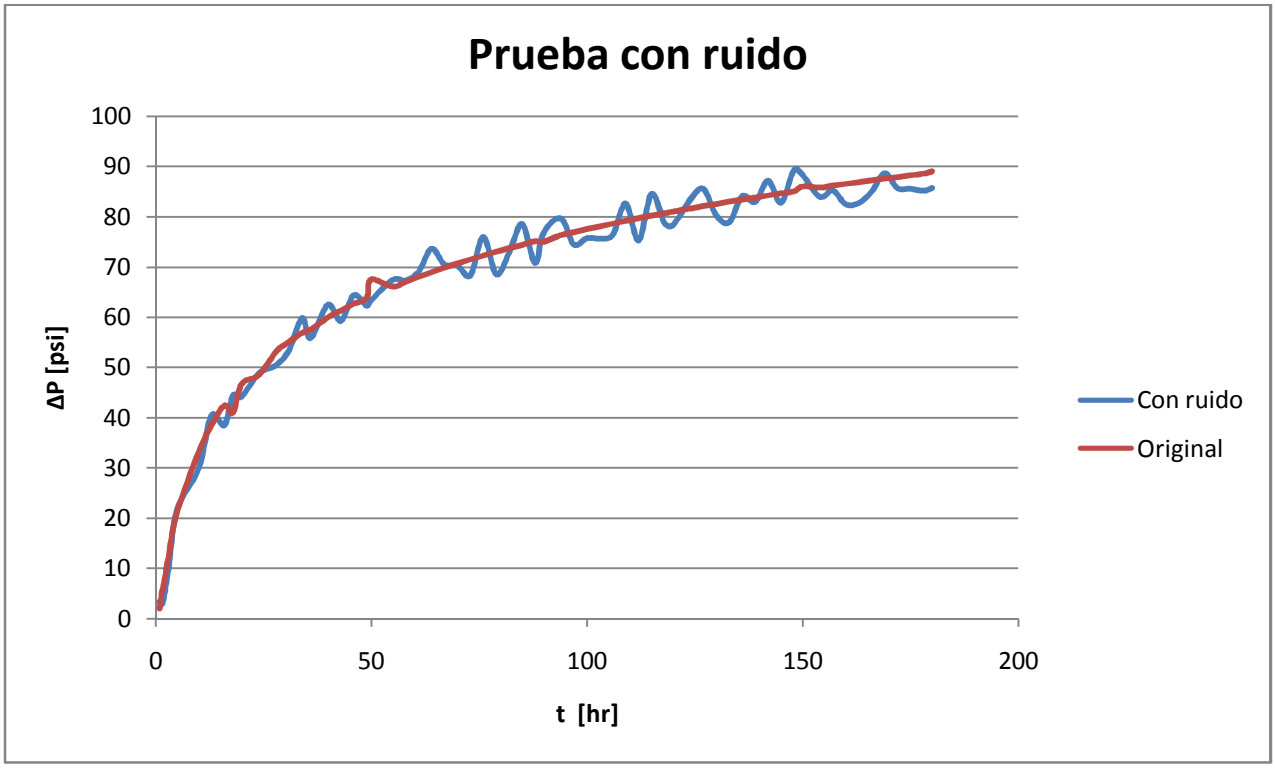


FIGURA 4.3 PRUEBA CON RUIDO

La curva suavizada y ajustada se ve de la siguiente manera:

