Construction d’un diagramme binaire EVL

Méthodes numériques et VBA

Livrable Projet N°2

Audrey Kuate Kengne 300327115

Chadjar Norlie Elva 300331199

Alima Ba 300203689

Bineta Ly 300382852

Dans le cadre du cours CHG1771

Méthode numérique et programmation donné par : Johnny Matta

Département de Génie chimique et biologique

Université d’Ottawa

05 avril 2024

Avant-propos

Ce rapport présente le développement d'un outil Excel VBA destiné à la visualisation des diagrammes T_{xy} binaires, outil inestimable pour l'analyse des équilibres liquide-vapeur dans les mélanges chimiques. Le projet met en œuvre des concepts thermodynamiques, s'appuyant sur des méthodes numériques adaptées à Excel. Parmi les équations clés intégrées, nous trouvons l'équation de Wagner pour les propriétés de saturation, la loi de Dalton pour les pressions partielles dans les mélanges de gaz, et la loi de Raoult pour calculer les pressions de vapeur des composants dans des solutions idéales.

Un défi majeur du projet a été de surmonter les limitations numériques d'Excel, telles que la gestion des très petites valeurs et les problèmes de division, qui sont inhérents aux calculs complexes exigés par l'équation de Wagner et les autres principes thermodynamiques appliqués. Pour résoudre ces équations avec une fiabilité et une précision optimale, nous avons adopté la méthode de *Ridders*, réputée pour sa capacité à fournir une convergence rapide et précise, même face à des équations non linéaires complexes.

Une attention particulière a été accordée à la représentation graphique d'un équilibre vapeur-liquide, permettant une interprétation visuelle et intuitive des températures du point de bulle et du point de rosée en fonction de la composition des mélanges.

Table des matières

Table des équations

Tables des figures

1 Contexte et conception

1.1 Contexte

Pour réussir ce projet, nous devons concevoir un outil flexible et efficace capable de calculer les températures du point de bulle et du point de rosée pour différents mélanges binaires de composés chimiques. Cette solution doit pouvoir s'adapter facilement à diverses pressions et compositions de mélanges, tout en restant robuste et générale. Nous utiliserons VBA dans Excel pour réaliser cela, en nous appuyant sur les paramètres de Wagner disponibles dans un fichier Excel supplémentaire pour assurer une mise à jour aisée et une réutilisabilité du code. En résumé, notre objectif est de fournir un outil pratique pour les professionnels de l'industrie chimique, permettant une modélisation précise des équilibres liquide-vapeur dans les mélanges binaires de manière efficace et accessible.

1.2 Conception

Dans le cadre de la conception, des calculs préliminaires ont été effectué à partir des différentes équations données. Effectivement, des lois et équations (cités ci-dessous) ont été utilisés et transformés, afin d'estimer la pression de vapeur des substances pures à différentes températures. Le calcul de la pression de vapeur est la base de l'analyse vapeur-liquide (EVL) pour notre mélange. Ensuite, ces équations non linéaires seront résolues grâce à la méthode de *Ridders*.

1.2.1 Équations et calculs:

Soit i un composé dans un mélange de 2 composés. P est la pression globale, P_i est la pression partielle de i dans un mélange de n composés. D'abord, nous allons commencer par la loi de Dalton qui se traduit par :

$$P = \sum_{i=1}^{n} P_i \qquad [1]$$

De ce fait, pour estimer notre Pression nous avons besoin de la pression partielle d'où intervient la loi de Raoult se traduisant par :

$$P_i = x_i P_i^*(T_{bp}) \qquad [2]$$

Avec x_i : Fraction molaire liquide du composé i pur dans le mélange et $P_i^*(T_{bp})$: Pression de vaporisation. Néanmoins n'ayant pas $P_i^*(T)$,nous devrons nous baser sur l'équation de Wagner pour obtenir l'équation finale. L'équation se traduit ci- suit :

$$\ln\left(\frac{P_i^*(T)}{P_c}\right) = \frac{A\tau + B\tau^{1,5} + C\tau^{2,5} + D\tau^5}{1-\tau} \quad [3]$$

Avec P_c : Pression critique de i et A, B, C les paramètres de Wagner. τ (tho) se traduit par l'équation suivante :

$$\tau = 1 - \frac{T}{T_c} \quad [4]$$

Avec T : Température donnée et T_c : Température critique de i. Enfin pour un système binaire avec deux composés A et B respectivement l'éthanol et p-xylène, l'équation de T_{pr} est :

$$P = \frac{1}{\frac{y_A}{P_A^*(T_{pr})} + \frac{1-y_A}{P_B^*(T_{pr})}} \quad [5]$$

Avec

$$y_B = 1 - y_A \quad [6]$$

A partir de l'équation de l'équation de Wagner, nous allons tirer $P_i^*(T)$ afin de le remplacer dans l'équation 5 :

$$P_i^*(T) = P_c . e^{\frac{A\tau + B\tau^{1,5} + C\tau^{2,5} + D\tau^5}{1-\tau}} \quad [7]$$

Ainsi nous obtenons l'équation non linéaire suivante :

$$\frac{1}{\frac{y_A}{P_{CA} . e^{\frac{A\tau_A + B\tau_A{}^{1,5} + C\tau_A{}^{2,5} + D\tau_A{}^5}{1-\tau_A}}} + \frac{1-y_A}{P_{CB} . e^{\frac{A\tau_B + B\tau_B{}^{1,5} + C\tau_B{}^{2,5} + D\tau_B{}^5}{1-\tau_B}}}} - P = 0 \quad [8]$$

Avec

$$P = x_A P_A^*(T_{bp}) + (1 - x_A) P_B^*(T_{bp}) \quad [9]$$

1.2.2 Méthode de Ridders:

La méthode de *Ridders* est une technique de recherche de racines pour les fonctions non linéaires. Elle est basée sur une approche itérative qui réduit progressivement l'intervalle contenant la racine jusqu'à ce qu'une approximation suffisamment précise de la racine soit obtenue. Voici comment cette méthode fonctionne en utilisant les notations données :

