UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA QUÍMICA E INGENIERÍA DE ALIMENTOS

USO DE HERRAMIENTAS DE COMPUTACIÓN CIENTÍFICA PARA LA RESOLUCIÓN DE PROBLEMAS EN INGENIERÍA QUÍMICA

PRESENTADO POR:

RAÚL ALEJANDRO ZURA ZAMORA

PARA OPTAR AL TÍTULO DE:

INGENIERO QUÍMICO

CIUDAD UNIVERSITARIA, OCTUBRE DE 2017

UNIVERSIDAD DE EL SALVADOR

RECTOR:

MSc. ROGER ARMANDO ARIAS ALVARADO

SECRETARIO GENERAL:

MSc. CRISTÓBAL HERNÁN RÍOS BENÍTEZ

FACULTAD DE INGENIERÍA Y ARQUITECTURA

DECANO:

ING. FRANCISCO ANTONIO ALARCÓN SANDOVAL

SECRETARIO:

ING. JULIO ALBERTO PORTILLO

ESCUELA DE INGENIERÍA QUIMICA E INGENIERÍA DE ALIMENTOS

DIRECTORA:

INGA. TANIA TORRES RIVERA

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE INGENIERÍA QUIMICA E INGENIERÍA DE ALIMENTOS

Trabajo de Graduación previo a la opción al Grado de:

INGENIERO QUÍMICO

Título:

USO DE HERRAMIENTAS DE COMPUTACIÓN CIENTÍFICA PARA LA RESOLUCIÓN DE PROBLEMAS EN INGENIERÍA QUÍMICA

Presentado por:

RAÚL ALEJANDRO ZURA ZAMORA

Trabajo de Graduación Aprobado por:

Docente Asesor:

M. Eng ALBA MARISELA SARAVIA CORTEZ

San Salvador, Octubre 2017

Trabajo de Graduación Aprobado por:

Docente Asesor:

M. Eng. ALBA MARISELA SARAVIA CORTEZ

Resumen

En el presente trabajo de graduación se desarrollaron un conjunto de rutinas escritas en las herramientas de computación científica Scilab 5.5.2 y Python 3.5.2, con la finalidad de resolver un conjunto de problemas identificados en las asignaturas de Termodinámica Química I (TQI-115), Termodinámica Química II (TQI-215), Operaciones Unitarias I (OPU-115) y Operaciones Unitarias III (OPU-315), de la carrera de Ingeniería Química.

Se describe de forma general los problemas de cálculo que implican la resolución de ecuaciones no lineales de distintas asignaturas de la carrera de ingeniería química. Considerando las preferencias de los docentes, se especifican los métodos numéricos a emplear en los programas elaborados: método de Regula Falsi, método de Newton, método de Regula Falsi modificado y el método de la secante.

La delimitación inicial de los problemas de cálculo fue llevada a cabo a través de una encuesta a los docentes a cargo de las asignaturas mencionadas. Luego, estos problemas de cálculo se seleccionaron en base a si requerían el uso de métodos numéricos para la resolución de ecuaciones no lineales, y a la frecuencia de aparición de estos problemas en las referencias bibliográficas. Los problemas de cálculo definitivos distribuyeron así: dos problemas en el área de TQI-115, cuatro en el área de TQI-215, tres en el área de OPU-115 y uno en el área de OPU-315.

Los programas en Scilab y Python se elaboraron en base a los paradigmas de programación estructurada y programación modular. En Python también se utilizó la programación orientada a objetos, que permitió el ahorro de líneas de código y proporcionó mayor legibilidad a los programas.

Los porcentajes de error obtenidos para los problemas de cálculo primero, segundo, tercero y séptimo fueron menores al 1% con respecto a las respuestas de los mismos ejercicios presentes en las referencias bibliográficas. Para los demás problemas de cálculo, los porcentajes de error varían en función de la ecuación utilizada para el cálculo de las presiones de vapor y de las capacidades caloríficas.

Agradecimientos

Estoy agradecido a mi familia por haberme brindado palabras de aliento y apoyo cuando lo necesitaba.

A mi docente asesor M. Eng. Marisela Saravia Cortez por haberme aconsejado y por haberme mostrado lo difícil pero gratificante que puede llegar a ser una investigación como lo ha sido este trabajo de graduación.

Y a los docentes de la Escuela de Ingeniería Química e Ingeniería de Alimentos, que a través de sus enseñanzas, me revelaron que lo valioso requiere esfuerzo.

Tabla de Contenido

Lista de Tablas vi

Lista de Figuras xii

Lista de Algoritmos xiii

Nomenclatura xxii

Introducción 1

1. Alcances y planteamiento del problema 3

1.1. Definición del problema 3

1.2. Objetivos 3

1.2.1. Objetivo general 3

1.2.2. Objetivos específicos 3

1.3. Justificación 4

1.4. Alcances y limitaciones 5

1.4.1. Alcances 5

1.4.2. Limitaciones 5

1.5. Antecedentes 6

2. Uso y métodos de resolución de ecuaciones no lineales en Ingeniería Química. Base teórica. 9

2.1. Computación Científica. Definición 9

2.2. La Ingeniería Química y las ecuaciones no lineales. 9

2.2.1. Química General II: 12

2.2.2. Fisicoquímica I: 13

2.2.3. Balance de Materia y Energía: 16

2.2.4. Fisicoquímica II: 18

2.2.5. Operaciones Unitarias I: 21
2.2.6. Termodinámica Química I: 24
2.2.7. Termodinámica Química II: 26
2.2.8. Operaciones Unitarias II 29
2.2.9. Operaciones Unitarias III: 30
2.2.10. Ingeniería de las Reacciones Químicas 31
2.2.11. Procesos de separación y de manejo de sólidos. 33
2.3. Métodos de solución de las ecuaciones no lineales 35
2.3.1. Método de la falsa posición 35
2.3.2. Método de la iteración de punto fijo para una ecuación no lineal 37
2.3.3. Método de Newton-Raphson para una ecuación no lineal 38
2.3.4. Método de la secante para una ecuación no lineal 40
2.4. Herramientas de computación científica para la resolución de ecuaciones no lineales 42
2.4.1. Software para cálculo numérico 42
2.4.2. Software con CAS 47
2.4.3. Hojas de cálculo de Excel 48
3. Metodología a desarrollar 51
3.1. Elaboración del formato de la encuesta a utilizar. 51
3.1.1. Elementos a tomar en cuenta 51
3.1.2. Afinamiento de los ítems del cuestionario de la encuesta y recolección final de datos 51
3.2. Selección de Problemas de Cálculo 52
3.3. Elaboración de programas 52
3.3.1. Resolución de problemas de cálculo sin el uso de rutinas informáticas. 53

3.3.2. Codificación en Scilab y Python 53
3.3.3. Documentación de programas 54
3.4. Elaboración de cuadernos Jupyter 54
3.5. Revisión de los programas por el Usuario 54
3.5.1. Revisión de programas por profesores 55
4. Resultados 56
4.1. Resultados de la encuesta 56
4.1.1. Termodinámica Química I 58
4.1.2. Termodinámica Química II 60
4.1.3. Operaciones Unitarias I 68
4.1.4. Operaciones Unitarias III 71
4.1.5. Selección final de problemas 73
4.2. Resultados de los programas elaborados 104
4.2.1. Semejanzas y diferencias entre Scilab y Python 109
4.2.2. Programas elaborados 110
4.2.3. Cuadernos Jupyter elaborados 133
4.2.4. Manual de usuario elaborado 141
5. Discusión de resultados 142
Conclusiones 146
Recomendaciones 150
Referencias Bibliográficas 151
Anexo A Formato de encuesta y Carta a la Escuela A-1
Anexo B Guía de Usuario B-1
Anexo C Seudocódigos para los programas elaborados en Scilab y Python C-1

Lista de Tablas

Tabla 1.1. Área Diferenciada y sus asignaturas asociadas de la carrera de Ingeniería Química 8
Tabla 2.1. Tipos de equilibrio químico abordados en Química General II 12
Tabla 2.2. Diferentes ecuaciones de estado para gases reales. 14
Tabla 2.3. La ley de Raoult aplicada a problemas de cálculo de punto de burbuja T y punto de rocío T. 19
Tabla 2.4. La ley de Raoult modificada aplicada a problemas de cálculo de punto de rocío P, punto de burbuja T y punto de rocío T. 20
Tabla 2.5. Tipos de problemas de tuberías simples 21
Tabla 2.6. La ley de Raoult modificada aplicada a problemas de cálculo de punto de burbuja T y punto de rocío T. 27
Tabla 2.7. Formulación de Ki para distintas ecuaciones para cálculos EVL. 28
Tabla 3.1. Conjunto de criterios utilizados en la selección de los problemas de cálculo para las distintas asignaturas 52
Tabla 4.1. Datos generales de la encuesta realizada 56
Tabla 4.2. Autores que presentan cálculos del volumen usando la ecuación de estado cúbica genérica 59
Tabla 4.3. Problemas de cálculo TQI-115, luego de aplicación de criterios 60
Tabla 4.4. Autores que presentan cálculos del volumen usando la ecuación de estado cúbica genérica 62
Tabla 4.5. Autores que presentan cálculos de equilibrio líquido-vapor 64
Tabla 4.6. Problemas de cálculo de TQI-215, luego de aplicación de criterios 67
Tabla 4.7. Problemas de OPU-115, luego de aplicación de criterios 71
Tabla 4.8. Problemas de cálculo de OPU-315, luego de aplicación de criterios. 73
Tabla 4.9. Aplicación de criterio adicional a problemas de cálculo 74
Tabla 4.10. Conjunto de problemas de cálculo en las cuatro asignaturas de interés. 80
Tabla 4.11. Métodos de resolución de ecuaciones no lineales ocupadas por los docentes encuestados 81

Tabla 4.12. Descripción de variables utilizadas en programas de los problemas 1°, 2° y 3° 83
Tabla 4.13. Descripción de variables utilizadas en programas del problema 4° 85
Tabla 4.14. Descripción de variables utilizadas en programas del problema 5° 89
Tabla 4.15. Descripción de variables utilizadas en programas del problema 6° 97
Tabla 4.16. Descripción de variables utilizadas en programas de los problemas 7°, 8° y 9° 103
Tabla 4.17. Descripción de variables utilizadas en programas del problema 10° 105
Tabla 4.18. Comparación entre las características de Scilab y Python en la elaboración de programas informáticos. 109
Tabla 4.19. Métodos numéricos utilizados para la resolución de los problemas de cálculo identificados en las asignaturas de TQI-115, TQI-215, OPU-115 y OPU-315. 111
Tabla 4.20. Volúmenes en saturación, calculados con ecuación cúbica genérica 113
Tabla 4.21. Volúmenes de una sustancia gaseosa calculada, ecuación del virial en términos del volumen 114
Tabla 4.22. Entalpía y entropía residuales calculados, ecuación cúbica genérica..... 116
Tabla 4.23. Punto de burbuja de un sistema ternario. Composición del componente 1 118
Tabla 4.24. Punto de burbuja de un sistema ternario. Composición del componente 2 118
Tabla 4.25. Punto de burbuja de un sistema ternario. Temperatura 119
Tabla 4.26. Punto de burbuja de un sistema cuaternario, calculando coeficientes del virial. Composición y del componente 1. 120

Tabla 4.27. Punto de burbuja de un sistema cuaternario, calculando coeficientes del virial. Composición y del componente 2. 121
Tabla 4.28. Punto de burbuja de un sistema cuaternario, calculando coeficientes del virial. Composición y del componente 3 122
Tabla 4.29. Punto de burbuja de un sistema cuaternario, calculando coeficientes del virial. Temperatura 122
Tabla 4.30. Punto de burbuja de un sistema cuaternario, coeficientes del virial dados. Composición y del componente 1. 123
Tabla 4.31. Punto de burbuja de un sistema cuaternario, coeficientes del virial dados. Composición y del componente 2. 124
Tabla 4.32. Punto de burbuja de un sistema cuaternario, coeficientes del virial dados. Composición y del componente 3. 125
Tabla 4.33. Punto de burbuja de un sistema cuaternario, coeficientes del virial ya dados. Temperatura 126
Tabla 4.34. Composiciones de equilibrio en la reacción de C_6H_6 y H_2 para formar C_6H_{12} a 1393 K. Composición C_6H_6. 127
Tabla 4.35. Composiciones de equilibrio en la reacción de C_6H_6 y H_2 para formar C_6H_{12} a 1393 K. Composición H_2 128
Tabla 4.36. Diámetro de una tubería simple, contando pérdidas primarias únicamente. 130
Tabla 4.37. Flujo volumétrico de una tubería simple, contando pérdidas primarias y secundarias 130
Tabla 4.38. Caudales a través de un sistema de 2 tuberías en paralelo, contando pérdidas primarias y secundarias 131
Tabla 4.39. Composiciones de equilibrio de los productos líquido y vapor de una destilación instantánea. Composicion x_1 (Benceno)..... 134
Tabla 4.40. Composiciones de equilibrio de los productos líquido y vapor de una destilación instantánea. Composición x_2 (Tolueno) 135
Tabla 4.41. Composiciones de equilibrio de los productos líquido y vapor de una destilación instantánea. Composicion y_1 (Benceno)..... 136

Tabla 4.42. Composiciones de equilibrio de los productos líquido y vapor de una destilación instantánea. Composicion y_2 (Tolueno) 137
Tabla B.1. Relación entre cal_fz (programa principal) y prueba_sat, fv2, rootRegulaFA2p2 y rootNewton_cubic2 para los problemas 1° y 2°.....B-7
Tabla B.2. Relación entre Hr_Sr_prin (programa principal) y Hr_Sr_calc, prueba_sat y fv2 para el problema 3° B-10
Tabla B.3. Relación entre fv2 y rootRegulaFA2p2 y rootNewton_cubic2 para el problema 3° B-10
Tabla B.4. Relación entre calc_ELV (programa principal) y opc y cpre_ing para los problemas 4° y 5° B-13
Tabla B.5. Relación entre coac, opc y BurbT para los problemas 4° y 5°.....B-13
Tabla B.6. Relación entre coac, BurbT, phiCM, Wcomp, NRTLcomp y UNIFAC para los problemas 4° y 5°.....B-14
Tabla B.7. Relación entre UNIFAC y tablas_read, tab_amk y tab_Rk_Qk para para los problemas 4° y 5° B-15
Tabla B.8. Relación entre BurbT y fpres_vap y rootSecantePlus2Ap2 para los problemas 4° y 5° B-15
Tabla B.9. Relación entre BurbT, imp_op_tab e imp_tab para los problemas 4° y 5°.....B-16
Tabla B.10. Relación entre Cal_EqC (programa principal) y func_cad y defK para el problema 6°.....B-18
Tabla B.11. Relación entre defK, ccp_ing y cK para el problema 6°.....B-18
Tabla B.12. Relación entre cK, RomberT, Calc_DH_DS y func_cad para el problema 6°.....B-19
Tabla B.13. Relación entre cK, Cal_EqC y rootSecantePlus2Ap2 para el problema 6°.....B-19
Tabla B.14. Relación entre Calc_D (programa principal) y D_TS_calc y f_fric para el problema 7°.....B-21
Tabla B.15. Relación entre D_TS_calc y rootSecante_Plus2Ap2 para el problema 7°.....B-22

Tabla B.16. Relación entre Calc_fl_vol (programa principal) y Fl_vol_TubS_calc para el problema 8° B-23
Tabla B.17. Relación entre Fl_vol_TubS_calc y rootSecantePlus2Ap2 para el problema 8° B-24
Tabla B.18. Relación entre Calc_tub_par (programa principal) y FL_vol_par_calc y f_fric para el problema 9° B-25
Tabla B.19. Relación entre FL_vol_par_calc, imp_op_tab e imp_tab para el problema 9° B-25
Tabla B.20. Relación entre f_fric y rootSecantePlus2Ap2 para el problema 9° B-26
Tabla B.21. Relación entre Kdatos (programa principal) y opcK, opdK y cpre_ing para el problema 10° B-29
Tabla B.22. Relación entre opcK, coac y BurP para el problema 10° B-29
Tabla B.23. Relación entre coac, Wcomp, NRTLcomp, UNIFAC, BurP y phiCM para el problema 10° B-30
Tabla B.24. Relación entre tablas_read, UNIFAC, tab_amk y tab_Rk_Qk para el problema 10° B-31
Tabla B.25. Relación entre imp_op_tab, imp_tab, BurP y fpres_vap para el problema 10° B-32
Tabla B.26. Relación entre opcK y RoP para el problema 10° B-32
Tabla B.27. Relación entre coac, Wcomp, NRTLcomp, UNIFAC, RoP y phiCM para el problema 10° B-33
Tabla B.28. Relación entre imp_op_tab, imp_tab, RoP y fpres_vap para el problema 10° B-34
Tabla B.29. Relación entre opcK y BurbT para el problema 10° B-34
Tabla B.30. Relación entre coac, Wcomp, NRTLcomp, UNIFAC, BurbT y phiCM para el problema 10° B-35
Tabla B.31. Relación entre BurbT, rootSecantePlus2Ap2 y fpres_vap para el problema 10° B-36
Tabla B.32. Relación entre BurbT, imp_op_tab e imp_tab para el problema 10° B-36

Tabla B.33. Relación entre opcK y RoT para el problema 10°.....B-37
Tabla B.34. Relación entre coac, Wcomp, NRTLcomp, UNIFAC, RoT y phiCM para el problema 10°.....B-38
Tabla B.35. Relación entre RoT, rootSecantePlus2Ap2 y fpres_vap para el problema 10°.....B-39
Tabla B.36. Relación entre RoT, imp_op_tab e imp_tab para el problema 10°.....B-39

Lista de Figuras

Figura 2.1. Una representación gráfica del método de falsa posición. Los triángulos semejantes utilizados para deducir la fórmula para el método están sombreados. 36
Figura 2.2. Representación gráfica de la convergencia y de la divergencia del método de punto fijo. 38
Figura 2.3. Representación gráfica del método de Newton-Raphson. Se extrapola una tangente a la función en x_i (esto es f'(x_i)) hasta el eje x para obtener una estimación de la raíz x_i+1. 39
Figura 2.4. Representación gráfica del método de la secante 41
Figura 2.5. Programa elaborado para almacenar a funF 43
Figura 2.6. Respuestas obtenidas para el primer caso de una función, descrita en la ecuación e.1 43
Figura 2.7. Programa que llama a la función descrita en la ecuación e.1 y a fsolve para el cálculo de la raíz 44
Figura 2.8. Búsqueda de raíces de dos polinomios utilizando a roots() de NumPy. 45
Figura 2.9. Ejemplos de cálculo de raíces utilizando las funciones solve, zeros, polyRoots, nSolve y factor. 48
Figura 2.10. Digitado de datos en la hoja de cálculo para el ejemplo de uso de Buscar Objetivo 50
Figura 2.11. Respuesta de Buscar Objetivo. 50
Figura 4.1. Número de autores que desarrollan los problemas identificados en la asignatura de TQI-215 del 3 al 21 79
Figura 4.2. Número de autores que desarrollan los problemas identificados en la asignatura de OPU-115 del 23 al 29 79
Figura 4.3. Ejemplo de interfase de programa desarrollado 112
Figura 4.4. Ejemplo de cuaderno electrónico Jupyter elaborado 138

Lista de Algoritmos

Métodos numéricos

RomberT C-1

TrapEq C-1

Romberg C-1

rootNewton_cubic2 ..... C-2

rootNewton_cubic2 ..... C-2

rootNewton_cubic2_evalf C-4

rootBracket_3p3 ..... C-4

rootBracket_3 ..... C-4

rootBracket_evalf C-5

rootRegulaFA2p2 ..... C-6

rootRegulaFA2 ..... C-6

rootRegulaFA2_evalf C-7

rootRegulaFA_mod C-8

rootRegulaFA_mod C-8

rootRegulaFA_mod_evalf C-10

rootSecantePlus2Ap2 ..... C-11

fp C-11

rootSecantePlus2A C-11

rootSecantePlus2A_evalf C-13

Termodinámica Química I

fv2 ..... C-14

fcg C-14

dfcg C-14
ffz C-14
dffz C-15
fv2 ..... C-15
prueba_sat C-20
calc_phi_lv_cubic C-20
calc_pres C-20
pres_sat_cubic C-21
desc_region_cubic C-22
desc_region_cubic_first C-23
desc_region_cubic_second C-24
cal_fz C-26

Termodinámica Química II

fv2 ..... C-29
ffz C-29
dffz C-29
fv2 ..... C-29
prueba_sat C-31
calc_phi_lv_cubic C-31
calc_pres C-31
pres_sat_cubic C-32
desc_region_cubic C-33
Hr_Sr_calc C-34
l_alf_evalf C-34
fp C-34

ln_alf C-35
calcI C-35
calcHr C-36
calcSr C-36
calcGr C-36
Hr_Sr_imp C-36
Hr_Sr_prin C-37
cpre_ing C-39
tab_c C-39
ingreso_constantes C-39
cpre_ing C-40
fpres_vap C-42
ant_van_ness C-42
ant_prausnitz_p_4ed C-42
ant_prausnitz_p_5ed C-42
wag_p C-42
antmod_p C-43
wagmod_p C-43
riedel_p C-43
ant_prausnitz_t_4ed C-44
ant_prausnitz_t_5ed C-44
ant_van_ness_t C-44
wag_t C-44
antmod_t C-44

wag_mod_t C-45
riedel_t C-45
fkt_p C-45
fkt_t C-45
t_pv C-46
Psat C-47
BurbT C-49
te2 ..... C-49
yBurbt C-49
coac C-53
mat_val C-53
mat_car C-53
coac C-54
in_dat_cof C-55
Wcomp C-59
NRTLcomp C-60
UNIFAC C-62
phiCM C-64
phisat C-64
phiCM C-65
tablas_read C-68
tab_split C-68
tab_eval C-68
col_extract C-69

take_once_adjacent_dups2 ..... C-69
sg_mod C-70
id_comp_Rk_Qk C-71
val_amk C-72
dict_create C-73
dict_mapping C-73
search_value C-73
tabs_RQ_av C-74
opc C-75
mat_carB C-75
ingreso_constantesB C-76
mat_valB C-76
B_tab C-77
opc C-78
imp_tab C-80
phitab_imp C-80
coactab_imp C-80
pstab_imp C-81
comptab_imp C-81
imp_op_tab C-81
calc_ELV C-83
names C-83
cad_comp_imp C-83
com_ing C-83
Calc_ELV C-84

rep C-87
CalcDH_DS C-87
cpH_SVN C-87
cpH_Prausnitz_4ed C-87
cpH_Prausnitz_5ed C-88
cpH_Perry7_1 ..... C-88
cpH_Perry7_2 ..... C-88
cpH_Perry7_3 ..... C-88
cpH_Perry8_1 ..... C-89
cpH_Perry8_2 ..... C-89
Int_cpH_evalf C-89
Int_cpS_evalf C-90
cK C-90
consEq C-90
comEq C-91
Rlimit C-91
defK C-92
ing_delHG C-92
defConsEq C-93
ccp_ing C-96
tab_c C-96
ingreso_constantes C-96
ccp_ing C-96
func_cad C-99

lab_imp C-99
names C-99
cad_comp_imp C-99
com_ing C-100
func_def C-101
func_cad_impH C-101
func_cad_impS C-102
Calc_EqC C-102
Calc_EqC C-102
rep C-107

Operaciones Unitarias I

f_fric C-108
FD_flujo_laminar C-108
FD_flujoturc_tub_rug C-108
FD_flujotr_Col C-108
Reynolds C-108
vel C-108
Q_calc C-109
calc_fdarcy_col C-109
D_TS_calc C-110
D_fun C-110
calc_DTs C-111
Calc_D C-112
Fl_vol_TubS_calc C-116

fl_vol_fun.....C-116
calc_fl_vol C-117
Calc_fl_vol C-118
Fl_vol_par_calc C-123
fl_tot C-123
h.....C-123
desvh.....C-124
vel_hf.....C-124
tab.....C-124
Qs_calc C-124
imp_tab.....C-126
fac_fric_imp.....C-126
vel_imp C-126
caudales_imp.....C-126
imp_op_tab C-127
Calc_tub_par.....C-128

Operaciones Unitarias III

BurbP C-132
pres C-132
yBurbp C-133
RoP.....C-135
pres C-135
xRop C-136
RoT C-140

te2 ..... C-140
xRot C-141
opcK C-148
mat_carB C-148
ingreso_constantesB C-149
mat_valB C-150
B_tab C-151
opcK C-152
opdK C-154
Vaporz_ins C-156
vpsat C-156
cx C-156
valini C-156
obtK C-157
fK C-158
vi C-158
cal_int C-161
imp_op_tab_Kcal C-163
Kdatos C-164
names C-164
cad_comp_imp C-165
com_ing C-165
Kdatos C-166

Nomenclatura

K_p: Constante de equilibrio para reacciones en fase gaseosa, utilizando presiones parciales.

P_A, P_B…: Presiones parciales de los compuestos A, B…

K_C: Constante de equilibrio para reacciones en fase gaseosa o líquida, utilizando concentraciones.

K_{ps}: Constante de equilibrio para reacciones acuosas entre sales de baja solubilidad, utilizando concentraciones.

$[A]$, $[B]$…: Concentraciones de los compuestos A, B…

P: Presión total de un compuesto gaseoso

P_c: Presión crítica de un compuesto gaseoso

T: Temperatura de un compuesto gaseoso o del contenido de un reactor

T_{ref}: Temperatura de referencia a la que se lleva a cabo una reacción

T_c: Temperatura crítica de un compuesto gaseoso

V_m ó $V_{m,gas}$: Volumen molar de un compuesto gaseoso

$V_{m,vapor}$: Volumen molar de saturación de un compuesto gaseoso

$V_{m,líquido}$: Volumen molar de saturación de un compuesto líquido

Z_{gas}: Coeficiente de compresibilidad de un compuesto gaseoso

Z_{vapor}: Coeficiente de compresibilidad de un compuesto gaseoso en estado de saturación

$Z_{líquido}$: Coeficiente de compresibilidad de un compuesto líquido en estado de saturación

a: parámetro de las ecuaciones de estado de Van der Waals, Redlich-Kwong y Peng-Robinson

a': parámetro de la ecuación de estado de Soave-Redlich-Kwong

b: parámetro de las ecuaciones de estado de Van der Waals, Redlich-Kwong, Soave-Redlich-Kwong, Peng-Robinson y de la ecuación de estado cúbica general.

λ: parámetro de las ecuaciones de estado de Soave-Redlich-Kwong

α: parámetro de las ecuaciones de estado de Peng-Robinson

κ: parámetro de las ecuaciones de estado de Soave-Redlich-Kwong y Peng-Robinson

T_r: Temperatura reducida de un compuesto gaseoso

P_r: Presión reducida de un compuesto gaseoso

B, C…: Coeficientes de la ecuación del virial en términos del volumen o volumen molar (ecuación de Kammerlingh-Onnes)

B', C' …: Coeficientes de la ecuación del virial en términos de la presión (ecuación de Holborn)

ω: factor acéntrico o parámetro de la ecuación de Benedict-Webb-Rubin

β: Parámetro de la ecuación Benedict-Webb-Rubin o de la ecuación de estado cúbica general

σ: Parámetro de la ecuación Benedict-Webb-Rubin o de la ecuación de estado cúbica general

η: Parámetro de la ecuación de Benedict-Webb-Rubin

γ: Parámetro de la ecuación de Benedict-Webb-Rubin o peso específico de un fluido

$F_{j,i}$: flujo másico (masa por unidad de tiempo) de la corriente de salida j para la especie i

ν_i: Número estequiométrico de la especie i

ν: Número estequiométrico total de un sistema reactivo

M_i: Peso molecular de la especie i

r: Velocidad de reacción

R_i: Producción o agotamiento molar de alguna especie química i

dQ/dt: Rapidez de transferencia de calor hacia el sistema

dW/dt: Rapidez del trabajo ejercida o aplicada sobre el sistema

$\widehat{H}_i(T_j)$: Entalpía específica de la especie i en la corriente de salida j a la temperatura T_j

ΔH_{Rxn}: Calor de reacción

$\widehat{H}_{i,ref}$: Entalpía específica de referencia de la especie i, definida a una temperatura, presión y fase de referencia.

$\widehat{H}_{i,2}$: Entalpía específica la especie i en un estado 2 distinto al de referencia.

$P_{sistema}$: Presión del sistema formado por una mezcla de componentes o por compuestos químicos de un reactor.

$P_i^{sat}(T)$: Presión de saturación de la especie i, definida a la temperatura T

T_i^{sat}: Temperatura de saturación de la especie i.

x_i: Fracción molar del componente i en la fase líquida de una mezcla

y_i: Fracción molar del componente i en la fase vapor de una mezcla

γ_i: Coeficiente de actividad del componente i en la fase líquida de una mezcla

$\{x_i\}$: Conjunto de las composiciones de la fase líquida de una mezcla

$\{y_i\}$: Conjunto de las composiciones de los componentes de la fase vapor de una mezcla

$\{\gamma_i\}$: Conjunto de coeficientes de actividad de los componentes en la fase líquida de una mezcla

Q: Caudal de un compuesto líquido o gaseoso en una tubería

L: Longitud de una tubería

D: Diámetro de una tubería o de un reactor de flujo pistón

v: Velocidad de un compuesto líquido o gaseoso en una tubería

ϵ: Rugosidad de una tubería o parámetro de la ecuación de estado cúbica general

g: Aceleración de la gravedad

$h_{L,s}$: Pérdida de cabeza por fricción superficial

$h_{L,f}$: Pérdida de cabeza por fricción de forma

h_L: Pérdida de cabeza por fricción

p_1: Presión de un fluido en el punto 1 de una tubería

p_2: Presión de un fluido en el punto 2 de una tubería

z_1: Elevación de un fluido en el punto 1 de una tubería

z_2: Elevación de un fluido en el punto 2 de una tubería

Ω: Parámetro adimensional de la ecuación de estado cúbica general

Ψ: Parámetro adimensional de la ecuación de estado cúbica general

$a(T)$: Parámetro no adimensional de la ecuación de estado cúbica general

$T_{\sigma,j}$: Temperatura en los alrededores

dS_G/dt: Generación de entropía por unidad de tiempo

$\hat{S}_i(T_j)$: Entropía específica de la especie i de la corriente de salida j a la temperatura T_j

$\langle C_P^{ig} \rangle_S$: Capacidad calorífica media utilizada en el cálculo de la entropía de un compuesto en un sistema cerrado

ΔH_{ideal}: Cambio de entalpía de un gas ideal

$|\nu_i|$: Número estequiométrico del reactivo o producto i en un reactor

ΔH_{ideal}: Cambio de entalpía de un gas ideal

$\Delta H_{f,298}^{\circ}$: Calores de formación a 298 K

$\Delta G_{f,298}^{\circ}$: Energías de Gibbs de formación a 298 K

H^R: Entalpía residual de un gas real

S^R: Entropía residual de un gas real

G^R: Energía de Gibbs residual de un gas real

Φ_i: Coeficiente Phi utilizado en la formulación Gamma-Phi

$\hat{\phi}_i$: Coeficiente de fugacidad de la especie i en solución

ϕ_i^{sat}: Coeficiente de fugacidad de la especie pura i, definida a la temperatura de saturación de la solución.