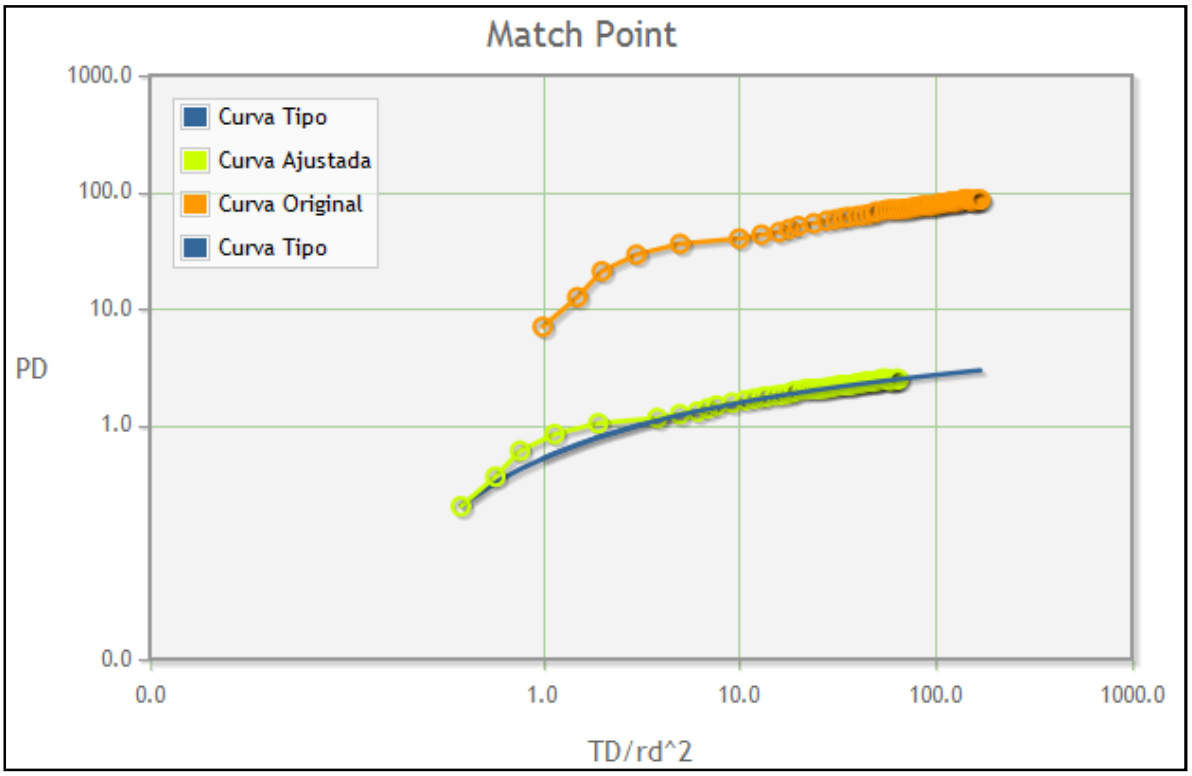


FIGURA 4.4 SUAVIZADO E INTERPRETACIÒN

Obteniendo los siguientes resultados:

TABLA 4-4

Parámetro	Prueba original	Prueba con ruido
Permeabilidad [mD]	55.45	68.17
Producto Ø*Ct[]	2.07E-006	6.058 E-7

Observamos un error importante en el cálculo de la permeabilidad, lo cual nos indica que el método de empate funciona bien, pero necesita otros métodos para evaluar la calidad del ajuste antes de calcular las propiedades.

Esta será la primera tarea en la que los autores y las personas interesadas, trabajaremos para desarrollar una versión mejorada del módulo.

No obstante, el módulo de suavizado mostró un buen desempeño al intentar corregir el ruido en los datos modificados.

La eficiencia del módulo de suavizado se aprecia en la siguiente figura:

