



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE QUÍMICA

DESARROLLO DE UNA LIBRERÍA EN JAVA PARA
EL CÁLCULO DE PROPIEDADES DE SUSTANCIAS
CON ECUACIONES DE ESTADO CÚBICAS

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO QUÍMICO

PRESENTA:

HUGO REDON RIVERA



MÉXICO, D.F.

2014

JURADO ASIGNADO:

PRESIDENTE: Profesor:

VOCAL: Profesor:

SECRETARIO: Profesor:

1er. SUPLENTE: Profesor:

2° SUPLENTE: Profesor:

SITIO DONDE SE DESARROLLÓ EL TEMA:

FACULTAD DE QUÍMICA

ASESOR DEL TEMA: DR. ENRIQUE RODOLFO BAZÚA RUEDA.

SUSTENTANTE:HUGO REDON RIVERA.

Índice general

1. Objetivos	6
2. Introducción	7
3. Herramientas	9
3.1. Java	10
3.2. JUnit	11
3.3. Git	12
3.4. GitHub	12
3.5. Maven	12
3.6. Netbeans	12
3.7. Openshift	12
3.8. Wildfly	13
3.9. Licencia ‘GNU GENERAL PUBLIC LICENSE Version 2’	13
3.10. Idioma	13
4. Instalación	16
4.1. Para uso de la librería (compilado)	17
4.2. Para extender o modificar la librería (código fuente)	18

<i>ÍNDICE GENERAL</i>	3
5. Uso de la librería	19
5.1. Sistema de Unidades	21
5.2. Ecuación de estado cúbica	22
5.2.1. Creación de una ecuación de estado cúbica	22
5.2.2. Cálculos con la ecuación cúbica	23
5.3. Compuestos	26
5.4. Parámetros de la ecuación de estado cúbica	27
5.4.1. Compuesto puro	28
5.4.2. Mezcla	30
5.5. Ecuación de capacidad calorífica	34
5.6. Materia homogénea	35
5.6.1. Presión	36
5.6.2. Factor de compresibilidad	37
5.6.3. Volumen molar	38
5.6.4. Fugacidad	39
5.6.5. Entalpía	41
5.6.6. Entropía	44
5.6.7. Energía libre de Gibbs	47
5.7. Materia Heterogénea	49
5.7.1. Temperatura de saturación	51
5.7.2. Presión de saturación	54
5.7.3. Temperatura de Burbuja	56
5.7.4. Temperatura de Rocío	60
5.7.5. Presión de Burbuja	63
5.7.6. Presión de Rocío	66
5.7.7. Flash	68

<i>ÍNDICE GENERAL</i>	4
5.8. Estimación de parámetros	70
5.8.1. Estimación de parámetros de α	71
5.8.2. Estimación de parámetros <i>Binarios</i>	76
6. Página de Internet	79
6.1. Selección de compuestos puros	80
6.2. Creación de sustancias	80
6.3. Creación de Mezclas	81
6.4. Gráficos	83
6.4.1. Presión-Volumen-Temperatura	84
6.4.2. Z-Presión-Temperatura	84
6.4.3. Fugacidad-Presión-Temperatura	85
6.4.4. Temperatura-Entalpía-Presión	85
6.4.5. Temperatura-Entropía-Presión	86
6.4.6. Temperatura-Gibbs-Presión	86
6.4.7. Temperatura-Presión-Volumen	86
6.5. Estimación de parámetros de α	86
6.6. Estimación de parámetros binarios	89
A. Ejemplo de uso con Netbeans	91
A.1. Requisitos	91
A.2. Manualmente	91
A.3. Maven	94
A.4. Código	95
B. ¿Cómo extender o modificar la librería desde el código fuente?	97
C. Solución de la ecuación de estado cúbica	98

<i>ÍNDICE GENERAL</i>	5
D. Ecuaciones	100
D.1. Entalpía	101
D.1.1. Entropía	102
D.1.2. Funciones objetivo para la estimación de parámetros	102
D.1.3. Ecuaciones para el cálculo de la presión de vapor	103
E. Herramientas para la página de internet	104
E.1. Expresiones de α	105

Capítulo 1

Objetivos

- Desarrollar una biblioteca escrita en el lenguaje de java para realizar:
 - El cálculo de propiedades de sustancias puras y mezclas con ecuaciones de estado cúbicas.
 - Cálculos de equilibrio Líquido-Vapor.
 - La estimación de parámetros de expresiones de α para el cálculo de la constante a de la ecuación de estado cúbica.
 - La estimación de parámetros binarios de las reglas de mezclado para el cálculo de las constantes a y b para mezclas.
- La biblioteca a desarrollar deberá ser versátil, robusta, simple de usar y fácilmente expandible.
- Desarrollar una interfaz de usuario que exponga las funciones de la librería.
- Proponer un medio de difusión de la librería, que sea capaz de involucrar a cualquier persona interesada en modificar y extender la librería.

Capítulo 2

Introducción

Esta tesis trata sobre el uso de métodos computacionales para el cálculo de propiedades volumétricas y puntos de equilibrio Líquido-Vapor con ecuaciones de estado cúbicas.

Como resultado se ha escrito una biblioteca de clases en java denominada **Materia**.

El diseño de la biblioteca en conjunto con las herramientas que se describen en la sección 3 hacen de este trabajo una plataforma para el desarrollo de aplicaciones de simulación y modelado de procesos químicos industriales.

El presente trabajo también propone un procedimiento para el desarrollo de futuras aplicaciones. Para lo cual se ha dotado de ciertas características a la biblioteca **Materia** asegurando el funcionamiento del proceso propuesto.

La biblioteca **Materia** se ha escrito para que su utilización sea sencilla y no se requieran grandes conocimientos sobre programación. El capítulo 5 muestra como utilizar la estructura de clases para realizar los cálculos de propiedades y de equilibrio, mostrando pequeños fragmentos de código.

El capítulo ?? muestra como extender la biblioteca de clases, guía en el proceso de creación de nuevas reglas de mezclado, expresiones de α , etc. Este capítulo supone un conocimiento mas avanzado en temas de programación orientada a objetos.

Se ha creado una página de internet para permitir el uso de la librería **Materia** a través de una interfaz de usuario, el capítulo ?? documenta las funciones de la página. También es posible extender las funciones de la página de internet, sin embargo son necesarios conocimientos que estan fuera del alcance de esta tesis. El apéndice E describe las tecnologías utilizadas para la creación de la página de internet, y una breve descripción de su estructura.

Capítulo 3

Herramientas

Las herramientas que se describen en este capítulo en conjunto con la librería **Materia** forman la plataforma de desarrollo propuesta en esta tesis. Estas herramientas dan la versatilidad necesaria para el desarrollo de futuras aplicaciones de modelado y simulación.

Las herramientas resuelven las siguientes necesidades:

- Java nos permite el desarrollo de los métodos computacionales de una forma sencilla y segura.
- JUnit asegura el funcionamiento de la librería sin importar los cambios que sean introducidos al código.
- Git permite administrar los cambios entre versiones de una forma segura.
- GitHub guarda el código fuente y permite compartir los cambios a través de internet.
- Maven permite construir las aplicaciones descargando los proyectos necesarios desde los servidores centrales de Maven.

- Netbeans es el ambiente de desarrollo que permite escribir, compilar y ejecutar las aplicaciones escritas.
- Licencia ‘GNU GENERAL PUBLIC LICENSE Version 2’ asegura que el código fuente sea libre para todos los usuarios.
- Idioma inglés, mantiene una fluidez en la lectura del código fuente.

3.1. Java

James Gosling ¹ define a java como:

Un lenguaje simple, orientado a objetos, distribuido, interpretado, robusto, de arquitectura neutral, portátil, de alto rendimiento, seguro, multiproceso y dinámico.^[4]

Simple: Una de las razones mas importantes por las que se decidió programar la librería en este lenguaje, es que java evita al programador la necesidad de realizar tareas de índole técnico sobre la computadora, como por ejemplo el almacenamiento de los datos en la memoria. Esto permite al programador concentrarse en el area de estudio deseada.

Orientado a objetos: Sin duda la perspectiva orientada a objetos de java es otro gran atractivo para las ciencias e ingenierías. La división y clasificación de las ramas de estudio permiten concentrarnos en todos y cada uno de los aspectos importantes sobre el tema. De igual manera la division de un programa en objetos nos permite concentrarnos en los aspectos y actividades importantes del programa. Tómese

¹Considerado creador del lenguaje Java

como ejemplo la clasificación de la materia en homogénea y heterogénea, el aspecto importante en esta clasificación es la presencia de uno o más estados de agregación en el sistema, se puede pensar así en un cálculo de equilibrio Líquido-Vapor para un sistema heterogéneo, pero no para un sistema homogéneo.

Distribuido: Java es una de las principales herramientas para el desarrollo web. La importancia del desarrollo web radica en su capacidad de difusión masiva. La difusión de un servicio o producto es tan importante como su creación. En conjunto con la librería de funciones se ha creado un sitio web donde se exponen algunas de las funciones mas importantes de la librería.

De arquitectura neutra: Java es multiplataforma, lo cual significa que puede ser ejecutado en la gran mayoría de los sistemas operativos existentes en el mercado actual.

3.2. JUnit

Es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera.

La biblioteca **Materia** ha sido creada con la idea de expandirse y ser modificada ó extendida por cualquier persona interesada en ello, por lo tanto es muy importante mantener una evidencia del funcionamiento de la librería. El uso de la tecnología JUnit es una forma para asegurar que los cambios introducidos al programa no han afectado el funcionamiento esperado de la librería.

Al momento de escribir este trabajo la librería cuenta con mas de 100 pruebas que definen el funcionamiento esperado.

3.3. Git

Git es un sistema de gestión y distribución de código fuente. Permite llevar un registro de los cambios realizados, utilizar las diferentes versiones, y compartir los cambios entre usuarios.

3.4. GitHub

GitHub es un servicio de depósitos de repositorios Git. Es un sitio donde se puede guardar el código fuente, es muy fácil contribuir al proyecto, compartir los cambios propuestos y es accesible a todo el público.

3.5. Maven

Maven es una herramienta de construcción de código, una de sus grandes ventajas es que puede descargar de manera segura versiones compiladas de proyectos con unas pocas líneas de código. En el apéndice A.3 se detalla el proceso de uso.

3.6. Netbeans

NetBeans es un entorno de desarrollo integrado principalmente diseñado para el lenguaje Java, pero también incluye otros lenguajes como, PHP, C / C++, y HTML5.

3.7. Openshift

OpenShift es una plataforma de programación en la nube orientada a servicios de Red Hat. Una versión para la nube privada se llama OpenShift Enterprise. El software

que ejecuta el servicio se encuentra bajo el nombre ‘OpenShift Origin’ de código abierto y está disponible en GitHub.

3.8. Wildfly

WildFly, anteriormente conocido como ‘JavaBeans Open Source Software Application Server’ es un servidor de aplicaciones que implementa la plataforma Java, Enterprise Edition. JBoss está escrito en Java y como tal es multiplataforma: utilizable en cualquier sistema operativo que soporte Java

3.9. Licencia ‘GNU GENERAL PUBLIC LICENSE Version 2’

‘GNU GENERAL PUBLIC LICENSE’ es una licencia libre, sin derechos para software y otro tipo de obras. Pretende garantizar la libertad de compartir y modificar todas las versiones de un programa - para asegurarse de que sigue siendo software libre para todos sus usuarios.

3.10. Idioma

Existen dos razones por la que se ha elegido el idioma inglés para expresar las funciones de la librería. La primer razon y las mas importante, es que java ha sido escrito en inglés y por lo tanto las estructuras de control y palabras reservadas. Pongamos como ejemplo el siguiente fragmento de código.

```
1 public boolean isItADog(Pet pet){
2     if ( pet.getSpeciesName().equals("Canis lupus familiaris")) {
3         return true;
4     }else{
5         return false;
6     }
7 }
```

El fragmento de código pretende conocer si el nombre de la especie de la mascota es el nombre científico “Canis lupus familiaris”, si es así devuelve verdadero, de lo contrario falso. Veamos ahora la versión en español para el mismo fragmento de código.

```
1 public boolean esUnPerro(Mascota mascota){
2     if ( mascota.getNombreDeLaEspecie().equals("Canis lupus familiaris")){
3         return true;
4     }else{
5         return false;
6     }
7 }
```

En inglés, el orden de las palabras denota la diferencia entre una pregunta y una afirmación, de modo que el nombre del método en inglés claramente indica la pregunta “Is it a Dog?” cuando en español la diferencia entre la pregunta y la afirmación debe ser escrita con un signo de interrogación “¿Es un perro?” sin ellos el nombre del método puede parecer la afirmación “¡Es un perro!”, desgraciadamente el signo de afirmación es un operador en java que indica negación, por lo cual no puede ser empleado para definir el nombre de un método.

Nótese el nombre del método “getNombreDeLaEspecie”, el prefijo “get” es una convención en java que significa recuperar, se usa para obtener el valor de la variable que

continúe al prefijo, sin el prefijo estaríamos afectando la funcionalidad de la librería.

Puede apreciarse que la lectura de la línea en inglés es fluida y existe la necesidad de realizar traducciones. Aunque parezca trivial en este ejemplo, en porciones mas grandes de código la diferencia es bastante notable.

Quiero hacer notar que en ningún momento se intenta hacer una comparación sobre los idiomas, sino señalar el beneficio de la fluidez que se obtiene al no mezclarlos.

Capítulo 4

Instalación

Para instalar la biblioteca **Materia** es necesario saber que tipo de trabajo se desea realizar con ella:

- Crear una aplicación que emplea las funciones ya definidas en la biblioteca.
- Crear una nueva funcionalidad de la biblioteca.

Por ejemplo si se desea escribir una aplicación que realice diagramas de presión contra volumen molar usando ecuaciones de estado cúbicas, solo será necesario instalar la forma compilada según la sección 4.1. Ya que las funciones para calcular la presión y el volumen molar con ecuaciones de estado cúbicas existen en la biblioteca , no será necesario modificar el código fuente, por lo tanto no es necesaria la instalación del código fuente. En cambio si se desea utilizar ecuaciones viriales para realizar los diagramas de presión, se deberán realizar las dos instalaciones descritas en este capítulo, ya que las ecuaciones viriales no forman parte del alcance de esta tesis, la librería deberá ser extendida para incluir dichas ecuaciones, una vez hecha la extensión al código fuente , la versión compilada puede ser empleada para realizar la aplicación que realice los diagramas.

4.1. Para uso de la librería (compilado)

Existen dos formas de utilizar la librería Materia en una aplicación java: Descargar el archivo .jar y agregarlo al folder /lib de la aplicación ó desde maven utilizando el archivo pom.xml.

La librería existe como un archivo .jar, se puede descargar desde la página creada para su difusión <http://hugoredon.github.io/Materia/>, o automáticamente desde los servidores de sonatype haciendo uso de maven.

Si se realiza la instalación manual la ubicación de la librería depende de la estructura del proyecto, por ejemplo para una aplicación web los archivos jar deberán ser agregados en el folder dentro del proyecto src/main/webapp/WEB-INF/lib.

Utilizando maven solo deberán agregarse las siguientes lineas de código al archivo pom.xml.

```
1 <dependencies>
2   <dependency>
3     <groupId>com.github.hugoredon</groupId>
4     <artifactId>materia</artifactId>
5     <version>1</version>
6   </dependency>
7 </dependencies>
```

En el apéndice A.2 se ejemplifica la instalación manual y en el apéndice A.3 se muestra la instalación vía maven, para una aplicación de escritorio.

4.2. Para extender o modificar la librería (código fuente)

El código fuente de la librería se expone de manera pública en la página <https://github.com/HugoRedon/Materia>, bajo la licencia GNU GENERAL PUBLIC LICENSE Version 2.

Para poder participar en el proyecto será necesario obtener de manera gratuita una cuenta en github, realizar una copia o clon ¹ de la librería a la nueva cuenta, copiar el código fuente a la computadora haciendo uso de git, realizar los cambios y agregarlos a la cuenta en GitHub ², finalmente hacer una petición para integrar los cambios a la librería original “Pull request”, y si los cambios son aceptados, se habrá logrado la participación al proyecto. El proceso se detalla en el apéndice B.

¹En github a una copia de un proyecto se conoce como “Fork”

²El procedimiento hace uso de los comandos ‘git clone’, ‘git commit’ y ‘git push’, que son explicados con mas detalle en el apéndice ??

Capítulo 5

Uso de la librería

Dada la extensión de la librería, no resulta posible ni deseable mostrar el código fuente completo¹, en este capítulo se presentan pequeños fragmentos de código que muestran la forma de utilizar la librería y su estructura.

La clase ‘Cubic’ realiza cálculos de presión, fugacidad, factor de compresibilidad y adimensionamiento de los parámetros a y b , pero no calcula los parámetros a y b . Para calcular los parámetros a y b existen las clases ‘Substance’ y ‘Mixture’ que definen el cálculo de los parámetros para un compuesto puro o para una mezcla respectivamente.

Las clases ‘Substance’ y ‘Mixture’ tienen en común los métodos ‘calculate_a_cubicParameter’, ‘calculate_b_cubicParameter’, además de otros que se condensan en la clase ‘Homogeneous’. La clase ‘Homogeneous’ utiliza los cálculos de los parámetros de la ecuación cúbica y la clase ‘Cubic’ para finalmente realizar el cálculo de la fugacidad, presión, factor compresibilidad, entalpía, entropía y energía libre de Gibbs.

¹El código completo puede ser consultado y descargado desde la página <https://github.com/HugoRedon/Materia>. El apéndice B guía al lector en el procedimiento de modificación o extensión del código fuente.

La clase ‘Homogeneous’ no realiza cálculos de equilibrio, ya que la clase representa una sola fase. Para realizar cálculos de equilibrio existe la clase ‘Heterogeneous’ que contiene dos fases una líquida y una vapor. A través de los cálculos de fugacidad de cada fase y empleando un algoritmo numérico se pueden realizar los cálculos de equilibrio Líquido-Vapor.

Las secciones del capítulo:

- Sección 5.1 Se definen las unidades que se emplearan durante todo el capítulo.
- Sección 5.2 La clase ‘Cubic’ realiza cálculos de presión, fugacidad, factor de compresibilidad y volumen molar.
- Sección 5.3 La clase ‘Compound’ para definir y utilizar las propiedades del compuesto puro dentro de la librería.
- Sección 5.4 La clase ‘Homogeneous’ y sus implementaciones ‘Substance’ y ‘Mixture’ para calcular los parámetros de la ecuación cúbica.
- Sección 5.6 La clase ‘Homogeneous’ para realizar cálculos de entalpía, entropía y energía libre de Gibbs.
- Sección 5.7 La clase ‘Heterogeneous’ y sus implementaciones ‘HeterogeneousSubstance’ y ‘HeterogeneousMixture’ para realizar cálculos de equilibrio Líquido-Vapor.
- Sección 5.8 Las clases ‘HeterogeneousSubstance’ y ‘HeterogeneousMixture’ realizan la optimización de los parámetros de la expresión de alfa en el cálculo del parámetro a de la ecuación cúbica y los parámetros de interacción binaria de las reglas de mezclado para los parámetros de la ecuación de estado cúbica respectivamente.

5.1. Sistema de Unidades

Las unidades que utiliza la librería se muestran en la tabla 5.1. Durante este escrito se utilizarán las mismas unidades, a menos que se indique lo contrario.

Tabla 5.1: *Sistema de unidades empleado por la librería*

Propiedad	Unidad	
Presión	Pa	Pascal
Temperatura	K	Kelvin
Volumen molar	$\frac{m^3}{kmol}$	Metro cúbico sobre kilomol
Entalpía molar	$\frac{kJ}{mol}$	Joule sobre kilomol
Entropía molar	$\frac{J}{kmolK}$	Joule sobre kilomol Kelvin
Energía libre de Gibbs molar	$\frac{kJ}{mol}$	Joule sobre kilomol

5.2. Ecuación de estado cúbica

La ecuación de estado cúbica representada por la clase ‘Cubic’, permite realizar los cálculos de:

- Presión
- Factor de compresibilidad
- Volumen molar
- Fugacidad

Cualquier ecuación de estado cúbica se puede crear dentro de la librería **Materia** usando los métodos ‘get’ y ‘set’ de la clase ‘Cubic’, ver la sección 5.2.1, pero para comodidad se han definido las ecuaciones de la tabla 5.2 para accederse a través de la clase ‘EquationsOfState’.

Tabla 5.2: *Ecuaciones de estado cúbicas accesibles desde la clase ‘EquationsOfState’*

Ecuación de estado	u	w	Ω_a	Ω_b
Van Der Waals	0	0	0,421875	0,125
Peng robinson	2	-1	0.45723553	0.077796074
Redlich Kwong	1	0	0.42748023	0.08664035
TST	2.5	-1.5	0.470507	0.0740740

5.2.1. Creación de una ecuación de estado cúbica

Existen dos formas de crear una ecuación cúbica dentro de la librería **Materia**.

La primera es a través del método constructor, que se accede con la palabra reservada ‘new’. De esta manera los valores u y w de la ecuación de estado son iguales a 0 y se les puede asignar un valor diferente haciendo uso de los métodos ‘get’ y ‘set’ como se muestra en el código 5.1.

Código 5.1 Creación de la ecuación de estado de Peng Robinson usando los metodos ‘Set’ de los parametros u y w

```
1  Cubic pengRobinson = new Cubic();
2  pengRobinson.setU(2);
3  pengRobinson.setW(-1);
4  pengRobinson.setOmega_a(0.45723553);
5  pengRobinson.setOmega_b(0.077796074);
```

La segunda forma de crear una ecuación de estado es a través de la clase ‘EquationsOfState’. Para obtener una ecuación con los parámetros previamente establecidos, el fragmento de código 5.2 muestra el procedimiento.

Código 5.2 Creación de la ecuación de estado de TST usando la clase ‘EquationsOfState’

```
1  Cubic tst = EquationsOfState.twoSimTassone();
```

5.2.2. Cálculos con la ecuación cúbica

Ya que la clase ‘Cubic’ no realiza los cálculos de sus parámetros estos deben ser proporcionados como argumento de los métodos.

Los métodos de la clase ‘Cubic’ raramente será necesario usarlos directamente, ya que es difícil obtener el cálculo de los parámetros a y b , en cambio la clase ‘Homogeneous’ define el cálculo de los parámetros y de las propiedades que a continuación se muestran.

Presión

El cálculo en el método ‘calculatePressure’ se realiza según la ecuación D.1 y se utiliza como se muestra en el código 5.3.

Código 5.3 *Cálculo de presión con una ecuación de estado cúbica, proporcionando como argumento la temperatura, el volumen y los parámetros a,b*

```
1    double pressure = cubic.calculatePressure(temperature, volume, a, b);
```

Factor de compresibilidad

Es necesario realizar la solución de la ecuación de estado cúbica, como se muestra en el apéndice C, para conocer el factor de compresibilidad. La solución de la ecuación se realiza en el método ‘calculateCompresibilityFactor’.

El método recibe los parámetros adimensionales A, B y la fase a la cual se desea calcular el factor de compresibilidad. La clase “Cubic” tiene los métodos necesarios para transformar los parámetros a y b a su forma adimensional A y B según las ecuaciones D.6.

El código 5.4 muestra como usar el método.

Código 5.4 *Cálculo del factor de compresibilidad con una ecuación de estado cúbica, proporcionando como argumento los parámetros adimensionales A,B, y la fase a la cual se desea calcular el factor*

```
1    double z = cubic.calculateCompresibilityFactor(A,B,phase);
```

Volumen molar

La ecuación D.5 proporciona un método para calcular el volumen molar a partir del factor de compresibilidad. El código 5.5 muestra el método.

Código 5.5 *Cálculo del volumen molar usando una ecuación de estado cúbica, el método recibe los parámetros de temperatura, presión y factor acéntrico*

```
1  double volume = calculateVolume(temperature, pressure, z);
```

Fugacidad

La fugacidad se calcula según la ecuación D.13. Para realizar el cálculo de la fugacidad es necesario conocer las derivadas de los parámetros con respecto a la cantidad de moles del componente i , es decir al componente para el cual se desea realizar el cálculo de fugacidad. El cálculo de la derivada de los parámetros se realiza en las clases que hereden a ‘Homogeneous’, por ejemplo las clases ‘Substance’ y ‘Mixture’. El método se usa como se indica en el código 5.6.

Código 5.6 *Cálculo de fugacidad usando una ecuación de estado cúbica, el método recibe los parámetros de temperatura, presión, los parámetros de la ecuación cúbica a y b , las derivadas con respecto a la cantidad de moles ∂_a , ∂_b y finalmente la fase a la que se desea el cálculo de la fugacidad.*

```
1  double fugacity = calculateFugacity(temperature, pressure,  
2      a, b, partial_a, partial_b, phase);
```

5.3. Compuestos

La clase ‘Compound’ permite definir las propiedades de un compuesto puro y utilizarlas dentro de la librería **Materia** .

Propiedades como la temperatura crítica, presión crítica, factor acéntrico y los parámetros para las expresiones de α pueden obtenerse de esta clase como se muestra en el código 5.7.

