UNIVERSIDAD SAN SEBASTIÁN
Facultad de Ingeniería, Arquitectura y Diseño
Escuela de Ingeniería
Ingeniería Civil en Minas
Sede Bellavista

TALLER EN EMPRESA I

INFORME FINAL

"Modelamiento de Densidad del ion LiCl con Modelos Teóricos."

Cliente: Claudio Suárez

AUTHIEVRE-CRC (Empresa canadiense)

Profesor Dr. Andrés Soto Bubert

Alejandro Díaz Quiroz

Francisco Paredes Velásquez

Laura Spinelli Cantillana

Julio, 2024

Resumen Ejecutivo

El presente estudio se centra en la modelación de la densidad de soluciones acuosas, en este caso, de salmueras del tipo Cloruro de Litio, evaluando su dependencia con la temperatura y composición química, un tema de gran relevancia para la industria minera del litio en regiones desérticas de Chile, Argentina y Bolivia. La precisión en el cálculo de la densidad de las salmueras es importante para optimizar los procesos de extracción, transporte y procesamiento del litio, un mineral de creciente demanda en la industria tecnológica.

En el contexto desértico, las empresas mineras enfrentan el desafío de operar en entornos con amplias oscilaciones térmicas. Las variaciones extremas de temperatura impactan significativamente la densidad de las salmueras, afectando directamente los balances de masa de los fluidos que ingresan y egresan de las pozas solares.

La necesidad de datos precisos sobre la densidad de las salmueras se intensifica en el contexto actual, donde las empresas mineras buscan optimizar sus operaciones mediante el uso de modelos computacionales predictivos. Estos modelos, son herramientas esenciales para la planificación y control de los procesos productivos, requieren de información confiable sobre las propiedades físicas de las salmueras, como su densidad, para generar simulaciones precisas y confiables.

Dada esto, luego de un intenso estudio en el cual se estima la densidad de soluciones acuosas mediante distintos modelamientos de cálculo de densidad, se ha determinado que el modelo adecuado que satisface las necesidades del contexto actual es el "***Modelo empírico de cálculo de densidad utilizando Inteligencia Artificial con Google Colab***", logrando realizar un análisis de las variables del problema, entregando un error porcentual de cálculo de 4,67% utilizando todos los datos disponibles, lo que indica que es un modelamiento que facilita y agiliza su proceso mediante la inteligencia artificial.

Índice

Tabla de Ilustraciones

Motivaciones

El interés presente en este estudio tiene relación con predecir la densidad para la salmuera del tipo Cloruro de Litio (LiCl), a una temperatura y composición química determinada, mediante modelamientos existentes de cálculo de densidad para soluciones acuosas, para conseguir realizar una recomendación en base a la comparación de estimaciones de error de cada modelo, y definir cuál es el adecuado para la predicción de la densidad de la salmuera LiCl.

Objetivos

Objetivo General

A partir de una base de datos experimental, se busca parametrizar en modelos predictivos teóricos, el comportamiento de la densidad del ion LiCl, frente a la variación de temperatura y composición química, con el fin de comparar el error que arrojan estos modelos predictivos.

Objetivos Específicos

1. Exponer modelamientos de cálculo de densidad existentes, útiles para una solución acuosa, incluyendo las variables temperatura y composición química.
2. Interpretar los comportamientos de la densidad para LiCl, con los diferentes modelamientos estudiados.
3. Estimar el error del cálculo de densidad respectivo de cada modelo.
4. Analizar modelos previamente evaluados, a través de una tabla comparativa.
5. Determinar cuál es el modelo adecuado que logra la menor estimación de error.

Introducción

Este informe corresponde a un ejercicio académico enfocado en la modelación, parametrización y simulación de una propiedad volumétrica de una solución acuosa, específicamente una salmuera continental proveniente de Chile, Argentina o Bolivia. A través de este estudio, se busca entender y predecir el comportamiento de dichas salmueras, específicamente del tipo “Cloruro de Litio”.

Es de interés la predicción de la densidad en salmueras y/o soluciones acuosas en función de temperatura y composición química, dado que la complejidad de las salmueras, compuestas por una mezcla de diversos iones disueltos en solución a diferentes concentraciones y temperaturas, añade un nivel adicional de dificultad al cálculo preciso de la densidad de estos fluidos, por lo tanto, se pretende obtener y comparar modelamientos de densidad de salmueras complejas con modelos predictivos teóricos.

Elección del modelo

Para la elección del modelo, se seleccionará aquel que presente el menor error para su continuación, considerando que cada modelo tiene sus ventajas y desventajas. Este parámetro de error será definido en la entrega final del trabajo, donde se adjuntará el análisis de los modelos propuestos, con sus comparaciones respectivas.

Metodología de análisis de datos para lograr la parametrización del modelo:

- Indagar modelos existentes de cálculo de densidad.
- Evaluar modelamientos de densidad de salmueras.
- Analizar los datos disponibles del cliente.
- Verificar los datos y variables, para la validación de los modelos.
- Calcular el error de los modelos.
- Comparar cuál modelo logra la menor estimación de error.
- Seleccionar el modelo adecuado para tener una correcta estimación en el cálculo de densidad de las salmueras, en este caso específico, Cloruro de Litio, gracias a la obtención del error más bajo.

Antecedentes

Para el desarrollo de este trabajo, se considera la investigación realizada previamente, donde se define la utilización de 4 modelos de cálculo de densidad para soluciones acuosas, los cuales se demuestran a continuación:

1. **Estimación de densidad de cualquier solución multielectrolítica. (Krumgalz et al.,1995)**

$$\rho = \left(1{,}000 + \sum m_i M_i\right) \Big/ \left(1{,}000 v^0 + \sum m_i \overline{V}_i^0 + V_{EX}\right)$$

Donde:

- m_i es la molalidad del componente iónico i.
- M_i es la masa molar del componente iónico i.
- v^0 es el volumen específico de agua pura ($v^0 = 1/\rho^o$).
- $\overline{V}_i^0$ es el volumen molar parcial en dilución infinita de cada componente iónico i.
- V_{EX} es el exceso de volumen de la solución.

2. **Modelamiento de cálculo de densidad de soluciones acuosas de electrolitos, basadas en volúmenes molares aparentes. (Laliberté – Cooper, 2004)**

$$\rho_{\mathrm{m}} = \frac{1}{\frac{W_{\mathrm{w}}}{\rho_{\mathrm{w}}} + \frac{W_s}{\rho_{\mathrm{app},s}}}$$

Donde:

- ρ_m es la densidad de la sal.
- ρ_w es la densidad de agua pura.
- $\rho_{app,s}$ es la densidad aparente de la sal.
- W_w es la fracción de masa del agua.
- W_s es la fracción de masa de la sal.

3. **Modelamiento de densidad de salmueras cloradas en sistemas de dos y tres componentes. (Soto-Bubert A., Miranda J., Bhardwaj R., Rajendran S., Acevedo R., 2024)**

Consiste en un modelo predictivo semi empírico que incorpora parámetros de corrección para tener en cuenta la no idealidad de las soluciones en términos de actividad iónica y fuerza de la solución. Además, introduce una componente dependiente de la temperatura, lo que lo convierte en un modelo variable continuo para la temperatura. Tiene como objetivo proporcionar una estimación más precisa de la densidad para sistemas ternarios.

Este modelamiento de densidad es desarrollado por los autores con el programa "Mathematica", y es una variante del modelo publicado por Hazim Qiblawey en "Scientific Reports" en el año 2020. Está basado en el método predictivo de densidad de soluciones salinas ternarias de Kumar (1986):

$$\rho = \frac{\left(1 + \sum_j m_j M_j\right)}{\left\{\left(\frac{1}{d_o(T)}\right)\left[\sum_j y_j(T)\left(\frac{d_0(T)}{d_j^{\circ}(T)} - 1\right) + 1\right] + \sum_j \frac{m_j M_j}{d_j^{\circ}[T]}\right\}}$$

Donde:

- $y_j(T)$ es la fracción de fuerza iónica del j – ésimo componente de la sal parametrizada.
- M_j es el peso molecular de la sal disuelta.
- m_j es la molalidad de la sal disuelta.
- $\boldsymbol{j}$ y d_j^0 es la densidad binaria del componente correspondiente al electrolito de la sal disuelta $\boldsymbol{j}$ en la solución.
- d_0 es la densidad del agua.