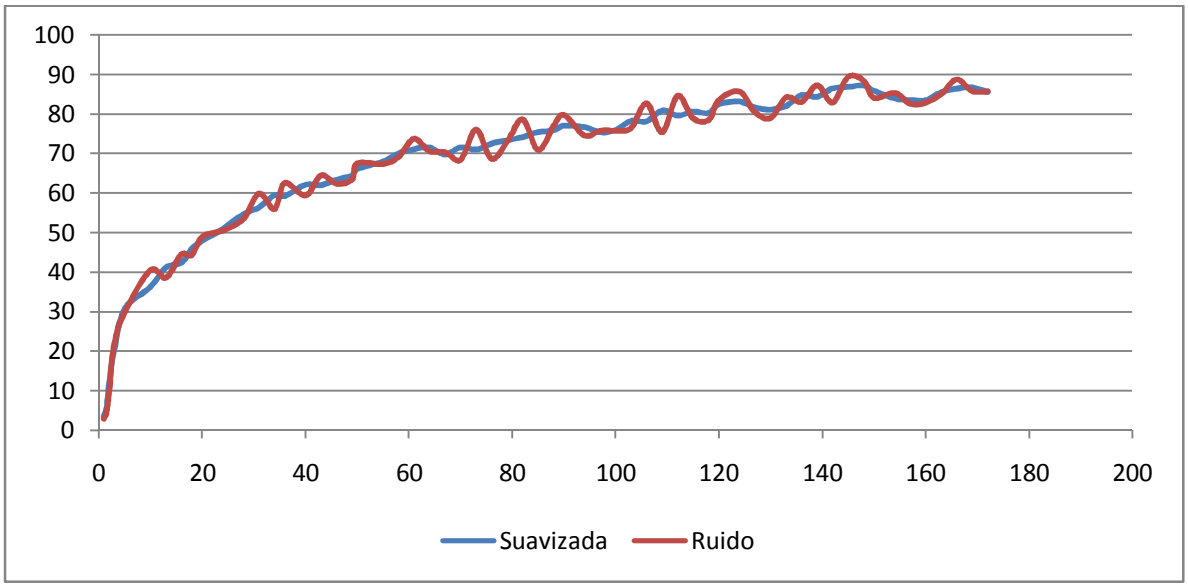


FIGURA 4.5 SUAVIZADO Y RUIDO

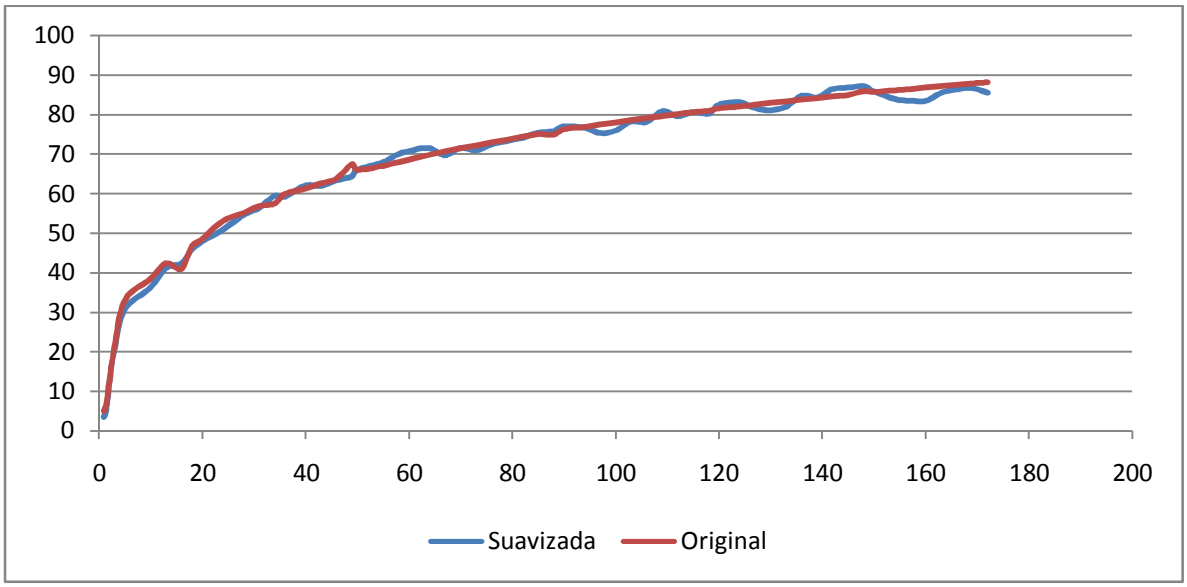


FIGURA 4.6 SUAVIZADO Y ORIGINAL

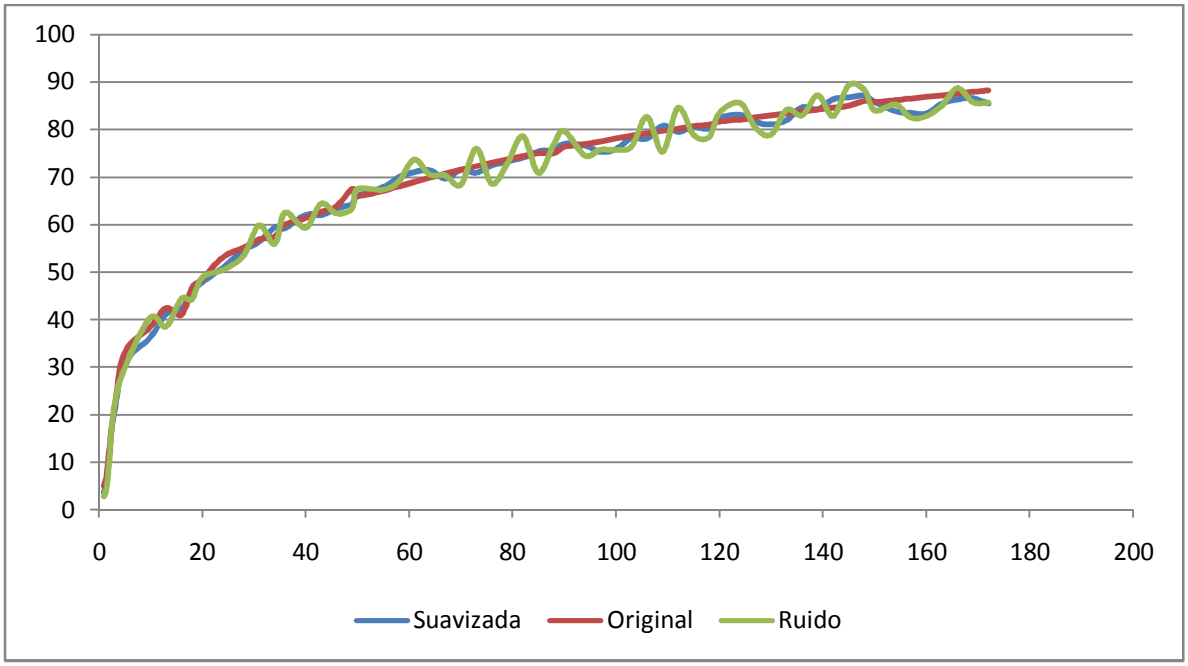


FIGURA 4.7 SUAVIZADO, ORIGINAL Y RUIDO

Como se mencionó en el capítulo anterior, los efectos de almacenamiento y daño pueden provocar que a tiempos breves después de iniciada la prueba, las lecturas de presión no sigan el comportamiento de la solución fuente lineal.

En cuanto a la potencia y la velocidad de la plataforma, mencionamos nuevamente que está en función del equipo que hospede el servicio.

Mientras que el servicio en www.atlahua.com/tesis está limitado utilizar 32 MB de RAM, puede procesar 7,500 datos y un tiempo máximo de ejecución para un programa, de 30 segundos. Mientras que una instalación local con 512 MB logró procesar 45,000 datos en 1 minuto 10 segundos.

En el foro de discusión y ayuda estará disponible el procedimiento para asignar dichos recursos a una instalación manual.

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Apantli demostró ser una plataforma de software capaz de funcionar en dispositivos con diferentes sistemas operativos y utilizando muy pocos recursos de hardware.

Debido a su arquitectura basada en un diseño Modelo Vista Controlador facilita la participación de otras personas en el proceso de ampliación y mantenimiento de la plataforma.

Un claro ejemplo es que el programador no debe preocuparse por la parte visual o por entender cómo funcionan todos los módulos. Simplemente debe conocer cómo llamar a un modulo, qué información proporcionarle y qué información esperar de regreso.

Este trabajo es una prueba de que se pueden estimular habilidades y actitudes para resolver problemas desarrollando programas de computo con herramientas de código abierto sin hacer ningún gasto en licencias, sin equipos muy sofisticados, sin privilegios de administración en equipos públicos, sin importar el sistema operativo y con la facilidad de encontrar bastantes recursos para aprender a todos los niveles y en diferentes idiomas, el lenguaje PHP. Lo que permite comenzar a programar de inmediato sin tener que aprender a utilizar una interfaz de desarrollo en particular.