- Initialisation : Tout d'abord, trois points sont choisis : la limite inférieure x_l, la limite supérieure x_u et le point médian x_M. Le point médian est calculé comme étant la moyenne des limites inférieure et supérieure, soit

$$x_M = \frac{(x_l + x_u)}{2} \quad [10]$$

- Calcul de la racine estimée : La racine estimée x_r est calculée à partir des valeurs de la fonction aux limites inférieure et supérieure, ainsi qu'au point médian, en utilisant une formule appropriée. Cette formule permet d'estimer la position de la racine de manière plus précise que la simple moyenne des limites inférieure et supérieure ;
- Détermination de l'intervalle contenant la racine : On évalue la fonction aux points x_M et x_r Si le produit de ces évaluations est négatif, cela signifie qu'il existe une racine entre x_l et x_u, donc on réduit l'intervalle en définissant de nouvelles limites inférieure et supérieure. Sinon, la racine se trouve en dehors de cet intervalle, donc on réajuste les limites en conséquence ;
- Réduction de l'intervalle : En fonction de l'intervalle où la racine est estimée, les limites inférieure et supérieure sont mises à jour pour se resserrer autour de la racine ;
- Critère de convergence : On répète les étapes précédentes jusqu'à ce que la différence relative entre deux itérations successives de la racine estimée x_r soit inférieure à un certain seuil de tolérance ξ_a, ou jusqu'à ce que la différence relative entre deux itérations successives de l'intervalle $x_u - x_l$ soit inférieure à un certain seuil ξ_s. Une fois ce critère de convergence atteint, la dernière valeur de x_r est considérée comme l'approximation finale de la racine.

En résumé, la méthode de *Ridders* est une méthode itérative qui utilise des points spécifiques et des évaluations de fonction pour estimer la position de la racine d'une manière qui converge rapidement et de manière fiable vers une solution précise.

1.2.3 Fonctionnement du code (noms des fonctions utilisées, description globales)

1.2.3.1 Structure Générale du Code

Notre code se compose de plusieurs modules et fonctions conçus pour calculer et afficher des diagrammes T_{xy}(Les températures en fonction de la composition) pour des mélanges binaires.

1.2.3.1.1 Module 1: Création de diagramme (CreateTxyDiagram)

Ce module crée un graphique (diagramme) qui représente les points de bulle et de rosée d'un mélange binaire en fonction de la composition. Voici ses principales composantes :

- **Initialisation et Nettoyage** : Le script commence par initialiser les variables et nettoyer les graphiques et données existantes pour préparer la feuille à de nouvelles données. Ceci pourrit éviter l'empilage de graphiques sachant, que ces derniers sont directement construits grâce au code ;
- **Création de Tableau de Données** : Un tableau est dynamiquement rempli avec des valeurs calculées pour les fractions molaires x et y, ainsi que les températures de bulle et de rosée, en Kelvin et Celsius. Ces calculs sont basés sur des itérations de la fraction molaire x de 0 à 1 avec un pas de 0.1 ;
- **Création et Configuration du Graphique** : Après le remplissage du tableau, un graphique est généré et configuré pour afficher les séries de données pour les températures de bulle et de rosée. Ce processus comprend la spécification du type de graphique, la source des données, et la personnalisation visuelle comme les titres des axes et du graphique.

1.2.3.1.2 Module 2: Initialisation des paramètres

Ce module est dédié à la configuration des variables globales et paramètres nécessaires aux calculs thermodynamiques, y compris :

- **Variables Globales** : Définition des constantes et paramètres, comme la pression, tolérance, et coefficients de l'équation transformée de Wagner ;
- **Fonctions de Calcul** : Des fonctions comme **`Wagner`**, **`Tpb`**, **`Tpr`**, **`PressionPointBulle, et PressionPointRosee`** sont définies pour effectuer les calculs spécifiques de pression basés sur les températures et compositions du mélange.

1.2.3.1.3 Modules 3 et 4: Calcul des Points de Bulle (T_{bp}) et de Rosée (T_{pr})

Ces modules contiennent des fonctions spécifiques pour calculer respectivement les températures de bulle et de rosée en utilisant :

- **Recherche Incrémentale** : La recherche incrémentale est une technique simple mais puissante pour localiser un intervalle contenant la solution d'une équation non-linéaire (comme la température de bulle ou de rosée) en explorant de manière séquentielle un ensemble de valeurs possibles. Cela est particulièrement utile pour des problèmes où une

solution analytique exacte est difficile à obtenir. Elle permet de trouver l'intervalle parfaite pour trouver la solution. De ce fait, ceci évite au programme de prendre trop de temps à charger ;

- **Méthode de Ridders** : Une méthode numérique efficace pour trouver une racine (ou température) qui satisfait l'équation de pression du mélange, avec un mécanisme d'affinement pour assurer la précision. Elle est plus complexe à implémenter que la recherche incrémentale, mais offre une convergence plus rapide et plus précise vers la solution.

1.2.3.2 Optimisations et Sécurité

Le code incorpore des mécanismes pour gérer les cas spéciaux, comme l'évitement de la division par zéro et la limitation des valeurs à des seuils physiquement raisonnables, par exemple, en s'assurant que les températures calculées ne dépassent pas les températures critiques des composants.

1.2.3.3 Déclaration des Variables et Constantes

- **Variables** : Les variables permettent de stocker des données temporaires qui seront manipulées et potentiellement modifiées par le code. En VBA, il est bon de spécifier le type de données (comme `Double`, `Long`, `String`) pour chaque variable afin d'optimiser l'utilisation de la mémoire et de guider le compilateur sur la nature des données traitées;
- **Constantes** : Les constantes sont des valeurs qui, une fois définies, ne changent pas au cours de l'exécution du code. Elles sont utiles pour stocker des valeurs fixes telles que la pression initiale `P_initial`, qui peut être utilisée à travers différents calculs sans risque de modification accidentelle.

