Simulación de una columna de destilación simple para una mezcla binaria de compuestos utilizados como biodiesel. Comparación entre un nuevo modelo propuesto y simuladores comerciales de ingeniería química.

AUTOR: DESIRE SANTANA CASTELLANO
TUTOR: JUAN ORTEGA SAAVEDRA
COTUTOR: FERNANDO ESPIAU CASTELLANO

Julio, 2010

Quiero expresar mi más sincero agradecimiento al
Dr. Don Juan Ortega Saavedra por su ayuda incondicional
y al Dr. Don Fernando Espiau Castellano
por su colaboración y su estimado apoyo.

Índice

1. ANTECEDENTES.

La creciente preocupación por el cambio climático cuyos principales responsables parecen ser las emisiones de gases de efecto invernadero junto con la incertidumbre de disponer de combustible fósil en el futuro, ha provocado un aumento de la producción de biocombustibles, destacando especialmente el biodiesel. A lo dicho hay que sumar el hecho de que, en el caso de España la dependencia energética del exterior es muy fuerte, importándose un 80 de la energía que se consume en el país. Esto hace que nuestra economía sea bastante vulnerable a los volátiles cambios en los precios internacionales del petróleo.

En España existen más de 44 plantas de biodiesel y se prevé que se construyan 25 más entre los años 2009-2011. En este sentido Canarias sería una región deficitaria puesto que en la actualidad solo se contempla la posibilidad de construir una planta de biodiesel. Para ello se necesitan procesos de separación factibles, dentro de los procesos químicos para la obtención de los integrantes del biodiesel en su más alta pureza. Por ello se ha creído conveniente un proyecto de simulación de procesos de separación para que la obtención de biodiesel sea eficaz y capaz de cubrir un buen porcentaje de la demanda que, de dicho combustible, se tendrá en las islas en los próximos años, reduciendo el consumo exterior de los combustibles fósiles.

Tal como se ha comentado en el párrafo anterior, el objetivo del presente proyecto es diseñar un proceso de separación que presente unas condiciones factibles para la obtención de uno de los integrantes del biodiesel, con la más alta pureza.

Se plantea, por tanto, como objetivo de este proyecto de Fin de Máster la simulación de una columna simple de destilación para la obtención de la separación del compuesto éster + metanol.

2. INTRODUCCIÓN.

Para una mayor comprensión de la finalidad de este proyecto, se procederá a definir las diferentes partes que integran el mismo. Para ello se procederá a explicar lo que se conoce como biodiesel, así como, los diferentes procesos de separación aplicados en la ingeniería química.

El objetivo de este trabajo consiste en implementar una interface de usuario con un método aplicado y comparar los resultados obtenidos con los programas comerciales utilizados para la simulación de sistemas de destilación.

A continuación se comienza con un poco de historia del biodiesel, ya que los datos aplicados en este trabajo pertenecen a la familia de los mismos.

3. BIODIESEL.

3.1. INTRODUCCIÓN.

Desde comienzos de la Revolución Industrial hasta el momento actual, los combustibles fósiles (fundamentalmente el petróleo y sus derivados) han sido la base en la que se ha sustentado la economía mundial.

Sin embargo, su carácter no renovable y las perspectivas actuales de agotamiento que presentan sus reservas, catalizadas por el constante aumento de su demanda está conduciendo a la búsqueda de fuentes alternativas que puedan aminorar en gran medida la dependencia que de estos combustibles tenemos en la actualidad.

Además el uso de los mismos tiene un impacto negativo sobre el medio ambiente, entre otras cosas porque sus emisiones contaminan la atmósfera y generan gases de efecto invernadero. Ello incide no solo en nuestra salud sino que, según la mayor parte de la comunidad científica internacional, es una de las causas principales del cambio climático que está sufriendo nuestro planeta.

Ello ha motivado la creación de Ministerios u oficinas de Medio Ambiente y Desarrollo Sostenible en gran número de países, y la adopción conjunta de medidas para frenar la contaminación ambiental y mejorar las condiciones de sostenibilidad como ha sido la firma del Protocolo de Kyoto, en el año 1997, que entró en vigor en febrero de 2005.

En la Unión Europea (UE), el transporte es responsable del 21% de las emisiones de gases de efecto invernadero y, representando más del 30% del total de consumo de energía depende en un 98% de los combustibles fósiles. El incremento de este sector es la principal causa de que la Unión Europea no cumpla con los objetivos y compromisos del Protocolo de Kyoto; de hecho se espera que el 90% del incremento de las emisiones de CO2 entre 1990 y 2010 se deban a este sector. [Biofuels Research Advisory Council UE, 2006].

Teniendo en cuenta los factores que inciden en el cambio climático, el continuado ncremento de los precios del petróleo y sus derivados así como la preocupación por garantizar su suministro, la utilización de biomasa, y en particular de biocombustibles (biodiesel y bioetanol), para usos energéticos tiene cada vez mayor interés.

Los biocarburantes son un sustituto directo e inmediato de los combustibles líquidos utilizados en el transporte y pueden ser fácilmente integrados en los sistemas logísticos actualmente en operación. Reemplazar un porcentaje de gasóleo y gasolinas de automoción por biocombustibles (biodiesel o bioetanol) es la forma más sencilla y barata de mejorar las emisiones generadas en el sector.

En la actualidad en la Unión Europea se producen 35 millones de metros cúbicos de biocombustibles (bioetanol y biodiesel, principalmente). Esta cifra está muy alejada de los objetivos fijados por el Protocolo de Kioto (Directiva 2003/30 de mayo 2003) que estaban en el 2% del consumo de carburantes en la UE. El objetivo para el 2010 es alcanzar una penetración de mercado del 5,75%.

La UE tiene un gran potencial para la producción de biocombustibles. A finales del año 2005 ésta se situaba en la Europa de los 25 (UE 25) en cerca de 2 Mtep (millones de toneladas equivalentes). Se estima que entre el 4% y el 13% del total de tierra destinada a agricultura en la UE será necesaria para producir la cantidad de biocombustibles que permita alcanzar el nivel de sustitución de los combustibles fósiles empleados en el transporte y lograr los objetivos marcados en la Directiva anteriormente citada.

Se prevé que en el año 2030 un cuarto de los combustibles empleados en el transporte provendrán de biocombustibles [Biofuels Research Advisory Council UE, 2006].

En cuanto a España, es el primer productor europeo de bioetanol, debido a la producción de etil-tercbutil éter (ETBE) que se utiliza como aditivo en la gasolina. Sin embargo en relación al biodiesel, los indicadores señalan que, al contrario que en el resto de la UE y a pesar que hay varias plantas de producción y otras están en fase de construcción o estudio, se encuentra retrasada a la hora de alcanzar sus objetivos.

Según datos de 2004 la producción de biodiesel en España se sitúa en 15.000 t/año, muy lejos de los datos del primer productor Europeo (Alemania con 1.035.000 t/año). El mercado existente para los cereales, su precio actual y las exenciones fiscales hacen económicamente rentable su utilización para la producción de bioetanol. Los cultivos de colza o girasol necesarios para obtención del biodiesel requieren mayor superficie, lo que presenta mayores dificultades para su desarrollo. La utilización de aceites usados es una alternativa muy atractiva que contribuye también a la eliminación de residuos, aunque requiere desarrollar una adecuada logística para su recogida.

Con todo ello, el Ministerio de Industria, Turismo y Comercio ha concedido una subvención destinada a fomentar en el ámbito de biodiesel la cooperación estable público-privada en investigación, desarrollo e innovación, mediante la formación de consorcios estratégicos nacionales de investigación técnica (proyectos CENIT). El objetivo de este consorcio liderado por Repsol YPF S.A. es la investigación y el desarrollo sobre la obtención y utilización de biodiesel, minimizando el uso de combustibles fósiles y favoreciendo otros de origen renovable, reduciendo la dependencia energética y contribuyendo a la disminución de gases de efecto invernadero. El programa CENIT se enmarca dentro de las iniciativas del Plan Ingenio 2010.

Además, desde 1998 España ha aumentado su consumo en diesel que ha crecido desde las cerca de 14.349 ktep de ese año hasta las más de 22.000 ktep en 2004. Éste hecho junto al objetivo que tiene nuestro país de cumplir el Protocolo de Kioto, incentiva la producción de biodiesel.

Ello traerá consecuencias beneficiosas, no solo en el ámbito ambiental donde se reducirán a la mitad las emisiones de CO2 y a cero las de óxidos de azufre, sino que también servirá para crear nuevas oportunidades y puestos de trabajo en sectores como el de la industria y la agricultura.

3.2. EVOLUCIÓN HISTÓRICA DEL BIODIESEL.

El primer motor Diesel de la historia funcionaba con aceite de cacahuete. Su creador, el inventor alemán Rudolf Diesel, lo presentó en la Exposición Universal de París de 1900 como un "motor de aceite" y pretendía con él potenciar la agricultura como fuente de energía. Fue pionero en desarrollar el biodiesel aunque estrictamente hablando no se puede considerar al aceite de cacahuete o de cualquier semilla oleaginosa como biodiesel, debido a que para eso es necesario un proceso de transesterificación de los aceites vegetales, algo que en aquella época se desconocía por completo.

Rudolf Diesel se dio cuenta de la importante función que el combustible fabricado a partir de biomasa desempeñaría en el futuro para el funcionamiento de los motores. Sin embargo, los fabricantes de biocarburantes de la época encontraron el proceso demasiado costoso y la industria del petróleo comenzó a expandirse de forma agresiva debido a que el diesel obtenido a partir del petróleo era más barato.

Los aceites vegetales eran mucho más viscosos que éste combustible y originaban diversos problemas, tales como taponamiento de filtros y de inyectores, depósitos de carbón en la cámara de combustión, excesivo desgaste del motor, degradación del aceite lubricante por polimerización,…. etc. Por estas causas no se podían utilizar directamente en los motores diesel, por lo que tuvieron, durante los años 20 del pasado siglo, modificados por los fabricantes para permitir el uso de petróleo. Como resultado, el combustible vegetal se dejó de lado y las instalaciones de su fabricación cayeron en decadencia.

A pesar de esta caída, los motores han seguido utilizando aceite vegetal como combustible en momentos en los que se han presentado problemas de abastecimiento de productos petrolíferos, especialmente durante las dos guerras mundiales; y en el período comprendido entre los años veinte y cincuenta, en países como el Reino Unido, Francia, Alemania, Brasil, China y Japón.

Durante la crisis energética (1973-1979) disminuyó la oferta de petróleo elevándose su precio de forma exorbitada. Se planteó entonces la urgente necesidad de ahorrar energía y utilizar recursos energéticos renovables, reiniciándose entre otras la investigación en el campo de los biocombustibles líquidos de origen vegetal.

A comienzos de la crisis (año 1973), al científico Chavanne de la Universidad de Bruselas en Bélgica se le concede la primera patente de obtención de biodiesel. Esta se basaba en el proceso de transesterificación o alcoholisis del aceite vegetal, que se transformó para su uso como combustible. El proceso partía del tratamiento de los aceites vegetales con alcohol etílico y metanol para posteriormente sustituir la

glicerina con alcohol después de la eliminación de los ácidos grasos de ésta. Este proceso de producción continuó hasta mediados los años setenta del pasado siglo.

Con la transesterificación se rebajaba la viscosidad de los aceites vegetales hasta valores cercanos a las de los gasóleos de origen fósil, por lo que podían usarse en los motores diesel disminuyendo con creces los problemas derivados del taponamiento, desgaste y acumulación de carbón.

En el periodo comprendido entre 1977 y 1989 se desarrolla el proceso de obtención del biodiesel en varias partes del mundo. En 1977, Parente (científico brasileño) patentó un proceso que empleaba etanol durante la transesterificación. Este proceso ya ha sido reconocido y aceptado por la industria automovilística.

Otro producto "bio-queroseno" (Tecbio Parente) también ha sido patentado y certificado por Boeing y la NASA para su utilización como biodiesel de aviación. En 1979, se desarrolló en Sudáfrica un proceso que permitía obtener biodiesel a partir de aceite de girasol, sin embargo no fue hasta 1983 cuando se obtuvo un biodiesel adecuado para su uso en motores de automóviles, aceptado en todo el mundo.

A comienzos de los años noventa se crearon en Europa varias plantas de biodiesel debido a la creciente preocupación social por proteger el medio ambiente. En 1997 un gran número de naciones firmaron el Protocolo de Kyoto dentro del Convenio Marco sobre Cambio Climático de la ONU (UNFCCC), motivando un resurgimiento del interés por este combustible en diferentes partes del planeta. Durante las últimas décadas, en los países industrializados, la tecnología de producción del biodiesel ha sido desarrollada satisfactoriamente y su uso en motores diesel ha sido probado con ostensible éxito. No obstante, aún persisten dos aspectos no resueltos que limitan una mayor expansión y difusión de su uso: por un lado, el alto costo de los aceites vegetales para la producción de biodiesel cuando se utilizan cultivos oleaginosos convencionales; y por otro, la logística para el aprovisionamiento de los insumos en el caso de la utilización de aceites comestibles usados.

Hoy en día países como Alemania, Austria, Canadá, Estados Unidos, Francia, Italia, Malasia y Suecia son pioneros en la producción, ensayo y uso de biodiesel en automóviles.

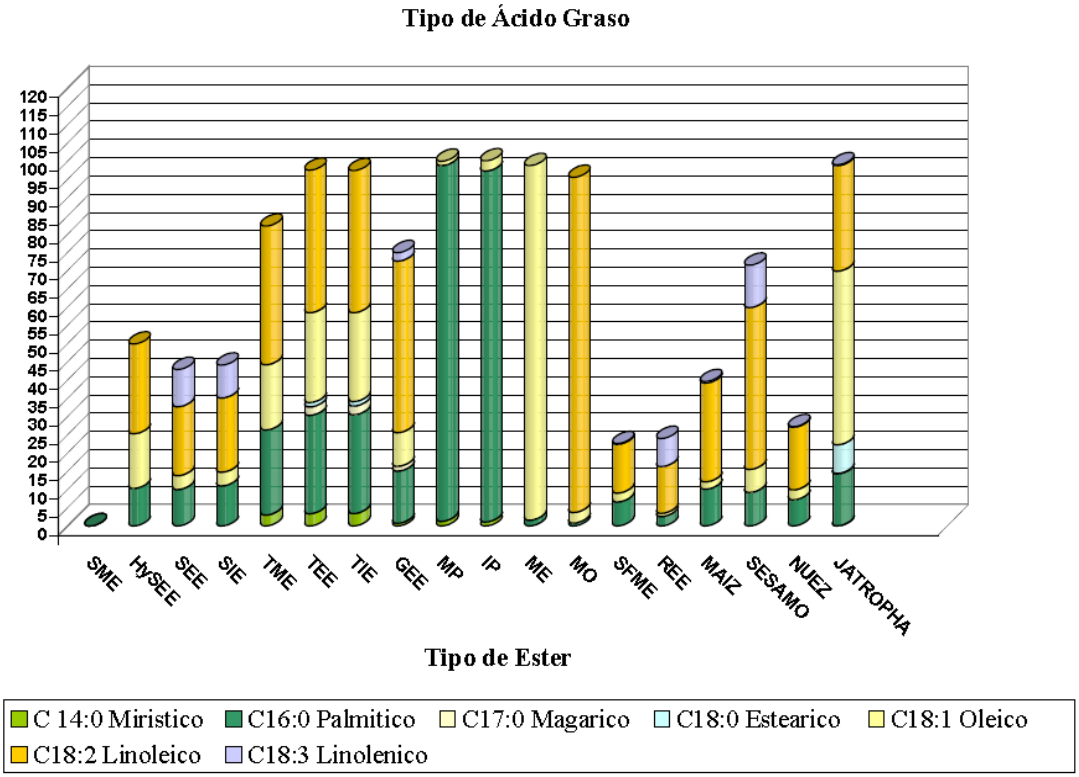
3.3. DEFINICIÓN DE BIODIESEL.

La "American Standards for Testing and Materials" define al biodiesel como: "ésteres monoalquílicos de ácidos grasos de cadena larga derivados de lípidos renovables, tales como aceites vegetales, que se emplea en los motores de ignición de compresión (motores diesel) o en calderas de calefacción."

Se trata por tanto de un biocombustible sintético líquido cuyo principal uso es como sustituto parcial o total de petrodiesel o gasóleo obtenido del petróleo.

Los componentes básicos de los aceites y las grasas son los triglicéridos, los cuales provienen tanto de ácidos grasos saturados como insaturados. Dependiendo de éstas y otras características del origen de los triglicéridos se obtendrán un biodiesel con unas cualidades u otras.

En la **gráfica 1** se presentan diferentes composiciones de los ácidos grasos más comunes presentes en los aceites.



Gráfica 1: Ácidos grasos comunes en aceites vegetales.

Donde:

- **SME :** Ester metilico de la soja.
- **HySEE:** ester etilico hidrogenado de la soja
- **SEE**: ester etililico de la soja
- **SIE:** Ester isopropilico de la soja.
- **TM**E: Ester metilico del sebo de la vaca.
- **TEE:** ester etilico del sebo de la vaca.
- **TIE:** ester isopropilico del sebo de la vaca.
- **GEE:** ester etilico de la grasa.
- **MP:** Metil palmitato.
- **IP:** Isopropil palmitato.
- **ME:** Metil estearato.
- **MO:** Metil oleato.
- **SFME:** Ester metilico de girasol de alto oléico.
- **REE:** ester etilico de la colza.

3.4. PROPIEDADES DEL BIODIESEL.

Las propiedades del biodiesel son parecidas a las del gasóleo de automoción, en cuanto a densidad y número de cetanos.

En cuanto a las propiedades de cada uno aparecen expuestas en la **Tabla 1.1**:

Tabla 1.1: Propiedades físico-químicas de biodiesel y diesel

Datos Físico-químicos	Biodiesel	diesel
Densidad a 20º C (kg/m^3)	870/890	840
Viscosidad a 40ºC	3,5/4,5	3
Poder calorífico inferior (MJ/Kg)	36/39	43
Número de cetano	49/54	48/51
Punto de ebullición ºC	190-340	180-335
Punto de ignición ºC	148,89	51,7
Punto de inflamación ºC	120-170	60-80
Punto de congelación ºC	0/-5	-20
Relación estequiométrica aire/combustible p/p	13,8	15

De la misma manera se pueden extraer las siguientes conclusiones:

- La densidad y viscosidad cinemática del biodiesel aumentan con respecto al gasoil.
- El punto de fusión es más favorable para el gasoil frente al biodiesel. Esto es sobre todo importante en las regiones frías.

- El índice de cetano, que guarda relación con el tiempo que transcurre entre la inyección del carburante y el comienzo de su combustión, es mayor en el metil éster que en el gasoil. Un menor índice de cetano implica un retraso de la ignición.

- El poder calorífico es menor en el biodiesel, por lo que su potencia es inferior, (en torno al 5%) lo que conlleva a un consumo de combustible ligeramente superior al del diesel tradicional. Sin embargo, este inconveniente se ve compensado por su mayor punto de ignición, que reduce el peligro de explosiones por emanación de gases durante el almacenamiento, y su superior lubricidad, que favorece el funcionamiento del circuito de alimentación y de la bomba de inyección.

- El punto de congelación del diésel es de -20ºC frente a los -5ºC del biodiesel. Por ello el uso de biodiesel en motores de automoción requiere de anticongelantes en zonas frías.