Al crear el ambiente de programación local, es posible utilizar la plataforma sin ninguna conexión a internet.

Al existir librerías que se enfocan a la parte gráfica y de imagen de la página, es muy didáctico y útil poder ver las gráficas requeridas o que arrojan las interpretaciones de los datos que se han obtenido de los diferentes procesos de los programas hospedados en Apantli, todo desde el mismo navegador, sin necesidad de un programa adicional.

Se encontró también congruencia con el porcentaje que relaciona los métodos de enseñanza con el aprendizaje. De acuerdo a lo establecido por Acuña, se prende:

- 5% de lo que se expone.
- 10% de lo que se lee.
- 20% de lo que se ve y se escucha.
- 30% de lo que se demuestra.
- 50% de lo que se discute en grupo.
- 75% de lo que se practica.
- 80% de lo que se enseña.

Por lo cual se manifiesta una total empatía con la filosofía de aprender enseñando.

Al utilizar las tecnologías recomendadas en este trabajo, un alumno con conocimientos básicos o nulos de programación podría facilitar su aprendizaje llevándolo incluso a un plano autodidacta que además le beneficiará en el resto de su desempeño profesional.

Respecto al funcionamiento del módulo de CurvaTipo y de Suavizado, en base a lo reportado en el Capítulo 4, se aprecia que resultó ser una herramienta que facilita y acelera el proceso de interpretación de una prueba de interferencia, con la aproximación que el usuario considere.

Gracias al propio método y a la disponibilidad del código, el usuario puede utilizar su criterio y experiencia para adaptar el programa a sus necesidades.

RECOMENDACIONES

Los creadores de la plataforma Apantli recomendamos que los profesores de las asignaturas de la carrera de Ingeniería Petrolera promuevan el uso de la programación orientada a objetos, el desarrollo de nuevo software para dar solución a los problemas de Ingeniería Petrolera y obviamente el uso de la plataforma Apantli.

Los autores de este trabajo deseamos que se le de continuidad y que se adhieran a la plataforma nuevos programas enfocados a las diferentes áreas de Ingeniería Petrolera.

Creemos que a largo plazo, Apantli podría ser una colección de software que además de ser desarrollada por la comunidad universitaria, sea capaz de ofrecer servicios a todas aquellas empresas o instituciones relacionadas con la industria petrolera, que no estén conformes con lo que el software que utilicen les ofrezca.

Algunas áreas en las que se puede mejorar los módulos existentes son:

Incluir el análisis de la primera y segunda derivadas.

Incluir otros módulos de suavizado, enfocados a fenómenos específicos que se presentan durante la prueba.

Mejorar la interfaz donde se grafican los resultados.

Conectar la plataforma a una base de datos de pruebas realizadas.

Incluir las ecuaciones para interpretar pruebas con otras geometrías de flujo.

Incluir módulos de otras áreas de Ingeniería Petrolera.

GLOSARIO

Apache: Es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1[1] y la noción de sitio virtual.

Bug: Fallo en la programación de un programa, generalmente detectado por usuarios finales posterior a la etapa de desarrollo.

Debian: Una de tantas distribuciones de Linux.

Excel: Hoja de cálculo desarrollada por Microsoft

Fortran: Lenguaje de programación orientado al cálculo científico.

Framework: Es un conjunto de herramientas para desarrollar proyectos de programación de manera más eficiente.

FTP: Protocolo de transferencia de archivos.

Hacking: Adquisición y manipulación no autorizada de información alojada en un sistema de cómputo.

Hardware: Agrupa los recursos físicos de un equipo de cómputo (memoria, disco duro, procesador, etcétera).

Hospedaje (Hosting): Es el servicio que provee a los usuarios de Internet un sistema para poder almacenar información, imágenes, vídeo, o cualquier contenido accesible vía Web.

IDE: Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI).

LINUX (GNU/LINUX): Es la combinación del núcleo o kernel libre similar a Unix denominado Linux. Su desarrollo es uno de los ejemplos más prominentes de software libre; todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL (Licencia Pública General de GNU) y otra serie de licencias libres.