4. Modelo empírico de cálculo de densidad utilizando Inteligencia Artificial con Google Colab.

Consiste en modelar la densidad de la salmuera tipo Cloruro de Litio, considerando las variables temperatura y molalidad de la composición química.

Para el desarrollo de este modelo, se trabajó con 591 datos disponibles, para los cálculos y la estimación del error del modelo, considerando las variables de molalidad, densidad y temperatura.

➢ **Base y Conceptos Fundamentales:**

- La molalidad (m) es una medida de la concentración de una solución en términos de moles de soluto por kilogramo de solvente.
- La densidad de la solución depende de la densidad del solvente puro y de cómo esta se ve afectada por la presencia del soluto (efecto del Cloruro de Litio en agua en este caso).

➢ **Aproximación Empírica:**

- En muchos casos prácticos, se utiliza una aproximación empírica para relacionar la densidad de la solución con la molalidad y la temperatura. Esta aproximación puede tomar la siguiente forma:

$$\rho = \rho_0 + a * m + b * T + c * m * T$$

Donde:

- ρ es la densidad de la solución.
- ρ_0 es la densidad del solvente puro (0,997 g/cm^3).
- m es la molalidad de la solución en mol/kg.
- T es la temperatura en grados Kelvin.
- $a, b\ y\ c$ son coeficientes empíricos que se determinan ajustando la fórmula a datos disponibles.

➢ **Determinación de los Coeficientes:**

- Los coeficientes $a, b\ y\ c$ se determinan ajustando la fórmula a datos conocidos de densidad para diferentes combinaciones de molalidad y temperatura.
- Estos coeficientes pueden variar dependiendo del soluto y del solvente específicos, así como de las condiciones experimentales.

- Una vez determinados los coeficientes, se puede predecir la densidad de la solución de LiCl para variables de molalidad y temperatura dentro del rango para el cual se determinaron los coeficientes.

La ecuación que empleada para calcular la densidad está basada en una aproximación empírica que relaciona la densidad de una solución acuosa (LiCl) con su molalidad y temperatura. Esta fórmula no proviene de una fuente específica de literatura científica estándar, sino que es una representación simplificada para propósitos prácticos.

Ecuación de error:

$$ADD = \left[\sum \frac{[\rho exp, i - \rho cal, i]}{\rho exp, i}\right] \frac{100}{n}$$

Se utiliza para evaluar la precisión de la estimación de densidad proporcionada por el modelo. El error se cuantifica utilizando la fórmula de desviación absoluta media máxima (AAD).

- $\boldsymbol{ADD}$ representa la desviación absoluta media máxima, que indica la desviación porcentual promedio entre los valores de densidad experimental ($\boldsymbol{\rho exp, i}$) y los valores de densidad calculados ($\boldsymbol{\rho cal, i}$) para el sistema ternario.
- La suma se toma de todos los puntos de datos ($\boldsymbol{i}$) del conjunto de datos.
- $\boldsymbol{n}$ representa el número total de puntos de datos utilizados para el cálculo.

Desarrollo

1. **Estimación de densidad de cualquier solución multielectrolítica. (Krumgalz et al.,1995)**

El método se basa en la integración de datos experimentales y modelos empíricos, considerando las interacciones iónicas y factores de corrección para proporcionar estimaciones precisas de la densidad de tales soluciones.

2. **Modelamiento de cálculo de densidad de soluciones acuosas de electrolitos, basadas en volúmenes molares aparentes. (Laliberté – Cooper, 2004)**

Se desarrollan modelos matemáticos que relacionan la densidad de la solución con los volúmenes molares aparentes de los iones presentes. Estos modelos permiten calcular la densidad de la solución a partir de la suma de los volúmenes molares aparentes de cada ion y el solvente.

El método considera las interacciones entre los iones, utilizando parámetros que describen cómo estas interacciones afectan los volúmenes molares aparentes y, por ende, la densidad de la solución.

3. **Modelamiento de densidad de salmueras cloradas en sistemas de dos y tres componentes. (Soto-Bubert A., Miranda J., Bhardwaj R., Rajendran S., Acevedo R., 2024)**

Utilizando la base de datos disponible, se ha establecido el cálculo para la determinación de la densidad de un ion (LiCl), cálculo para el cual se ha utilizado el programa "Mathematica", donde fueron utilizados 136 datos para la ejecución del programa.

El ejercicio y la metodología del modelamiento se indican en Anexos "**MODELAMIENTO DENSIDAD EN MATHEMATICA**". Sin embargo, a continuación, se muestra el resultado gráfico y la explicación respectiva de este.

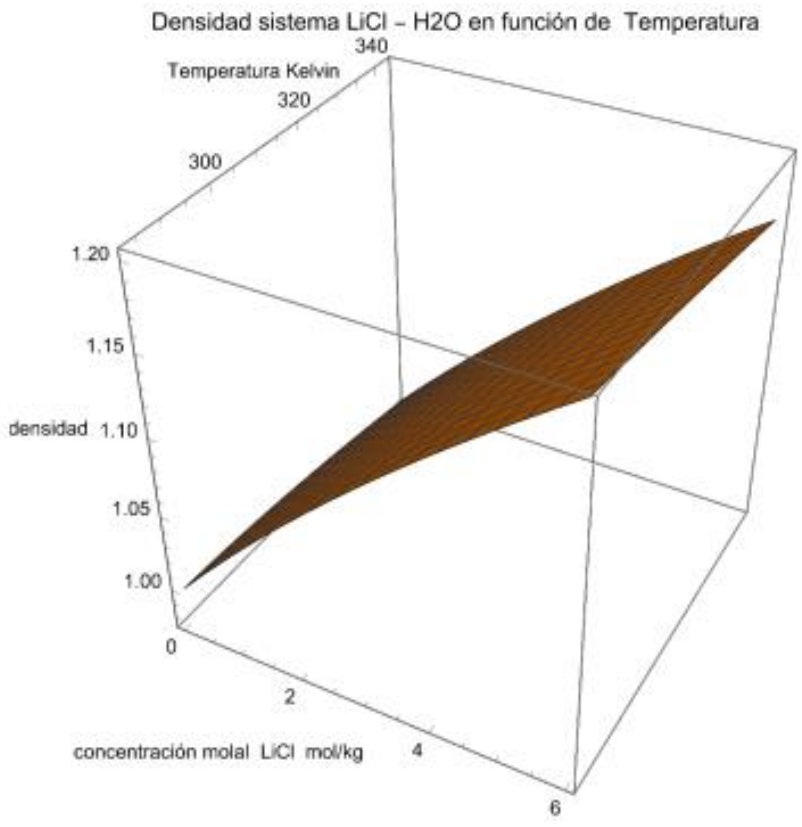


Ilustración 1: Gráfico de densidad para LiCl – H_2O en función de Temperatura

Este gráfico tridimensional muestra la densidad del sistema LiCl-H_2O en función de la temperatura y la concentración molal de LiCl. Se puede inferir que la densidad incrementa significativamente con el aumento de la concentración molal, como se observa en la pendiente ascendente de la superficie hacia la derecha. Aunque el efecto de la temperatura es menos pronunciado, también contribuye al incremento de la densidad, evidenciado por la inclinación de la superficie hacia arriba.

La combinación de ambas variables indica que la densidad se ve más afectada por la molalidad que por la temperatura, aunque esta última también influye ligeramente. Los valores más altos de densidad se encuentran en la esquina superior derecha del gráfico, donde tanto la concentración molal como la temperatura son elevadas. En conclusión, la densidad del sistema LiCl-H_2O está principalmente influenciada por la molalidad, con una influencia adicional de la temperatura.