Código 5.7 *Se muestra la creación de un objeto tipo ‘Compound’ para definir y utilizar las propiedades del cyclohexano, haciendo uso de los métodos ‘get’ y ‘set’.*

```
1  Compound compound = new Compound("Cyclohexane");
2  compound.setCriticalPressure(4073000);
3  compound.setCriticalTemperature(553.5);
4  compound.setAcentricFactor(0.211);
5  compound.setA_Mathias_Copeman(0.0);
6  ....
7  double acentricFactor = compound.getAcentricFactor();
8  double pc = compound.getCriticalPressure();
9  double tc = compound.getCriticalTemperature();
10 double mathias_A = compound.getA_Mathias_Copeman();
```

5.4. Parámetros de la ecuación de estado cúbica

El cálculo de los parámetros depende de la cantidad de compuestos presentes en el sistema.

- Para los compuestos puros la clase ‘Substance’ define el cálculo de los parámetros usando una expresión de α , los parámetros Ω_a y Ω_b de la ecuación de estado y las propiedades del compuesto puro.
- Para las mezclas, la clase ‘Mixture’ define el cálculo de los parámetros usando una regla de mezclado y el conjunto de los parámetros a y b calculados a partir de los compuestos puros.

Esta estructura se muestra en la figura 5.1

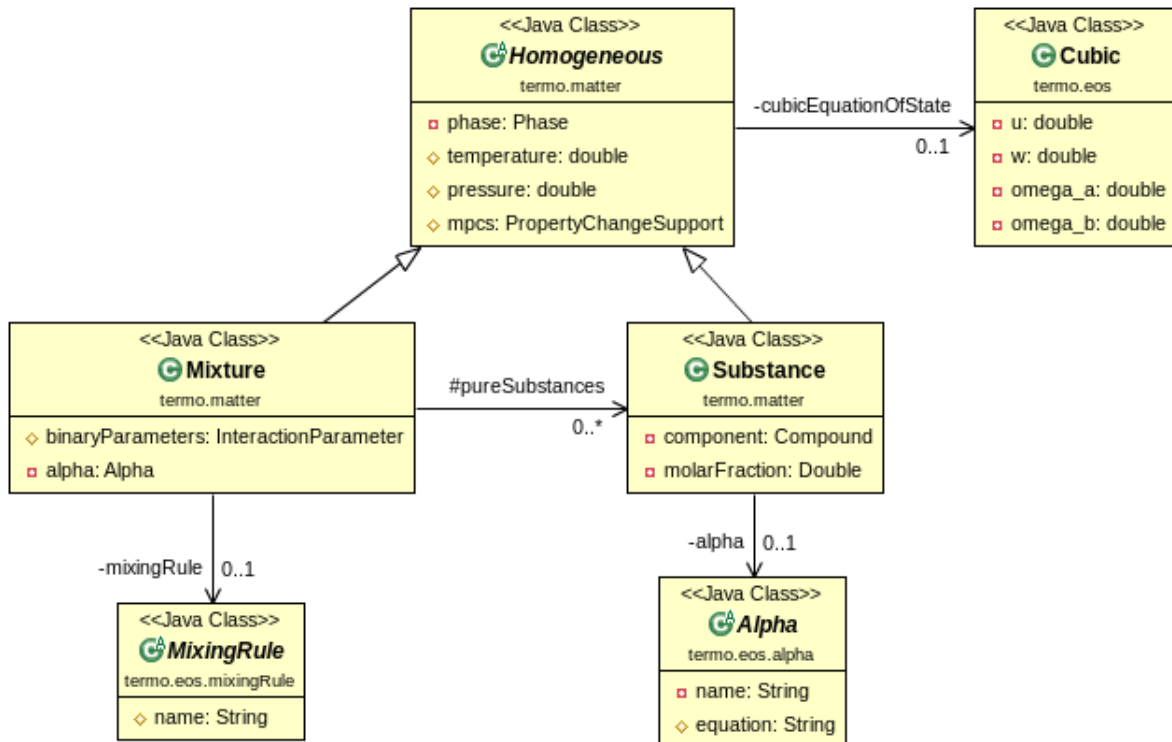


Figura 5.1: Estructura de la librería para el cálculo de propiedades.

Nótese de la figura 5.1 los siguiente:

- Que la clase ‘Homogeneous’ contiene una ecuación del tipo ‘Cubic’.
- Que las clases ‘Substance’ y ‘Mixture’ heredan las propiedades de la clase ‘Homogeneous’, como la ecuación cúbica, y puede hacer uso de ella.

Cualquiera de las dos implementaciones de la clase ‘Homogeneous’ tiene los métodos necesarios para calcular los parámetros de la ecuación cúbica según el código 5.8.

Código 5.8 *Cualquier objeto tipo ‘Homogeneous’ puede calcular los parámetros de la ecuación de estado cúbica a y b*

```
1 Homogeneous homogeneous = ...// Objeto tipo ‘Homogeneous’.  
2 double a = homogeneous.calculate_a_cubicParameter();  
3 double b = homogeneous.calculate_b_cubicParameter();
```

5.4.1. Compuesto puro

Como se muestra en la figura 5.1 la clase ‘Substance’ contiene una expresión de α , y una ecuación de estado, necesarias para realizar el cálculo de los parámetros, según las ecuaciones D.2 y D.3.

La expresión de α puede ser una función de la temperatura, las expresiones implementadas en el presente trabajo se muestran en la tabla 5.3.

Creación de un objeto tipo ‘Substance’

Es necesario utilizar el método constructor de la clase ‘Substance’ para crear un objeto de este tipo, se debe proporcionar como parámetros la ecuación de estado deseada, la

Tabla 5.3: Expresiones de α disponibles en la librería

Expresión	Parámetros	Ecuación de estado
Soave	—	PR
Peng and Robinson	—	PR
Mathias	A	SRK
Stryjek and Vera	k_1	PRSV
Adachi and Lu	A, B	SRK,PR
Soave	A, B	SRK,PR
Melhem, et al.	A, B	SRK,PR
Androulakis et al.	A, B, C	SRK,PR
Mathias and Copeman	A, B, C	SRK,PR
Yu and Lu	A, B, C	SRK,PR
Stryjek and Vera	A, B, C	PR
Twu	L, M, N	TST
Twu	—	TST,PR
GCEOS	—	(Cualquier u y w)

expresión de α , una instancia de la clase ‘Compound’ como se muestra en la sección 5.3 y la fase de la sustancia.

En la sección 5.2.1 se mostró como crear una ecuación de estado cúbica previamente definida, a través de la clase ‘EquationOfState’. Los valores de Ω_a y Ω_b se muestran en la tabla 5.2. De forma muy similar al de las ecuaciones de estado, la selección de la expresión de α se realiza a través de la clase ‘Alphas’.

El código 5.9 muestra como crear un objeto del tipo sustancia.

Código 5.9 Creación de un objeto tipo ‘Substance’ para el compuesto Ciclohexano, con la ecuación de estado Soave Redlich Kwong y la expresión de α de mathias

```

1
2  Compound compound = new Compound("Cyclohexane");
3  compound.setCriticalPressure(4073000);
4  compound.setCriticalTemperature(553.5);
5  compound.setAcentricFactor(0.211);
6
7
```

```
8 Cubic srk = EquationsOfState.redlichKwongSoave();
9 Alpha mathias = Alphas.getMathiasExpression();
10
11
12 Substance substance = new Substance(srk, mathias, compound, Phase.LIQUID);
```

Finalmente se utiliza el objeto creado para realizar el cálculo de los parámetros de la ecuación de estado cúbica, en el código 5.10.

Código 5.10 *Cálculo de los parámetros para la ecuación de estado cúbica con la clase ‘Substance’.*

```
1 double a = substance.calculate_a_cubicParameter();
2 double b = substance.calculate_b_cubicParameter();
```

Nótese que el código 5.8 y el código 5.10 es idéntico y aunque parece redundante, solo se muestra para señalar el polimorfismo de la librería, es decir que el objeto substance es del tipo ‘Homogeneous’ además del tipo ‘Substance’, y tiene acceso a los métodos en las dos clases.

5.4.2. Mezcla

El cálculo de los parámetros para una mezcla depende de la regla de mezclado, y en el caso de las reglas de mezclado basadas en la energía libre de exceso, también del modelo de actividad.

Las reglas de mezclado incluidas en este trabajo se muestran en la tabla 5.4, y los modelos de actividad se listan en la tabla 5.5.

Tabla 5.4: Reglas de mezclado implementadas

Regla	Parámetros	
Van Der Waals	k_{ij}	$k_{ij} = k_{ji}$
Mathias-Klotz-Prausnitz	k_{ij}	$k_{ij} \neq k_{ji}$
Huron Vidal	Según el model de actividad	
Wong Sandler	k_{ij} + los parámetros del modelo de actividad	$k_{ij} = k_{ji}$

Tabla 5.5: Modelos de actividad

Modelo de actividad	Parámetros	
Wilson	a_{ij}, b_{ij}	
NRTL	$a_{ij}, b_{ij}, \alpha_{ij}$	$\alpha_{ij} = \alpha_{ji}$

Creación de un objeto tipo ‘Mixture’

Un objeto tipo ‘Mixture’ contiene dentro de sí un conjunto de objetos tipo ‘Substance’, esto hace que su creación sea mas compleja. Se ha escrito la clase ‘MixtureBuilder’ para facilitar la creación de los objetos de la clase ‘Mixture’.

La clase ‘MixtureBuilder’ facilita la creación de los objetos ‘Substance’ que forman el conjunto de compuestos de la mezcla, y permite asignar una expresión de α diferente para cada compuesto.

El código 5.11 muestra la creación de una mezcla, donde la expresión de α es la misma para cada compuesto puro.

Código 5.11 Creación de una mezcla con la clase ‘MixtureBuilder’ asignando la misma expresión de α para cada compuesto puro.

```

1
2 Compound cyclohexane = new Compound("Cyclohexane");
3 cyclohexane.setCriticalPressure(4073000);
4 cyclohexane.setCriticalTemperature(553.5);
5 cyclohexane.setAcentricFactor(0.211);
6

```

```
7  Compound pentane = new Compound("N-pentane");
8  pentane.setCriticalPressure(3370000);
9  pentane.setCriticalTemperature(469.7);
10 pentane.setAcentricFactor(0.251);
11
12 Cubic equationOfState = EquationsOfState.pengRobinson();
13 Alpha alpha = Alphas.getMathiasAndCopemanExpression();
14
15 Mixture mixture = new MixtureBuilder()
16     .addCompounds(cyclohexane,pentane)
17     .setAlpha(alpha)
18     .setEquationOfState(equationOfState)
19     .setPhase(Phase.VAPOR)
20     .build();
```

Es posible crear la mezcla con una expresión de α diferente para cada compuesto puro según el código 5.12.

Código 5.12 Código para el cálculo de los parámetros de la ecuación de estado en una mezcla, con diferentes expresiones de α

```
1  Mixture mixture = new MixtureBuilder()
2      .addCompound(cyclohexane,Alphas.getPengAndRobinsonExpression())
3      .addCompound(pentane,Alphas.getStryjekAndVeraExpression())
4      .setEquationOfState(eos)
5      .setPhase(phase)
6      .setMixingRule(mixingRule)
7      .setInteractionParameter(k)
8      .build();
```

Finalmente se pueden calcular los parámetros de la ecuación cúbica con el código 5.13.

Código 5.13 *Cálculo de parámetros de la mezcla*

```
1  double a = mixture.calculate_a_cubicParameter();  
2  double b = mixture.calculate_b_cubicParameter();
```

5.5. Ecuación de capacidad calorífica

Los cálculos de entalpía y entropía necesitan de una ecuación de capacidad calorífica.

La librería **Materia** define una interface para obligar que todas las implementaciones de la ecuación tengan los métodos para calcular la entalpía y entropía del gas ideal según la ecuaciones D.8, D.11.

Incluidas para el presente trabajo la librería **Materia** implementa las ecuaciones 107 del DIPPR y una ecuación Polinomial.

Predeterminadamente se usa la ecuación la ecuación 107 de DIPPR.

Faltan
las refe-
rencias a
las ecua-
ciones
del cp

5.6. Materia homogénea

La clase ‘Homogeneous’ representa a las sustancias o mezclas que se encuentran en una sola fase, las propiedades que se pueden calcular de una fase son:

- Presión
- Factor compresibilidad
- Volumen molar
- Fugacidad
- Entalpía
- Entropía
- Energía libre de Gibbs.

5.6.1. Presión

Un cálculo de presión, se realiza como se muestra en el código 5.14.

Código 5.14 *Cálculo de presión para un objeto tipo homogeneous*

```
1  double pressure = homogeneous.calculatePressure(temperature, volume);
```

Como ya se mencionó antes el objeto ‘Homogeneous’ del código 5.14 puede ser del tipo ‘Substance’ o ‘Mixture’.

En la figura 5.2 se muestra un ejemplo de uso del cálculo para realizar gráficas de presión.

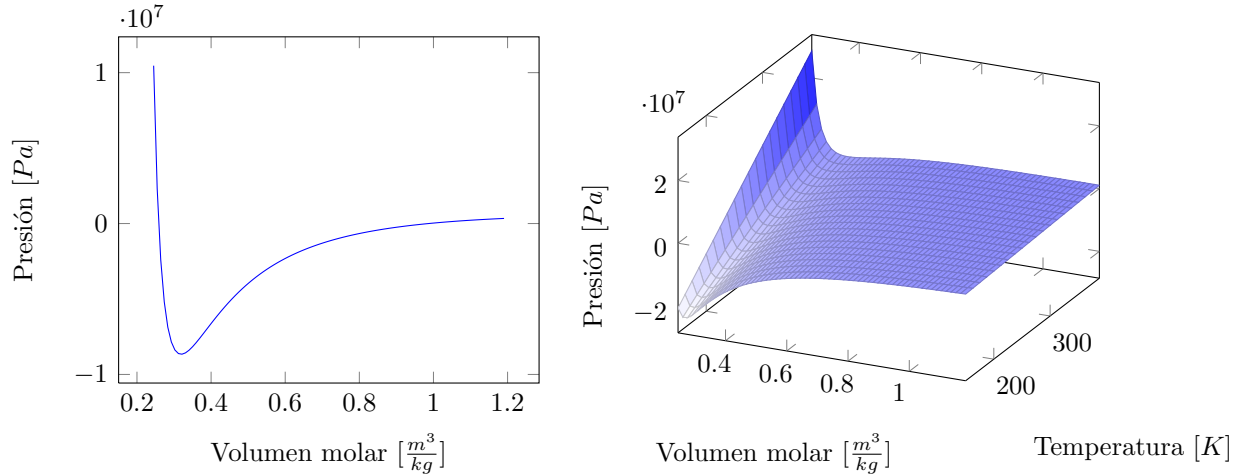


Figura 5.2: *Diagramas de presión para el heptano usando la eq. de Van Der Waals ($u=w=0$)*

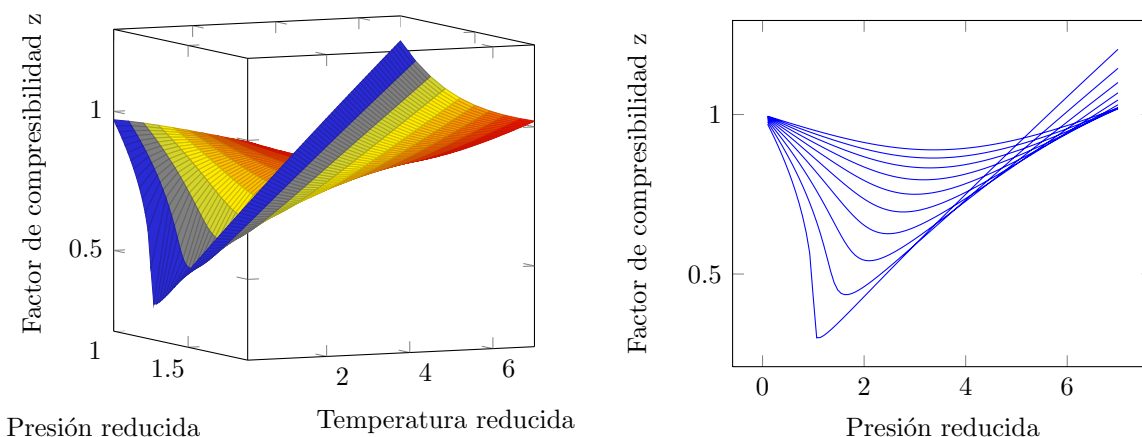


Figura 5.3: Diagramas del factor de compresibilidad con la ecuación de estado de Van Der Waals para el heptano

5.6.2. Factor de compresibilidad

En el fragmento de código 5.15 se muestra el cálculo de el factor de compresibilidad para una sustancia o mezcla homogénea.

Código 5.15 Cálculo del factor de compresibilidad, y adimensionamiento de los parámetros a y b con la clase “Cubic”

```

1 substance.setPressure(pressure);
2 substance.setTemperature(temperature);
3 double z = substance.calculateCompressibilityFactor();

```

Para un rango de presión y a diferentes temperaturas podemos formar los diagramas de la figura 5.3.

5.6.3. Volumen molar

El código 5.16 muestra como calcular el volumen molar para una substancia o mezcla homogénea.

Código 5.16 *Cálculo del factor de double volume = cubic.*

```
1  double volume = calculateVolume(temperature, pressure, z);
```

En la figura 5.4 se muestra la relación que tiene el volumen con el factor de compresibilidad a diferentes temperaturas y presiones.

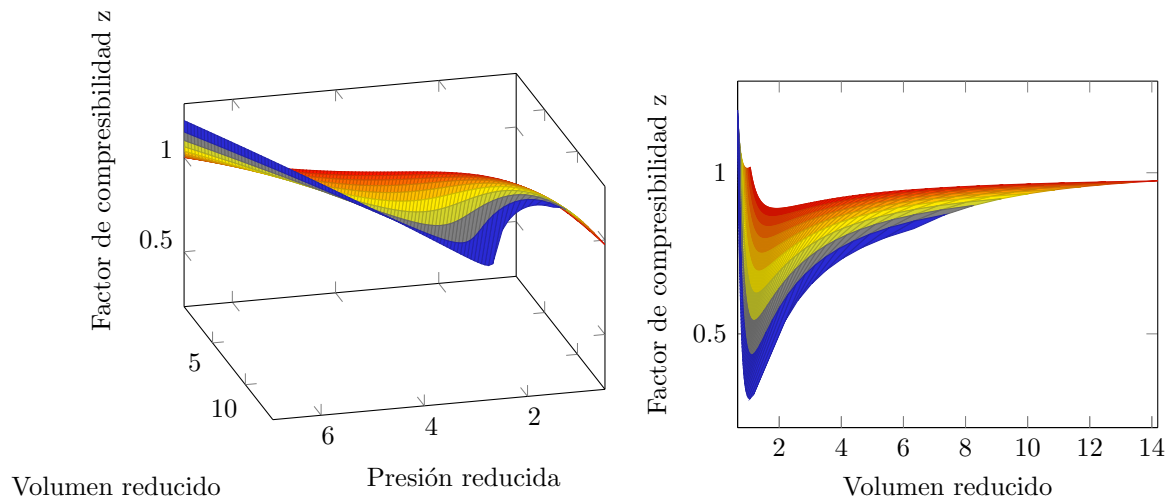


Figura 5.4: *Relación entre el factor de compresibilidad y el volumen*

5.6.4. Fugacidad

El cálculo de fugacidad se realiza para un compuesto en específico. Si el cálculo se desea hacer para una mezcla homogénea, será necesario especificar el compuesto para el cual se desea la fugacidad. Si el cálculo es para una sustancia no será necesario especificar el compuesto, ya que la sustancia solo tiene un compuesto puro. En el código 5.17 se realiza el cálculo de fugacidad para una sustancia, y en el código 5.18 se realiza el cálculo de fugacidad para un compuesto en una mezcla.

Código 5.17 *Cálculo de la fugacidad para una sustancia homogénea.*

```
1  double fugacity = substance.calculateFugacity();
```

Código 5.18 *Cálculo de la fugacidad para un compuesto en una mezcla.*

```
1  double fugacity = mixture.calculateFugacity(compound);
```

Para crear la figura 5.5 se calculó la fugacidad de una sustancia en su fase vapor, y después en su fase líquida, los planos se cruzan en la línea de equilibrio Líquido-Vapor.

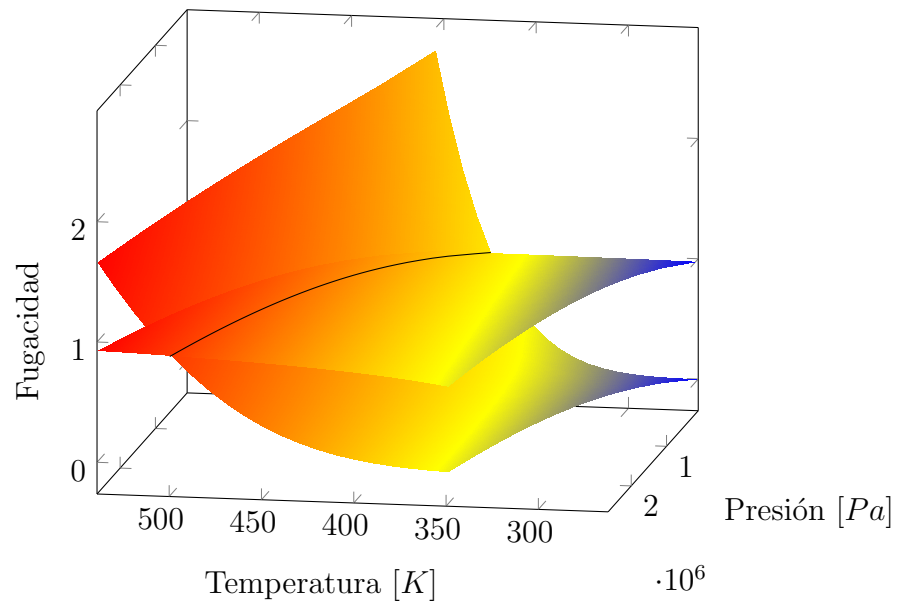


Figura 5.5: La fugacidad del líquido y del vapor coinciden en la línea de equilibrio.

5.6.5. Entalpía

Para conocer la entalpía real de una substancia o mezcla, es necesario conocer la entalpía del gas ideal y la entalpía residual.

Entalpía del gas ideal con la ecuación del calor específico

Para realizar el cálculo de la entalpía segun el gas ideal, es necesaria una ecuación que represente el calor específico. El cálculo depende de la forma de la ecuación del calor específico y sus parámetros. En la sección 5.5 se muestran las ecuaciones de c_p incluidas en la librería y como usarlas.

Con la ecuación del c_p la clase ‘Homogeneous’ puede calcular la entalpía del gas ideal como se muestra en el código 5.19.

Código 5.19 *Cálculo de la entalpía del gas ideal.*

```
1  double idealGasEnthalpy = homogeneous.calculateIdealGasEnthalpy();
```

Entalpía real

La entalpía residual no esta separada en un método particular dentro de la librería **Materia** , sino que esta incluido en el cálculo de la entalpía real. Es muy facil realizar la separación, pero para los objetivos de la presente tesis no fue necesario realizarlo.

El cálculo de la entalpía se realiza en el método ‘calculateEnthalpy()’ como se muestra en el fragmento de código 5.20 según la ecuación D.9.

Código 5.20 *Cálculo de la entalpía real*

```
1  double enthalpy = homogeneous.calculateEnthapy();
```

Las figuras 5.6 y 5.7 muestran diagramas de entalpía creados con ayuda de la librería **Materia** .