3.5. VENTAJAS E INCONVENIENTES DEL USO DEL BIODIESEL.

Incidiendo aún más en lo anterior seguidamente se comentarán las ventajas e inconvenientes que presenta el biodiesel frente al diesel convencional, desde el punto de vista de automoción, socioeconómico y medioambiental.

Entre los inconvenientes de usar biodiesel como sustituto total o parcial del gasoil se destaca lo siguiente:

a) *Inconvenientes en la automoción:*

• Biodiesel puro. En el caso de usar biodiesel puro hay que efectuar unas pequeñas modificaciones técnicas en los motores diésel, como sería modificar el compuesto de la goma y/o cauchos de los manguitos y latiguillos del circuito del combustible. Ello es debido a que el biodiesel 100% tiene la particularidad de disolver la goma. Para vehículos posteriores a los años 90 no se requiere ninguna modificación.

• Alto punto de congelación. Otro problema añadido al uso de biodiesel 100% es su relativamente alto punto de congelación (entre 0ºC y -5ºC), por lo que su uso podría acarrear problemas en regiones frías. En cualquier caso, existen actualmente aditivos que rebajan el punto de congelación hasta -20ºC y cuya aplicación, por lo tanto, elimina dichos riesgos. Tanto el punto de congelación (PC), como el punto de nube (PN), y el punto de obstrucción por filtros fríos (POFF) son desde ligeramente superiores a muy superiores según sea la procedencia del éster (aceite de maíz, oliva, jatropha, etc…). Para un biodiesel procedente de un aceite residual el POFF se encuentra entre -7º y 0ºC, lo cual es insuficiente en invierno. Utilizando un biodiesel de origen animal el POFF es todavía superior.

• Poder calorífico inferior. Debido a que el biodiesel posee un poder calorífico inferior al diesel tiene una potencia en torno a un 5% menor que éste y requiriendo un mayor consumo de combustible.

b) *Inconvenientes socioeconómicas:*

• Alto coste de producción del biodiesel. Según un informe de la Comisión Nacional de la Energía, la producción de un litro de biodiesel varía entre los 17 y los 70 céntimos de euro, en función de la materia prima empleada (aceites usados o aceite sin usar). El coste de producción del biodiesel es, por tanto, mayor que el del combustible derivado del petróleo, al menos por ahora, porque el alza del crudo puede llegar a igualar estos costes a medio plazo.

• Impuestos sobre hidrocarburos. Debido a la diferencia existente en cuanto al valor del impuesto sobre hidrocarburos en los diferentes países de la UE (Gráfico 1.2) la industria española de biocarburantes se encuentra en desventaja competitiva respecto de la de otros que tienen un impuesto más alto, lo que proporciona al biocarburante un mayor margen para competir con los carburantes fósiles.

Con esto el biocarburante sería dedicado a la exportación y no al consumo interno, poniendo en peligro la consecución de los objetivos para 2010. Este es un

hecho ya observado en 2005, donde la cuota de consumo de biocarburantes no se alcanzó (0,44% según fuente APPA frente al 2% previsto), a la vez que se estaba exportando el 60% de la producción de biodiesel.

- Precio y seguridad de abastecimiento de la materia prima. Los productores de biocarburantes actualmente recurren a dos sistemas para comprar su materia prima: por una parte, pueden optar por emplear aceites vegetales usados, que resultan más económicos, pues su precio, por ser un residuo, es mucho menor que el de el aceite sin usar. Sin embargo, presenta una serie de dificultades logísticas para su recogida, control y trazabilidad, por la dispersión de los núcleos de producción (domésticos, restaurantes, etc.), la falta de concienciación de su capacidad contaminante y de su potencial reciclable. Además, en España, dicha recogida no está siendo promovida enérgicamente por la Administración pese a que la Ley 10/98 de Residuos establece la prohibición de verter aceites usados. Otra opción es acceder a la materia prima en los mercados agrícolas, sujetos a una alta volatilidad definida por las cosechas y la demanda para uso alimentario.

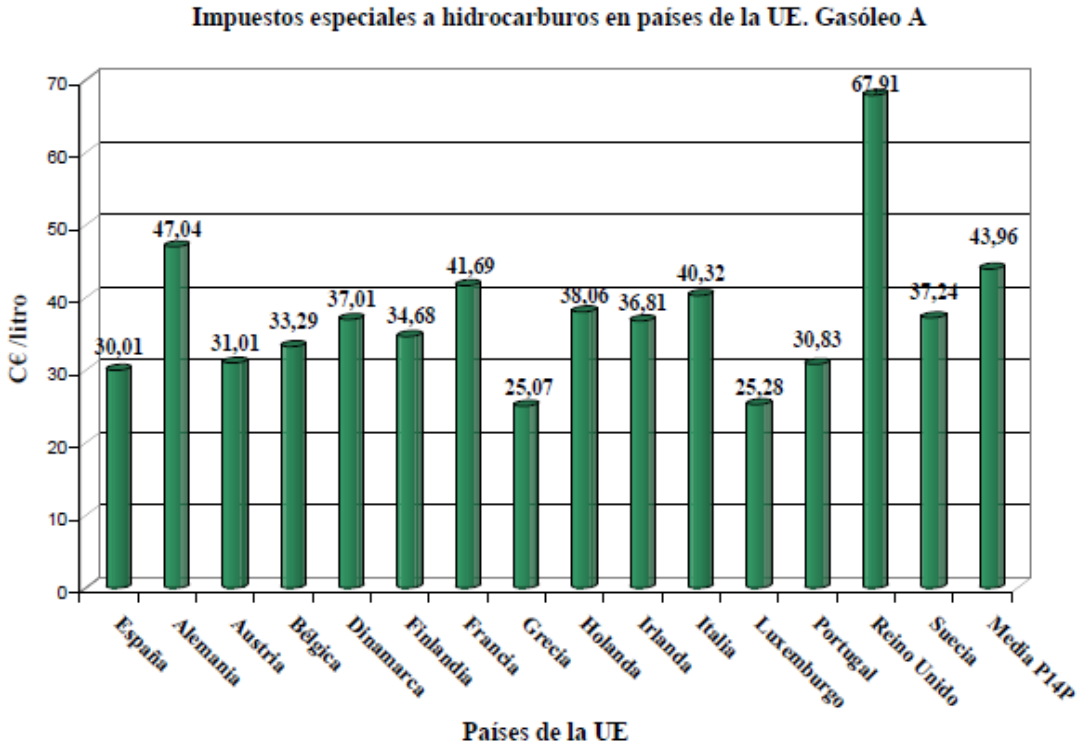


Gráfico 1.2: Impuestos especiales a hidrocarburos en países de la UE. Gasóleo A. Fuente: Asociación de productores de energías renovables (APPA). "Una estrategia de biocarburantes para España (2005-2010)."

El rumbo que han de tomar los cultivos energéticos de cara al futuro será el de una mayor productividad (colza etíope, cardo, jatropha), y una menor competitividad con el sector alimentario, para eliminar la volatilidad de los precios. Sin embargo, la introducción de nuevas especies presenta barreras de tres tipos:

• Técnicas y climatológicas: en función de la adaptación de las mismas a las condiciones españolas.

• Medioambientales: ya que la introducción de una especie nueva puede alterar los ecosistemas actuales.

• Culturales: por el desconocimiento de las mismas por parte de los agricultores y su resistencia a cultivarlas.

• Valorización de subproductos. Uno de los retos al que los productores de biodiesel tienen que hacer frente, es la gestión y la valorización de los subproductos, es decir, desarrollar e integrar los subproductos que se obtienen en la reacción de transesterificación, como son:

-*Los ácidos grasos libres*: que pueden ser destinados a la fabricación de detergentes, champús o cosméticos.

- *Las sales de sulfato o* fosfato (procedentes del catalizador de la reacción): que pueden ser empleadas como fertilizante.

- *La glicerina*. La industria del biodiesel genera, en su proceso de producción, una cantidad de glicerina equivalente al 10% del biodiesel total producido. Es decir, que con los objetivos marcados por la Unión Europea la contribución europea al mercado de glicerina será del orden de millones de toneladas. Por ello, se prevé una saturación del mercado a corto plazo, por lo que parece razonable considerar que el precio de la glicerina seguirá descendiendo en los próximos años con lo cual disminuye considerablemente la rentabilidad de su venta como principal subproducto.

Entre las ventajas de diversa índole cabe destacar las siguientes:

a*) Ventajas del uso en la automoción:*

• Mayor punto de ignición. Lo cual reduce el peligro de explosiones por emanación de gases durante el almacenamiento, mejora el proceso de combustión, permite aumentar la relación de compresión del motor (conlleva un aumento del rendimiento de éste) y produce menos ruido.

• Mayor lubricidad. Favorece el funcionamiento del circuito de alimentación y de la bomba de inyección. Y conlleva una disminución en la necesidad de incluir adictivos en el combustible para mejorar esta propiedad.

• Sustitución total o parcial del gasóleo. Cuando el biodiesel se usa puro se denomina B100 y cuando participa en una mezcla, por ejemplo, 20/80 con gasoil se denomina B20. Pero esto no significa que no se pueda usar en otras proporciones. De hecho en Francia y otros países usan el B2, B5, etc. Al usar mezclas se solucionan problemas como el elevado punto de congelación y la perdida de potencia.

b) *Ventajas medioambientales:*

• El biodiesel es altamente biodegradable (aprox. 21 días). Su origen vegetal lo hace compatible con la naturaleza, y la ausencia de compuestos químicos y sintéticos lo hace inocuo con nuestro medio y en caso de derrame y/o accidente, no pone en peligro ni el suelo ni las aguas subterráneas.

• No contiene azufre. No genera emanaciones de SOx que son los responsables de la lluvia ácida y permite el uso de catalizadores con la consiguiente mejora de la combustión y minimización de los gases de escape. Tampoco contiene ni benceno, ni otras sustancias aromáticas cancerígenas (hidrocarburos aromáticos policíclicos).

• Reducción de contaminantes. El biodiesel puro (100%) en comparación con el diesel, reduce las emisiones de todos los contaminantes atmosféricos, excepto el NOx, por lo cual en el balance general se ve reducido el "smog" potencial. En el siguiente **Gráfico 1.3** se observa los porcentajes de contaminantes que produce el biodiesel puro y la mezcla B20:

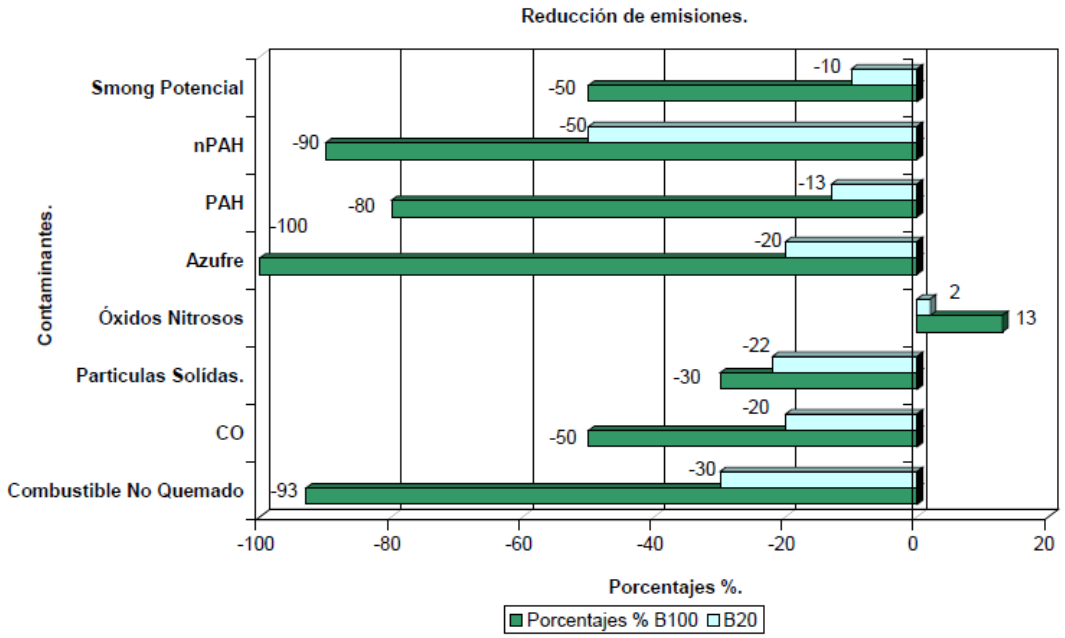


Gráfico 1.3: Reducción de contaminantes del biodiesel puro y en mezcla B20.

En el mismo se observa una notable mejora en el uso del biodiesel en cuanto a reducción de contaminantes:

• Reducción de las emisiones de hollín (hasta casi un 50% desapareciendo el humo negro y olor desagradable). Dado que la molécula de biodiesel aporta, por unidad de volumen, más átomos de oxígeno que lo que aporta el mismo volumen de gasóleo convencional, la presencia de inquemados es menor utilizando biodiesel, dado que hay menos moléculas de carbono elemental (hollín) y de monóxido de carbono (CO).

• Produce, durante su combustión menor cantidad de CO2 que el que las plantas absorben para su crecimiento (ciclo cerrado de CO2). El dióxido de

carbonoCO2 que emite a la atmósfera durante su combustión no provoca ningún incremento adicional, ya que es el mismo que captó la planta oleaginosa utilizada para extraer el aceite durante su etapa de crecimiento.

- Con la mezcla B20 (20% de biodiesel y 80% de diésel) se contamina más que utilizando el biodiesel puro, salvo en las emisiones de óxidos de nitrógeno donde el porcentaje es menor para la mezcla. Sin embargo, la mayoría del biodiesel es vendido como B20 debido a que el biodiesel puro reduce la potencia y la economía del combustible en un 10%. (en otras palabras, se necesitan 1,455 litros de biodiesel por cada litro de diésel reemplazado), y además al ser una mezcla reduce los problemas de congelación que pueden surgir en climas fríos.

c) *Ventajas socioeconómicas*:

- Ahorro de combustibles agotables. En la medida en la que se sustituye el empleo de derivado del petróleo por biodiesel de origen renovable.

- Desarrollo agrícola. Una fuente renovable de producción de combustibles alternativos con origen en la agricultura, permite a la sociedad disponer de una fuente de empleo adicional y de un aprovechamiento de terrenos que, en algunos casos, no pueden ser usados para otros cultivos por restricciones políticas o condiciones del terreno.

- Aprovechamiento de un residuo. La manipulación, tratamiento y evacuación de residuos supone un coste energético y económico a las empresas que lo producen y a la sociedad. Éste también es el caso de los residuos de aceites vegetales. Si éstos son empleados en la elaboración de biodiesel, se consiguen dos objetivos: reducción de los costes por el tratamiento o evacuación del residuo y minimización de los costes relacionados con la posible contaminación ambiental.

Resumiendo el biodiesel constituye una alternativa viable a los derivados del petróleo para la automoción dado que ninguno de los inconvenientes citados supone una barrera insalvable. Además, las contraindicaciones se pueden minimizar si el biodiesel se emplea mezclado con gasóleo.

3.6. ESPECIFICACIONES DE CALIDAD.

Las especificaciones de calidad para que una mezcla de biodiesel pueda ser comercializada quedan definidas en el RD 1700/2003.

Los ésteres metílicos de los ácidos grasos (FAME) denominados biodiesel son productos de origen vegetal o animal, cuya composición y propiedades están definidas actualmente en la norma EN 14214, con excepción del índice de yodo cuyo valor máximo establecido es 140. En la **Tabla 1.2** se incluye los parámetros de control de calidad para el biodiesel.

Parámetro	Norma	Unidades	CEN 14214 Biodiesel
Densidad a 15ºC	EN ISO 12185	g/cm^3	0,860-0,900
Viscosidad Cinemática 40ºC	EN ISO 3104	cSt	3,5-5,0
Punto de inflamación	EN 22719 ISO/DIS 3679	ºC	120 min.
Azufre	EN ISO 14596 pr EN-ISO 20846-84	ppm	10 máx.
Residuo carbonoso (10%)	EN ISO 10370	%	0,30 máx.
Contaminación total	EN 12662	ppm	24 máx.
Agua	EN ISO 12937	ppm	500 máx.
Corrosión al cobre	EN ISO 2160	-	Clase 1

Parámetro	Norma	Unidades	CEN 14214 Biodiesel
Estabilidad a la oxidación	EN ISO 12205 pr EN 14112	mg/l	6 h min
Número de cetano	EN ISO 5165	-	51 min.
Índice de cetano	EN ISO 4264	-	
PAH´s	IP 391	-	
Destilación 65% recogido 85% recogido 95% recogido	EN ISO 3405	ºC	
POFF Invierno Verano	EN 116	ºC	Depende del país
Lubricidad	ISO 12156-1	µm	
Contenido en cenizas	EN ISO 6245	%	
Color	ASTM D 1500		
Transparencia y brillo	ASTM D 4176		
Cenizas sulfatadas	ISO 3987	%	0,02 máx.
TAN	prEN 141042	mgKOH/g	0,5 máx.
Metanol	prEN 14110	%(m/m)	0,2 máx.
Monoglicéridos	prEN 14105	%(m/m)	0,8 máx.
Diglicéridos	prEN 14105	%(m/m)	0,2 máx.
Triglicéridos	prEN 14105	%(m/m)	0,2 máx.
Glicerina libre	prEN 14105-06	%(m/m)	0,02 máx.
Glicerol total	prEN 14105	%(m/m)	0,25 máx.
Contenido en éster	prEN 14103	%(m/m)	min. 96,5
Ésteres metílicos de ác. Linolénico y poli-insaturados	prEN 14103	%(m/m)	máx. 12 Ésteres Metílicos de

			Linolénico.
Índice de Yodo	prEN 14111	-	120 máx.
Fósforo	prEN 14107	mg/kg	10 máx.
GI: Sodio + Potasio GII: Calcio + Magnesio	prEN 14108-09 prEN 14538	mg/kg	5 máx. 5 máx.

Tabla 1.2: Especificaciones de calidad del biodiesel.

Los parámetros de la **Tabla 1.2** proporcionan unas características particulares del biodiesel:

• Punto de inflamación. Este parámetro generalmente se determina para satisfacer temas legales de seguridad. También es útil para conocer si existe una cantidad excesiva de alcohol que no reaccionado en el proceso de obtención.

• Viscosidad. Debe poseer una viscosidad mínima para evitar pérdidas de potencia debidas a fugas en la bomba de inyección y en el inyector. Además, le da características de lubricidad al combustible. Por otra parte también se limita la viscosidad máxima por consideraciones de diseño y tamaño de los motores, y por en las características del sistema de inyección.

• Densidad. Da idea del contenido en energía del combustible. Mayores densidades indican mayor energía térmica y una mejor economía de combustible.

• Cenizas sulfatadas. Los materiales que forman cenizas en un biodiesel se presentan de tres formas:

- Sólidos abrasivos
- Jabones metálicos solubles
- Catalizadores no eliminados en el proceso.

En el caso del diésel, normalmente solo aparecen los primeros o gomas solubles. Tanto los sólidos abrasivos como los catalizadores no eliminados favorecen el desgaste del inyector, bomba de inyección, pistón y anillos, además de contribuir a la formación de depósitos en el motor.

Los jabones metálicos solubles tienen un efecto menor en el desgaste pero pueden afectar más a la colmatación de filtros y depósitos en el motor.