4. Modelo empírico de cálculo de densidad utilizando Inteligencia Artificial con Google Colab.

A partir de la elaboración del código de programación para determinar la metodología de este modelamiento de densidad, el cual se evidencia en Anexos "**MODELAMIENTO DENSIDAD INTELIGENCIA ARTIFICIAL CON GOOGLE COLAB**", y considerando los 591 datos existentes, se concluye mediante el siguiente gráfico que:

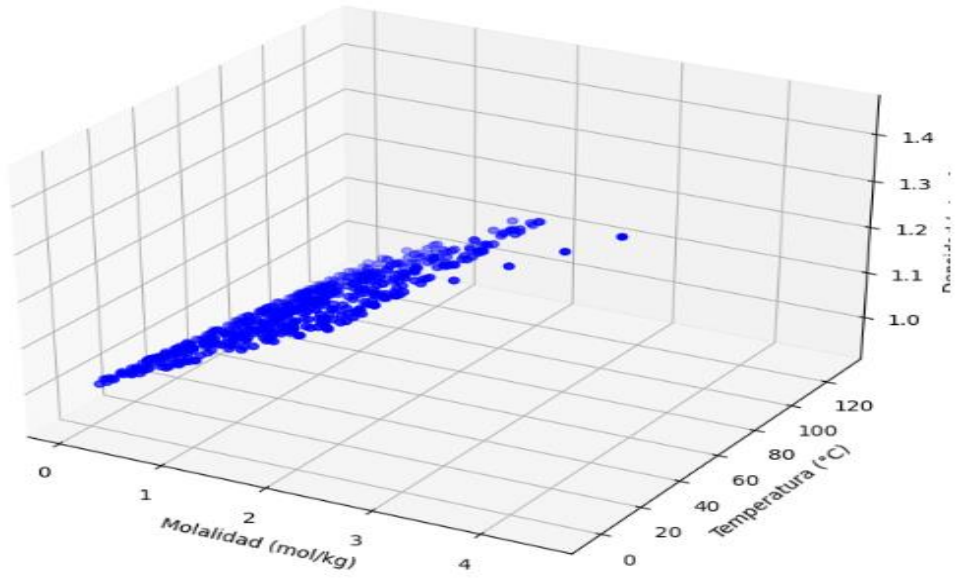


Ilustración 2: Relación entre Molalidad, Temperatura y Densidad.

El gráfico representa la distribución de datos con respecto a la densidad, temperatura y molalidad. Junto a esto, presenta un error porcentual de cálculo de un 4,67%.

Relación Molalidad-Densidad:

Existe una correlación directa: a medida que la molalidad se incrementa, la densidad también se incrementa. Esto se evidencia por la concentración de puntos de color más claro (amarillo) en el lado derecho del gráfico.

Relación Temperatura-Densidad:

La influencia de la temperatura en la densidad resulta ser menos significativa en contraste con la de la molalidad; sin embargo, se observan pequeñas variaciones. A temperaturas más bajas, la densidad tiende a ser ligeramente inferior en relación con las temperaturas más altas, aunque esta variación no es tan pronunciada como la relación entre molalidad y densidad.

Distribución General:

Los datos se disponen en una pendiente ascendente desde la esquina inferior izquierda (baja molalidad y temperatura) hacia la esquina superior derecha (alta molalidad y temperatura). La mayoría de los puntos se concentra en una franja que sugiere la existencia de una combinación óptima de molalidad y temperatura que maximiza la densidad.

Con esta demostración de estadígrafos del modelamiento de densidad, se concluye que la densidad del LiCl depende más intensamente de la molalidad que de la temperatura. Esto indica que al incrementar la concentración de soluto (LiCl) en una solución, la densidad de esta aumenta significativamente, aunque también puede haber un leve efecto térmico. Para aplicaciones que requieran una densidad específica, sería más eficiente ajustar la molalidad en lugar de la temperatura.

Conclusiones

Se evaluaron distintos modelos para la estimación de la densidad de soluciones acuosas de Cloruro de Litio (LiCl). Cada modelo tiene sus propias ventajas y desventajas, pero se determina que el mejor modelo es el que presenta el menor error en sus estimaciones de densidad.

Mejor Método: Modelo Empírico con Inteligencia Artificial usando Google Colab (Python).

Razón de la Elección:

- **Precisión:** Este modelo presenta el error más bajo en comparación con los otros modelos analizados. Utiliza un enfoque empírico que ajusta los datos de molalidad, densidad y temperatura mediante coeficientes específicos determinados por inteligencia artificial.
- **Adaptabilidad:** Este modelo permite la consideración de múltiples variables (molalidad y temperatura) y se ajusta a datos experimentales específicos, lo que mejora su precisión en escenarios prácticos.
- **Innovación:** La integración con Google Colab y el uso de inteligencia artificial proporciona una metodología moderna y eficiente para manejar grandes conjuntos de datos y realizar cálculos complejos.

El modelo predictivo semi empírico y el modelo basado en volúmenes molares aparentes también son eficaces, pero no alcanzan el nivel de precisión del modelo empírico con inteligencia artificial. En resumen, para la estimación precisa de la densidad de LiCl en solución acuosa, se recomienda el uso del modelo empírico con inteligencia artificial debido a su menor margen de error y su capacidad para manejar variaciones en molalidad y temperatura de manera efectiva.

Cabe mencionar que el modelo analizado y escogido puede variar en cuanto al cálculo del error con respecto al modelo en Mathematica, ya que su error no pudo ser identificado debido a variables de cálculo.

Discusiones

Analizar en profundidad los distintos métodos para cálculos de densidades, con esto poder generar varios cálculos de determinación de errores con distintos iones, con el fin de obtener un resultado más específico y concluyente de cuál es el mejor método de cálculo para las densidades de un ion.

En el caso de ser 2 o más iones, se debe proceder al cálculo de estas densidades con el método previamente seleccionado y así poder tener claridad de los datos con los cuales se estima la densidad, a través de los métodos pertinentes de cálculo para estos.

Bibliografía

Arshad, M., Easa, A., Qiblawey, H., Nasser, M., Benamor, A., Bhosale, R., & Al-Ghouti, M. (2020). *Experimental measurements and modelling of viscosity and density of calcium and potassium chlorides ternary solutions.* Scientific Reports, 10(1).

Kumar, A. (1986). *Densities and apparent molal volumes of aqueous potassium chloride-calcium chloride mixtures at 298.15 K.* Journal of Chemical and Engineering Data/Journal of Chemical & Engineering Data, 31(1), 21-23.

Kumar, A. (1986). *Prediction of Densities of Concentrated Brines by Pitzer Theory.* Journal of Chemical and Engineering Data/Journal of Chemical & Engineering Data, 31(1), 19-20.

Krumgalz, B. S., Pogorelsky, R., Pitzer, K. S. (1995). *Ion interaction approach to calculations of volumetric properties of aqueous multiple-solute electrolyte solutions.* Journal of Solution Chemistry, 24(10), 1025-1038.

Laliberté, M., Cooper, W. E. (2004). *Model for Calculating the Density of Aqueous Electrolyte Solutions.* Journal Of Chemical and Engineering Data/Journal of Chemical & Engineering Data, 49(5), 1141-1151.

Sharp, J. D., Albehadili, M. H. M., Millero, F. J., & Woosley, R. J. (2015). *Estimating the Density and Compressibility of Natural Hypersaline Brines Using the Pitzer Ionic Interaction Model.* Aquatic Geochemistry, 21(1), 11-29.

Soto-Bubert, A., Miranda, J., Bhardwaj, R., Rajendran, S., Acevedo, R. (2024). *Density modelling of chlorinated brines in two and three-component systems.*

Zhang, J., Clennell, M. B., Dewhurst, D. N. (2023). *Transport Properties of NaCl in Aqueous Solution and Hydrogen Solubility in Brine*. The Journal of Physical Chemistry. B, 127(41), 8900-8915.