1.2.3.4 Structures de Contrôle

- `Do Loop Until` : Cette boucle permet d'exécuter un ensemble d'instructions jusqu'à ce qu'une condition spécifique soit remplie. Dans le code, elle est utilisée pour la recherche incrémentale et pour assurer la convergence des méthodes de calcul.
- `If`, `If...Else` : Ces instructions conditionnelles permettent d'exécuter différents blocs de code en fonction de la véracité d'une ou plusieurs conditions. Elles sont cruciales pour gérer les décisions logiques dans le code, comme la sélection d'intervalle pour la méthode de Ridders ou la vérification de la convergence des résultats.
- `Call` : Cette instruction est utilisée pour invoquer d'autres procédures ou fonctions au sein du code. Elle facilite la modularisation et la réutilisation du code, en permettant de structurer le programme en blocs fonctionnels distincts.

2 Résultat et discussion

D'abord, le diagramme EVL Txy généré via VBA illustre une corrélation claire entre la fraction molaire des composants et les températures correspondantes de bulle et de rosée. Comme attendu, la température de bulle décroît avec une augmentation de la fraction molaire de l'éthanol, ce qui est conforme aux attentes compte tenu de la volatilité plus élevée de l'éthanol par rapport au p-xylène. La température de rosée suit une tendance inverse, augmentant avec la fraction molaire du p-xylène.

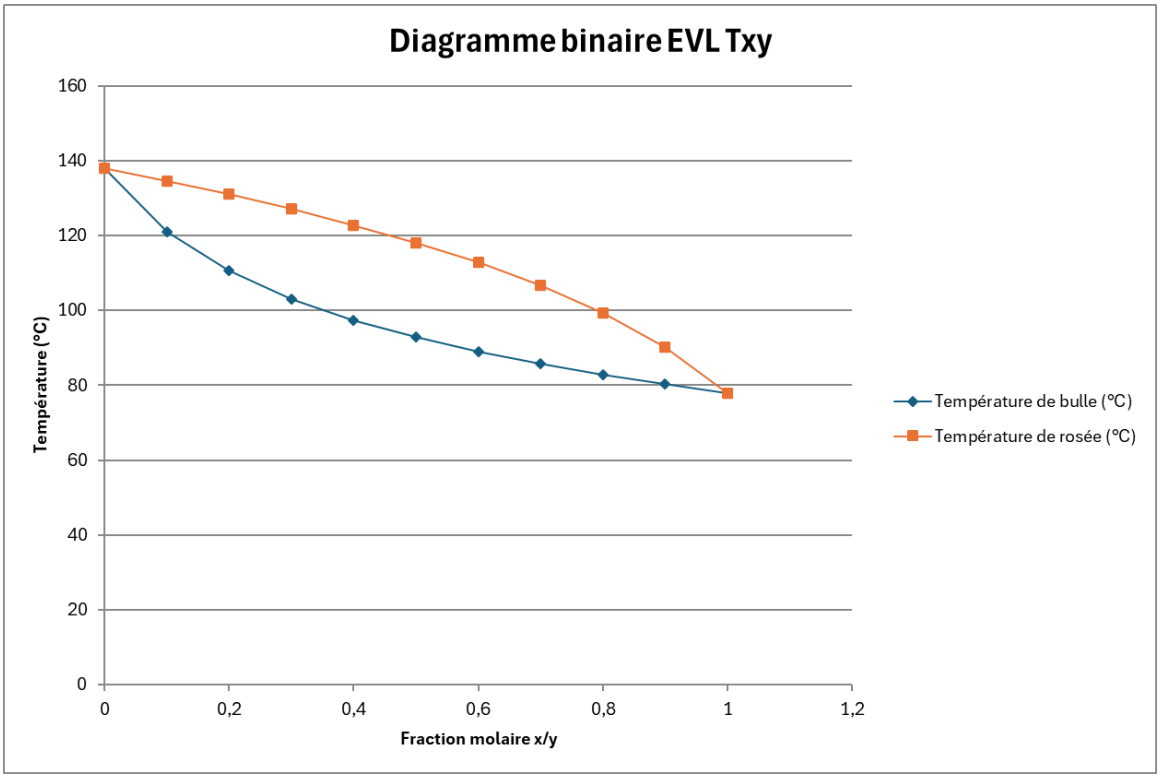

Figure 2-1-Diagramme binaire EVL Txy obtenu à partir d'Excel

Ainsi, en comparant les courbes de température de bulle et de rosée issues du code VBA par rapport à celles dérivées des calculs Excel et des données expérimentales, on note une similitude dans les tendances. Toutefois, des écarts mineurs peuvent être observés, probablement en raison des différences méthodologiques et des approximations des modèles thermodynamiques. En particulier, la présence d'un point azéotropique dans le graphique expérimental, absent du résultat VBA, pourrait refléter des limites dans les méthodes numériques employées ou dans la caractérisation des composés à certaines concentrations.