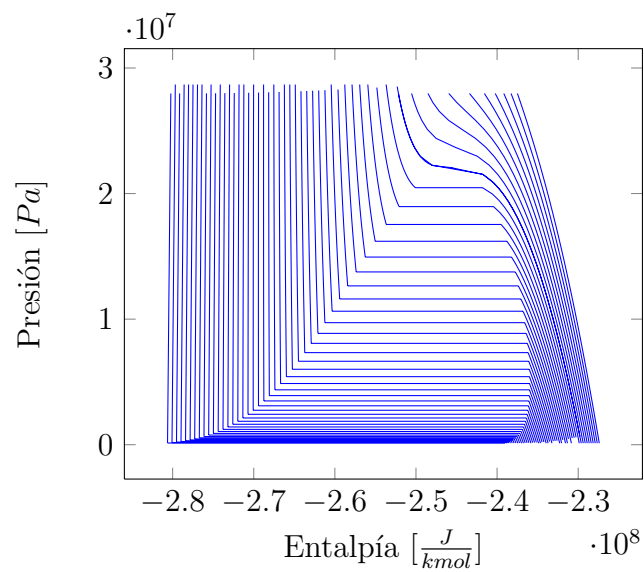


Figura 5.6: *Diagrama de entalpía para el agua.*

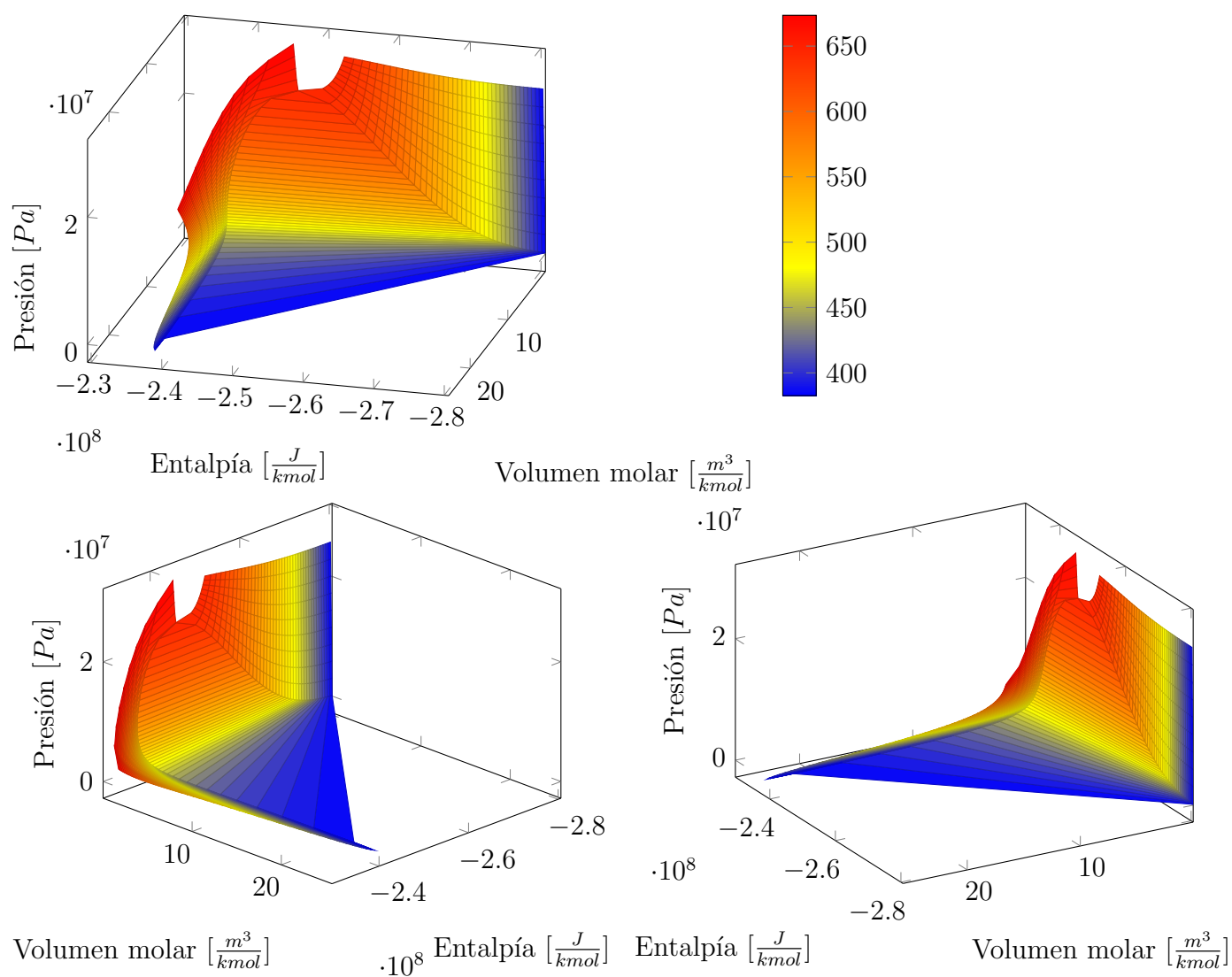


Figura 5.7: Diagramas tridimensionales presión-entalpía-‘volumen molar’ del agua.

5.6.6. Entropía

El cálculo de la entropía es muy semejante al de la entalpía, es necesario conocer la entropía del gas ideal y la entropía residual.

Entropía del gas ideal con la ecuación del calor específico

El cálculo de la entropía segun el gas ideal, también necesita una ecuación que represente el calor específico. En la sección 5.5 se muestran las ecuaciones de cp incluidas en la librería y como usarlas.

En el fragmento de código 5.21 se muestra el cálculo de la entropía del gas ideal según la ecuación D.11.

Código 5.21 *Cálculo de la entropía absoluta del gas ideal.*

```
1  double idealGasEntropy = homogeneous.calculateIdealGasEntropy();
```

Entropía real

El cálculo de la entalpía se realiza según la ecuación D.10 y su uso se muestra en el código 5.22.

Código 5.22 *Cálculo de la entropía absoluta.*

```
1  double entropy = homogeneous.calculateEntropy();
```

Las figuras 5.8 y 5.9 muestran diagramas de entalpía creados con ayuda de la librería **Materia** .

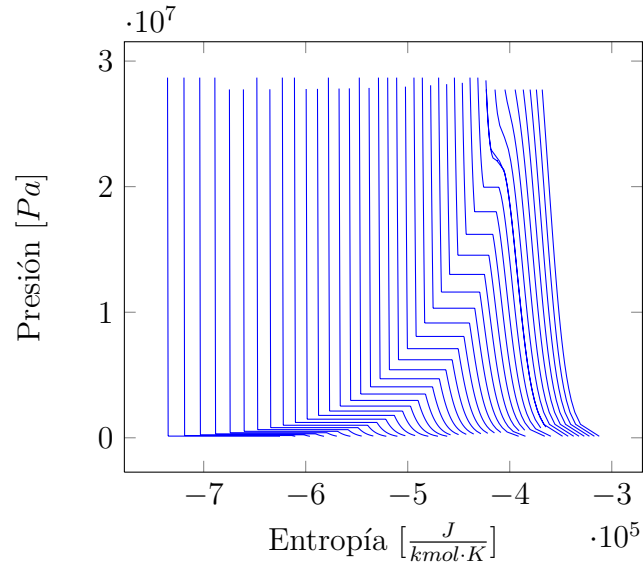


Figura 5.8: *Diagrama de presión-entropía para el agua.*

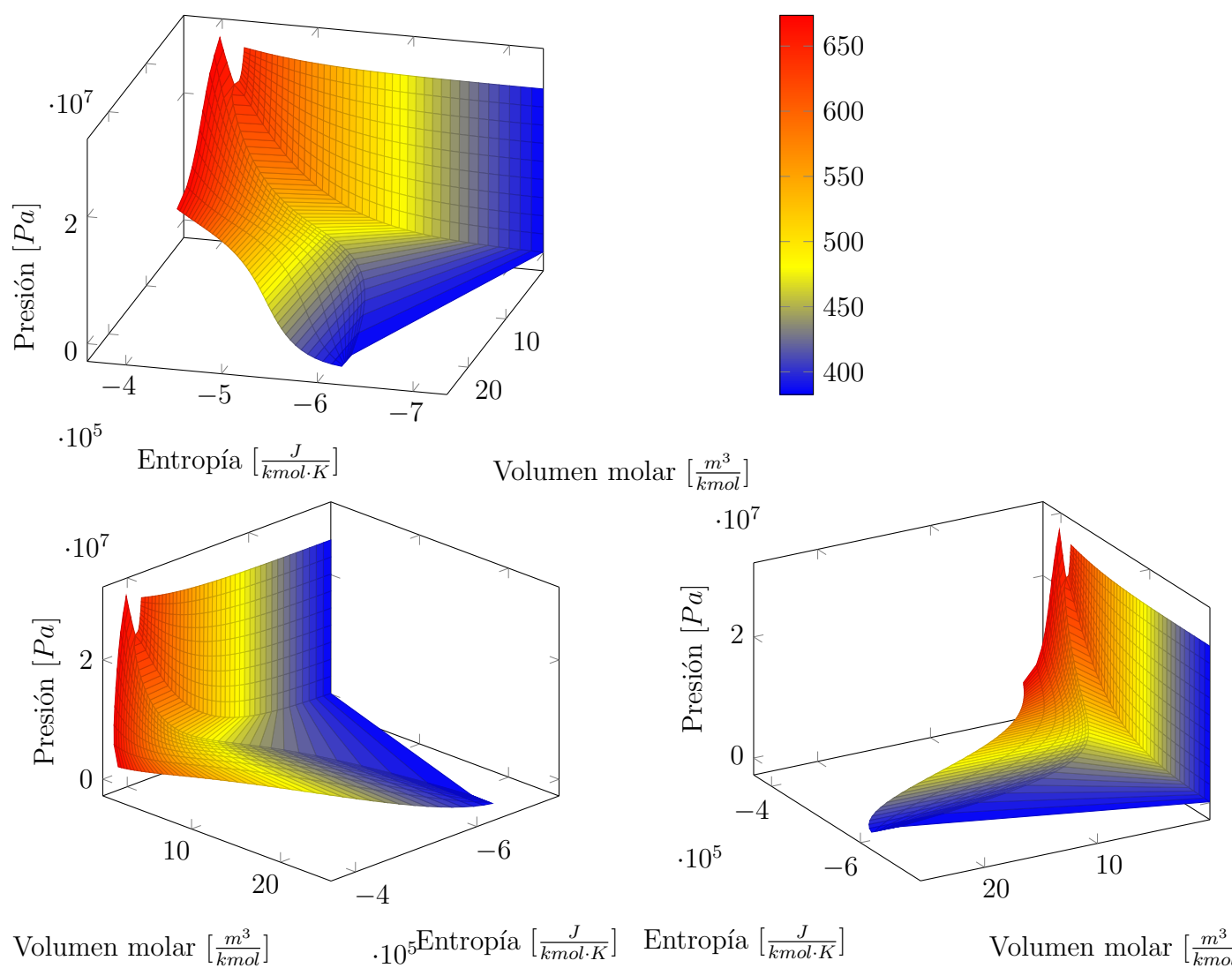


Figura 5.9: Diagramas tridimensionales presión-entropía-‘volumen molar’ para el agua.

5.6.7. Energía libre de Gibbs

El cálculo de la energía libre de Gibbs se realiza a partir de la entropía y la entalpía según la ecuación D.12 y su cálculo se demuestra en el código 5.23.

Código 5.23 *Cálculo de la energía libre de Gibbs con la librería **Materia***

```
1      double gibbs = homogeneous.calculateGibbs();
```

Las figuras 5.10 y 5.11 muestran diagramas de entalpía creados con ayuda de la librería **Materia**.

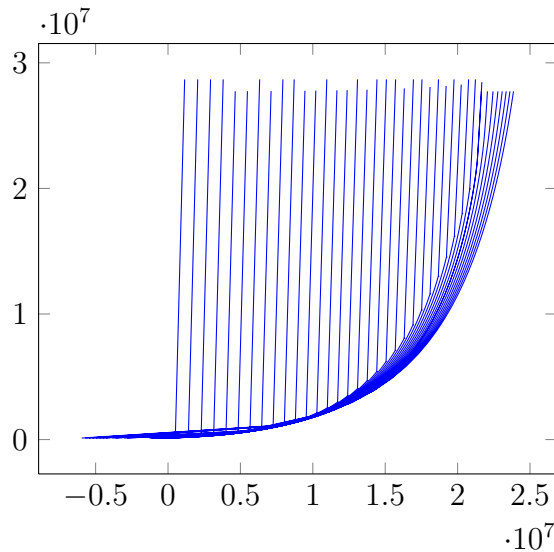


Figura 5.10: *Diagrama de la energía libre de Gibbs para el agua.*

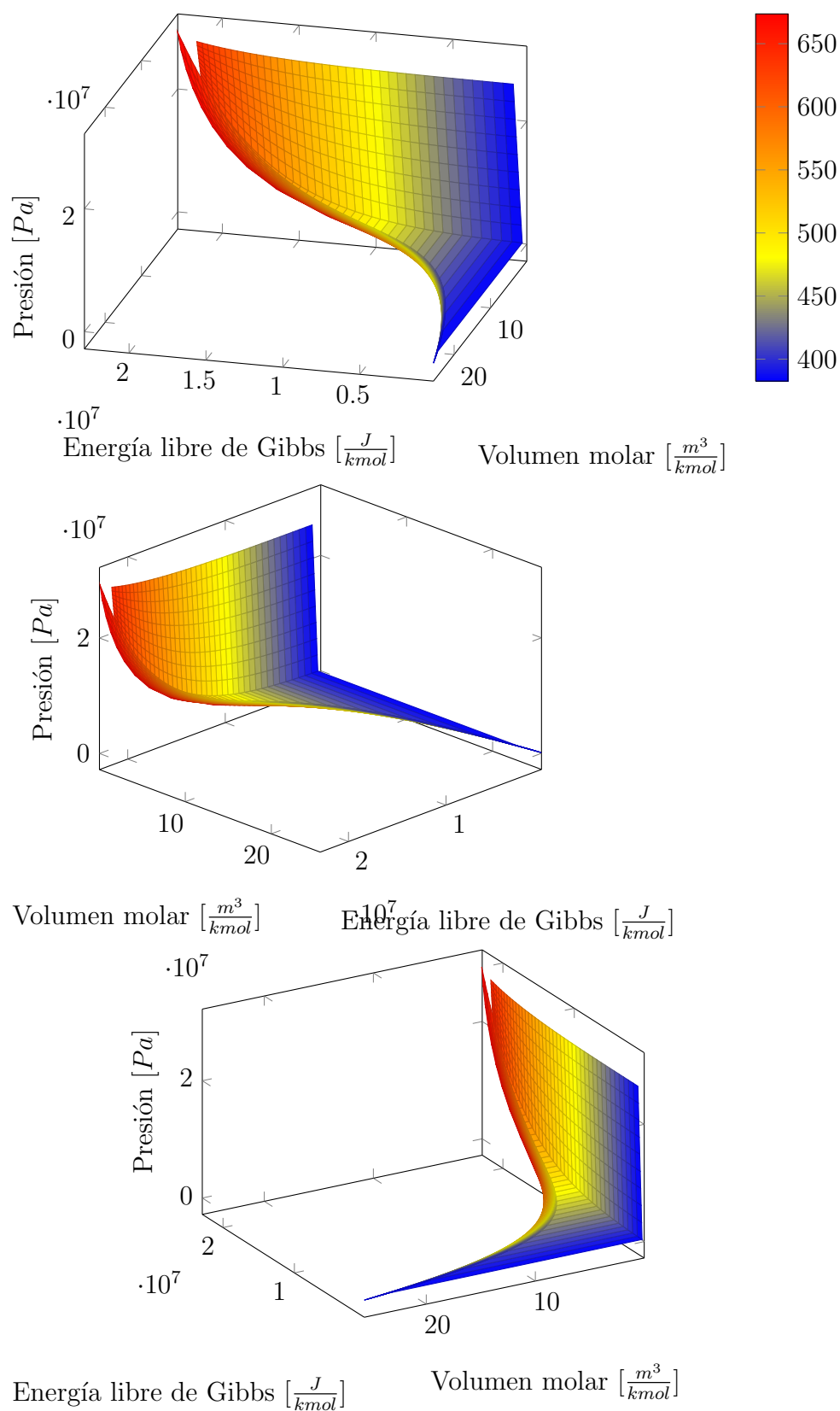


Figura 5.11: Diagramas tridimensionales de presión-‘energía libre de Gibbs’-‘volumen molar’ para el agua.

5.7. Materia Heterogénea

La clase ‘Heterogeneous’ se encarga de los cálculos de equilibrio entre las fases líquido y vapor. La clase contiene dos instancias de la clase ‘Homogeneous’ que representan a la fase líquida y a la fase vapor. La clase implementa los algoritmos numéricos para igualar los cálculos de la fugacidad entre las fases.

Los cálculos de equilibrio que se implementan en este trabajo son:

- Substancias
 - Temperatura de saturación.
 - Presión de saturación.
- Mezclas
 - Presión de burbuja, sección 5.7.5.
 - Presión de rocío, sección 5.7.6.
 - Temperatura de burbuja, sección 5.7.3.
 - Temperatura de rocío, sección 5.7.4.
 - Flash temperatura-presión 5.7.7.

Los algoritmos son muy similares, las principales diferencias consisten en la función objetivo.

La clase ‘HeterogeneousSubstance’ realiza los cálculos de equilibrio para las sustancias y la clase ‘HeterogeneousSubstance’ realiza los cálculos de equilibrio para las mezclas, en la figura 5.12 se muestra la estructura.

Los cálculos de la presente sección se realizan de forma muy similar, primero indicando la variable conocida con el método ‘set’, después invocando el método que realiza el

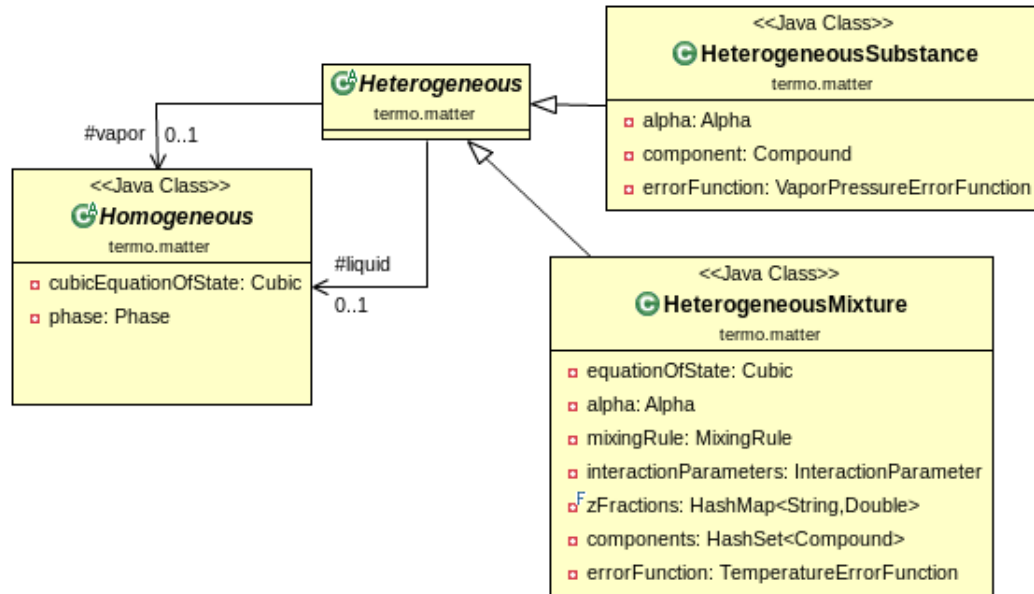


Figura 5.12: Estructura de la librería para el cálculo de equilibrio Líquido-Vapor.

algoritmo para conocer la incognita, el método puede recibir o no un estimado inicial y finalmente obtener el resultado de la variable con el método ‘get’.

Cada sección muestra el uso de la librería con fragmentos de código que realizan el cálculo con y sin el estimado inicial. También se indica que método se usa para estimar la variable incognita en caso de no proporcionar un estimado inicial.

5.7.1. Temperatura de saturación

El cálculo de temperatura de saturación se realiza como se indica en la figura 5.13. Usando un objeto del tipo ‘HeterogeneousSubstance’ de la librería **Materia** el cálculo se realiza como se muestra en los fragmentos de código 5.25 y 5.24.

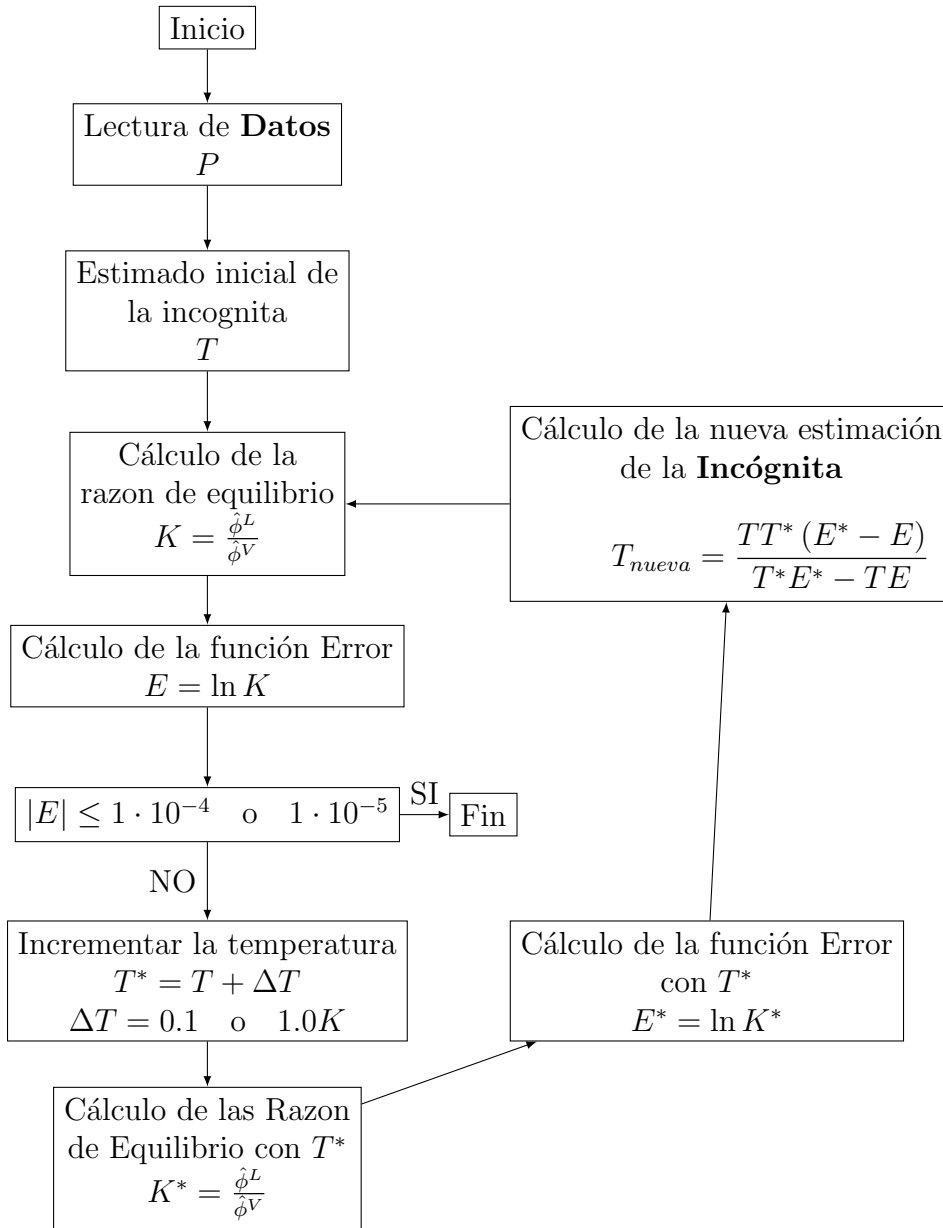
Si al método no se le proporciona un estimado inicial, como se muestra en el código 5.25 entonces la clase realiza la estimación como se muestra en el diagrama 5.14.

Código 5.24 *Cálculo de la temperatura de saturación proporcionando un estimado inicial.*

```
1      heterogeneousSubstance.setPressure(pressure);
2      heterogeneousSubstance.saturationTemperature(temperatureEstimate);
3      double temperature = heterogeneousSubstance.getTemperature();
```

Código 5.25 *Cálculo de la temperatura de saturación.*

```
1      heterogeneousSubstance.setPressure(pressure);
2      heterogeneousSubstance.saturationTemperature();
3      double temperature = heterogeneousSubstance.getTemperature();
```

**Figura 5.13:** Algoritmo de temperatura de saturación.

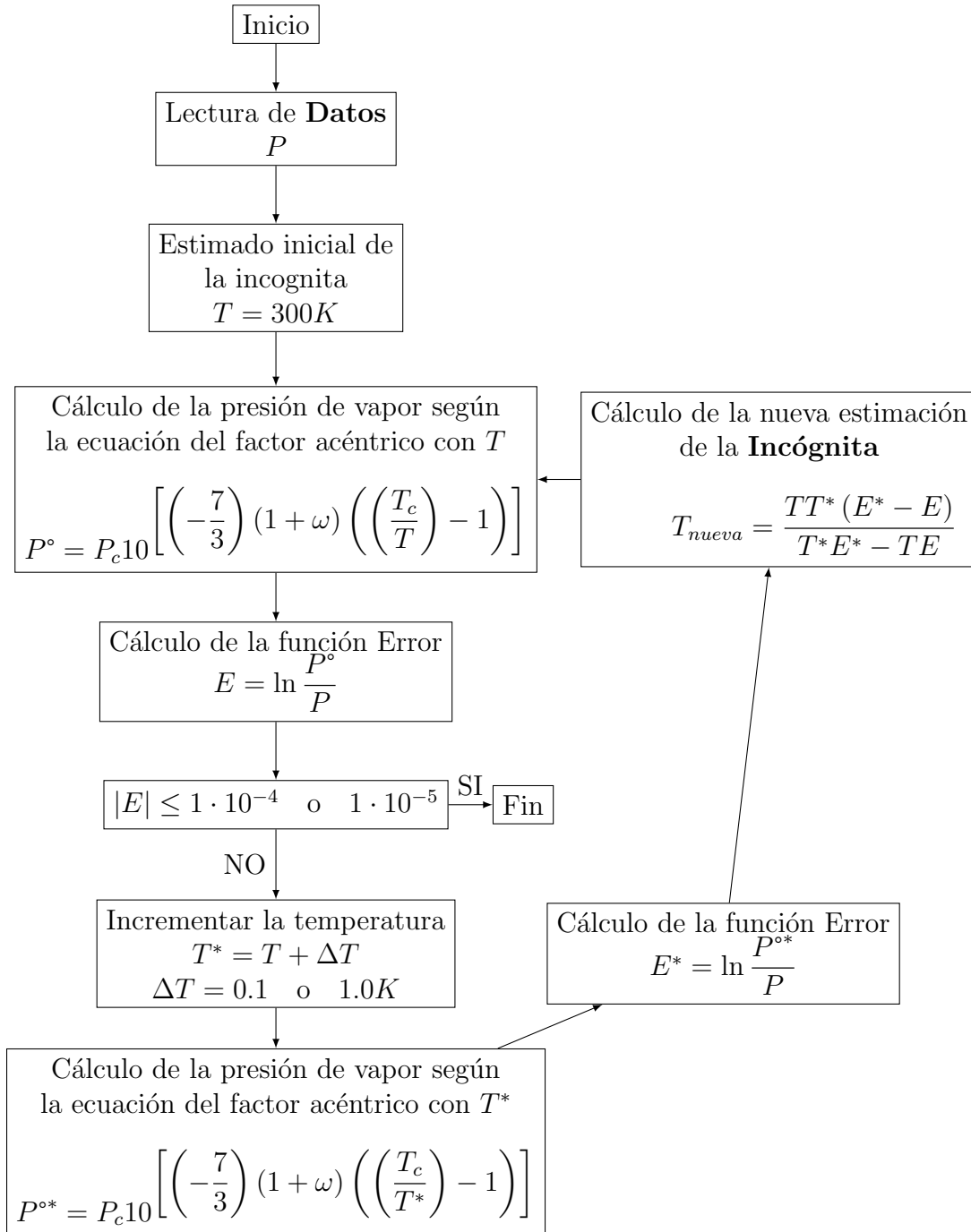


Figura 5.14: Algoritmo de estimación de temperatura de saturación usando la ecuación del factor acéntrico.