Anexos

MODELAMIENTO DENSIDAD EN MATHEMATICA:

```
ListaLiCl :=
 {{{17.8`, 252.59`, 1288.4`}, {14.`, 252.59`, 1240.7`}, {11.`, 252.59`, 1200.2`},
   {8.`, 252.59`, 1155.4`}, {6.`, 252.59`, 1121.7`}, {4.5`, 252.59`, 1095.9`},
   {3.`, 252.59`, 1067.8`}, {2.`, 252.59`, 1047.1`}, {1.`, 252.59`, 1024.4`},
   {0.5`, 252.59`, 1013.4`}, {17.8`, 258.15`, 1284.2`}, {14.`, 258.15`, 1237.4`},
   {11.`, 258.15`, 1197.1`}, {8.`, 258.15`, 1152.6`}, {6.`, 258.15`, 1119.8`},
   {4.5`, 258.15`, 1094.1`}, {3.`, 258.15`, 1066.4`}, {2.`, 258.15`, 1046.3`},
   {1.2128`, 258.15`, 1029.18`}, {1.`, 258.15`, 1024.`}, {0.721`, 258.15`, 1017.81`},
   {0.5`, 258.15`, 1013.7`}, {0.4786`, 258.15`, 1011.99`}, {0.4786`, 258.15`, 1011.99`},
   {0.3581`, 258.15`, 1009.05`}, {0.2383`, 258.15`, 1006.07`},
   {0.1189`, 258.15`, 1003.06`}, {17.8`, 260.93`, 1282.2`}, {14.`, 260.93`, 1235.8`},
   {11.`, 260.93`, 1195.6`}, {8.`, 260.93`, 1151.4`}, {6.`, 260.93`, 1118.8`},
   {4.5`, 260.93`, 1092.8`}, {3.`, 260.93`, 1065.4`}, {2.`, 260.93`, 1045.3`},
   {1.`, 260.93`, 1023.3`}, {0.5`, 260.93`, 1012.5`}, {19.584`, 263.71`, 1296.8`},
   {18.729`, 263.71`, 1287.4`}, {17.906`, 263.71`, 1278.`}, {17.8`, 263.71`, 1280.2`},
   {17.249`, 263.71`, 1271.3`}, {15.827`, 263.71`, 1254.8`}, {15.063`, 263.71`, 1246.3`},
   {14.067`, 263.71`, 1233.6`}, {14.`, 263.71`, 1234.1`}, {12.923`, 263.71`, 1219.4`},
   {11.963`, 263.71`, 1206.`}, {11.`, 263.71`, 1194.`}, {10.913`, 263.71`, 1191.9`},
   {10.`, 263.71`, 1178.88`}, {9.863`, 263.71`, 1177.1`}, {8.8268`, 263.71`, 1162.1`},
   {8.`, 263.71`, 1149.08`}, {8.`, 263.71`, 1149.8`}, {7.2417`, 263.71`, 1137.8`},
   {6.`, 263.71`, 1117.96`}, {6.`, 263.71`, 1117.4`}, {5.7128`, 263.71`, 1113.5`},
   {4.5`, 263.71`, 1091.5`}, {4.1814`, 263.71`, 1086.2`}, {4.`, 263.71`, 1082.84`},
   {3.`, 263.71`, 1064.1`}, {3.`, 263.71`, 1064.3`}, {2.5925`, 263.71`, 1056.6`},
   {2.`, 263.71`, 1044.09`}, {2.`, 263.71`, 1044.1`}, {1.2128`, 263.71`, 1027.7`},
   {1.009`, 263.71`, 1023.2`}, {1.`, 263.71`, 1022.47`}, {1.`, 263.71`, 1022.2`},
   {0.721`, 263.71`, 1016.53`}, {0.5`, 263.71`, 1011.21`}, {0.5`, 263.71`, 1011.21`},
   {0.5`, 263.71`, 1011.6`}, {0.4786`, 263.71`, 1010.85`}, {0.4786`, 263.71`, 1010.85`},
   {0.3581`, 263.71`, 1007.97`}, {0.2383`, 263.71`, 1005.06`},
   {0.1189`, 263.71`, 1002.12`}, {0.1`, 263.71`, 1001.54`}, {0.05`, 263.71`, 1000.34`},
   {19.584`, 269.26`, 1293.1`}, {18.729`, 269.26`, 1283.7`}, {17.906`, 269.26`, 1274.3`},
   {17.8`, 269.26`, 1276.4`}, {17.249`, 269.26`, 1267.6`}, {15.827`, 269.26`, 1251.1`},
   {15.063`, 269.26`, 1242.7`}, {14.067`, 269.26`, 1230.1`}, {14.`, 269.26`, 1230.7`},
   {12.923`, 269.26`, 1216.1`}, {11.963`, 269.26`, 1203.1`}, {11.`, 269.26`, 1190.9`},
   {10.913`, 269.26`, 1188.8`}, {10.`, 269.26`, 1175.89`}, {9.863`, 269.26`, 1174.2`},
   {8.8268`, 269.26`, 1159.1`}, {8.`, 269.26`, 1146.26`}, {8.`, 269.26`, 1147.`},
   {7.2417`, 269.26`, 1134.8`}, {6.`, 269.26`, 1114.82`}, {6.`, 269.26`, 1114.6`},
   {5.7128`, 269.26`, 1110.4`}, {4.5`, 269.26`, 1089.1`}, {4.1814`, 269.26`, 1083.4`},
   {4.`, 269.26`, 1080.18`}, {3.`, 269.26`, 1061.53`}, {3.`, 269.26`, 1061.8`},
   {2.5925`, 269.26`, 1053.9`}, {2.`, 269.26`, 1041.63`}, {2.`, 269.26`, 1041.8`},
   {1.2128`, 269.26`, 1025.28`}, {1.08789`, 269.26`, 1022.51`}, {1.009`, 269.26`, 1020.2`},
   {1.`, 269.26`, 1020.08`}, {1.`, 269.26`, 1020.2`}, {0.97888`, 269.26`, 1020.07`},
   {0.87037`, 269.26`, 1017.63`}, {0.76073`, 269.26`, 1015.14`},
   {0.721`, 269.26`, 1014.24`}, {0.65179`, 269.26`, 1012.65`}, {0.5`, 269.26`, 1009.02`},
   {0.5`, 269.26`, 1009.02`}, {0.5`, 269.26`, 1009.4`}, {0.4786`, 269.26`, 1008.63`},
   {0.4786`, 269.26`, 1008.63`}, {0.43406`, 269.26`, 1007.58`},
   {0.3581`, 269.26`, 1005.79`}, {0.32518`, 269.26`, 1005.`}, {0.2383`, 269.26`, 1002.92`},
```

```
    {0.21689`, 269.26`, 1002.4`}, {0.1189`, 269.26`, 1000.02`},
    {0.10782`, 269.26`, 999.74`}, {0.1`, 269.26`, 999.47`}, {0.05`, 269.26`, 998.28`}}}

ρna252[x_] := 0.9997661550148879` + 0.04268531841464265` x - 0.0021946301186045575` x^2 +
   0.0001964234592812042` x^3 - 0.00001990875208820208 3` x^4 + 1.0913602952955867`*^-6 x^5;

ρna258[x_] := 0.9970749836264344` + 0.040982535728326136` x - 0.0020206752181188507` x^2 +
   0.0002058272252960082 8` x^3 - 0.00002395442830959672 4` x^4 + 1.348570107252118`*^-6 x^5;

ρna260[x_] := 0.9922381071390132` + 0.039906790170254454` x - 0.0018728478198010 48` x^2 +
   0.0001919013446565113 3` x^3 - 0.00002323545560429348 3` x^4 + 1.3195884123827944`*^-6 x^5;

ρna263[x_] := 0.9857231100513258` + 0.0392688662918923` x - 0.0017549716938444846` x^2 +
   0.0001643719678504666 4` x^3 - 0.00001893857181278022` x^4 + 1.041246104122895`*^-6 x^5;

ρna269[x_] := 0.9777958106494394` + 0.039029892447205845` x - 0.0017533808260002755` x^2 +
   0.0001696854480107564 6` x^3 - 0.00001991651617210956 5` x^4 + 1.0971106219106938`*^-6 x^5;

A1 := {{252.59, 0.9997661550148879`},
   {258.15, 0.9970749836264344}, {260.93, 0.9922381071390132},
   {263.71, 0.9857231100513258}, {269.26, 0.9777958106494394}};
ρA1[t_] := -1.4030240351098493` + 0.027655164155043956` t -
   0.0001184115987205896 6` t^2 + 2.261188818876219`*^-7 t^3 - 1.660627264510487`*^-10 t^4;
Fit[A1, {1, t, t^2, t^3, t^4}, t]
ajusta

Table[{A1[[i, 1]], A1[[i, 2]], ρA1[A1[[i, 1]]]}, {i, 1, Length[A1]}] // MatrixForm
tabla                                                longitud    forma de matriz
```