• Azufre. Contribuye al desgaste del motor y a la aparición de depósitos, que varían considerablemente dependiendo en gran medida de la condiciones de funcionamiento del motor. También puede afectar al funcionamiento del sistema de control de emisiones y a límites medioambientales.

• Corrosión a la lámina de cobre. Mediante la comprobación del desgaste de una lámina de cobre se puede observar si existen, en el sistema, compuestos corrosivos y/o presencia de ácidos que puedan atacar al cobre o a sus aleaciones.

• Numero de cetano. Es una medida de la calidad de ignición de un combustible e influye en las emisiones de humos y en la calidad de la combustión. Éste número depende del diseño y tamaño del motor, de la variaciones de carga y de la velocidad y condiciones de arranque. Un bajo número de cetano conlleva a ruidos en el motor, prolongando el retraso de la ignición y aumentando el peso molecular de las emisiones.

• Índice de yodo. Indica la tendencia a la oxidación de un biodiesel porque da idea del grado de instauraciones que poseen sus ésteres.

• Punto de nube. Indica la temperatura a la cual empiezan a precipitar ciertos compuestos del combustible (parafinas, materia insaponificable,…). Es una medida muy importante a tener en cuenta cuando se usa el motor en climas fríos. El valor debe ser definido por el usuario, ya que depende del clima en el que se utilice el motor.

• Agua y sedimentos. El agua se puede formar por condensación en el tanque de almacenamiento. La presencia de agua y sólidos de desgaste normalmente pueden colmatar filtros y darle al combustible unas propiedades de lubricidad menores. El biodiesel puede absorber hasta 40 veces más agua que el diesel.

El agua puede provocar dos problemas en el motor:

1) Corrosión en sus componentes, generalmente herrumbre. El agua se acidifica y acaba atacando a los tanques de almacenamiento.

2) Contribuye al crecimiento de microorganismos (fungi, bacterias,…). Forma lodos y limos que pueden colmatar los filtros. Además, algunos de estos microorganismos pueden convertir el azufre que posea el combustible en acido sulfúrico, que indudablemente corroerá la superficie metálica del tanque.

El agua se puede presentar en el tanque de dos formas:

a) Disuelta en el combustible. La cantidad de agua depende de la solubilidad de ésta en el biodiesel.

b) Separada de la fase de combustible en forma libre. La cantidad de la misma depende de cómo se manipule y trasporte el combustible.

Los sedimentos pueden ser debidos principalmente a un mal proceso de purificación del combustible o contaminación. Afectan principalmente a la temperatura de cristalización y al número de cetano.

• Residuo carbonoso. Da una idea de la tendencia del combustible a formar depósitos carbonosos. Normalmente para el diesel se suele utilizar el 10% que queda en la destilación, pero debido a que el biodiesel tiene un perfil muy diferente de destilación (en un pequeño rango de temperaturas se destila toda la muestra, ya que

posee una distribución de moléculas diferentes muy pequeñas) se debe utilizar el 100% de la muestra. También se puede obtener información, aparte de la contaminación (glicerina libre y total), de la calidad de purificación del biodiesel cuando se fabrica.

• Destilación. Indica la temperatura máxima a la que se debe evaporar el combustible a unas condiciones de presión y temperatura.

- El biodiesel a la temperatura de 360ºC tiene que estar el 90% destilado, según la norma ASTM D1160.
- El diesel a la temperatura de 360ºC tiene que estar el 95% destilado, según la norma ASTM D86.

• Número ácido, TAN. Determina el nivel de ácido grasos, ó generados por degradación, que se presentan en el combustible. Si posee un alto grado de acidez se formarán una cantidad importante de depósitos y también se producirá mayor corrosión en el sistema.

• Contenido en metales (Na, K, P,..) y ácidos grasos libres. Contribuyen al aumento del residuo carbonoso de manera notable y también a las cenizas, generando residuos inorgánicos parcialmente quemados. Además, también se pueden formar jabones que colmatan los filtros del combustible.

• Lubricidad. Es la cualidad de un líquido para proporcionar una lubricación adecuada para prevenir el desgaste entre dos superficies en movimiento. Los combustibles con un contenido bajo en azufre o con baja viscosidad tienden a tener una lubricidad menor.

• Glicerina libre. Determina el nivel de glicerina no enlazada presente en el biodiesel. Su presencia normalmente se debe a una mala purificación del mismo. Niveles altos pueden causar problemas de depósitos en el inyector, así como la colmatación de filtros. Puede dañar los sistemas de inyección debido a los compuestos inorgánicos y jabones que se acumulan en la glicerina. Si la cantidad de ésta es superior al 0,5% puede afectar al contenido del residuo carbonoso.

• Glicerina total. Determina el nivel de glicerina enlazada y no enlazada presente en el combustible. Niveles bajos significan que se ha producido un alto grado de conversión en el aceite o grasa, y se han formado gran cantidad de monoésteres. Niveles alto de mono, di y triglicéridos pueden provocar la colmatación de los filtros, depósitos carbonosos en los inyectores y afectar adversamente a las propiedades a bajas temperaturas. Esto es debido a que al poseer temperaturas de ebullición superiores provocan que la combustión sea bastante peor. Además, aumentan la viscosidad del biodiesel.

• Contenido en alcohol. Puede provocar problemas de lubricidad y en el número de cetano. Desde el punto de vista de la seguridad, el punto de inflamación

disminuye. Por otro lado, junto a la presencia de alcohol puede venir asociada glicerina disuelta en este con los consiguientes problemas antes comentados.

• Estabilidad a la oxidación. Se determina la vida de almacenamiento y la degradación potencial de un combustible durante su almacenamiento. La oxidación de un combustible suele venir acompañada de la formación de gomas solubles e insolubles que pueden actuar de la siguiente manera:

- Gomas insolubles. Problemas de colmatación de filtros.
- Gomas solubles. Formación de depósitos en la punta del inyector y fallos en las boquillas de los inyectores.

Hay que tener en cuenta que el biodiesel se puede usar puro o mezclado con el gasoil. Esto hará que las propiedades que sean más diferentes entre ellos se igualen o diferencien según sea el porcentaje utilizado.

3.7. APLICACIONES DEL BIODIESEL.

El empleo más frecuente del biodiesel se encuentra en el sector del transporte. Pero existen otros sectores, como puede ser el de generación eléctrica, en los que también puede utilizarse.

Las aplicaciones más destacadas se listan a continuación:

1. Combustión en calderas. Se trata de una aplicación que no precisa de especificaciones especiales, en la que posiblemente pueda emplearse el aceite directamente sin esterificar. El problema reside en el coste relativo a otros combustibles, incluida la biomasa.

2. Aplicación a motores diesel estacionarios para generación de energía eléctrica o para motobombas en las propias zonas de cultivo. La ventaja consiste en que estos motores no necesitan combustibles tan sofisticados como los de automoción.

3. Aplicación a tractores agrícolas y otra maquinaria agrícola. Además de contar con la ventaja mencionada anteriormente, se reduce el coste del transporte si el biocarburante es producido en las cercanías del cultivo.

4. Aplicación a motores de barcos marinos o fluviales, con planteamientos similares.

5. Aplicación a vehículos diesel pesados (camiones y autobuses) y ligeros (pequeños camiones, microbuses y turismos). A esta aplicación corresponden generalmente las especificaciones del combustible más estrictas. Los vehículos antiguos, así como los diseñados para trabajo pesado, donde las exigencias relativas a prestaciones y emisiones no son extremas, pueden emplear combustibles con unas

especificaciones menos exigentes que las de los automóviles ligeros en Europa, EEUU o Japón. Así, un vehículo diesel de diseño antiguo, equipado con sistemas de control mecánico de inyección, no exige un carburante, y en particular un biodiesel, con especificaciones muy estrictas. Éstas corresponden al biodiesel de automoción para motores de combustión interna en vehículos ligeros.

4. ESTUDIO DE VIABILIDAD.

4.1. INTRODUCCIÓN.

El desarrollo del biodiesel puede significar nuevos nichos de mercado para el comercio de productos agrícolas y derivados a escala mundial. Actualmente son mercados protegidos, de pequeño volumen y subvencionados por las administraciones públicas, pero con su desarrollo muchos países agrícolas podrían aprovechar el sector para colocar su producción.

Según datos del IFP, (Institut Français du Pétrole, Francia) la producción mundial de biodiesel como biocarburante (concentrada principalmente en la Unión Europea) ascendió en 2003 a 1,6 millones de toneladas. Sin embargo, a este respecto es preciso señalar que Brasil inició a finales de 2004 un programa de fomento del biodiesel (mezcla al 2% y al 5% a medio plazo), generado a partir de aceites de soja y de ricino.

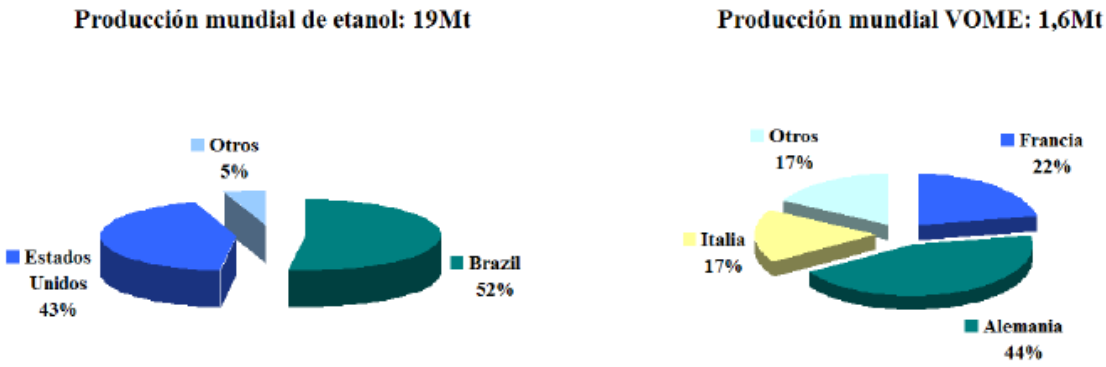


Gráfico 1.4: Distribución de la producción mundial de bioetanol y biodiesel en 2003. Fuente IFP. 2005

Esta distribución de la producción, entre países y entre tipos de biocarburantes, se debe a las condiciones geoclimatológicas y productivas de cada área. El consumo relativo de gasolina es mayor en América, a la vez que las grandes extensiones cerealistas de EE.UU. y el clima tropical de Brasil (caña de azúcar) favorecen el desarrollo de una industria del bioetanol.

Por el contrario, la infraestructura europea de "cracking" se construyó históricamente para optimizar la producción de gasolina en detrimento del gasóleo, y aunque actualmente se intenta realizar mejoras en la producción de éste combustible, es muy difícil optimizar, de manera significativa, la misma sin renovar completamente

las instalaciones. En los últimos años el incremento experimentado por el consumo de gasóleo ha creado un mercado que favorece su mezcla con el biodiesel elevándose por tanto la cantidad disponible de combustible).

4.2. DEPENDENCIA ENERGÉTICA.

España es un país con una dependencia muy fuerte de las importaciones energéticas. De hecho, importa casi un 80% de la energía que consume, por lo que nuestra economía es bastante vulnerable a cambios en los precios del petróleo y del gas. Geográficamente, esta dependencia se concentra en países no totalmente fiables ni en sus suministros ni en sus políticas, (Argelia, Nigeria, Rusia, ...).

Además, la economía española, en contra de la tendencia de la UE, tiene un tejido productivo con una intensidad energética alta y ascendente. Ambos factores añaden un valor especial a la energía producida con fuentes de energía renovables, autóctonas, e independientes de los vaivenes de precios en los mercados internacionales.

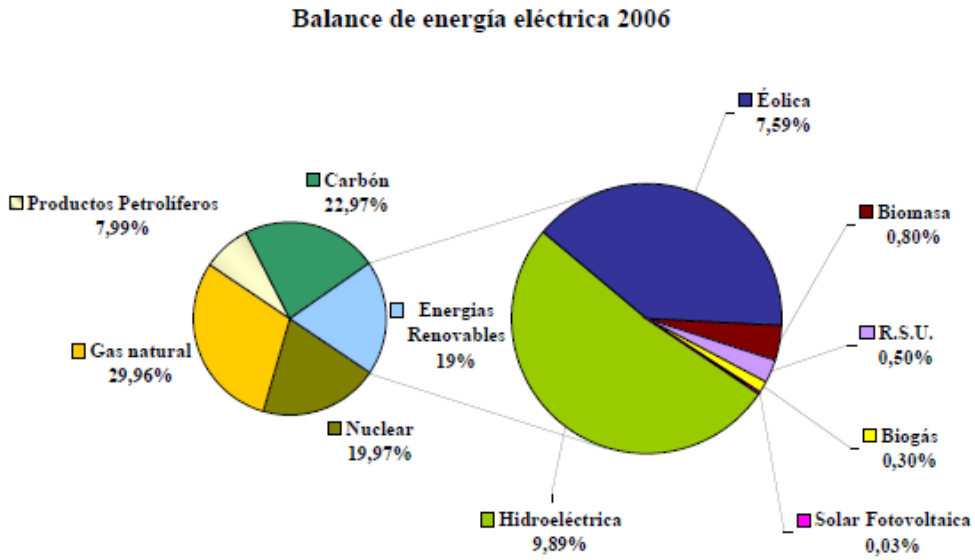


Gráfico 1.5: Balance español de producción de energía eléctrica correspondiente a 2006.

Por otro lado, aunque las energías renovables están cada vez más presentes en el cómputo energético total, la inmensa mayoría de la energía que se consume tiene origen fósil (petróleo, carbón y gas natural), con lo que es altamente contaminante. De hecho, España es el país europeo más alejado de cumplir con el Protocolo de Kyoto que establece unos límites cuantificados y obligatorios de emisión de gases de efecto invernadero (GEI) para los países jurídicamente vinculantes.

En el año 2005 en España, las emisiones de gases de efecto invernadero habían aumentado un 48,05% desde 1990 (**Gráfico 1.6**). Con el escenario actual, las emisiones

GEI alcanzarán el 52,5% en 2011. Este crecimiento triplica el 15% de aumento promedio permitido a España por el Protocolo para el periodo 2008-2012.

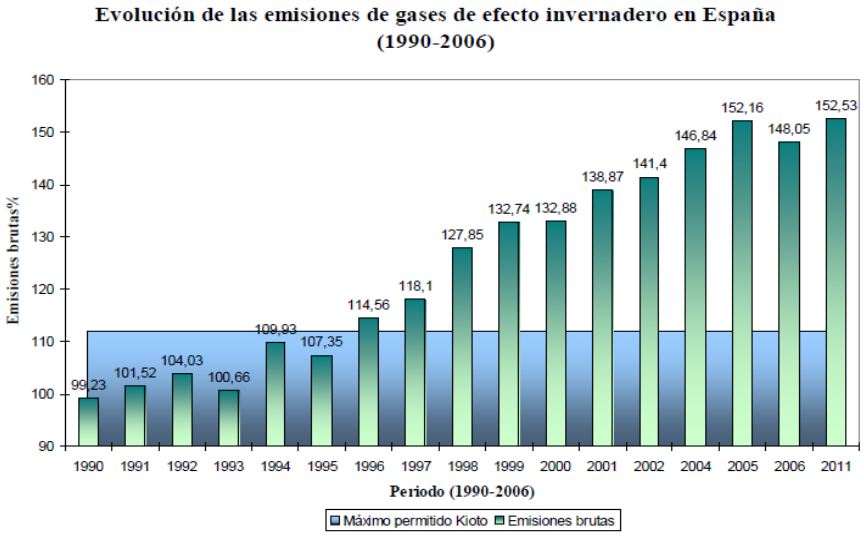


Gráfico 1.6: Evolución de las emisiones de gases de efecto invernadero en España (1990-2006)

5. LA INGENIERÍA DE PROCESOS DE SEPARACIÓN.

5.1. INTRODUCCIÓN.

Las separaciones son fundamentales en la ingeniería química. Una planta química típica es un reactor rodeado por separadores, tal como se muestra en el siguiente diagrama de flujo de la **figura 1.1.**

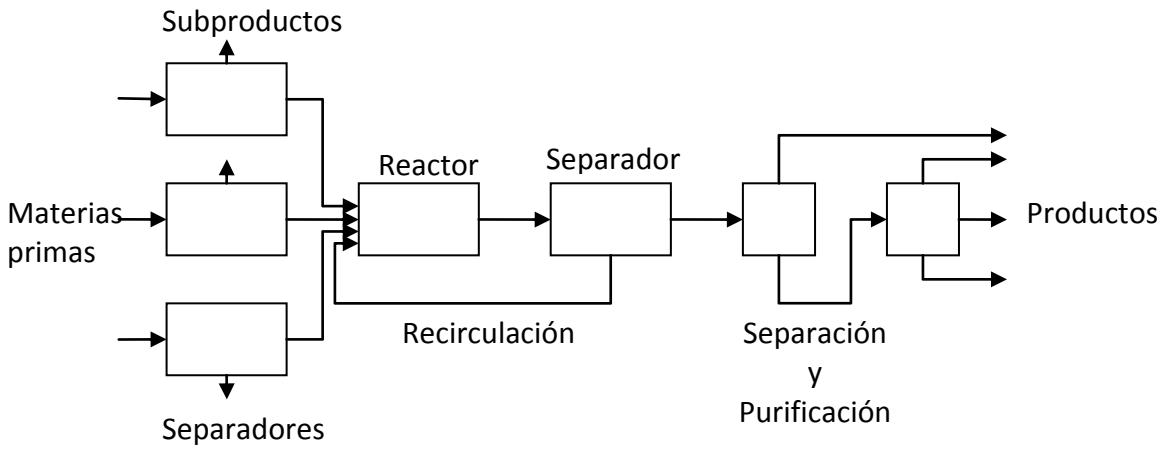


Figura 1.1: Diagrama de flujo típico para una planta química.

Las materias primas se purifican previamente en dispositivos de separación y se alimentan al reactor químico. La alimentación que no reacciona se separa de los productos de reacción y se recircula al reactor. Los productos se siguen separando y purificando antes de poder venderlos. Algunos ejemplos de procesos tradicionales

están ilustrados en Couper et al. (2005), Shreve y Austin (1984), Matar y Hatch (2001), Turton et al. (2003), Chenier (2002) y Speight (2002), mientras que procesos más recientes aparecen en la revista Chemical Engineering.

Los métodos de separación en ingeniería están a la orden del día, en todas ellas se ponen en contacto dos fases y pueden diseñarse y analizarse como procesos con etapas en equilibrio.

El concepto de etapa en equilibrio se aplica cuando el proceso puede construirse como una serie de etapas discretas, en las que se ponen en contacto dos fases para luego separarlas. Se supone que las dos fases que se separan están en equilibrio entre sí. Por ejemplo, en la destilación se ponen en contacto un vapor y un líquido sobre un plato metálico agujerado. A causa del contacto íntimo entre las dos fases, puede pasar soluto de una fase a otra. Arriba del plato, el vapor se separa del líquido. Tanto el líquido como el vapor se pueden mandar a más etapas, para aumentar la separación. Esto se vera mejor en el siguiente apartado.

Otro concepto útil es el de operación unitaria. En este caso la idea es que, aunque el diseño específico puede variar de acuerdo con las sustancias que se están separando, los principios básicos de la destilación siempre son los mismos. Por ejemplos, básicos de la destilación siempre son los mismos, sea que estemos separando etanol de agua o bien varios hidrocarburos, o biodiesel.