5.7.2. Presión de saturación

El cálculo de presión de saturación se realiza como se indica en la figura 5.15. Usando un objeto del tipo ‘HeterogeneousSubstance’ de la librería **Materia** el cálculo se realiza como se muestra en los fragmentos de código 5.27 y 5.26.

Si al método no se le proporciona un estimado inicial, como se muestra en el código 5.27 entonces la clase realiza la estimación de la presión de vapor con la ecuación del factor acéntrico D.14.

Código 5.26 *Cálculo de la presión de saturación proporcionando un estimado inicial.*

```
1      heterogeneousSubstance.setTemperature(temperature);
2      heterogeneousSubstance.saturationPressure(pressureEstimate);
3      double pressure = heterogeneousSubstance.getPressure();
```

Código 5.27 *Cálculo de la presión de saturación.*

```
1      heterogeneousSubstance.setTemperature(temperature);
2      heterogeneousSubstance.saturationPressure();
3      double pressure = heterogeneousSubstance.getPressure();
```

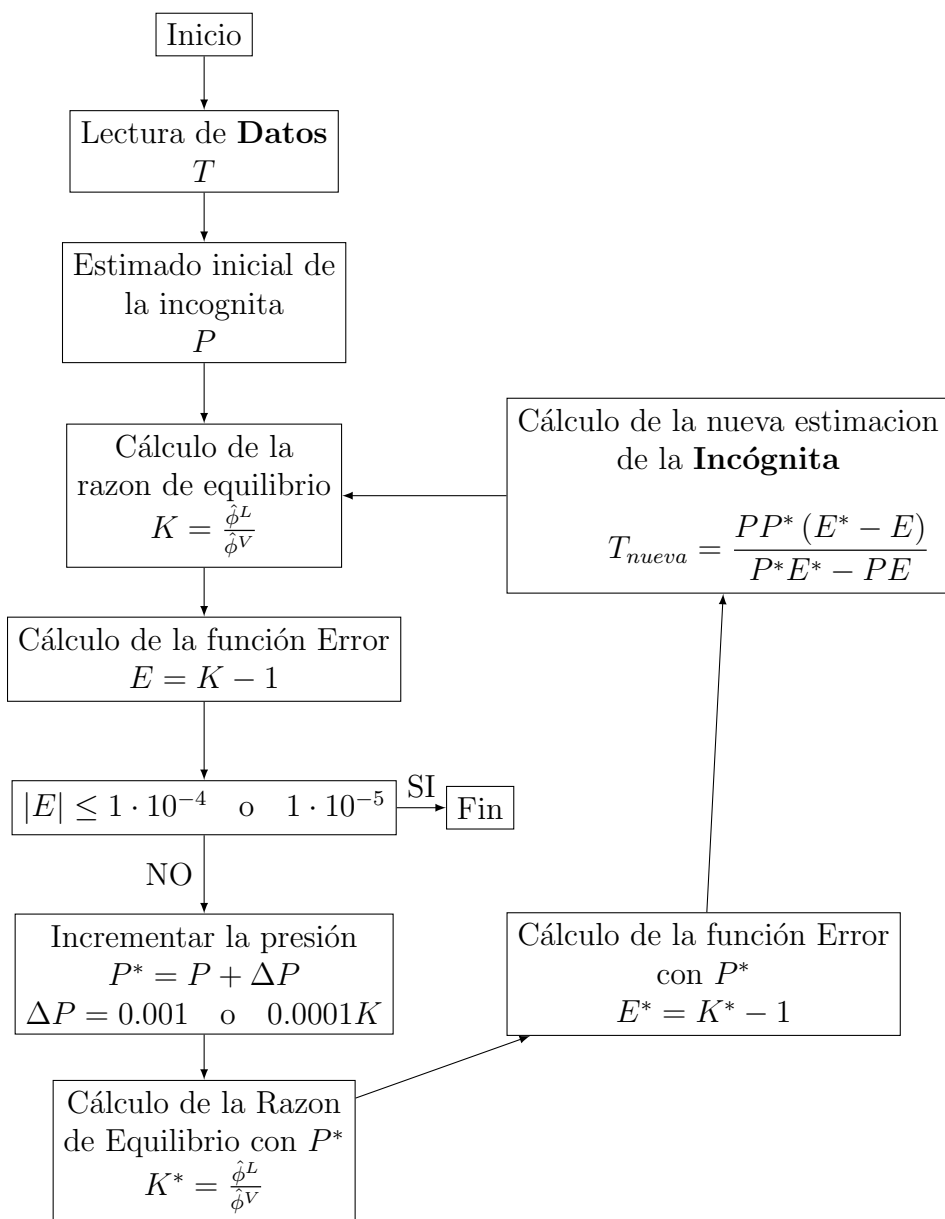


Figura 5.15: Algoritmo para el cálculo de presión de saturación.

5.7.3. Temperatura de Burbuja

El cálculo de temperatura de burbuja para una mezcla se realiza como se indica en la figura 5.17. Usando un objeto del tipo ‘HeterogeneousMixture’ de la librería **Materia** el cálculo se realiza como se muestra en los fragmentos de código 5.29 y 5.28.

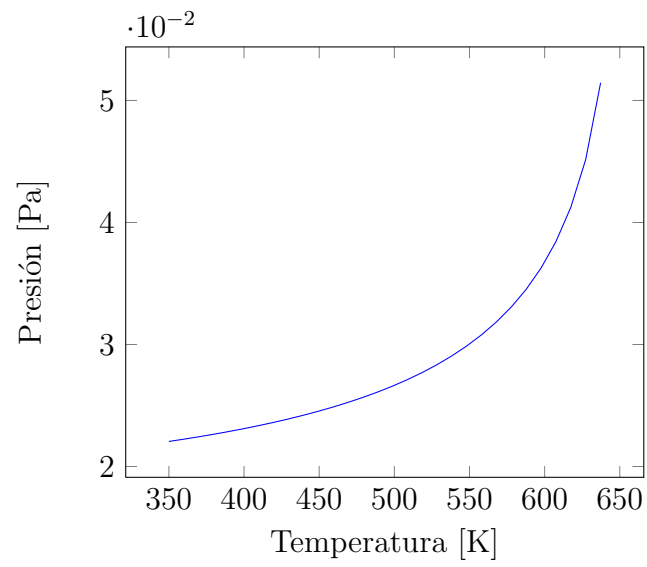
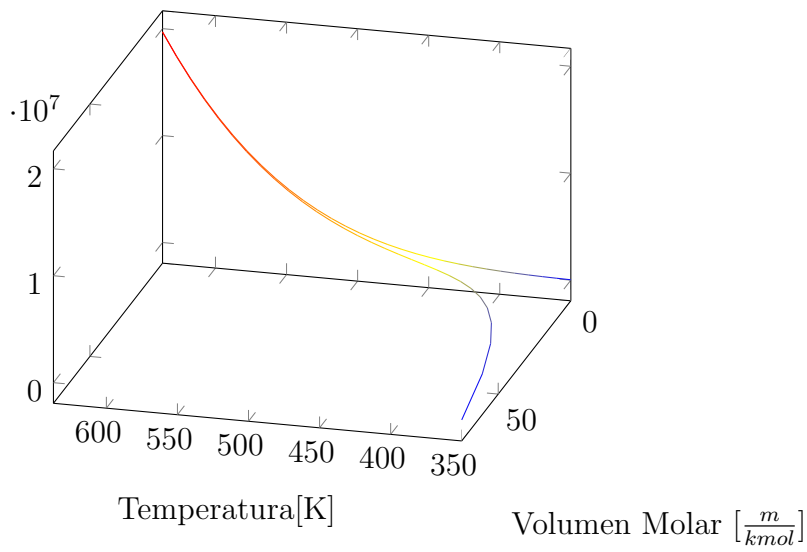
Si al método no se le proporciona un estimado inicial, como se muestra en el código 5.29 entonces la clase realiza la estimación de la temperatura como se muestra en la figura 5.18.

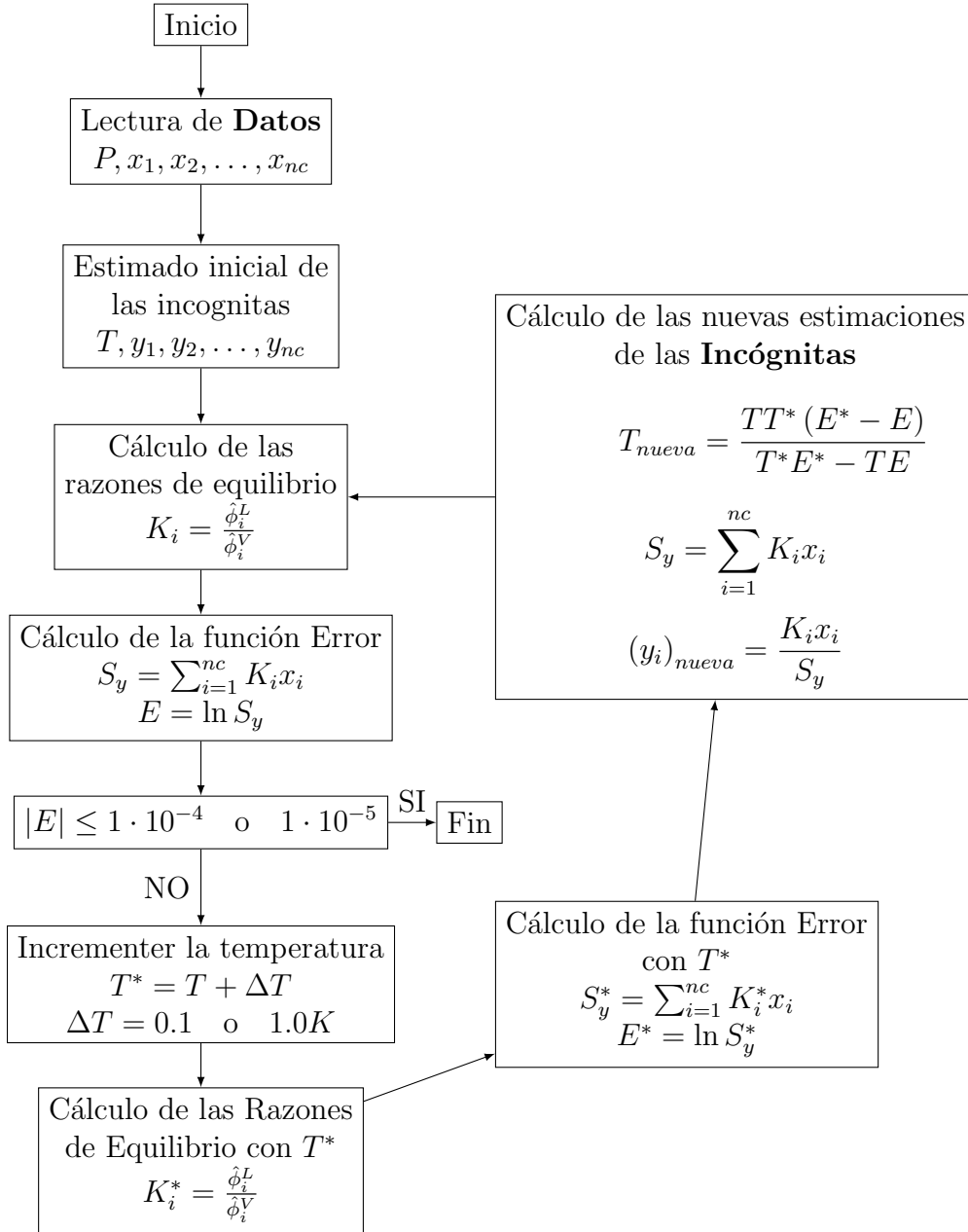
Código 5.28 *Cálculo de la temperatura de burbuja proporcionando un estimado inicial.*

```
1
2     heterogeneousMixture.setZFraction(compound1,molarFraction1);
3     //..... asignar la fracción molar para todos los compuestos
4
5     heterogeneousMixture.setPressure(pressure);
6     heterogeneousMixture.bubbleTemperature(temeperatureEstimate);
7     double temperature = heterogeneousMixture.getTemperature();
```

Código 5.29 *Cálculo de la temperatura de burbuja.*

```
1
2     heterogeneousMixture.setZFraction(compound1,molarFraction1);
3     //..... asignar la fracción molar para todos los compuestos
4
5     heterogeneousMixture.setPressure(pressure);
6     heterogeneousMixture.bubbleTemperature();
7     double temperature = heterogeneousMixture.getTemperature();
```

**Figura 5.16**

**Figura 5.17:** Algoritmo para el cálculo de temperatura de burbuja.

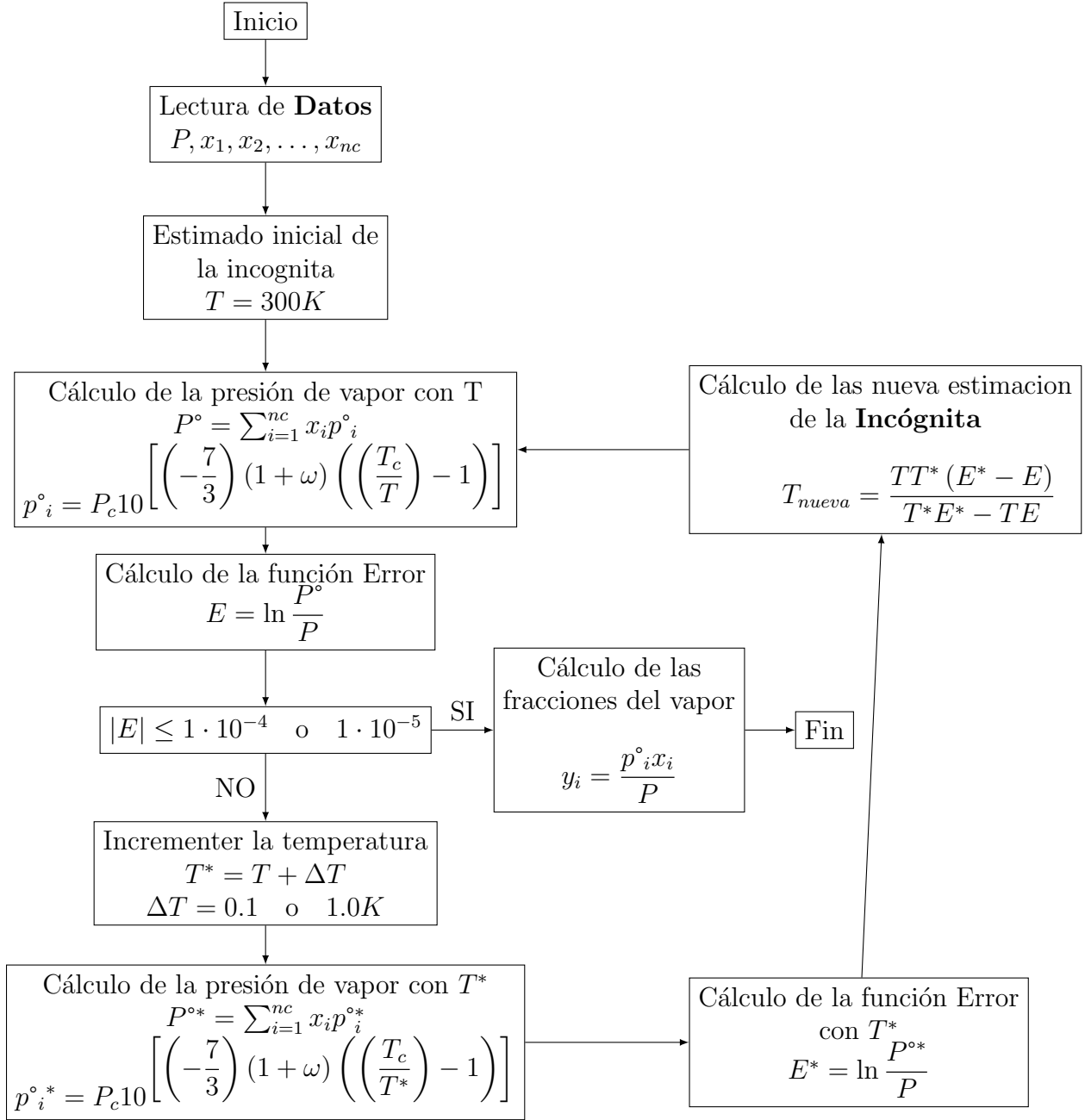


Figura 5.18: Algoritmo para el cálculo de la temperatura de burbuja con la ecuación del factor acéntrico para la presión.

5.7.4. Temperatura de Rocío

El cálculo de temperatura de rocío para una mezcla se realiza como se indica en la figura 5.19. Usando un objeto del tipo ‘HeterogeneousMixture’ de la librería **Materia** el cálculo se realiza como se muestra en los fragmentos de código 5.31 y 5.30.

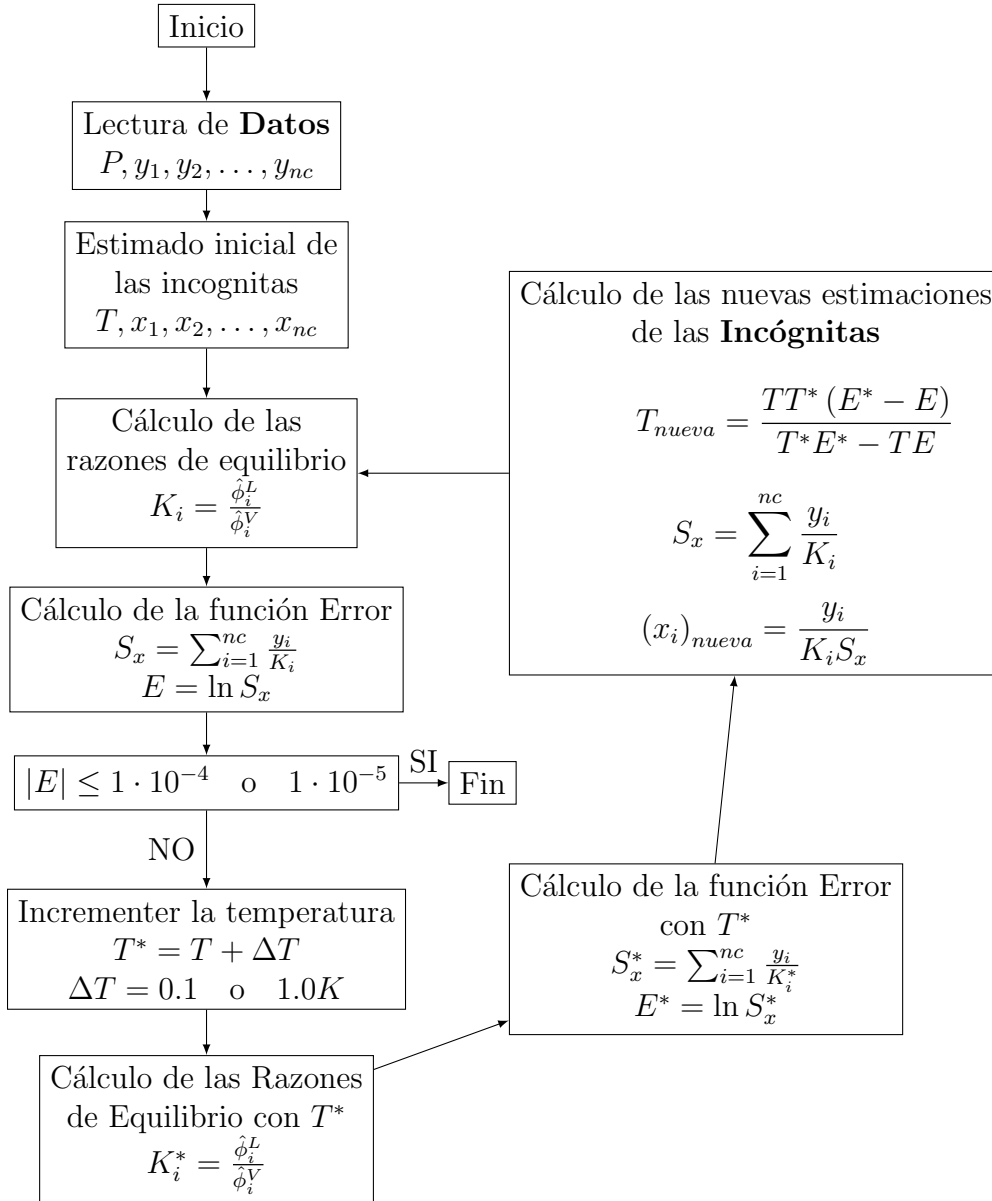
Si al método no se le proporciona un estimado inicial, como se muestra en el código 5.31 entonces la clase realiza la estimación de la temperatura como se muestra en la figura 5.20.

Código 5.30 *Cálculo de la temperatura de rocío proporcionando un estimado inicial.*

```
1
2     heterogeneousMixture.setZFraction(compound1,molarFraction1);
3     //..... asignar la fracción molar para todos los compuestos
4
5     heterogeneousMixture.setPressure(pressure);
6     heterogeneousMixture.dewTemperature(temeperatureEstimate);
7     double temperature = heterogeneousMixture.getTemperature();
```

Código 5.31 *Cálculo de la temperatura de rocío.*

```
1
2     heterogeneousMixture.setZFraction(compound1,molarFraction1);
3     //..... asignar la fracción molar para todos los compuestos
4
5     heterogeneousMixture.setPressure(pressure);
6     heterogeneousMixture.dewTemperature();
7     double temperature = heterogeneousMixture.getTemperature();
```

**Figura 5.19:** Algoritmo para el cálculo de temperatura de rocío

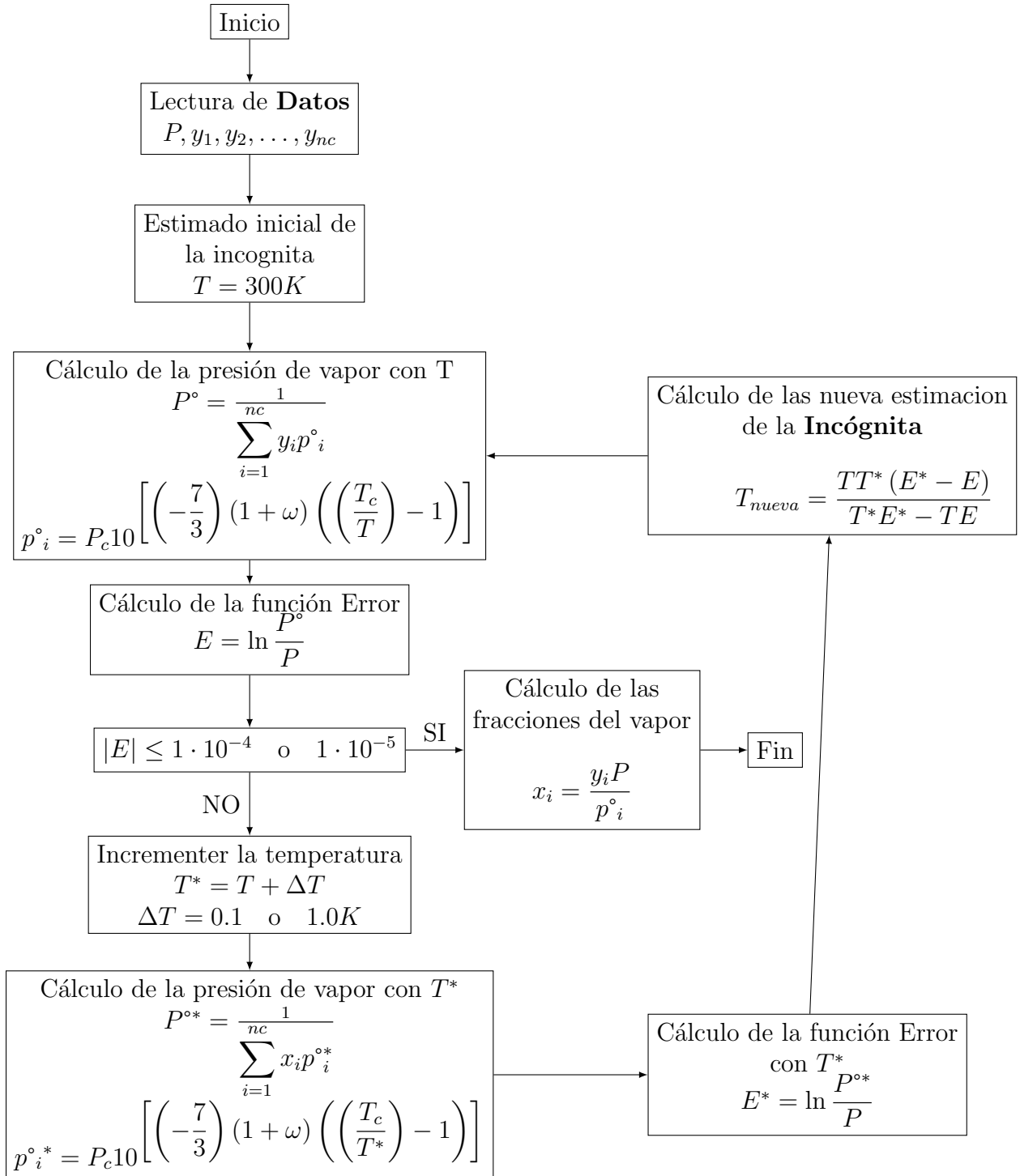


Figura 5.20: Algoritmo para el cálculo de la temperatura de rocío con la ecuación del factor acéntrico para la presión.