Out[28]=

$$4201.96 - 65.2522\,t + 0.379922\,t^2 - 0.000982707\,t^3 + 9.52776\times 10^{-7}\,t^4$$

Out[29]//MatrixForm=

252.59	0.999766	0.995605
258.15	0.997075	0.997574
260.93	0.992238	0.99833
263.71	0.985723	0.998943
269.26	0.977796	0.999766

```
A1 := {{252.59, 0.04268531841464265`},
   {258.15, 0.040982535728326136`}, {260.93, 0.039906790170254454`},
   {263.71, 0.0392688662918923`}, {269.26, 0.039029892447205845` }};
ρA2[t_] := 1.512942810648095` - 0.017503976659143067` t +
   0.0000790415635639981` t^2 - 1.6062819741202393`*^-7 t^3 + 1.237397551486106`*^-10 t^4;
Fit[A2, {1, t, t^2, t^3, t^4}, t]
ajusta
Table[{A2[[i, 1]], A2[[i, 2]], ρA2[A2[[i, 1]]]}, {i, 1, Length[A2]}] // MatrixForm
tabla                                                longitud      forma de matriz
```

Out[32]=

$$-1122.56 + 17.1279\,t - 0.0979615\,t^2 + 0.000248925\,t^3 - 2.37118\times 10^{-7}\,t^4$$

In[38]:=

$$\begin{pmatrix} 252.59` & 0.04268531841464265` & 0.04967075523092124` \\ 258.15` & 0.040982535728326136` & 0.04790702195930496` \\ 260.93` & 0.039906790170254454` & 0.0471267503578636` \\ 263.71` & 0.0392688662918923` & 0.04639826660387225` \\ 269.26` & 0.039029892447205845` & 0.04511415782513151` \end{pmatrix}$$

```
A3 := {{252.59, -0.0021946301186045575` },
   {258.15, -0.0020206752181188507}, {260.93, -0.001872847819801048},
   {263.71, -0.0017549716938444846}, {269.26, -0.0017533808260002755}};
ρA3[t_] := -0.5945132174592646` + 0.007728161766392484` t -
   0.00003791646071249194` t^2 + 8.283750068476853`*^-8 t^3 - 6.790964243611025`*^-11 t^4;

Fit[A3, {1, t, t^2, t^3, t^4}, t]
ajusta
Table[{A3[[i, 1]], A3[[i, 2]], ρA3[A3[[i, 1]]]}, {i, 1, Length[A3]}] // MatrixForm
tabla                                                longitud      forma de matriz
```

Out[38]=

```
{{252.59, 0.0426853, 0.0496708},
 {258.15, 0.0409825, 0.047907}, {260.93, 0.0399068, 0.0471268},
 {263.71, 0.0392689, 0.0463983}, {269.26, 0.0390299, 0.0451142}}
```

Out[41]=

$$42.0298 - 0.622013\,t + 0.00344483\,t^2 - 8.4611\times 10^{-6}\,t^3 + 7.77572\times 10^{-9}\,t^4$$

In[43]:=
$$\begin{pmatrix} 252.59` & -0.0021946301186045575` & -0.003046611388513476` \\ 258.15` & -0.0020206752181188507` & -0.0027936826434218354` \\ 260.93` & -0.001872847819801048` & -0.002689324171463614` \\ 263.71` & -0.0017549716938444846` & -0.00259755013740115` \\ 269.26` & -0.0017533808260002755` & -0.0024461205255470464` \end{pmatrix}$$

```
A4 := {{252.59, 0.0001964234592812042`},
   {258.15, 0.00020582722529600828`}, {260.93, 0.00019190134465651133`},
   {263.71, 0.00016437196785046664`}, {269.26, 0.00016968544801075646` }};

ρA4[t_] := 0.24512794639909227` - 0.0032827947693236576` t +
   0.00001644215068029441 3` t^2 - 3.646958907476246`*^-8 t^3 + 3.022238900909521`*^-11 t^4;

Fit[A4, {1, t, t^2, t^3, t^4}, t]
ajusta

Table[{A4[[i, 1]], A4[[i, 2]], ρA4[A4[[i, 1]]]}, {i, 1, Length[A4]}] // MatrixForm
tabla                                                longitud   forma de matriz
```

Out[43]=

```
{{252.59, -0.00219463, -0.00304661},
 {258.15, -0.00202068, -0.00279368}, {260.93, -0.00187285, -0.00268932},
 {263.71, -0.00175497, -0.00259755}, {269.26, -0.00175338, -0.00244612}}
```

Out[46]=

$$54.8265 - 0.847421\,t + 0.00491059\,t^2 - 0.0000126438\,t^3 + 1.2205 \times 10^{-8}\,t^4$$

In[48]:=
$$\begin{pmatrix} 252.59` & 0.0001964234592812042` & 0.00025713379230091205` \\ 258.15` & 0.00020582722529600828` & 0.0002188463493547066` \\ 260.93` & 0.00019190134465651133` & 0.00020661145404166081` \\ 263.71` & 0.00016437196785046664` & 0.00019809246188084995` \\ 269.26` & 0.00016968544801075646` & 0.00018979865865631496` \end{pmatrix}$$

```
A5 := {{252.59, -0.00001990875208820 83`},
   {258.15, -0.00002395442830959 6724}, {260.93, -0.00002323545560429 3483`},
   {263.71, -0.00001893857181278022`}, {269.26, -0.00001991651617210 9565`}};

ρA5[t_] := -0.05223585238168377` + 0.0006964427878345262` t -
   3.470098924009328`*^-6 t^2 + 7.655349001856084`*^-9 t^3 - 6.309466170170971`*^-12 t^4;

Fit[A5, {1, t, t^2, t^3, t^4}, t]
ajusta

Table[{A5[[i, 1]], A5[[i, 2]], ρA5[A5[[i, 1]]]}, {i, 1, Length[A5]}] // MatrixForm
tabla                                                longitud   forma de matriz
```

Out[48]=

```
{{252.59, 0.000196423, 0.000257134},
 {258.15, 0.000205827, 0.000218846}, {260.93, 0.000191901, 0.000206611},
 {263.71, 0.000164372, 0.000198092}, {269.26, 0.000169685, 0.000189799}}
```

Out[51]=

$$-13.5214 + 0.208407\,t - 0.0012043\,t^2 + 3.09223 \times 10^{-6}\,t^3 - 2.97673 \times 10^{-9}\,t^4$$

In[53]:=
$$\begin{pmatrix} 252.59` & -0.000019908752088202083` & -0.0000321812451807707` \\ 258.15` & -0.000023954428309596724` & -0.000023646055837017732` \\ 260.93` & -0.000023235455604293483` & -0.000020944869982164604` \\ 263.71` & -0.00001893857181278022` & -0.000019094995347565535` \\ 269.26` & -0.000019916516172109565` & -0.000017425576051693925` \end{pmatrix}$$

```
A6 := {{252.59, 1.0913602952955867`*^-6},
   {258.15, 1.348570107252118`*^-6}, {260.93, 1.3195884123827944`*^-6 },
   {263.71, 1.041246104122895`*^-6 }, {269.26, 1.0971106219106938`*^-6 }};

ρA6[t_] := 0.003798177233327877` - 0.000050370468302711924` t +
   2.497170497501202`*^-7 t^2 - 5.483376523083284`*^-10 t^3 + 4.499889267627936`*^-13 t^4;

Fit[A6, {1, t, t^2, t^3, t^4}, t]
[ajusta

Table[{A6[[i, 1]], A6[[i, 2]], ρA6[A6[[i, 1]]]}, {i, 1, Length[A6]}] // MatrixForm
[tabla                                          [longitud          [forma de matriz
```

Out[53]=
$$\{\{252.59, -0.0000199088, -0.0000321812\},\ \{258.15, -0.0000239544, -0.0000236461\}, \{260.93, -0.0000232355, -0.0000209449\},\ \{263.71, -0.0000189386, -0.000019095\}, \{269.26, -0.0000199165, -0.0000174256\}\}$$

Out[56]=
$$0.951107 - 0.0146508\,t + 0.0000846108\,t^2 - 2.17126\times10^{-7}\,t^3 + 2.08895\times10^{-10}\,t^4$$