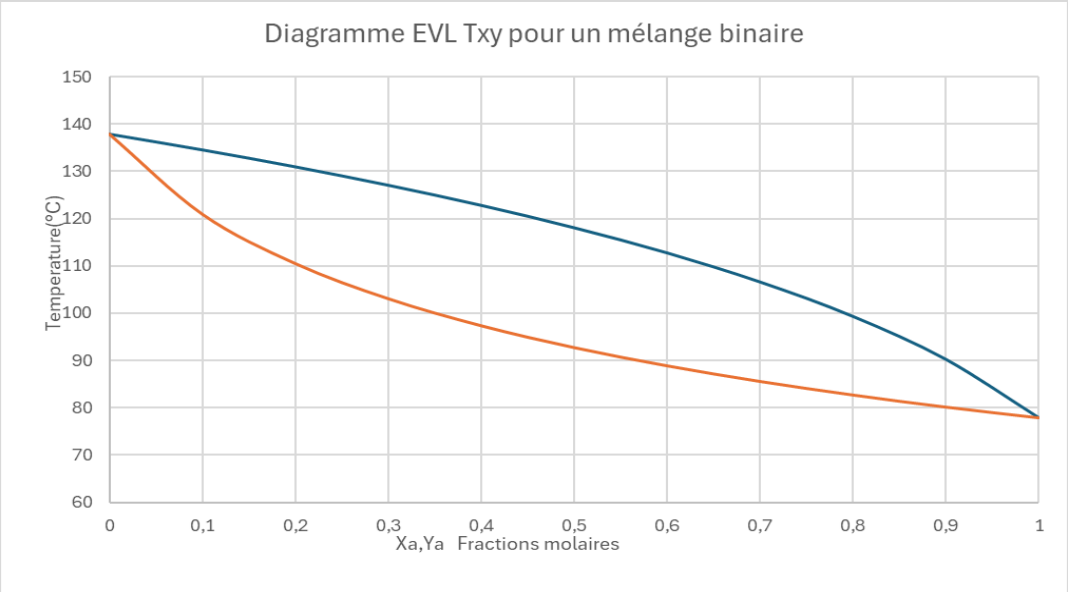

Figure 2-3 Diagramme Binaire Txy EVL à partir des calculs Excel

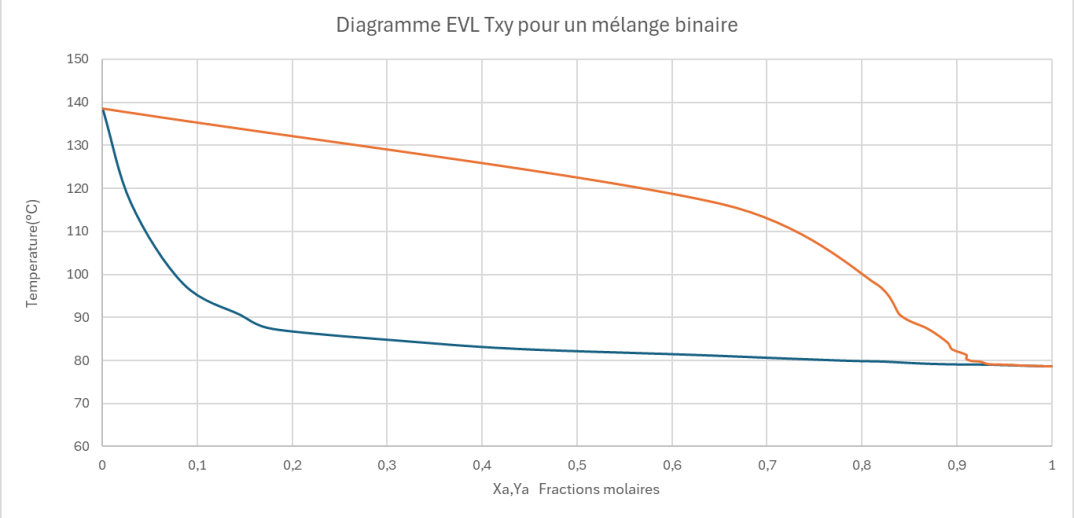

Figure 2-2- Diagramme Binaire EVL Txy à partir des données expérimentales

De ce fait, la correspondance entre les températures des points de bulle et de rosée obtenues par les calculs VBA et les valeurs expérimentales pour un système binaire d'éthanol et de p-xylène valide l'application de l'équation de Wagner dans le cadre d'une pression constante pour ce type de mélange. Il faut noter que les calculs VBA marchent pour énormément de composés. Néanmoins, ces calculs ne marcheront pas sur certains composés cités ci-dessous.

Ensuite, la tolérance de 0,001 est choisie pour équilibrer précision et efficacité calculatoire, évitant des itérations inutilement longues tout en assurant des résultats fiables. Le facteur de correction de 0,1 optimise la convergence des méthodes numériques, accélérant le processus sans compromettre la stabilité des résultats. En effet, un facteur trop élevé pourrait risquer de

sauter par-dessus la solution réelle, tandis qu'un facteur trop faible pourrait rendre la convergence trop lente, augmentant le nombre d'itérations requises pour atteindre la tolérance spécifiée. Cependant, des défis émergent lors de la modélisation de composés tels que l'hélium et l'hydrogène à cause de leurs paramètres de Wagner inadaptés, mettant en évidence les limites du modèle standard face à des propriétés physiques non conventionnelles. Des solutions ont été envisagées en vain, ils n'ont pas pu régler le problème. Ainsi, pour gérer les problèmes liés à la gestion des très petites valeurs, comme celles rencontrées avec les paramètres de l'hélium et de l'hydrogène dans l'équation de Wagner, nous avons étudié l'ajustement des valeurs avant de procéder aux calculs. Cela incluait la vérification et la limitation des valeurs de l'exponentielle pour éviter les erreurs d'exécution dues à des nombres trop grands ou trop petits pour la fonction Exp. Voici un extrait du genre de code que nous avons envisagé pour cela :

```
Dim exponent As Double
exponent = (A * tho + B * tho ^ 1.5 + C * tho ^ 2.5 + D * tho ^ 5) / (T / Tc)
' Limiter les valeurs de l'exponent pour éviter les dépassements
If exponent > 709 Or exponent < -708 Then ' Les valeurs exactes peuvent varier
    Wagner = CVErr(xlErrNum) ' Utiliser xlErrNum pour indiquer une erreur
numérique
Else
    Wagner = Exp(exponent) * Pc
End If
```

Et pour les très petites valeurs qui pourraient causer une division par zéro ou des résultats incorrects lors du calcul de tho :

```
Dim tho As Double
tho = 1 - T / Tc
Dim almostZero As Double
almostZero = 0.00000000000001

If Abs(tho) < almostZero Then
    tho = almostZero
End If
```

L'`almostZero` est utilisé pour éviter les problèmes dans les calculs de l'équation de Wagner, où une valeur très proche de zéro pourrait autrement causer une division par zéro ou des résultats imprécis lors du calcul des pressions de vapeur. En assignant une petite valeur positive à `almostZero`, nous assurons que les calculs restent valides et stables, même pour des composés comme l'hélium et l'hydrogène, qui présentent des défis particuliers en raison de leurs très petites valeurs dans certaines conditions thermodynamiques.

En définitive, ces défis révèlent l'impératif d'adapter ou de peaufiner les modèles thermodynamiques afin de capturer avec précision les comportements distincts de divers

composés, surtout ceux qui divergent de l'idéalité. Ce projet illustre bien la complexité inhérente à la modélisation des systèmes chimiques et souligne la nécessité de trouver un compromis entre rigueur théorique et applicabilité pratique.