Mac Os X: es un sistema operativo desarrollado y comercializado por Apple Inc. Está basado en UNIX.

Macros: Programas que se desarrollan y ejecutan principalmente en hojas de cálculo.

Mandrake: Una de tantas distribuciones de Linux.

Match Point: Punto de ajuste utilizado para la interpretación de una prueba de presión.

Mathlab: Software que utiliza uno o varios lenguajes de programación para facilitar el estudio de ecuaciones matemáticas.

Office (Microsoft Office): Suite de productividad famosa por su procesador de texto, hoja de cálculo y creador de presentaciones.

RAM: Memoria de acceso aleatorio (Read Access Memory) indispensable para el desempeño de un equipo de cómputo, forma parte del hardware.

RedHat: Una de tantas distribuciones de Linux.

Script: Programa de cómputo constituido por un número pequeño de líneas de código.

Servidor: Es un equipo dedicado a hospedar y ejecutar servicios para otros usuarios conectados a este. Entre los servicios mas populares está el de correo electrónico.

Smart Phone: Teléfono celular que incorpora capacidades avanzadas de procesamiento de datos y ejecución de programas que lo asemejan más a una pequeña computadora personal.

SuSE: Una de tantas distribuciones de Linux.

UNIX: Sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T

Vi: Editor de texto, comúnmente utilizado en plataformas UNIX / LINUX.

Windows: Son una serie de sistemas operativos fabricados por Microsoft desde 1981

XAMPP: Es un servidor de plataforma independiente, que consiste principalmente en la base de datos MySQL, el servidor Web Apache y los intérpretes para lenguajes de script: PHP y Perl. El nombre proviene del acrónimo de X (para cualquiera de los diferentes sistemas operativos), Apache, MySQL, PHP, Perl. El programa está liberado bajo la licencia GNU y actúa como un servidor Web libre, fácil de usar y capaz de interpretar páginas dinámicas. Actualmente XAMPP esta disponible para Microsoft Windows, GNU/Linux, Solaris, y MacOS X.

BIBLIOGRAFÍA

1. http://www.ep-solutions.com/PDF/Literature/5297_PanSystem.pdf
2. http://www.fekete.com/services/welltesting.asp
3. http://www.aqtesolv.com/
4. http://www.openoffice.org
5. http://www.fsfla.org/svnwiki/about/what-is-free-software.es.html
6. http://www.fortran.com/compilers.html
7. http://www.fortran.com/fortran-bin/fortran_store/commerce.cgi
8. http://gcc.gnu.org/fortran/
9. http://www.nexen.net/chiffres_cles/phpversion/index.php#evolution.global
10. http://en.wikipedia.org/wiki/PHP#Compilers
11. http://www.codeplex.com/
12. http://www.php.net/manual/es/language.oop5.basic.php
13. http://www.w3schools.com
14. http://www.atlahua.com/tesis
15. http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html
16. http://phplot.sourceforge.net/phplotdocs/SetXScaleType.html
17. http://www.symfony-project.org/
18. Earolugher, R.; Advances in Well Test Analysis,
    Society of Petroleum Engineers, 1977.
19. Economides, M.J.; Petroleum Production Systems,
    Prentice Hall, 1994.
20. Bourdet, D.; Well Test Analysis: The use of advanced interpretation models,
    Elsevier, 2002.
21. Rodriguez N.R; Principios de Mecánica de Yacimientos,
    Facultad de Ingeniería, Universidad Nacional Autónoma de México, 1979.
22. Lee, J.; Rollins, J. B.; Spivey, J. P.: Pressure Transient Testing,
    Society of Petroleum Engineers, 2003.
23. Dake, L.P.; Fundamentals of reservoir engineering,
    Elsevier, Decimoséptima Reimpresión 1998.
24. Rodriguez N.R.; Rodriguez T.M.; Un nuevo modelo educativo basado en el aprendizaje, Proyecto PAPIME PE101707,
    Facultad de Ingeniería, Universidad Nacional Autónoma de México, 2008.
25. Spiegel R.M.; Estadística,
    Mc Graw Hill, Segunda Edición,1991.