In[64]:=
$$\begin{pmatrix} 252.59` & 1.0913602952955867`*^-6 & 2.3954211691929253`*^-6 \\ 258.15` & 1.348570107252118`*^-6 & 1.6591189918226806`*^-6 \\ 260.93` & 1.3195884123827944`*^-6 & 1.4121668898987166`*^-6 \\ 263.71` & 1.041246104122895`*^-6 & 1.2318463993195305`*^-6 \\ 269.26` & 1.0971106219106938`*^-6 & 1.0326653125548174`*^-6 \end{pmatrix}$$

```
ρLiCl[x_, t_] := ρA1[t] + x * ρA2[t] + x^2 * ρA3[t] + x^3 * ρA4[t] + x^4 * ρA5[t] + x^5 * ρA6[t];
ρLiCl[x, t]
```

Out[64]=
$$\{\{252.59, 1.09136\times10^{-6}, 2.39542\times10^{-6}\},\ \{258.15, 1.34857\times10^{-6}, 1.65912\times10^{-6}\}, \{260.93, 1.31959\times10^{-6}, 1.41217\times10^{-6}\},\ \{263.71, 1.04125\times10^{-6}, 1.23185\times10^{-6}\}, \{269.26, 1.09711\times10^{-6}, 1.03267\times10^{-6}\}\}$$

In[67]:=
```
-1.4030240351098493` + 0.027655164155043956` t - 0.0001184115987205896` t^2 +
 2.261188818876219` *^-7 t^3 - 1.660627264510487` *^-10 t^4 +
 (1.512942810648095` - 0.017503976659143067` t + 0.0000790415635639981` t^2 -
    1.6062819741202393` *^-7 t^3 + 1.237397551486106` *^-10 t^4) x +
 (-0.5945132174592646` + 0.007728161766392484` t - 0.00003791646071249194` t^2 +
    8.283750068476853` *^-8 t^3 - 6.790964243611025` *^-11 t^4) x^2 +
 (0.24512794639909227` - 0.0032827947693236576` t + 0.00001644215068029441 3` t^2 -
    3.646958907476246` *^-8 t^3 + 3.022238900909521` *^-11 t^4) x^3 +
 (-0.05223585238168377` + 0.0006964427878345262` t - 3.470098924009328` *^-6 t^2 +
    7.655349001856084` *^-9 t^3 - 6.309466170170971` *^-12 t^4) x^4 +
 (0.003798177233327877` - 0.00005037046830271192 4` t + 2.497170497501202` *^-7 t^2 -
    5.483376523083284` *^-10 t^3 + 4.499889267627936` *^-13 t^4) x^5

aa := Table[{ListaLiCl[[i, 1]], ListaLiCl[[i, 2]],
    tabla
   ListaLiCl[[i, 3]], ρLiCl[ListaLiCl[[i, 1]], ListaLiCl[[i, 2]]],
   √((ListaLiCl[[i, 3]] - ρLiCl[ListaLiCl[[i, 1]], ListaLiCl[[i, 2]]])^2)}, {i,
   1, Length[ListaLiCl]}];
     longitud

aa // MatrixForm
     forma de matriz
```

Out[67]=

$$-1.40302 + 0.0276552\,t - 0.000118412\,t^2 + 2.26119\times10^{-7}\,t^3 - 1.66063\times10^{-10}\,t^4 + \left(1.51294 - 0.017504\,t + 0.0000790416\,t^2 - 1.60628\times10^{-7}\,t^3 + 1.2374\times10^{-10}\,t^4\right)x + \left(-0.594513 + 0.00772816\,t - 0.0000379165\,t^2 + 8.28375\times10^{-8}\,t^3 - 6.79096\times10^{-11}\,t^4\right)x^2 + \left(0.245128 - 0.00328279\,t + 0.0000164422\,t^2 - 3.64696\times10^{-8}\,t^3 + 3.02224\times10^{-11}\,t^4\right)x^3 + \left(-0.0522359 + 0.000696443\,t - 3.4701\times10^{-6}\,t^2 + 7.65535\times10^{-9}\,t^3 - 6.30947\times10^{-12}\,t^4\right)x^4 + \left(0.00379818 - 0.0000503705\,t + 2.49717\times10^{-7}\,t^2 - 5.48338\times10^{-10}\,t^3 + 4.49989\times10^{-13}\,t^4\right)x^5$$

Out[69]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 17.8 \\ 252.59 \\ 1288.4 \end{pmatrix} & \begin{pmatrix} 14. \\ 252.59 \\ 1240.7 \end{pmatrix} & \begin{pmatrix} 11. \\ 252.59 \\ 1200.2 \end{pmatrix} & \begin{pmatrix} 2286.75 \\ 2.33595\times10^{6} \\ 1.21053\times10^{15} \end{pmatrix} & \begin{pmatrix} 2275.75 \\ 2.3357\times10^{6} \\ 1.21053\times10^{15} \end{pmatrix} \end{pmatrix}$$

In[70]:=
```
(Sum[(√((aa[[i, 3]] - aa[[i, 4]])^2)/1) * (1/aa[[i, 3]]), {i, 1, Length[aa]}]) * 1/Length[aa]
```

In[71]:=
```
{206.88609806397545`, 9247.007446904543`, 1.0086030303446519`*^12}
Plot3D[ρLiCl[x, t], {x, 0, 6}, {t, 283, 343},
 representación gráfica 3D
 AxesLabel → {"concentración molal  LiCl  mol/kg", "Temperatura Kelvin", "densidad"},
 etiqueta de ejes
 BoxRatios → {5, 5, 5},
 cocientes de caja
 PlotLabel → "Densidad sistema LiCl – H2O en función de  Temperatura"]
 etiqueta de representación
```

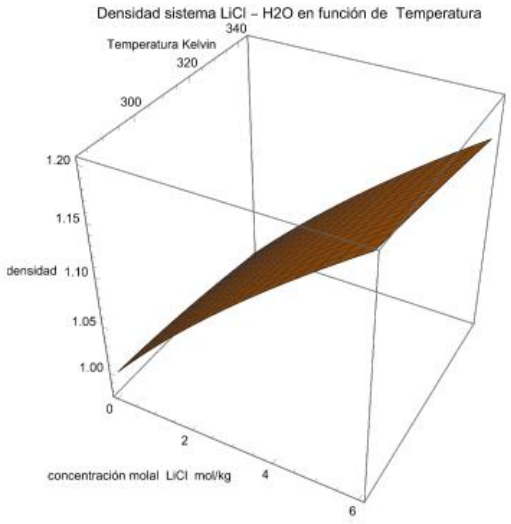
Out[71]=

$$\{206.886,\ 9247.01,\ 1.0086 \times 10^{12}\}$$

In[73]:=


```
bb :=
  Table[{ListaLiCl[[i, 1]], ListaLiCl[[i, 2]], ListaLiCl[[i, 3]]}, {i, 1, Length[ListaLiCl]}];
  tabla                                                            longitud
bb // MatrixForm
      forma de matriz
ListPointPlot3D[bb]
representación 3D en puntos de lista
```

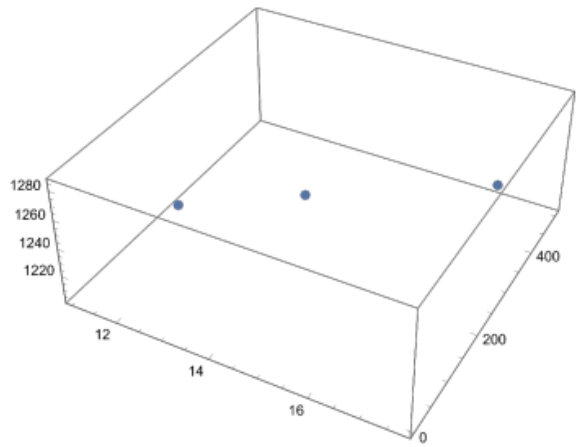
Out[73]=



Out[75]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 17.8 \\ 252.59 \\ 1288.4 \end{pmatrix} & \begin{pmatrix} 14. \\ 252.59 \\ 1240.7 \end{pmatrix} & \begin{pmatrix} 11. \\ 252.59 \\ 1200.2 \end{pmatrix} \end{pmatrix}$$