3 Conclusion

Ce projet visait à créer un outil dans Excel via VBA pour visualiser les diagrammes T_{xy}, utilisant les données de l'éthanol et du p-xylène pour modéliser les équilibres liquide-vapeur. La méthode de *Ridders* a permis d'affiner les calculs, qui ont été validés par comparaison avec des résultats expérimentaux. Des problèmes sont survenus, notamment dans la modélisation de l'hélium et de l'hydrogène, mettant en évidence les limites de l'outil face aux gaz légers. Ces résultats soulignent l'efficacité de l'outil pour la plupart des applications mais aussi le besoin d'améliorations continue pour gérer des cas plus complexes.

4 Références

-Vapor-liquid Equilibrium, Activity Coefficient, Fugacity | Chemical Engineering | JoVE

5 Annexe

5.1 Table des variables

Variables	Définition	Unité
$\boldsymbol{P}$	Pression totale du mélange	bar
P_i	Pression partielle du composant i dans le mélange	bar
$\boldsymbol{x}_i$	Fraction molaire liquide du composant i dans le mélange	adimensionelle
$\boldsymbol{y}_i$	Fraction molaire vapeur du composant i dans le mélange	adimensionelle
T_{bp}	Température de bulle du mélange	Kelvin
T_{pr}	Température de rosée du mélange	Kelvin
$P_i^*(T_{pr})$	Pression de liquide du mélange	bar
$P_i^*(T_{bp})$	Pression de vapeur du mélange	bar
P_c	Pression critique	bar
$\boldsymbol{T_c}$	Température critique	Kelvin

5.2 Tableau du mélange binaire

Ci-dessous le tableau du mélange binaire utilisé pour réaliser la solution préliminaire.

Tableau 1 : Tableau du mélange binaire

Propriétés des composés	Composé A	Composé B
Nom	Ethanol	p-Xylene
Paramètre de Wagner (A)	-8.68587	-7.71694
Paramètre de Wagner (B)	1.17831	1.89119
Paramètre de Wagner (C)	-4.8762	-2.39695
Paramètre de Wagner (D)	1.588	-3.63026
Propriété du système		
Pression globale (bar)		1

5.3 Tableau de Wagner

Ci-dessous, un tableau des différents composés chimiques et de leurs paramètres de Wagner est présenté :

Tableau 2. Paramètres de Wagner et paramètres critiques pour différents composés chimiques

Composé chimique (en anglais)	A	B	C	D	T_c (K)	Pc (bar)
2-Methyl propanoic acid (C4H8O2)	-8,53258	1,30605	-5,2242	-2,05813	605	37
3-Methyl butanoic acid (C5H10O2)	-8,67381	1,62939	-6,51756	-2,08757	629	34
Acetic Acid (C2H4O2)	-8,29430	0,97928	-0,21745	-5,72367	592,71	57,86
Butanoic acid (C4H8O2)	-8,42953	1,34333	-5,37332	-2,74438	624	40,3
Decanoic acid (C10H20O2)	-9,07060	2,77535	-11,1014	-2,43545	726	22,3
Formic acid (CH2O2)	-7,24917	0,44255	-0,35558	-0,96906	588	58,07
Octanoic acid (C8H16O2)	-9,04015	2,16529	-8,66117	-4,69516	695	26,4
Pentanoic acid (C5H10O2)	-8,76701	1,5499	-6,19961	-4,21927	643	35,8
Propanoic acid (C3H6O2)	-8,14882	0,7959	-3,1836	-3,81338	604	45,3
1-butanol (C4H10O)	-8,40615	2,2301	-8,2486	-0,711	563,05	44,24
1-Decanol (C10H22O)	-9,75478	4,18634	-7,0572	-15,98	689	24,1
1-dodecanol (C12H26O)	-9,91901	3,61884	-5,8537	-18,204	720	20,8
1-eicosanol (C20H42O)	11,23154	3,669	-7,0775	-14,321	809	13
1-heptadecanol (C17H36O)	10,73125	3,55515	-6,3591	-15,696	780	15
1-Heptanol (C7H16O)	-9,68778	5,35716	-10,1672	-8,01	632,5	31,35
1-hexadecanol (C16H34O)	10,54087	3,4726	-6,077	-15,939	770	16,1
1-Hexanol (C6H14O)	-9,49034	5,13288	-10,5817	-5,154	610,7	34,7
1-Nonanol (C9H20O)	-9,91542	5,1367	-8,8075	-12,497	671,5	26,3
1-octadecanol (C18H38O)	10,91637	3,57835	-6,6199	-15,06	790	14,4
1-Octanol (C8H18O)	10,01437	5,90629	-10,4026	-9,048	652,5	28,6
1-Pentanol (C5H12O)	-8,98005	3,91624	-9,9081	-2,191	588,15	39,09
2-butanol (C4H10O)	-8,09820	1,64406	-7,49	-5,27355	536,01	41,98
2-ethyl-1-hexanol (C8H18O)	-9,61812	5,17861	-9,1144	-11,004	640,5	27,99
2-Octanol (C8H18O)	-9,37352	4,7376	-8,3382	-11,646	638	28,9
Benzyl alcohol (C7H8O)	-7,29099	1,17084	-4,7167	-5,53	715	43
Cyclohexanol (C6H12O)	-7,12838	1,40189	-5,60756	-9,57158	650	42,6
Ethanol (C2H6O)	-8,68587	1,17831	-4,8762	1,588	513,92	61,32
Isopropyl alcohol (C3H8O)	-8,73656	2,1624	-8,70785	4,77927	508,3	47,62
Methanol (CH4O)	-8,63571	1,17982	-2,479	-1,024	512,64	80,92
Propanol (C3H8O)	-8,53706	1,96214	-7,6918	2,945	536,78	51,68
Tert-butanol (C4H10O)	-8,47927	2,47845	-9,27918	-2,53992	506,2	39,73
Acetone (C3H6O)	-7,55098	1,60784	-1,9944	-3,2002	508,1	47,02
Cyclopentanone (C5H8O)	-7,36589	1,54092	-2,28143	-3,0514	624,5	46
Methyl isobutyl ketone (C6H12O)	-7,70040	1,69968	-2,80448	-3,81623	574,6	32,7
Benzene (C6H6)	-7,01433	1,55256	-1,8479	-3,713	562,16	48,98
Ethylbenzene (C8H10)	-7,53139	1,75439	-2,42012	-3,57146	617,2	36
Naphthalene (C10H8)	-7,61444	1,91553	-2,5075	-3,23	748,4	40,5
Toluene (C7H8)	-7,31600	1,59425	-1,93165	-3,7222	591,8	41,06
Pentafluorobenzene (C6HF5)	-7,86799	1,71659	-2,53582	-4,59937	530,97	35,37
Pentafluorotoluene (C7H3F5)	-8,08717	1,76131	-2,72838	-4,13797	566,52	31,24