Se usará el método de etapa por etapa, donde los cálculos se terminan para una etapa, y los resultados se usan para calcular la etapa siguiente, desarrollando así una comprensión básica.

5.2. EL CONCEPTO DE EQUILIBRIO.

El concepto de equilibrio como se ha comentado en el apartado anterior, se establece que las corrientes que salen de una etapa están en equilibrio. Para explicar mejor este equilibrio en la etapa, imaginemos un vapor y un líquido que están en contacto entre sí, como muestra la **figura 1.2**. Las moléculas de líquido se están evaporando continuamente, mientras que las moléculas de vapor se están condensando continuamente. Si están presentes dos especies químicas, en general se condensarán y evaporarán con velocidades diferentes. Cuando no están en equilibrio, el líquido y el vapor pueden estar a distintas presiones y temperaturas, y estar presentes en diferentes fracciones molares. En el equilibrio, las temperaturas, presiones y fracciones de las dos fases cesan de cambiar. Aunque las moléculas continúan evaporándose y condensándose, la velocidad con la que se condesa cada especie es igual a la velocidad con la que se evapora. Aunque en escala molecular nada se ha detenido, en escala macroscópica, donde solemos observar los procesos, ya no hay más cambios en la temperatura, presión o composición.

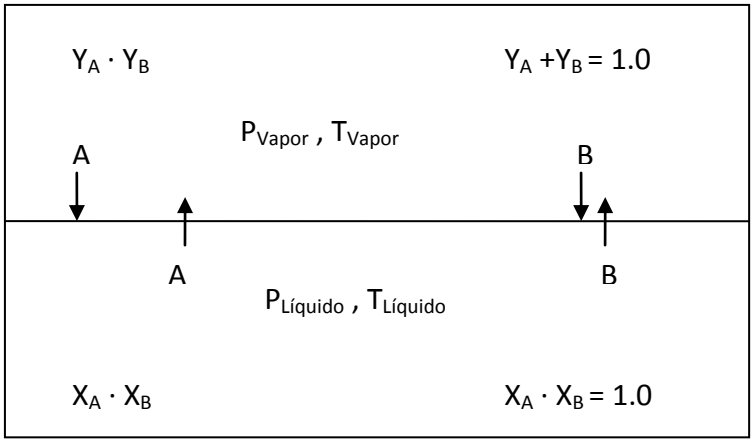


Figura 1.2: Sistema de contacto entre vapor y líquido

Las condiciones de equilibrio se pueden subdividir en forma conveniente en equilibrio térmico, mecánico y de potencial químico. En equilibrio térmico, las transferencias de calor cesa y las temperaturas de las dos fases con iguales. Por lo tanto:

$$T_{líquido} = T_{Vapor} \quad \text{(en el equilibrio)} \qquad (1\text{-}1)$$

En el equilibrio mecánico, las fuerza entre vapor y líquido se balancean. En los procesos de separación suele implicar que las presiones son iguales. Así:

$$P_{líquido} = P_{Vapor} \quad \text{(en el equilibrio)} \qquad (1\text{-}2)$$

Si la interfaz entre líquido y vapor es curva, la igualdad de fuerzas no implica igualdad de presiones. Es este caso se puede deducir la ecuación de Laplace (Levich, 1962).

En el equilibrio entre fases, la velocidad con la que se evapora cada especie es exactamente igual a la evaporación con la que se condensa. Así, no hay cambio de composición (fracción mol en la **figura 1.2**). Sin embargo, en general las composiciones del líquido y el vapor no son iguales. Si las composiciones fueran iguales no podría alcanzarse separación alguna en proceso de equilibrio alguno. Si la temperatura y la presión son constantes, la igualdad de velocidades de evaporación y condensación requieren que en el sistema haya un mínimo de energía libre. La condición resultante de para los equilibrios entre fases es:

$$(potencial\acute{}químico)_{líquido} = (Potencial\ `químico)_{Vapor} \qquad (1\text{-}3)$$

El desarrollo de esta ecuación (1-3), no es motivo de este trabajo, pero si se quiere profundizar más en ella hay muchos libros de termodinámica que definen los conceptos necesarios para su compresión, por ejemplo, Smith et al. 2005; Balzhizer et

al., 1972; Denbigh, 1981; Elliot y Lira, 1999). Sin embargo esta ecuación requiere cierta relación entre las composiciones del líquido y el vapor.

Como se ha comentado en el apartado (introducción) los procesos de separación son muy importantes en la ingeniería, pudiéndose encontrar diferentes tipos de destilaciones. Se ha descrito el concepto de equilibrio y se podría profundizar más en diferentes tipos de destilación, ya que existe gran cantidad de ellas. Pero solo se hablará en este trabajo de la destilación en columna que es la que se utilizará para separar los componentes de Ester de metilo con Metanol.

5.3. INTRODUCCIÓN A LA DESTILACIÓN EN COLUMNA.

La destilación es, por mucho, la técnica más común en la industria de procesos químicos, con ella se realizan de 90 a 95% de las separaciones en la industria. En Estados Unidos, por ejemplo, hay aproximadamente 400,000 columnas de destilación activas que consumen un 40% de la energía que se usan las industrias de procesos químicos en ese país, es decir, el equivalente a la asombrosa cantidad de 1,2 millones de barriles de petróleo crudo al día (Humphrey y Keller, 1997).

La destilación de columna es un procedimiento de conexión en cascada. El procedimiento de conexión en cascada se puede ampliar a un proceso que produzca un vapor puro y un líquido puro. Las columnas de destilación están formadas por una serie de platos (en cascada) donde tiene lugar cada uno de los equilibrios. En cada plato de la columna se mezcla la fase vapor con la fase líquida llegando al equilibrio ambas fases. Las columnas de destilación poseen una entrada de alimentación, donde entra la mezcla binaria a separar, como se puede ver en la **figura 1.3.** La mezcla binaria entra con una temperatura y una presión a la columna de destilación, y uno de los dos productos a separar es más volátil que el otro. Al ser uno de los dos componentes más volátil, lo que ocurre cuando la mezcla entra en la columna, en el plato de entrada (plato de alimentación) se produce el equilibrio formándose una fase vapor y una líquida. La fase vapor esta enriquecida con el componente más volátil mientras que la fase líquida esta empobrecida con el componente más volátil.

Como se puede ver en la **figura 1.3**, F es la alimentación, mientras que L_n y V_n son la fase líquida y vapor de cada una de las etapas o platos. Las etapas por encima de la alimentación, a medida que van subiendo cada etapa o plato va siendo más rico en la sustancia más volátil. Esto es debido porque en cada plato una vez alcanzado el equilibrio, como se dijo anteriormente, se crea un líquido y un vapor, el vapor sube a la siguiente etapa o plato mientras que el líquido pasa a una etapa inferior. Esto quiere decir, que en siguiente plato no solo habrá el liquido y el vapor de esa etapa sino que también se unirá la fase se vapor o líquida del anterior. En el caso de que sea el vapor el que se une esto hace que la concentración del líquido en esa etapa sea mas rica de la sustancia más volátil.

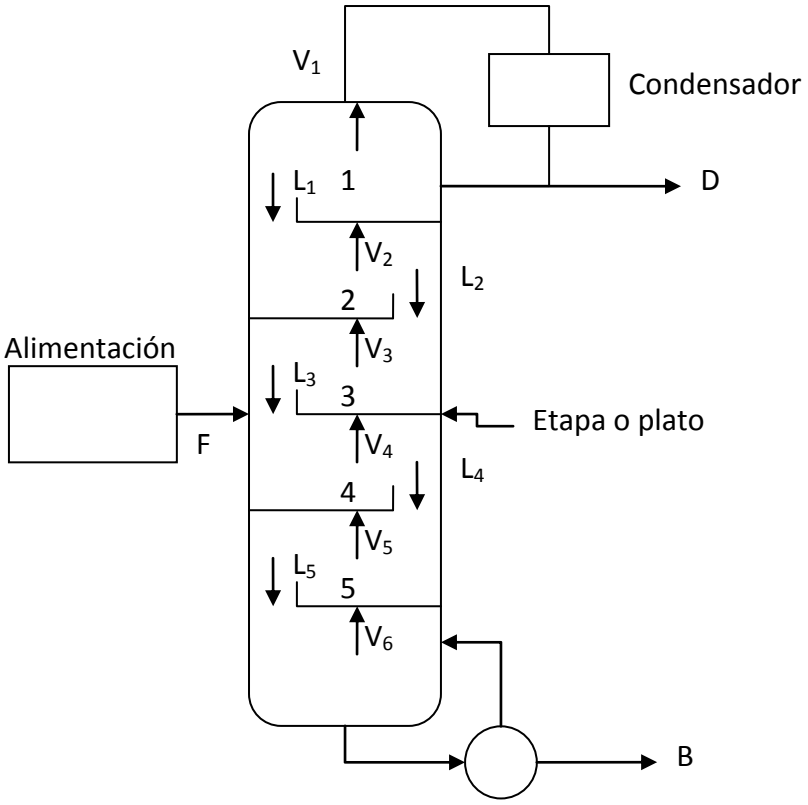


Figura 1.3: Columna de destilación: Esquema de una columna con cinco platos ($T_1<T_2<T_3<T_4<T_5$).

Por lo tanto V_1 es el vapor más rico en el componente de la mezcla más volátil, mientras que L_5 es el líquido más rico en el componente menos volátil. El vapor que sale por el plato 1 se condesa, obteniendo así el destilado representado por la letra D. Mientras que en el fondo de la columna se obtiene el residuo de la propia destilación representada por B.

La columna estará representada por tantos platos como sean necesarios para la destilación. A continuación se procederá a los cálculos internos de la columna que es de lo que trata este trabajo.

5.4. BALANCES INTERNOS DE LA COLUMNA DE DESTILACIÓN.

Los balances internos se van a definir para sistemas binarios. El número de etapas necesario para la separación se puede obtener fácilmente usando balances de etapa por etapa.

Se comienza en la parte superior de la columna, escribiendo los balances y la relación de equilibrio para la primera etapa; una vez determinadas las variables desconocidas para la primera, se escriben los balances para la segunda etapa. Usando las variables que se acaban de calcular, se pueden calcular otra vez las incógnitas. De

esta manera se procede hacia debajo de la columna, etapa por etapa, hasta llegar al fondo. Podríamos también comenzar por el fondo y avanzar hacia arriba. Este procedimiento supone que cada etapa está en equilibrio, pero puede suceder que esta hipótesis no sea válida.

En la sección de enriquecimiento de la columna conviene usar una envolvente de balance que rodee la etapa que se desea y al condensador. Esto se muestra en la **figura 1.4**. Para la primera etapa la envolvente de balance se muestra en la **figura 1.4A**. Entonces, el balance general de masa es:

$$V_2 = L_1 + D \qquad \text{(1-4, etapa 1)}$$

El balance de masa del componente más volátil es:

$$V_2 y_2 = L_1 x_1 + D x_D \qquad \text{(1-5, etapa 1)}$$

Para una columna adiabática, bien aislada, el balance de energía es:

$$V_2 H_2 + Q_c = L_1 h_1 + D h_D \qquad \text{(1-6, etapa 1)}$$

Suponiendo que cada etapa está en equilibrio, sabemos que el líquido y el vapor que salen de cada etapa están en equilibrio también. Para un sistema binario la regla de las fases de Gibbs es:

$$Grados \quad de \quad Libertad = C - P + 2 = 2 - 2 + 2 = 2$$

Como se ha fijado la presión, queda sólo un grado de libertad disponible. Así, para cada etapa en equilibrio, todas las variables son funciones de una sola variable. Para el líquido saturado se puede escribir:

$$h_1 = h_1(x_1) \qquad \text{(1-7a, etapa 1)}$$

Y para el vapor saturado,

$$H_2 = H_2(y_2) \qquad \text{(1-7b, etapa 1)}$$

También están relacionadas las fracciones molares en líquido y vapor que salen de una etapa:

$$x_1 = x_1(y_1) \qquad \text{(1-7c, etapa 1)}$$

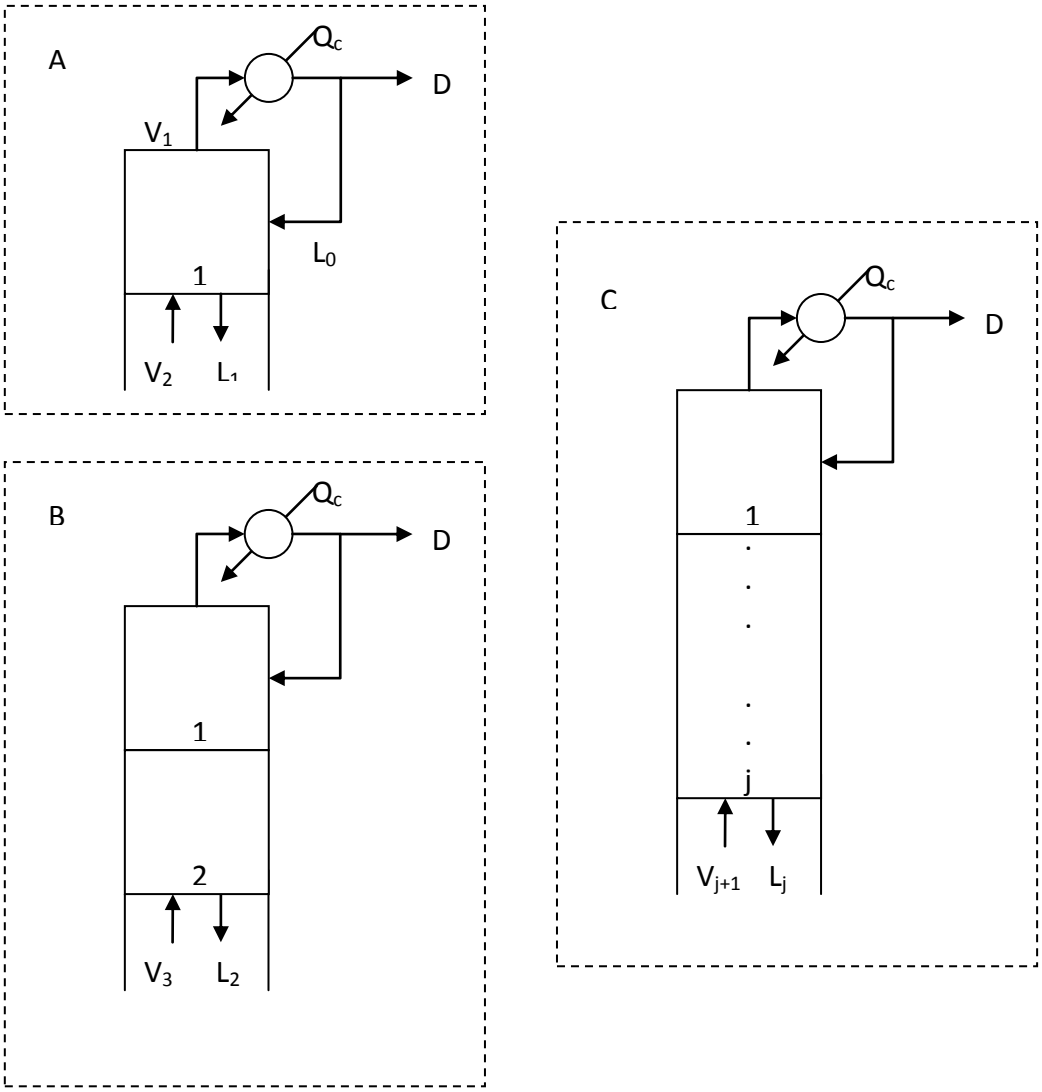


Figura 1.4: Envolvente de balance en la sección de enriquecimiento: A-Etapa 1, B-Etapa 2; C-Etapa J.

Las ecuaciones (1-7) para la etapa uno representan la relación de equilibrio. La forma exacta de éstas depende del sistema químico que se esté separando. Estas ecuaciones son seis, con seis incógnitas: L_1, V_2, x_1, y_2, H_2 y h_1.

Como se tienen seis ecuaciones con seis incógnitas es posible despejar las seis incógnitas., para entonces avanzar a la segunda etapa. Para la segunda etapa usaremos la envolvente de balance que muestra la **figura 1.4B.** Ahora, los balances de masa son:

$$V_3 = L_2 + D \qquad \text{(1-4, etapa 2)}$$

$$V_3 y_3 = L_2 x_2 + D x_D \qquad \text{(1-5, etapa 2)}$$

Mientras que el balance de energía es:

$$V_3H_3 + Q_c = L_2h_2 + Dh_D \qquad \text{(1-6, etapa 2)}$$

Las relaciones de equilibrio son:

$$h_2 = h_2(x_2) \qquad H_3 = H_3(y_3) \qquad x_2 = x_2(y_2) \qquad \text{(1-7, etapa 2)}$$

Otra vez tenemos seis ecuaciones con seis incógnitas. Las incógnitas son ahora L_2, V_3, x_2, y_3, H_3 y h_2. Se puede avanzar a la tercera etapa y usar los mismos procedimientos. Después pasar cuarta etapa, luego quinta y así sucesivamente. Para una etapa cualquiera j (j puede ir de 1 hasta f-1, siendo f la etapa de la alimentación) en la sección de enriquecimiento, la envolvente de balance se muestra en la **figura 1-4C**. Para esta etapa, los balances de masa y energía son:

$$V_{j+1} = L_j + D \qquad \text{(1-4, etapa j)}$$

$$V_{j+1}y_{j+1} = L_jx_j + Dx_D \qquad \text{(1-5, etapa j)}$$

y

$$V_{j+1}H_{j+1} + Q_c = L_jh_j + Dh_D \qquad \text{(1-6, etapa j)}$$

Mientras que las relaciones de equilibrio son:

$$h_j = h_j(x_j) \qquad H_{j+1} = H_{j+1}(y_{j+1}) \qquad x_j = x_j(y_j) \qquad \text{(1-7, etapa j)}$$

Cuando se llega a la etapa j, se conocen los valores de y_j, Q_c, D y h_D, y las variables desconocidas serán L_j, V_{j+1}, x_j, y_{j+1}, H_{j+1} y h_j. En la etapa de alimentación cambiarán los balances de masa y energía, por la entrada de corriente de alimentación.

Antes de continuar nos detendremos para observar la simetría de los balances de masa y energía y las relaciones de equilibrio al pasar de una etapa a otra. Las ecuaciones (1-4) para las etapa 1,2 y j tienen la misma estructura; difieren sólo en los subíndices. Las ecuaciones (1-4, etapa1) o (1-4, etapa 2) se pueden obtener a partir de la ecuación general (1-4, etapa j) sustituyendo j por 1 o 2, respectivamente. Las mismas observaciones se hacen para las demás ecuaciones (1-5, 1-6 y 1-7). Las variables desconocidas, al pasar de una a otra etapa también se parecen, sólo difieren en el subíndice.

Además de esta simetría en las ecuaciones de etapa a etapa, hay simetría ente las ecuaciones para una misma etapa. Así, todas las ecuaciones (1-4, etapa j), (1-5, etapa j) y (1-6, etapa j) son balances en estado estable, que indican:

Entrada = Salida

En las tres ecuaciones, la salida (general de masa, soluto o energía) está asociada con las corrientes L_j y D. La entrada está asociada con la corriente V_{j+1} y (para la energía) con la carga de enfriamiento, Q_c.