$\{\Phi_i\}$: Conjunto de los coeficientes Phi de los componentes en fase vapor de una mezcla

z_i: Composición global del componente i de una mezcla

$\{z_i\}$: Conjunto de las composiciones globales de una mezcla

K_i: Coeficiente K del componente i de una mezcla

V_i: Moles de vapor de una mezcla en destilación instantánea

U_D: Coeficiente total de transferencia de calor de un intercambiador incrustado.

U_C: Coeficiente total de transferencia de calor de un intercambiador limpio.

R_D: Resistencia térmica de la incrustación

T_v: Temperatura global de la fase gaseosa

H_v: Entalpía del gas

T_i: Temperatura de la interfase gas-líquido

H_i: Entalpía de la interfase gas-líquido

h_C: coeficiente de transferencia de masa en unidades de (moles transferidos)/(tiempo* área*(moles/volumen))

k_Y: coeficiente de transferencia de masa en unidades de (masa transferida)/(tiempo*

área*(masa de A/masa de B))

h_L: coeficiente de transferencia de masa en unidades de (masa transferida)/(tiempo* área*(moles/volumen))

a: superficie específica interfacial promedio para la transferencia de masa en unidades de área/volumen

τ_i: Espacio-tiempo de un reactivo i a través de un reactor

r_i: Velocidad de reacción del reactivo i

C_i: Concentración del efluente de salida del i-ésimo reactor

A: Área a través de la cual ocurre la transferencia de calor y en la cual se basa el coeficiente de transferencia de calor U

$\dot{Q}$: Velocidad de suministro de calor

U: Coeficiente de transferencia de calor

T_m: Temperatura de la fuente de calor

F_{A0}: Velocidad molar inicial del reactivo A

F_i: Velocidad molar de los productos i

$x_{A\ entrada}$: Conversión de A a la entrada del reactor

$x_{A\ salida}$: Conversión de A a la salida del reactor

$\Delta H_{R\ a\ T0}$: Cambio de entalpía de reacción a una temperatura de entrada T0, por unidad de grado de avance ε.

$\bar{C}_{p,i}$: Calor específico promedio por unidad de masa del producto i.

k_0: Constante de reacción a una temperatura de referencia T_{ref}

E: Energía de activación de la reacción

ρ_p: Densidad de partícula

ρ: Densidad de un fluido

D_p: Diámetro de partícula para partículas esféricas o el diámetro característico para partículas no esféricas

C_D: Coeficiente de rozamiento

μ: Viscosidad de un fluido

N_{Re}: número de Reynolds

x_r: valor respuesta para un método de resolución de ecuaciones no lineales

x_l: menor valor de x de un intervalo que encierra al valor respuesta x_r, utilizando el método de Regula Falsi

x_u: mayor valor de x de un intervalo que encierra al valor respuesta x_r, utilizando el método de Regula Falsi

$f(x_l)$: función f evaluada en x_l

$f(x_u)$: función f evaluada en x_u

ε_a: porcentaje de error absoluto entre el valor de la aproximación de la iteración actual x_r^{nuevo} y la iteración anterior $x_r^{anterior}$.

$f'(x_i)$: derivada de una función evaluada en el punto x_i

δx_i: pequeño valor fraccionario utilizado para el cálculo de la derivada de una función f en un punto x_i.

Introducción

Una ecuación no lineal puede modelarse como:

$$f(x) = 0$$

para la que se busca los valores x que vuelven 0 a f(x).

Ejemplos de ecuaciones no lineales son las ecuaciones polinómicas, logarítmicas y exponenciales. Para este tipo de ecuaciones deberá ocuparse un método numérico para determinar los valores x que resuelvan a f(x).

Los métodos numéricos utilizados para este tipo de ecuaciones comúnmente se dividen en métodos cerrados y métodos abiertos. Los métodos cerrados, como el método de regula falsi o de regula falsi modificado, requieren de intervalos que encierran o contienen la raíz, y que después reducen sistemáticamente el tamaño de dichos intervalos. Mientras que los métodos abiertos, como el método de Newton-Raphson, de la secante y de iteración de punto fijo, no requieren de un intervalo inicial y son más eficientes en cuanto al número de iteraciones requeridas pero la convergencia a la respuesta no está garantizada.

Estos métodos numéricos tienen utilidad en diversos problemas de cálculo de Ingeniería Química, abarcando asignaturas como Química II, Físicoquímica I, Balances de Materia y Energía, Físicoquímica II, Operaciones Unitarias I Termodinámica Química I, Termodinámica Química II, Operaciones Unitarias II, Operaciones Unitarias III, Ingeniería de las Reacciones Químicas y Procesos de separación y manejo de sólidos.

La delimitación de los problemas de cálculo iterativo para cuatro de las asignaturas mencionadas fue realizada en base a una encuesta hecha a los docentes de la Escuela de Ingeniería Química e Ingeniería de alimentos, en diciembre del 2016. Se identificaron los siguientes problemas:

Termodinámica Química I:

- Cálculo de volúmenes de saturación a partir de ecuaciones para gases reales (ecuación cúbica general y ecuación del virial), donde además se realiza el cálculo de la presión de saturación, a elección del usuario;

Termodinámica Química II:

- Cálculo de las composiciones de los componentes de una mezcla líquido-vapor y de la temperatura de burbuja, donde también se resuelven funciones no lineales de la temperatura de saturación, contando con la presión de saturación;
- Cálculo del avance de reacción para reacciones simples de gases ideales;

Operaciones Unitarias I:

- Cálculo de los diámetros de tubería, flujos volumétricos de fluidos a través de tuberías en serie y de caudales en sistemas de tuberías en paralelo, donde también es necesario resolver funciones no lineales del factor de fricción.

Operaciones Unitarias III:

- Cálculo de las composiciones de los productos en un proceso de destilación instantánea, donde también se resuelven funciones no lineales de la temperatura de saturación, contando con la presión de saturación;

El uso de software para la aplicación de estos métodos numéricos permite fijar la atención en la respuesta del problema y no en el proceso de resolución. Actualmente las alternativas al software comercial, tales como Scilab y Python, no implican costo alguno.

Scilab cuenta con un ambiente de desarrollo integrado, el cual es una interfase gráfica de usuario en el que se incluye un editor de código, la consola o terminal y la documentación.

Python es un lenguaje de programación y el ambiente de desarrollo integrado lo proporciona Spyder, otro software de licencia libre. Spyder reúne una terminal de Python simple, una terminal IPython, que es una terminal de Python más amigable, el cuaderno electrónico Jupyter y los vínculos a la documentación en línea. Además el software libre Anaconda permite la instalación de paquetes externos a la biblioteca estándar de Python, evitando cualquier dificultad de compatibilidad entre los paquetes.

Se escribieron programas propios en Scilab y Python para los métodos numéricos. Los cuadernos Jupyter hacen uso de los programas escritos en Python, pero además permiten el formateo de ecuaciones, adición de texto en forma de títulos y de texto normal y la inclusión de recursos multimedia.

1. Alcances y planteamiento del problema

1.1. Definición del problema

Los problemas de cálculo en las asignaturas de operaciones unitarias y termodinámica química en ingeniería química, a menudo implican el uso de algoritmos de solución repetitivos. Es posible automatizar dichos cálculos a través del uso de herramientas de software, en especial del software de licencia libre. Este trabajo de graduación pretende desarrollar una serie de programas informáticos escritos en Python 3.5.2 y Scilab 5.5.2 para un conjunto de problemas de cálculo seleccionados, en las asignaturas de operaciones unitarias y termodinámica química, explorando a su vez las opciones interactivas de dichos programas en la interfaz web del cuaderno electrónico Jupyter Notebook.

1.2. Objetivos

1.2.1. *Objetivo general*

- Desarrollar un conjunto de rutinas para uso de herramientas de computación científica de acceso libre como Scilab 5.5.2 y Python 3.5.2, que permitan la resolución de problemas de cálculo seleccionados, en áreas del conocimiento de ingeniería química.

1.2.2. *Objetivos específicos*

1. Seleccionar problemas de cálculo de ingeniería química en las áreas de Operaciones Unitarias y Termodinámica Química, a ser solucionados a través del uso de herramientas de computación científica.
2. Desarrollar la solución de los problemas seleccionados, a través de métodos que planteen el uso de software de acceso libre como Scilab 5.5.2 y Python 3.5.2.
3. Comparar los resultados obtenidos de la aplicación de herramientas de computación científica, con la resolución de dichos problemas por otros métodos de cálculo.

4. Examinar las características interactivas del cuaderno Jupyter, haciendo uso de las rutinas elaboradas en Python 3.5.2, para la resolución de problemas de cálculo seleccionados, en las áreas de Operaciones Unitarias y Termodinámica Química de Ingeniería Química.

1.3. Justificación

La resolución de problemas de cálculo en Ingeniería Química, que involucran ecuaciones no lineales, es un proceso laborioso dada la naturaleza iterativa de este tipo de problemas. Su resolución se facilita a través de herramientas de computación científica como Scilab y Python. Se eligieron las áreas temáticas de Termodinámica Química I y II y de Operaciones Unitarias I y III.

La principal razón para utilizar Scilab en lugar de Octave (ambos software de licencia libre) es debido a la experiencia personal en cuanto a su uso en la asignatura "Sistemas Electromecánicos" donde se desarrollaron operaciones de manipulación de matrices. Otra buena razón consiste en la continuación del trabajo de una serie de guías de laboratorios destinados a la asignatura "Termodinámica Química II", escritos en Scilab.

Una contribución adicional de esta investigación consistirá en la presentación de las características del cuaderno Jupyter. Esto incluye la combinación de codificación y ejecución de código, edición de texto narrativo y ecuaciones matemáticas.

1.4. Alcances y limitaciones

1.4.1. *Alcances*

1) Se seleccionarán problemas de cálculo a resolver de las áreas de Operaciones Unitarias y de Termodinámica Química, de acuerdo al contenido programático de las asignaturas del plan de estudios de la carrera de Ingeniería Química de la UES.
2) Se planea comentar en forma apropiada las rutinas informáticas elaboradas para intercalar explicaciones concisas de sus componentes. Una vez escritas las rutinas, se comparará los resultados obtenidos con los de los ejercicios resueltos, según se plantean, en los libros de referencia.
3) Se consultará en libros digitales y recursos en línea acerca de los lenguajes de programación de Scilab 5.5.2 y Python 3.5.2, cualquier dificultad encontrada en la escritura de las rutinas informáticas.

1.4.2. *Limitaciones*

La creación de flujogramas en un archivo de Word 2013 puede complicarse debido a que la modificación del archivo puede ocasionar que las figuras y las líneas conectoras cambien de lugar, desordenando todo el trabajo hecho en el flujograma; además la conexión de las figuras es trabajoso. Una solución será el uso de Dia 0.97.2, un software más adecuado porque soluciona los defectos referidos en Word.

La creación de interfaces gráficas de usuario o GUI's como complemento a los programas a elaborar es un proceso largo y complejo. Una solución sustitutiva será la utilización del cuaderno electrónico Jupyter, para aprovechar las capacidades de visualización de resultados y la adición de recursos multimedia.

1.5. Antecedentes

La elaboración de programas, permite reconocer las siguientes asunciones para la resolución de problemas de cálculo (Edgar, T.F., 2006), tales como:

- Como debe lucir una respuesta correcta
- Cuáles son los datos de entrada y salida
- Una clara organización de pensamiento, lógica y de cálculos
- La posible existencia de errores.

La ingeniería química se auxilia con frecuencia de herramientas para la computación científica. La computación científica o ciencia computacional no es más que una colección de herramientas, técnicas y teorías requeridas para resolver en una computadora, un problema científico o ingenieril capaz de ser representado a través de modelos matemáticos (Golub & Ortega, 1992, pp. 1–2).

En ingeniería química, los cálculos de punto de rocío, punto de burbuja y vaporización instantánea isotérmica para mezclas multi-componentes, el cálculo de la conversión en un reactor isotérmico y su volumen, y el cálculo del factor de compresibilidad o del coeficiente de fugacidad requieren la solución de una ecuación no lineal (Shacham, M., 1989). Matlab® se considera una excelente herramienta para la computación científica principalmente debido a su flexibilidad, generalidad, gran utilidad para aplicaciones de ingeniería, estabilidad y simplicidad en el uso y en el aprendizaje. Sin embargo se reconoce la dificultad económica que representa el costo por el uso del software para los departamentos de ingeniería, en la forma de "licencias de sitio" (Öhrström, L., Svenson, G., Larson, S., Christie, M., & Niklasson, C., 2005). Encontrar una alternativa a Matlab en computación científica, entre los distintos paquetes de software libre que se han desarrollado a lo largo de los años, implica la evaluación de las características de distintos software. Soni (2008) realizó la comparación de distintos ambientes para la computación científica considerando a: C en Visual C++ 2005 Express Edition, Java en Eclipse 3.2, LabVIEW 8.2, Maple 11 en una maquina Athena del MIT, Mathematica 5.2, Matlab 7.2 (R2006a), Octave 2.1.72, Python 2.4, R 2.4.1, Scilab 4.1.2 comparando los aspectos de tiempo de ejecución en operaciones de álgebra lineal y en la

resolución de ecuaciones diferenciales, la gestión de la memoria en las operaciones realizadas, los asuntos de incompatibilidades entre los lenguajes, la sintaxis del código y la convergencia del programa a la solución correcta en problemas de optimización.

En un estudio más reciente, Sharma y Gobbert (2010) compararon los paquetes de software libre Freemat, Scilab y Octave con Matlab. Los resultados de los paquetes de software fueron idénticos en varios aspectos aunque Scilab exhibió una limitación en el tamaño del sistema lineal a resolver. Sharma y Gobbert (2010) concluyen que GNU Octave es el más compatible con Matlab debido a sus habilidades numéricas y la similitud de su sintaxis. Ahora bien Scilab (Scilab Enterprises, 2015) cuenta con Scicos, un modelador de sistemas dinámicos parecido a Simulink de Matlab. Ni Freemat ni Octave cuenta con dicha característica.

Otro lenguaje de programación de alto nivel es Python, el cual combina sintaxis sencilla, recursos en línea abundantes y un rico ecosistema de herramientas enfocadas científicamente con un fuerte énfasis en la comunidad (Perkel, J. M., 2015a). Anaconda, una distribución gratuita de Python, agrupa alrededor de 200 de las bibliotecas más populares de Python para la ciencia, las matemáticas, la ingeniería y el análisis de datos (Van Noorden, R., 2015), aunque las bibliotecas más utilizadas son NumPy (matrices matemáticas), SciPy (álgebra lineal, ecuaciones diferenciales y procesamiento de señales), SymPy (matemáticas simbólicas), matplotlib (ploteado de gráficos) y Pandas (análisis de datos) (Perkel, J. M., 2015a).

Las asignaturas indicadas en la Justificación para la identificación de problemas de cálculo iterativo con ecuaciones no lineales, forman parte del Área Diferenciada de la carrera de Ingeniería Química (Universidad de El Salvador, 2011, pp. 197–202).

Dicha Área Diferenciada contiene múltiples Áreas Específicas de Conocimiento, entre las cuales están las categorías de Operaciones Unitarias de Transporte y Manejo de Fluidos, de Operaciones Unitarias de Transporte de Masa y de Termodinámica Química aplicada al manejo de sistemas de plantas de potencia y de refrigeración y al uso eficiente de energía. La Tabla 1.1 también presenta las demás categorías.

La categoría de Termodinámica Química, con un total de 8 U.V. abarca un 4.42 % del total de U.V. de las asignaturas en el Pensum de la carrera. Esta categoría contiene a las asignaturas

de Termodinámica Química I y Termodinámica Química II, impartidas en el ciclo VI y VII, respectivamente.

Mientras que las categorías relacionadas con las Operaciones Unitarias equivalen a 12 U.V. y corresponden al 6.63 % del total de U.V. Las Operaciones Unitarias de Transporte y Manejo de Fluidos está asociada a Operaciones Unitarias I (ciclo VI) y las Operaciones Unitarias de Transporte de Masa con Operaciones Unitarias III (ciclo VIII). Cada una cuenta con 4 U.V. (2.21% del total de U.V. en el pensum).

Tabla 1.1. Área Diferenciada y sus asignaturas asociadas de la carrera de Ingeniería Química

Área Diferenciada			
No.	Área específica de conocimiento	Asignaturas asociadas	U.V.
1	Fisicoquímica	Fisicoquímica I	4
		Fisicoquímica II	4
2	Balance de Masa y Energía	Balance de Materia y Energía	4
3	Operaciones Unitarias de Transporte y Manejo de Fluidos	Operaciones Unitarias I	4
4	Operaciones Unitarias de Transporte de Calor	Operaciones Unitarias II	4
5	Operaciones Unitarias de Transporte de Masa	Operaciones Unitarias III	4
6	Manejo y Separación de Sólidos en Procesos Industriales	Proceso de Separación y Manejo de Sólidos	4
7	Termodinámica Química aplicada al manejo de sistemas de plantas de potencia y de refrigeración y al uso eficiente de energía	Termodinámica Química I	4
		Termodinámica Química II	4
8	Fenómenos de Corrosión	Principios de Electroquímica y Corrosión	4
9	Ingeniería de las Reacciones Químicas	Ingeniería de las Reacciones Químicas	4
10	Análisis Químico	Química Analítica I	4
		Análisis Instrumental	4
11	Ciencias Químicas Básicas	Química Orgánica I	4
		Química Inorgánica I	4
		∑	60

2. Uso y métodos de resolución de ecuaciones no lineales en Ingeniería Química. Base teórica.

2.1. Computación Científica. Definición

Según Golub y Ortega (1992, p. 1), La computación científica es la colección de herramientas, técnicas y teorías requeridas para resolver, en una computadora, modelos matemáticos en Ciencia e Ingeniería. La mayoría de estas herramientas tienen su origen en el grupo de teorías y técnicas matemáticas conocidas como matemáticas numéricas o análisis numérico. La segunda área de la que depende fuertemente la computación científica es la ciencia computacional. Esta área considera aspectos relacionados con lenguajes de programación, sistemas operativos, gestión de grandes cantidades de datos y corrección de programas.

2.2. La Ingeniería Química y las ecuaciones no lineales.

La resolución de problemas en ingeniería requiere por lo general de algún método numérico. Los métodos numéricos son técnicas mediante las cuales es posible formular problemas matemáticos, de tal forma que puedan resolverse utilizando operaciones aritméticas. La característica común de los diferentes tipos de métodos numéricos es que requieren de un buen número de cálculos aritméticos, aspecto que en la actualidad es manejable con el uso de computadoras más eficientes y rápidas (Chapra & Canale, 2007, p. 3).

Antes de la aparición de las computadoras, en ingeniería se utilizaban dos métodos para la solución de problemas:

- Métodos exactos o analíticos. Sus soluciones son útiles y proporcionan una comprensión excelente del comportamiento de algunos sistemas, pero están limitadas a problemas de modelos lineales, de geometría simple y de baja dimensión.
- Métodos gráficos. Sus soluciones toman la forma de gráficos o nomogramas; aunque pueden utilizarse para resolver problemas complejos, los resultados no son muy

precisos y el proceso de solución es tedioso; además las soluciones que proporcionan los métodos gráficos están limitadas a problemas de tres dimensiones o menos.

Según Chapra y Canale (2007, p. 5), el estudio de los métodos numéricos en ingeniería es importante porque:

- Los métodos numéricos permiten manipular sistemas de ecuaciones grandes, manejar no linealidades y resolver geometrías complicadas.
- El conocimiento de la teoría básica de los métodos numéricos permite hacer un uso eficiente de los paquetes de software disponibles comercialmente ("programas enlatados").
- Los métodos numéricos permiten la conversión de las matemáticas superiores en operaciones aritméticas básicas, de esta manera se puede profundizar en los temas que de otra forman resultan difíciles de comprender.

Uno de los retos que afrontan los métodos numéricos es el de determinar estimaciones del error en ausencia del conocimiento de valores verdaderos. Por ejemplo, los métodos numéricos de resolución de ecuaciones algebraicas no lineales requieren de un proceso iterativo para calcular en forma sucesiva mejores aproximaciones y el error a menudo se calcula como la diferencia entre la aproximación previa y la actual (Chapra & Canale, 2007, p. 58).

Dado que los resultados de los métodos numéricos son aproximados, es necesario desarrollar criterios para determinar su grado de confiabilidad. Un criterio aceptable es expresar la confiabilidad en términos de cifras significativas (Chapra & Canale, 2007, p. 55).

Dos de los errores numéricos más comunes son los errores de redondeo y los errores de truncamiento (Chapra & Canale, 2007, p. 54). Los errores de redondeo se producen porque la computadora representa cantidades con un número finito de cifras significativas. Mientras que los errores de truncamiento representan la diferencia entre una formulación matemática exacta de un problema y su aproximación obtenida por un método numérico.

Junto con las limitaciones del sistema numérico de una computadora, la aplicación de manipulaciones aritméticas sobre los números también genera errores de redondeo, sobre

todo en la cancelación de dos números en operaciones de resta. Los métodos numéricos pueden llegar a requerir de una cantidad extremadamente grande de manipulaciones aritméticas. Con frecuencia, estos cálculos son dependientes de los resultados previos y así, aunque el error de redondeo individual sea pequeño, el efecto acumulativo durante el proceso de muchos cálculos puede ser importante (Chapra & Canale, 2007, pp. 70–71).

En este trabajo se describirán los métodos numéricos de resolución de ecuaciones no lineales utilizando las herramientas de computación científica, tales como Scilab y Python, aplicados a problemas de Ingeniería Química. Los problemas que implican el uso de ecuaciones no lineales se relacionan con el valor de una variable o parámetro que satisface una ecuación no lineal. Su solución en proyectos de ingeniería es sumamente valiosa porque con frecuencia resulta imposible despejar de manera analítica los parámetros de las ecuaciones de diseño (Chapra & Canale, 2007, pp. 5–6).

Aunque en este trabajo se elaboraron programas para problemas de cálculo con ecuaciones no lineales en las asignaturas de Termodinámica Química I, Termodinámica Química II, Operaciones Unitarias I y Operaciones Unitarias III, aquí se describirá cómo se incorpora el estudio de ecuaciones no lineales en un grupo de asignaturas del pénsum de Ingeniería Química de la Universidad de El Salvador, tomando como base el plan de estudio de 1998. Estas asignaturas son:

- Química General II (QUR-115)
- Fisicoquímica I (FQR-115)
- Balance de Materia y Energía. (BME-115)
- Fisicoquímica II (FQR-115)
- Ingeniería Económica (IEC-115)
- Operaciones Unitarias I (OPU-115)
- Operaciones Unitarias II (OPU-215)
- Termodinámica Química I (TQI 115)
- Termodinámica Química II (TQI-215)
- Ingeniería de las Reacciones Químicas (IRQ-115)
- Operaciones Unitarias III (OPU-315)
- Proceso de separación y de manejo de sólidos (PSM-115)

2.2.1. *Química General II:*

Equilibrio químico: Equilibrio ácido-base y Equilibrio en soluciones: Kps

En QUR-215 se abordan problemas de equilibrio de reacciones en fase gaseosa, de equilibrios ácidos-base y de equilibrios iónicos de sales con solubilidad muy baja. El objetivo de estos problemas es el cálculo de las concentraciones de equilibrio de las especies químicas presentes en la reacción. En la Tabla 2.1 se especifica el tipo de equilibrio químico a evaluar, los datos que se proporcionan y la expresión de la ecuación de equilibrio utilizada para una reacción como $aA + bB \Leftrightarrow cC + dD$ ó $C_mD_n(s) \Leftrightarrow mC^{+n}(aq) + nD^{-m}(aq)$.

Tabla 2.1. Tipos de equilibrio químico abordados en Química General II

Tipo de equilibrio químico	Datos proporcionados	Expresión de la ecuación de equilibrio utilizada
Equilibrio en reacciones gaseosas a presiones bajas.	Constante de equilibrio K_p y presiones parciales p_A, p_B, p_C y p_D.	$K_p = \frac{p_C^c * p_D^d}{p_A^a * p_B^b}$ $(ec.\ 2.1)$
Equilibrio en reacciones ácido-base en soluciones diluidas.	Constante de equilibrio K_c y concentraciones iniciales [A0], [B0], [C0] y [D0].	$K_c = \frac{[C]^c[D]^d}{[A]^a[B]^b}$ $(ec.2.2)$
Equilibrio iónico de sales con solubilidad muy baja en soluciones diluidas.	Constante de equilibrio K_{ps} y concentraciones iniciales [A0], [B0], [C0] y [D0].	$K_{ps} = [C]^m[D]^n$ $(ec.2.3)$

La característica de este tipo de problemas de equilibrio químico es que a menudo el cambio de concentraciones es pequeño comparado con los valores de las concentraciones iniciales, esto permite manipular una ecuación de equilibrio simplificada.

2.2.2. *Fisicoquímica I:*

Termodinámica. Determinación del Factor de comprensibilidad y ecuación de estado de un gas real.

En FQR-115, se realiza una introducción a la resolución de ecuaciones de estado para gases reales. En QUR-215 se permitía idealizar el comportamiento de los gases. Esto no es posible en condiciones a altas presiones y bajas temperaturas donde las fuerzas intermoleculares y el volumen no nulo ocupado por las propias moléculas se vuelven factores importantes (Levine, 2004, p. 279). Algunas ecuaciones de estado se detallan en la Tabla 2.2, en donde R es la constante universal de los gases, T_c es la temperatura crítica, P_c es la presión crítica, P es la presión, T es la temperatura y V_m es el volumen molar.

Las primeras cuatro ecuaciones de estado son cúbicas en el volumen molar; las dos siguientes ecuaciones son las ecuaciones viriales, las cuales se expresan como una serie de potencias. La ecuación de Benedict-Webb-Rubin, está inspirada en la ecuación virial de Kammerlingh-Onnes. A menudo las ecuaciones cúbicas y la ecuación del virial para el volumen de Kammerling-Onnes (truncada al segundo o tercer términos viriales) se arreglan para poder expresarlas en función de Z, que es el factor de comprensibilidad. Es común utilizar Z y otras cantidades adimensionales porque esto simplifica el número de parámetros a manipular en la ecuación de estado.

Tabla 2.2. Diferentes ecuaciones de estado para gases reales.

Nombre	Ecuación	Descripción de los parámetros
Ecuación de van der Waals	$\left(P+\frac{a}{V_m^2}\right)(V_m-b)=R\,T \quad (ec.\ 2.4)$	$a=\left(\frac{27}{64}\right)\frac{R^2T_c^2}{p_c}$ $b=\left(\frac{1}{8}\right)\frac{R\,T_c}{p_c}$
Ecuación de Redlich-Kwong	$P=\frac{R\,T}{(V_m-b)}-\frac{a}{T^{\frac{1}{2}}V_m\,(V_m+b)} \quad (ec.\ 2.5)$	$a=0.42748\frac{R^2T_c^{2.5}}{p_c}$ $b=0.08664\frac{R\,T_c}{p_c}$
Ecuación de Soave-Redlich-Kwong (SRK)	$P=\frac{R\,T}{(V_m-b)}-\frac{a'\lambda}{V_m\,(V_m+b)} \quad (ec.2.6)$	$a'=0.42748\frac{R^2T_c^2}{p_c}$ $b=0.08664\frac{R\,T_c}{p_c}$ $\lambda=\left[1+\kappa\left(1-T_r^{1/2}\right)\right]^2$ $\kappa=(0.480+1.574\,\omega-0.176\,\omega^2)$

Pasa…

Tabla 2.2. Diferentes ecuaciones de estado para gases reales (continuación).

Nombre	Ecuación	Descripción de los parámetros
Ecuación de Peng-Robinson (PR)	$P = \frac{R\,T}{(V_m - b)} - \frac{a\,\alpha}{V_m\,(V_m + b) + b(V_m + b)} \quad (ec.\ 2.7)$	$a = 0.45724\,\frac{R^2 T_c^2}{p_c}$ $b = 0.07780\,\frac{R\,T_c}{p_c}$ $\alpha = \left[1 + \kappa\left(1 - T_r^{1/2}\right)\right]^2$ $\kappa = (0.37464 + 1.54226\,\omega - 0.29992\,\omega^2)$
Ecuación de Holborn	$P\,V_m = R\,T\,(1 + B'p + C'p^2 + \cdots) \quad (ec.\ 2.8)$	B', C', … deben ser proporcionados.
Ecuación de Kamerlingh-Onnes	$P\,V_m = R\,T\left(1 + \frac{B}{V_m} + \frac{C}{V_m^2} + \cdots\right) \quad (ec.\ 2.9)$	B, C, … deben ser proporcionados.
Ecuación de Benedict-Webb-Rubin	$P\,V_m = R\,T + \frac{\beta}{V_m} + \frac{\sigma}{V_m^2} + \frac{\eta}{V_m^2}\left(1 + \frac{\gamma}{V_m^2}\right) + \frac{\omega}{V_m^5} \quad (ec.\ 2.10)$	$\beta = R\,T\,B_0 - A_0 - \frac{C_0}{T^2}$ $\sigma = b\,R\,T - a$ $\eta = \frac{c}{T^2}\exp\left(-\frac{\gamma}{V_m^2}\right)$ $\omega = a\,\alpha$ $A_0, B_0, C_0, a, b, c, \alpha$ y γ son constantes que deben proporcionarse.

Fuente: Himmelblau y Riggs (2004, p. 461)

2.2.3. *Balance de Materia y Energía:*

Para las ecuaciones de balance global de materia y energía se consideraron a los sistemas abiertos en estado estacionario, con múltiples entradas y salidas; y con S especies químicas.

Balances de Materia en Estado Estable

Caso no reactivo. El balance de materia para un sistema no reactivo es:

$$\sum_{i=1}^{S}\left[\sum_{\substack{corriente\\de\ salida\\j}} F_{j,i} = \sum_{\substack{corriente\\de\ entrada\\k}} F_{k,i}\right] \quad (ec.\ 2.11)$$

donde $F_{j.i}$ es el flujo másico (masa por unidad de tiempo) de la corriente de salida j para la especie i y $F_{k,i}$ es el flujo másico de la corriente de entrada k para la especie i.

Caso de una reacción química. La ecuación (2.11) se modifica para abarcar a un sistema reactivo con una reacción química, agregando al lado derecho el término adicional $r\sum_{i=1}^{S}\nu_i M_i$. La velocidad de reacción r se define como $r = \frac{R_i}{\nu_i}$ y no depende de la especie química i. R_i es la producción o agotamiento molar de alguna especie química i. M_i es el peso molecular de la especie i.

Es común que se proporcione algún tipo de especificación (por ejemplo el porcentaje de conversión del reactivo limitante).

El problema de calcular las corrientes de material saliendo o entrando al sistema consiste en la resolución de ecuaciones algebraicas. Estas ecuaciones son por lo general lineales a excepción de algunas especificaciones (Reklaitis, 1983, p. 49). Un ejemplo de esto último es la especificación de la razón de las composiciones de alguna especie en dos corrientes distintas (Reklaitis, 1983, p. 336).