m-Xylene (C8H10)	-7,67717	1,8024	-2,47745	-3,66068	617,05	35,38
o-Xylene (C8H10)	-7,60491	1,75383	-2,27531	-3,73771	630,33	37,35
p-Xylene (C8H10)	-7,71694	1,89119	-2,39695	-3,63026	616,23	35,16
Acetic Anhydride (C4H6O3)	-8,35130	1,8905	-2,8357	-5,1156	606	40
Butane (C4H10)	-7,01763	1,6777	-1,9739	-2,172	425,25	37,92
Diethyl ether (C4H10O)	-7,43301	1,78847	-2,4793	-3,2811	466,74	36,5
Decane (C10H22)	-8,60643	2,44659	-4,2925	-3,908	617,65	21,05
Dodecane (C12H26)	-9,08593	2,77846	-5,1985	-4,173	658	18,2
Eicosane (C20H42)	10,97958	4,25588	-8,9573	-5,043	769	11,6
Ethane (C2H6)	-6,47500	1,41071	-1,14400	-1,85900	305,33	48,71
Heptadecane (C17H36)	10,23600	3,54177	-7,18980	-5,00000	735	13,7
n-Heptane (C7H16)	-7,77404	1,85614	-2,82980	-3,50700	540,15	27,35
Hexadecane (C16H34)	10,03664	3,41426	-6,86270	-4,86300	722	14,35
n-Hexane (C6H14)	-7,53998	1,83759	-2,54380	-3,16300	507,9	30,35
Methane (CH4)	-6,02242	1,26652	-0,57070	-1,36600	190,55	45,99
Nonadecane (C19H40)	10,68217	3,98054	-8,30300	-4,99500	758	12,3
Nonane (C9H20)	-8,32886	2,25707	-3,82570	-3,73200	594,9	22,9
Octadecane (C18H38)	-10,47230	3,69655	-7,57790	-5,10900	746	13
Octane (C8H18)	-8,04937	2,03865	-3,31200	-3,64800	568,95	24,9
Pentadecane (C15H32)	-9,80239	3,29217	-6,53170	-4,58400	708	15,15
n-Pentane (C5H12)	-7,30698	1,75845	-2,16290	-2,91300	469,8	33,75
Propane (C3H8)	-6,76368	1,55481	-1,58720	-2,02400	369,83	42,48
R152a (C2H4F2)	-7,43344	1,75554	-2,16995	-2,77469	386,41	45,17
n-Tetradecane (C14H30)	-9,54470	3,06637	-6,00700	-4,53000	693	16,1
Tridecane (C13H28)	-9,32959	2,89925	-5,55500	-4,47000	676	17,1
Undecane (C11H24)	-8,85076	2,60205	-4,73050	-4,08100	638,85	19,55
Argon (Ar)	-5,92654	1,20826	-0,50988	-1,59089	150,69	48,63
Nitrogen (N2)	-6,11102	1,21890	-0,69366	-1,89893	126,2	34
Ammonia (NH3)	-7,28322	1,57160	-1,85672	-2,39312	405,5	113,53
Water (H2O)	-7,86194	1,87924	-2,26680	-2,12862	647,1	220,64
Helium (He normal)	-4,26523	1,57125	0,47980	0,75127	5,2	2,27
Hydrogen (H2 normal)	-4,90262	1,06500	0,73731	0,05313	33,15	12,96