Debajo de la etapa de alimentación, las ecuaciones de balance deben cambiar, pero las relaciones de equilibrio en las ecuaciones (1-7a, b, c) quedarán intactas. Las envolventes de balance en la sección de empobrecimiento se muestran en la **figura 1.5** para una columna con un vaporizador parcial. Las rayas sobre las tasas de flujo indican que están en la sección de arrastre o empobrecimiento. Lo más sencillo es escribir los balances de la sección de empobrecimiento en torno al fondo de la columna usando la envolvente de balance de la **figura 1.5**. Así, los balances en torno a la etapa f+1 (inmediatamente debajo de la etapa de alimentación) son:

$$\overline{V}_{f+1} = \overline{L}_f - B \qquad \text{(1-4, etapa f+1)}$$

$$\overline{V}_{f+1} y_{f+1} = \overline{L}_f x_f + B x_B \qquad \text{(1-5, etapa f+1)}$$

$$\overline{V}_{f+1} H_{f+1} = \overline{L}_f h_f - B h_B + Q_R \qquad \text{(1-6, etapa f+1)}$$

Las relaciones de equilibrio son las ecuaciones (1-7) formuladas para la etapa f+1:

$$h_f = h_f(x_f) \qquad H_{f+1} = H_{f+1}(y_{f+1}) \qquad x_f = x_f(y_f) \qquad \text{(1-7, etapa f+1)}$$

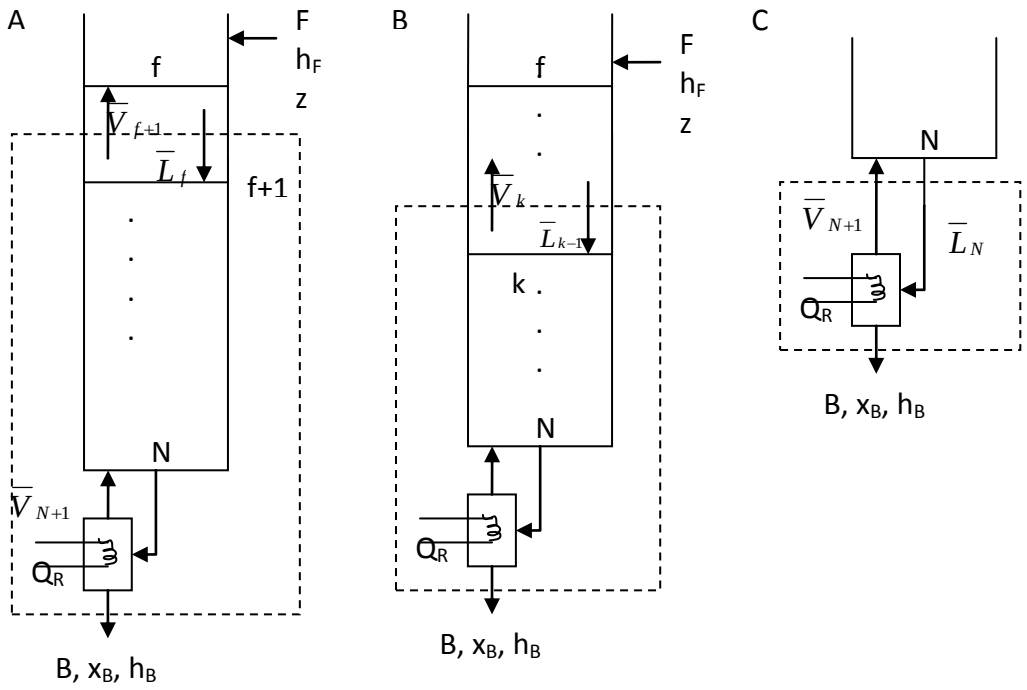


Figura 1.5: Envolventes de balance en la sección de agotamiento: A-debajo de la etapa de alimentación (etapa f+1), B- etapa k, C- vaporizador parcial.

Estas seis ecuaciones tienen seis incógnitas: L_f, V_{f+1}, x_f, y_{f+1}, H_{f+1} y h_f. En la formulación del problema se especifica x_B; B y Q_R se calcularon a partir de los balances en la columna, y y_f (necesarios para la última ecuación) se obtuvo resolviendo las ecuaciones (1-4, etapa j) a (1-7c, etapa j) con j=f-1. En la etapa de alimentación cambiamos de un conjunto a otro de envolvente de balance.

Hay que tener en cuenta que las mismas ecuaciones se obtiene si escribimos los balances arriba de la etapa f+1 y en torno a la parte superior de la columna de destilación. Eso se ve con facilidad en el balance general de masa que ahora es:

$$\overline{V}_{f+1} + F = D + \overline{L}_f$$

Este se ordena como sigue:

$$\overline{V}_{f+1} = \overline{L}_f - (F - D)$$

Sin embargo, como el balance externo de la columna indica que F-D=B, la última ecuación se transforma en:

$$\overline{V}_{f+1} = \overline{L}_f - B$$

que es la ecuación (1-4, etapa f+1). Para las demás ecuaciones de balance, se obtienen resultados parecidos.

Una vez resueltas las seis ecuaciones para la etapa f + 1, se puede bajar por la columna a la siguiente etapa f + 2. Con una envolvente de balance en torno a una etapa general k, como se ve en la **figura 1.5 B**, las ecuaciones son:

$$\overline{V}_k = \overline{L}_{k-1} - B \qquad \text{(1-4, etapa k)}$$

$$\overline{V}_k y_k = \overline{L}_{k-1} x_{k-1} + B x_B \qquad \text{(1-5, etapa k)}$$

$$\overline{V}_k H_k = \overline{L}_{k-1} h_{k-1} - B h_B + Q_R \qquad \text{(1-6, etapa k)}$$

Las relaciones de equilibrio son las ecuaciones (1-7) formuladas para la etapa k-1:

$$h_{k-1} = h_{k-1}(x_{k-1}) \qquad H_k = H_k(y_k) \qquad x_{k-1} = x_{k-1}(y_{k-1}) \qquad \text{(1-7, etapa k)}$$

Un vaporizador parcial, como el de la **figura 1.5 C**, funciona como un contacto en equilibrio. Si se considera que vaporizador es la etapa N + 1, se obtienen los balances para la envolvente de la **figura 1.5 C**, haciendo que k = N + 1, y k – 1= N en las ecuaciones (1-7).

Si $x_{B+1} = X_B$, los N + 1 contactos de equilibrio nos llevan exactamente a la separación requerida y el problema termina. Si $x_{N+1}<x_B$, mientras que $x_N> X_B$, los N+1 contactos de equilibrio producen un poco más de separación de la requerida.

Así como las ecuaciones de balance en la sección de enriquecimiento son simétricas de una etapa a la otra, también lo son en la sección de empobrecimiento.

Una vez visto como funciona una columna de destilación y sus balances internos, ya se puede comprender mejor su utilización. Para ello en este trabajo se desarrollara un programa que nos proporcione los platos de la columna y el mejor plato de alimentación para las sustancias a separar. Para ello se aplicará en el programa el método de Mcabe-Thiele, que se explicara a continuación para su mejor compresión.

6. MÉTODO DE MCABE-THIELE.

McCabe y Thiele (1925) inventaron un método de solución gráfica basado en el método de Lewis y la observación de que en las ecuaciones de operación se grafican líneas rectas en un diagrama y-x. En esta gráfica, se puede resolver la relación de equilibrio a partir de la curva de equilibrio y-x y los balances de masa a partir de las líneas de operación.

Suponemos que disponemos de datos de equilibrio a la presión de operación de la columna. Estos datos se grafican en la **figura 1.6**. Arriba de la columna hay un condensador total, esto quiere decir que $y_1 = x_D = x_0$. El vapor que sale de la primera etapa está en equilibrio con el líquido que sale de la primera etapa. La composición de este líquido, x_1, se puede determinar en la curva de equilibrio donde $y = y_1$. Esto se ve en la **figura 1.6**.

Por la corriente de líquido L_1, de composición x_1, pasa la corriente de vapor V_2, de composición y_2, dentro de la columna. Cuando se escriben los balances de masa alrededor de la etapa 1 y la parte superior de la columna (ver apartado anterior), el resultado, suponiendo derrame molal constante y después de algunas maniobras algebraicas, se obtiene la ecuación (1-8). Al graficar esta ecuación resulta una línea recta en el diagrama y-x.

$$y = \frac{L}{V}x + \left(1 - \frac{L}{V}\right)x_D \qquad (1\text{-}8)$$

Esta ecuación se aplica a las corrientes pasantes. La gráfica de la ecuación (1-8) es una línea recta con pendiente $\frac{L}{V}$ y ordenada al origen (x=0) igual a $\left(1-\frac{L}{V}\right)x_D$. Una vez trazada la gráfica de la ecuación, se determina y_2 con facilidad a partir del valor y cuan $x = x_1$. Esto se ve en la **figura 1.7**. La línea de operación superior pasa por el punto (y_1, x_D), porque esas coordenadas satisfacen la ecuación (1-8).

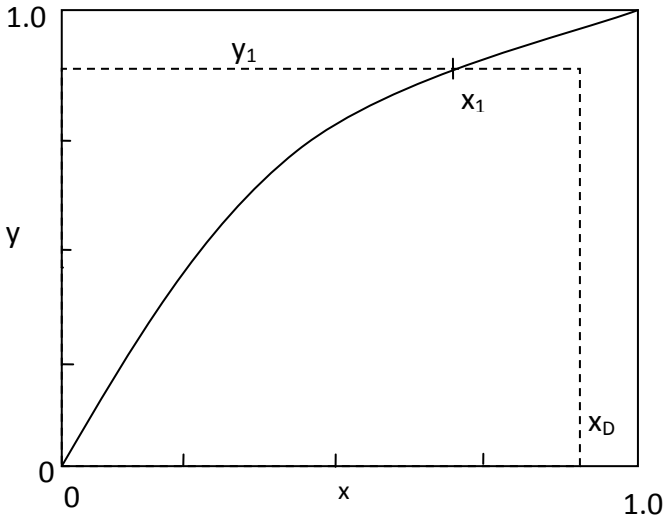


Figura 1.6: Equilibrio para la etapa superior en el diagrama de McCabe-Thiele.

Conocida y2, se puede proseguir columna abajo. Como x2 y y2 están en equilibrio, se obtiene x2 con facilidad en la curva de equilibrio. Entonces se obtiene y3 con la línea de operación, porque x2 y y3 tienen las composiciones de las corrientes pasantes. Este procedimiento de escalonamiento de etapas se ve en la figura 1.8. Se puede continuar mientras estemos en la sección de rectificación. Observe que se produce una escalera en el diagrama y-x, o diagrama de McCabe-Thiele. En lugar de memorizar este procedimiento, usted debe seguir los puntos en el diagrama y compararlos con los esquemas de una columna de destilación. Observe que las líneas horizontales y verticales no tienen significado físico. Los puntos en la curva de equilibrio (cuadrados) representan las corrientes de líquido y vapor que salen de una etapa de equilibrio. Los puntos en la curva de operación (círculos) representan las corrientes de líquido vapor pasantes entre sí en la columna.

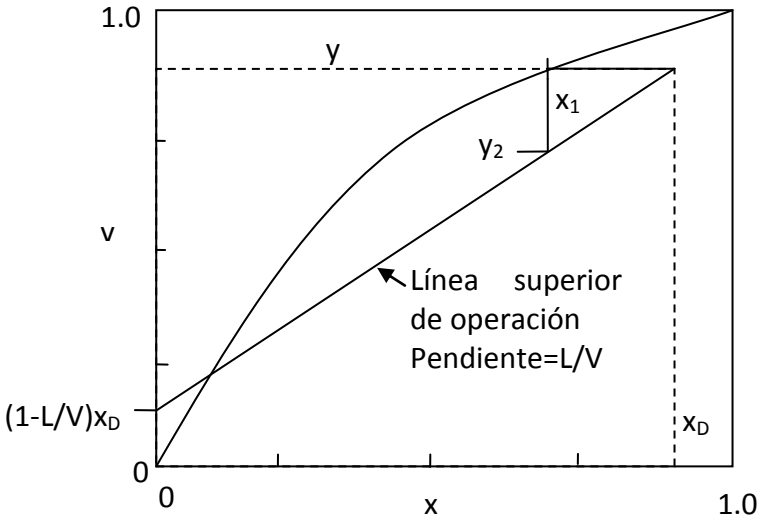


Figura 1.7: Cálculo de la etapa 1 en el diagrama de Mccabe-Thiele.

En la sección de agotamiento y no es válida la línea de operación superior porque se requieren diferentes balances de masa. En consecuencia, una ecuación de operación diferente. La ecuación de operación para la sección de agotamiento es:

$$y = \frac{\bar{L}}{\bar{V}}x - \left(\frac{\bar{L}}{\bar{V}} - 1\right)x_B \qquad (1\text{-}9)$$

La gráfica de la ecuación (1-9) es una recta con pendiente $\frac{\bar{L}}{\bar{V}}$ y ordenada al origen $-\left(\frac{\bar{L}}{\bar{V}} - 1\right)x_B$, como se ve en la figura 1.9. Esta línea de operación inferior se aplica a corrientes en contacto mutuo en la sección de agotamiento. Comenzando con el líquido que sale del vaporizador parcial, cuya fracción mol $x_B = X_{N+1}$, sabemos que el vapor que sale del vaporizador parcial está en equilibrio con x_B. Entonces se puede determinar y_{N+1} con la curva de equilibrio, y x_N con la línea de operación inferior, ya que el líquido de composición x_N es una corriente pasante en vapor de composición y_{N+1}. Mientras se esté en la sección de agotamiento, se puede continuar alternando entre la curva de equilibrio y la línea de operación inferior.

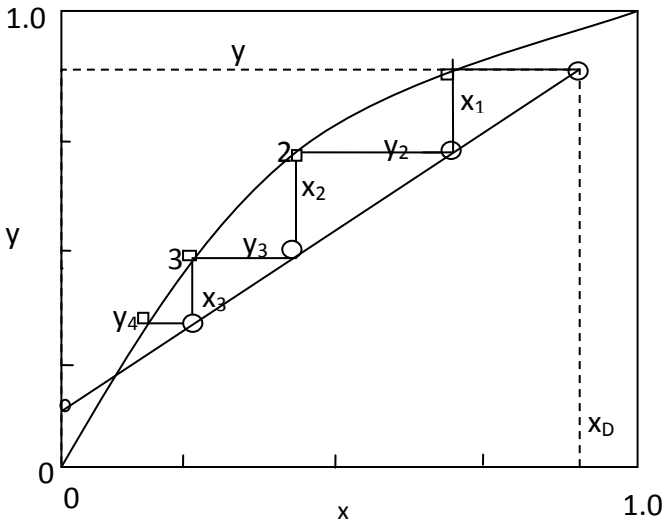


Figura 1.8: Escalonamiento de las etapas en la sección de enriquecimiento.

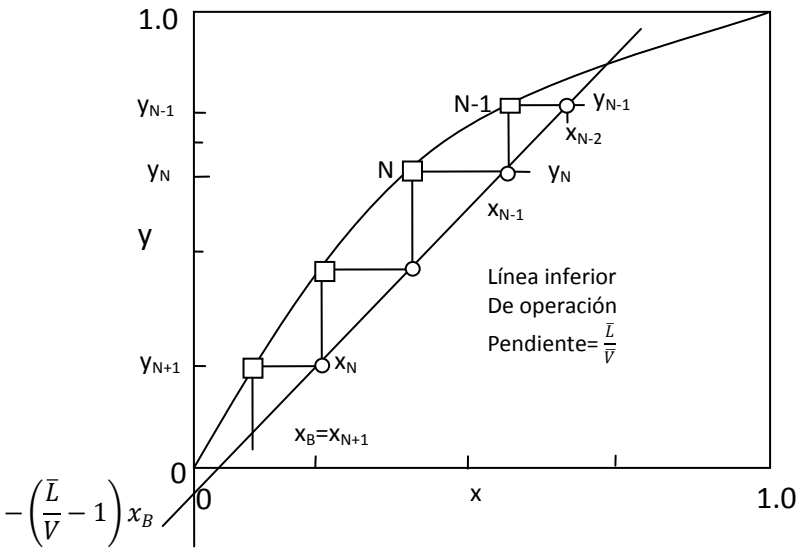


Figura 1.9: Escalonamiento de etapas en la sección de agotamiento.

Si escalonamos etapas bajando la columna, en la etapa de alimentación f cambiamos de la línea de operación superior a la línea de operación inferior. Arriba de la etapa de alimentación, x_{f-1} se calcula con la curva de equilibrio, y y_f con la línea de operación superior. Ya que se supone que el líquido y el vapor que salen de la etapa de alimentación están en equilibrio, x_f se puede determinar con la curva de equilibrio en y = y_f, para entonces determinar yf+1 en la línea de operación inferior. El procedimiento se ilustra en la figura 1.10 A, donde la etapa 2 es la de alimentación. Cuando se unas la etapa 2 como la etapa de alimentación la separación que muestra la figura 1.10 A requiere 4 etapas de equilibrio y un vaporizador parcial en equilibrio, es decir, 5 contactos de equilibrio. En este caso, la etapa 2 es la etapa óptima de alimentación. Es decir, una separación necesitará una cantidad mínima de etapas cuando se usa la etapa 2 como etapa de alimentación. Si, por ejemplo, se usara la etapa 2 o la etapa 3, se necesitarían más etapas totales (figuras 1.10 B y C). Para la destilación binaria, es fácil determinar el plato óptimo de alimentación, ya que siempre será el plato donde la etapa en la escalera contenga el punto de intersección de las dos líneas de operación, eso se ve comparando las figuras 1.10 A, B y C.

Cuando se escalonan las etapas de arriba abajo, se puede calcular el número fraccionario de etapas como sigue:

$$Fracción = \frac{distancia\ \ de\ la\ línea\ \ de\ operaci\ ón\ a\ x_B}{distancia\ \ de\ la\ línea\ \ de\ operaci\ ón\ a\ la\ curva\ \ de\ equilibrio} \qquad (1\text{-}10)$$

que se obtiene midiendo las distancias horizontales en el diagrama.