Balances de energía

Caso no reactivo. El balance de energía para sistemas no reactivos es (Reklaitis, 1983, p. 459):

$$\frac{dQ}{dt}+\frac{dW}{dt}=\sum_{i=1}^{S}\left[\sum_{\substack{corrientes\\de\ salida\\j}} F_{i,j}\,\widehat{H}_i(T_j)-\sum_{\substack{corrientes\\de\ entrada\\k}} F_{i,k}\,\widehat{H}_i(T_k)\right] \quad (ec.\ 2.12)$$

donde $\widehat{H}_i$ es una entalpía específica de la especie i en la corriente de salida j a la temperatura T_j o de la corriente de entrada k a la temperatura T_k, $\frac{dQ}{dt}$ es la rapidez de transferencia de calor y $\frac{dW}{dt}$ es la rapidez del trabajo ejercida o aplicada sobre el sistema. Se considera que las composiciones másicas, la presión del sistema y la distribución de fases se han determinado, que los cambios de energía potencial y cinética son reducidos con respecto a los términos de calor y trabajo; y que el comportamiento de las corrientes se aproxima al de mezclas ideales.

Caso de una reacción química. El balance de energía para un sistema reactivo con una reacción química será (Reklaitis, 1983, p. 489):

$$\frac{dQ}{dt}=r\Delta H_{Rxn}+\sum_{i=1}^{S}\left(\sum_{\substack{corriente\\de\ salida\\j}} F_{i,j}\left(\widehat{H}_{i,2}-\widehat{H}_{i,ref}\right)-\sum_{\substack{corriente\\de\ entrada\\k}} F_{i,k}\left(\widehat{H}_{i,2}-\widehat{H}_{i,ref}\right)\right) (ec.\ 2.13)$$

donde el calor de reacción ΔH_{Rxn}, y la entalpía específica de referencia de la especie i, $\widehat{H}_{i,ref}$, se definen a una temperatura, presión y fase de referencia determinadas.

Las ecuaciones (2.12) y (2.13) tratan básicamente de la primera ley de la termodinámica y son no lineales con respecto a los flujos de las especies químicas y las temperaturas. Si se desconoce alguna de estas variables, la solución requerirá un proceso iterativo (Reklaitis, 1983, p. 460).

En la asignatura de BME-115, las corrientes de salida y entrada de un sistema en problemas de reacciones de combustión se trabajan a presión atmosférica y en fase gaseosa o fase líquida, y por lo general no será necesario realizar correcciones para la fase y las presiones.

La obtención de datos de entalpía para problemas de humidificación y de mezclado de soluciones se simplifica en gran medida debido al uso de gráficas y tablas de datos termodinámicos, donde la interpolación alrededor de dos valores cercanos de entalpía produce la temperatura buscada. Además en la mayoría de casos se considera que gases y soluciones líquidas tienen comportamiento ideal.

2.2.4. *Fisicoquímica II:*

Soluciones ideales

Según Moore (1986, p. 182) la cantidad observable más importante de la teoría de soluciones es la presión de vapor de la solución. Esta presión de vapor parcial es una medida de la tendencia de las especies a escapar de la solución hacia la fase vapor. Dicha tendencia se relaciona de manera directa con su potencial químico dentro de la solución (recuérdese el tema de equilibro material de FQR-115).

El concepto de solución ideal sirve como un punto de referencia para las soluciones reales, de la misma manera que el concepto de gas ideal lo es para gases reales. El comportamiento de una solución ideal[1] para el componente i es descrito por la ecuación de Raoult (Smith, Van Ness, & Abbot, 2007, pp. 350–351):

$$P_i = x_i P_i^{sat}(T) \quad (ec.\ 2.15)$$

donde P_i es la presión de vapor parcial del componente i, x_i es la fracción molar del componente i en la fase líquida y P_i^{sat} es por lo general una función no lineal de la temperatura T y representa la presión de vapor puro en saturación del componente i.

Además se considera como ideal el comportamiento del vapor:

$$y_i P = x_i P_i^{sat} \quad (ec.\ 2.16)$$

donde y_i es la composición molar de la fase vapor y P es la presión total del sistema.

[1] Levine (2004, p. 2004) hace una diferenciación entre disolución ideal (pág. 319) y disolución idealmente diluidas (pág. 327).

El uso de la ley de Raoult para calcular la temperatura de los puntos de rocío y de burbuja a una presión y composiciones conocidas implica un proceso iterativo de solución. La Tabla 2.3 presenta las ecuaciones a utilizar en estos problemas de cálculo:

Tabla 2.3. La ley de Raoult aplicada a problemas de cálculo de punto de burbuja T y punto de rocío T.

Tipo de Problema	Datos proporcionados	Datos a calcular	Forma de la ley de Raoult.
Punto de burbuja T	$\{x_i\}$ P	$\{y_i\}$ T	$P = \sum_i x_i P_i^{sat} \quad (ec.\ 2.17)$
Punto de rocío T	$\{y_i\}$ P	$\{x_i\}$ T	$P = \frac{1}{\sum_i y_i / P_i^{sat}} \quad (ec.\ 2.18)$

Para el punto de burbuja T, la ecuación (2.16) se suma sobre todas las especies para eliminar y_i (considerando que $\sum_i y_i = 1$), luego se obtiene la ec. (2.17): $\sum_i y_i = 1 = \sum_i (x_i P_i^{sat})/P$. Un procedimiento similar se lleva a cabo para obtener la ecuación (2.18). Como se observa, estas ecuaciones son no lineales en la temperatura T.

Desviación del comportamiento ideal. Soluciones reales

Caso de la ley de Raoult modificada. Las soluciones reales pueden llegar a presentar fuertes desviaciones del comportamiento descrito en la ecuación (2.16)[2]. Para presiones de bajas a moderadas, una mejor opción para el cálculo de propiedades en equilibrio líquido vapor es la ley de Raoult modificada (Smith et al., 2007, pp. 358–359):

$$y_i P = x_i \gamma_i P_i^{sat} \quad (ec.\ 2.19)$$

donde γ_i es el coeficiente de actividad. A presiones bajas y moderadas puede considerarse que el coeficiente de actividad es una función únicamente de la temperatura y de la

[2] Se puede consultar al respecto en Levine (2004, Capítulo 10) y Moore (1986, Capítulo 10).

composición de la fase líquida. El coeficiente de actividad se utiliza para medir el grado de divergencia del comportamiento de la sustancia i con respecto al comportamiento de la solución ideal (Levine, 2004, p. 343). Al igual que la ecuación (2.16), la ecuación (2.19) se manipula de dos maneras considerando ya sea a $\sum_i y_i = 1$ o a $\sum_i x_i = 1$, dependiendo del vector de composiciones que se desconozca. Esta situación se describe en la Tabla 2.4.

Tabla 2.4. La ley de Raoult modificada aplicada a problemas de cálculo de punto de rocío P, punto de burbuja T y punto de rocío T.

Tipo de Problema	Datos proporcionados	Datos a calcular	Forma de la ley de Raoult.
Punto de rocío P	$\{y_i\}$ T	$\{x_i\}$, $\{\gamma_i\}$ y P	$P = \frac{1}{\sum_i y_i/\gamma_i P_i^{sat}} \quad (ec.\ 2.20)$
Punto de burbuja T	$\{x_i\}$ P	$\{y_i\}$, $\{\gamma_i\}$ y T	$P = \sum_i x_i \gamma_i P_i^{sat} \quad (ec. 2.21)$
Punto de rocío T	$\{y_i\}$ P	$\{x_i\}$, $\{\gamma_i\}$ y T	$P = \frac{1}{\sum_i y_i/\gamma_i P_i^{sat}} \quad (ec. 2.22)$

A bajas presiones y a temperatura constante, γ_i depende únicamente de las composiciones molares de la especies en solución líquida, mientras que P_i^{sat} se calcula típicamente a través de ecuaciones o correlaciones que, por lo general, son no lineales con respecto a T. Por lo tanto, la resolución de estas ecuaciones implica un proceso iterativo.

2.2.5. *Operaciones Unitarias I:*

Cálculos para la conducción de fluidos incompresibles

Problema de tuberías simples

Los problemas de tuberías simples se refieren a tuberías en las cuales la fricción de superficie es la única pérdida. Las tuberías pueden formar cualquier ángulo con la horizontal. Seis variables entran en el problema: la tasa de flujo volumétrico Q, la longitud de la tubería L, el diámetro de la tubería D, la pérdida de cabeza por fricción de superficie $h_{L,s}$, la viscosidad cinemática ν y la rugosidad ϵ. En general, L, ν y ϵ son dados o pueden determinarse. La Tabla 2.5 muestra la clasificación de los problemas de tuberías simples (Mott, 2006, p. 294).

Tabla 2.5. Tipos de problemas de tuberías simples

Tipo	Dado	Incógnita
I	Q, L, D, ν, ϵ	$h_{L,s}$
II	$h_{L,s}$, L, D, ν, ϵ	Q
III	$h_{L,s}$, Q, L, ν, ϵ	D

Los tres tipos de problemas se resuelven utilizando la ecuación de Darcy-Weisbach (ec. 2.23), la ecuación de continuidad[3] (ec. 2.24) y el diagrama de Moody (o alguna correlación para el coeficiente de fricción). En principio los 3 tipos de problema implican un proceso iterativo de solución, sin embargo Streeter, Wylie & Bedford (2000, pp. 294–297) proporcionan correlaciones empíricas con alto grado de exactitud para la evaluación directa de las incógnitas de los problemas tipo I, II y III en tuberías horizontales.

[3] Se consideran únicamente fluidos incompresibles en flujo permanente o estacionario.

$$h_{f,s} = f\,\frac{L}{D}\,\frac{v^2}{2g}$$ (ec. 2.23)

$$A_1 v_1 = A_2 v_2$$ (ec. 2.24)

Para tuberías no horizontales, el cálculo de las presiones o de las elevaciones se realiza a través de la ecuación de Bernoulli[4] (ec. 2.25) corregida para la fricción y evaluada en el punto 1 y 2:

$$\frac{V_1^2}{2\,g} + \frac{p_1}{\gamma} + z_1 = \frac{V_2^2}{2\,g} + \frac{p_2}{\gamma} + z_2 + h_L \quad (ec.\ 2.25)$$

Problemas de sistemas de tuberías

Los sistemas de tuberías pueden ser tuberías en serie, tuberías en paralelo, tuberías ramificadas y sistema de tuberías en forma de malla. Estos sistemas de tuberías se describirán a continuación.

Tuberías en serie. Es el caso en el que dos tuberías de tamaños o rugosidades diferentes se conectan de tal manera que el mismo caudal fluye por todas las tuberías y las pérdidas de cabeza se acumulan. Para estos arreglos de tuberías se supone que las longitudes, la viscosidad cinemática ν y las rugosidades de las tuberías son dados o pueden calcularse. Se consideran tres tipos de problemas:

Tipo I: El caudal y los diámetros de las tuberías están determinados. El número de Reynolds se calcula fácilmente y los coeficientes de fricción se pueden determinar a partir del diagrama de Moody. Luego la carga o pérdidas del sistema pueden evaluarse mediante sustitución directa.

Tipo II: Se proporciona la carga y los diámetros de las tuberías. Las incógnitas son las velocidades y los coeficientes de fricción de cada tubería, para las que se emplea un método

[4] En la ecuación de Bernoulli, el término p/γ es la cabeza o carga de presión, el término $V_2^2/2\,g$ es la cabeza de velocidad, z es la elevación y h_L representa las pérdidas de cabeza por fricción, pudiendo ser de dos tipos: pérdidas de cabeza por fricción superficial $h_{L,s}$ y pérdidas de cabeza por fricción de forma $h_{L,f}$.

iterativo puesto que la ecuación de Darcy y la ecuación para el cálculo de pérdidas menores son no lineales con respecto a las velocidades.

Tipo III: Se proporciona la carga y el caudal. Las incógnitas son los diámetros de las tuberías y los coeficientes de fricción en cada tubería. Se requiere un método iterativo puesto que la ecuación de Darcy y la ecuación para el cálculo de pérdidas menores son no lineales con respecto a los diámetros.

Tuberías en paralelo. Este sistema consiste en la combinación de dos o más tuberías conectadas, de tal manera que el caudal se divide entre las tuberías y luego se une nuevamente. En las tuberías en paralelo las pérdidas de cabeza son las mismas en cada una de las líneas y los caudales son acumulables (Streeter et al., 2000, pp. 553–554). Para estos arreglos de tuberías se supone que la viscosidad cinemática ν, las rugosidades, longitudes y diámetros de las tuberías son dados o pueden calcularse. Esta clasificación aplica para dos o tres tuberías en paralelo. Ocurren dos tipos de problemas:

Tipo I. Con elevaciones conocidas, se requiere determinar el caudal total Q. Este tipo de problema se puede tratar como un problema de tubería simple para la obtención del caudal en cada tubería. Estos caudales se suman para determinar el caudal total.

Tipo II. Con el caudal total Q conocido, se desea determinar la distribución del caudal y las pérdidas de cabeza. Este segundo tipo de problema es más complejo, ya que no se conoce ni la pérdida de cabeza ni el caudal para cualquiera de las tuberías. Se requiere de un método iterativo para satisfacer la ecuación de Darcy-Weisbach, la ecuación para el cálculo de pérdidas menores y la ecuación de continuidad para cada tubería.

Tuberías ramificadas. Para este tipo de problema, el caudal en cada tubería se puede determinar si se conocen las elevaciones en los depósitos, los diámetros y tipos de tuberías y las propiedades del fluido. Se deben satisfacer la ecuación de Darcy-Weisbach, la ecuación para el cálculo de pérdidas menores y la ecuación de continuidad para cada tubería.

Sistemas de tuberías en forma de malla. Cuando un sistema de flujo en tuberías tiene cuatro ramas o más, se le denomina red. Las redes son indeterminadas porque hay más factores desconocidos que ecuaciones independientes que los relacionen. Un método de resolución

de este tipo de sistemas complejos es el método de Hardy Cross (Streeter et al., 2000, p. 559). De acuerdo con esto el sistema de tuberías de tres ramas pertenece a esta categoría, pero Streeter et al. (2000, pp. 553–555) y Barderas Valiente (2002, pp. 283–284) tratan dicho problema como un sistema de tuberías en paralelo.

2.2.6. *Termodinámica Química I:*

Propiedades Volumétricas de los Fluidos Puros.

Comportamiento de vapores y gases. Uso del factor acéntrico, ecuaciones viriales. En Fisicoquímica I se introdujo el estudio de las ecuaciones de estado cúbicas para gases reales, pero estas ecuaciones no se limitan a gases. Las ecuaciones cúbicas de estado permiten obtener un mínimo de tres respuestas para el volumen.

En el caso en el que las tres raíces son reales se obtienen los volúmenes de líquidos saturados no polares (primera raíz real) y los volúmenes de los gases en condiciones de saturación (tercera raíz real); la segunda raíz real carece de significado físico. La obtención de una sola raíz corresponde ya sea al punto crítico o a un estado de sobrecalentamiento (Levine, 2004, pp. 285–288).

Smith et al. (2007, p. 293) reformula las ecuaciones de estado cúbica dadas en la Tabla 2.2 en base a la siguiente ecuación:

$$P = \frac{R\,T}{V_m - b} - \frac{a(T)}{(V_m + \epsilon\, b)(V_m + \sigma\, b)} \quad (ec.\ 2.26)$$

Y la convierte en:

$$Z = \frac{Z}{Z - \beta} - \frac{q\,Z\,\beta}{(Z + \epsilon\,\beta)(Z + \sigma\,\beta)} \quad (ec.\ 2.27)$$

Donde β, Ω, Ψ son parámetros adimensionales que permiten una manipulación más sencilla de la ecuación dado que ella contiene una menor cantidad de parámetros y variables a evaluar.

Segunda ley de la termodinámica.

La segunda ley de la termodinámica trata de la entropía, una variable termodinámica que permite tomar en cuenta el grado de irreversibilidad de los procesos. La entropía al contrario que la entalpía no se conserva. El cambio de entropía total para sistemas abiertos en estado estacionario debe ser (Smith et al., 2007, p. 177):

$$\sum_{i=1}^{S}\left[\sum_{\substack{corrientes\\ de\ salida\\ j}} F_{i,j}\,\hat{S}_i(T_j) - \sum_{\substack{corrientes\\ de\ entrada\\ k}} F_{i,k}\,\hat{S}_i(T_k)\right] - \sum_j \frac{1}{T_{\sigma,j}}\frac{dQ_j}{dt} = \frac{dS_G}{dt} \geq 0 \quad (ec.\ 2.28)$$

donde $\hat{S}_i$ es la entropía específica de la especie i de la corriente de salida j a la temperatura T_j o de la corriente de entrada k a T_k, $\frac{dQ_j}{dt}$ es la rapidez de transferencia de calor con respecto a una parte específica de la superficie de control[5] asociada con $T_{\sigma,j}$ (el símbolo σ, j denota una temperatura en los alrededores) y $\frac{dS_G}{dt}$.es la relación de generación de entropía. La ecuación anterior establece que el cambio de entropía total asociado a cualquier proceso deber ser positivo, excepto cuando se trata de procesos reversibles, donde el cambio de entropía es cero.

La entropía al igual que la entalpía es una función no lineal de la temperatura. Para un sistema cerrado, con un estado inicial a temperatura T_0 y presión P_0 y estado final con una temperatura T_1 y presión P_1, el cambio de entropía asociado se describe mediante la siguiente ecuación (Smith et al., 2007, p. 171):

$$\Delta S_{total} = \langle C_P^{ig}\rangle_S \ln\frac{T_2}{T_1} - R\ln\frac{P_2}{P_1} \quad (ec.\ 2.29)$$

Cuando se desarrolla un proceso reversible, $\Delta S_{total} = 0$, dadas T_1, P_1 y P_2, es necesario implementar un procedimiento iterativo para encontrar T_2 porque $\langle C_P^{ig}\rangle_S$ es no lineal con respecto a la temperatura. $\langle C_P^{ig}\rangle_S$ se define en Smith et al. (2007, p. 172).

[5] Véase Smith et al. (2007, p. 46) sobre la descripción de una superficie de control para un proceso.

Los cálculos de entalpías y entropías de los procesos vistos en TQI-115, implicarían el uso de ecuaciones no lineales pero esto se evita a través del uso de gráficas y tablas de datos termodinámicos.

2.2.7. *Termodinámica Química II:*

Propiedades termodinámicas de los fluidos

El cambio de entalpía y de entropía de un sistema cerrado y en estado estacionario que cambia de un estado real inicial 1 a un estado real final 2, es descrito por (Smith et al., 2007, p. 234):

$$\Delta H_{total} = \Delta H_{ideal} + H_2^R - H_1^R \quad (ec. 2.30)$$

$$\Delta S_{total} = \Delta S_{ideal} + S_2^R - S_1^R \quad (ec.\ 2.31)$$

donde H_2^R y H_1^R son las entalpías residuales y S_2^R y S_1^R son las entropías residuales. ΔH_{ideal} se define por Smith et al. (2007, p. 130) y ΔS_{ideal} se define por Smith et al. (2007, p. 170). Cuando se desconoce la temperatura, es necesario utilizar un método iterativo en las ecuaciones (2.28) y (2.29).

Equilibrio de fases en sistemas multi-componentes.

Se toma como base la formulación Gamma-Phi del Equilibrio vapor-líquido (Smith et al., 2007, pp. 145–146):

$$y_i \Phi_i P = x_i \gamma_i P_i^{sat} \quad (ec.\ 2.32)$$

Φ_i se define de la siguiente manera:

$$\Phi_i = \frac{\hat{\phi}_i}{\phi_i^{sat}} \quad (ec.\ 2.33)$$

donde $\hat{\phi}_i$ es el coeficiente de fugacidad de la especie i en solución y ϕ_i^{sat} es el coeficiente de fugacidad de la especie pura i, definida a la temperatura de saturación de la solución.
La ecuación (2.32) se suma sobre todas las especies en la fase vapor ($\sum_i y_i = 1$) o en la fase líquida ($\sum_i x_i = 1$), dependiendo del conjunto de composiciones que se desconozca. Esta situación se describe en la Tabla 2.6 (Smith et al., 2007, p. 547).

Tabla 2.6. La ley de Raoult modificada aplicada a problemas de cálculo de punto de burbuja T y punto de rocío T.

Tipo de Problema	Datos proporcionados	Datos a calcular	Forma de la formulación γ-Φ.
Punto de burbuja P	$\{x_i\}$ P	$\{y_i\}$, $\{\gamma_i\}$, $\{\Phi_i\}$ y P	$P = \sum_i \frac{x_i \gamma_i P_i^{sat}}{\Phi_i}$ $(ec.\ 2.34)$
Punto de rocío P	$\{y_i\}$ T	$\{x_i\}$, $\{\gamma_i\}$, $\{\Phi_i\}$ y P	$P = \frac{1}{\sum_i y_i \Phi_i / \gamma_i P_i^{sat}}$ $(ec. 2.35)$
Punto de burbuja T	$\{x_i\}$ P	$\{y_i\}$, $\{\gamma_i\}$, $\{\Phi_i\}$ y T	$P = \sum_i \frac{x_i \gamma_i P_i^{sat}}{\Phi_i}$ $(ec.\ 2.36)$
Punto de rocío T	$\{y_i\}$ P	$\{x_i\}$, $\{\gamma_i\}$, $\{\Phi_i\}$ y T	$P = \frac{1}{\sum_i y_i \Phi_i / \gamma_i P_i^{sat}}$ $(ec.\ 2.37)$

Φ depende de la temperatura, la presión y las composiciones molares y será necesario utilizar un método numérico para calcular la temperatura o la presión del sistema.

Cálculos de vaporización instantánea. El cálculo de las moles de vapor V, moles de líquido L y composiciones de las fases vapor $\{y_i\}$ y líquido $\{x_i\}$ que constituyen un sistema no reactivo de dos fases en equilibrio a T, P y *composición global* $\{z_i\}$ fijas se obtiene con la siguiente ecuación:

$$\sum_i \frac{z_i K_i}{1 + V(K_i - 1)} = 1 \qquad (ec.\ 2.38)$$

donde K_i se define como:

$$K_i = \frac{y_i}{x_i} \quad (ec.\ 2.39)$$

K_i es un coeficiente de distribución y mide la tendencia de una especie química i para repartirse de preferencia entre las fases de vapor y de líquido. K_i se calcula dependiendo de

la ecuación utilizada para cálculos EVL. Esto se muestra en la Tabla 2.7 (Smith et al., 2007, pp. 364, 552).

Tabla 2.7. Formulación de K_i para distintas ecuaciones para cálculos EVL.

Ecuación para cálculos EVL	Formulación de K_i
Ley de Raoult	$K_i = \frac{P_i^{sat}}{P} \quad (ec.\ 2.40)$
Ley de Raoult modificada	$K_i = \frac{\gamma_i P_i^{sat}}{P} \quad (ec. 2.41)$
Formulación gamma-phi	$K_i = \frac{\gamma_i P_i^{sat}}{\Phi_i P} \quad (ec.\ 2.42)$

Por lo tanto el cálculo de V requerirá un cálculo iterativo para:

- Únicamente para V en la ecuación (2.40);
- Para V, $\{K_i\}$, $\{\gamma_i\}$ en la ecuación (2.41);
- Para V, $\{K_i\}$, $\{\gamma_i\}$ y $\{\Phi_i\}$ en la ecuación (2.42).

Equilibrio de reacciones químicas.

Reacciones en fase gas. La ecuación de equilibrio para reacciones en fase gas se define así:

$$\prod_i \left(y_i \hat{\phi}_i\right)^{\nu_i} = \left(\frac{P}{P^0}\right)^{-\nu} K \quad (ec.\ 2.43)$$

donde y_i es la composición molar de la especie i, $\hat{\phi}_i$ es el coeficiente de fugacidad en solución de i, P es la presión del sistema, P^0 es la presión en el estado estándar y la constante de reacción K se calcula tomando en cuenta el estado estándar de las especies químicas. Además $\nu = \sum_i \nu_i$, donde ν_i es el coeficiente estequiométrico de i. Este modelo puede simplificarse si se supone que la mezcla en equilibrio se comporta como una solución ideal, o un gas ideal (a presiones bajas y temperaturas suficientemente altas), es decir $\{\hat{\phi}_i\}$ son igual a la unidad .

Por lo general, para las ecuaciones de equilibrio se suele proporcionar el valor de la presión, quedando como incógnitas la temperatura y las composiciones molares de equilibrio. Si también la temperatura se proporciona sólo será necesario determinar las composiciones de equilibrio de las especies químicas a través de un método iterativo.

2.2.8. *Operaciones Unitarias II*

Cálculos de los coeficientes de película para procesos de transferencia de convección natural y convección forzada.

Se cuenta con diversas correlaciones para el cálculo de los coeficientes de película, que dependen de la geometría y orientación del sistema. Para el caso de fluidos que circulan a través de tubos, ejemplos de correlaciones a ocupar son: las correlaciones de Dittus y Boelter, de Colburn y de Seider y Tate (Welty, Wicks, & Wilson, 2008, pp. 444–445). La utilización de estas correlaciones requiere tener datos de temperatura de entrada, temperatura de salida, temperatura de película y/o temperatura media aparente del fluido circulante. En el caso en el que falten más de una de dichas propiedades, se deberá ocupar un proceso iterativo de solución (de prueba y error) para su estimación.

Diseño de intercambiadores de calor

De acuerdo con Kern (1950, Capítulos 11–12), el diseño de intercambiadores de calor abarca la determinación de las siguientes características:

- Número de tubos
- Número de pasos a utilizar en la coraza y en los tubos.
- Espacio mínimo entre cada deflector.
- Caída de presión del lado de los tubos y de la coraza.
- Coeficiente total de transferencia de calor del intercambiador limpio (U_c) e incrustado (U_D) y la resistencia térmica de la incrustación (R_d). Kern estima U_D y obtiene los elementos anteriores; entonces calcula U_D y compara los valores de U_D.

2.2.9. *Operaciones Unitarias III:*

Humidificación

El método gráfico de Mickley permite obtener una curva de la temperatura global de la fase gaseosa (T_v) contra la entalpía del gas (H_v), ocupando como base la siguiente ecuación (Foust, Wenzel, Clump, Maus, & Andersen, 2006, pp. 450, 436, 443):

$$\frac{dT_v}{dH_v} = \frac{T_i - T_v}{H_i - H_v} \quad (ec. 2.44)$$

Donde las propiedades en la interfase son las de T_i y H_i, y las propiedades de la fase gaseosa son T_v y H_v.

También se ocupan las siguientes ecuaciones auxiliares:

- Relación de Lewis:

$$c_p \approx \frac{h_c}{k_Y} \quad (ec. 2.45)$$

- Ecuación de diseño integrada:

$$-\frac{h_L a}{k_Y a} = \frac{H_v - H_i}{T_L - T_i} \quad (ec. 2.46)$$

Dependiendo de las condiciones proporcionadas, con este método es posible calcular ya sea la temperatura de salida de la fase gaseosa o las constantes de velocidad $k_Y a$, $h_c a$ y $h_L a$.

Destilación

Destilación instantánea

Según Treybal (1986, p. 401), "la evaporación instantánea o evaporación en el equilibrio es una operación de una sola etapa en donde se evapora parcialmente una mezcla líquida, se permite que el vapor alcance el equilibrio con el líquido residual y se separan y eliminan del aparato las fases vapor y líquido resultante".

Estos cálculos se llevan a cabo utilizando las ecuaciones descritas para los cálculos vaporización instantánea en el apartado 2.2.8 de TQI-215.

Rectificación continua. Mezclas binarias

De acuerdo con Treybal (1986, p. 410), "la rectificación continua o fraccionamiento, es una operación a contracorriente a varias etapas". Para mezclas binarias, se suele ocupar dos métodos:

- El método de Ponchon-Savarit, que es un método riguroso y requiere de datos detallados de entalpía.
- El método de McCabe-Thiele, el cual es un método simplificado que sólo requiere de datos de equilibrios de concentración.

Ambos métodos son gráficos que permiten obtener el número de etapas a ocupar en la torre de destilación.

En los métodos de Mickley, Ponchon-Savarit y de McCabe-Thiele, no se ocupan los métodos para la resolución de ecuaciones no lineales como el método de Newton.

2.2.10. *Ingeniería de las Reacciones Químicas*

Determinación de la mejor combinación de un grupo de reactores de mezcla completa (RMC) en serie y de distintos tamaños

Levenspiel (2004, pp. 131–134) y Hill (1977, pp. 281–285) describen un método gráfico para el cálculo de los tamaños de un grupo de reactores de mezcla completa dispuestos de tal manera que, a una conversión dada, minimicen el volumen total del sistema.

La ecuación a utilizar es:

$$-\frac{1}{\tau_i} = \frac{(-r)_i}{C_i - C_{i-1}} \quad (ec. 2.47)$$

donde τ_i es el espacio-tiempo del reactivo i en el reactor, $(-r)_i$ es el negativo de la velocidad de reacción, C_i es la concentración del efluente de salida del i-ésimo reactor; y C_{i-1} es la concentración es la concentración del (i-1)-ésimo reactor.

Hill (1977, pp. 286–287) también explica la versión algebraica del problema, tomando en cuenta que deben conocerse la forma funcional y las constantes involucradas en la expresión

de la velocidad de reacción, siendo posible iterar con respecto a la conversión de uno de los reactores de mezcla completa.

Operación no isotérmica de reactores ideales: reactor de mezcla completa (RMC) y reactor de flujo pistón (RFP)

Este tipo de operaciones no isotérmicas se ven ejemplificadas en la operación adiabática de un grupo de RMC en serie y la operación no isotérmica y no adiabática de un RFP. Hill (1977, pp. 351–354, 361–362) desarrolla estos casos considerando las siguientes ecuaciones:

RMC:

$$\dot{Q} = U\,A\,(T_m - T)\;(ec.2.48)$$

$$\dot{Q} = \frac{F_{A0}\,(x_{A\,salida} - x_{entrada})}{-\nu_A}\,\Delta H_{R\,a\,T0} + \sum_i F_i \int_{T0}^{T_{salida}} \bar{C}_{p,i}\,dT \quad (ec.2.49)$$

$$k = k_0 e^{-\frac{E}{R\,T}} \quad (ec.2.50)$$

$$\tau = \frac{C_{A0} x_A}{-r_A} \quad (ec.2.51)$$

La ecuación (2.48) representa la velocidad a la que es suministrado el calor ($\dot{Q}$), donde U es el coeficiente de transferencia de calor, A es el área a través de la cual ocurre la transferencia de calor y en la cual se basa U, T_m es la temperatura de la fuente de calor y T es la temperatura del contenido del reactor.

La ecuación (2.49) representa el balance de energía en una operación de estado estacionario, donde $\dot{Q}$ es la velocidad a la que es suministrado el calor, F_{A0} es la velocidad molar inicial del reactivo A, $x_{A\,salida}$ es la conversión de A a la salida del reactor, $x_{A\,entrada}$ es la conversión de A a la entrada del reactor, $\Delta H_{R\,a\,T0}$ es el cambio de entalpía de reacción a la temperatura de entrada T0, por unidad de grado de avance ε; ν_A es el coeficiente estequiométrico, F_i representa la velocidad molar de los productos i y $\bar{C}_{p,i}$ es el calor específico promedio por unidad de masa del producto i.

La ecuación (2.50) representa la ecuación de la constante de la velocidad de reacción k, donde E es la energía de activación de la reacción, T es la temperatura y R es la constante de los gases ideales.

La ecuación (2.51) representa la ecuación de funcionamiento del RMC, donde τ es el espacio-tiempo (o tiempo espacial), C_{A0} es la concentración inicial de A en la corriente de entrada al reactor y r_A es la velocidad de reacción en base a A.