5.4 Code VBA

5.4.1 Module 1 : Création de diagramme

```
Option Explicit
'Ce module est dédié à la création de graphique qui fonctionne grace au
bouttonqui est déjà reglé dans le sheet directement
Sub CreateTxyDiagram()
    Dim P_initial As Double
```

```
    Dim x As Double
    Dim y As Double
    Dim T_bulle_K As Double ' Température de bulle en Kelvin
    Dim T_rosee_K As Double ' Température de rosée en Kelvin
    Dim T_bulle_C As Double ' Température de bulle en Celsius
    Dim T_rosee_C As Double ' Température de rosée en Celsius
    Dim chartSheet As Worksheet
    Dim lastRow As Long
    Dim dataRange As Range
    Dim chartObj As ChartObject
    Dim chartLeft As Double
    Dim chartTop As Double

    ' Récupération de P_initial à partir de la cellule N3
    P_initial = ThisWorkbook.Sheets("VBA").Range("N3").Value

    ' Référence à la feuille de calcul pour le diagramme
    Set chartSheet = ThisWorkbook.Sheets("VBA")

    ' Effacer les données et les graphiques existantes si on veut changer les
valeurs
    chartSheet.Range("A25:F" & chartSheet.Rows.Count).ClearContents
    For Each chartObj In chartSheet.ChartObjects
        chartObj.Delete
    Next chartObj

    ' Je veux creer mon propre tableau directement,ceux sont les noms pour les
colonnes
    With chartSheet
        .Range("A25:F25").Value = Array("Fraction molaire x", "Fraction molaire
y", "Température de bulle (K)", "Température de rosée (K)", "Température de bulle
(°C)", "Température de rosée (°C)")
        .Range("A25:F25").Font.Bold = True
    End With

    ' Calcul et remplissage des données de manière dynamique
    For x = 0 To 1 Step 0.1
        y = x
        T_bulle_K = Tpb(x) 'Tpb(x) et Tpr(y) renvoient la température en Kelvin
        T_rosee_K = Tpr(y)
        T_bulle_C = T_bulle_K - 273.15 ' Conversion en °C
        T_rosee_C = T_rosee_K - 273.15 ' Conversion en °C

        lastRow = chartSheet.Cells(chartSheet.Rows.Count, 1).End(xlUp).Row + 1
```

```
        chartSheet.Cells(lastRow, 1).Resize(1, 6).Value = Array(x, y, T_bulle_K,
T_rosee_K, T_bulle_C, T_rosee_C)
    Next x

    ' Définition de la plage de données pour le graphique
    Set dataRange = chartSheet.Range("A25:F" & lastRow)

    ' Ajout des bordures au tableau
    dataRange.Borders.LineStyle = xlContinuous

    ' Positionnement du graphique à partir de H18, pour ne pas encombrer la
feuille
    chartLeft = chartSheet.Range("H18").Left
    chartTop = chartSheet.Range("H18").Top

    ' Création et configuration du diagramme de manière dynamique
    Set chartObj = chartSheet.ChartObjects.Add(Left:=chartLeft, Width:=500,
Top:=chartTop, Height:=300)
    With chartObj.Chart
        .ChartType = xlXYScatterLines
        .SetSourceData Source:=dataRange

        ' Supprimer toutes les séries existantes,j'avais un problèmes avec les
séries, donc j'avais plusieurs courbes dans le graphique
        Do While .SeriesCollection.Count > 0
            .SeriesCollection(1).Delete
        Loop

        ' Ajouter la série pour la température de bulle en °C
        .SeriesCollection.NewSeries
        With .SeriesCollection(1)
            .Name = "Température de bulle (°C)"
            .XValues = chartSheet.Range("A26:A" & lastRow)
            .Values = chartSheet.Range("E26:E" & lastRow)
            .Format.Line.Weight = 1 ' Épaisseur de la ligne
            .Format.Line.Visible = msoTrue ' Masquer/Afficher les points
        End With

        ' Ajouter la série pour la température de rosée en °C
        .SeriesCollection.NewSeries
        With .SeriesCollection(2)
            .Name = "Température de rosée (°C)"
            .XValues = chartSheet.Range("A26:A" & lastRow)
            .Values = chartSheet.Range("F26:F" & lastRow)
            .Format.Line.Weight = 1 ' Épaisseur de la ligne
```

```
            .Format.Line.Visible = msoTrue ' Afficher/Masquer les points
        End With

        ' Configuration des titres et des axes
        .HasTitle = True
        .ChartTitle.Text = "Diagramme binaire EVL Txy"
        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, xlPrimary).AxisTitle.Text = "Fraction molaire x/y"
        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue, xlPrimary).AxisTitle.Text = "Température (°C)"

        ' Réglage de la taille des courbes
        .Parent.Width = 600
        .Parent.Height = 400
    End With
End Sub
```

5.4.2 Module 2 : Initialisation des paramètres

```
Option Explicit

' Déclaration des variables globales
Global P As Double
Global tolerance As Double
Global facteur_correction As Double
Global MAX_ITERATIONS As Long
Global A1 As Double, B1 As Double, C1 As Double, D1 As Double, Tc1 As Double, Pc1
As Double
Global A2 As Double, B2 As Double, C2 As Double, D2 As Double, Tc2 As Double, Pc2
As Double

' Fonction pour l'équation transformée de Wagner
Function Wagner(ByVal A As Double, ByVal B As Double, ByVal C As Double, ByVal D
As Double, ByVal Tc As Double, ByVal Pc As Double, ByVal T As Double) As Double
    Dim tho As Double
    tho = 1 - T / Tc
    Dim almostZero As Double
    almostZero = 0.00000000000001

    If tho = 1 Then
        tho = tho + almostZero ' Pour gérer l'erreur de division par zéro
    End If
```

```
    Wagner = Exp((A * tho + B * tho ^ 1.5 + C * tho ^ 2.5 + D * tho ^ 5) / (T / Tc)) * Pc ' Pression de vapeur

    If Wagner < almostZero Then Wagner = almostZero ' Gérer une pression très faible

End Function

' Fonction pour calculer la pression totale du mélange binaire au point de bulle
Function PressionPointBulle(ByVal T As Double, ByVal x As Double) As Double
    Dim p_deb1 As Double
    Dim p_deb2 As Double

    ' Assurez-vous que les paramètres de Wagner sont initialisés
    InitializeVariablesAndParams

    p_deb1 = Wagner(A1, B1, C1, D1, Tc1, Pc1, T)
    p_deb2 = Wagner(A2, B2, C2, D2, Tc2, Pc2, T)

    PressionPointBulle = x * p_deb1 + (1 - x) * p_deb2 ' Pression totale pour le point de bulle
End Function

' Fonction pour calculer la pression totale du mélange binaire au point de rosée
Function PressionPointRosee(ByVal T As Double, ByVal y As Double) As Double
    Dim p_deb1 As Double
    Dim p_deb2 As Double

    ' Assurez-vous que les paramètres de Wagner sont initialisés
    InitializeVariablesAndParams

    p_deb1 = Wagner(A1, B1, C1, D1, Tc1, Pc1, T)
    p_deb2 = Wagner(A2, B2, C2, D2, Tc2, Pc2, T)

    PressionPointRosee = 1 / (y / p_deb1 + (1 - y) / p_deb2) ' Pression totale pour le point de rosée
End Function
' Initialise les variables et les paramètres de Wagner à partir du tableau
Sub InitializeVariablesAndParams()
    Dim ws As Worksheet
    Set ws = ThisWorkbook.Sheets("VBA")

    'Les valeurs pour P, tolerance et facteur_correction sont stockées dans la feuille
```

```
    ' Ici, je déclare  que P est en N3, tolerance en N4 et facteur_correction en
N5
    P = ws.Cells(3, 14).Value ' Ligne 3, colonne N
    tolerance = ws.Cells(4, 14).Value ' Ligne 4, colonne N
    facteur_correction = ws.Cells(5, 14).Value ' Ligne 5, colonne N

    ' Ajustement pour la récupération dynamique des paramètres de Wagner à l'aide
de VLOOKUP
    ' Je pose  que le nom du composé A est en D4 et le composé B est en D5
    ' Récupération des noms des composés sélectionnés à partir de la feuille de
calcul
    Dim nomComposéA As String
    Dim nomComposéB As String

    nomComposéA = ThisWorkbook.Sheets("VBA").Range("D4").Value
    nomComposéB = ThisWorkbook.Sheets("VBA").Range("D5").Value

    ' Utilisation des noms des composés pour récupérer les valeurs
correspondantes à partir de la table de données
    A1 = Application.WorksheetFunction.VLookup(nomComposéA, ws.Range("D4:J" &
ws.Rows.Count), 2, False)
    B1 = Application.WorksheetFunction.VLookup(nomComposéA, ws.Range("D4:J" &
ws.Rows.Count), 3, False)
    C1 = Application.WorksheetFunction.VLookup(nomComposéA, ws.Range("D4:J" &
ws.Rows.Count), 4, False)
    D1 = Application.WorksheetFunction.VLookup(nomComposéA, ws.Range("D4:J" &
ws.Rows.Count), 5, False)
    Tc1 = Application.WorksheetFunction.VLookup(nomComposéA, ws.Range("D4:J" &
ws.Rows.Count), 6, False)
    Pc1 = Application.WorksheetFunction.VLookup(nomComposéA, ws.Range("D4:J" &
ws.Rows.Count), 7, False)

    A2 = Application.WorksheetFunction.VLookup(nomComposéB, ws.Range("D4:J" &
ws.Rows.Count), 2, False)
    B2 = Application.WorksheetFunction.VLookup(nomComposéB, ws.Range("D4:J" &
ws.Rows.Count), 3, False)
    C2 = Application.WorksheetFunction.VLookup(nomComposéB, ws.Range("D4:J" &
ws.Rows.Count), 4, False)
    D2 = Application.WorksheetFunction.VLookup(nomComposéB, ws.Range("D4:J" &
ws.Rows.Count), 5, False)
    Tc2 = Application.WorksheetFunction.VLookup(nomComposéB, ws.Range("D4:J" &
ws.Rows.Count), 6, False)
    Pc2 = Application.WorksheetFunction.VLookup(nomComposéB, ws.Range("D4:J" &
ws.Rows.Count), 7, False)
    ' Définition du nombre maximal d'itérations
```

```
    MAX_ITERATIONS = 1000
End Sub
```