5.7.5. Presión de Burbuja

El cálculo de presión de burbuja para una mezcla se realiza como se indica en la figura 5.22. Usando un objeto del tipo ‘HeterogeneousMixture’ de la librería **Materia** el cálculo se realiza como se muestra en los fragmentos de código 5.33 y 5.32.

Si al método no se le proporciona un estimado inicial, como se muestra en el código 5.33 entonces la clase realiza la estimación de la presión de vapor con la ecuación del factor acéntrico D.14.

Código 5.32 *Cálculo de la presión de burbuja proporcionando un estimado inicial.*

```
1
2     heterogeneousMixture.setZFraction(compound1,molarFraction1);
3     //..... asignar la fracción molar para todos los compuestos
4
5     heterogeneousMixture.setTemperature(temperature);
6     heterogeneousMixture.bubblePressure(pressureEstimate);
7     double pressure = heterogeneousMixture.getPressure();
```

Código 5.33 *Cálculo de la presión de burbuja.*

```
1     heterogeneousMixture.setZFraction(compound1,molarFraction1);
2     //..... asignar la fracción molar para todos los compuestos
3
4     heterogeneousMixture.setTemperature(temperature);
5     heterogeneousMixture.bubblePressure();
6     double pressure = heterogeneousMixture.getPressure();
```

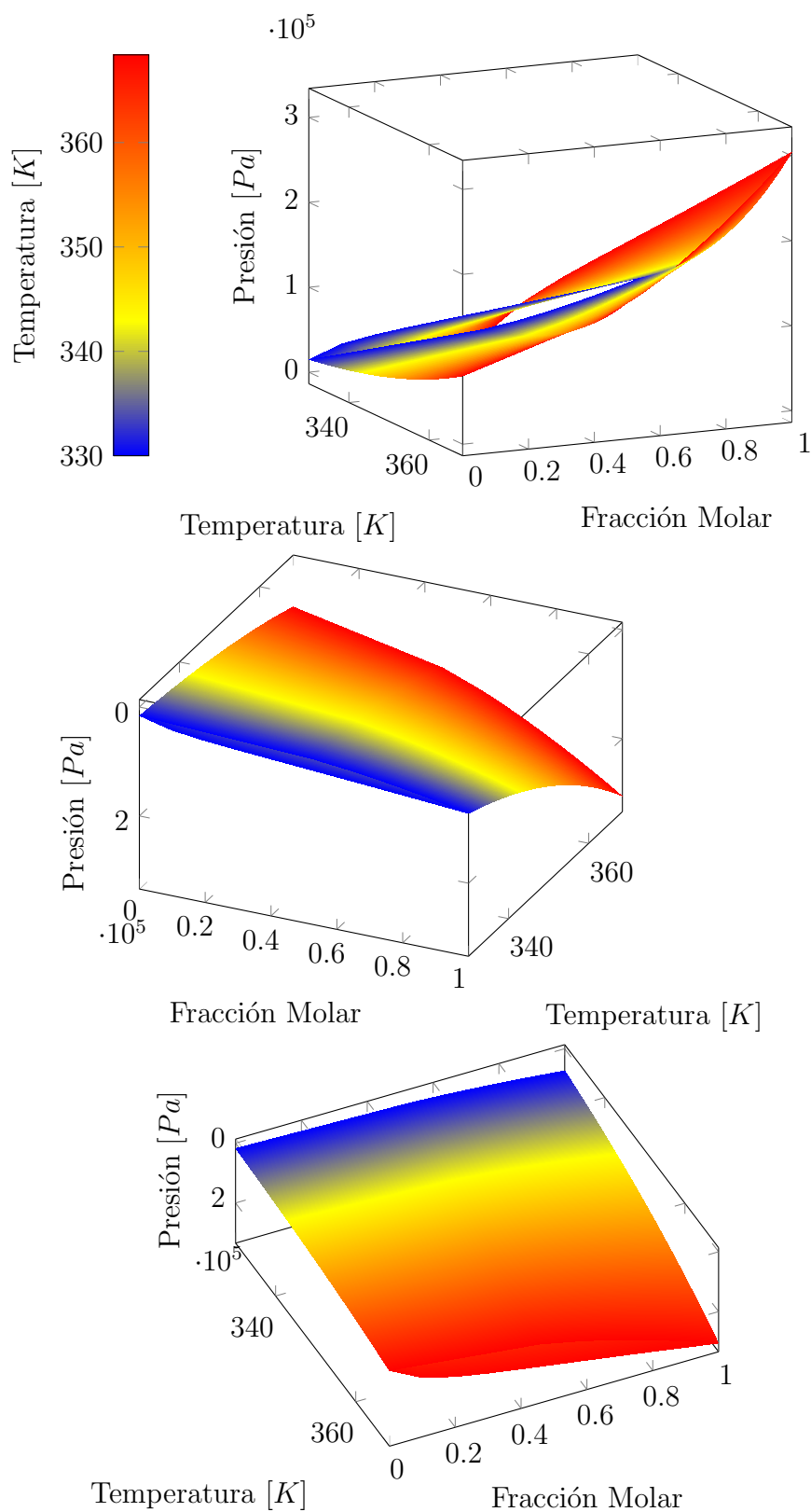
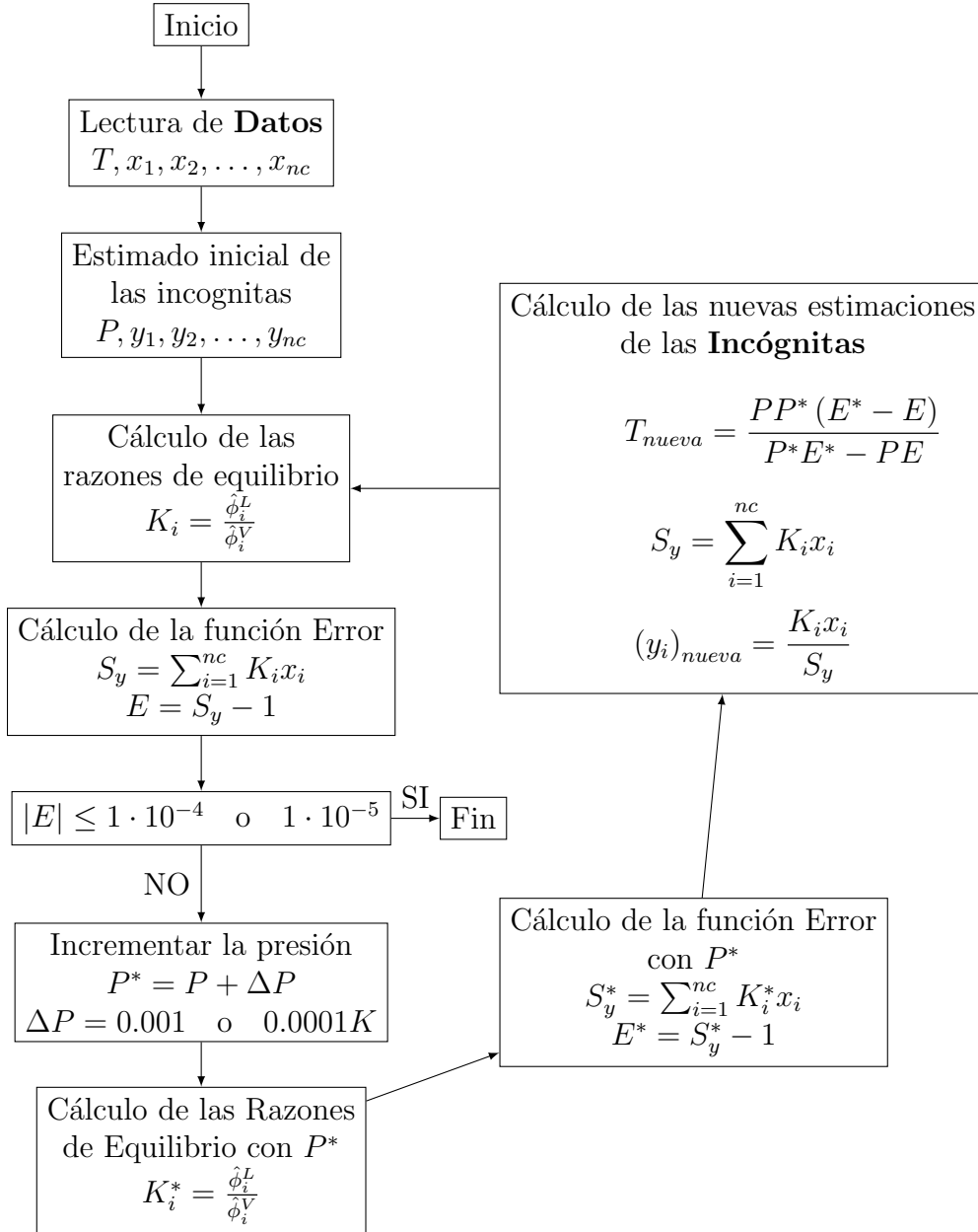


Figura 5.21: Diagramas tridimensionales de presión-composición-temperatura para el sistema metanol-agua.

**Figura 5.22:** Algoritmo para el cálculo de presión de burbuja

5.7.6. Presión de Rocío

El cálculo de presión de rocío para una mezcla se realiza como se indica en la figura 5.23. Usando un objeto del tipo ‘HeterogeneousMixture’ de la librería **Materia** el cálculo se realiza como se muestra en los fragmentos de código 5.35 y 5.34.

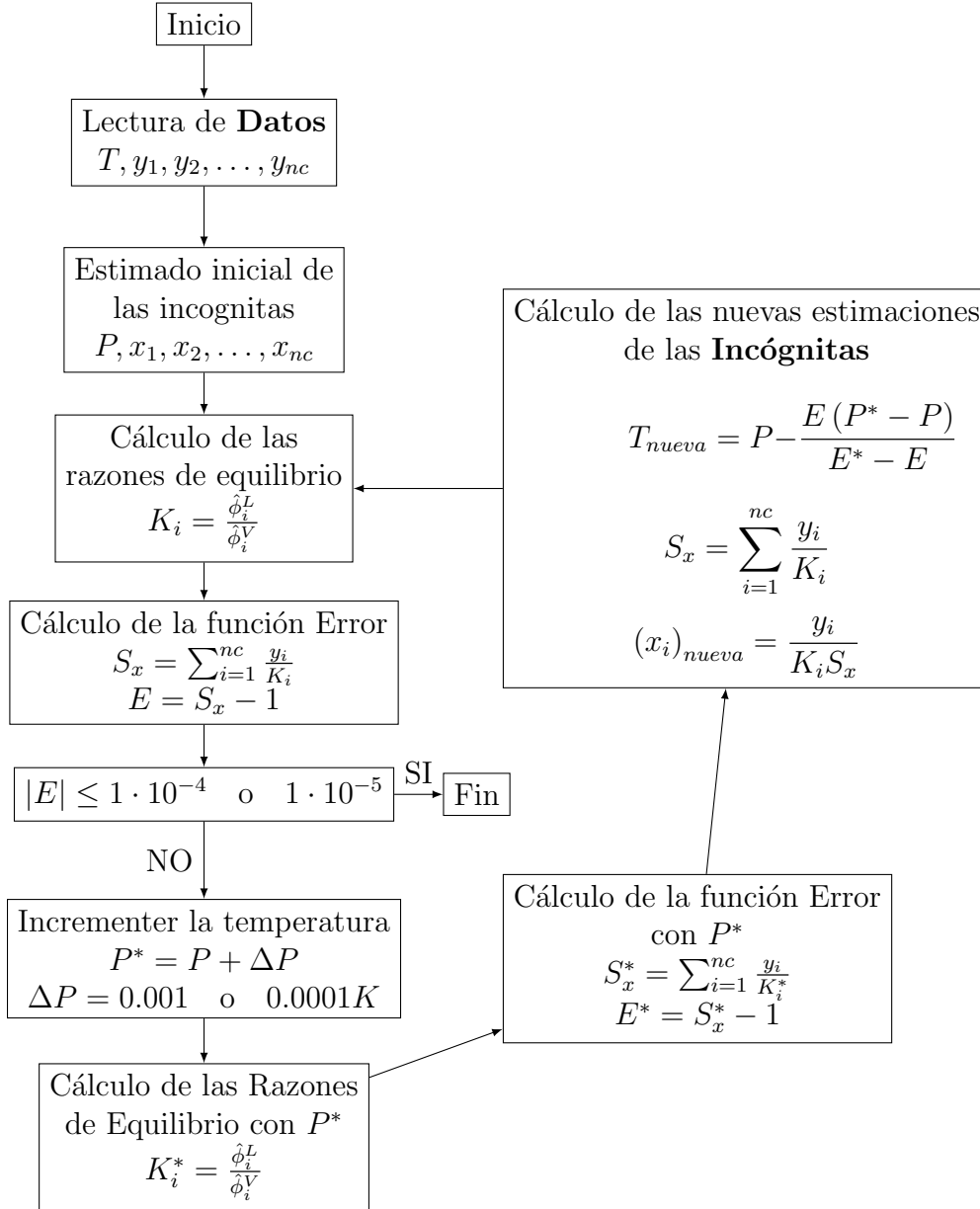
Si al método no se le proporciona un estimado inicial, como se muestra en el código 5.35 entonces la clase realiza la estimación de la presión de vapor con la ecuación del factor acéntrico D.14.

Código 5.34 *Cálculo de la presión de rocío proporcionando un estimado inicial.*

```
1
2     heterogeneousMixture.setZFraction(compound1,molarFraction1);
3     //..... asignar la fracción molar para todos los compuestos
4
5     heterogeneousMixture.setTemperature(temperature);
6     heterogeneousMixture.dewPressure(pressureEstimate);
7     double pressure = heterogeneousMixture.getPressure();
```

Código 5.35 *Cálculo de la presión de rocío.*

```
1
2     heterogeneousMixture.setZFraction(compound1,molarFraction1);
3     //..... asignar la fracción molar para todos los compuestos
4
5     heterogeneousMixture.setTemperature(temperature);
6     heterogeneousMixture.dewPressure();
7     double pressure = heterogeneousMixture.getPressure();
```

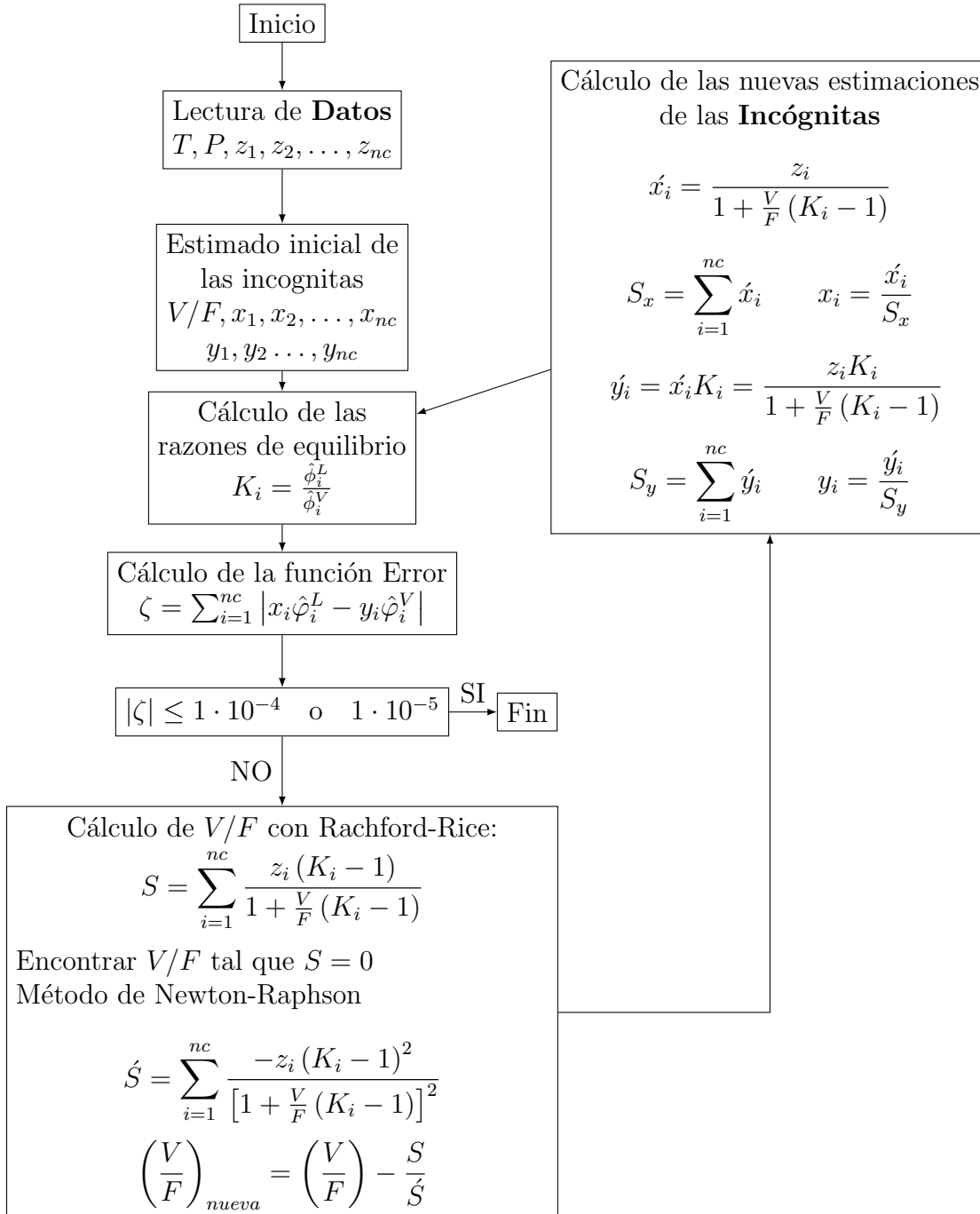
**Figura 5.23:** Algoritmo para el cálculo de la presión de rocío.

5.7.7. Flash

El cálculo del flash temperatura-presión se realiza como se indica en la figura 5.24. Usando un objeto del tipo ‘HeterogeneousMixture’ de la librería **Materia** el cálculo se realiza como se muestra en el fragmento de código 5.36 .

Código 5.36 *Cálculo del flash temperatura-presión.*

```
1
2  //asignar la fraccion molar para todos los compuestos
3  heterogeneousMixture.setZFraction(compound1,molarFraction1);
4  //.....
5
6  double vF = heterogeneousMixture.flash(temperature,pressure);
7
8  //leer las fracciones del vapor y del liquido para todos los compuestos
9  double x1 = heterogeneousMixture.getLiquid()
10     .getReadOnlyFractions().get(compound1);
11  double y1 = heterogeneousMixture.getVapor()
12     .getReadOnlyFractions().get(compound1);
13  //.....
```

**Figura 5.24:** Algoritmo para el cálculo del flash Temperatura-Presión.

5.8. Estimación de parámetros

Para el cálculo del parámetro a de la ecuación de estado cúbica es necesaria una expresión de α como se muestra en la ecuación D.2. Todas las expresiones de α escritas para este trabajo se listan en la tabla 5.3. Estas expresiones pueden tener parámetros que deben ser estimados en base a datos experimentales de compuestos puros. La estimación de los parámetros de α se realiza en la clase ‘HeterogeneousSubstance’.

Para realizar el cálculo de los parámetros de la ecuación cúbica para una mezcla, es necesario utilizar una regla de mezclado como se mostró en la sección 5.4.2. Las reglas de mezclado escritas para este trabajo se listan en la tabla 5.4. Las reglas de mezclado usan parámetros que deben ser estimados en base a datos experimentales de mezclas binarias. La estimación de los parámetros binarios se realiza en la clase ‘HeterogeneousMixture’.

- Sección 5.8.1 Estimación de parámetros de la expresión de α .
- Sección ?? Estimación de parámetros de interacción binaria para las reglas de mezclado.

5.8.1. Estimación de parámetros de α

La estimación de los parámetros de la expresión de α se realiza para un compuesto puro, y es necesaria una lista de datos experimentales Temperatura-Presión de la sustancia en equilibrio.

La clase ‘ErrorFunction’ calcula el error total como se muestra en la ecuación D.16. Un objeto del tipo ‘ErrorFunction’ pertenece como variable de instancia a la clase ‘HeterogeneousSubstance’, y es a través de él como se estiman los parámetros de la expresión de α .

Primero es necesario crear una lista, darle un nombre e indicar la fuente de donde se obtienen los datos, después, haciendo uso del método ‘addExperimentalData’ agregar todos los datos a la lista y finalmente asignar la lista a la función error de la sustancia heterogénea, como se muestra en el código 5.37.

Código 5.37 Creación de una lista de datos experimentales presión-temperatura con la clase ‘ExperimentalDataList’

```
1      ExperimentalDataList datalist = new ExperimentalDataList();
2      datalist.setName("Nombre de la lista");
3      datalist.setSource("Origen de los datos ejem. base de datos DIPPR");
4
5      //agregar todos los datos a la lista
6      datalist.addExperimentalData(temperature, pressure);
7      //....
8
9      heterogeneousSubstance.getErrorFunction().setExperimental(datalist);
```

Una vez que la función error contiene datos experimentales es posible obtener una comparación de los datos con el modelo de la sustancia, la figura 5.25 muestra una

gráfica presión-temperatura y la gráfica del ‘error relativo’-temperatura.

Los datos experimentales para el agua en esta sección fueron generados con la ecuación D.17, con los parámetros $A = 98.515, B = -8530.7, C = -10.984, D = 0.0000063663, E = 2$, para el rango de temperaturas $263.15 - 647.29[K]$.

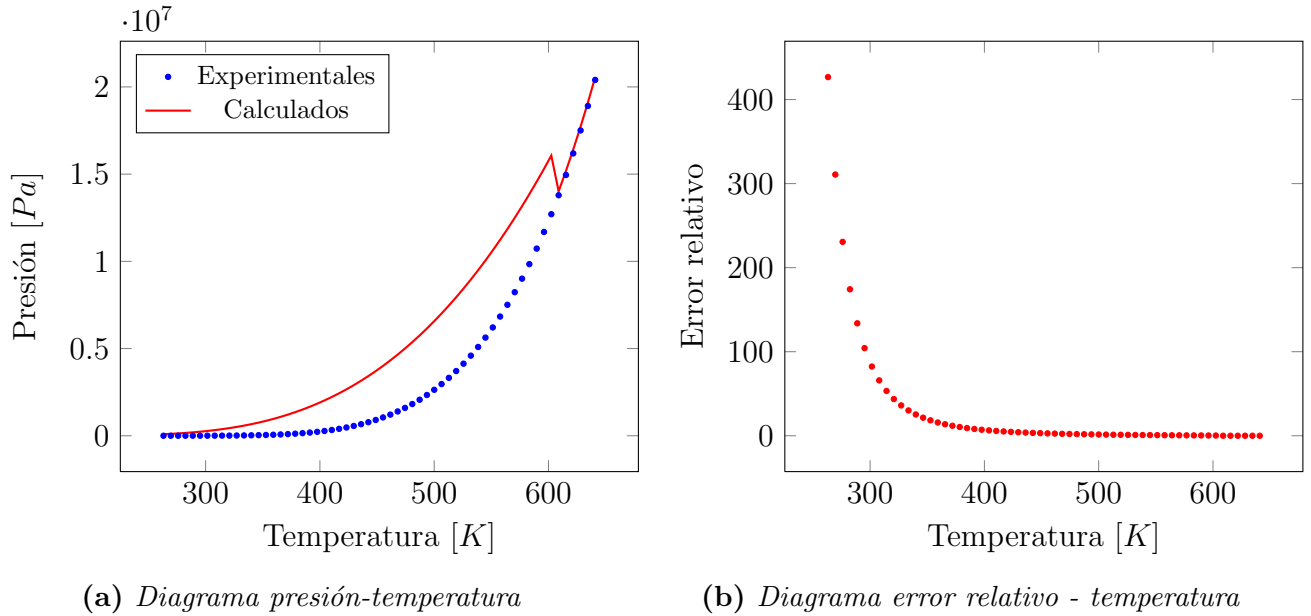


Figura 5.25: Diagramas para el compuesto puro ‘agua’ con la ecuación de estado Peng Robinson, y la expresión alpha de Soave de 2 parámetros E.13, con los valores $A = 0$ y $B = 0$

La clase ‘NewtonMethodSolver’ ejecuta el algoritmo de Newton Multivariable para minimizar la función error. Una instancia de la clase ‘NewtonMethodSolver’ se encuentra como variable de la clase ‘ErrorFunction’. Para realizar la estimación de parámetros se invoca el método ‘minimize’ de la función error, como se muestra en el código 5.38.