RFP:

$$Q = \int_{x_{A\,entrada}}^{x_{A\,salida}} U\,(T_m - T)\,\frac{4}{D} F_{A0}\,\frac{dx_A}{(-r_A)} \quad (ec.\,2.52)$$

$$Q = \sum_i F_i \int_{T0}^{T_{salida}} \bar{C}_{p,i}\,dT \; - \frac{F_{A0}\,(x_{A\,salida} - x_{A\,entrada})}{\nu_A}\,\Delta H_{R\,a\,T0} \quad (ec.\,2.53)$$

$$k = k_0 e^{-\frac{E}{R\,T}} \quad (ec.\,2.54)$$

$$\tau = C_{A0} \int_{x_{A\,entrada}}^{x_{A\,salida}} \frac{dx_A}{(-r_A)} \quad (ec.\,2.55)$$

La nomenclatura para el RFP es la misma que para el RMC. La ecuación (2.52) representa el calor total suministrado, donde D es el diámetro del RFP.

La ecuación (2.53) es el balance de energía para una operación en estado estacionario, donde Q es el calor total suministrado; la ecuación (2.54) es la ecuación de la constante de la velocidad de reacción; y la ecuación (2.55) representa la ecuación de funcionamiento del reactor.

2.2.11. *Procesos de separación y de manejo de sólidos.*

Operaciones de Sedimentación

Dentro de las operaciones de sedimentación, encontramos a las técnicas de sedimentación por gravedad, que implican la separación de las partículas de un fluido por la acción de gravedad.

A medida que la partícula cae, su velocidad aumenta y seguirá incrementándose hasta alcanzar una velocidad constante o "velocidad terminal", que es la velocidad en la que las fuerzas de aceleración y resistencia son iguales.

Para el caso de partículas esféricas, la velocidad terminal para cualquier régimen de flujo se calcula con (McCabe, Smith, & Harriott, 2007, pp. 182–183):

$$v_t = \sqrt{\frac{4\left(\rho_p - \rho\right) g\, D_p}{3\, C_D\, \rho}} \quad (ec.\ 2.56)$$

donde ρ_p es la densidad de la partícula, ρ es la densidad del fluido, g es la aceleración de la gravedad, D_p es el diámetro de partícula para partículas esféricas o el diámetro característico para partículas no esféricas y C_D es el coeficiente de rozamiento. El C_D se puede leer en las gráficas proporcionadas por McCabe et al. (2007, pp. 169–180). Dichas gráficas muestran al C_D como una función del número de Reynolds N_{Re}, el cual es función de la velocidad. Por lo tanto será necesario utilizar un método iterativo gráfico para calcular la velocidad terminal.

El método de solución se simplifica para casos en los que:

$N_{Re} < 1$

$$v_t = \frac{g\, D_p^2\left(\rho_p - \rho\right)}{18\,\mu} \quad (ec.\ 2.57)$$

donde μ es la viscosidad del líquido.

$1000 < N_{Re} < 200000$

$$v_t = 1.75\sqrt{\frac{g\, D_p\left(\rho_p - \rho\right)}{\rho}} \quad (ec. 2.58)$$

Otra opción es utilizar una correlación como la de Schiller y Nauman aplicable a partículas esféricas cuando $N_{Re} < 800$ (Ahuja, 2010, p. 24).

2.3. Métodos de solución de las ecuaciones no lineales

Se describirán a continuación los siguientes métodos para la solución de ecuaciones no lineales: método de la falsa posición, método de la falsa posición modificada, método de iteración de punto fijo, método de Newton y método de la secante.

2.3.1. *Método de la falsa posición*[6]

Este método consiste en "unir" $f(x_l)$ y $f(x_u)$ con una línea recta (véase Figura 2.1). La intersección de esta línea con el eje de las x representa una mejor aproximación de la raíz. El hecho de que se reemplace la curva por una línea recta da una "falsa posición" de la raíz; de aquí el nombre de método de la falsa posición, o en latín, regula falsi. También se le conoce como método de interpolación lineal.

La ecuación del método de Regula – Falsi es:

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} \quad (ec.\ 2.59)$$

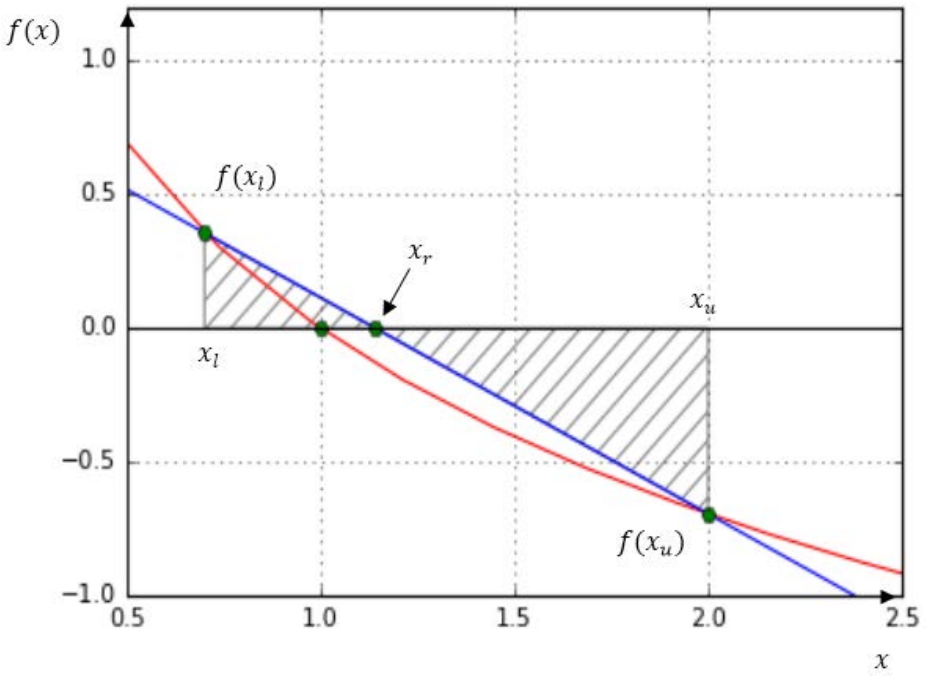
El valor de x_r calculado con la ecuación (2.59) reemplazará, después a cualquiera de los dos valores iniciales, x_l o x_u. De esta manera, los valores x_l y x_u siempre encierran la verdadera raíz.

Criterio de paro y estimaciones de errores

$$\varepsilon_a = \left| \frac{x_r^{nuevo} - x_r^{anterior}}{x_r^{nuevo}} \right| 100\% \quad (ec.\ 2.60)$$

Donde x_r^{nuevo} es la raíz en la iteración actual y $x_r^{anterior}$ es la raíz en la iteración anterior. Se utiliza el valor absoluto, ya que por lo general importa sólo la magnitud de ε_a sin considerar el signo. Cuando ε_a es menor que un valor previamente fijado ε_s, el cálculo termina.

[6] Esta sección se basa en Chapra y Canale (2007, pp. 131–138).



Adaptado de Chapra y Canale (Chapra & Canale, 2007, p. 133).

Figura 2.1. Una representación gráfica del método de falsa posición. Los triángulos semejantes utilizados para deducir la fórmula para el método están sombreados.

Una importante desventaja del método de la falsa posición es su unilateralidad. Es decir, conforme se avanza en las iteraciones, uno de los extremos del intervalo tiende a permanecer fijo. Esto puede llevar a una mala convergencia, especialmente en funciones con una curvatura importante.

Una forma de disminuir la naturaleza unilateral de la falsa posición consiste en obtener un algoritmo que detecte cuando “se estanca” uno de los límites del intervalo. Si ocurre esto, se divide a la mitad el valor de la función en el punto de “estancamiento”. A este método se le llama método de la falsa posición modificada.

2.3.2. *Método de la iteración de punto fijo para una ecuación no lineal.*[7]

El método de iteración de punto fijo es un método abierto porque requiere únicamente de un solo valor de inicio. Este método algunas veces divergirá o se alejará de la raíz verdadera a medida que se avanza en el cálculo. Sin embargo, cuando converge, en general lo hace más rápido que los métodos cerrados como el método de regula-falsi.

El método de iteración de punto fijo utiliza una fórmula que se desarrolla como "una iteración simple de punto fijo" (también llamada iteración de un punto o sustitución sucesiva o método de punto fijo), al arreglar la ecuación f(x)=0 de tal modo que x esté del lado izquierdo de la ecuación:

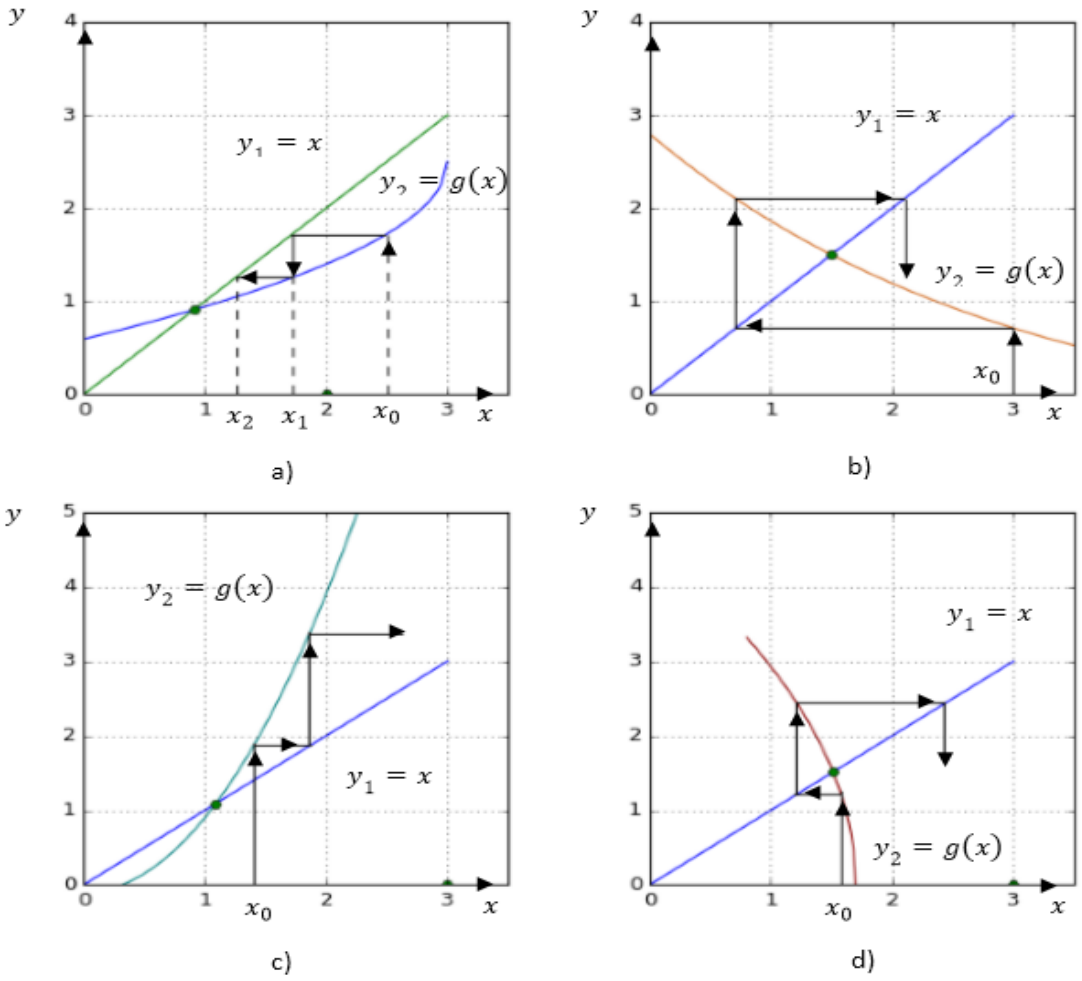
$$x = g(x) \quad (ec.\ 2.61)$$

Esta transformación se realiza mediante operaciones algebraicas o simplemente sumando x a cada lado de la ecuación original. De esta manera, dado un valor inicial para la raíz x_i, la ecuación (2.62) se utiliza para obtener una nueva aproximación x_{i+1} expresada por la fórmula iterativa:

$$x_{i+1} = g(x_i) \ (ec.\ 2.62)$$

El error aproximado de esta ecuación se calcula usando la ecuación (2.60). El método de la iteración simple converge si se cumple que en la región de interés $|g'(x)|<1$ (véase Figura 2.2). En otras palabras, la convergencia ocurre si la magnitud de la pendiente de g(x) es menor que la pendiente de la recta f(x)=x. Cuando el método converge, el error es proporcional y menor que el error en la iteración anterior. Por tal razón se dice que este método tiene convergencia lineal.

[7] Esta sección se basa en Chapra y Canale (2007, pp. 143–148) y en Nieves Hurtado y Domínguez Sánchez (2014, pp. 32–47).



Adaptado de Chapra y Canale (2007, p. 146)

Figura 2.2. Representación gráfica de la convergencia y de la divergencia del método de punto fijo.

2.3.3. *Método de Newton-Raphson para una ecuación no lineal*[8]

Si el valor inicial para la raíz es x_i, entonces se puede trazar una tangente desde el punto [x_i, $f(x_i)$] de la curva (véase Figura 2.3). Por lo común, el punto donde esta tangente cruza al eje x representa una aproximación mejorada de la raíz.

[8] Esta sección se basó en Chapra y Canale (2007, pp. 148–154)

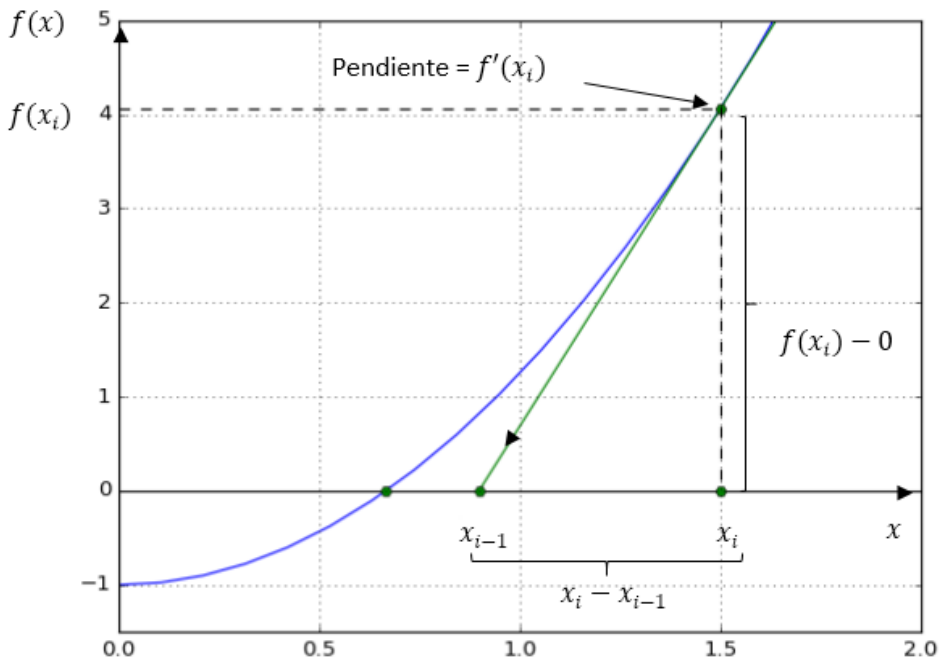
La ecuación del método de Newton es:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad (ec.\ 2.63)$$

Se ocupa el mismo criterio de terminación de la ecuación (2.60). La convergencia es cuadrática, lo que significa que el error es proporcional al cuadrado del error anterior.

El método de Newton se comporta de manera deficiente en los siguientes casos:

1. Caso de múltiples raíces.
2. Cuando la raíz ocurre en la cercanía de puntos de inflexión (esto es f''(x)=0) hay divergencia.
3. Comportamiento oscilante alrededor de un máximo o mínimo local. Esta tendencia a alejarse del área de interés se debe a que se encuentran pendientes cercanas a cero (f'(x)=0). Esto es catastrófico porque causa una división entre cero en la fórmula de Newton-Raphson.



Adaptado de Chapra y Canale (2007, p. 148).

Figura 2.3. Representación gráfica del método de Newton-Raphson. Se extrapola una tangente a la función en x_i (esto es $f'(x_i)$) hasta el eje x para obtener una estimación de la raíz x_i+1.

No hay un criterio general de convergencia para el método de Newton-Raphson. Su convergencia depende de la naturaleza de la función y de la exactitud del valor inicial. La única solución en estos casos es tener un valor inicial que sea suficientemente cercano a la raíz. Los valores iniciales se predicen con un conocimiento del problema físico.

2.3.4. *Método de la secante para una ecuación no lineal*[9]

Un problema potencial en la implementación del método de Newton-Raphson es la evaluación de la derivada. Aunque esto no es un inconveniente para los polinomios, existen funciones cuyas derivadas en ocasiones resultan muy difíciles de calcular. En esos casos, la derivada se puede aproximar mediante una diferencia finita dividida hacia atrás[10]:

$$f'(x_i) = \frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i} \quad (ec.\ 2.64)$$

Esta aproximación se sustituye en la ecuación (2.63) para obtener la siguiente ecuación iterativa:

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)} \quad (ec.\ 2.65)$$

El método requiere de dos valores iniciales de x. Sin embargo, dado que no es necesario que f(x) cambie de signo entre los valores dados, este método no se clasifica como un método cerrado (véase Figura 2.4).

La ecuación (2.59) del método de regula-falsi y la ecuación (2.65) del método de la secante son idénticas en los términos. Ambos métodos usan dos valores iniciales para calcular una aproximación de la pendiente de la función que se utiliza para proyectar hacia el eje x una nueva aproximación de la raíz. Sin embargo tienen una diferencia crítica consistente en la forma en que uno de los valores iniciales se reemplaza por la nueva aproximación.

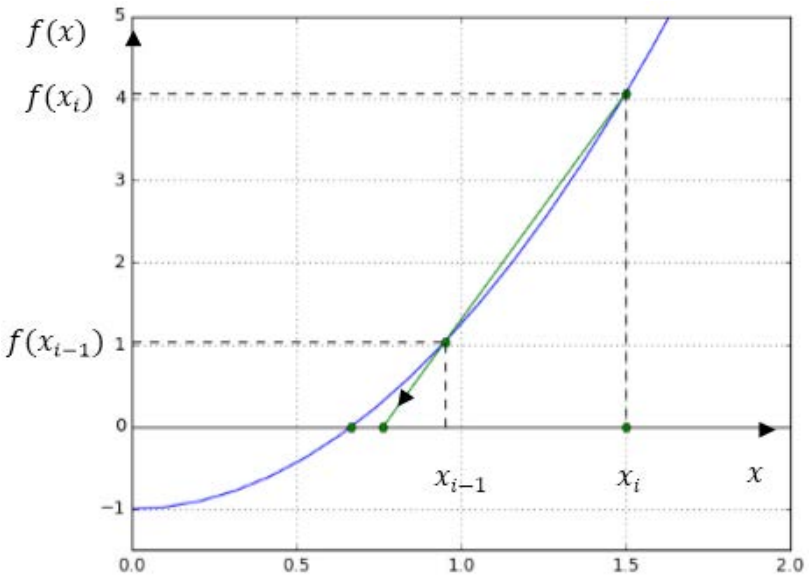
En el método de la falsa posición, la última aproximación de la raíz reemplaza cualquiera de los valores iniciales que dé un valor de la función con el mismo signo que $f(x_r)$. En

[9] Esta sección se basa en Chapra y Canale (2007, pp. 154–159).

[10] En la elaboración de la función del método de la secante, se ocupó la fórmula de la diferencia dividida finita centrada de cinco puntos.

consecuencia, las dos aproximaciones siempre encierran a la raíz. Por lo tanto, para todos los casos, el método siempre converge.

El método de la secante reemplaza los valores en secuencia estricta: con el nuevo valor de x_{i+1} se reemplaza a x_i y x_i reemplaza a x_{i-1}. En consecuencia, algunas veces los dos valores están en el mismo lado de la raíz. Esto puede llevar a divergencias. A pesar de esto, cuando el método de la secante converge lo hace más rápidamente que el método de la falsa posición.



Adaptado de Chapra y Canale (2007, p. 154).

Figura 2.4. Representación gráfica del método de la secante.

La ecuación (2.64) se puede modificar para considerar un cambio fraccionario de la variable independiente para estimar f'(x):

$$f'(x) = \frac{f(x_i + \delta x_i) - f(x_i)}{\delta x_i} \quad (ec.\ 2.66)$$

donde δx_i es un pequeño cambio fraccionario. Esta aproximación se sustituye en la ecuación (2.63) que da la siguiente ecuación iterativa:

$$x_{i+1} = x_i - \frac{\delta x_i f(x_i)}{f(x_i + \delta x_i) - f(x_i)} \quad (ec.\ 2.67)$$

La elección de un valor adecuado para δx_i no es automática. Si δx_i es muy pequeño, el método puede no tener éxito por el error de redondeo, causado por la cancelación por resta en el denominador de la ecuación (2.64). Si éste es muy grande, la técnica puede llegar a ser ineficiente y hasta divergente. No obstante, si se selecciona correctamente proporciona una adecuada alternativa en los casos donde la evaluación de la derivada se dificulta.

2.4. Herramientas de computación científica para la resolución de ecuaciones no lineales.

Los softwares de computación científica a utilizar son Scilab y Spyder. El primero es un lenguaje de programación por sí mismo mientras que Spyder es un ambiente de desarrollo para crear programas escritos en Python. También se consideró el uso de la calculadora TI-Nspire y de Excel para una mayor facilidad en los cálculos.

2.4.1. *Software para cálculo numérico*

Scilab

Programas y funciones en Scilab.

Cuando se requiere ejecutar varios comandos en Scilab, los archivos generados por el editor de código SciNotes resultan de gran utilidad. Estos archivos corren al hacer clic sobre el botón “ejecutar”, o el botón “guardar y ejecutar” de la barra de herramientas del SciNotes. El archivo generado puede tener dos tipos de extensiones:

- .sci: archivo que contiene únicamente funciones y al utilizar exec, este archivo se carga al entorno de scilab pero no devuelve ningún valor. Para obtener los resultados, es necesario llamar a las funciones e ingresarle los argumentos requeridos.
- sce: archivo que contiene funciones de scilab y enunciados ejecutables.

Funciones de Scilab para la búsqueda de raíces en ecuaciones no lineales

Scilab cuenta con dos funciones para la resolución de ecuaciones no lineales: roots y fsolve. El uso de fsolve se detalla en el siguiente ejemplo para la resolución de ecuaciones no lineales.

- Función a ocupar:

$$f(x) = \ln(x^2) - 1 \ \ (ec\ e.1),$$

x0=2

Se considerará la definición de las funciones a través de "function". Primero se crea una función para la ecuación (e.1) en un archivo determinado, luego se llama a dicho archivo y a fsolve. Se imprimen tanto el valor de la raíz, así como los valores de la función (e.1) al ser evaluada en dicha raíz. Esto último se realiza con el fin de comprobar que fsolve converge a la solución.

```
1
2  //Definición de la funcion funD
1  function y = funF(x)
2      y = log(x^2)-1
3  endfunction
6
```

Figura 2.5. Programa elaborado para almacenar a funF

La Figura 2.5 muestra el código de la función de la ecuación (e.1), realizando la definición de funciones a través de function. La Figura 2.6 presenta los resultados de la consola de Scilab y la Figura 2.7 muestra el código que llama a la función descrita en la ecuación no lineal (e.1) y a fsolve para calcular la raíz.

```
-->exec('C:\Users\RAZZ\Desktop\eval_fun.sce', -1)

 Raíz          funcs. evaluadas
 1.64872           0.00000
```

Figura 2.6. Respuestas obtenidas para el primer caso de una función, descrita en la ecuación e.1

```
1
2  exec("C:\Users\RAZZ\Desktop\funF.sci",-1)
3
4  //establece el valor inicial
5  x01 =2
6
7  //se llama a fsolve requiriendo que devuelva la raiz
8  //"r1F" y el valor de fun1F, "f1F"
9  [r1F,f1F] = fsolve(x01,funF)
10
11 printf("\n Raíz          funcs. evaluadas\n")
12 printf(" %2.5f          %2.5f\n", r1F, f1F)
```

Figura 2.7. Programa que llama a la función descrita en la ecuación e.1 y a fsolve para el cálculo de la raíz

Python, Spyder, IPython y el Cuaderno Jupyter en Anaconda 3

Características de Python

Python es un lenguaje de propósito general que resulta ventajoso sobre otros lenguajes de programación en algunos aspectos:

- Es un lenguaje relativamente simple de aprender debido a su sintaxis sencilla (Guttag, J. V., 2013, p. 7).
- Provee el soporte en línea de una diversidad de módulos científicos (Perkel, J. M., 2015b).
- Es un lenguaje multiplaforma y de licencia libre.

Para este trabajo de graduación se utilizará Spyder, el cual es un ambiente de desarrollo integrado (IDE) para Python (similar a Matlab, aunque de licencia libre).

Módulos en Python

La mayoría de la funcionalidad de Python es proporcionada por las bibliotecas o módulos.

El uso de módulos en los programas permite la reutilización del código. El resultado es que se mejora la legibilidad, se reducen los errores y aumenta la capacidad de dar mantenimiento a un programa, permitiendo una depuración y una adición de código más sencillas.

Python permite la programación modular en varios niveles (Johanssonn, 2016):

- ✓ De bajo nivel: funciones y clases.
- ✓ De alto nivel: módulos de Python.

En el desarrollo de los programas en Python, se utilizará con frecuencia el módulo Numpy de Python. La función roots de NumPy[11] permite obtener las raíces de un polinomio. Supongamos que tenemos los siguientes polinomios:

$$f1(x) = x^3 - 7 * x^2 + 14 * x + 8$$

y

$$f2(x) = x^3 + 3 * x^2 + 5 * x + 5$$

La función roots necesita como argumentos los coeficientes del polinomio y devuelve un vector con las raíces, ya sea reales o imaginarias. Entonces ingresamos en el cuaderno electrónico[12] (véase Figura 2.8):

```
In [1]: p=[1,-7,14,-8]
        import numpy as np
        np.roots(p)

Out[1]: array([ 4.,  2.,  1.])

In [2]: p=[1,3,5,5]
        np.roots(p)

Out[2]: array([-1.7709170+0.j        , -0.6145415+1.56388451j,
               -0.6145415-1.56388451j])
```

Figura 2.8. Búsqueda de raíces de dos polinomios utilizando a roots() de NumPy.

[11] Para mayor información sobre Numpy, veáse Scipy Developers (2016)
[12] Véase más adelante la descripción del cuaderno electrónico Jupyter

IPython

IPython fue creado por Fernando Pérez en 2001 y funciona como un intérprete del lenguaje de programación Python con una facilidad de manejo mejorada. IPython ofrece una rápida exploración de algoritmos a través del historial de comandos ("command history"), análisis de datos y visualización (Perez & Granger, 2007). IPython también brinda una interfaz web, para el cuaderno electrónico.

El cuaderno electrónico Jupyter[13].

La interfaz web del cuaderno Jupyter está basada en los cuadernos electrónicos de Mathematica y Sage y fue lanzado en 2011 con el objetivo de[14]:

- Servir como un entorno para la escritura y ejecución de código de una manera interactiva y exploratoria.
- Ofrecer además un editor de texto más conveniente que la consola original de IPython, para la escritura de texto narrativo, de ecuaciones matemáticas y otro tipo de texto enriquecido.
- Avanzar y retroceder fácilmente entre todos los elementos anteriores.

El cuaderno Jupyter funciona en la mayoría de navegadores web, puede ser convertido a formatos estáticos como HTML (Granger & Pérez, 2016) y es compatible con una diversidad de lenguajes de programación para el análisis de datos tales como Julia, Python y R (Van Noorden, R., 2014). Si bien el cuaderno electrónico utiliza un navegador web, el cuaderno Jupyter se corre localmente o desde la misma computadora que abre el navegador.

[13] A partir del 2015 el cuaderno electrónico IPython se renombró como cuaderno Jupyter.
[14] Véase Mascarelli (2014) y Rossant (2016).

Anaconda

Anaconda (Continuum Analytics, 2015), es un software multiplataforma de código abierto, basada en la tecnología de Python para el análisis de datos. Su principal característica es la de permitir una instalación eficiente y libre de problemas de compatibilidad entre distintos módulos de Python a través de Conda, un gestor de paquetes, de dependencia y de entorno. Anaconda hace uso del cuaderno Jupyter y de Spider.

2.4.2. *Software con CAS*

Las herramientas de computación científica que cuentan con sistemas de álgebra computacional o CAS (Computer Algebraic System en inglés) permiten manipular ecuaciones matemáticas y expresiones en una forma simbólica (Zotos, 2008).
Los sistemas CAS pueden aparecer en la forma de software como Mathematica y Maple o como una interfaz de calculadora graficadora como la TI-92, Voyage 200 o TI-Nspire. Otros paquetes de software con características de CAS son Mathcad, Maxima, Derive y el módulo Sympy de Python.

Calculadoras TI

Aplicación Calculadora

Las calculadoras Texas Instrument (TI) como la Nspire CX CAS pueden resolver ecuaciones no lineales. La Nspire CX CAS ocupa para ello las funciones solve(), zeros(), nSolve(), polyRoots() y factor(). Las funciones cSolve(), cZeros() y cPolyRoots() también resuelven ecuaciones no lineales pero devuelven tanto respuestas reales como imaginarias.
En la Figura 2.9 se muestran ejemplos de cálculo de raíces con solve(), zeros(), polyRoots(), nSolve() y factor()[15]

[15] Texas Instruments (2016) describe con mayor detalle el uso de las funciones referidas.

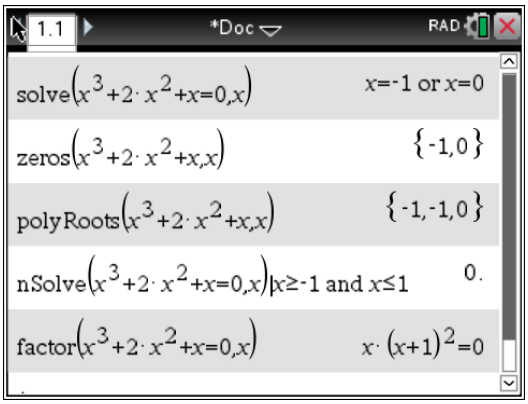


Figura 2.9. Ejemplos de cálculo de raíces utilizando las funciones solve, zeros, polyRoots, nSolve y factor

2.4.3. *Hojas de cálculo de Excel*

Ventajas de la hoja de cálculo de Microsoft Excel

Según Baker y Sugden (2007), las hojas de cálculo brindan una revisión inmediata a cambios en los datos o fórmulas; permiten la disponibilidad simultánea en la pantalla de datos, de fórmulas y de gráficos; y finalmente habilitan la resolución de problemas complejos, la investigación de distintos escenarios y el manejo de grandes cantidades de datos sin necesidad de programación.

La hoja de cálculo de Microsoft Excel en especial es ventajosa con respecto a otros software en términos de una razón costo/beneficio baja, y su uso constante en escuelas, oficinas, y fábricas (Moura, 2005).

Cálculos numéricos en Excel

Excel cuenta con dos herramientas para la búsqueda de raíces de ecuaciones: Buscar Objetivo y Solver. Buscar Objetivo es adecuado para la búsqueda de raíces en ecuaciones algebraicas, mientras que Solver también se ocupa en la resolución de sistemas de ecuaciones no lineales, y en problemas de optimización lineal y no lineal restringida, problemas de optimización

lineal y no lineal no restringida; así como en problemas de regresión no lineal (Ferreira, Lima, & Salcedo, 2004; Rosen & Partin, 2000).