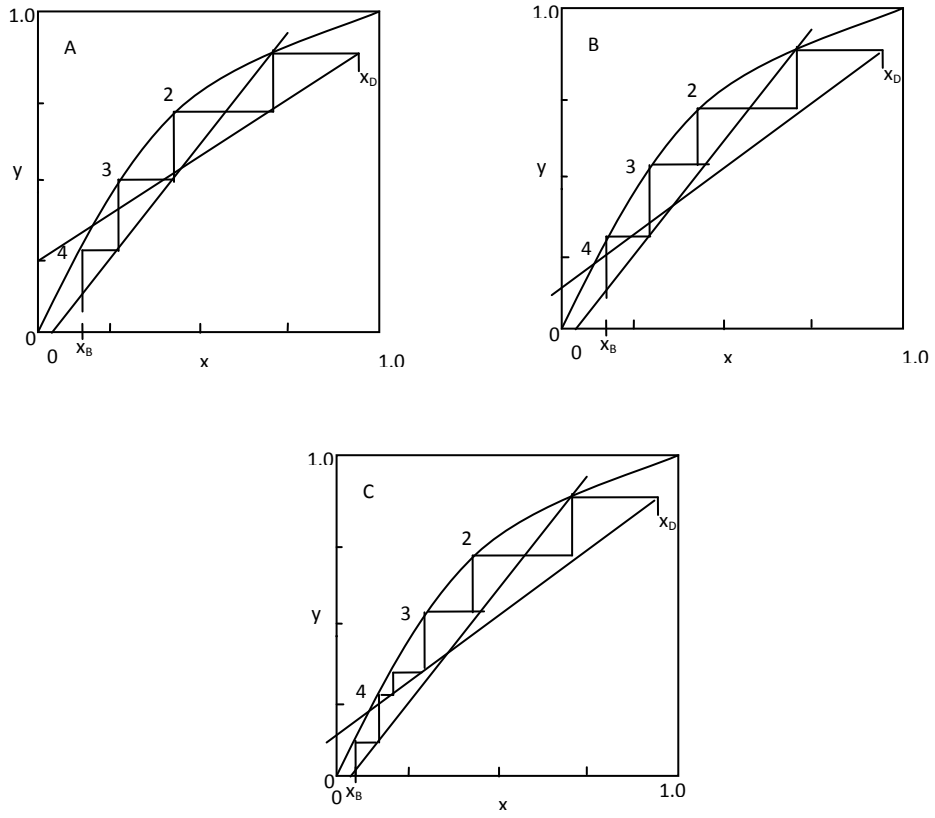


Figura 1.10: Diagrama de McCabe-Thiele para toda la columna; A etapa óptima de alimentación (etapa 2), B etapa de alimentación muy alta, C etapa de alimentación muy baja.

Este es el método de McCabe-Thiele, y es el que se ha implementado en Matlab mediante una interface, en este trabajo.

7. PROCEDIMIENTO EXPERIMENTAL

7.1. INTRODUCCIÓN

Como se ha comentado como fundamento principal de este trabajo de fin de máster es llevar a cabo parte de los conocimientos adquiridos en el máster cursado. Por ello se ha procedido a implementar un programa en Matlab, el cual nos facilita saber datos importantes de las sustancias con las que trabaja el grupo de investigación para el cual estoy colaborando.

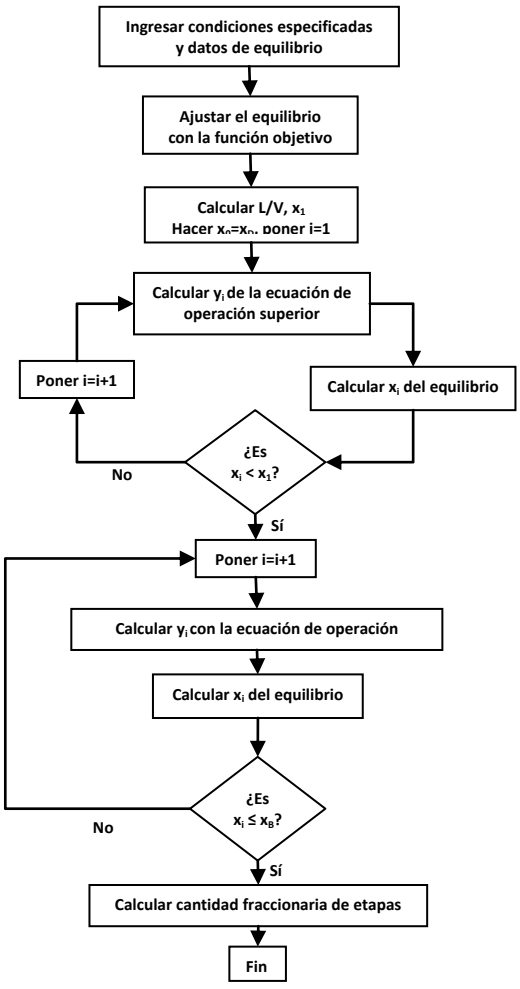
El programa implementado en Matlab, es el método anteriormente descrito, el método de McCabe-Thiele, que como se ilustró es uno de los métodos de separación para mezclas binarias más utilizados dentro de un gran grupo de métodos de separación. Debido a que las sustancias con las que vamos a trabajar, en el caso del biodiesel (esteres de metilo + metanol) son mezclas binaria, este método además de sencillo es el ideal para la obtención de los resultados que estamos buscando.

Estos datos de mezclas binarias son obtenidos a partir de datos experimentales que se han realizado en el laboratorio. Estos datos se obtienen por el proceso de equilibrio de ambas sustancias, el cual se ha explicado con anterioridad en este trabajo.

Una vez realizado el programa de Matlab, se ha procedido a realizar comparaciones con dos programas muy utilizados a nivel comercial en los procesos simulación de separaciones y equilibrio de mezclas. Estos son el Chemcad y el Aspen Plus.

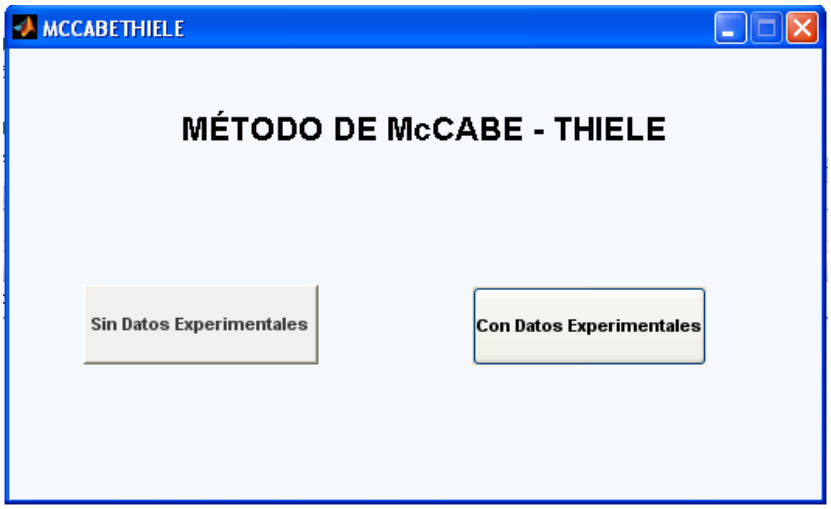
7.2. PROGRAMACIÓN EN MATLAB.

El programa se ha realizado por medio de una interface gráfica de usuario que posee el Matlab llamada Gui-s. Las GUI-s son herramientas muy útiles para entregar aplicaciones de trabajo, y a su vez facilitar al usuario el manejo con solo guiarse por una serie de botones sin saber el código interno, para los cálculos pertinentes que tiene el programa. Por ello se ha implementado el método de McCabe-Thiele en una Gui-s, para que el usuario pueda introducir sus datos experimentales sin saber los cálculos internos de este y obtener unos datos óptimos con un costo de tiempo razonable. El programa sigue el siguiente diagrama de flujo:

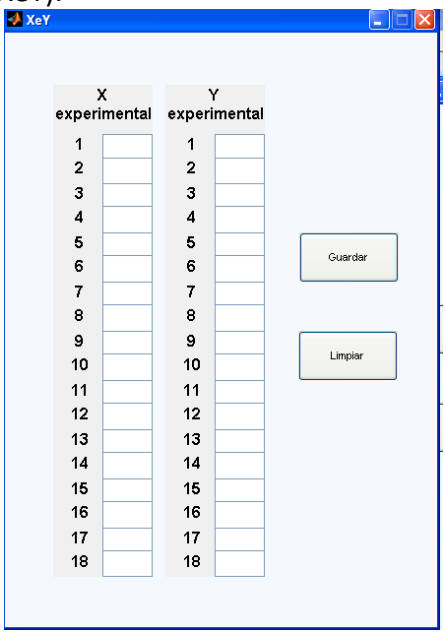


Observando cómo se comporta el programa a través del diagrama de flujo, pasamos a ver las ventanas con las que nos vamos a encontrar.

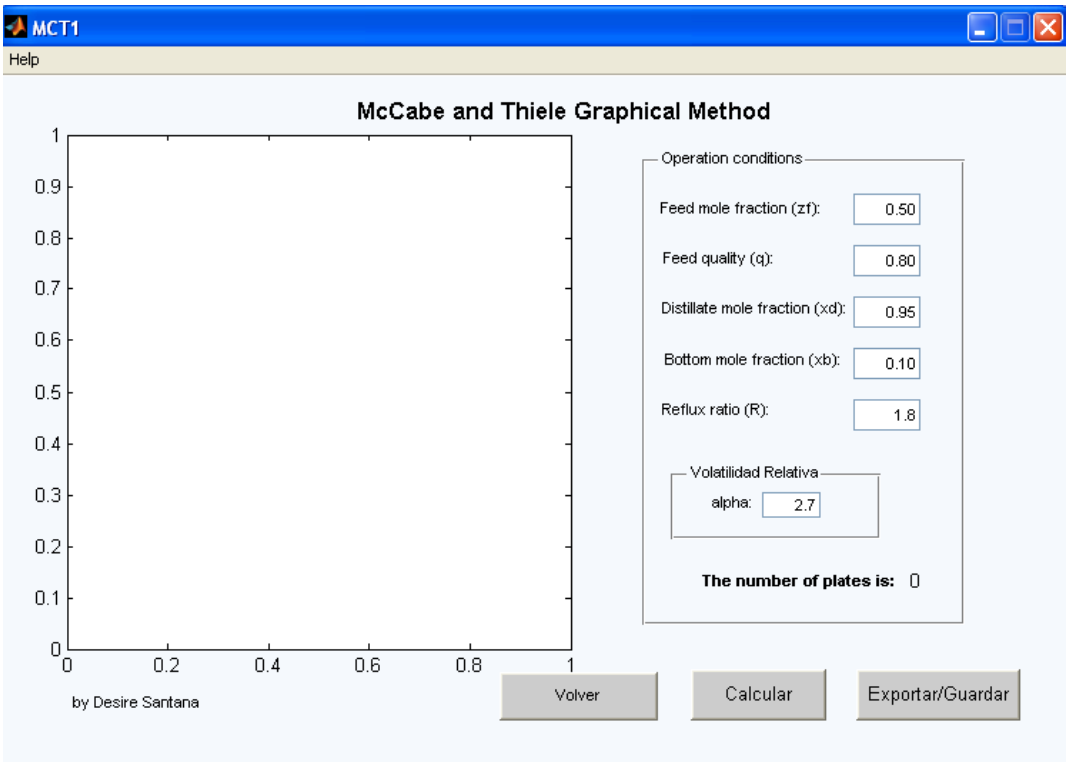
El programa de Matlab tiene dos vertiente; una de ellas es aplicar el método sin datos experimentales, este caso sirve para conocer el aspecto del método aplicar, y la otra es a partir de datos experimentales obtenidos por el usuario, aplicándole el método a los datos obtenidos por el usuario. Para ello se procede a ejecutar desde Matlab el fichero MCCCABETHIELE.m, al ejecutarlo se muestra la siguiente ventana:



A continuación según hayamos seleccionado una opción u otra vamos obteniendo la interface correspondiente. La diferencia entre una pestaña y la otra es una interface de diferencia, aunque las dos llegan a la misma ventana final de cálculos. Para ello solo demostraremos la de la pestaña con datos experimentales que es la más importante de esta interface aunque en el Anexo I se pondrá el código general. Cuando se selecciona la pestaña "Con datos Experimentales", se abre una ventana en la cual hay que introducir los datos experimentales obtenidos en el equilibrio (se ejecuta automáticamente XeY).



Como se puede comprobar esta ventana nos da la opción de guardar los datos o de limpiar datos anteriormente introducidos. Al guardar los datos se nos crea un archivo .xls, que se guardará en la misma carpeta donde se ejecuta el programa. Una vez guardados los datos de equilibrio introducidos por el usuario, se procede con esos datos a los cálculos de método mencionado anteriormente.



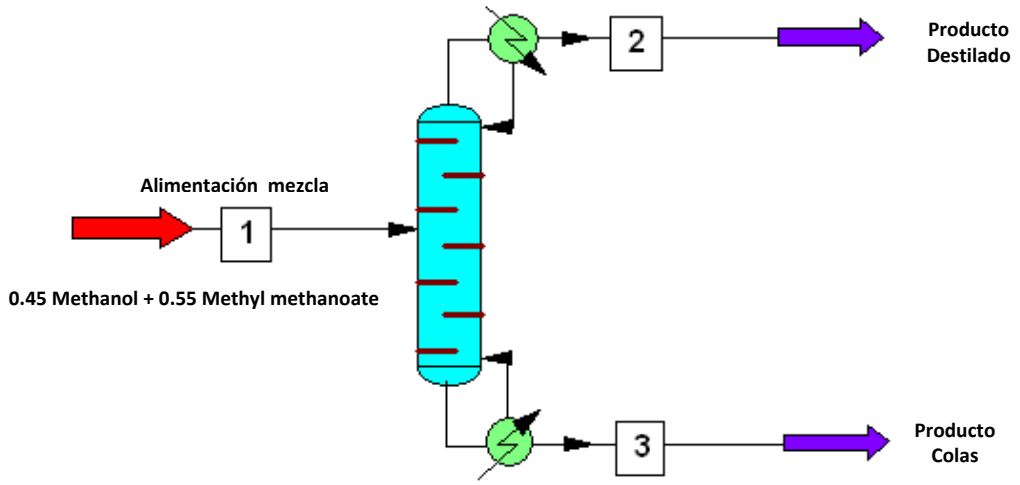
En esta ventana, es donde se procede al cálculo de los datos a partir de unos datos fijados por el propio usuario. Como se puede comprobar, la ventana nos da la posibilidad de volver, calcular y exportar/guardar. Una vez finalizados los cálculos, se procede a la obtención de los datos.

7.3. EL CHEMCAD Y EL ASPEN PLUS.

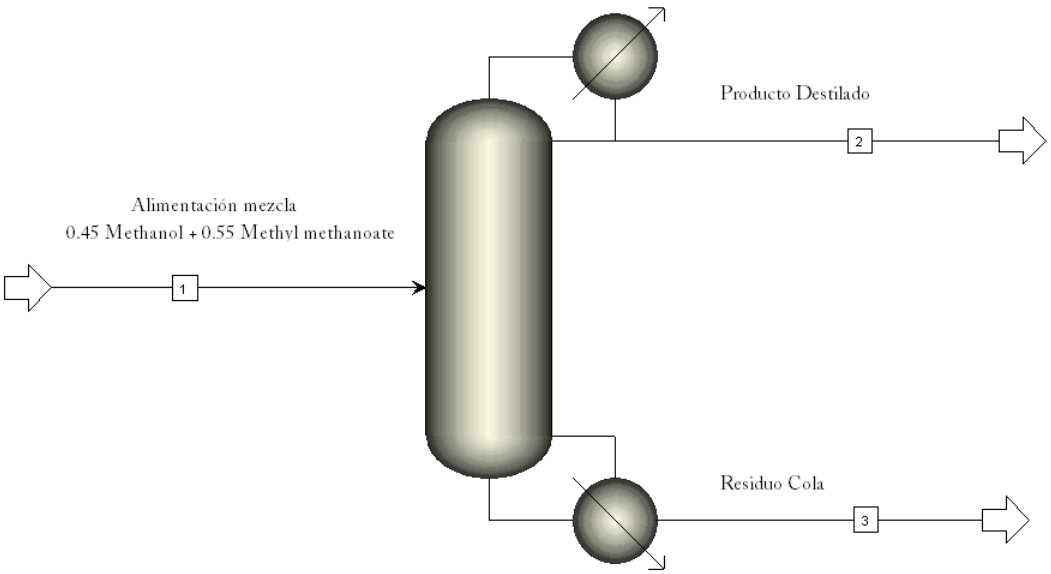
El Chemcad y el Aspen plus son programas comerciales que sirven para el estudio de compuestos puros así como el comportamiento de mezclas, y la separación de las mismas. Tienen una potente y amplia base de datos para la realización de sus simulaciones.

El Chemcad es un programa menos fiable que el Aspen plus, pero ha sido uno de los más utilizados hasta la aparición del Aspen. Ambos son fáciles de manejar conociendo tanto los compuestos con los que se va a realizar las diferentes simulaciones, así como teniendo un poco de conocimiento del manejo de cada uno de ellos. El chemcad utilizado en el trabajo es el Chemcad 5.1.x mientras que el Aspen es el Aspen Plus 11.1. En cada uno de ellos se ha utilizado una columna de destilación similar. Podemos ver las columnas utilizadas en cada uno de ellos:

a) Chemcad 5.1.x.: en el chemcad hemos utilizado una columna denominada Tower#1. Hemos seleccionado este tipo de columna porque es la más apropiada para nuestra mezcla debido a que posee cálculos rigurosos de equilibrio Liquido Vapor, usa el algoritmo Entrada-Salida, puede ser usado para modelos al estado estacionario o dinámicos. A dicha columna se le ha puesto una entrada de alimentación y dos salidas, una de productos y otra de residuo. El esquema que presenta el chemcad es el siguiente:



b) Aspen Plus 11.1: se ha utilizado la columna de destilación Ratefract1, que es la más similar a la destilación que queremos realizar. Este tipo de columna nos permite unos exhaustivos cálculos de los datos de equilibrio líquido vapor, usando un algoritmo de entrada-salida. El esquema que presenta el aspen en su pantalla es el siguiente:

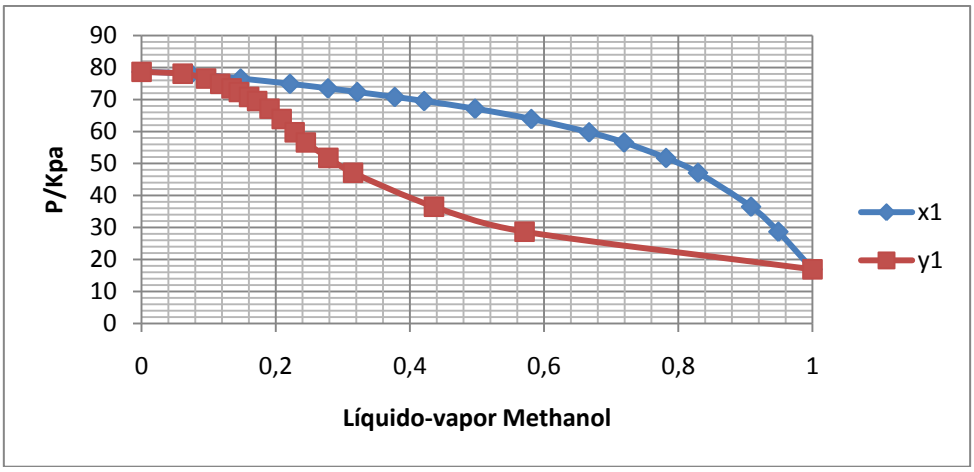


En el manejo de estos software no se va a profundizar debido a que nos extenderíamos demasiado y no es el objetivo de dicho trabajo. Podemos consultar para ello www.Chemcad Ejemplos\Tutorial Chemcad\Scutcol.htm para el manejo del Chemcad, y Process simulation and control using Aspen para el Aspen plus.

8. PROBLEMA PROPUESTO.

Una vez comprendido el concepto de equilibrio, los balances, la importancia de la destilación en los procesos de ingeniería química, las columnas de destilación y el método de McCabe-Thiele, procedemos al problema propuesto para la implementación del programa realizado, así como su posterior comparación con los programas comerciales.