Out[76]=



MODELAMIENTO DENSIDAD INTELIGENCIA ARTIFICIAL CON GOOGLE COLAB:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Definir la función para calcular la densidad
def calcular_densidad(molalidad, temperatura):
    rho_0 = 0.997  # Densidad del agua pura a 25°C en g/cm^3
    a = 0.1       # Coeficiente para la molalidad
    b = -0.0005   # Coeficiente para la temperatura
    c = 0.0001    # Coeficiente para el término cruzado molalidad-temperatura

    densidad = rho_0 + a * molalidad + b * temperatura + c * molalidad * temperatura
    return densidad

# Datos de temperatura y molalidad proporcionados
temperaturas = [
    5, 5, 5, 5, 10, 10, 10, 10, 15, 15, 15, 15, 20, 20, 20, 20,
    24.96, 24.98, 24.99, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
    25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
    25.01, 25.01, 25.01, 25.01, 25.01, 25.01, 25.01, 25.02, 25.02, 25.02,
    25.03, 30, 30, 30, 30, 30, 30, 30, 35, 35, 35, 35, 40, 40, 40, 40,
    40, 40, 40, 45, 45, 45, 45, 50, 50, 50, 50, 50, 50, 50, 55, 55, 55,
    55, 60, 60, 60, 60, 60, 60, 60, 65, 65, 65, 65, 70, 70, 70, 70, 70,
    70, 70, 75, 75, 75, 75, 80, 80, 80, 80, 80, 80, 80, 85, 85, 85, 85,
    90, 90, 90, 90, 90, 90, 90, 95, 95, 95, 95, 100, 100, 100, 100, 100,
    100, 100, 105, 105, 105, 105, 110, 110, 110, 110, 110, 110, 110, 115,
    115, 115, 115, 120, 120, 120, 120, 120, 120, 120, 130, 130, 130
```

```
temperaturas = [
    5, 5, 5, 5, 10, 10, 10, 10, 15, 15, 15, 15, 20, 20, 20, 20,
    24.96, 24.98, 24.99, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
    25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
    25.01, 25.01, 25.01, 25.01, 25.01, 25.01, 25.01, 25.02, 25.02, 25.02,
    25.03, 30, 30, 30, 30, 30, 30, 30, 35, 35, 35, 35, 40, 40, 40, 40,
    40, 40, 40, 45, 45, 45, 45, 50, 50, 50, 50, 50, 50, 50, 55, 55, 55,
    55, 60, 60, 60, 60, 60, 60, 60, 65, 65, 65, 65, 70, 70, 70, 70, 70,
    70, 70, 75, 75, 75, 75, 80, 80, 80, 80, 80, 80, 80, 85, 85, 85, 85,
    90, 90, 90, 90, 90, 90, 90, 95, 95, 95, 95, 100, 100, 100, 100, 100,
    100, 100, 105, 105, 105, 105, 110, 110, 110, 110, 110, 110, 110, 115,
    115, 115, 115, 120, 120, 120, 120, 120, 120, 120, 130, 130, 130
]

molalidades = [
    0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498,
    0.4993, 1.0036, 0.0999, 0.2498, 0.4993, 1.0036, 0.0400, 0.4904, 0.2504, 0.0453,
    0.0616, 0.0625, 0.0791, 0.0999, 0.1004, 0.1504, 0.2102, 0.2232, 0.2498, 0.3038,
    0.4772, 0.4993, 0.5000, 0.7173, 0.9604, 1.0000, 1.0000, 1.0000, 1.0036, 1.5000,
    2.0000, 2.5000, 3.0000, 3.5000, 4.0000, 4.5, 0.0625, 0.0897, 0.0897, 0.1600,
    0.1600, 0.2504, 2.4614, 0.0400, 0.4904, 1.0000, 2.4614, 0.0999, 0.2498, 0.4970,
    0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498,
    0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999,
    0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036,
    0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993,
    1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498,
    0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999,
    0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550,
    0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036,
    1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690,
    1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993,
```

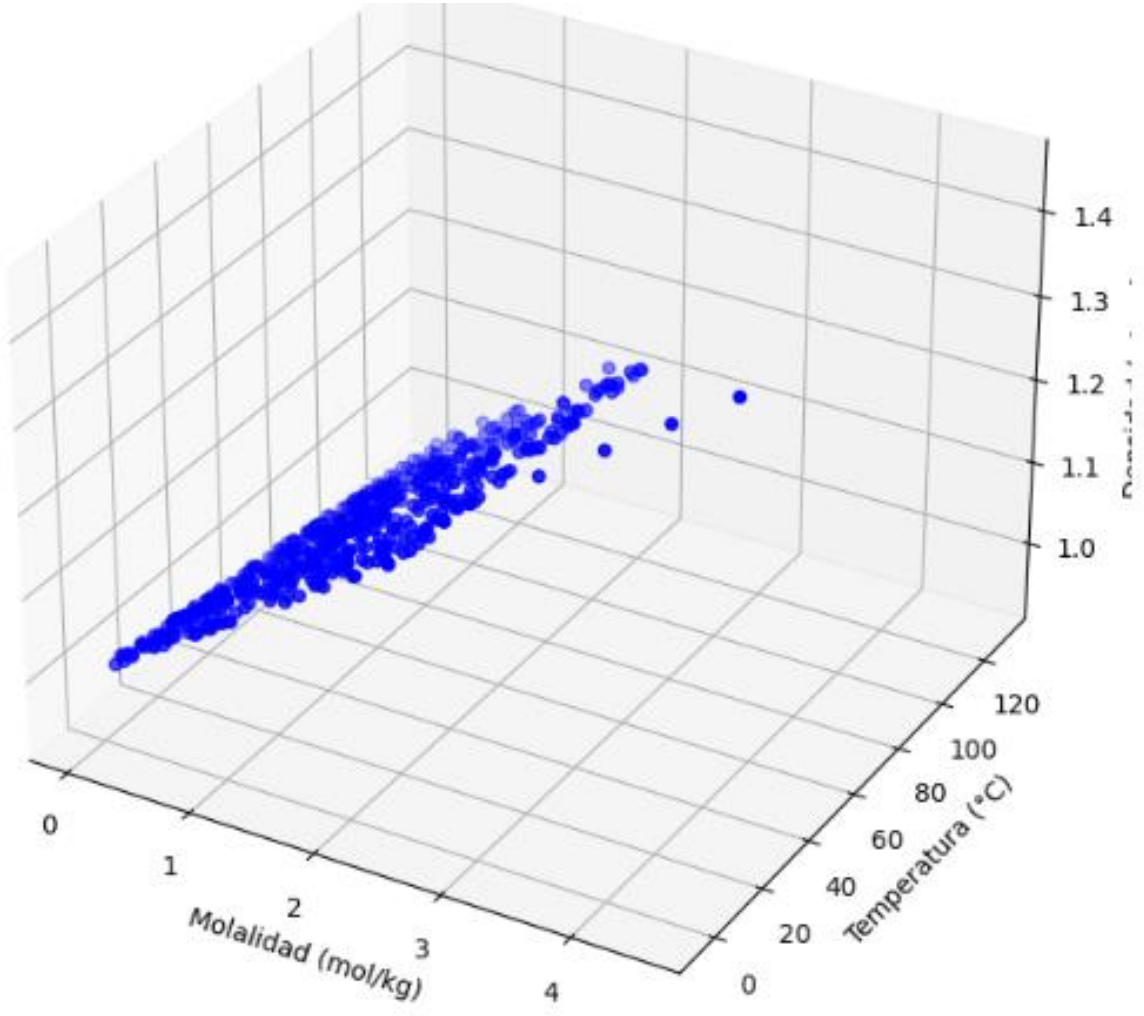
```
molalidades = [
    0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498,
    0.4993, 1.0036, 0.0999, 0.2498, 0.4993, 1.0036, 0.0400, 0.4904, 0.2504, 0.0453,
    0.0616, 0.0625, 0.0791, 0.0999, 0.1004, 0.1504, 0.2102, 0.2232, 0.2498, 0.3038,
    0.4772, 0.4993, 0.5000, 0.7173, 0.9604, 1.0000, 1.0000, 1.0000, 1.0036, 1.5000,
    2.0000, 2.5000, 3.0000, 3.5000, 4.0000, 4.5, 0.0625, 0.0897, 0.0897, 0.1600,
    0.1600, 0.2504, 2.4614, 0.0400, 0.4904, 1.0000, 2.4614, 0.0999, 0.2498, 0.4970,
    0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498,
    0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999,
    0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036,
    0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993,
    1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498,
    0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999,
    0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550,
    0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036,
    1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690,
    1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993,
    0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970,
    0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498,
    0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.4970, 0.9690, 1.7550
]