Resolución de problemas de ingeniería química con Excel/VBA

Dado que la hoja de cálculo de Excel/VBA está bien adaptada para problemas que requieran aproximaciones sucesivas, es posible resolver una diversidad de problemas en ingeniería química tales como[16]:

- Revalúo de las tablas de flujo de efectivo cuando se modifican asunciones económicas.
- Integración numérica de las ecuaciones de diseño de columnas de destilación, absorbedores y otro tipo de equipo de separación; así como en problemas de transferencia de calor en dos dimensiones.
- Balances de masa y energía sin la ayuda de paquetes de diseño de procesos. Las hojas de cálculo, por sí solas, son adecuadas únicamente para problemas de balance de procesos de tamaño mediano con pocas o ninguna corriente de reciclo; caso contrario es más útil la programación en VBA.
- Ensayos de prueba y error para la determinación de la composición en la interfaz de fases inmiscibles para procesos de separación basados en el transporte de masa.

Ejemplos de uso de Buscar Objetivo y de Solve

Chapra y Canale (2007, pp. 187–190) presentan un ejemplo para el uso de la función Buscar Objetivo que se reproducen a continuación utilizando Excel 2013.

Para el ejemplo de Buscar objetivo se ocupará la función: $f(x) = x - \cos(x)$. Para el intervalo $x = [0, 2]$, evaluamos la función f(x) en el vector x, ingresamos un valor inicial de x0 en la celda A11 y evaluamos la función f(x0) en B11 (véase Figura 2.10). Buscar objetivo

[16] Véase Grulke (1986) y Hinestroza y Papadopoulos (2003).

cambiará el valor de la celda B11 hasta que sea aproximadamente cero y así obtener la raíz en A11.

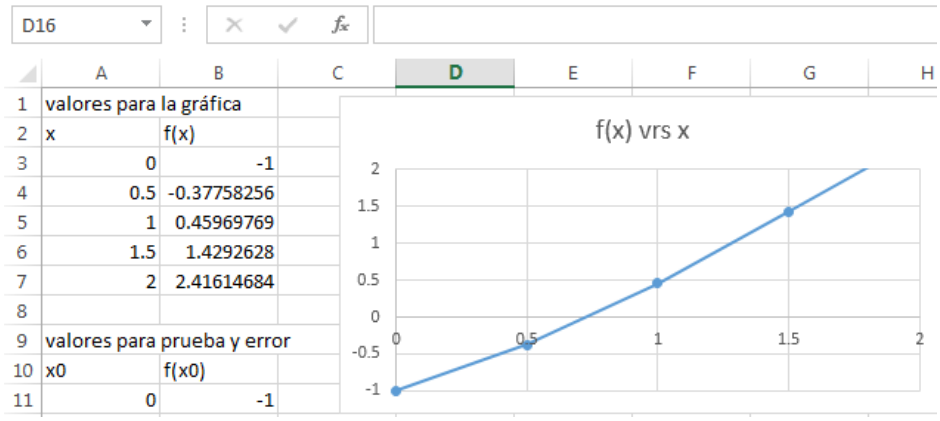


Figura 2.10. Digitado de datos en la hoja de cálculo para el ejemplo de uso de Buscar Objetivo

La Figura 2.11 muestra el valor de la raíz x=0.739024 para un valor de f(x)=-3.7945E-05.

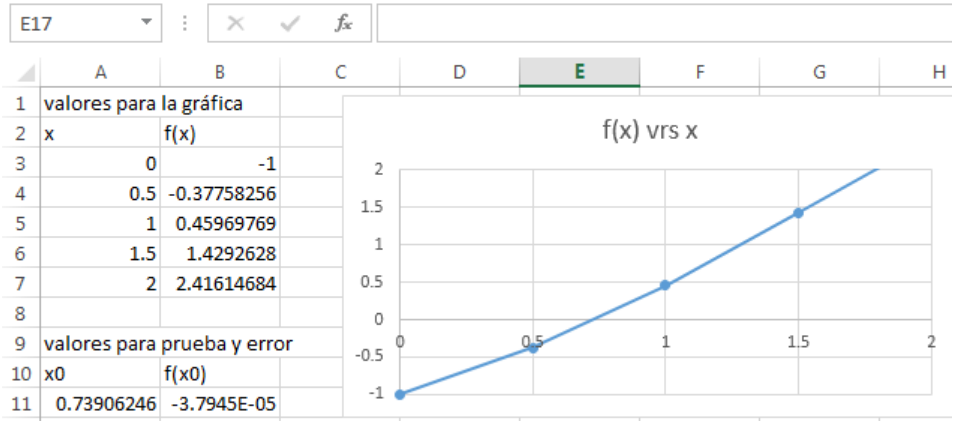


Figura 2.11. Respuesta de Buscar Objetivo.

3. Metodología a desarrollar

3.1.Elaboración del formato de la encuesta a utilizar.

Para recabar los datos necesarios para la selección final de los problemas de cálculo, se elige el instrumento del cuestionario en encuesta puesto que su realización implica un consumo reducido de tiempo por parte de los docentes. El universo de la encuesta estará conformado por los docentes que enseñan en las asignaturas de Termodinámica Química I (TQI-115), Termodinámica Química II (TQI-215), Operaciones Unitarias I (OPU-115) y Operaciones Unitarias III (OPU-315). El formato de la encuesta y la solicitud dirigida a la Escuela de Ingeniería Química e Ingeniería de Alimentos para la realización de la encuesta se encuentran en el Anexo A.

3.1.1. *Elementos a tomar en cuenta*

La encuesta se elaborará con el fin de recabar la siguiente información:

- Tipos de herramientas de software utilizadas en la asignatura.
- Métodos numéricos utilizados y/o conocidos por los docentes en la resolución de problemas de cálculo con ecuaciones no lineales.
- Tipos de problemas de cálculo que implican la resolución de ecuaciones no lineales en la asignatura.
- Grado de aceptación del docente hacia el uso de las herramientas informáticas diseñadas.

3.1.2. *Afinamiento de los ítems del cuestionario de la encuesta y recolección final de datos*

Se realizará un ensayo en el cual se haga la entrega del cuestionario a los docentes a encuestar. Esto se llevará con el fin de establecer si los ítem de dicho cuestionario son comprensibles y pertinentes a la asignatura en cuestión. Luego de haber modificado el formato se hará la recolección final de los datos

3.2.Selección de Problemas de Cálculo

La selección los problemas de cálculo se hará en base a los siguientes criterios:

Tabla 3.1. Conjunto de criterios utilizados en la selección de los problemas de cálculo para las distintas asignaturas

Criterio	Descripción
1	El problema de cálculo debe ser parte de la temática de la asignatura en cuestión
2	Debe haber al menos un problema resuelto del tipo mencionado por los docentes en la encuesta en al menos uno de los libros de referencia utilizados en la asignatura en cuestión
3	Debe ser posible resolver el problema de cálculo utilizando al menos uno de los métodos numéricos preferidos por los docentes para la resolución de ecuaciones no lineales o sistemas de ecuaciones no lineales.
4	El problema propuesto debe referirse a un problema puntual de resolución de una ecuación no lineal.
5	El problema a resolver debe ser desarrollado en un número elevado de referencias bibliográficas de la asignatura en cuestión.

De los cinco criterios presentados en la Tabla 3.1, los criterios 1, 2, 3 y 4 se ocuparon siempre en la selección de los problemas de cálculo. El criterio 5 se ocupó en TQI-115 y OPU-115 con la finalidad de volver manejable el número de problemas identificados.

3.3.Elaboración de programas

A continuación se describen los procesos de resolución de problemas de cálculo sin el uso de rutinas informáticas, de generación de algoritmos, de flujogramas y de codificación en Scilab y Python.

3.3.1. *Resolución de problemas de cálculo sin el uso de rutinas informáticas.*

Los problemas de cálculo se resolverán primero sin rutinas informáticas, haciendo uso de diversas herramientas como:

- ✓ La línea de comandos de IPython y de Scilab que permiten la evaluación por etapas de cálculos.
- ✓ Hojas de cálculo como las de Excel, que, luego de establecer las celdas con valores invariantes y de editar las fórmulas, permite imitar el comportamiento de una estructura iterativa como el ciclo while y for de Scilab y Python, mediante la opción de llenado de celdas.
- ✓ Las aplicaciones de calculadora y hoja de cálculo del dispositivo TI nspire CX-CAS. La primera aplicación de TI nspire CX-CAS tiene la misma funcionalidad que las líneas de comandos de IPython y Scilab, mientras que la segunda aplicación proporciona, al igual que Excel, la capacidad de realizar una gran cantidad de cálculos con la opción de llenado de celdas.

3.3.2. *Codificación en Scilab y Python*

Los paradigmas de programación a utilizar en la elaboración de programas serán los de programación estructurada y programación modular, porque resultan ser más conocidos que otros paradigmas.

El paradigma de programación estructurada implica que un programa se puede escribir únicamente con tres estructuras de control: estructuras secuenciales, estructuras de selección y estructuras de repetición. Mientras que el paradigma de programación modular es apto para programas extensos, dado que divide el programa en piezas pequeñas o módulos. Se incluirá en el Anexo C el conjunto de seudocódigos utilizados en los programas generados en Scilab y Python.

3.3.3. *Documentación de programas*

Se comentarán los programas con el fin de explicar sus componentes a los usuarios. Además se elaborará un manual de usuario donde se explique de forma detallada, mediante texto y diagramas, el funcionamiento de los programas elaborados abarcando aspecto como: tipo de datos de entrada que se introducirán al programa y tipo de datos de salida que se obtendrán, el método de resolución de ecuaciones no lineales utilizado para el programa en cuestión y fuentes de consulta del programa elaborado.

Los flujogramas se elaborarán auxiliándose del software de acceso libre Dia 0.97.2 (The Dia Developers, 2014), el cual permite una mayor facilidad en el manejo de las figuras y líneas conectoras, con respecto a las "Formas" de Word 2013.

3.4.Elaboración de cuadernos Jupyter

Los cuadernos Jupyter se prepararán como módulos de enseñanza, para ello:

- Se agregarán títulos y subtítulos,
- Se utilizará texto narrativo para explicar algún aspecto del problema de cálculo a resolver;
- Se ocupará lenguaje Latex, el cual permite editar ecuaciones con un formato similar al de los libros de texto,
- Se agregarán programas simples que luego pueden modificarse para agregar detalles más complejos,
- Se usarán recursos multimedia, como videos, que aporten alguna explicación valiosa al problema en cuestión.

3.5.Revisión de los programas por el Usuario

Una vez finalizados los programas para los problemas de cálculo en Ingeniería Química, se pretende realizar una serie de sesiones para conocer el grado de aceptabilidad de dichos

programas. Estas sesiones se realizarán para los profesores de las asignaturas de TQI-115, TQI-215I, OPU-115 y OPU-315.

3.5.1. *Revisión de programas por profesores*

En las sesiones de revisión de programas, los profesores de TQI-115, TQI-215, OPU-115 y OPU-315 evaluaron los programas elaborados y realizaron distintas observaciones sobre las forma de evaluar los programas informáticos presentados.

4. Resultados

4.1. Resultados de la encuesta

Como resultado del análisis de los resultados de la encuesta, se seleccionaron un conjunto de 12 problemas de cálculo a resolver con Scilab y Python: 2 problemas de Termodinámica Química I (TQI-115), 5 problemas de Termodinámica Química II (TQI-215), 4 problemas de Operaciones Unitarias I (OPU-115) y 1 problema de Operaciones Unitarias III (OPU-315).

Tabla 4.1. Datos generales de la encuesta realizada

Lugar de la encuesta[17]	Escuela de Ingeniería Química e Ingeniería de Alimentos, tercera planta del edificio K de la Facultad de Ingeniería y Arquitectura, Universidad de El Salvador.
Primera entrega del instrumento (afinamiento de los ítems del cuestionario):	8 y 9 de diciembre
Fecha de recolección final de datos:	12 al 14 de diciembre de 2016
Universo bajo estudio:	Docentes activos por un periodo de cinco años en la enseñanza de TQI-115, TQI-215, OPU-115 y OPU-315
Total encuestados	7 docentes encuestados

[17] La carta a la Escuela de Ingeniería Química e Ingeniería de Alimentos sobre la solicitud de información de personal docente activo se encuentra en el Anexo A.5

La Tabla 4.1 muestra los datos generales de la encuesta realizada para recabar datos sobre los problemas de cálculo de interés. El objetivo de la encuesta fue el de determinar los métodos numéricos de resolución de ecuaciones no lineales en problemas de cálculo de las asignaturas mencionadas; también se pretendía establecer un conjunto de problemas de cálculo a resolver mediante los programas en Scilab y Spyder, de entre los cuales se seleccionaron algunos que cumplan con los siguientes criterios:

- El problema de cálculo debe ser parte de la temática de la asignatura en cuestión (OPU-115, OPU-315, TQI-115 o TQI-215) (**Criterio 1**).
- Debe haber al menos un problema resuelto del tipo mencionado por los docentes en la encuesta en al menos uno de los libros de referencia utilizados en la asignatura en cuestión (**Criterio 2**).
- Debe ser posible resolver el problema de cálculo utilizando al menos uno de los métodos numéricos preferidos por los docentes para la resolución de ecuaciones no lineales o sistemas de ecuaciones no lineales (**Criterio 3**).
- El problema propuesto debe referirse a un problema puntual de resolución de una ecuación no lineal (**Criterio 4**).

El cuestionario utilizado en la encuesta constaba de 4 preguntas abiertas y 1 cerrada (véase Anexo A.1). Las tres primeras preguntas estaban dirigidas a obtener información sobre cuáles eran los métodos numéricos utilizados o conocidos por los docentes para la resolución de ecuaciones o sistemas de ecuaciones no lineales. Las últimas dos preguntas recaban información sobre distintos tipos de problemas de cálculo, que involucraban ecuaciones no lineales y sistemas de ecuaciones no lineales, así como el posible grado de aceptación, por parte de los docentes de los programas elaborados. A continuación se detallan los resultados y el respectivo análisis para cada asignatura.

En la encuesta, los docentes sugirieron distintos tipos de problemas de cálculo. Se consideró en un inicio la consulta de las referencias bibliográficas especificadas en los planes de estudio de las asignaturas, pero fue necesario llevar a cabo la actualización de dichas referencias, con respecto a la fecha de publicación de las ediciones disponibles.

4.1.1. *Termodinámica Química I*

En caso de requerir un método numérico para la resolución de ecuaciones no lineales y/o sistemas de ecuaciones no lineales, los docentes de TQI-115 ocuparían:

- Método de regula-falsi
- Método de sustitución sucesiva
- Método de Newton

Entre los paquetes de software preferidos están Thermograph 5.1, Thermonator, CATT-II y IAWPS para la búsqueda de propiedades termodinámicas en tablas de vapor y la función Solver de la calculadora TI-Nspire ocupada en la solución de ecuaciones de estado para gas y líquido saturados. Los docentes expresaron que desconocen a Scilab y a Spyder y por lo tanto sería necesario presentar y describir las características de dichos programas para el conocimiento de los docentes.

Los problemas de cálculo propuestos son:

- Cálculo del volumen molar en ecuaciones cúbicas de estado.
- Relaciones PVT de fluidos puros (cálculo de volumen y densidad específica, presiones de vapor).
- Variación de parámetros de operación en ciclos termodinámicos de ciclos de potencia.

El primer problema se ubica en la unidad 2 de “Propiedades volumétricas de los fluidos puros”, subunidad 2.2 “Comportamiento de vapores y gases. Uso del factor acéntrico, ecuaciones viriales”. Se busca el cálculo de los volúmenes de líquido y vapor saturados (sólo es apropiado utilizar las ecuaciones cúbicas de estado para sustancias no polares). Los ejemplos presentados en las referencias bibliográfícas se muestran en la Tabla 4.2.
El segundo problema se ubica en la unidad 2 de “Propiedades volumétricas de los fluidos puros”, subunidades 2.1 “Comportamiento PVT y superficies termodinámicas de sustancias puras. Uso de tablas de propiedades termodinámicas”; 2.2 “Comportamiento de vapores y gases. Uso del factor acéntrico, ecuaciones viriales”.

Tabla 4.2. Autores que presentan cálculos del volumen usando la ecuación de estado cúbica genérica

Autor	Ejemplo numérico		Método numérico utilizado
	Cálculos y resultados	Sólo resultados	
Smith et al. (2007, pp. 96–99)	Sí	No	Método de sustitución sucesiva
Reid et al. (1987, pp. 42–43, 46)	No	Si	--

En este tipo de problema el uso de la correlación generalizada de Pitzer del tercer coeficiente del virial en conjunción con la ecuación de Kammerlingh-Onnes[18], implica cálculos iterativos.

Únicamente Smith et al. (2007, pp. 103–106) resuelve un ejercicio numérico mediante el método de sustitución sucesiva, aunque no presentan los detalles del procedimiento seguido.

El tercer problema es posible ubicarlo en la unidad 7 “Generación de potencia a través de ciclos termodinámicos”, subunidad 7.1 “Planta de vapor. Ciclos de vapor y su análisis termodinámico”.

Este problema parte del hecho de que es posible realizar mejoras al ciclo Rankine (el modelo más simple para una planta de energía) en cuanto a su eficiencia. Básicamente la eficiencia del ciclo Rankine puede incrementarse al disminuir la presión y temperatura del vapor exhausto, al incrementar la presión y temperatura durante la adición de calor y al sobrecalentar el vapor (Faires & Simmang, 1983, p. 242; Moran & Shapiro, 2004, p. 390; Smith et al., 2007, p. 296; Sonntag, Borgnakke, & Van Wylen, 2003, p. 388).

[18] Se suelen utilizar los coeficientes de la ecuación de Holborn, aprovechando la equivalencia de estos con los coeficientes de la ecuación de Kamerlingh-Onnes.

Moran y Shapiro (2004, p. 394) también incorporan un ejercicio donde se explora como varía la eficiencia de un ciclo con recalentamiento con respecto a la variación de la eficiencia de la turbina, sin la realización de cálculos iterativos.
La Tabla 4.3 resume lo dicho anteriormente para TQI-115.

Tabla 4.3. Problemas de cálculo TQI-115, luego de aplicación de criterios

Métodos de resolución de ecuaciones no lineales preferidos por los docentes
➢ Método de Regula-falsi ➢ Método de sustitución sucesiva ➢ Método de Newton
Problemas que implican ecuaciones no lineales
➢ Volumen molar en ecuaciones cúbicas de estado, utilizando parámetros de la ecuación de Van der Waals, Redlich-Kwong, Soave-Redlich-Kwong, Peng-Robinson. ➢ Volumen molar mediante la ecuación de Kammerlingh-Onnes y las correlaciones para el segundo y el tercer coeficientes del virial.

4.1.2. *Termodinámica Química II*

En caso de requerir un método numérico para la resolución de ecuaciones no lineales y/o sistemas de ecuaciones no lineales, los docentes de TQI-215 ocuparían:

- Método de la bisección,
- Método de regula falsi
- Método de Newton-Raphson
- Método de la secante

Entre los paquetes de software preferidos están Word, Datafit_80 ocupado en la correlación de datos, Maple 2016, que es un software de aplicación matemática; las hojas de cálculo de Excel para la realización de laboratorios y manejo de datos, Matlab, Mathcad, Scilab y

Octave para la realización de laboratorios. Los docentes expresaron que desconocen a Spyder y por lo tanto sería necesario presentar y describir las características de dicho software. También los docentes hicieron la observación de que, dada la variabilidad del software libre de una versión a la siguiente, se vuelve difícil el uso de este software y que sería ventajoso que los programas elaborados en Scilab y Spyder fueran aplicativos y desarrollados.

Los problemas de cálculo propuestos son:

- Cálculo de propiedades de sustancias puras a partir del uso de ecuaciones de estado
- Cálculo del coeficiente de fugacidad
- Cálculo del coeficiente de actividad (por ejemplo cálculos de UNIFAC-UNIQUAC)
- Problemas de equilibrio de fase, incluyendo ELV (cálculos del punto de burbuja T y de rocío T)
- Problemas de equilibrio químico

El primer problema puede ubicarse en la unidad 2 de “Propiedades Termodinámicas de los Fluidos”, subunidades 2.3 “Propiedades Termodinámicas de los Fluidos”; 2.6 “Aplicación del cálculo de propiedades termodinámicas en el análisis de procesos de flujo”.

La obtención de propiedades termodinámicas como la entalpía, entropía, energía de Helmholtz y energía de Gibbs para sustancias puras implica el uso de propiedades residuales para gases. Los ejemplos presentados en las referencias bibliográficas se muestran en la Tabla 4.4.

Smith et al. (2007, pp. 265–266, 270–272, 276–277) también presentan un grupo de ejercicios para procesos de flujo de gases reales, en donde se utiliza las fórmulas de propiedades residuales, basadas en la correlación de Pitzer, y se busca la temperatura final de un proceso dado mediante el método de sustitución sucesiva.

El segundo y tercer problemas se ubican en la unidad 3 de “Propiedades Termodinámicas en las Mezclas Homogéneas”, subunidades 3.4 “Fugacidad y coeficiente de fugacidad i de componentes en solución”; 3.5 “La solución ideal. Fugacidad de soluciones ideales”; 3.6 “Coeficiente de actividad de i componentes en solución. Relación con la fugacidad”; 3.9 “Coeficiente de actividad y propiedades de exceso".

Tabla 4.4. Autores que presentan cálculos del volumen usando la ecuación de estado cúbica genérica

Autor	Ejemplo numérico		Método numérico utilizado
	Cálculos y resultados	Sólo resultados	
Smith et al. (2007, pp. 219–220)	Sí	No	Método de sustitución sucesiva
Reid et al. (1987, pp. 111, 120–121)	No	Si	--
Poling et al. (2001, p. 6.9-6.13)	No	Si	--

Para gases puros y especies en solución[19], los cálculos del coeficiente de fugacidad, realizados con la ecuación de estado cúbica general o una ecuación de estado cúbica específica (Sandler, 2006, pp. 298, 300, 425; Smith et al., 2007, pp. 564–565), son iterativos, porque se debe calcular el factor de compresibilidad.

Los ejemplos numéricos muestran que los cálculos de los coeficientes de actividad son no iterativos (Abbott & Van Ness, 1975, pp. 263–264).

El cuarto problema se ubica en la unidad 4 “Equilibrio de Fases en Sistemas Multicomponentes”, subunidades 4.3 “Equilibrio liquido – vapor (ELV)”; 4.4 “Diagrama de fases para sistemas miscibles. Cálculos”; y 4.8 “Cálculos de ELV con correlaciones generalizadas”. Sólo se describirá el equilibrio líquido vapor (ELV) puesto que el programa de la asignatura sólo toma en cuenta este tipo de equilibrio. Los ejemplos presentados en las referencias bibliográficas se muestran en la Tabla 4.5.

El quinto problema se ubica en la unidad 5 “Equilibrio de Reacciones Químicas”, subunidad 5.5 “Relación de la constante de equilibrio con la composición”.

[19] Al calcular propiedades de mezcla, el uso de las ecuaciones de estado debe considerar a las reglas de mezclado.

En general, la expresión de equilibrio a resolver depende de los reactivos, productos y fases involucradas y de las suposiciones hechas. Los ejemplos presentan los cálculos realizados para dos o más de las iteraciones requeridas para resolver el problema y se pueden encontrar en:

Para mezcla de gases, los siguientes autores brindan ejemplos numéricos:

Mezclas de gases ideales: Abbott y Van Ness (1975, pp. 291, 315–316, 2008, pp. 4–37); Smith et al (2007, pp. 502–504, 505–507); Sandler (2006, pp. 760–763). Hougen y Watson (1947, pp. 716–717) utiliza un método de prueba y error para resolver un problema de este tipo.

Soluciones ideales de gases: Smith et al (2007, pp. 504–505); Abbott y Van Ness (1975, pp. 315–316, 2008, pp. 4-37-4–38).

Mezclas gaseosas a altas presiones: Hougen y Watson (1947, pp. 717–718).

Múltiples reacciones para mezclas de gases ideales: Abbott y Van Ness (1975, pp. 295–296), Hougen y Watson (1947, pp. 730–731), Sandler (2006, pp. 752–754) y Smith et al. (2007, pp. 519–520, 520–524, 527–528).

Para reacciones en fase líquida, Sandler (2006, p. 731) y Smith et al. (2007, pp. 507–508) presentan los resultados a ejercicios de reacciones para una solución ideal de compuestos líquidos.

En algunos de los ejemplos de reacciones en sistemas heterogéneos, se supone que la actividad de compuestos sólidos es 1 y por lo tanto no influye en la expresión de equilibrio, y que los compuestos gaseosos que participan en la reacción son ideales. Así, Sandler (2006, pp. 737–738) utiliza un método analítico para resolver un ejercicio de esta naturaleza, mientras que Sandler (2006, pp. 749–750) únicamente muestra los resultados a los problemas. Smith et al. (2007, pp. 511–514) muestra los resultados de un ejercicio de reacción heterogénea en donde se asume que la fase gaseosa es una solución ideal y que en la fase líquida, las fugacidades de líquido se pueden sustituir por las fugacidades de líquido saturado.

Tabla 4.5. Autores que presentan cálculos de equilibrio líquido-vapor

Tipo de cálculo	Número de componentes	Ecuación para el cálculo ELV ocupada	Autor	Ejemplo numérico		Método numérico utilizado
				Cálculos y resultados	Sólo resultados	
Burbuja T	2	Raoult o Raoult modificada	Abbott y Van Ness (1975, pp. 273–274)	Si	--	Sustitución sucesiva
			Abbott y Van Ness (2008, pp. 4-30-4–31)	--	Si	--
			Smith et al. (2007, pp. 356, 360–361)	Si	--	Sustitución sucesiva
			Sandler (2006, p. 502)	Si	--	Prueba y error
			Poling et al. (2001, p. 8.99-8.102)	Si	--	Sustitución sucesiva
	2	Soave-Redlich-Kwong y Formulación $\hat{\phi} - \hat{\phi}$	Reid et al. (1987, p. 350)	Si	--	Método de Newton–Raphson
			Poling et al. (2001, p. 8.130)	Si	--	Método de Newton–Rapshon
	3	Raoult o Raoult modificada	Poling et al. (2001, p. 8.104-8.108)	--	Si	Método de sustitución sucesiva
			Reid et al (1987, pp. 281–283)	--	Si	Método de Newton-Raphson

Pasa…

Tabla 4.5. Autores que presentan cálculos de equilibrio líquido-vapor (continuación)

Tipo de cálculo	Número de componentes	Ecuación para el cálculo ELV ocupada	Autor	Ejemplo numérico		Método numérico utilizado
				Cálculos y resultados	Sólo resultados	
Burbuja T	3	Raoult o Raoult modificada	Poling et al. (2001, p. 8.40-8.42)	--	Si	Método de Newton-Raphson
		Valores experimentales de K	Hougen y Watson (1947, pp. 674–676)	--	Si	Prueba y error
	4	Formulación $\gamma - \Phi$	Smith et al. (2007, p. 551)	Si	--	--
Rocío T	2	Raoult o Raoult modificada	Abbott y Van Ness (2008, pp. 4-30-4–31)	--	Si	--
			Smith et al. (2007, pp. 356, 361–362)	Si	--	--
			Sandler (2006, p. 502)	Si	--	Prueba y error
	3	Valores experimentales de K	Hougen y Watson (1947, pp. 674–676)	--	Si	Prueba y error

Pasa…

Tabla 4.5. Autores que presentan cálculos de equilibrio líquido-vapor (continuación)

Tipo de cálculo	Número de componentes	Ecuación para el cálculo ELV ocupada	Autor	Ejemplo numérico		Método numérico utilizado
				Cálculos y resultados	Sólo resultados	
Burbuja P	2	Soave-Redlich-Kwong y Formulación $\hat{\phi} - \hat{\phi}$	Smith et al (2007, pp. 566–568)	Si	--	Iteración de punto fijo
		Raoult o Raoult modificada	Abbott y Van Ness (2008, pp. 4-30-4–31)	Si	--	--
Rocio P	2	Raoult o Raoult modificada	Abbott y Van Ness (2008, pp. 4-30-4–31)	Si	--	--

Para problemas de búsqueda de una temperatura, Hougen y Watson (1947, pp. 723–724) utiliza un método de prueba y error para encontrar la temperatura necesaria para producir una reacción de descomposición; mientras que Smith et al. (2007, pp. 143–144, 508–510) usa método de sustitución sucesiva en el cálculo de la temperatura final adiabática para una reacción de mezcla de gases ideales. Abbot y Van Ness (1975, pp. 313–314) sólo proporciona los resultados a un ejemplo numérico de este tipo.

La Tabla 4.6 resume lo descrito anteriormente para TQI-215:

Tabla 4.6. Problemas de cálculo de TQI-215, luego de aplicación de criterios

Métodos de resolución de ecuaciones no lineales preferidos por los docentes
➢ Método de la bisección ➢ Método de Regula falsi ➢ Método de Newton-Raphson ➢ Método de la secante
Problemas que implican ecuaciones no lineales
➢ Entalpía, entropía y energía de Gibbs residual de un gas puro, mediante la ecuación cúbica genérica. ➢ Temperatura de una corriente gaseosa pura en procesos de flujo. ➢ Coeficiente de fugacidad de especies en solución, mediante una ecuación cúbica de estado. ➢ Punto de burbuja T y rocío T utilizando la ecuación de Raoult para mezclas binarias, ternarias y cuaternarias. ➢ Punto de burbuja T y rocío T utilizando la ecuación de Raoult modificada para mezclas binarias, ternarias y cuaternarias. ➢ Punto de burbuja T y rocío T utilizando la formulación gamma-phi para mezclas binarias, ternarias y cuaternarias.

Pasa…

Tabla 4.6. Problemas de cálculo de TQI-215, luego de aplicación de criterios (continuación)

Problemas que implican ecuaciones no lineales
➢ Punto de burbuja T y rocío T utilizando la formulación phi-phi para mezclas binarias, ternarias y cuaternarias y la ecuación cúbica genérica. ➢ Reacción única para mezclas de gases ideales ➢ Reacción para soluciones de gases ideales ➢ Reacción para mezclas gaseosas a altas presiones ➢ Múltiples reacciones para mezclas de gases ideales. ➢ Reacciones para soluciones ideales en fase líquida ➢ Reacciones heterogéneas, que sólo consideran compuestos gaseosos ➢ Reacciones heterogéneas, que consideran compuestos gaseosos y líquidos ➢ Búsqueda de la temperatura en una reacción de descomposición de un compuesto sólido. ➢ Búsqueda de la temperatura en una reacción adiabática para mezclas de gases ideales.

4.1.3. *Operaciones Unitarias I*

En caso de requerir un método numérico para la resolución de ecuaciones no lineales y/o sistemas de ecuaciones no lineales, los docentes de OPU-115 ocuparían:

- Método de sustitución sucesiva,
- Método de Newton-Raphson
- Métodos iterativos con las ecuaciones que se dan en la teoría.