Para ello partimos de unos datos experimentales de equilibrio publicados en la revista Int. Data Ser. (Sel. Data mixtures, Ser. A 1997, 25(4), 280-303) para la mezcla methyl methanoate + methanol. Se utilizará como sustancia primera el methanol y como segunda el ester. La representación gráfica de estos datos de equilibrio son:



En esta representación gráfica se puede observar esos datos experimentales frente a la presión, recordando que x_1 e y_1 son respectivamente la fase líquido y la fase vapor del compuesto 1 de la mezcla.

Estos datos experimentales se someten a un ajuste para la aplicación del método, en este problemas se ha decidido aplicar el principio de mínimos cuadrados para ajustar sencillos modelos manuales, ya que el empleo del modelo de las Zetas, u otros habitualmente utilizados en la literatura (NRTL, Wilson, UNiQUAC), complicaría muchísimo el trabajo de resolución de los datos, así como el aumento del tiempo del cálculo del programa, quedando como una cuestión pendiente para un posible trabajo futuro.

Como se había comentado los datos experimentales se someten a un ajuste de mínimos cuadrados, debido a que es el ajuste más sencillo y efectivo que se puede aplicar en mezclas binarias. Se tiene que recordar que el método de McCabe-Thiele no acepta mezclas azeotrópicas. Las mezclas azeotrópicas son aquellas que poseen un

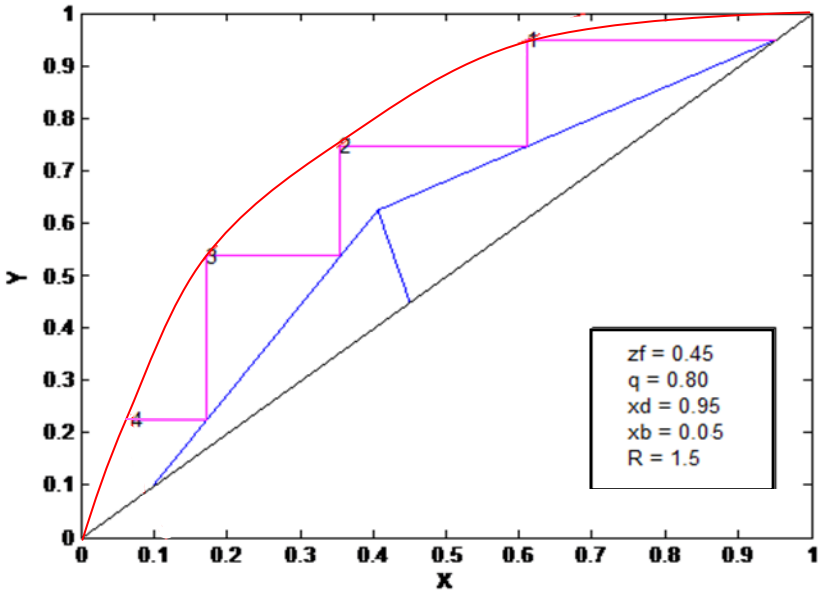
punto de ruptura en la fase del equilibrio, el cual dificulta la separación de las mezclas. A este tipo de mezclas es imposible aplicarles el ajuste de mínimos cuadrados debido a que poseen un comportamiento cúbico.

Siguiendo con el problema se fijan las condiciones del mismo, para ello la mezcla binaria entra en la fase de alimentación con una temperatura de 298,15 K, una presión de 101,32 KPa, una fracción molar de 0,45% de metanol y 0,55% de methyl methanoate. Siendo la calidad de la mezcla de un 80% (q=0,80), el reflujo es de 1,5 (R=1,5), la fracción molar que se quiere destilar es del 95% (x_D=0,95) y el residuo en cola de un 5% (x_B=0,05).

Con los datos experimentales y estas condiciones lo implementamos tanto en el programa realizado en el Matlab, como en el Chemcad y el Aspen plus. Y se obtienen los siguientes resultados según el programa aplicado.

8.1. RESULTADOS.

En Matlab con los datos experimentales se obtienen los siguientes resultados.



Como se puede comprobar en la gráfica, para los datos experimentales y los datos anteriormente fijados, nos sale que la mejor separación para la mezcla binaria de methanol + methyl methanoate, es una columna de destilación de 4 platos, siendo el plato óptimo de alimentación el segundo. Aplicamos las mismas condiciones al programa del Chemcad y al Aspen plus y se obtienen los siguientes resultados mostrados en la **tabla 1.3**.

Tabla 1.3. Datos obtenidos (fase líquida, fase vapor y temperatura) para cada uno de los platos para la mezcla binaria Methanol +Methyl Methanoate, utilizando el McCabe-Thiele, el Chemcad y el Aspen Plus.

McCabe-Thiele				Chemcad				Aspen Plus			
Plato	x_1	y_1	T/K	Plato	x_1	y_1	T/K	Plato	x_1	y_1	T/K
1	0,6127	0,7476	305,22	1	0,7819	0,8319	306,23	1	0,6955	0,2714	305,71
2	0,3546	0,5928	306,98	2	0,6107	0,7215	310,81	2	0,4111	0,1698	307,30
3	0,1724	0,2238	307,17	3	0,4713	0,6568	316,24	3	0,3379	0,1498	308,05
4	0,0696	0,0481	312,13	4	0,4127	0,5923	321,12	4	0,1498	0,0922	313.17
				5	0,2820	0,4274	321,14				
				6	0,1369	0,1757	321,26				

Como se puede comprobar nuestro método nos da cuatro platos de columna para una separación óptima y el plato de alimentación el segundo. Si comparamos nuestro método con el Aspen vemos que poseen los mismos platos de columna y la alimentación es la misma. Mientras que el Chemcad con las mismas condiciones de entrada, no sale una columna de 6 platos y con plato de alimentación el cuarto.

Si comparamos estos resultados, nuestros resultados se asemeja más al Aspen que al Chemcad. Los resultados obtenidos en el Chemcad son muy diferentes tanto comparados con nuestro método como con el chemcad. Se debe tener en cuenta que cuando menos platos tenga una columna de destilación menos costo tendrá, por lo tanto, para fabricar una columna de destilación se prefieren los datos obtenidos por el Aspen y nuestro método.

Ahora bien para comprobar que de verdad nuestro método es tan fiable como el Aspen y algo diferente al Chemcad, se procede a la captura de los datos utilizados por cada uno de los programas, datos de equilibrio, y se representan en nuestro método. Al aplicarlos se obtienen los resultados que se muestran en la siguiente **tabla 1.4.**

Tabla 1.4. Datos obtenidos (fase líquida, fase vapor y temperatura) por el método de McCabe-Thiele a partir de los datos de equilibrio proporcionados por el Chemcad y el Aspen Plus para la mezcla binaria Methanol +Methyl Methanoate

Método de McCabe-Thiele Con Datos del Chemcad				Método de McCabe-Thiele Con Datos del Aspen Plus			
Plato	x_1	y_1	T/K	Plato	x_1	y_1	T/K
1	0,7999	0,8599	306,10	1	0,6847	0,7908	305,69
2	0,6427	0,7656	308,25	2	0,4502	0,6501	307,19
3	0,4973	0,6784	311,14	3	0,2889	0,5533	307,31
4	0,3807	0,6084	315,21	4	0,1195	0,1333	308,07
5	0,2720	0,4074	317,54	5	0,0668	0,0433	313,25
6	0,1269	0,1737	319,12				
7	0,0397	0,0335	321,10				

Como se puede ver en la tabla 1.4 al aplicarle a nuestro método los datos de Chemcad se ha obtenido 7 platos de equilibrio, siendo el mismo el plato de alimentación. Mientras que con los datos del Aspen, aumenta un plato la columna y el plato de alimentación es el tercero.

8.2. CONCLUSIONES.

Como se puede observar, nuestro programa con los datos del chemcad coincide con el plato de alimentación pero no con el número total de platos de la columna, y con el aspen aumenta en los platos totales pero simultáneamente baja el plato de alimentación, pudiéndose equiparar con los datos obtenidos anteriormente.

Con la obtención de estos resultados se pueden elaborar una serie de afirmaciones:

- Los resultados obtenidos por nuestro método son fiables.
- Los datos que utiliza el Chemcad no son tan buenos como los datos que usa el Aspen.
- Los datos que usa el Aspen plus son más parecidos a los experimentales, debido a eso sus resultados son tan semejantes como los del método de McCabe-Thiele.
- Para una futura simulación de columnas de destilación se utilizara tanto nuestra interfaz como el Aspen plus para la obtención de una buena columna.

Con todas estas hipótesis se puede llegar a la conclusión que se ha alcanzado el objetivo de dicho trabajo, que consistía implementar un método en una interface para la simulación de columnas de destilación.

9. OTRO MÉTODO.

Como se comento anteriormente, un posible trabajo futuro puede ser la implementación de esta interface que tenga como función objetivo el modelo de las Zetas. Dicho modelo pertenece al Grupo de Termodinámica y Fisicoquímica de Fluidos de la Universidad de Las Palmas de Gran Canaria.

Dicho modelo se ha aplicado en los últimos años a diversos trabajos del grupo. El modelo consiste en un modelo paramétrico que determina la correlación de las propiedades termodinámicas de exceso en función de la composición, temperatura y presión, tanto para sistemas binarios como ternarios. Las propiedades termodinámicas de exceso son aquellas que nos dicen el comportamiento de las sustancias. La expresión genérica de esta función es:

$$M_N^E = z_i(1 - z_i)[A_0 + A_1 z_i + A_2 z_i^2 + \cdots + A_{N-2} z_i^{N-2}]$$

donde z_i es:

$$z_i = \frac{x_i}{x_i + \sum_{j\neq 1}^{n} k_{ji}\, x_j}$$

siendo z_i específica para cada sistema binario o terciario, debido a que no solo depende de la fracción molar del mismo sino de la constante k_{ji} que es específica de cada sustancia.

Como se puede observar si nosotros aplicamos a nuestro método la función objetivo anterior se nos quedaría una expresión muy complicada, lo cual llevaría un tiempo considerable la comprobación del mismo. Por eso se deja planteado como un posible trabajo futuro.

10. Bibliografía.

- www.Chemcad Ejemplos\Tutorial Chemcad\Scutcol.htm
- Process simulation and control using Aspen
- Introducción a la Termodinámica en Ingeniería Química, McGraw-Hill, quinta edición.
- Ingeniería de procesos de separación, Pearson.
- http://www.biodiesel-expansion.com/articles/history-of-biodiesel.php
- Producción de Biodiesel. Aplicaciones a países en desarrollo, Edición: Ingeniería Sin Fronteras
- http://www.biocarburantesmagazine.com
- Biodiesel state-of-theart and innovation existing and new Technologies applied in the recycling of ufo and biodiesel production.

11.Anexo I.

En este anexo se encuentra el código del programa utilizado en Matlab, para una posible reproducción del mismo.

11.1.MCCABETHIELE.m

```
function varargout = MCCABETHIELE(varargin)
% MCCABETHIELE M-file for MCCABETHIELE.fig
%      MCCABETHIELE, by itself, creates a new MCCABETHIELE or raises
the existing
%      singleton*.
%
%      H = MCCABETHIELE returns the handle to a new MCCABETHIELE or
the handle to
%      the existing singleton*.
%
%      MCCABETHIELE('CALLBACK',hObject,eventData,handles,...) calls
the local
%      function named CALLBACK in MCCABETHIELE.M with the given input
arguments.
%
%      MCCABETHIELE('Property','Value',...) creates a new MCCABETHIELE
or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before MCCABETHIELE_OpeningFcn gets called.
An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to MCCABETHIELE_OpeningFcn via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MCCABETHIELE

% Last Modified by GUIDE v2.5 16-Jul-2010 10:55:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @MCCABETHIELE_OpeningFcn, ...
                   'gui_OutputFcn',  @MCCABETHIELE_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MCCABETHIELE is made visible.
function MCCABETHIELE_OpeningFcn(hObject, eventdata, handles,
varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to MCCABETHIELE (see VARARGIN)

% Choose default command line output for MCCABETHIELE
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MCCABETHIELE wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = MCCABETHIELE_OutputFcn(hObject, eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close MCCABETHIELE %Cierra el GUI actual
MCT1 %Abre el siguiente GUI llamado MCT1

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close MCCABETHIELE %Cierra el GUI actual
XeY %Abre el siguiente GUI llamado XeY
```

11.2. MCT1.m

```
function varargout = MCT1(varargin)
```

```
% MCT1 M-file for MCT1.fig
%      MCT1, by itself, creates a new MCT1 or raises the existing
%      singleton*.
%
%      H = MCT1 returns the handle to a new MCT1 or the handle to
%      the existing singleton*.
%
%      MCT1('Property','Value',...) creates a new MCT1 using the
%      given property value pairs. Unrecognized properties are passed
via
%      varargin to MCT1_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      MCT1('CALLBACK') and MCT1('CALLBACK',hObject,...) call the
%      local function named CALLBACK in MCT1.M with the given input
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MCT1

% Last Modified by GUIDE v2.5 16-Jul-2010 11:42:22

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @MCT1_OpeningFcn, ...
    'gui_OutputFcn',  @MCT1_OutputFcn, ...
    'gui_LayoutFcn',  [], ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MCT1 is made visible.
function MCT1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for MCT1
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MCT1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = MCT1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function zf_Callback(hObject, eventdata, handles)
% hObject    handle to zf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of zf as text
%        str2double(get(hObject,'String')) returns contents of zf as a
double

% --- Executes during object creation, after setting all properties.
function zf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function q_Callback(hObject, eventdata, handles)
% hObject    handle to q (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of q as text
%        str2double(get(hObject,'String')) returns contents of q as a
double

% --- Executes during object creation, after setting all properties.
function q_CreateFcn(hObject, eventdata, handles)
% hObject    handle to q (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function xd_Callback(hObject, eventdata, handles)
% hObject    handle to xd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xd as text
%        str2double(get(hObject,'String')) returns contents of xd as a
double

% --- Executes during object creation, after setting all properties.
function xd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function xb_Callback(hObject, eventdata, handles)
% hObject    handle to xb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xb as text
%        str2double(get(hObject,'String')) returns contents of xb as a
double

% --- Executes during object creation, after setting all properties.
function xb_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function R_Callback(hObject, eventdata, handles)
% hObject    handle to R (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of R as text
```

```
%        str2double(get(hObject,'String')) returns contents of R as a
double

% --- Executes during object creation, after setting all properties.
function R_CreateFcn(hObject, eventdata, handles)
% hObject    handle to R (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function alpha_Callback(hObject, eventdata, handles)
% hObject    handle to alpha (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of alpha as text
%        str2double(get(hObject,'String')) returns contents of alpha
as a double

% --- Executes during object creation, after setting all properties.
function alpha_CreateFcn(hObject, eventdata, handles)
% hObject    handle to alpha (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in calculate.
function calculate_Callback(hObject, eventdata, handles)
% hObject    handle to calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get data from GUI
alpha = str2double(get(handles.alpha,'String'));
R = str2double(get(handles.R,'String'));
q = str2double(get(handles.q,'String'));
zf = str2double(get(handles.zf,'String'));
xb = str2double(get(handles.xb,'String'));
xd = str2double(get(handles.xd,'String'));

% Error checks
if any([zf xb xd] >= 1) || any([zf xb xd] <= 0)
    errordlg('The molar fractions (zf,xb,xd) must be between 0 and
1!')
    return
```

```
end

if q > 1 || q < 0
    errordlg('The feed quality must be between 0 and 1 (0 <= q <=
1)!')
    return
end

if alpha < 1
    errordlg('Alpha must be greater than 1!')
    return
end

% Computation of equilibrium curve
ye = 0:0.001:1;
xe = equilib(ye,alpha);

% Computing the intersection of feed line and operating lines
xi = (-(q-1)*(1-R/(R+1))*xd-zf)/((q-1)*R/(R+1)-q);
yi = (zf+xd*q/R)/(1+q/R);

yi2 = interp1(xe,ye,xi);
if yi > yi2
    errordlg('The distillation is not possible! Try a different
operation condition.')
    return
end

% plotting operating feed lines and equilibrium curve
axes(handles.axes1)
cla
hold on
plot(xe,ye,'r','LineWidth',1)
xlabel('X','FontWeight','bold'), ylabel('Y','FontWeight','bold')
axis([0 1 0 1])
set(line([xd xi,zf xi,xb xi],[xd yi,zf yi,xb yi]),'Color','b')
set(line([0 1],[0 1]),'Color','k')

% Stepping off stages
% Rectifying section
i = 1;
xp(1) = xd;
yp(1) = xd;
y = xd;
while xp(i) > xi
    xp(i+1)= equilib(y,alpha);
    yp(i+1)= R/(R+1)*xp(i+1)+xd/(R+1);
    y = yp(i+1);
    set(line([xp(i) xp(i+1)],[yp(i) yp(i)]),'Color','m')
    text(xp(i+1),yp(i),num2str(i))
    if xp(i+1) > xi
        set(line([xp(i+1) xp(i+1)],[yp(i) yp(i+1)]),'Color','m')
    end
    i = i+1;
end

% Stripping section
ss = (yi-xb)/(xi-xb);
yp(i) = ss*(xp(i)-xb)+xb;
```

```
y = yp(i);
set(line([xp(i) xp(i)],[yp(i-1) yp(i)]),'Color','m')

while xp(i) > xb
    xp(i+1) = equilib(y,alpha);
    yp(i+1) = ss*(xp(i+1)-xb)+xb;
    y = yp(i+1);
    set(line([xp(i) xp(i+1)],[yp(i) yp(i)]),'Color','m');
    text(xp(i+1),yp(i),num2str(i))
    if xp(i+1) > xb
        set(line([xp(i+1) xp(i+1)],[yp(i) yp(i+1)]),'Color','m')
    end
    i = i+1;

end
hold off

% Write on the GUI the final number of plates
set(handles.numplates,'String',i-1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%
function x = equilib(y,alpha)
% Constant relative volatility model
x = y./(alpha-y*(alpha-1));

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

fig2 = figure;

alpha = get(handles.alpha,'String');
R = get(handles.R,'String');
q = get(handles.q,'String');
zf = get(handles.zf,'String');
xb = get(handles.xb,'String');
xd = get(handles.xd,'String');

% copy axes into the new figure (this is not trivial)
new_handle = copyobj(handles.axes1,fig2);
set(new_handle, 'units', 'normalized', 'position', [0.13 0.11 0.775
0.815]);
text(0.75,0.35,['zf = ' zf]); text(0.75,0.3,['q = ' q]);
text(0.75,0.25,['xd = ' xd])
text(0.75,0.2,['xb = ' xb]); text(0.75,0.15,['R = ' R]);
text(0.75,0.1,['alpha = ' alpha])
rectangle('Position',[0.7,0.05,0.25,0.35])

% Save the graph with a unique name
hgsave(new_handle,genvarname(['mctd_' datestr(clock, 'HHMMSS')]))

% --------------------------------------------------------------------
function HelpMenu_Callback(hObject, eventdata, handles)
% hObject    handle to HelpMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --------------------------------------------------------------------
function AboutMenu_Callback(hObject, eventdata, handles)
% hObject    handle to AboutMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Credits and final reference (Text Box in Help/About)
help.message = {{'Esto es una GUI que ha sido creada para mostrar el
método the McCabe and Thiele: '; ...
    '';'McCabe, Smith and Harriott. Unit Operations of Chemical
Engineering, '; ...
    'McGraw-Hill, 7th Edition, 2004.'; ...
    '';'The autor wants to acknowledge the function "McCabe-Thiele
Method for an Ideal Binary Mixture" (FileID = 4472) by Housam
Binous.';...
    '';'Comments, bugs or suggestions, please write to
cgelmi@gmail.com'; ...
      ;'Desire Santana, Pontificia Universidad Católica de Chile.';
...
    '';'http://www.systemsbiology.cl'};'About this GUI'};
msgbox(help.message{1},help.message{2})


% --- Executes on button press in Volver.
function Volver_Callback(hObject, eventdata, handles)
% hObject    handle to Volver (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close MCT1 %cierra el programa
MCCABETHIELE
```