5.4.3 Module 3 : Point de bulle

```
Option Explicit
Function Tpb(ByVal x As Double) As Double
    Dim xl, xu, rt, dx As Double
    Dim fl, fu, fr As Double
    Dim i As Long

    xl = 273.15 ' Température de départ en K
    dx = 10 ' Pas initial pour la recherche incrémentale
    xu = xl + dx

    ' Recherche incrémentale pour localiser un intervalle contenant la racine
    Do
        fl = PressionPointBulle(xl, x) - P
        fu = PressionPointBulle(xu, x) - P

        If fl * fu < 0 Then Exit Do

        xl = xu
        xu = xl + dx
    Loop While xl < 1000 ' Limite supérieure pour éviter une boucle infinie

    ' Méthode de Ridders
    For i = 1 To MAX_ITERATIONS
        Dim xm, fm As Double

        xm = (xl + xu) / 2
        fm = PressionPointBulle(xm, x) - P

        rt = xm + Sgn(fl - fu) * fm * (xm - xl) / Sqr(Abs(fm ^ 2 - fl * fu))

        ' Ajout de la condition de sécurité basée sur Tc1 et Tc2
        If rt > Application.Min(Tc1, Tc2) Then rt = xm

        fr = PressionPointBulle(rt, x) - P

        If Abs(xu - xl) < tolerance Or Abs(fr) < tolerance Then
            Tpb = rt
            Exit Function
        End If
```

```
        ' Mise à jour des intervalles basée sur le signe de fr
        If fr * fl < 0 Then
            xu = rt
            fu = fr
        Else
            xl = rt
            fl = fr
        End If
    Next i

    ' Si la méthode n'a pas convergé, renvoie une erreur.
    Tpb = CVErr(xlErrValue)
End Function
```

5.4.4 Point de rosée

```
Option Explicit

Function Tpr(ByVal y As Double) As Double
    Dim xl, xu, rt, dx As Double
    Dim fl, fu, fr As Double
    Dim i As Long
    Call InitializeVariablesAndParams

    xl = 273.15 ' Température de départ en K (point de congélation de l'eau)
    dx = 10 ' Pas initial pour la recherche incrémentale
    xu = xl + dx

    ' Recherche incrémentale pour localiser un intervalle contenant la racine
    Do
        fl = PressionPointRosee(xl, y) - P
        fu = PressionPointRosee(xu, y) - P

        ' Teste si un changement de signe est trouvé
        If fl * fu < 0 Then
            Exit Do
        Else
            xl = xu
            xu = xl + dx
        End If
    Loop While xl < 1000 ' Limite supérieure arbitraire pour éviter la boucle
infinie

    ' Méthode de Ridders
    For i = 1 To MAX_ITERATIONS
```

```
        Dim xm, fm As Double ' Température moyenne et pression associée

        xm = (xl + xu) / 2
        fm = PressionPointRosee(xm, y) - P

        rt = xm + Sgn(fl - fu) * fm * (xm - xl) / Sqr(Abs(fm ^ 2 - fl * fu))
        fr = PressionPointRosee(rt, y) - P

        ' Ajout de la condition de sécurité basée sur Tc1 et Tc2, test de la
convergence
        If rt > Application.Min(Tc1, Tc2) Then rt = xm

        fr = PressionPointRosee(rt, y) - P

        If Abs(xu - xl) < tolerance Or Abs(fr) < tolerance Then
            Tpr = rt
            Exit Function
        End If


        ' Mise à jour des intervalles basée sur le signe de fr
        If fr * fl < 0 Then
            xu = rt
            fu = fr
        Else
            xl = rt
            fl = fr
        End If
    Next i

    ' Si la méthode n'a pas convergé, renvoie une erreur.
    Tpr = CVErr(xlErrValue)
End Function
```