Entre los paquetes de software preferidos están Word para guiones de clase, discusiones y laboratorios, aplicación en Android Re y Moody, programa para el cálculo de la viscosidad en calculadoras Texas Instrument, Datafit_80 ocupado en la correlación de datos, Maple 2016; las hojas de cálculo de Excel para el cálculo de notas y promedios, Mathcad. Los

docentes expresaron que podrían aplicar los programas elaborados en Scilab y Spyder, pero sería ventajoso si los mismos fueran aplicativos y desarrollados.
Los problemas de cálculo propuestos son:

- Cálculo del factor o coeficiente de fricción
- Flujo incompresible de fluidos
    - Tuberías simples: cálculo del caudal Q; cálculo del diámetro D requerido
    - Sistemas de tuberías: comprobación del caudal Q para tuberías en serie, cálculo del caudal Q en tuberías en paralelo; cálculo del caudal Q en redes de tuberías.
- Flujo compresible de gases.
- Flujo a través de cuerpos sumergidos.

El primer problema se ubica en la unidad 5 de "Flujo de Fluidos Incompresibles en Ductos y Capas Delgadas", subunidades 5.3 "Pérdidas primarias en flujo en tuberías".
Todos los autores consultados determinan el factor de fricción a través del método gráfico del Diagrama de Moody.

El segundo problema cuenta con varios subproblemas que se ubican en la unidad 6 de "Cálculos para la Conducción de Fluidos Incompresibles", subunidades 6.1 "Cálculos de flujo en sistemas simples y múltiples de tuberías" y 6.2 "Cálculo del diámetro económico".
Los autores consultados utilizan el método de prueba y error para resolver los subproblemas de cálculo del caudal y del diámetro requerido para tuberías simples; y comprobación del caudal en tuberías en serie y tuberías en paralelo para sistemas de tuberías. Estos autores presentan ejemplos numéricos donde se consideran ya sea solo pérdidas primarias o pérdidas primarias y secundarias. Los ejemplos presentan los cálculos realizados dos o más de las iteraciones requeridas para resolver el problema y se pueden encontrar en:

Tuberías simples. Pérdidas primarias

Cálculo de la velocidad o del caudal: Mataix (1986, pp. 224–225); Foust et al. (2006, p. 556).

Cálculo del diámetro: Geankoplis (1998, pp. 103–104) y Pritchard y Mitchell (2015, pp. 316–318), Streeter et al. (2000, pp. 296–297) y Valiente Barderas (2002, pp. 294–295, 295–296).

Tuberías simples. Pérdidas primarias y secundarias

Cálculo del caudal: Pritchard y Mitchell (2015, pp. 315–316), McCabe et al. (2007, pp. 132–133) y Streeter (2000, pp. 301–302).

Cálculo del diámetro: Streeter et al. (2000, p. 303).

Tuberías en serie. Cálculo del caudal: Mataix (1986, pp. 262–265) y Streeter et al. (2000, pp. 551–552, 552–553).

Tuberías en paralelo. Cálculo de los caudales: Mataix (1986, pp. 265–267), Streeter et al. (2000, pp. 554–556) y Valiente Barderas (2002, pp. 296–300, 300–303).

En todos estos subproblemas se hace un uso del método de prueba y error.

Redes de tuberías. Cálculo de los caudales: Pritchard y Mitchell (2015, pp. 323–326), Mataix (1986, pp. 269–272), Streeter et al. (2000, pp. 560–561, 564–566) y Valiente y Barderas (2002, pp. 324–328, 329–340). En los problemas de redes de tuberías, se utiliza el método de Hardy-Cross. Pritchard y Mitchell (2015) ocupan software para resolver este tipo de problemas.

El tercer y cuarto problemas propuestos son demasiado generales, abarcando cada uno una unidad entera del programa de estudio de OPU-115. Así el tercero es equivalente a la unidad 7 “Flujo de fluidos compresibles” y el cuarto equivale a la unidad 8 “Flujo a través de cuerpos sumergidos”. Basándose en el cuarto criterio, estos dos problemas no se tomarán en cuenta para la elaboración de programas en Scilab y Spyder.

La Tabla 4.7 resume lo mencionado anteriormente.

Tabla 4.7. Problemas de OPU-115, luego de aplicación de criterios.

Métodos de resolución de ecuaciones no lineales preferidos por los docentes
➢ Método de sustitución sucesiva, ➢ Método de Newton-Raphson ➢ Métodos iterativos con las ecuaciones que se dan en la teoría
Problemas que implican ecuaciones no lineales
➢ Velocidad y caudal en tuberías simples considerando pérdidas primarias únicamente. ➢ Diámetro en tuberías simples, considerando pérdidas primarias únicamente. ➢ Velocidad y caudal en tuberías simples considerando pérdidas primarias y secundarias. ➢ Diámetro en tuberías simples, considerando pérdidas primarias y secundarias. ➢ Caudal en tuberías simples. ➢ Caudal en tuberías en paralelo. ➢ Caudal en redes de tuberías.

4.1.4. *Operaciones Unitarias III*

En caso de requerir un método numérico para la resolución de ecuaciones no lineales y/o sistemas de ecuaciones no lineales, los docentes de OPU-315 ocuparían:

- Método de Newton-Raphson
- Método de la secante
- Solución numérica de sistemas no lineales

Entre los paquetes de software preferidos están los programas de la calculadora TI-89, Matlab, Octave, Maxima, Scilab, Maple 2016; las hojas de cálculo de Excel, Mathcad, Datafit. Los docentes expresaron que podrían aplicar los programas elaborados en Scilab y Spyder, pero sería ventajoso si los mismos fueran aplicativos y desarrollados.

Los problemas de cálculo propuestos son:

- Aplicación del método de Micley para la determinación de las condiciones de salida del gas de una torre de enfriamiento.
- Problemas relacionados con destilación (cálculo de los puntos de burbuja y de rocío, cálculo de los valores K, *destilación instantánea*).
- Problemas relacionados con procesos de absorción.
- Solución de sistemas de ecuaciones no lineales.

El primer problema puede ubicarse en la unidad 5 “Operaciones de Humidificación”, subunidad 5.3 “Ecuaciones de diseño para torres de enfriamiento de agua y de humidificación de aire o gases”. Geankoplis (1998, p. 679), Foust et al. (2006, p. 448) y Coulson et al. (1999, p. 772) presentan la ecuación utilizada en el método de Micley, si bien este método se aplica de manera gráfica para obtener la temperatura de salida del gas proveniente de una torre de enfriamiento. Foust et al. (2006, pp. 450–451) aplica este método para la construcción de una curva y obtener la temperatura buscada. De acuerdo con el criterio 3, este problema propuesto no se considerará en la elaboración de los programas en Scilab y Spyder porque no se utiliza ningún método numérico de cálculo iterativo.

El segundo problema se ubica en la unidad 8 “Destilación”. El problema propuesto originalmente consistía en el cálculo de puntos de rocío y burbuja, de valores K, considerando procesos relacionados con la destilación. Si bien no se especifica el tipo de destilación a tomar en cuenta, la unidad 8 menciona la utilización de los métodos gráficos de McCabe-Thiele y Ponchon-Savarit. Estos métodos se utilizan únicamente en el diseño de columnas de destilación para la rectificación continua.

Además los autores consultados hacen uso de los valores K en destilación instantánea en mezclas multicomponentes. Entonces se elaborarán programas en Scilab y Spyder para procesos de destilación instantánea de mezclas multicomponentes, que corresponde a las subunidades 8.2 “Destilación instantánea (flash)” y 8.5 “Eficiencia en destilación. Elementos de destilación multicomponente”. Se eligió la destilación instantánea isotérmica para demostrar la aplicación de los métodos numéricos para la resolución de ecuaciones no lineales. Los resultados a un ejercicio de este tipo son proporcionados en Treybal (1986, p.

405) y McCabe et al. (2007, p. 775), mientras que Holland (1981, pp. 20–22) aplica un método de prueba y error para resolver el ejercicio presentado.

El tercer problema abarca una unidad completa del programa de estudio de OPU-315: la unidad 7 “Absorción de gases”, y en base al cuarto criterio no se considerará para la realización de programas en Scilab y Spyder. El cuarto problema propuesto no indica la aplicación específica en algún problema de OPU-315 y por lo tanto, según el criterio 4, no se considerará en la realización de programas en Scilab y Spyder. La Tabla 4.8 resume lo dicho anteriormente sobre OPU-315.

Tabla 4.8. Problemas de cálculo de OPU-315, luego de aplicación de criterios.

Métodos de resolución de ecuaciones no lineales preferidos por los docentes
➢ Método de Newton-Raphson ➢ Método de la secante ➢ Solución numérica de sistemas no lineales
Problemas que implican ecuaciones no lineales
➢ Composiciones de los productos de líquido y vapor, en procesos de destilación instantánea de mezclas multicomponentes

4.1.5. *Selección final de problemas*

El conjunto de problemas a resolver se redujo para hacer manejable la cantidad de problemas a resolver mediante Scilab y Spyder. Para ello, se consideró el número de autores que presentan cierto tipo de ejercicios numéricos en TQI-115, TQI-215, OPU-115 y OPU-315. Sólo se elegirán aquellos problemas que cuentan con la mayor cantidad de autores en cada asignatura. La aplicación de este criterio adicional se muestra en la Tabla 4.9.

Tabla 4.9. Aplicación de criterio adicional a problemas de cálculo

Problema	Problemas que implican ecuaciones no lineales	Autores[i]
TQI-115		
1	Volumen molar en ecuaciones cúbicas de estado, utilizando parámetros de la ecuación de Van der Waals, Redlich-Kwong, Soave-Redlich-Kwong, Peng-Robinson.	Smith et al. (2007); Reid et al. (1987)
2	Volumen molar mediante la ecuación de Kammerlingh-Onnes y las correlaciones para el segundo y el tercer coeficientes del virial.	Smith et al. (2007)
TQI-215		
3	Entalpía, entropía y energía de Gibbs residual de un gas puro, mediante la ecuación cúbica genérica.	Smith et al. (2007), Reid et al. (1987) y Poling et al. (2001).
4	Temperatura de una corriente gaseosa pura en procesos de flujo.	Smith et al. (2007).
5	Coeficiente de fugacidad de especies en solución, mediante la ecuación de Kammerlingh-Onnes y la correlación de Pitzer para el segundo coeficiente del virial.	Sandler (2006) y Smith et al. (2007).
6	Punto de burbuja T utilizando la ecuación de Raoult para mezclas binarias, ternarias y cuaternarias.	Abbott y Van Ness (1975) y Sandler (2006).

[i] Aquellos que resuelven un ejercicio de este tipo

Pasa…

Tabla 4.9. Aplicación de criterio adicional a problemas de cálculo (continuación)

Problema	Problemas que implican ecuaciones no lineales	Autores
TQI-215		
7	Punto de burbuja T utilizando la ecuación de Raoult modificada para mezclas binarias, ternarias y cuaternarias.	Abbott y Van Ness (2008); Poling et al. (2001) y Reid et al. (1987).
8	Punto de burbuja T utilizando la formulación gamma-phi para mezclas binarias, ternarias y cuaternarias.	Smith et al. (2007); Poling et al. (2001) y Reid et al. (1987).
9	Punto de burbuja T utilizando la formulación phi-phi para mezclas binarias, ternarias y cuaternarias y la ecuación cúbica genérica.	Smith et al. (2007).
10	Punto de rocío T utilizando la ecuación de Raoult para mezclas binarias, ternarias y cuaternarias.	Sandler (2006)
11	Punto de rocío T utilizando la ecuación de Raoult modificada para mezclas binarias, ternarias y cuaternarias.	Abbott y Van Ness (2008) y Smith et al. (2007).
12	Punto de rocío T utilizando la formulación gamma-phi para mezclas binarias, ternarias y cuaternarias.	---
13	Punto de rocío T utilizando la formulación phi-phi para mezclas binarias, ternarias y cuaternarias y la ecuación cúbica genérica.	Reid et al. (1987) y Poling et al. (2001).

Pasa…

Tabla 4.9. Aplicación de criterio adicional a problemas de cálculo (continuación)

Problema	Problemas que implican ecuaciones no lineales	Autores
TQI-215		
14	Composición de equilibrio para reacciones de mezclas de gases ideales	Abbott y Van Ness (1975); Smith et al. (2007); Sandler (2006) y Hougen y Watson (1947).
15	Composiciones de equilibrio para reacciones en soluciones ideales de gases	Smith et al. (2007) y Abbott y Van Ness (1975)
16	Composiciones de equilibrio para reacciones en mezclas gaseosas a altas presiones.	Hougen y Watson (1947).
17	Composiciones de equilibrio para múltiples reacciones en mezclas de gases ideales.	Abbott y Van Ness (1975); Hougen y Watson (1947); Sandler (2006) y Smith et al. (2007).
18	Composiciones de equilibrio para reacciones en soluciones ideales en fase líquida.	Sandler (2006) y Smith et al. (2007).
19	Composiciones de equilibrio para reacciones heterogéneas, que sólo consideran compuestos gaseosos	Sandler (2006)
20	Composiciones de equilibrio para reacciones heterogéneas, que consideran compuestos gaseosos y líquidos	Smith et al. (2007).

Pasa…

Tabla 4.9. Aplicación de criterio adicional a problemas de cálculo (continuación)

Problema	Problemas que implican ecuaciones no lineales	Autores
TQI-215		
21	Búsqueda de la temperatura en una reacción de descomposición de un compuesto sólido.	Hougen y Watson (1947)
22	Búsqueda de la temperatura en una reacción adiabática para mezclas de gases ideales.	Smith et al. (2007) y Abbott y Van Ness (1975).
OPU-115		
23	Velocidad y caudal en tuberías simples considerando pérdidas primarias únicamente.	Mataix (1986); Foust et al. (2006).
24	Diámetro en tuberías simples, considerando pérdidas primarias únicamente.	Geankoplis (1998); Pritchard y Mitchell (2015); Streeter et al (2000) y Valiente Barderas (2002).
25	Velocidad y caudal en tuberías simples considerando pérdidas primarias y secundarias.	Pritchard y Mitchell (2015); McCabe et al. (2007) y Streeter et al. (2000).
26	Diámetro en tuberías simples, considerando pérdidas primarias y secundarias.	Streeter et al. (2000)
27	Caudal en tuberías en serie.	Mataix (1986); Streeter et al. (2000)

Pasa…

Tabla 4.9. Aplicación de criterio adicional a problemas de cálculo (continuación)

Problema	Problemas que implican ecuaciones no lineales	Autores
OPU-115		
28	Caudal en tuberías en paralelo.	Mataix (1986); Streeter et al. (2000) y Valiente Barderas (2002)
29	Caudal en redes de tuberías	Pritchard y Mitchell (2015); Mataix (1986); Streeter et al. (2000) y Valiente Barderas (2002).
OPU-315		
30	Composiciones de los productos de líquido y vapor, en procesos de destilación instantánea de mezclas multicomponentes.	Treybal (1986); McCabe et al. (2007) y Holland (1981).

Las asignaturas de TQI-115 y OPU-315 contaron con un número reducido de problemas identificados (2 y 1 respectivamente) y no se realizó una mayor reducción de estos. En base a la frecuencia de los autores que resuelven un determinado problema, las Figuras 4.1 y 4.2 permitieron visualizar los problemas a resolver de las asignaturas de TQI-215 y de OPU-115. Se eligieron los problemas que presentaron las dos mayores frecuencias en TQI-215 y OPU-315. La Tabla 4.10 presenta el conjunto de problemas identificados. Además la Tabla 4.11, se resume los métodos numéricos conocidos o utilizados por los docentes encuestados en problemas de cálculo para la resolución de ecuaciones no lineales.

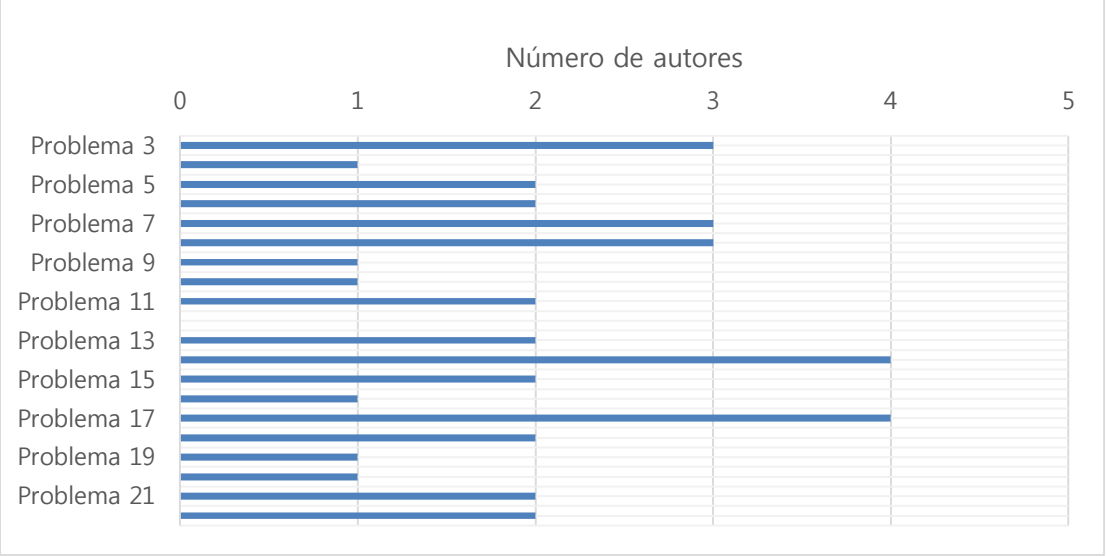


Figura 4.1. Número de autores que desarrollan los problemas identificados en la asignatura de TQI-215 del 3 al 21

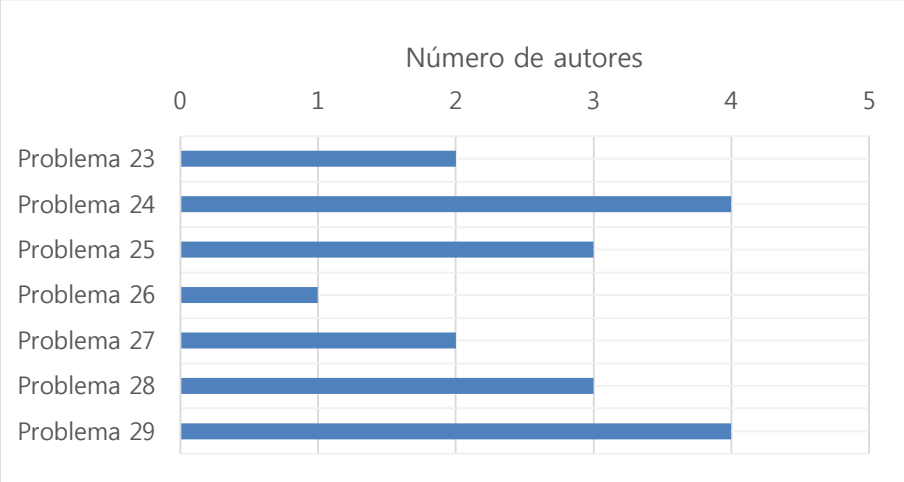


Figura 4.2. Número de autores que desarrollan los problemas identificados en la asignatura de OPU-115 del 23 al 29

Tabla 4.10. Conjunto de problemas de cálculo en las cuatro asignaturas de interés.

Asignatura	Número	Problemas a resolver mediante Scilab y Spyder
TQI-115	1°	Volumen molar en ecuaciones cúbicas de estado, utilizando parámetros de la ecuación de Van der Waals, Redlich-Kwong, Soave-Redlich-Kwong, Peng-Robinson.
	2°	Volumen molar mediante la ecuación de Kammerlingh-Onnes y las correlaciones para el segundo y el tercer coeficientes del virial.
TQI-215	3°	Entalpía, entropía y energía de Gibbs residual de un gas puro, mediante la ecuación cúbica genérica.
	4°	Punto de burbuja T utilizando la ecuación de Raoult modificada para mezclas binarias, ternarias y cuaternarias.
	5°	Punto de burbuja T utilizando la formulación gamma-phi para mezclas binarias, ternarias y cuaternarias.
	6°	Composición de equilibrio en reacciones para mezclas de gases ideales.
	----[ii]	Composición de equilibrio en múltiples reacciones para mezclas de gases ideales
OPU-115	7°	Diámetro en tuberías simples, considerando pérdidas primarias únicamente.
	8°	Velocidad y caudal en tuberías simples considerando pérdidas primarias y secundarias.
	9°	Caudal en tuberías en paralelo.

[ii] Por razones que se explicarán más adelante los problemas de composición de equilibrio en múltiples reacciones para mezclas de gases ideales y de caudal en redes de tuberías, no se tomaron en cuenta en la elaboración de programas en Scilab y Python

Pasa…

Tabla 4.10. Conjunto de problemas de cálculo en las cuatro asignaturas de interés (continuación).

Asignatura	Número	Problemas a resolver mediante Scilab y Spyder
OPU-115	----	Caudal en redes de tuberías
OPU-315	10°	Composiciones de los productos de líquido y vapor, en procesos de destilación instantánea de mezclas multicomponentes.

Tabla 4.11. Métodos de resolución de ecuaciones no lineales ocupadas por los docentes encuestados

Asignatura / Método utilizado	TQI-115	OPU-115	TQI-215	OPU-315
Método de Regula Falsi	X		X	
Método de sustitución sucesiva	X	X		
Método de Newton-Raphson	X	X	X	X
Método de la bisección			X	
Método de la secante			X	X
Método de prueba y error		X		

Enunciados de problemas seleccionados

Termodinámica Química I

Problema 1° (Basado en el ejercicio 3-3 de Reid et al. (1987, p. 46))

> "Calcule los volúmenes de líquido y vapor saturados y la presión de isobutano a 300 K por los siguientes métodos:
> (a) La ecuación de Soave-Redlich-Kwong; (b) La ecuación de Peng-Robinson"

Problema 2° (Basado en el ejercicio 3.10 de Smith et al. (2007, pp. 105–106))

> "Determine el volumen molar de n-butano a 510 K y 25 bar mediante la forma reducida de la ecuación del virial y las correlaciones generalizadas
> Calcule los volúmenes de líquido y vapor saturados y la presión de isobutano a 300 K por los siguientes métodos:
> (a) La ecuación de Soave-Redlich-Kwong; (b) La ecuación de Peng-Robinson"

Termodinámica Química II

Problema 3° (Basado en el ejercicio 6.4 de Smith et al. (2007, pp. 219–220))

> "Encuentre los valores para la entalpía H^R y la entropía S^R para el gas n-butano a 500 K y 50 bar, utilizando las siguientes ecuaciones de estado cúbicas:
> (a) La ecuación de Van der Waals; (b) La ecuación de Redlich-Kwong
> (c) La ecuación de Soave-Redlich-Kwong; (d) La ecuación de Peng-Robinson"

En la Tabla 4.12 se presentan las variables de ingreso, los valores asignados a dichas variables, y las variables respuesta en los ejemplos utilizados para los problemas 1°, 2° y 3°, respectivamente.

Tabla 4.12. Descripción de variables utilizadas en programas de los problemas 1°, 2° y 3°

Variables de ingreso	Valores asignados a variables	Variables respuesta
Problema 1		
Nombre de la sustancia de trabajo	Isobutano	Factor de compresibilidad de líquido ($Z_{liquido}$) y vapor saturado (Z_{vapor}). Volumen molar de saturación de líquido ($V_{m,liquido}$) y de vapor saturado ($V_{m,vapor}$).
Temperatura de saturación (T_i^{sat})	300 K	
Presión de saturación (P_i^{sat})	3.706 bar[20] (Soave-Redlich-Kwong); 3.680 bar (Peng-Robinson);	
Temperatura crítica (T_c)	408.2 K	
Presión crítica (P_c)	36.5 bar	
Factor acéntrico (ω)	0.183	
Problema 2		
Nombre de la sustancia de trabajo	n-butano	Factor de compresibilidad de gas (Z_{gas}). Volumen molar de gas ($V_{m,gas}$).
Temperatura (T)	510 K	
Presión (P)	25 bar	
Temperatura crítica (T_c)	425.1 K	
Presión crítica (P_c)	37.96 bar	
Factor acéntrico (ω)	0.200	

Pasa…

[20] Si el usuario no conoce la presión de saturación, es posible ingresar una presión mayor o menor a la de saturación. El programa indicará que la sustancia no se encuentra en saturación e imprimirá la presión de saturación calculada. El usuario puede entonces correr el programa con la presión calculada.

Tabla 4.12. Descripción de variables utilizadas en programas de los problemas 1°, 2° y 3° (continuación)

Variables de ingreso	Valores asignados a variables	Variables respuesta
Problema 3°		
Nombre de la sustancia de trabajo	n-butano	Entalpía residual (H^R). Entropía residual (S^R). Energía de Gibbs residual (G^R).
Temperatura (T)	500 K	
Presión (P)	50 bar	
Temperatura crítica (T_c)	425.1 K	
Presión crítica (P_c)	37.96 bar	
Factor acéntrico (ω)	0.200[21]	

Problema 4° (Basado en el ejercicio 8.16 de Poling et al. (2001, p. 8.104, 8.106-8.108))

> "Utilizando el método UNIFAC, calcule la temperatura y las composiciones en la fase líquida para el sistema ternario acetona (1)/2-butanona (2)/etil acetato (3) a 760 mmHg con x1 = 0.800 y x2 = 0.160.

Problema 5° (Basado en los datos de equilibrio de Smith et al. (2007, p. 550))

> "Utilizando el método UNIFAC, calcule la temperatura y las composiciones en la fase líquida para el sistema cuaternario etanol (1)/metilciclopentano (2)/benceno (3)/ hexano (4) a 760 mmHg con x1 = 0.068, x2 = 0.656 y x3 = 0.114.

En las Tablas 4.13 y 4.14 se presentan las variables de ingreso, los valores asignados a dichas variables, y las variables respuesta en los ejemplos utilizados para los problema 4° y 5°, respectivamente.

[21] En las ecuaciones de Van der Waals y de Redlich-Kwong no es necesario utilizar factor acéntrico.

Tabla 4.13. Descripción de variables utilizadas en programas del problema 4°

<table>
<tr><th>Variables de ingreso</th><th colspan="6">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td>Sistema</td><td colspan="6">acetona (1)/ 2-butanona (2)/ etil acetato (3)</td><td rowspan="6">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>Presión del sistema ($P_{sistema}$)</td><td colspan="6">101.325 Kpa</td></tr>
<tr><td>Valor inicial de temperatura (T)</td><td colspan="6">300 K</td></tr>
<tr><td>Composiciones molares de la fase líquida ($\{x_i\}$)</td><td colspan="6">x1 = 0.8; x2 = 0.16[22]</td></tr>
<tr><td rowspan="5">Constantes para el cálculo de las presiones de vapor</td><td colspan="6">Ecuación 1 de Reid et. al (1987)</td></tr>
<tr><td>A</td><td>B</td><td>C</td><td>D</td><td>T_c (K)</td><td>P_c (bar)</td></tr>
<tr><td colspan="6">Acetona</td></tr>
<tr><td>-7.45514</td><td>1.202</td><td>-2.43926</td><td>-3.3559</td><td>508.1</td><td>4700</td></tr>
<tr><td colspan="6">2-Butanona</td></tr>
<tr><td>-7.71476</td><td>1.71061</td><td>-3.6877</td><td>-0.75169</td><td>536.8</td><td>4210</td></tr>
</table>

Pasa…

[22] La tercera composición se determina al sumar las composiciones de los demás componentes y restando el resultado a 1.