11.3. XeY.m

```
function varargout = XeY(varargin)
% XEY M-file for XeY.fig
%      XEY, by itself, creates a new XEY or raises the existing
%      singleton*.
%
%      H = XEY returns the handle to a new XEY or the handle to
%      the existing singleton*.
%
%      XEY('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in XEY.M with the given input
arguments.
%
%      XEY('Property','Value',...) creates a new XEY or raises the
%      existing singleton*.  Starting from the left, property value
pairs are
%      applied to the GUI before XeY_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to XeY_OpeningFcn via varargin.
%
```

```
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help XeY

% Last Modified by GUIDE v2.5 17-Jul-2010 14:32:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @XeY_OpeningFcn, ...
                   'gui_OutputFcn',  @XeY_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before XeY is made visible.
function XeY_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to XeY (see VARARGIN)

% Choose default command line output for XeY
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes XeY wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = XeY_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1
as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7
as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8
as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9
as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12
as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13
as a double


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14
as a double


% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
%        str2double(get(hObject,'String')) returns contents of edit15
as a double


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit16
as a double


% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit17_Callback(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit17 as text
%        str2double(get(hObject,'String')) returns contents of edit17
as a double


% --- Executes during object creation, after setting all properties.
function edit17_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit18_Callback(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit18 as text
%        str2double(get(hObject,'String')) returns contents of edit18
as a double


% --- Executes during object creation, after setting all properties.
```

```
function edit18_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit19_Callback(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit19 as text
%        str2double(get(hObject,'String')) returns contents of edit19
as a double


% --- Executes during object creation, after setting all properties.
function edit19_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit20_Callback(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit20 as text
%        str2double(get(hObject,'String')) returns contents of edit20
as a double


% --- Executes during object creation, after setting all properties.
function edit20_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit21_Callback(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit21 as text
%        str2double(get(hObject,'String')) returns contents of edit21
as a double


% --- Executes during object creation, after setting all properties.
function edit21_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit21 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit22_Callback(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit22 as text
%        str2double(get(hObject,'String')) returns contents of edit22
as a double


% --- Executes during object creation, after setting all properties.
function edit22_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function edit23_Callback(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit23 as text
%        str2double(get(hObject,'String')) returns contents of edit23
as a double


% --- Executes during object creation, after setting all properties.
function edit23_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit23 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit24_Callback(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit24 as text
%        str2double(get(hObject,'String')) returns contents of edit24
as a double


% --- Executes during object creation, after setting all properties.
function edit24_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit24 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit25_Callback(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit25 as text
%        str2double(get(hObject,'String')) returns contents of edit25
as a double


% --- Executes during object creation, after setting all properties.
function edit25_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit25 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit26_Callback(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit26 as text
%        str2double(get(hObject,'String')) returns contents of edit26
as a double


% --- Executes during object creation, after setting all properties.
function edit26_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit27_Callback(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit27 as text
%        str2double(get(hObject,'String')) returns contents of edit27
as a double


% --- Executes during object creation, after setting all properties.
function edit27_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit28_Callback(hObject, eventdata, handles)
% hObject    handle to edit28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit28 as text
%        str2double(get(hObject,'String')) returns contents of edit28
as a double


% --- Executes during object creation, after setting all properties.
function edit28_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit29_Callback(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit29 as text
%        str2double(get(hObject,'String')) returns contents of edit29
as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit29_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit30_Callback(hObject, eventdata, handles)
% hObject    handle to edit30 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit30 as text
%        str2double(get(hObject,'String')) returns contents of edit30
as a double


% --- Executes during object creation, after setting all properties.
function edit30_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit30 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit31_Callback(hObject, eventdata, handles)
% hObject    handle to edit31 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit31 as text
%        str2double(get(hObject,'String')) returns contents of edit31
as a double


% --- Executes during object creation, after setting all properties.
function edit31_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit31 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit32_Callback(hObject, eventdata, handles)
% hObject    handle to edit32 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit32 as text
%        str2double(get(hObject,'String')) returns contents of edit32
as a double


% --- Executes during object creation, after setting all properties.
function edit32_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit32 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit33_Callback(hObject, eventdata, handles)
% hObject    handle to edit33 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit33 as text
%        str2double(get(hObject,'String')) returns contents of edit33
as a double


% --- Executes during object creation, after setting all properties.
function edit33_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit33 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end



function edit34_Callback(hObject, eventdata, handles)
% hObject    handle to edit34 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit34 as text
%        str2double(get(hObject,'String')) returns contents of edit34
as a double


% --- Executes during object creation, after setting all properties.
function edit34_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit34 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit35_Callback(hObject, eventdata, handles)
% hObject    handle to edit35 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit35 as text
%        str2double(get(hObject,'String')) returns contents of edit35
as a double


% --- Executes during object creation, after setting all properties.
function edit35_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit35 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit36_Callback(hObject, eventdata, handles)
% hObject    handle to edit36 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit36 as text
%        str2double(get(hObject,'String')) returns contents of edit36
as a double


% --- Executes during object creation, after setting all properties.
function edit36_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit36 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in Guardar.
function Guardar_Callback(hObject, eventdata, handles)
% hObject    handle to Guardar (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x1= str2double(get(handles.edit1, 'String'));
x2= str2double(get(handles.edit2, 'String'));
x3= str2double(get(handles.edit3, 'String'));
x4= str2double(get(handles.edit4, 'String'));
x5= str2double(get(handles.edit5, 'String'));
x6= str2double(get(handles.edit6, 'String'));
x7= str2double(get(handles.edit7, 'String'));
x8= str2double(get(handles.edit8, 'String'));
x9= str2double(get(handles.edit9, 'String'));
x10= str2double(get(handles.edit10, 'String'));
x11= str2double(get(handles.edit11, 'String'));
x12= str2double(get(handles.edit12, 'String'));
x13= str2double(get(handles.edit13, 'String'));
x14= str2double(get(handles.edit14, 'String'));
x15= str2double(get(handles.edit15, 'String'));
x16= str2double(get(handles.edit16, 'String'));
x17= str2double(get(handles.edit17, 'String'));
x18= str2double(get(handles.edit18, 'String'));
X=[x1;x2;x3;x4;x5;x6;x7;x8;x9;x10;x11;x12;x13;x14;x15;x16;x17;x18]
%set(handles.text2,'String',X);
%save X.mat
y1= str2double(get(handles.edit19, 'String'));
y2= str2double(get(handles.edit20, 'String'));
y3= str2double(get(handles.edit21, 'String'));
y4= str2double(get(handles.edit22, 'String'));
y5= str2double(get(handles.edit23, 'String'));
y6= str2double(get(handles.edit24, 'String'));
y7= str2double(get(handles.edit25, 'String'));
y8= str2double(get(handles.edit26, 'String'));
```

```
y9= str2double(get(handles.edit27, 'String'));
y10= str2double(get(handles.edit28, 'String'));
y11= str2double(get(handles.edit29, 'String'));
y12= str2double(get(handles.edit30, 'String'));
y13= str2double(get(handles.edit31, 'String'));
y14= str2double(get(handles.edit32, 'String'));
y15= str2double(get(handles.edit33, 'String'));
y16= str2double(get(handles.edit34, 'String'));
y17= str2double(get(handles.edit35, 'String'));
y18= str2double(get(handles.edit36, 'String'));
Y=[y1;y2;y3;y4;y5;y6;y7;y8;y9;y10;y11;y12;y13;y14;y15;y16;y17;y18]
%-----------------------------------------------------------------
f=1;

for i=1:18

% excel en a guardamos valor de X
      pri = 'a'; %guardo en el excel la fecha variando el contador
      ff=int2str(f);
      reg  = [pri,ff];
      vec = X(f);

      %      xlswrite ('valores.xls', vec, 'Sheet 1', reg)

      f=f+1;   %incremento contador para ir memorizando la fecha en el
excel


end

f=1;
for i=1:18

% excel en b guardamos valor de
      pri = 'b'; %guardo en el excel la fecha variando el contador
      ff=int2str(f);
      reg  = [pri,ff];
      vec = Y(f);

%      xlswrite ('valores.xls', vec, 'Sheet 1', reg)

      f=f+1;   %incremento contador para ir memorizando la fecha en el
excel


end

xlswrite ('desi.xls', [X,Y])
%[out]=fminsearch('cuadratica2', [1 1 1])
%a = out(1)
%b = out(2)
%c = out(3)
close XeY %Cierra el GUI actual
MCT2 %Abre el siguiente GUI llamado MCT2

% --- Executes on button press in Limpiar.
function Limpiar_Callback(hObject, eventdata, handles)
% hObject    handle to Limpiar (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%Limpiar pantalla
ini=char(' ');
set(handles.edit1,'String',ini);
set(handles.edit2,'String',ini);
set(handles.edit3,'String',ini);
set(handles.edit4,'String',ini);
set(handles.edit5,'String',ini);
set(handles.edit6,'String',ini);
set(handles.edit7,'String',ini);
set(handles.edit8,'String',ini);
set(handles.edit9,'String',ini);
set(handles.edit10,'String',ini);
set(handles.edit11,'String',ini);
set(handles.edit12,'String',ini);
set(handles.edit13,'String',ini);
set(handles.edit14,'String',ini);
set(handles.edit15,'String',ini);
set(handles.edit16,'String',ini);
set(handles.edit17,'String',ini);
set(handles.edit18,'String',ini);
set(handles.edit19,'String',ini);
set(handles.edit20,'String',ini);
set(handles.edit21,'String',ini);
set(handles.edit22,'String',ini);
set(handles.edit23,'String',ini);
set(handles.edit24,'String',ini);
set(handles.edit25,'String',ini);
set(handles.edit26,'String',ini);
set(handles.edit27,'String',ini);
set(handles.edit28,'String',ini);
set(handles.edit29,'String',ini);
set(handles.edit30,'String',ini);
set(handles.edit31,'String',ini);
set(handles.edit32,'String',ini);
set(handles.edit33,'String',ini);
set(handles.edit34,'String',ini);
set(handles.edit35,'String',ini);
set(handles.edit36,'String',ini);
```

11.4. MCT2.m

```
function varargout = MCT2(varargin)
% MCT2 M-file for MCT2.fig
%      MCT2, by itself, creates a new MCT2 or raises the existing
%      singleton*.
%
%      H = MCT2 returns the handle to a new MCT2 or the handle to
%      the existing singleton*.
%
%      MCT2('Property','Value',...) creates a new MCT2 using the
%      given property value pairs. Unrecognized properties are passed
via
%      varargin to MCT2_OpeningFcn.  This calling syntax produces a
%      warning when there is an existing singleton*.
%
%      MCT2('CALLBACK') and MCT2('CALLBACK',hObject,...) call the
%      local function named CALLBACK in MCT2.M with the given input
```

```
%      arguments.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help MCT2

% Last Modified by GUIDE v2.5 17-Jul-2010 14:11:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @MCT2_OpeningFcn, ...
    'gui_OutputFcn',  @MCT2_OutputFcn, ...
    'gui_LayoutFcn',  [], ...
    'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before MCT2 is made visible.
function MCT2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   unrecognized PropertyName/PropertyValue pairs from the
%            command line (see VARARGIN)

% Choose default command line output for MCT2
handles.output = hObject;

% Copia datos entrada
handles.abc=varargin;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes MCT2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = MCT2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function zf_Callback(hObject, eventdata, handles)
% hObject    handle to zf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of zf as text
%        str2double(get(hObject,'String')) returns contents of zf as a
double

% --- Executes during object creation, after setting all properties.
function zf_CreateFcn(hObject, eventdata, handles)
% hObject    handle to zf (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function q_Callback(hObject, eventdata, handles)
% hObject    handle to q (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of q as text
%        str2double(get(hObject,'String')) returns contents of q as a
double

% --- Executes during object creation, after setting all properties.
function q_CreateFcn(hObject, eventdata, handles)
% hObject    handle to q (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function xd_Callback(hObject, eventdata, handles)
% hObject    handle to xd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xd as text
```

```
%        str2double(get(hObject,'String')) returns contents of xd as a
double

% --- Executes during object creation, after setting all properties.
function xd_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xd (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function xb_Callback(hObject, eventdata, handles)
% hObject    handle to xb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of xb as text
%        str2double(get(hObject,'String')) returns contents of xb as a
double

% --- Executes during object creation, after setting all properties.
function xb_CreateFcn(hObject, eventdata, handles)
% hObject    handle to xb (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function R_Callback(hObject, eventdata, handles)
% hObject    handle to R (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of R as text
%        str2double(get(hObject,'String')) returns contents of R as a
double

% --- Executes during object creation, after setting all properties.
function R_CreateFcn(hObject, eventdata, handles)
% hObject    handle to R (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function alpha_Callback(hObject, eventdata, handles)
% hObject    handle to alpha (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of alpha as text
%        str2double(get(hObject,'String')) returns contents of alpha
as a double

% --- Executes during object creation, after setting all properties.
function alpha_CreateFcn(hObject, eventdata, handles)
% hObject    handle to alpha (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




% --- Executes on button press in calculate.
function calculate_Callback(hObject, eventdata, handles)
% hObject    handle to calculate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%global a
%global b
%global c

[out]=fminsearch('cuadratica2', [1 1 1])
a = out(1);
b = out(2);
c = out(3);
save desi.mat




% obtener datos de la GUI
alpha = str2double(get(handles.alpha,'String'));
R = str2double(get(handles.R,'String'));
```

```
q = str2double(get(handles.q,'String'));
zf = str2double(get(handles.zf,'String'));
xb = str2double(get(handles.xb,'String'));
xd = str2double(get(handles.xd,'String'));

% Error de controles
if any([zf xb xd] >= 1) || any([zf xb xd] <= 0)
    errordlg('!La Fracción molar(zf,xb,xd) debe ser entre 0 y 1!')
    return
end

if q > 1 || q < 0
    errordlg('The feed quality must be between 0 and 1 (0 <= q <=
1)!')
    return
end

if alpha < 1
    errordlg('Alpha debe ser mayor que 1!')
    return
end

% Calculo de la curva de equilibrio
ye = 0:0.001:1;
xe = prueba(ye, a, b, c);

% Cálculo de la intersección de la línea de alimentación y líneas de
% operación
xi = (-(q-1)*(1-R/(R+1))*xd-zf)/((q-1)*R/(R+1)-q);
yi = (zf+xd*q/R)/(1+q/R);

yi2 = interp1(xe,ye,xi);

if yi > yi2
    errordlg('¡La destilación no es posible! Intente unas condiciones
de destilación diferentes.')
    return
end

% trazado de funcionamiento las líneas de alimentación y la curva de
% equilibrio
axes(handles.axes1)
cla
hold on
plot(xe,ye,'r','LineWidth',1)
xlabel('X','FontWeight','bold'), ylabel('Y','FontWeight','bold')
axis([0 1 0 1])
set(line([xd xi,zf xi,xb xi],[xd yi,zf yi,xb yi]),'Color','b')
set(line([0 1],[0 1]),'Color','k')

% Etapas de salida
% Sección de retificación
i = 1;

xp(1) = xd;
yp(1) = xd;
y = xd;
while xp(i) > xi
```

```
    xp(i+1)= prueba(y, a, b, c);
    xd1= xp(i+1);

    yp(i+1)= R/(R+1)*xp(i+1)+xd/(R+1);
    y = yp(i+1);

    set(line([xp(i) xp(i+1)],[yp(i) yp(i)]),'Color','m')
    text(xp(i+1),yp(i),num2str(i))
    if xp(i+1) > xi
        set(line([xp(i+1) xp(i+1)],[yp(i) yp(i+1)]),'Color','m')
    end
    i = i+1;

    xd1
    y


  end

% Sección de cola
ss = (yi-xb)/(xi-xb);
yp(i) = ss*(xp(i)-xb)+xb;
y = yp(i);
set(line([xp(i) xp(i)],[yp(i-1) yp(i)]),'Color','m')

while xp(i) > xb
    xp(i+1) = prueba(y,a, b, c);
    xb1 = xp(i+1);

    yp(i+1) = ss*(xp(i+1)-xb)+xb;
    y = yp(i+1);

        set(line([xp(i) xp(i+1)],[yp(i) yp(i)]),'Color','m');
    text(xp(i+1),yp(i),num2str(i))
    if xp(i+1) > xb
        set(line([xp(i+1) xp(i+1)],[yp(i) yp(i+1)]),'Color','m')
    end
    i = i+1;
    xb1
    y

end
hold off

xi
yi

% Escribir en la GUI el número final de platos
set(handles.numplates,'String',i-1);
load desi.mat




%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function x = prueba(y, a, b, c)
x = c.*y.^2+b.*y+a;

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

fig2 = figure;

alpha = get(handles.alpha,'String');
R = get(handles.R,'String');
q = get(handles.q,'String');
zf = get(handles.zf,'String');
xb = get(handles.xb,'String');
xd = get(handles.xd,'String');


% Copia los ejes en la nueva gráfica (esto no es trivial)
new_handle = copyobj(handles.axes1,fig2);
set(new_handle, 'units', 'normalized', 'position', [0.13 0.11 0.775
0.815]);
text(0.75,0.35,['zf = ' zf]); text(0.75,0.3,['q = ' q]);
text(0.75,0.25,['xd = ' xd])
text(0.75,0.2,['xb = ' xb]); text(0.75,0.15,['R = ' R]);
text(0.75,0.1,['alpha = ' alpha])
rectangle('Position',[0.7,0.05,0.25,0.35])

% Guarda la gráfica con un unico nombre
hgsave(new_handle,genvarname(['mctd_' datestr(clock, 'HHMMSS')]))

% --------------------------------------------------------------------
function HelpMenu_Callback(hObject, eventdata, handles)
% hObject    handle to HelpMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function AboutMenu_Callback(hObject, eventdata, handles)
% hObject    handle to AboutMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Credits and final reference (Text Box in Help/About)
help.message = {{'Esto es una GUI que ha sido creada para mostrar el
método the McCabe and Thiele: '; ...
    '';'McCabe, Smith and Harriott. Unit Operations of Chemical
Engineering, '; ...
    'McGraw-Hill, 7th Edition, 2004.'; ...
    '';'The autor wants to acknowledge the function "McCabe-Thiele
Method for an Ideal Binary Mixture" (FileID = 4472) by Housam
Binous.';...
    '';'Comments, bugs or suggestions, please write to
cgelmi@gmail.com'; ...
     ;'Desire Santana, Pontificia Universidad Católica de Chile.';
...
```

```
    '';'http://www.systemsbiology.cl'};'About this GUI'};
msgbox(help.message{1},help.message{2})


% --- Executes on button press in Volver.
function Volver_Callback(hObject, eventdata, handles)
% hObject    handle to Volver (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close MCT2 %cierra el programa
MCCABETHIELE
```