Código 5.38 Estimación de los parámetros al minimizar el valor de la función error.

```
1 heterogeneousSubstance.getErrorFunction().minimize();
```

Podemos observar el efecto de la estimación en las gráficas 5.26.

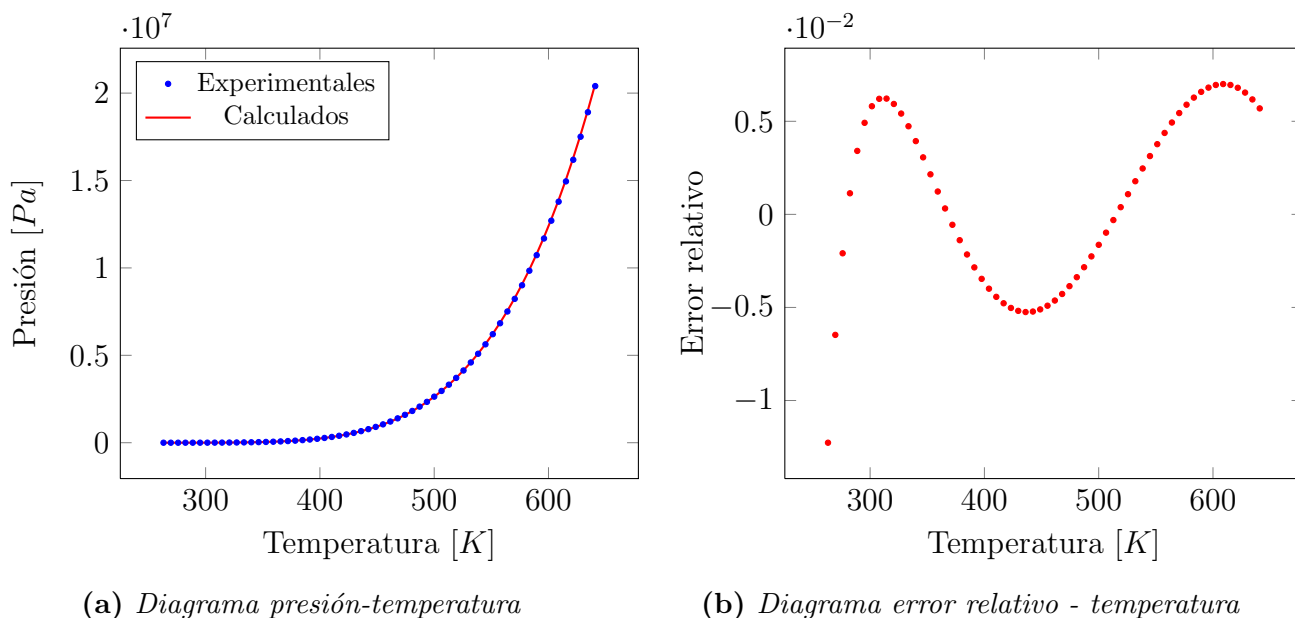


Figura 5.26: Diagramas para el compuesto puro ‘agua’ con la ecuación de estado Peng Robinson, y la expresión alpha de Soave de 2 parámetros E.13, con los valores $A = 0.70428$ y $B = 0.22243$

Una vez realizada la estimación es posible leer los parámetros de la expresión de α , en el código 5.39 se muestra como leer los parámetros de una forma general a través de la sustancia heterogénea, también es posible leer los valores desde el compuesto puro, sin embargo es necesario saber la expresión de α usada y el nombre de los parámetros dentro del código, lo cual lo hace menos general, ver el código 5.40.

Código 5.39 Lectura de los parámetros a través de la sustancia heterogénea.

```

1      for(AlphaParameter parameter: heterogeneousSubstance.alphaParameters()){
2          String name = parameter.getName();
3          double value = parameter.getValue();
4      }
```

Código 5.40 *Lectura de los parámetros a través del compuesto puro para la expresión α Soave de dos parámetros E.13*

```
1      Compound compound = heterogeneousSubstance.getComponent();
2      compound.getA_Soave();
3      compound.getB_Soave();
```

La clase ‘NewtonMethodSolver’ guarda la historia de convergencia de la ultima estimación, se puede acceder a los datos como se muestra en el código 5.41.

Código 5.41 *Lectura de la historia de convergencia*

```
1      List<Parameters_Error> convergenceHistory =
2          heterogeneousSubstance.getErrorFunction().getOptimizer()
3              .getConvergenceHistory();
4
5      for(Parameters_Error paramError:convergenceHistory){
6          int iteration = paramError.getIteration();
7          double[] params = paramError.getParameters();
8          double error=paramError.getError();
9      }
```

En la figura 5.27 se muestra la historia de convergencia, podemos notar como el error total disminuye conforme avanzan las iteraciones.

Dado que la expresión de α tiene dos parámetros, es posible apreciar la trayectoria en una gráfica de superficie, figura 5.28.

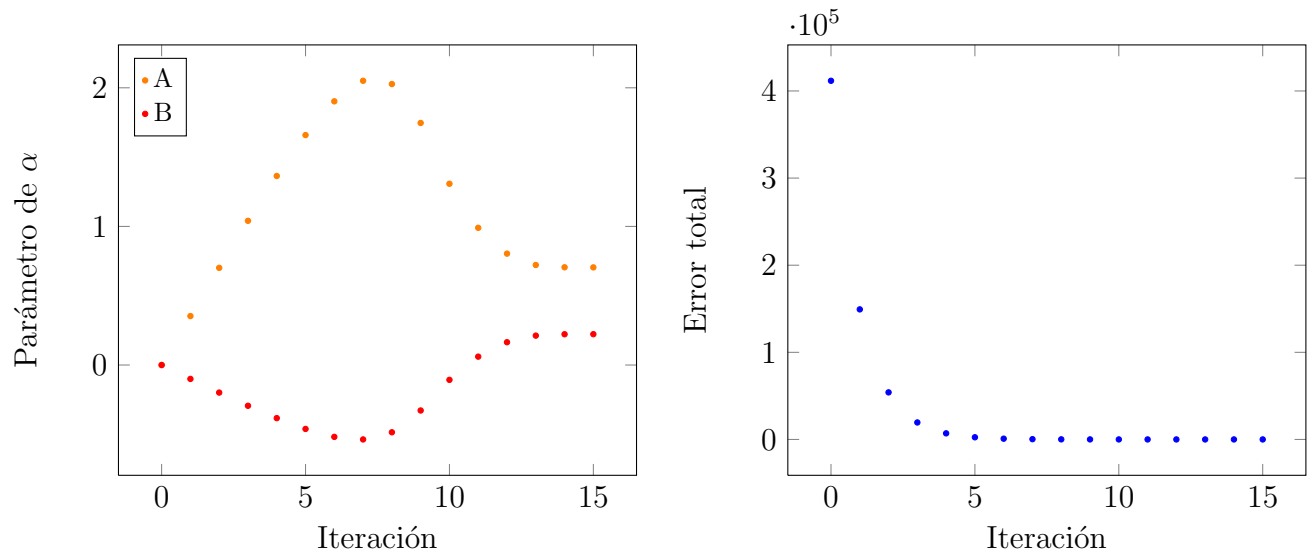


Figura 5.27: Historia de convergencia de la estimación de los parámetros para el agua de la expresión α de Soave de 2 parámetros E.13, ecuación de estado Peng Robinson.

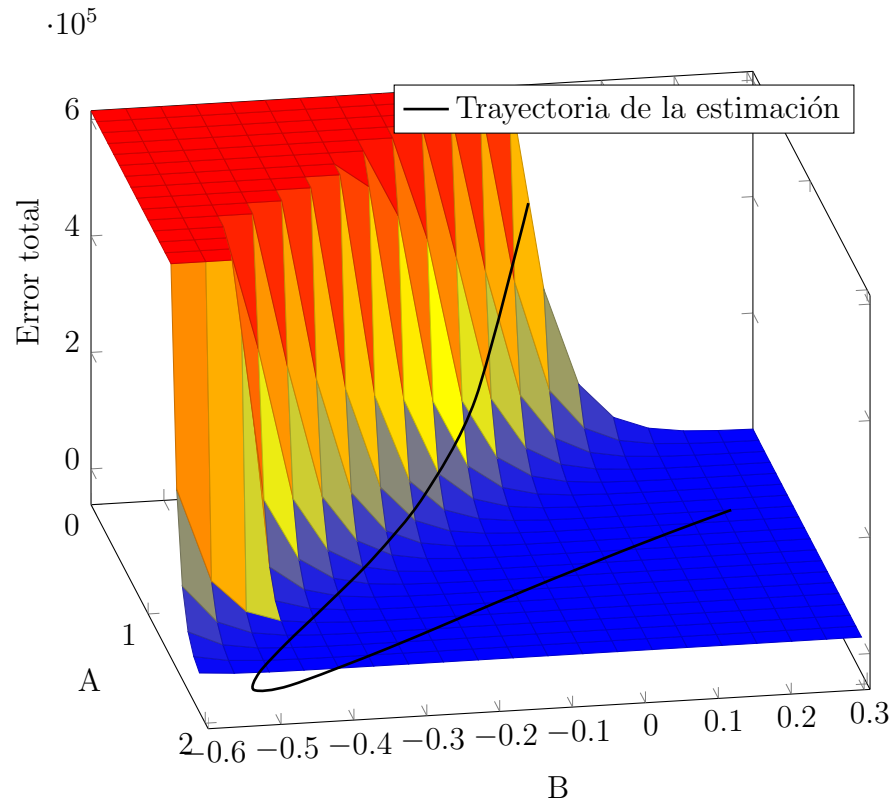


Figura 5.28: Trayectoria de la estimación de parámetros para el agua con la expresión α de Soave de dos parámetros E.13, ecuación de estado Peng Robinson.

5.8.2. Estimación de parámetros *Binarios*

Para realizar la estimación de los parámetros binarios de las reglas de mezclado, es necesaria una lista de datos experimentales de presión, temperatura y fracciones molares del líquido y del vapor.

La clase ‘ErrorFunction’ realiza el cálculo de la función error, y la clase ‘NewtonMethodSolver’ realiza el algoritmo para la estimación de los parámetros minimizando el valor de la función error.

Para crear una lista de datos experimentales para una mezcla binaria, es necesario indicar si la experimentación se llevó a cabo de manera isobárica o isotérmica, el tipo de experimentación se indica con el método ‘setType’ y se puede dar como argumento del método las opciones ‘ExperimentalDataBinaryType.isobaric’ ó ‘ExperimentalDataBinaryType.isothermic’. Cuando se indican las fracciones molares del líquido y del vapor solo se indican las de un compuesto, por lo tanto se debe decir a la lista de datos cual es el compuesto de referencia y cual es el de no referencia, ello se indica con los métodos ‘setReferenceComponent’ y ‘setNonReferenceComponent’.

Es muy importante que los compuestos de la mezcla sean iguales a los indicados en la lista, sin embargo en este trabajo no se ha implementado una forma para asegurar que esto suceda, esto se deberá realizarse en un trabajo futuro.

El código 5.42 muestra la creación de la lista de datos experimentales y la asigna a la mezcla heterogénea.

Código 5.42 *Creación de la lista de datos experimentales y la asignación de esta a la mezcla heterogénea.*

```
1  ExperimentalDataBinaryList blist =  
2      new ExperimentalDataBinaryList("Ejemplo lista");  
3  blist.setSource("Fuente de los datos");  
4  blist.setType(experimentalDataBinaryType);
```

```

5
6     blist.setReferenceComponent(referenceCompound);
7     blist.setNonReferenceComponent(nonReferenceCompound);
8
9     //agregar todos los datos
10    blist.addExperimentalDataToList(temperature, pressure, x, y);
11    //.....
12
13    heterogeneousMixture.getErrorfunction().setExperimental(blist);

```

La figura 5.29 muestra el diagrama temperatura-fracción molar antes de la estimación de parámetros, para el sistema metanol-agua, con la ecuación PRSV, y la regla de mezclado de Van Der Waals.

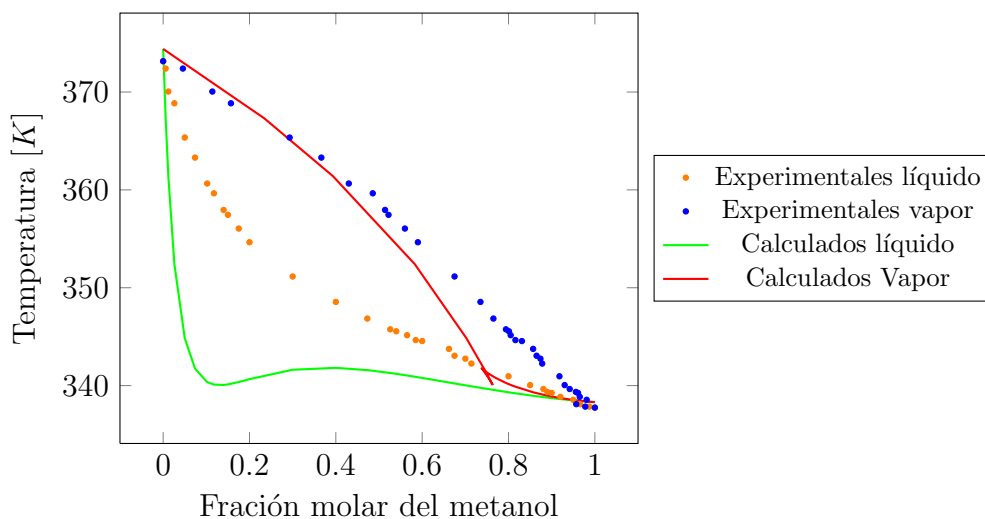


Figura 5.29: Diagrama temperatura-fracción molar para el sistema metanol-agua, con la ecuación PRSV y la regla de mezclado de Van Der Waals. Los parámetros de la expresión de alfa para los compuestos es igual a cero, y tambien los parámetros binarios de la regla de mezclado

Es necesario realizar la estimación de los parámetros de la expresión de α para cada compuesto antes de la estimación de los parámetros binarios.

El resultado de la estimación de los parámetros de la expresión de α para cada compuesto es:

- Metanol, $k : -2.7 \cdot 10^{-5}$
- Agua, $k : 0.009276$

La estimación de los parámetros binarios se hace a través de la función error, como se muestra en el código 5.43.

Código 5.43 *Estimación de los parámetros binarios de la regla de mezclado minimizando el valor de la función error.*

```
1 heterogeneousMixture.getErrorfunction().minimize();
```

En la figura 5.30 se muestra el resultado de la estimación de parámetros.

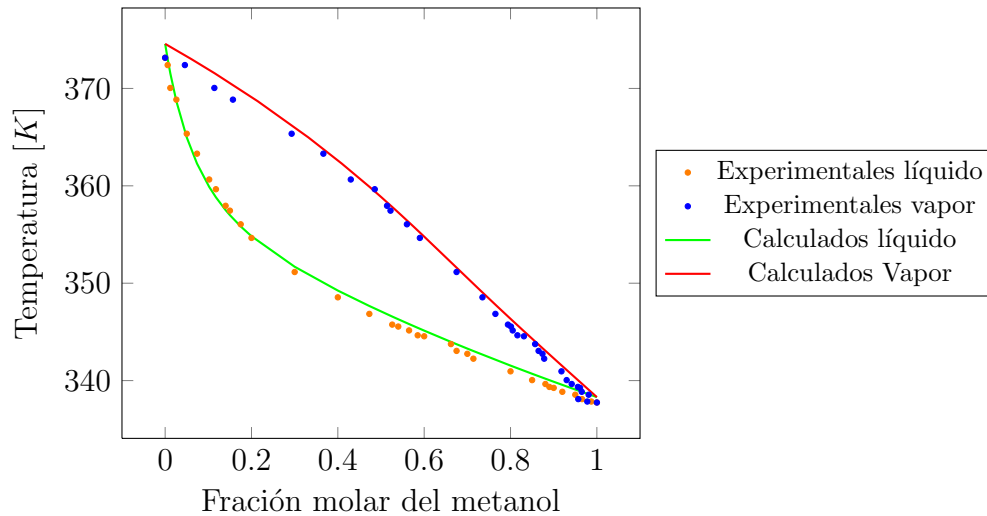


Figura 5.30: *Diagrama temperatura-fracción molar para el sistema agua-metanol, con la ecuación PRSV, parámetro binario $k_{ij} = -0.0739981$, parámetros de α : Metanol $k = -2.7 \cdot 10^{-5}$, Agua $k = 0.009276$*

Capítulo 6

Página de Internet

Como se comentó en el capítulo de introducción una gran ventaja del lenguaje java es su capacidad para crear páginas de internet. En el presente trabajo se creó una página de internet cuyo objetivo es exponer las funciones principales de la librería materia.

La página se encuentra en la dirección web: `chimicae-materia.rhcloud.com`, se recomienda utilizar un navegador moderno como google chrome.

El presente capítulo trata sobre como utilizar la página para:

- Crear
 - Substancias Sección 6.2
 - Mezclas Sección 6.3
- Gráficas
 - Presión-Volumen-Temperatura
 - Z-Presión-Temperatura
 - Fugacidad-Presión-Temperatura
 - Temperatura-Entalpía-Presión

- Temperatura-Entropía-Presión
 - Temperatura-Gibbs-Presión
 - Temperatura-Presión-Volumen
- Estimación de parámetros
 - Expresión de α
 - Regla de Mezclado

6.1. Selección de compuestos puros

La página de internet dispone de la base de datos ChemSep v6.96 derechos de autor Harry Kooijman y Ross Taylor (2013) bajo la licencia ‘Artistic License’: http://www.perlfoundation.org/artistic_license_2_0.

Primero se debe cargar los compuestos puros, en la sección ‘Creación’ -¿ ‘Compuesto Puro’, en el recuadro de texto se debe escribir el nombre del compuesto puro deseado en inglés y dar click en el boton buscar. Si la página encuentra coincidencia en los nombre mostrará una lista debajo del recuadro con los compuestos encontrados.

Para agregar el compuesto basta con dar click en el boton ‘Agregar’ .En la tabla que se muestra a la derecha de la página se muestran los compuestos agregados. Ver la figura 6.1.

6.2. Creación de sustancias

En la sección ‘Creación’ -¿ ‘Sustancia’ se permite crear materia de un solo compuesto, para después utilizarse en las secciones de graficación y estimación de parámetros de la expresión de α .

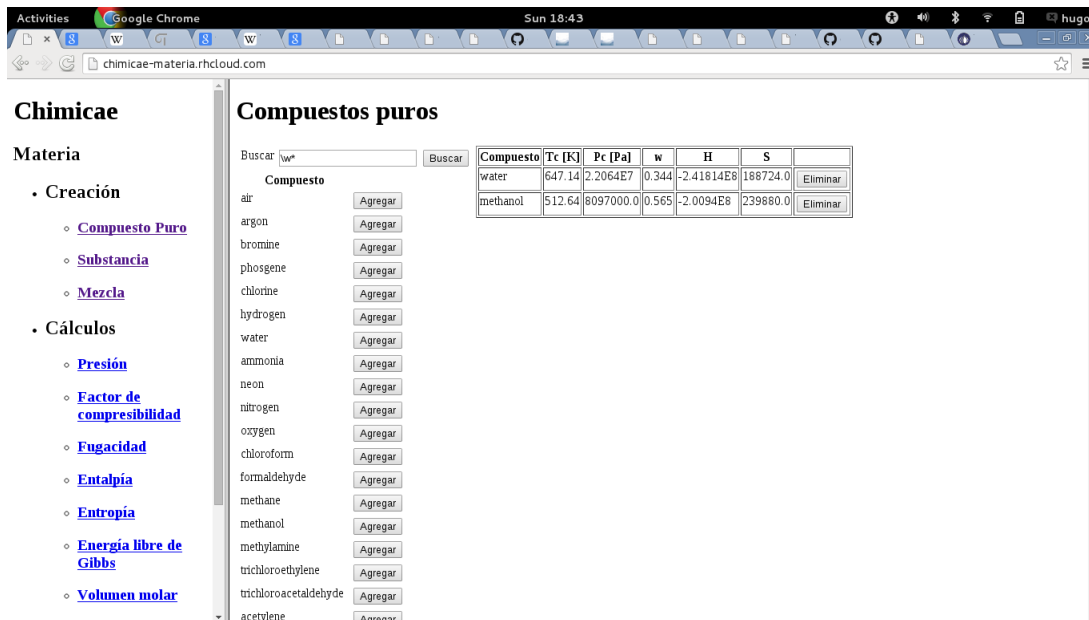


Figura 6.1: Formulario para seleccionar y cargar compuestos puros en la página de internet

Esta sección permite elegir la ecuación cúbica, la expresión de α , el compuesto y la fase homogénea con la cual se creará la sustancia. Al final de la sección el botón ‘Aceptar’ creará la Sustancia y mostrará el resultado de la creación.

La figura 6.2 muestra la interfaz de usuario para la creación de sustancias. La figura 6.3 muestra las propiedades de la sustancia creada.

6.3. Creación de Mezclas

En la sección ‘Creación’ -¿ ‘Mezcla’ que permite crear materia con múltiples compuestos puros, solo aquellas mezclas que contengan dos compuestos estarán disponibles en la sección Estimación de parámetros binarios.

El formulario permite elegir la ecuación cúbica, la fase, la regla de mezclado, elegir cada compuesto y definir su expresión de α y su fracción molar. Ver figura 6.4.

El botón con la etiqueta ‘Parámetros’ permite definir los parámetros binarios para la

Chimicae

Materia

- Creación
 - [Compuesto Puro](#)
 - [Substancia](#)
 - [Mezcla](#)
- Cálculos
 - [Presión](#)
 - [Factor de compresibilidad](#)
 - [Fugacidad](#)
 - [Entalpía](#)
 - [Entropía](#)
 - [Energía libre de Gibbs](#)
 - [Volumen molar](#)

Substancia

Ecuación de estado Twu-Sim-Tassone

Los parámetros u y w para la ecuación de estado son:

$u = 2.5$
 $w = -1.5$
 $\Omega_a = 0.470507$
 $\Omega_b = 0.074074$

Expresión de $\alpha(T)$ Expresión independiente de T (Van Der Waals)

----- $T < T_c$
 $\alpha(T) = \left[1 + m \left(1 - \sqrt{T_r} \right) - k_1 (1 - T_r) (0.7 - T_r) \right]^2$
 $m = 0.378893 + 1.4897153\omega - 0.17131848\omega^2 + 0.0196554\omega^3$
 ----- $T > T_c$
 $\alpha(T) = \left[1 + m \left(1 - \sqrt{T_r} \right) \right]^2$
 $m = 0.378893 + 1.4897153\omega - 0.17131848\omega^2 + 0.0196554\omega^3$

Componente water

Fase homogénea LIQUID

Figura 6.2: Formulario para crear sustancias en la página de internet

Chimicae

Materia

- Creación
 - [Compuesto Puro](#)
 - [Substancia](#)
 - [Mezcla](#)
- Cálculos
 - [Presión](#)
 - [Factor de compresibilidad](#)
 - [Fugacidad](#)
 - [Entalpía](#)
 - [Entropía](#)
 - [Energía libre de Gibbs](#)
 - [Volumen molar](#)

Ecuación de estado: Ecuación de estado cúbica Twu-Sim-Tassone

Expresión de $\alpha(T)$: Expresión independiente de T (Van Der Waals)

Estado de agregación: LIQUID

Compuesto: water

[Ver todos los sistemas](#)

Figura 6.3: Página que muestra las propiedades de la sustancia recién creada

The screenshot shows a web browser window with the URL `localhost:8080/chimicae/`. The page is titled 'Chimicae' and has a sidebar on the left with the following menu items:

- Materia
 - Creación
 - Compuesto Puro
 - Substancia
 - Mezcla
 - Cálculos
 - Presión
 - Factor de compresibilidad
 - Fugacidad
 - Entalpía
 - Entropía
 - Energía libre de Gibbs
 - Volumen molar

The main content area is titled 'Mezcla' and contains the following fields and controls:

- Ecuación de estado:** A dropdown menu set to 'Peng-Robinson'.
- Text: 'Los parámetros u y w para la ecuación de estado son:'
- Parameters:
 - $u = 2.0$
 - $w = -1.0$
 - $\Omega_a = 0.45723553$
 - $\Omega_b = 0.077796074$
- Fase homogénea:** A dropdown menu set to 'LIQUID'.
- Regla de Mezclado:** A dropdown menu set to 'Van Der Waals'.
- A table for adding components:

Compuesto	Expresión de α	Fración molar	
water	Stryjek and Vera		Eliminar
methanol	Stryjek and Vera		Eliminar
- Buttons: 'Agregar', 'Parámetros', 'Aceptar'.

Figura 6.4: *Formulario para la creación de mezclas*

regla de mezclado seleccionada. Finalmente el botón ‘Aceptar’ construye la mezcla y la pone a disposición para las siguientes secciones.

6.4. Gráficos

Las secciones de gráficos dividen el recuadro derecho de forma horizontal, en la parte superior se localizan los controles de las gráficas, y en la parte inferior la gráfica tridimensional.

En esta sección estan disponibles las sustancias y mezclas creadas anteriormente, basta con dar click en la caja ‘combobox’ y seleccionar el sistema, esto es igual para todas las gráficas de la sección.

En algunas gráficas se muestran los resultados de la fase líquida y vapor sin importar la fase seleccionada en el momento de la creación del sistema.

6.4.1. Presión-Volumen-Temperatura

En este tipo de gráfica no se involucra la fase del sistema.