Tabla 4.13. Descripción de variables utilizadas en programas del problema 4° (continuación)

<table>
<tr><th>Variables de ingreso</th><th colspan="6">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="11">Constantes para el cálculo de las presiones de vapor</td><td colspan="6">Ecuación 1 de Reid et. al (1987) (Wagner)</td><td rowspan="11">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>A</td><td>B</td><td>C</td><td>D</td><td>Tc (K)</td><td>Pc (bar)</td></tr>
<tr><td colspan="6">Etil acetato</td></tr>
<tr><td>-7.68521</td><td>1.36511</td><td>-4.0898</td><td>-1.75342</td><td>523.2</td><td>3830</td></tr>
<tr><td colspan="6">Ecuación 1 de Poling et al. (2001) (Antoine)</td></tr>
<tr><td colspan="2">A</td><td colspan="2">B</td><td colspan="2">C</td></tr>
<tr><td colspan="6">Acetona</td></tr>
<tr><td colspan="2">4.2184</td><td colspan="2">1197.01</td><td colspan="2">228.06</td></tr>
<tr><td colspan="6">2-Butanona</td></tr>
<tr><td colspan="2">4.1386</td><td colspan="2">1232.63</td><td colspan="2">218.69</td></tr>
<tr><td colspan="6">Etil acetato</td></tr>
<tr><td colspan="2">4.13361</td><td colspan="2">1195.13</td><td colspan="2">212.47</td></tr>
</table>

Pasa…

Tabla 4.13. Descripción de variables utilizadas en programas del problema 4° (continuación)

<table>
<tr><th>Variables de ingreso</th><th colspan="5">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="8">Constantes para el cálculo de las presiones de vapor</td><td colspan="5">Ecuación de Liley et. al (1999) (Riedel)</td><td rowspan="8">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>C1</td><td>C2</td><td>C3</td><td>C4</td><td>C5</td></tr>
<tr><td colspan="5">Acetona</td></tr>
<tr><td>69.006</td><td>-5599.6</td><td>-7.0985</td><td>6.2237E-6</td><td>2</td></tr>
<tr><td colspan="5">2-Butanona</td></tr>
<tr><td>72.698</td><td>-6143.6</td><td>-7.5779</td><td>5.6476E-6</td><td>2</td></tr>
<tr><td colspan="5">Etil acetato</td></tr>
<tr><td>66.824</td><td>-6227.6</td><td>-6.41</td><td>1.7914E-17</td><td>6</td></tr>
<tr><td rowspan="6">Constantes para el cálculo de los coeficientes de actividad (método UNIFAC)</td><td colspan="2">Componente</td><td colspan="2">Grupo de contribución</td><td>Frecuencia de repetición</td></tr>
<tr><td colspan="2" rowspan="2">1</td><td colspan="2">"CH3"</td><td>1</td></tr>
<tr><td colspan="2">"CH3CO"</td><td>1</td></tr>
<tr><td colspan="2" rowspan="3">2</td><td colspan="2">"CH3"</td><td>1</td></tr>
<tr><td colspan="2">"CH2"</td><td>1</td></tr>
<tr><td colspan="2">"CH3CO"</td><td>1</td></tr>
</table>

Pasa…

Tabla 4.13. Descripción de variables utilizadas en programas del problema 4° (continuación)

Variables de ingreso	Valores asignados a variables			Variables respuesta
Constantes para el cálculo de los coeficientes de actividad (método UNIFAC)	Componente	Grupo de contribución	Frecuencia de repetición	Temperatura de burbuja ($T_{burbuja}$)
	3	"CH3COO"	1	
		"CH2"	1	Composiciones molares de la fase vapor ($\{y_i\}$)
		"CH3"	1	

Pasa…

Tabla 4.14. Descripción de variables utilizadas en programas del problema 5°

<table>
<tr><th>Variables de ingreso</th><th colspan="6">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td>Sistema</td><td colspan="6">etanol (1)/ metilciclopentano (2)/ benceno (3)/ hexano (4)</td><td rowspan="5">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>Presión del sistema ($P_{sistema}$)</td><td colspan="6">101.325 Kpa</td></tr>
<tr><td>Valor inicial de temperatura (T)</td><td colspan="6">300 K</td></tr>
<tr><td>Composiciones molares de la fase líquida ($\{x_i\}$)</td><td colspan="6">x1 = 0.068; x2 = 0.656 y x3 = 0.114</td></tr>
<tr><td rowspan="4">Constantes para el cálculo de las presiones de vapor</td><td colspan="6">Ecuación 1 de Reid et. al (1987) (Wagner)</td></tr>
<tr><td>A</td><td>B</td><td>C</td><td>D</td><td>Tc (K)</td><td>Pc (bar)</td></tr>
<tr><td colspan="6">Etanol</td></tr>
<tr><td>-8.51838</td><td>0.34163</td><td>-5.73683</td><td>8.32581</td><td>513.9</td><td>6140</td></tr>
<tr><td></td><td colspan="6">Metilciclopentano</td></tr>
<tr><td></td><td>-7.15937</td><td>1.48017</td><td>-2.92482</td><td>-1.98377</td><td>532.7</td><td>3780</td></tr>
</table>

Pasa…

Tabla 4.14. Descripción de variables utilizadas en programas del problema 5° (continuación)

<table>
<tr><th>Variables de ingreso</th><th>Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="11">Constantes para el cálculo de las presiones de vapor</td><td>Ecuación 1 de Reid et. al (1987) (Wagner)</td><td rowspan="11">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>A | B | C | D | Tc (K) | Pc (bar)</td></tr>
<tr><td>Benceno</td></tr>
<tr><td>-6.98273 | 1.33213 | -2.62863 | -3.33399 | 562.2 | 4890</td></tr>
<tr><td>Hexano</td></tr>
<tr><td>-7.46765 | 1.44211 | -3.28222 | -2.50941 | 507.5 | 3010</td></tr>
<tr><td>Ecuación 1 de Poling et al. (2001)</td></tr>
<tr><td>A | B | C</td></tr>
<tr><td>Etanol</td></tr>
<tr><td>5.33675 | 1648.22 | 230.918</td></tr>
<tr><td>Metilciclopentano</td></tr>
<tr><td>4.18199 | 1295.543 | 238.39</td></tr>
</table>

Pasa…

Tabla 4.14. Descripción de variables utilizadas en programas del problema 5° (continuación)

Variables de ingreso	Valores asignados a variables					Variables respuesta
Constantes para el cálculo de las presiones de vapor	Ecuación 1 de Poling et al. (2001)					Temperatura de burbuja ($T_{burbuja}$) Composiciones molares de la fase vapor ($\{y_i\}$)
	A	B	C			
	Benceno					
	3.98523	1184.24	217.572			
	Hexano					
	4.00139	1170.875	224.317			
	Ecuación de Liley et. al (1999) (Riedel)					
	C1	C2	C3	C4	C5	
	Etanol					
	73.304	-7122.3	-7.1424	2.8853E-6	2	
	Metilciclopentano					
	55.368	-5149.8	-5.0136	3.222E-6	2	
	Benceno					
	83.107	-6486.2	-9.2194	6.9844E-6	2	

Pasa…

Tabla 4.14. Descripción de variables utilizadas en programas del problema 5° (continuación)

<table>
<tr><th>Variables de ingreso</th><th colspan="5">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="10">Constantes para el cálculo de las presiones de vapor</td><td colspan="5">Ecuación de Liley et. al (1999) (Riedel)</td><td rowspan="10">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>C1</td><td>C2</td><td>C3</td><td>C4</td><td>C5</td></tr>
<tr><td colspan="5">Hexano</td></tr>
<tr><td>104.65</td><td>-6995.5</td><td>-12.702</td><td>1.2381E-5</td><td>2</td></tr>
<tr><td colspan="5">Ecuación de Smith et al. (2007)[23]</td></tr>
<tr><td colspan="2">A</td><td colspan="2">B</td><td>C</td></tr>
<tr><td colspan="5">Etanol</td></tr>
<tr><td colspan="2">16.896933</td><td colspan="2">3803.98</td><td>231.47</td></tr>
<tr><td colspan="5">Metilciclopentano</td></tr>
<tr><td colspan="2">13.787333</td><td colspan="2">2731</td><td>226.04</td></tr>
<tr><td colspan="5">Benceno</td></tr>
<tr><td colspan="2">13.885833</td><td colspan="2">2788.51</td><td>220.79</td></tr>
</table>

Pasa…

[23] Dado que Smith et al. (2007) proporciona las constantes de Antoine para presiones en mmHg y temperaturas en K, éstas han sido ajustadas para trabajar con presiones en Kpa y temperaturas en °C.

Tabla 4.14. Descripción de variables utilizadas en programas del problema 5° (continuación)

<table>
<tr><td>Variables de ingreso</td><td colspan="3">Valores asignados a variables</td><td>Variables respuesta</td></tr>
<tr><td rowspan="3">Constantes para el cálculo de las presiones de vapor</td><td colspan="3">Ecuación de Smith et al. (2007)</td><td rowspan="3">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>A</td><td>B</td><td>C</td></tr>
<tr><td colspan="3">Hexano</td></tr>
<tr><td></td><td>13.821633</td><td>2697.55</td><td>224.37</td><td></td></tr>
<tr><td rowspan="11">Constantes para el cálculo de los coeficientes de actividad
(método UNIFAC)</td><td>Componente</td><td>Grupo de contribución</td><td>Frecuencia de repetición</td><td rowspan="11"></td></tr>
<tr><td rowspan="3">1</td><td>"CH3"</td><td>1</td></tr>
<tr><td>"CH2"</td><td>1</td></tr>
<tr><td>“OH”</td><td>1</td></tr>
<tr><td rowspan="3">2</td><td>"CH3"</td><td>1</td></tr>
<tr><td>"CH2"</td><td>4</td></tr>
<tr><td>"CH"</td><td>1</td></tr>
<tr><td>3</td><td>"ACH"</td><td>6</td></tr>
<tr><td rowspan="2">4</td><td>"CH3"</td><td>2</td></tr>
<tr><td>"CH2"</td><td>4</td></tr>
</table>

Pasa…

Tabla 4.14. Descripción de variables utilizadas en programas del problema 5° (continuación)

<table>
<tr><th>Variables de ingreso</th><th colspan="5">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="11">Coeficientes del Virial o propiedades requeridas en el cálculo de los coeficientes del virial</td><td colspan="5">Coeficientes calculados</td><td rowspan="11">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>Tc (K)</td><td>Pc (bar)</td><td>Vc (cm^3/mol)</td><td>Zc</td><td>ω</td></tr>
<tr><td colspan="5">Etanol</td></tr>
<tr><td>513.9</td><td>61.48</td><td>167</td><td>0.24</td><td>0.645</td></tr>
<tr><td colspan="5">Metilciclopentano</td></tr>
<tr><td>532.8</td><td>37.85</td><td>319</td><td>0.272</td><td>0.23</td></tr>
<tr><td colspan="5">Benceno</td></tr>
<tr><td>562.2</td><td>48.98</td><td>259</td><td>0.271</td><td>0.21</td></tr>
<tr><td colspan="5">Hexano</td></tr>
<tr><td>507.6</td><td>30.25</td><td>371</td><td>0.266</td><td>0.301</td></tr>
<tr><td colspan="5">Coeficientes dados</td></tr>
<tr><td colspan="2">B11</td><td colspan="2">B12</td><td>B13</td><td>B14</td></tr>
<tr><td colspan="2">-1174.7</td><td colspan="2">-621.8</td><td>-589.7</td><td>-657</td></tr>
</table>

Pasa…

Tabla 4.14. Descripción de variables utilizadas en programas del problema 5° (continuación)

<table>
<tr><td>Variables de ingreso</td><td colspan="4">Valores asignados a variables</td><td>Variables respuesta</td></tr>
<tr><td rowspan="6">Coeficientes del Virial o propiedades requeridas en el cálculo de los coeficientes del virial</td><td colspan="4">Coeficientes dados</td><td rowspan="6">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>B21</td><td>B22</td><td>B23</td><td>B24</td></tr>
<tr><td>-621.8</td><td>-1191.9</td><td>-1137.9</td><td>-1274.2</td></tr>
<tr><td>B31</td><td>B32</td><td>B33</td><td>B34</td></tr>
<tr><td>-589.7</td><td>-1137.9</td><td>-1086.9</td><td>-1218.8</td></tr>
<tr><td>B41</td><td>B42</td><td>B43</td><td>B44</td></tr>
<tr><td></td><td>-657</td><td>-1274.2</td><td>-1218.8</td><td>-1360.1</td><td></td></tr>
</table>

Pasa…

Problema 6° (Basado en el ejercicio 6 de Abbott y Van Ness (2008, pp. 4–37).

> "En la hidrogenación de benceno se produce ciclohexano a través de la reacción:
>
> $$C_6H_6 + 3H_2 \longrightarrow C_6H_{12}$$
>
> Para una temperatura de T = 600 K, una presión de P = 15 bar y razón molar de la corriente de alimentación de $H_2/C_6H_6 = 3$, determine las composiciones de equilibrio de los compuestos químicos".

En la Tabla 4.15 se presentan las variables de ingreso, los valores asignados a dichas variables, y las variables respuesta en el ejemplo utilizados para el problema 6°.

Operaciones Unitarias I

Problema 7° (Basado en el ejercicio 6.11 de Streeter et al. (2000, pp. 296–297))

> "Determinar el tamaño de una tubería de hierro dúctil limpio, requerida para transportar 4000 gpm (galones por minuto) de petróleo, $\nu = 0.0001$ pie^2/s; para 10000 pies con una pérdida de cabeza de 75 pies lb/lb".

Problema 8° (Basado en el ejercicio 3-3 de Reid et al. (1987, p. 46))

> "El petróleo crudo, que tiene una gravedad específica de 0.93 y una viscosidad de 4 cP, se descarga por gravedad desde el fondo de un tanque. La profundidad del líquido sobre la conexión de descarga en el tanque es de 6 m. La línea de descarga es una tubería de 3 in. norma 40 para la tubería. Su longitud es de 45 m, y contiene un codo de 90° y dos válvulas de compuerta. El crudo descarga a la atmósfera en un punto situado a 9 m por debajo del punto de conexión al tanque. ¿Qué velocidad de flujo, en metros cúbicos por hora, cabe esperar a través de la línea de descarga?

Tabla 4.15. Descripción de variables utilizadas en programas del problema 6°

Variables de ingreso	Valores asignados a variables	Variables respuesta
Reactivos	Benceno (C_6H_6), Hidrógeno (H_2)	Cantidades molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{n_i\}$). Composiciones molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{y_i\}$).
Producto	Ciclohexano (C_6H_{12})	
Presión del sistema ($P_{sistema}$)	15 bar	
Temperatura (T)	600 K	
Temperatura de referencia (T_{ref})	298.15 K	
Números estequiométricos $\lvert\nu_i\rvert$	Benceno: 1, Hidrógeno: 3, Ciclohexano: 1	
Razón molar de la corriente de alimentación	$H_2/C_6H_6 = 3$	
Calores de formación a 298 K, ($\Delta H^{\circ}_{f,298}$)	Benceno: 82630 J/mol Hidrógeno: 0 J/mol Ciclohexano: -123140 J/mol	
Energías de Gibbs de reacción a 298 K, ($\Delta G^{\circ}_{f,298}$)	Benceno: 129665 J/mol Hidrógeno: 0 J/mol Ciclohexano: 31920 J/mol	

Pasa…

Tabla 4.15. Descripción de variables utilizadas en programas del problema 6° (continuación)

<table>
<tr><th>Variables de ingreso</th><th colspan="5">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="11">Constantes para el cálculo de las presiones de vapor</td><td colspan="5">Ecuación de Reid et. al (1987)</td><td rowspan="11">Cantidades molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{n_i\}$).

Composiciones molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{y_i\}$).</td></tr>
<tr><td>A</td><td>B</td><td>C</td><td>D</td></tr>
<tr><td colspan="4">Benceno</td></tr>
<tr><td>-3.392E1</td><td>4.739E-1</td><td>-3.017E-4</td><td>7.130E-8</td></tr>
<tr><td colspan="4">Hidrógeno</td></tr>
<tr><td>2.714E1</td><td>9.274E-3</td><td>-1.381E-5</td><td>7.645E-9</td></tr>
<tr><td colspan="4">Ciclohexano</td></tr>
<tr><td>-5.454E1</td><td>6.113E-1</td><td>-2.523E-4</td><td>1.321E-8</td></tr>
<tr><td colspan="5">Ecuación de Poling et al. (2001)</td></tr>
<tr><td>a0</td><td>a1</td><td>a2</td><td>a3</td><td>a4</td></tr>
<tr><td colspan="5">Benceno</td></tr>
<tr><td>3.551</td><td>-6.184E-3</td><td>14.365E-5</td><td>-19.807E-8</td><td>8.234E-11</td></tr>
<tr><td colspan="5">Hidrógeno</td></tr>
<tr><td>2.883</td><td>3.681E-3</td><td>-0.772E-5</td><td>0.692E-8</td><td>-0.213E-11</td></tr>
</table>

Pasa…

Tabla 4.15. Descripción de variables utilizadas en programas del problema 6° (continuación)

Variables de ingreso	Valores asignados a variables					Variables respuesta
Constantes para el cálculo de las presiones de vapor	Ecuación de Poling et al. (2001)					Cantidades molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{n_i\}$).
	a0	a1	a2	a3	a4	
	Ciclohexano					
	4.035	-4.433E-3	16.384E-5	-20.775E-8	7.746E-11	
	Ecuación de Liley et. al (1999)					
	C1	C2	C3	C4	C5	Composiciones molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{y_i\}$).
	Benceno					
	3.551	-6.184E-3	14.365E-5	-19.807E-8	8.234E-11	
	Hidrógeno					
	2.883	3.681E-3	-0.772E-5	0.692E-8	-0.213E-11	
	Ciclohexano					
	4.035	-4.433E-3	16.384E-5	-20.775E-8	7.746E-11	

Pasa…

Tabla 4.15. Descripción de variables utilizadas en programas del problema 6° (continuación)

<table>
<tr><th>Variables de ingreso</th><th colspan="5">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="12">Constantes para el cálculo de las presiones de vapor</td><td colspan="5">Ecuación de Poling et al (2008)</td><td rowspan="12">Cantidades molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{n_i\}$).

Composiciones molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{y_i\}$).</td></tr>
<tr><td>C1</td><td>C2</td><td>C3</td><td>C4</td><td>C5</td></tr>
<tr><td colspan="5">Benceno</td></tr>
<tr><td>4.035</td><td>-4.433E-3</td><td>16.384E-5</td><td>-20.775E-8</td><td>7.746E-11</td></tr>
<tr><td colspan="5">Hidrógeno</td></tr>
<tr><td>4.035</td><td>-4.433E-3</td><td>16.384E-5</td><td>-20.775E-8</td><td>7.746E-11</td></tr>
<tr><td colspan="5">Ciclohexano</td></tr>
<tr><td>4.035</td><td>-4.433E-3</td><td>16.384E-5</td><td>-20.775E-8</td><td>7.746E-11</td></tr>
<tr><td colspan="5">Ecuación de Smith et al. (2007)</td></tr>
<tr><td colspan="2">A</td><td>B</td><td>C</td><td>D</td></tr>
<tr><td colspan="5">Benceno</td></tr>
<tr><td colspan="2">-0.206</td><td>39.064E-3</td><td>-13.301E-6</td><td>0E5</td></tr>
<tr><td colspan="5">Hidrógeno</td></tr>
<tr><td colspan="2">3.249</td><td>0.422E-3</td><td>0E6</td><td>0.083E5</td></tr>
</table>

Pasa…

Tabla 4.15. Descripción de variables utilizadas en programas del problema 6° (continuación)

<table>
<tr><th>Variables de ingreso</th><th colspan="4">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="4">Constantes para el cálculo de las presiones de vapor</td><td colspan="4">Ecuación de Smith et al. (2007)</td><td rowspan="4">Cantidades molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{n_i\}$).

Composiciones molares de equilibrio de los compuestos químicos (benceno, hidrógeno y ciclohexano), ($\{y_i\}$).</td></tr>
<tr><td>A</td><td>B</td><td>C</td><td>D</td></tr>
<tr><td colspan="4">Ciclohexano</td></tr>
<tr><td>-3.876</td><td>63.249E-3</td><td>-20.928E-6</td><td>0E5</td></tr>
</table>

Problema 9° (Basado en el ejercicio 3-3 de Reid et al. (1987, p. 46))

> "Por un sistema de conducción de agua a 20 °C formado por dos tuberías de hierro fundido, con un caudal total de 456 l/s. Determinar el caudal a través de cada tubería. La tubería 1 tiene una longitud de 1500 m y un diámetro de 12 in. cd 40 y la tubería 2 tiene una longitud de 900 m y un diámetro de 16 in. cd 40."

En la Tabla 4.16 se presentan las variables de ingreso, los valores asignados a dichas variables, y las variables respuesta en los ejemplos utilizados para los problemas 7°, 8° y 9°, respectivamente.

Operaciones Unitarias III

Problema 10° (Basado en el ejercicio 3-3 de Reid et al. (1987, p. 46))

> "Calcule los volúmenes de líquido y vapor saturados y la presión de isobutano a 300 K por los siguientes métodos:
> (a) La ecuación de Soave-Redlich-Kwong; (b) La ecuación de Peng-Robinson"

En la Tabla 4.17 se presentan las variables de ingreso, los valores asignados a dichas variables, y las variables respuesta en el ejemplo utilizado para el problema 10°.

Tabla 4.16. Descripción de variables utilizadas en programas de los problemas 7°, 8° y 9°

Variables de ingreso	Valores asignados a variables	Variables respuesta
Problema 7		
Nombre de la sustancia de trabajo	Petróleo	Diámetro de la tubería (D)
Caudal (Q)	4000 gpm	
Viscosidad cinemática (ν)	0.0001 pies^2/s	
Pérdida de cabeza debido a fricción superficial ($h_{L,s}$)	75 pie lbf/lb	
Longitud de la tubería (L)	10000 pie	
Rugosidad absoluta (ε)	0.00015 pie	
Problema 8		
Nombre de la sustancia de trabajo	Petróleo crudo	Caudal (Q)
Diámetro de la tubería (D)	3 in. norma 40	
Viscosidad dinámica (μ)	4 cP	
Densidad (ρ)	928 kg/m^3	
Pérdida de cabeza (h_L)	75 pie lbf/lb	
Longitud de la tubería (L)	45 m	
Rugosidad absoluta (ε)	0.00015 pie	
Diferencia de altura (z_1-z_2)	15 m	
Coeficientes K debido a accesorios de tuberías	1.49	

Pasa…

Tabla 4.16. Descripción de variables utilizadas en programas de los problemas 7°, 8° y 9° (continuación)

Variables de ingreso	Valores asignados a variables	Variables respuesta
Problema 9°		
Nombre de la sustancia de trabajo	Agua	Caudal de la tubería 1 (Q_1) y de la tubería (Q_2).
Temperatura (T)	20 °C	
Longitudes de las tuberías (L)	Tubería 1: 1500 m Tubería 2: 900m	
Diámetros de las tuberías (D)	Tubería 1: 12 in Tubería 2: 16 in	
Caudal (Q)	456 L/s	

4.2. Resultados de los programas elaborados

Se observaron un conjunto de diferencias y semejanzas en la escritura de programas en Scilab y Python, pero el funcionamiento de los programas en Scilab y Python es básicamente el que se describen en los anexos A y C.

La resolución de los problemas de cálculo se llevó a cabo utilizando los siguientes métodos numéricos de resolución de ecuaciones no lineales: Regula Falsi, Newton-Raphson, Regula Falsi modificado, de la secante y de iteración de punto fijo. Se descartaron dos problemas de cálculo de los doce problemas originales porque requerían mayor experticia en el área de resolución de sistemas de ecuaciones no lineales (problema de composiciones de equilibrio en múltiples reacciones para mezclas de gases ideales) y de elaboración de interfaces gráficas de usuario (problema de cálculo de caudales en redes de tuberías).

Los porcentajes de error entre los resultados obtenidos en los programas y los proporcionados en las referencias fueron variables para cada asignatura.

Tabla 4.17. Descripción de variables utilizadas en programas del problema 10°

Variables de ingreso	Valores asignados a variables						Variables respuesta
Sistema	benceno (1)/ tolueno (2)/ o-xileno (3)						Temperatura de burbuja ($T_{burbuja}$)
Presión del sistema ($P_{sistema}$)	1 atm						
Temperatura (T)	100 °C						Composiciones molares de la fase vapor ($\{y_i\}$)
Composiciones molares globales ($\{z_i\}$)	x1 = 0.5; x2 = 0.25 y x3 = 0.25						
Constantes para el cálculo de las presiones de vapor	Ecuación 1 de Reid et. al (1987)						
	A	B	C	D	T_c (K)	P_c (bar)	
	Benceno						
	-6.98273	1.33213	-2.62863	-3.33399	562.2	48.90	
	Tolueno						
	-7.28607	1.38091	-2.83433	-2.79168	591.8	41.00	

Pasa…

Tabla 4.13. Descripción de variables utilizadas en programas del problema 4°

<table>
<tr><th>Variables de ingreso</th><th colspan="6">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="11">Constantes para el cálculo de las presiones de vapor</td><td colspan="6">Ecuación 1 de Reid et. al (1987) (Wagner)</td><td rowspan="11">Temperatura de burbuja ($T_{burbuja}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>A</td><td>B</td><td>C</td><td>D</td><td>Tc (K)</td><td>Pc (bar)</td></tr>
<tr><td colspan="6">o-Xileno</td></tr>
<tr><td>-7.53357</td><td>1.40968</td><td>-3.10985</td><td>-2.85992</td><td>630.3</td><td>37.30</td></tr>
<tr><td colspan="6">Ecuación 1 de Poling et al. (2001) (Antoine)</td></tr>
<tr><td colspan="2">A</td><td colspan="2">B</td><td colspan="2">C</td></tr>
<tr><td colspan="6">Benceno</td></tr>
<tr><td colspan="2">3.98523</td><td colspan="2">1184.24</td><td colspan="2">217.572</td></tr>
<tr><td colspan="6">Tolueno</td></tr>
<tr><td colspan="2">4.0543</td><td colspan="2">1327.62</td><td colspan="2">217.625</td></tr>
<tr><td colspan="6">o-Xileno</td></tr>
<tr><td colspan="2">4.09789</td><td colspan="2">1458.706</td><td colspan="2">212.041</td></tr>
</table>

Pasa…

Tabla 4.13. Descripción de variables utilizadas en programas del problema 4°

Variables de ingreso	Valores asignados a variables						Variables respuesta
Constantes para el cálculo de las presiones de vapor	Ecuación 3 de Poling et al. (2001) (Antoine)						Composiciones molares de la fase vapor ($\{y_i\}$)
	Tc (K)	a	b	c	d	Pc (bar)	
	Benceno						Composiciones molares de la fase vapor ($\{y_i\}$)
	562.16	-7.01433	1.55256	-1.8479	-3.7130	48.98	
	Tolueno						
	591.8	-7.316	1.59425	-1.93165	-3.7222	41.06	
	o-Xileno						
	630.33	-7.60491	1.75383	-2.27531	-3.73771	3735	
	Ecuación de Liley et. al (1999) (Riedel)						
	C1	C2	C3	C4	C5		
	Benceno						
	83.918	-6517.7	-9.3453	7.1182E-6	2		
	Tolueno						
	80.877	-6902.4	-8.7761	5.80347E-6	2		

Pasa…

Tabla 4.13. Descripción de variables utilizadas en programas del problema 4°

<table>
<tr><th>Variables de ingreso</th><th colspan="5">Valores asignados a variables</th><th>Variables respuesta</th></tr>
<tr><td rowspan="10">Constantes para el cálculo de las presiones de vapor</td><td colspan="5">Ecuación de Liley et. al (1999) (Riedel)</td><td rowspan="10">Composiciones molares de la fase vapor ($\{y_i\}$)

Composiciones molares de la fase vapor ($\{y_i\}$)</td></tr>
<tr><td>C1</td><td>C2</td><td>C3</td><td>C4</td><td>C5</td></tr>
<tr><td colspan="5">o-Xileno</td></tr>
<tr><td>90.356</td><td>-7948.7</td><td>-10.08</td><td>1,5.9756E-6</td><td>2</td></tr>
<tr><td colspan="5">Ecuación de Smith et al. (2007)[24]</td></tr>
<tr><td colspan="2">A</td><td colspan="2">B</td><td>C</td></tr>
<tr><td colspan="5">Benceno</td></tr>
<tr><td colspan="2">13.7819</td><td colspan="2">2726.81</td><td>217.572</td></tr>
<tr><td colspan="5">Tolueno</td></tr>
<tr><td colspan="2">13.932</td><td colspan="2">3056.96</td><td>217.625</td></tr>
<tr><td colspan="5">o-Xileno</td></tr>
<tr><td colspan="2">14.0415</td><td colspan="2">3358.79</td><td>212.041</td></tr>
</table>

[24] Dado que Smith et al. (2007) proporciona las constantes de Antoine para presiones en mmHg y temperaturas en K, éstas han sido ajustadas para trabajar con presiones en Kpa y temperaturas en °C.

4.2.1. *Semejanzas y diferencias entre Scilab y Python*

Durante la elaboración de los programas, pudo observarse una serie de semejanzas y diferencias en el uso de Scilab y Python, las cuales se muestran en la Tabla 4.15.

Tabla 4.18. Comparación entre las características de Scilab y Python en la elaboración de programas informáticos.

Semejanzas entre Scilab y Python
• Los programas de Scilab y Python no utilizan llaves para abrir y cerrar condicionales if, ciclos while y for y funciones. • Scilab y Python permiten utilizar funciones como argumentos de otras funciones. • Scilab y Python cuentan con estructuras para la creación de listas y diccionarios (struct en Scilab, dict en Python). • La depuración, se facilitó en ambos software tras permitir en los programas el ingreso de un dato a la vez: número o carácter en Scilab o carácter en Python.
Diferencias entre Scilab y Python
• Python permite un mayor economía de código en cuanto que los programas no utilizan “end” al final de condicionales, ciclos while y for y funciones. • En Scilab si es necesario utilizar “end”. • Python permite la creación de clases y de objetos. Scilab no cuenta con dicha característica. • Los recursos de consulta disponibles en la Web son más extensos en el caso de Python que de Scilab. • Python no permite anular ciclos infinitos while, aunque si cuenta con la opción de abrir una terminal Ipython sin necesidad de cerrar Spyder. Scilab si permite anular ciclos while infinitos.

Pasa…

Tabla 4.16. Comparación entre las características de Scilab y Python en la elaboración de programas informáticos (continuación).

Diferencieas entre Scilab y Python
• Python no permite ingresar datos distintos de caracteres o listas de caracteres. Para convertir estos caracteres es necesario utilizar los comandos int o float y realizar la adición de los elementos numéricos convertidos a listas. • Finalmente, cualquier tipo de error en códigos demasiado extensos para los programas generados en Scilab, genera un mensaje sobre "Stacking problem" (problema de apilamiento) que impide cualquier tipo de acción en Scilab y es necesario cerrar y abrir la consola de dicho software. Python no cuenta con esta dificultad.

4.2.2. *Programas elaborados*

La interfase de los programas generados, tanto en Scilab como en Python, consiste en la impresión de mensajes para el ingreso e impresión de datos en las consolas de Scilab y Python. La Tabla 4.19 presenta los métodos numéricos utilizados en los distintos problemas de cálculo identificados y la Figura 4.3 presentan un ejemplo de la interfase desarrollada, ejecutando el tercer problema.

A continuación se presentan los resultados obtenidos con los programas elaborados, comparándolos con las respuestas dadas en ejemplos numéricos de las referencias bibliográficas.

Tabla 4.19. Métodos numéricos utilizados para la resolución de los problemas de cálculo identificados en las asignaturas de TQI-115, TQI-215, OPU-115 y OPU-315.

Número identificador del problema para las asignaturas				Método de resolución de ecuaciones no lineales
TQI-115	TQI-215	OPU-115	OPU-315	
1, 2	--	--	--	Método de Newton-Raphson
1, 2	--	--	--	Método de Regula-Falsi
--	6	--	--	Método de Regula-Falsi modificado
1, 2	--	--	--	Método de iteración de punto fijo
--	3, 4, 5	7, 8, 9	10	Método de la secante

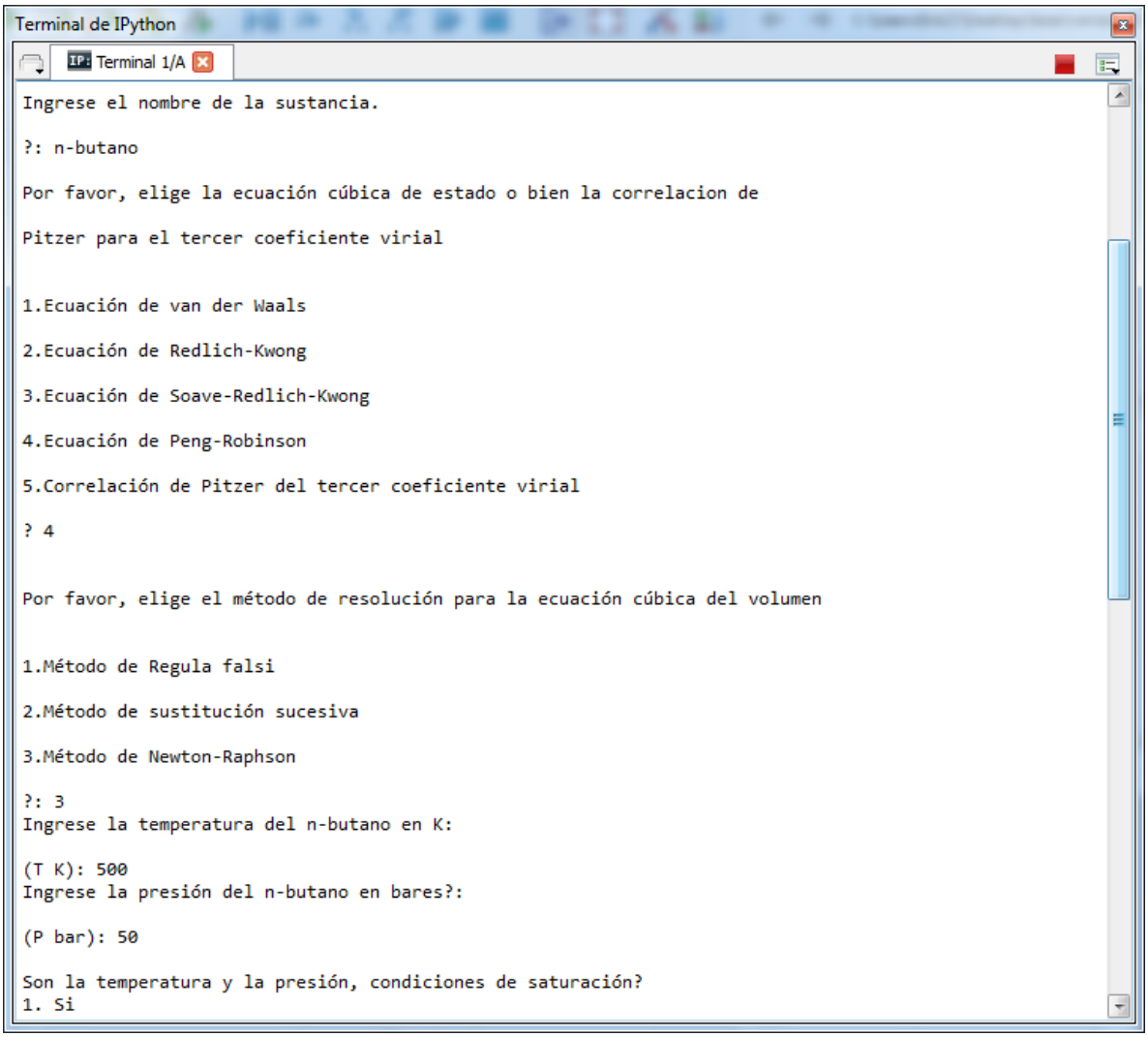


Figura 4.3. Ejemplo de interfase de programa desarrollado.