# Generar 400 datos adicionales de temperatura y molalidad
np.random.seed(0)  # Para reproducibilidad
n_extra = 400
temperaturas_extra = np.random.uniform(-5, 100, n_extra)
molalidades_extra = np.random.uniform(0.01, 2.5, n_extra)  # Molalidad en un rango razonable

# Combinar los datos proporcionados con los datos generados
temperaturas_combined = np.concatenate((temperaturas, temperaturas_extra))
molalidades_combined = np.concatenate((molalidades, molalidades_extra))
```

```python
# Ajustar las listas combinadas para que tengan la misma longitud
min_length = min(len(temperaturas_combined), len(molalidades_combined))
temperaturas_combined = temperaturas_combined[:min_length]
molalidades_combined = molalidades_combined[:min_length]

# Calcular densidades para todos los datos
densidades = [calcular_densidad(m, t) for m, t in zip(molalidades_combined, temperaturas_combined)]

# Crear un DataFrame con los resultados
df = pd.DataFrame({
    'Temperatura (°C)': temperaturas_combined,
    'Molalidad (mol/kg)': molalidades_combined,
    'Densidad (g/cm^3)': densidades
})

# Imprimir la tabla de valores
print(df)

# Crear la gráfica 3D
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')

# Crear la gráfica de dispersión
ax.scatter(df['Molalidad (mol/kg)'], df['Temperatura (°C)'], df['Densidad (g/cm^3)'], c='b', marker='o')

# Etiquetas de los ejes
ax.set_xlabel('Molalidad (mol/kg)')
ax.set_ylabel('Temperatura (°C)')
ax.set_zlabel('Densidad (g/cm^3)')

# Mostrar la gráfica
plt.show()
```

Tabla de Datos obtenida por medio de la integración de datos, utilizando la Inteligencia Artificial y un código de programación para que estos sean impresos de manera automática en conjunto con su gráfica.

	Temperatura (°C)	Molalidad (mol/kg)	Densidad (g/cm^3)
0	5.000000	0.099900	1.004540
1	5.000000	0.249800	1.019605
2	5.000000	0.499300	1.044680
3	5.000000	1.003600	1.095362
4	10.000000	0.099900	1.002090
..	...	...	...
561	95.693186	0.423459	0.995552
562	32.313729	2.307664	1.219066
563	32.454223	0.742251	1.057407
564	-3.285507	1.138205	1.112089
565	14.449394	1.239955	1.115562

[566 rows x 3 columns]



```
temperature1 = np.array([5, 5, 5, 5, 10, 10, 10, 10, 15, 15, 15, 15, 20, 20, 20, 20,
    24.96, 24.98, 24.99, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
    25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25,
    25.01, 25.01, 25.01, 25.01, 25.01, 25.01, 25.01, 25.01, 25.02, 25.02, 25.02,
    25.03, 30, 30, 30, 30, 30, 30, 30, 35, 35, 35, 35, 40, 40, 40, 40,
    40, 40, 40, 45, 45, 45, 45, 50, 50, 50, 50, 50, 50, 50, 55, 55, 55,
    55, 60, 60, 60, 60, 60, 60, 60, 65, 65, 65, 65, 70, 70, 70, 70, 70,
    70, 70, 75, 75, 75, 75, 80, 80, 80, 80, 80, 80, 80, 85, 85, 85, 85,
    90, 90, 90, 90, 90, 90, 90, 95, 95, 95, 95, 100, 100, 100, 100, 100,
    100, 100, 105, 105, 105, 105, 110, 110, 110, 110, 110, 110, 110, 115,
    115, 115, 115, 120, 120, 120, 120, 120, 120, 120, 130, 130, 130])  # Ejemplo de datos de temperatura
density1 = np.array([ 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498,
    0.4993, 1.0036, 0.0999, 0.2498, 0.4993, 1.0036, 0.0400, 0.4904, 0.2504, 0.0453,
    0.0616, 0.0625, 0.0791, 0.0999, 0.1004, 0.1504, 0.2102, 0.2232, 0.2498, 0.3038,
    0.4772, 0.4993, 0.5000, 0.7173, 0.9604, 1.0000, 1.0000, 1.0000, 1.0036, 1.5000,
    2.0000, 2.5000, 3.0000, 3.5000, 4.0000, 4.5, 0.0625, 0.0897, 0.0897, 0.1600,
    0.1600, 0.2504, 2.4614])  # Ejemplo de datos de densidad

# Datos del segundo gráfico
concentration2 = np.array([0, 1, 2, 3, 4])  # Ejemplo de datos de concentración
temperature2 = np.array([20, 40, 60, 80, 100, 120])  # Ejemplo de datos de temperatura
density2 = np.array([1.0, 1.1, 1.2, 1.3, 1.4])  # Ejemplo de datos de densidad

# Crear una malla de puntos para interpolar los datos
concentration_mesh, temperature_mesh = np.meshgrid(concentration2, temperature2)
points = np.array([(c, t) for c in concentration1 for t in temperature1])
values = np.array([density1 for _ in range(len(temperature1))]).flatten()

# Interpolar los datos del primer gráfico a los puntos del segundo gráfico
density1_interpolated = griddata(points, values, (concentration_mesh, temperature_mesh), method='linear')

# Aplanar los arrays para calcular el error
density2_flat = density2.flatten()
density1_interpolated_flat = density1_interpolated.flatten()
```

```
import numpy as np
from scipy.interpolate import griddata
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error

# Supongamos que tenemos los datos de los gráficos en los siguientes arrays
# Datos del primer gráfico (usualmente se extraen manualmente o con una herramienta de digitización)
concentration1 = np.array([0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498,
    0.4993, 1.0036, 0.0999, 0.2498, 0.4993, 1.0036, 0.0400, 0.4904, 0.2504, 0.0453,
    0.0616, 0.0625, 0.0791, 0.0999, 0.1004, 0.1504, 0.2102, 0.2232, 0.2498, 0.3038,
    0.4772, 0.4993, 0.5000, 0.7173, 0.9604, 1.0000, 1.0000, 1.0000, 1.0036, 1.5000,
    2.0000, 2.5000, 3.0000, 3.5000, 4.0000, 4.5, 0.0625, 0.0897, 0.0897, 0.1600,
    0.1600, 0.2504, 2.4614, 0.0400, 0.4904, 1.0000, 2.4614, 0.0999, 0.2498, 0.4970,
    0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498,
    0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999,
    0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036,
    0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993,
    1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498,
    0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.0999,
    0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036, 1.7550,
    0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690, 1.0036,
    1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993, 0.9690,
    1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970, 0.4993,
    0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498, 0.4970,
    0.4993, 0.9690, 1.0036, 1.7550, 0.0999, 0.2498, 0.4993, 1.0036, 0.0999, 0.2498,
    0.4970, 0.4993, 0.9690, 1.0036, 1.7550, 0.4970, 0.9690, 1.7550])  # Ejemplo de datos de concentración
```

```python
# Calcular el error cuadrático medio (RMSE)
rmse = mean_squared_error(density2_flat, density1_interpolated_flat, squared=False)

print(f"Error cuadrático medio (RMSE): {rmse}")

# Visualizar los datos y la interpolación para asegurarnos de que todo está correcto
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(concentration2, temperature2, density2, c='blue', label='Datos del segundo gráfico')
ax.plot_surface(concentration_mesh, temperature_mesh, density1_interpolated, color='orange', alpha=0.5, label='Interpolación del primer gráfico')
ax.set_xlabel('Concentración (mol/kg)')
ax.set_ylabel('Temperatura (°C)')
ax.set_zlabel('Densidad')
plt.legend()
plt.show()
```