La sección de controles permite elegir el rango del volumen molar y el rango de temperatura.

El propósito de la figura 6.5 es observar los puntos de inflexión característicos de las ecuaciones cúbicas, en el caso que ellos se encuentren dentro del dominio seleccionado.

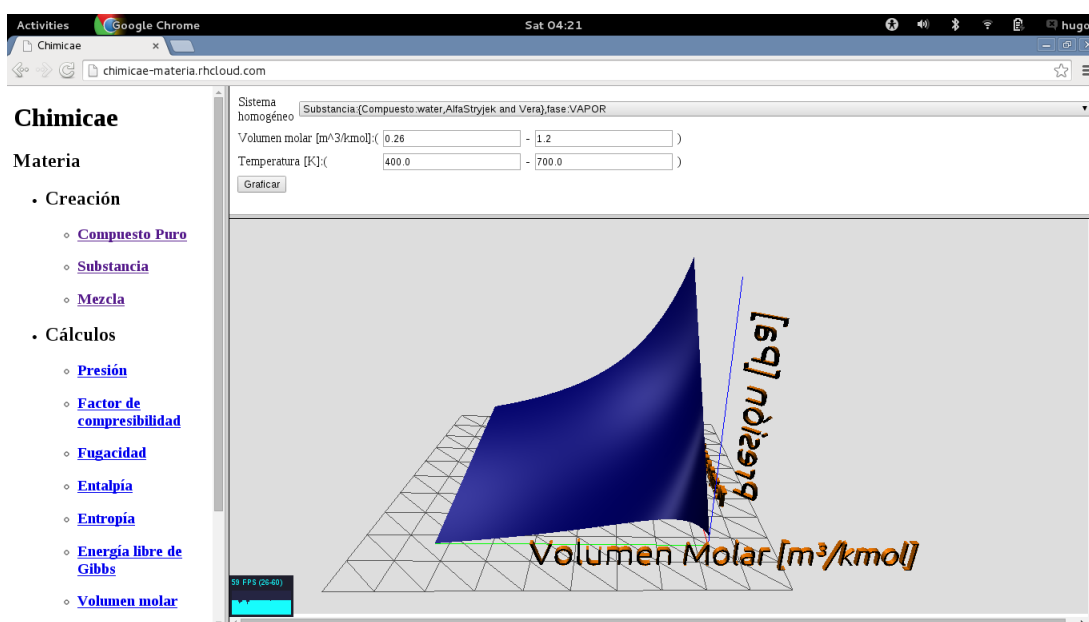


Figura 6.5: Gráfica de presión-volumen-temperatura para el agua con la ecuación de estado de Peng-Robinson y la expresión de α de stryjek and Vera que esta cargada por default. El rango de volumen: $0.26 - 1.2 \left[\frac{\text{m}^3}{\text{kmol}} \right]$ y temperatura: $400 - 700 [\text{K}]$

6.4.2. Z-Presión-Temperatura

Usualmente las gráficas del factor de compresibilidad se crean en dos dimensiones mostrando diferentes temperaturas y no es posible ver efecto que se tiene a altas temperaturas y presiones.

Esta gráfica depende de la fase seleccionada para el sistema y solo se muestra la

seleccionada.

Semejante a la sección anterior se debe determinar un rango de presión y temperatura. La figura 6.6 muestra el efecto en tercera dimensión.

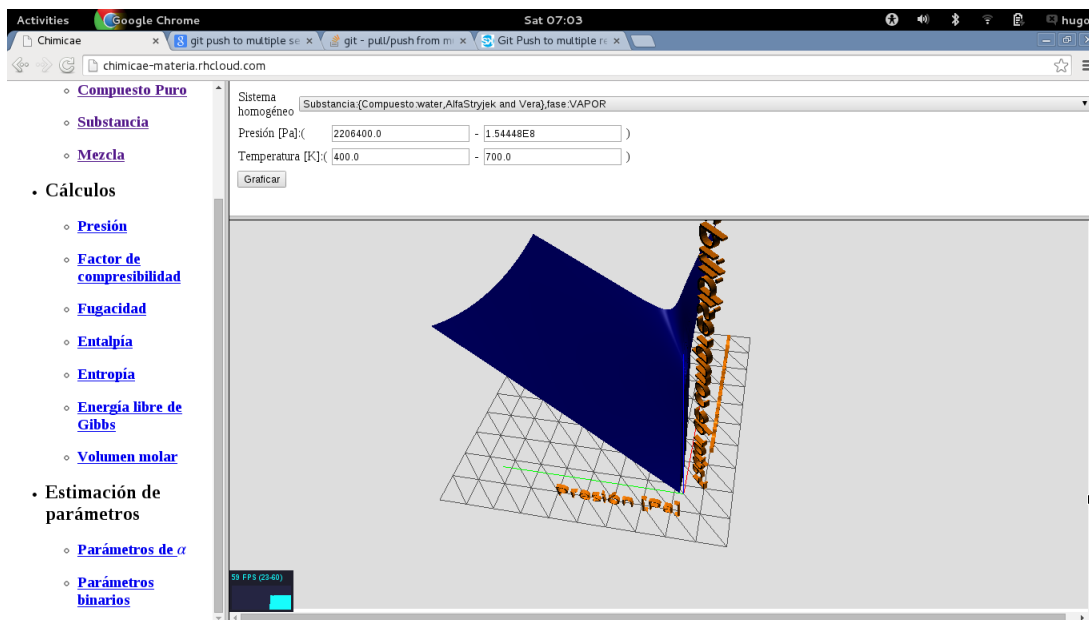


Figura 6.6: Gráfica de z (Factor de compresibilidad)-Presión-Temperatura para el agua con la ecuación de estado de Peng-Robinson y la expresión de α stryjek and Vera.

6.4.3. Fugacidad-Presión-Temperatura

En esta gráfica se dibujan las curvas de fugacidad para el líquido (rojo) y el vapor (azul) para el dominio determinado (Presión y temperatura).

El objetivo de la gráfica es observar el equilibrio líquido-vapor en la línea de intersección de los planos. Ver la figura 6.7.

6.4.4. Temperatura-Entalpía-Presión

En esta y las siguientes gráfica se dibujan la región de dos fases (rojo) y la región líquida y vapor (azul).

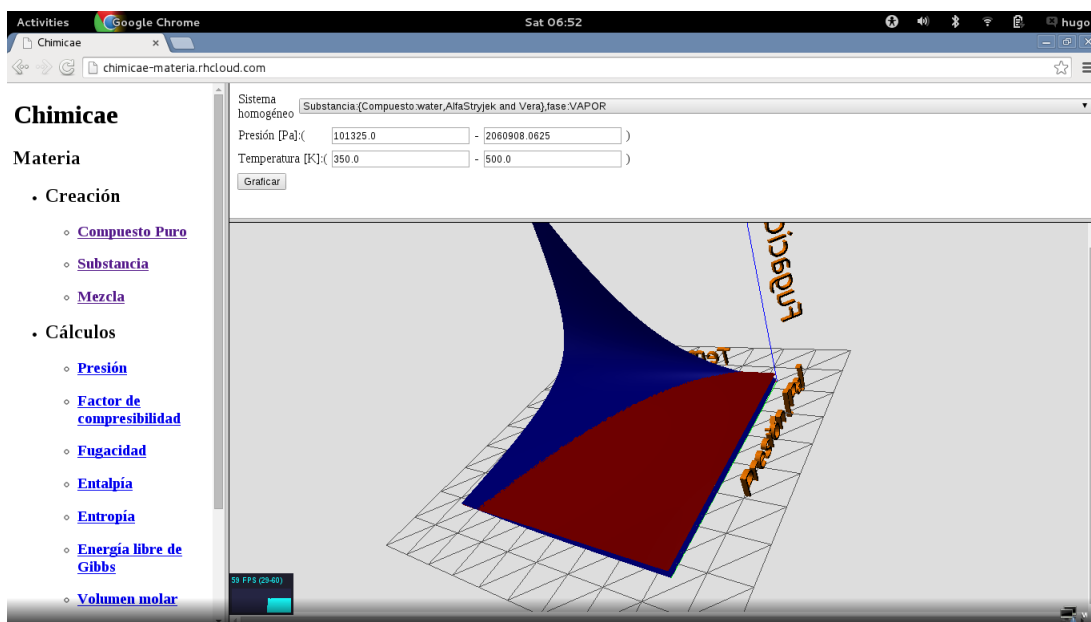


Figura 6.7: Gráfica de Fugacidad-Presión-Temperatura para el agua con la ecuación de estado de Peng-Robinson y la expresión de α stryjek y Vera

En estas gráfica no se debe determinar el rango, ya que este se calcula con los puntos críticos y la diferencia de propiedades en las regiones de dos fases.

Podemos ver la forma de la gráfica en la figura 6.8.

6.4.5. Temperatura-Entropía-Presión

6.4.6. Temperatura-Gibbs-Presión

6.4.7. Temperatura-Presión-Volumen

6.5. Estimación de parámetros de α

En la sección de 'Estimación de parámetros' - 'Parámetros de α ' podemos seleccionar una de las sustancias creadas y estimar sus parámetros con una serie de datos generados a través de la ecuación ?? que nos proporciona la base de datos Chemsep.

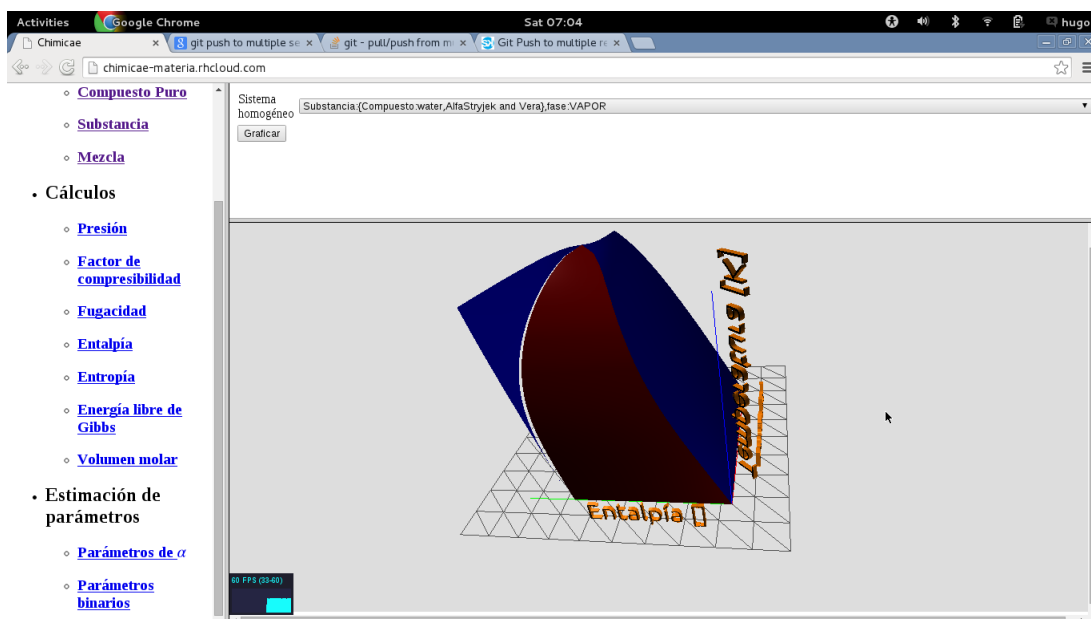


Figura 6.8: Gráfica de Temperatura-Entalpía-Presión para el agua con la ecuación de estado de Peng-Robinson y la expresión de α Stryjek y Vera

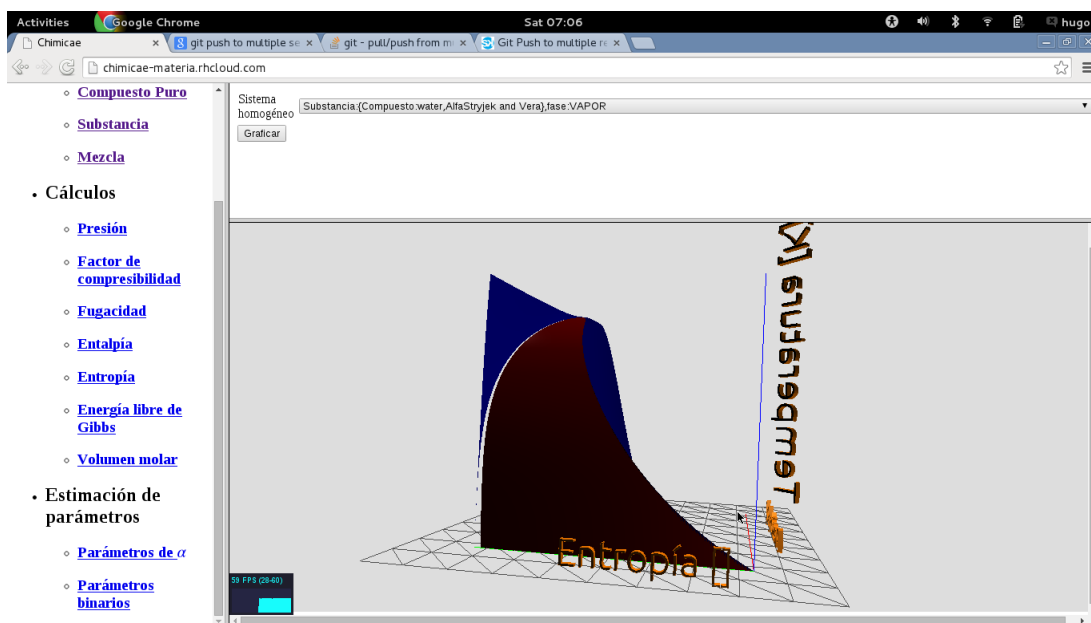


Figura 6.9: Gráfica de Temperatura-Entropía-Presión para el agua con la ecuación de estado de Peng-Robinson y la expresión de α Stryjek y Vera

La sección se muestra una gráfica que compara el modelo con los datos generados, una que muestra el error relativo, y finalmente una que muestra la historia de convergencia

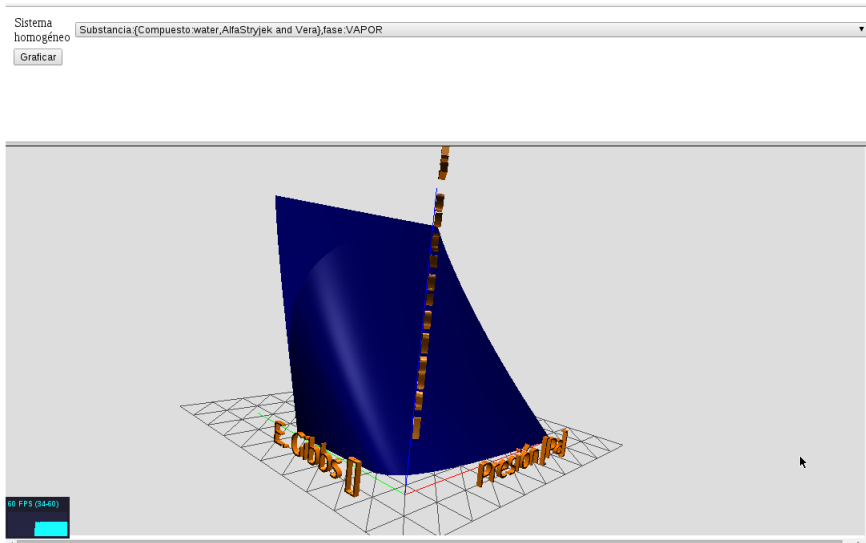


Figura 6.10: Gráfica de Temperatura-(Energía libre de Gibbs)-Presión para el agua con la ecuación de estado de Peng-Robinson y la expresión de α de Stryjek y Vera

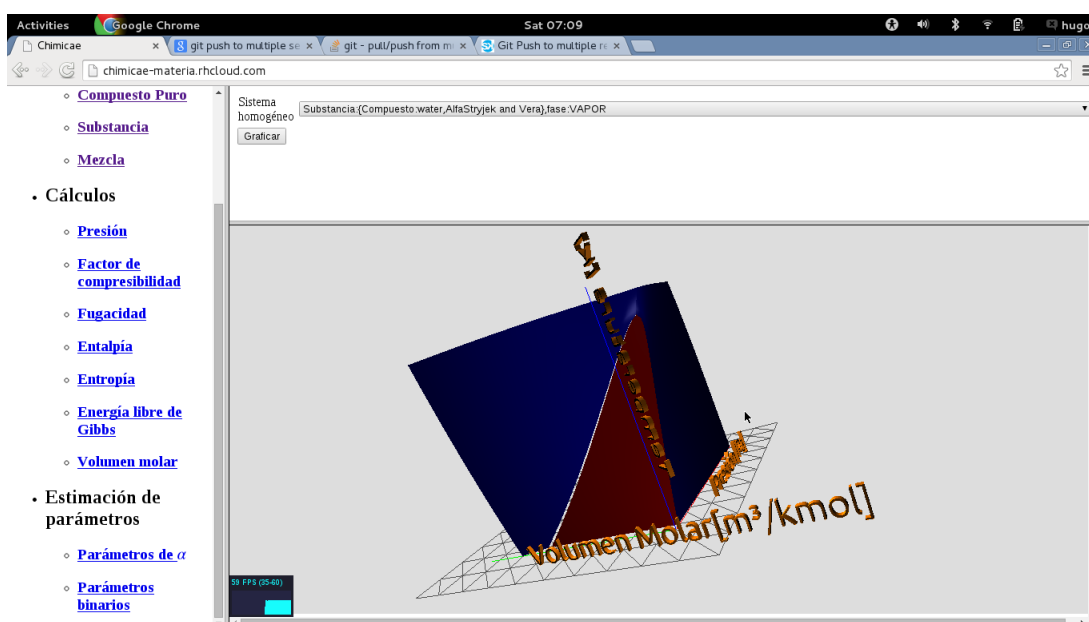


Figura 6.11: Gráfica de Temperatura-Presión-Volumen molar para el agua con la ecuación de estado de Peng-Robinson y la expresión de α de Stryjek y Vera

después de estimar los parámetros.

Una imagen del formulario se presenta en la figura 6.12

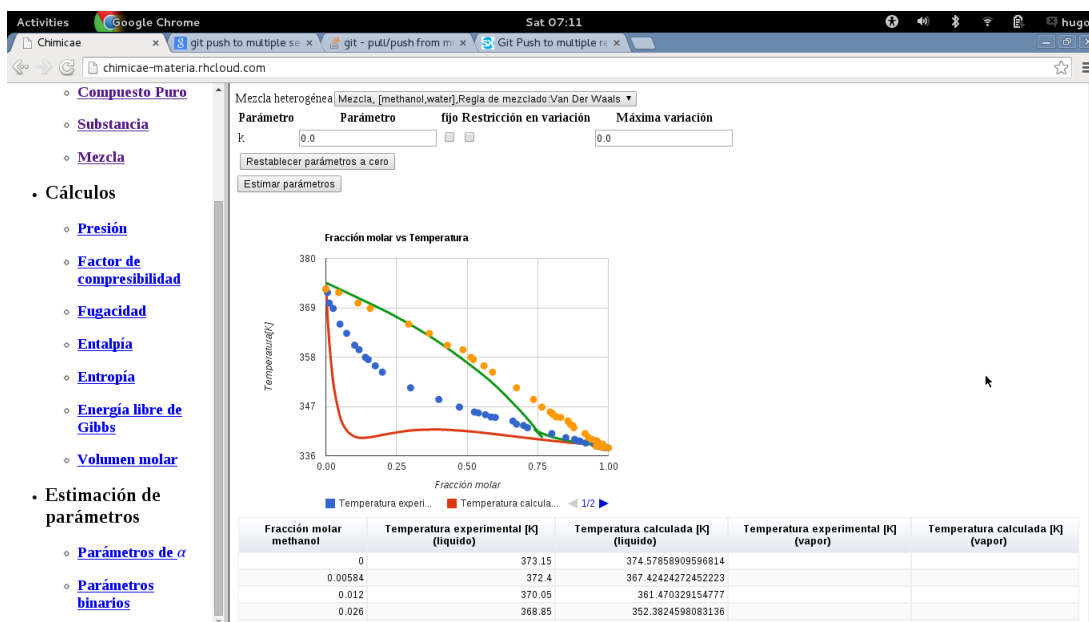


Figura 6.12: Formulario para la estimación de parámetros de la expresión de α

6.6. Estimación de parámetros binarios

En la sección de 'Estimación de parámetros' - 'Parámetros binarios' podemos seleccionar una de las mezclas creadas previamente y estimar los parámetros de la regla de mezclado con datos que deberán ser agregados previamente a la página de internet.

De manera semejante a la sección anterior en la página se realiza una comparación de los datos experimentales con el modelo, se muestra una gráfica del error relativo y la historia de la convergencia.

Una imagen del formulario se presenta en la figura 6.13

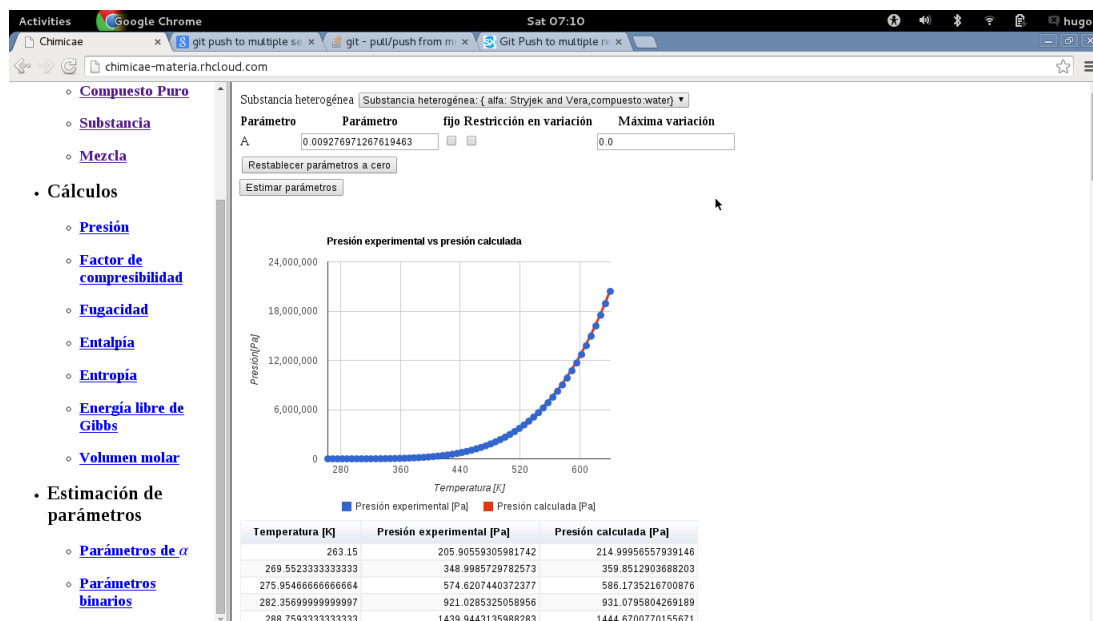


Figura 6.13: Formulario para la estimación de parámetros de la regla de mezclado de Van Der Waals

Apéndice A

Ejemplo de uso con Netbeans

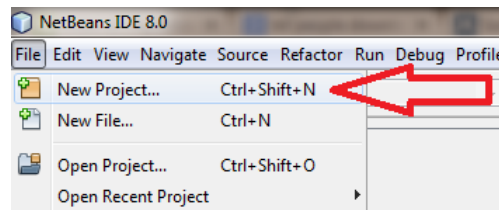
A.1. Requisitos

1. Necesitamos tener instalado kit de desarrollo Jdk de java que se puede descargar desde la página (<http://www.oracle.com/technetwork/java/javase/downloads/>).
2. Necesitamos tener instalado el ambiente de desarrollo Netbeans que se puede descargar de la página (<https://netbeans.org/downloads/>).

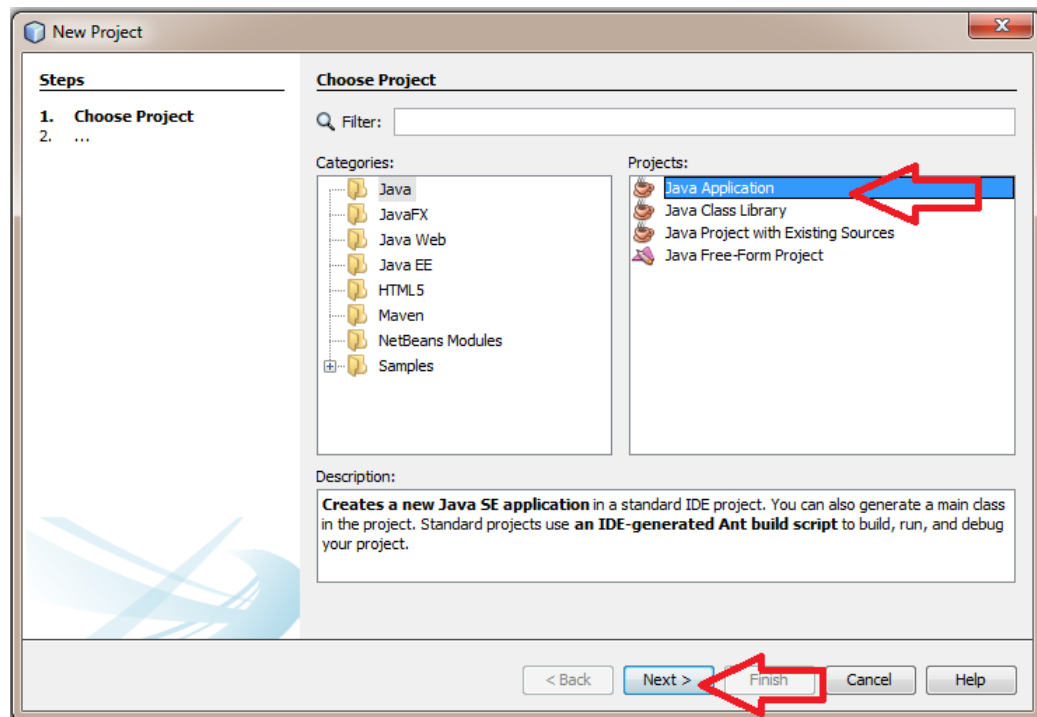
A.2. Manualmente

Descargar el archivo .jar y agregarlo al folder /lib de la aplicación

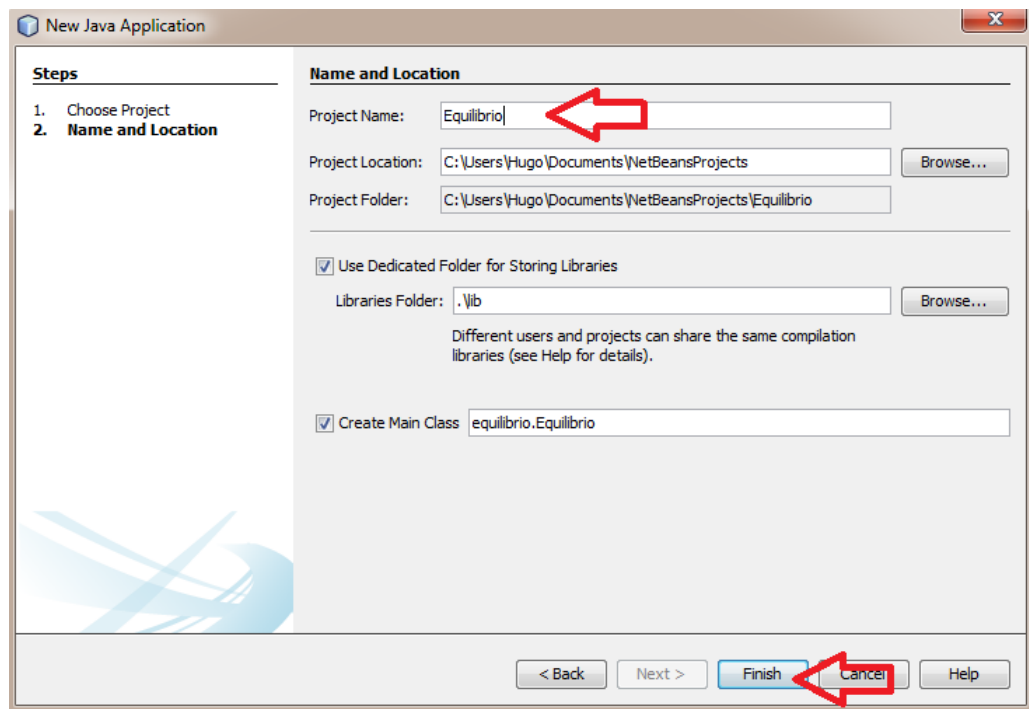
1. Desde la página oficial de EQ PRO(ingenieria-eqpro.rhcloud.com) se puede descargar el archivo jar.
2. Crear un nuevo proyecto desde Netbeans.



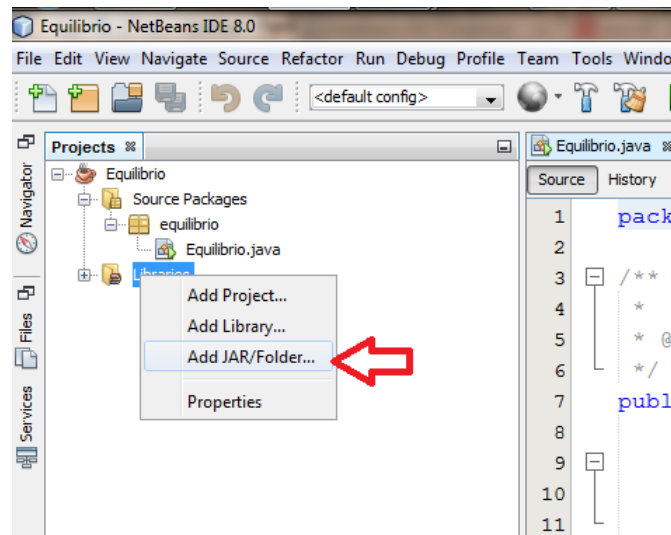
3. Elegir el tipo de aplicación java.application



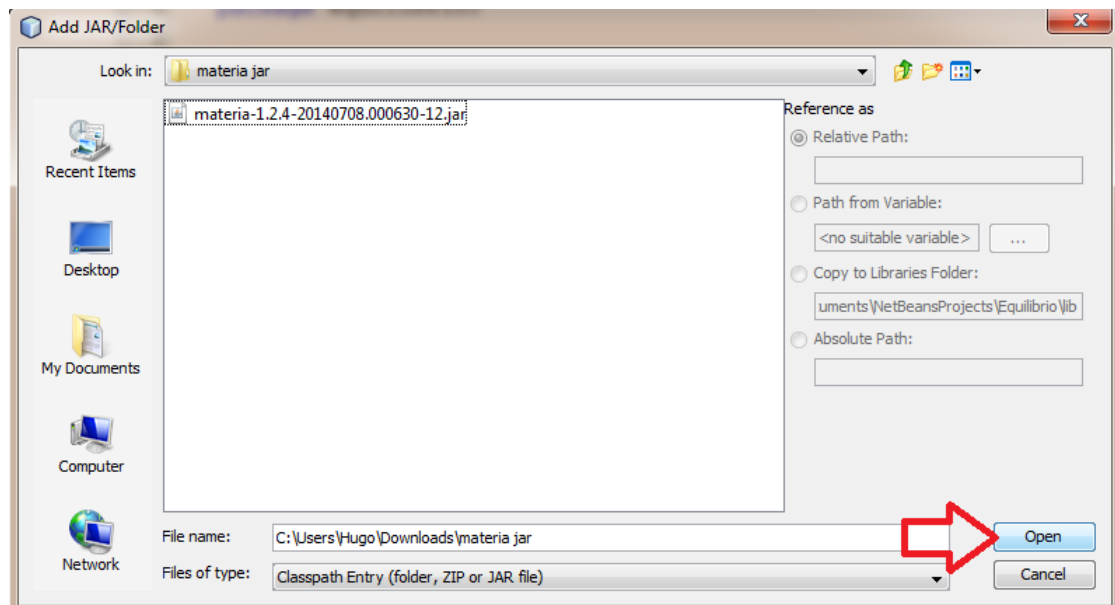
4. Asignar el nombre del proyecto



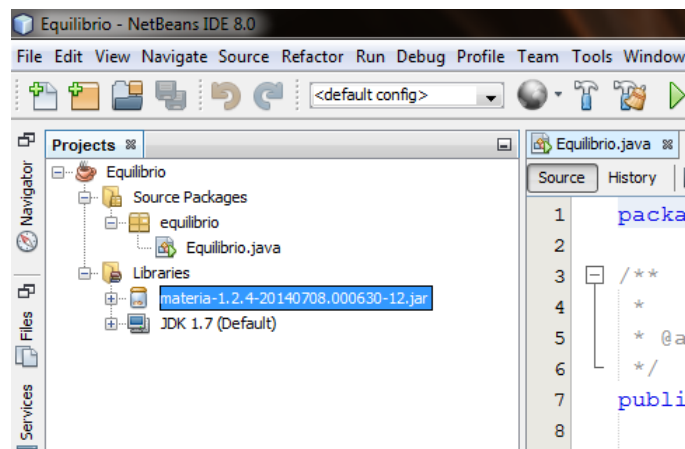
5. En la pestaña proyectos de netbeans, con click derecho en la carpeta “Libraries” elegir la opción “Agregar jar/Folder “.



6. Navegar entonces hasta la ruta donde se descargo el archivo jar.



Se puede ver entonces la librería agregada al proyecto.



A.3. Maven

Desde maven utilizando el archivo pom.xml. Crear nuevo proyecto :new project Elegir la categoría -¿Maven -¿Java Applicationmaven java app Elegir nombre del proyecto y dar click en finalizar.maven name Podemos ver en la carpeta del proyecto la siguiente estructura


```
Maven_Equilibrio
|-- pom.xml
'-- src
    -- main
        '-- java
            '-- hugo
                '-- ejemplos
                    '-- maven_equilibrio
```

Abrimos el archivo pom.xml y agregamos las siguientes etiquetas

```
1 <dependencies>
2   <dependency>
3     <groupId>com.github.hugoredon</groupId>
4     <artifactId>materia</artifactId>
5     <version>1</version>
6   </dependency>
7 </dependencies>
```

5- Inmediatamente se ve agregada la dependencia Materia, cuando el proyecto se compile, se descargará el archivo jar automáticamente.

```
maven materia added
```

6. Crear una clase java en cualquier paquete dentro de Source packages.maven create java class

Escribimos dentro de esta clase el mismo código que en la entrada anterior.

A.4. Código

```
1
2  public class Equilibrio {
3      public static void main(String[] args) {
4          Compound agua = new Compound("agua");
5          agua.setCriticalTemperature(647.3);
6          agua.setCriticalPressure(2.212E7);
7          agua.setAcentricFactor(0.344861);
8
9          Cubic cubicEquationOfState = EquationOfStateFactory.pengRobinsonBase();
10         Alpha alphaExpression = AlphaFactory.getStryjekAndVeraExpression();
11
12         HeterogeneousSubstance substance =
13         new HeterogeneousSubstance(cubicEquationOfState, alphaExpression, agua);
14         double pressure = 101325;
15         substance.setPressure(pressure);
16         substance.bubbleTemperature();
17         double temperature = substance.getTemperature();
18
19         System.out.println("(Presión "+pressure+" [Pa])Temperatura de burbuja: " + t
20     }
21 }
```

Ejecutamos el código y el resultado es:

(Presión 101325.0 [Pa])Temperatura de burbuja: 374.5312063949659[K]

Apéndice B

¿Cómo extender o modificar la librería desde el código fuente?

Apéndice C

Solución de la ecuación de estado cúbica

$$z = \frac{PV}{RT} \quad A = \frac{ap}{(RT)^2} \quad B = \frac{bp}{RT} \quad (\text{C.1})$$

$$z^3 - [1 - (u - 1)B] z^2 + [A - uB - uB^2 + wB^2] z - [AB + wB^2 + wB^3] = 0 \quad (\text{C.2})$$

$$\alpha = 1 - (u - 1)B \quad (\text{C.3})$$

$$\beta = A - uB - uB^2 + wB^3 \quad (\text{C.4})$$

$$\gamma = AB + wB^2 + 2B^3 \quad (\text{C.5})$$

$$C = 3\beta - \alpha^2 \quad (\text{C.6})$$

$$D = -\alpha^3 + 4.5\alpha\beta - 13.5\gamma \quad (\text{C.7})$$

$$Q = C^3 + D^2 \quad (\text{C.8})$$

- Si $Q \leq 0$ $\vartheta = \arccos \left[\frac{-D}{\sqrt{-C^3}} \right]$

Líquido $z = \frac{1}{3} \left[\alpha + 2\sqrt{-C} \cos \left(\frac{\vartheta}{3} + 120^\circ \right) \right]$

Vapor $z = \frac{1}{3} \left[\alpha + 2\sqrt{-C} \cos \left(\frac{\vartheta}{3} \right) \right]$

Nota: En caso de que el z del líquido sea menor que B , entonces hay que calcularla como si fuera vapor.

- Si $Q > 0$ $z = \frac{1}{3} \left[\alpha + (-D + \sqrt{Q})^{\frac{1}{3}} + (-D - \sqrt{Q})^{\frac{1}{3}} \right]$

Apéndice D

Ecuaciones

$$P = \frac{RT}{v - b} - \frac{a}{v^2 + ubv + wb^2} \quad (D.1)$$

- P : presión en $[Pa]$.
- v : volumen molar en $[\frac{m^3}{kmol}]$
- a : Es una medida de la atracción entre las partículas. $[\frac{m^5}{kg \cdot s}]$
- b : volumen excluido por un kmol de partículas. $[\frac{m^3}{kmol}]$
- u y w : Son los parámetros diferentes para cada ecuación de estado, ver tabla 5.2
- R : Constante universal de los gases ideales en $\frac{m^3 Pa}{kmol K}$

$$b_i = \Omega_b \frac{RT_{ci}}{p_{ci}} \quad (D.2)$$

$$a_i = \Omega_a \frac{(RT_{ci})^2}{p_{ci}} \alpha_i \quad (D.3)$$

$$z = \frac{PV}{RT} \quad (D.4)$$

$$V = \frac{RT}{zP} \quad (\text{D.5})$$

$$A = \frac{ap}{(RT)^2} \quad B = \frac{bp}{RT} \quad (\text{D.6})$$

$$z^3 - [1 - (u - 1)B]z^2 + [A - uB - uB^2 + wB^2]z - [AB + wB^2 + wB^3] = 0 \quad (\text{D.7})$$

Entalpía del gas ideal

$$h^\neq = \sum_{i=1}^{nc} x_i \left[h_i^{ref} + \int_{T_{ref}}^T C_{p_i}^\neq dT \right] \quad (\text{D.8})$$

D.1. Entalpía

$$h = h^\neq + \left[\frac{T(\frac{\partial a}{\partial T}) - a}{b\sqrt{u - 4w}} \right] \ln \left[\frac{2v + b(u + \sqrt{u - 4w})}{2v + b(u - \sqrt{u - 4w})} \right] + pv - RT \quad (\text{D.9})$$

Código D.1 *Some Code*

```

1 private double calculateEnthalpy( double volume){
2     double idealGasEnthalpy = calculateIdealGasEnthalpy();
3     double a = calculate_a_cubicParameter();
4     double b = calculate_b_cubicParameter();
5     double L = cubicEquationOfState.calculateL(volume, b);
6     double partial_aPartial_temperature = partial_aPartial_temperature( );
7
8     return idealGasEnthalpy + ((partial_aPartial_temperature - a)/b) * L
9     + pressure * volume - Constants.R *temperature;
10 }

```

D.1.1. Entropía

$$s = s^\neq + R \ln \left[\frac{z(v-b)}{v} \right] + \frac{\frac{\partial a}{\partial T}}{b\sqrt{u^2-4w}} \ln \left[\frac{2v+b(u+\sqrt{u-4w})}{2v+b(u-\sqrt{u-4w})} \right] \quad (\text{D.10})$$

$$s^\neq = \sum_{i=1}^{nc} x_i \left[s_i^{ref} + \int_{T_{ref}}^T \frac{Cp_i^\neq}{T} dT - R \ln \left(\frac{p}{p_{ref}} \right) - R \ln x_i \right] \quad (\text{D.11})$$

$$g = h - T * s; \quad (\text{D.12})$$

$$\begin{aligned} \ln \hat{\phi}_i = & -\ln \left(\frac{v-b}{v} \right) + (z-1) \left[\frac{1}{b} \frac{\partial bN}{\partial N_i} \right] + \frac{a}{RTb\sqrt{u^2-4w}} \\ & \left[\frac{1}{b} \frac{\partial bN}{\partial N_i} - \frac{1}{aN} \frac{\partial aN}{\partial N_i} \right] \ln \left[\frac{2v+b(u+\sqrt{u-4w})}{2v+b(u-\sqrt{u-4w})} \right] - \ln z \end{aligned} \quad (\text{D.13})$$

$$P^\circ = P_c 10 \left[\left(-\frac{7}{3} \right) (1+\omega) \left(\left(\frac{T_c}{T} \right) - 1 \right) \right] \quad (\text{D.14})$$

D.1.2. Funciones objetivo para la estimación de parámetros

Error relativo para el dato i

$$ER_i = \frac{P_i^{calculada} - P_i^{experimental}}{P_i^{experimental}} \quad (\text{D.15})$$

Error total

$$E_{total} = \sum_{i=1}^{nd} ER_i^2 \quad (\text{D.16})$$

nd : numero de datos

D.1.3. Ecuaciones para el cálculo de la presión de vapor

Ecuación 101

$$P = e^{\left(A + \frac{B}{T} + C \ln T + DT^E\right)} \quad (\text{D.17})$$

Apéndice E

Herramientas para la página de internet

- Java Enterprise Edition
- Java Server Faces
- Hibernate
- MySQL
- Enterprise Java Beans 3
- Wildfly
- Openshift
- HTML, CSS
- Javascript JQuery

E.1. Expresiones de α

Soave^[9]

$$\alpha^{1/2} = 1 + m \left(1 - \sqrt{T_r} \right) \quad (\text{E.1})$$

$$m = 0.48508 + 1.55171\omega - 0.15613\omega^2 \quad (\text{E.2})$$

Peng and Robinson^[8]

$$\alpha^{1/2} = 1 + m \left(1 - \sqrt{T_r} \right) \quad (\text{E.3})$$

$$m = 0.37464 + 1.54226\omega - 0.2699\omega^2 \quad (\text{E.4})$$

Mathias^[5]

▪ $T < T_C$

$$\alpha^{1/2} = 1 + m \left(1 - \sqrt{T_r} \right) - A(1 - T_r)(0.7 - T_r) \quad (\text{E.5})$$

$$m = 0.48508 + 1.55191\omega - 0.15613\omega^2 \quad (\text{E.6})$$

▪ $T > T_c$

$$\alpha = \exp \left[\left(\frac{c-1}{c} \right) (1 - T_r^c) \right] \quad (\text{E.7})$$

$$c = 1 + \frac{m}{2} + 0.3A \quad (\text{E.8})$$

Stryjek and Vera(PRSV)^[11]

$$\blacksquare T < T_C$$

$$\alpha^{1/2} = 1 + \kappa_0 \left(1 - \sqrt{T_r}\right) - \kappa_1 (1 - T_r) (0.7 - T_r) \quad (\text{E.9})$$

$$\kappa_0 = 0.378893 + 1.4897153\omega - 0.17131848\omega^2 + 0.0196554\omega \quad (\text{E.10})$$

$$\blacksquare T > T_c$$

$$\alpha^{1/2} = 1 + \kappa_0 \left(1 - \sqrt{T_r}\right) \quad (\text{E.11})$$

Adachi and Lu^[1]

$$\alpha = A \cdot 10^{B(1-T_r)} \quad (\text{E.12})$$

Soave^[10]

$$\alpha = 1 + (1 - T_r) \left(A + \frac{B}{T_r}\right) \quad (\text{E.13})$$

Melhem, et al.^[7]

$$\ln \alpha = A (1 - T_r) + B \left(1 - \sqrt{T_r}\right)^2 \quad (\text{E.14})$$

Androulakis,et al.^[2]

$$\blacksquare T < T_C$$

$$\alpha = 1 + A \left(1 - T_r^{2/3}\right) + B \left(1 - T_r^{2/3}\right)^2 + C \left(1 - T_r^{2/3}\right)^3 \quad (\text{E.15})$$

- $T > T_c$

$$\alpha = e^{A(1-T_r^{2/3})} \quad (\text{E.16})$$

Mathias and Copeman.^[3,6]

- $T < T_c$

$$\alpha^{1/2} = 1 + A \left(1 - \sqrt{T_r}\right) + B \left(1 - \sqrt{T_r}\right)^2 + C \left(1 - \sqrt{T_r}\right)^3 \quad (\text{E.17})$$

- $T > T_c$

$$\alpha^{1/2} = 1 + A \left(1 - \sqrt{T_r}\right) \quad (\text{E.18})$$

1

Yu and Lu^[15]

- $T < T_c$

$$\log_{10} \alpha = (A + BT_r + CT_r^2) (1 - T_r) \quad (\text{E.19})$$

- $T > T_c$

$$\log_{10} \alpha = (A + B + C) (1 - T_r) \quad (\text{E.20})$$

¹La ecuación no se incluyó en el trabajo original de Mathias and Copeman;^[6] esta expresión fue incorporada en el trabajo de Dahl and Michelsen^[3]

Stryjek and Vera^[12]

$$\blacksquare T < T_c$$

$$\alpha^{1/2} = 1 + \kappa \left(1 - \sqrt{T_r}\right) \quad (\text{E.21})$$

$$\kappa = m + \left[A + B(C - T_r) \left(1 - \sqrt{T_r}\right)\right] \left[\left(1 + \sqrt{T_r}\right) (0.7 - T_r)\right] \quad (\text{E.22})$$

$$m = 0.378893 + 1.4897153\omega - 0.17131848\omega^2 + 0.0196554\omega^3 \quad (\text{E.23})$$

$$\blacksquare T > T_c$$

$$\alpha^{1/2} = 1 + m \left(1 - \sqrt{T_r}\right) \quad (\text{E.24})$$

Twu^[13]

$$\alpha = T_r^{N(M-1)} \exp \left(L \left(1 - T_r^{NM}\right) \right) \quad (\text{E.25})$$

Twu^[14]

$$\blacksquare T < T_c$$

$$\alpha = \alpha^{(0)} + \omega \left(\alpha^{(1)} - \alpha^{(0)} \right) \quad (\text{E.26})$$

$$\text{Para } \alpha^{(0)}$$

$$L = 0.196545 \quad M = 0.906437 \quad N = 1.26251 \quad (\text{E.27})$$

$$\text{Para } \alpha^{(1)}$$

$$L = 0.704001 \quad M = 0.790407 \quad N = 2.13076 \quad (\text{E.28})$$

- $T > T_c$

$$\alpha = \alpha^{(0)} + \omega (\alpha^{(1)} - \alpha^{(0)}) \quad (\text{E.29})$$

Para $\alpha^{(0)}$

$$L = 0.358826 \quad M = 4.23478 \quad N = -0.2 \quad (\text{E.30})$$

Para $\alpha^{(1)}$

$$L = 0.0206444 \quad M = 1.22942 \quad N = -8.0 \quad (\text{E.31})$$

GCEOS (AspenHYSYS)

$$\alpha^{1/2} = 1 + \kappa (1 - \sqrt{T_r}) \quad (\text{E.32})$$

$$\kappa = \kappa_0 + [\kappa_1 + (\kappa_2 - \kappa_3 T_r) (1 - T_r^{\kappa_4})] \left[(1 + \sqrt{T_r}) (0.7 - T_r) \right] T^{\kappa_5} \quad (\text{E.33})$$

$$\kappa_0 = A + B\omega + C\omega^2 + D\omega^3 \quad (\text{E.34})$$

Bibliografía

- [1] Adachi, Y.; Lu, B.C. Y.: *Simplest equation of state for vapor-liquid equilibrium calculation: a modification of the van der Waals equation*. AIChE Journal, páginas 30, 991., 1984.
- [2] Androulakis, I.P.; Kalospiros, N.S.; Tassios D.P.: *Thermophysical properties of pure polar and nonpolar compounds with a modified VdW-711 equation of state*. Fluid Phase Equilibria, páginas 45,135, 1989.
- [3] Dahl, S.; Michelsen, M.: *High-pressure vapor-liquid equilibrium with a UNIFAC based equation of state*. AIChE Journal, páginas 47,189, 1990.
- [4] Gosling, James: *Java: an Overview*. 1995.
- [5] Mathias, P.M.: *A versatile phase equilibrium equation of state*. Ind. Eng. Chem. Process Des. Dev., páginas 22,385., 1983.
- [6] Mathias, P.M; Copeman T.W.: *Extension of the Peng-Robinson equation of state to complex mixture: Evaluation of the various forms of the local composition concept*. Fluid Phase Equilibria., páginas 13,91, 1983.
- [7] Melhem, G.A.; Saini, R.; Goodwin M.: *A modified Peng-Robinson equation of state*. Fluid Phase Equilibria, páginas 47,189, 1989.

- [8] Peng, D.Y. y D.B. Robinson: *A new two constant equation of state*. Ind. Eng. Chem. Fundam, páginas 15,59, 1976.
- [9] Soave, G.: *Equilibrium constants from a modified Redlich-Kwong equation of state*. Chemical Engineering Science, páginas 27,1197, 1972.
- [10] Soave, G.: *Rigorous and simplified procedures for determining the pure component parameters in the Redlich-wong-Soave equation of state*. Chemical Engineering Science, páginas 35,1725, 1983.
- [11] Stryjek, R.;Vera, J.H.: *PRSV: An improved Peng-Robinson equation of state for pure compounds and mixture*. The Canadian Journal of Chemical Engineering, página 323, 1986.
- [12] Stryjek, R.;Vera, J.H.: *PRSV: An improved Peng-Robinson equation of state with new mixing rules for strongly nonideal mixtures*. The Canadian Journal of Chemical Engineering, páginas 64,334, 1986.
- [13] Twu, C.H.; Black, D.; Cunningham J.R.;Coon j.E.: *A cubic equation of state with new alpha function and a new mixing rule*. Fluid Phase Equilibria, páginas 69, 33–50, 1991.
- [14] Twu, C.h.; Sim, W.D.; Tassone V.: *A versatil liquid activity model for SRK, PR and a new cubic equation of state TST*. Fluid Phase Equilibria, páginas 194–197,385–399, 2002.
- [15] Yu, J.-M.; Lu, B.C. Y.: *A three-parameter cubic equation of state for axymmetric density calculations*. Fluid Phase Equilibria, páginas 34,1, 1987.