Termodinámica Química I

Problema 1°

La Tabla 4.20 presenta los volúmenes obtenidos para vapor y líquido saturado. Sin importar el método ocupado para la resolución de ecuaciones no lineales (Regula Falsi, Newton-Raphson o Sustitución Sucesiva), el tipo de ecuación cúbica en el volumen ocupada

(ecuación de Van der Waals, ecuación de Redlich-Kwong, ecuación de Soave-Redlich-Kwong o ecuación de Peng-Robinson) o el software utilizado (Scilab o Python), los volúmenes de líquido y vapor saturado tienen un error menor del 1 % con respecto a las proporcionados en las referencias mencionadas.

Tabla 4.20. Volúmenes en saturación, calculados con ecuación cúbica genérica

Tipo de ecuación	Volumen (cm^3/mol)				% Error	
	Respuesta base[iii]	Metodo utilizado	Programa elaborado		Scilab	Python
			Scilab	Python		
Vapor saturado						
Ecuacion de Soave-Redlich-Kwong	6096	Regula Falsi Sustitucion sucesiva Newton-Raphson	6096	6096	0.00	0.00
Ecuacion de Peng-Robinson	6105	Regula Falsi Sustitucion sucesiva Newton-Raphson	6111	6111	0.10	0.10
Líquido saturado						
Ecuacion de Soave-Redlich-Kwong	113.5	Regula Falsi Sustitucion sucesiva Newton-Raphson	113.5	113.5	0.00	0.00
Ecuacion de Peng-Robinson	100.2	Regula Falsi	100.2	100.2	0.00	0.00
		Sustitucion sucesiva	100.1	100.1	0.10	0.10
		Newton-Raphson	100.2	100.2	0.00	0.00

[iii] Se ocupó el ejemplo 3-3 de Reid et al. (1987, p. 46).

Problema 2°

La Tabla 4.21 presenta las respuestas calculadas en Scilab y Python; tienen un error menor del 1 % con respecto a las proporcionadas en la referencia consultada, sin importar el método ocupado para la resolución de ecuaciones no lineales: Regula Falsi, Newton-Raphson o

Sustitución Sucesiva; es indiferente el tipo de ecuación cúbica en el volumen utilizada (ecuación de Van der Waals, ecuación de Redlich-Kwong, ecuación de Soave-Redlich-Kwong o ecuación de Peng-Robinson).

Tabla 4.21. Volúmenes de una sustancia gaseosa calculada, ecuación del virial en términos del volumen

Tipo de ecuación	Volumen para gas (cm^3/mol)				% Error	
	Respuesta base[iv]	Metodo utilizado	Programa elaborado		Scilab	Python
			Scilab	Python		
Virial en términos del volumen y correlaciones tipo Pitzer para el segundo y tercer coeficiente de virial	1485.8	Regula Falsi	1485.0	1485.0	0.05	0.05
		Newton-Raphson	1485.0	1485.0	0.05	0.05
		Sustitución sucesiva	1485.0	1485.0	0.05	0.05

[iv] Se ocupó el ejercicio 3.10 de Smith et al. (2007, pp. 105–106)

Durante la revisión por parte del docente, se determinó que los programas elaborados para TQI-115 arrojan respuestas erróneas en la cercanía del punto crítico para la sustancia considerada. Además se efectuaron algunos cambios en el uso de la ecuación del virial, dado que se obtenían respuestas equivocadas en condiciones de saturación.

La evaluación de los factores de compresibilidad y volúmenes de saturación de las sustancias, haciendo uso de la ecuación de estado general cúbica para temperaturas menores a la crítica fue satisfactoria. Los valores calculados para el estado de sobrecalentamiento también son correctos.

Además el programa principal debió modificarse en cuanto al ingreso de datos por parte del usuario: falta de capacidad del programa principal para identificar las sustancias involucradas en los cálculos, siendo posible que el usuario cometiera errores; el ingreso de números para

la identificación de opciones era problemático porque no se detallaba cual era el tipo de datos a ingresar; también el programa no permitía reingresar los datos en caso de haberse equivocado el usuario; y las unidades de las presiones y temperaturas no se detallaban. Los programas desarrollados para el cálculo del volumen molar en ecuaciones cúbicas de estado, utilizando parámetros de la ecuación de Van der Waals, Redlich-Kwong, Soave-Redlich-Kwong, Peng-Robinson, de TQI-115, no pudieron ser modificados en cuanto a las respuestas erróneas cerca del punto crítico.

Termodinámica Química II

Para TQI-215, se había contemplado originalmente la elaboración de un conjunto de programas para la obtención de composiciones de equilibrio en múltiples reacciones para mezclas de gases ideales. Dicho problema es más complicado porque requiere de un método numérico que permita la restricción de los posibles valores calculados y cuente con un proceso de búsqueda óptima de dicho valores, tal y como se describe en Chapra y Canale (2007, pp. 409–410). Se decidió no elaborar dicho conjunto de programas porque habría implicado un uso excesivo de tiempo. Para la resolución de este tipo de ejercicios, resultaría más conveniente la utilización de un paquete de software comercial como Mathcad o Maple que ya incluyen este tipo métodos numéricos en sus rutinas, o la exploración de bibliotecas en Python con métodos numéricos establecidos como la biblioteca de Numpy o Scipy.

Problema 3°

De acuerdo con la Tabla 4.22, los porcentajes de error obtenidos entre los valores calculados en Scilab y Python y los de la referencia son menores al 1%. Esto aplica incluso para los diferentes tipos de ecuación cúbica en el volumen ocupados (ecuación de Van der Waals, ecuación de Redlich-Kwong, ecuación de Soave-Redlich-Kwong o ecuación de Peng-Robinson).

Tabla 4.22 Entalpía y entropía residuales calculados, ecuación cúbica genérica.

Propiedad Residual	Tipo de ecuación	Respuesta base[v]	Respuestas obtenidas en los programas elaborados		% Error	
			Scilab	Python	Scilab	Python
Entalpía residual (J/mol)	Van der Waals	-3937	-3935	-3935	0.05	0.05
	Redlich-Kwong	-4505	-4503	-4503	0.04	0.04
	Soave-Redlich-Kwong	-4824	-4821	-4821	0.06	0.06
	Peng-Robinson	-4988	-4985	-4985	0.06	0.06
Entropía residual (J/mol K)	Van der Waals	-5.424	-5.420	-5.420	0.07	0.07
	Redlich-Kwong	-6.546	-6.542	-6.542	0.06	0.06
	Soave-Redlich-Kwong	-7.413	-7.408	-7.408	0.07	0.07
	Peng-Robinson	-7.426	-7.421	-7.421	0.07	0.07

[v] Se ocupó el ejercicio 6.4 de Smith et al. (2007, pp. 105–106)

Problema 4°

Los resultados se muestran en las Tabla 4.23 - Tabla 4.25. Poling et al presentan los cálculos para varias composiciones, pero se eligió las composiciones x1 = 0.8, x2 = 0.5 y x3 = 0.14 para observar el grado de concordancia entre los programas elaborados y los resultados dados en los libros. En el ejemplo se utiliza el método UNIFAC para el cálculo de los coeficientes de actividad.

En la Tabla 4.23 y Tabla 4.24, se muestran los valores de composición molar para dos de los tres componentes de la mezcla, utilizando la ecuación de Wagner, Antoine, Wagner modificada, y Riedel para el cálculo de presiones de vapor. El tercer componente no se muestra porque su composición es posible calcularla fácilmente de la suma de los otros componentes presentes. La Tabla 4.25 muestra las temperaturas calculadas.
Las composiciones molares de acetona en la fase gaseosa de la Tabla 4.23 poseen porcentajes de error menores al 1% tanto para Scilab como Python; la Tabla 4.24 muestra que las composiciones molares de 2-butanona en la fase gaseosa poseen un porcentaje de error menores o iguales al 1.23% tanto para Scilab como para Python; además las temperaturas obtenidas en la Tabla 4.25, tanto en Scilab como en Python, presentan un porcentaje de error menores al 1%.

Tabla 4.23. Punto de burbuja de un sistema ternario. Composición del componente 1

Tipo de ecuación	Composición molar				% Error	
	x_1 (acetona)	y_1				
		Respuesta base[vi]	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.8	0.896	0.898	0.896	0.22	0.00
Ecuación 1 de Poling et. al (2001)	0.8	0.896	0.898	0.895	0.22	0.11
Riedel de Liley et al. (1999)	0.8	0.896	0.897	0.895	0.11	0.11

[vi] Se ocupó el ejercicio 8-16 de Poling et al. (2001, p. 8.104-8.108)

Tabla 4.24. Punto de burbuja de un sistema ternario. Composición del componente 2

Tipo de ecuación	Composición molar				% Error	
	x_2 (2-Butanona)	y_2				
		Respuesta base	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.16	0.081	0.080	0.080	1.23	1.23
Ecuación 1 de Poling et. al (2001)	0.16	0.081	0.081	0.081	0.00	0.00
Riedel de Liley et al. (1999)	0.16	0.081	0.081	0.080	0.00	1.23

Tabla 4.25. Punto de burbuja de un sistema ternario. Temperatura

Tipo de ecuación	Temperatura en K			% Error	
	Respuesta base	Respuesta obtenida			
		Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	332.55	332.66	332.58	0.03	0.01
Ecuación 1 de Poling et. al (2001)	332.55	332.65	332.57	0.03	0.01
Riedel de Liley et al. (1999)	332.55	332.68	332.59	0.04	0.01

Problema 5°

Smith et al (2007) calcularon los coeficientes de actividad del sistema etanol (1)/ metilciclopentano (2)/ benceno (3)/ n-hexano (4) utilizando el método UNIFAC. Los resultados obtenidos se presentan en las Tabla 4.26 - Tabla 4.33.

En la Tabla 4.26, Tabla 4.27 y Tabla 4.28, se muestran los valores de composición molar de los primeros tres componentes de la mezcla, ocupando las ecuaciones mencionadas para el calculo de la presión de vapor. Los coeficientes del virial, necesarios para calcular los coeficientes φ de fugacidad, se obtuvieron ocupando las siguientes propiedades: presiones críticas, volúmenes críticos, temperaturas críticas, factores de compresibilidad críticos y los factores acentricos[25]. La Tabla 4.29 muestra las temperaturas calculadas.

Las composiciones molares de etanol en la fase gaseosa muestran un porcentaje de error entre el 1.79% y el 2.51% para Scilab y entre el 1.79% y el 2.15% para Python (véase Tabla 4.26); el error porcentual de las composiciones molares del metilciclopentano son menores al 1%, tanto en Scilab y Python (véase Tabla 4.27); las composiciones molares del benceno en la

[25] Estas propiedades pueden consultarse en el Apéndice B de Smith et al. (2007)

fase gaseosa son del 1.22%, para Scilab y Python (véase Tabla 4.28). Los porcentajes de error de las temperaturas obtenidas, tanto en Scilab como en Python, son menor al 1% (véase Tabla 4.29).

En la Tabla 4.30, Tabla 4.31 y Tabla 4.32, se muestran los valores de composición molar de los primeros tres componentes de la mezcla, ocupando las ecuaciones mencionadas para el cálculo de la presión de vapor y utilizando los coeficientes del virial proporcionados por Smith et al. La Tabla 4.33 muestra las temperaturas calculadas.

Tabla 4.26. Punto de burbuja de un sistema cuaternario, calculando coeficientes del virial. Composición y del componente 1.

Tipo de ecuación	Composición molar				% Error	
	x_1 (Etanol)	y_1				
		Respuesta base[vii]	Respuestas obtenidas[viii]			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.068	0.279	0.286	0.285	2.51	2.15
Ecuación 1 de Poling et. al (2001)	0.068	0.279	0.285	0.285	2.15	2.15
Riedel de Liley et al. (1999)	0.068	0.279	0.285	0.285	2.15	2.15
Antoine de Smith et al (2007).	0.068	0.279	0.284	0.284	1.79	1.79

[vii] Se ocuparon los datos de equilibrio de el ejercicio 8-16 de Smith et al. (2007, p. 550)

[viii] Se utilizaron las propiedades de factores acéntricos, así como las temperaturas, presiones, factores de compresibilidad y volúmenes críticos para el cálculo de los coeficientes del virial. Dichas propiedades se obtuvieron de Smith et. al (2007, p. Apéndice B).

Tabla 4.27. Punto de burbuja de un sistema cuaternario, calculando coeficientes del virial. Composición y del componente 2.

Tipo de ecuación	Composición molar				% Error	
	x_2 (Metilciclo-pentano)	y_2				
		Respuesta base	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.656	0.5	0.496	0.496	0.80	0.80
Ecuación 1 de Poling et. al (2001)	0.656	0.5	0.496	0.496	0.80	0.80
Riedel de Liley et al. (1999)	0.656	0.5	0.497	0.497	0.60	0.60
Antoine de Smith et al (2007).	0.656	0.5	0.497	0.497	0.60	0.60

Las composiciones molares de etanol en la fase gaseosa muestran un porcentaje de error entre el 1.08% y el 1.43% para Scilab y Python (véase Tabla 4.30); el error porcentual de las composiciones molares del metilciclopentano es menor al 1% para Scilab y para Python (véase Tabla 4.31); las composiciones molares calculadas del n-hexano son igual o menores al 1.22% para Scilab y para Python (véase Tabla 4.32). Además, de acuerdo con la Tabla 4.33), las temperaturas obtenidas, tanto en Scilab como en Python, presentan un porcentaje de error menor al 1%.

Tabla 4.28. Punto de burbuja de un sistema cuaternario, calculando coeficientes del virial. Composición y del componente 3

Tipo de ecuación	Composición molar				% Error	
	x_3 (Benceno)	y_3				
		Respuesta base	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.114	0.082	0.081	0.081	1.22	1.22
Ecuación 1 de Poling et. al (2001)	0.114	0.082	0.081	0.081	1.22	1.22
Riedel de Liley et al. (1999)	0.114	0.082	0.081	0.081	1.22	1.22
Antoine de Smith et al (2007).	0.114	0.082	0.081	0.081	1.22	1.22

Tabla 4.29. Punto de burbuja de un sistema cuaternario, calculando coeficientes del virial. Temperatura

Tipo de ecuación	Temperatura en K			% Error	
	Respuesta base	Respuesta obtenida			
		Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	334.82	334.61	334.61	0.06	0.06
Ecuación 1 de Poling et. al (2001)	334.82	334.61	334.61	0.06	0.06
Riedel de Liley et al. (1999)	334.82	334.55	334.55	0.08	0.08
Antoine de Smith et al (2007).	334.82	334.60	334.60	0.07	0.07

Tabla 4.30. Punto de burbuja de un sistema cuaternario, coeficientes del virial dados. Composición y del componente 1.

Tipo de ecuación	Composición molar				% Error	
	x_1 (Etanol)	y_1				
		Respuesta base	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.068	0.279	0.283	0.283	1.43	1.43
Ecuación 1 de Poling et. al (2001)	0.068	0.279	0.283	0.283	1.43	1.43
Riedel de Liley et al. (1999)	0.068	0.279	0.283	0.283	1.43	1.43
Antoine de Smith et al (2007).	0.068	0.279	0.282	0.282	1.08	1.08

[ix] Se hizo uso de coeficientes del virial dados en Smith et. al (2007, p. 550)

Tabla 4.31. Punto de burbuja de un sistema cuaternario, coeficientes del virial dados. Composición y del componente 2.

Tipo de ecuación	Composición molar				% Error	
	x_2 (Metilciclo-pentano)	y_2				
		Respuesta base	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.656	0.5	0.498	0.498	0.40	0.40
Ecuación 1 de Poling et. al (2001)	0.656	0.5	0.498	0.498	0.40	0.40
Riedel de Liley et al. (1999)	0.656	0.5	0.499	0.499	0.20	0.20
Antoine de Smith et al (2007).	0.656	0.5	0.499	0.499	0.20	0.20

Tabla 4.32. Punto de burbuja de un sistema cuaternario, coeficientes del virial dados. Composición y del componente 3.

Tipo de ecuación	Composición molar				% Error	
	x_3 (Benceno)	y_3				
		Respuesta base	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.114	0.082	0.081	0.081	1.22	1.22
Ecuación 1 de Poling et. al (2001)	0.114	0.082	0.082	0.082	0.00	0.00
Riedel de Liley et al. (1999)	0.114	0.082	0.081	0.081	1.22	1.22
Antoine de Smith et al (2007).	0.114	0.082	0.081	0.081	1.22	1.22

Tabla 4.33. Punto de burbuja de un sistema cuaternario, coeficientes del virial ya dados. Temperatura

Tipo de ecuación	Temperatura en K			% Error	
	Respuesta base	Respuesta obtenida			
		Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	334.82	334.84	334.84	0.01	0.01
Ecuación 1 de Poling et. al (2001)	334.82	334.84	334.84	0.01	0.01
Riedel de Liley et al. (1999)	334.82	334.78	334.78	0.01	0.01
Antoine de Smith et al (2007).	334.82	334.83	334.83	0.00	0.00

Problema 6°

Se usaron diversas ecuaciones especificadas en las referencias, para el cálculo de las capacidades caloríficas de reactivos y productos:

- Ecuaciones de ajuste polinomial de grado 3 de Reid et al. (1987, pp. 656–732).
- Ecuaciones de ajuste polinomial de grado 4 de Poling et al. (2001, p. A.35-A.46)
- Ecuaciones de ajuste hiperbólico, de ajuste polinomial y ecuaciones con términos polinomiales y logarítmicos de Liley et al. (1999, p. 2.178-2.182) y de Poling et al. (2008, p. 2.174-2.181)
- Ecuaciones de ajuste polinomial de Smith et al. (2007, p. 684)

La Tabla 4.34 y Tabla 4.35 presentan las composiciones calculadas para el benceno e hidrogeno.

Tabla 4.34. Composiciones de equilibrio en la reacción de C_6H_6 y H_2 para formar C_6H_{12} a 1393 K. Composición C_6H_6.

Tipo de ecuación	Composición molar $y_{C_6H_6}$			% Error	
	Respuesta base[xxi]	Respuestas obtenidas[xii]			
		Scilab	Python	Scilab	Python
Reid et. al (1987)	0.119	0.120	0.120	0.84	0.84
Poling et. al (2001)	0.119	0.122	0.121	2.52	1.68
Liley et al. (1999)	0.119	0.122	0.122	2.52	2.52
Poling et al (2008)	0.119	0.121	0.121	1.68	1.68
Smith et al (2007).	0.119	0.119	0.119	0.00	0.00

[x] El dato proporcionado en el libro es el de avance de reacción de equilibrio, a partir del cual se calcularon las composiciones de equilibrio

[xi] Se utilizó el ejercicio 6 de Abbott y Van Ness (2008, pp. 4–37)

[xii] La constante de equilibrio se calculó haciendo uso de la ecuación 13.28 de Smith et. al (2007, p. 499)

Para las composiciones molares de benceno de la Tabla 4.34, se obtuvo un porcentaje de error que variaba desde 0% con la ecuación proporcionada en Smith (2007) y 2.52% con la ecuación 2 de Poling (2008), tanto en Scilab como en Python. En el caso del hidrógeno de la Tabla 4.35, el porcentaje de error variaba entre 0% utilizando Scilab y Python, con la ecuación de Smith (2007) y 2.24% ocupando Scilab para la ecuación de Poling (2001) y 2.24 % usando Python para la ecuación de Liley (1999).

Tabla 4.35. Composiciones de equilibrio en la reacción de C_6H_6 y H_2 para formar C_6H_{12} a 1393 K. Composición H_2

Tipo de ecuación	Composición molar y_{H_2}			% Error	
	Respuesta base	Respuestas obtenidas			
		Scilab	Python	Scilab	Python
Reid et. al (1987)	0.357	0.360	0.360	0.84	0.84
Poling et. al (2001)	0.357	0.367	0.362	2.80	1.40
Liley et al. (1999)	0.357	0.365	0.365	2.24	2.24
Poling et al (2008)	0.357	0.364	0.364	1.96	1.96
Smith et al (2007).	0.357	0.357	0.357	0.00	0.00

Luego de la revisión de los programas por el docente, para el caso de los problemas de cálculo cuarto y quinto sobre punto de burbuja T utilizando la ecuación de Raoult modificada y la formulación gamma-phi para mezclas binarias, ternarias y cuaternarias, se modificó el cálculo de los coeficientes de actividad a través del método UNIFAC puesto que tomaba en cuenta un número reducido de compuestos. El programa encargado del ingreso de las constantes para el cálculo de las presiones de vapor, lanzaba un conjunto de mensajes bastante repetitivo que era posible abreviar.

Los programas principales para estos problemas de cálculos y los demás considerados en TQI-215 además requirieron ser modificados en cuanto al ingreso de datos por parte del usuario con respecto a: falta de capacidad del programa principal para identificar las sustancias involucradas en los cálculos, siendo posible que el usuario cometiera errores; el ingreso de números para la identificación de opciones era problemático porque no se detallaba cuál era el tipo de datos a ingresar; también el programa no permitía reingresar los datos en caso de haberse equivocado el usuario; y las unidades de las presiones y temperaturas no se detallaban.

Para el caso del cálculo de composiciones de equilibrio en reacciones simples para mezclas de gases ideales (problema sexto), no se especificaba con claridad el ingreso de coeficientes estequiométricos o números estequiométricos, que son tipos diferentes de cantidades numéricas. Además, el programa encargado del ingreso de las constantes para el cálculo de las capacidades caloríficas, lanzaba un conjunto de mensajes bastante repetitivo que era posible abreviar.

Operaciones Unitarias I

Se consideró en un comienzo la elaboración de un conjunto de programas para el cálculo de caudales en sistemas de redes de tuberías. Estos programas hubieran necesitado de una interfase tal que permitiera que el usuario construyera la red de tuberías con todos sus accesorios y luego el programa continuaría con el cálculo de los caudales. La razón para no continuar con ello es se requiere de un conocimiento avanzado de programación orientada a objetos para la elaboración de la interface.

Problema 7°

La Tabla 4.36 presenta los resultados obtenidos en el cálculo de diámetros de tuberías.
Se nota que en este ejercicio, las respuestas calculadas presentan un porcentaje de error menor del 1%, tanto para Scilab como para Python, con respecto a las respuestas del libro.

Tabla 4.36. Diámetro de una tubería simple, contando pérdidas primarias únicamente.

Diámetro de tubería en m			% Error	
Respuesta base[xiv]	Respuestas obtenidas			
	Scilab	Python	Scilab	Python
0.422 m (16.6 in)	0.421	0.421	0.237	0.237

[xiv] Se utilize el ejercicio 6.11 de Streeter et al. (2000, pp. 296–297)

Problema 8°

Las respuestas obtenidas para el ejemplo 5.2 de McCabe et al, tanto para Scilab y Python, tienen un 1.43 % de error con respecto a la respuesta del libro. La Tabla 4.37 muestra los resultados.

Tabla 4.37. Flujo volumétrico de una tubería simple, contando pérdidas primarias y secundarias

Caudal en m^3/s			% Error	
Respuesta base[xv]	Respuestas obtenidas			
	Scilab	Python	Scilab	Python
0.0210	0.0213	0.0213	1.43	1.43

[xv] Se utilizó el ejercicio 5.2 de McCabe et al. (2007, pp. 132–133)

Problema 9°

La Tabla 4.38 presenta los resultados obtenidos en el cálculo de flujos volumétricos para un sistema de 2 tuberías. El caudal calculado para la tubería de 1500 m, tanto en Scilab como en Python, posee un 3.15% de error, con respecto a la respuesta del libro; mientras que para la tubería de 900 m, el caudal tiene un 1.22% de error, tanto en Scilab como en Python.

Tabla 4.38. Caudales a través de un sistema de 2 tuberías en paralelo, contando pérdidas primarias y secundarias

Caudal en m^3/s			% Error	
Respuesta base[xvi]	Respuestas obtenidas			
	Scilab	Python	Scilab	Python
0.127 m^3/s (127 L/s)	0.123	0.123	3.15	3.15
0.329 m^3/s (329 L/s)	0.333	0.333	1.22	1.22

[xvi] Se utilizó el ejercicio 7.7 de Valiente Barderas (2002, pp. 300–303)

Según el docente encargado, los programas elaborados para los problemas identificados en Operaciones Unitarias I proporcionaron respuestas satisfactorias. De acuerdo con el docente, un programa más provechoso seria uno que calculara el flujo volumétrico de tuberías en serie, porque entre los ejemplos desarrollados en la asignatura, es bastante frecuente encontrar ese tipo de cálculo. Este problema de cálculo estaba presente en las sugerencias dadas en la encuesta pero se descartó dada la frecuencia de aparición en los libros de las referencias bibliográficas del programa de la asignatura.

Operaciones Unitarias III

Problema 10°

Al igual que en el Cuarto Problema y Quinto Problema, se ocuparon las ecuaciones de Wagner, Antoine, Wagner modificada y Riedel para el cálculo de las presiones de vapor. Los resultados obtenidos de los programas elaborados se muestran en las Tabla 4.39 - Tabla 4.42.

Las Tabla 4.39 y Tabla 4.40 muestras las composiciones molares de la fase líquida del componente 1 (benceno) y 2 (tolueno); mientras que las Tabla 4.41 y Tabla 4.42 presentan las composiciones molares de la fase vapor del componente 1 y 2.

Las composiciones del benceno líquido de la Tabla 4.39 tienen un porcentaje de error entre el 1% y el 2%, ocupando Scilab o Python; mientras que las composiciones calculadas el tolueno líquido de la Tabla 4.40 muestran un porcentaje de error menor del 1%.

Las composiciones del benceno gaseoso de la Tabla 4.41 tienen un porcentaje de error menor del 1%, ocupando Scilab o Python; mientras que las composiciones calculadas el tolueno gaseoso de la Tabla 4.42 muestran un porcentaje de error entre 0.59 - 1.01%.

Como resultado de la revisión de los programas elaborados con los docentes, se recomendó el siguiente conjunto de problemas de cálculo para la asignatura:

- Aplicación del método de McCabe-Thiele para la búsqueda del número de etapas a ocupar en la destilación de mezclas binarias.
- Generación de la curva de secado de un compuesto y determinación del tiempo de secado.
- Determinación de la altura de relleno de torres de absorción, utilizando distintas formas geometrías para los empaques.

Sin embargo estos tres problemas de OPU-315 no formaron parte de las sugerencias proporcionadas en la encuesta. Además los problemas primero y segundo mencionados arriba, implican el uso de métodos gráficos, donde no se utilizan métodos de resolución de

ecuaciones no lineales, como el método de Newton y por lo tanto quedan fuera del alcance del trabajo de graduación.

Se hizo notar además que el programa principal debía ser modificado porque las respuestas obtenidas mostraban una diferencia notable con respecto al valor presentado en el libro consultado.

4.2.3. *Cuadernos Jupyter elaborados*

Los cuadernos Jupyter elaborados se encuentran ubicados en "…\Python\Cuadernos_Jupyter". Los cuadernos se clasifican según la asignatura y el problema de cálculo. Por ejemplo el primer problema se encontrará en:

"…\Python\Cuadernos_Jupyter\TQI_I\Problema 1_Volumen molar en ecuaciones cúbicas de estado (VdW, RK, SRK, PR)". Los cuadernos Jupyter elaborados no abordan los problemas de forma general, sino que están aplicados a ejemplos específicos. La razón principal de utilizar el cuaderno Jupyter fue la de explorar su utilidad en cuanto a la impresión de ecuaciones, edición de texto narrativo e inclusión de porciones de código no muy extensas. Para todos los cuadernos Jupyter se han escrito las ecuaciones pertinentes para la realización de los cálculos.

Tabla 4.39. Composiciones de equilibrio de los productos líquido y vapor de una destilación instantánea. Composicion x_1 (Benceno)

Tipo de ecuación	Composición molar				% Error	
	z_1 (Benceno)	x_1				
		Respuesta base[xvii]	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.5	0.397	0.404	0.404	1.76	1.76
Ecuación 1 de Poling et. al (2001)	0.5	0.397	0.402	0.402	1.26	1.26
Ecuación 3 de Poling et. al (2001)	0.5	0.397	0.402	0.403	1.26	1.51
Riedel de Liley et al. (1999)	0.5	0.397	0.403	0.403	1.51	1.51
Antoine de Smith et al (2007).	0.5	0.397	0.401	0.401	1.01	1.01

[xvii] Se ocupó el ejercicio de Treybal(1986, p. 405)

Tabla 4.40. Composiciones de equilibrio de los productos líquido y vapor de una destilación instantánea. Composición x_2 (Tolueno)

Tipo de ecuación	Composición molar				% Error	
	z_2 (Tolueno)	x_2				
		Respuesta base	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.25	0.274	0.273	0.273	0.36	0.36
Ecuación 1 de Poling et. al (2001)	0.25	0.274	0.273	0.273	0.36	0.36
Ecuación 3 de Poling et. al (2001)	0.25	0.274	0.273	0.273	0.36	0.36
Riedel de Liley et al. (1999)	0.25	0.274	0.273	0.273	0.36	0.36
Antoine de Smith et al (2007).	0.25	0.274	0.273	0.273	0.36	0.36

Tabla 4.41. Composiciones de equilibrio de los productos líquido y vapor de una destilación instantánea. Composicion y_1 (Benceno)

Tipo de ecuación	Composición molar				% Error	
	z_1 (Benceno)	y_1				
		Respuesta base	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.5	0.715	0.717	0.717	0.28	0.28
Ecuación 1 de Poling et. al (2001)	0.5	0.715	0.715	0.715	0.00	0.00
Ecuación 3 de Poling et. al (2001)	0.5	0.715	0.716	0.716	0.14	0.14
Riedel de Liley et al. (1999)	0.5	0.715	0.715	0.715	0.00	0.00
Antoine de Smith et al (2007).	0.5	0.715	0.715	0.715	0.00	0.00

Tabla 4.42. Composiciones de equilibrio de los productos líquido y vapor de una destilación instantánea. Composicion y_2 (Tolueno)

Tipo de ecuación	Composición molar				% Error	
	z_2 (Tolueno)	y_2				
		Respuesta base	Respuestas obtenidas			
			Scilab	Python	Scilab	Python
Ecuación 1 de Reid et. al (1987)	0.25	0.198	0.199	0.199	0.51	0.51
Ecuación 1 de Poling et. al (2001)	0.25	0.198	0.200	0.200	1.01	1.01
Ecuación 3 de Poling et. al (2001)	0.25	0.198	0.200	0.200	1.01	1.01
Riedel de Liley et al. (1999)	0.25	0.198	0.200	0.200	1.01	1.01
Antoine de Smith et al (2007).	0.25	0.198	0.200	0.200	1.01	1.01

La Figura 4.4 presenta un ejemplo del cuaderno electrónico Jupyter desarrollado para el tercer problema.

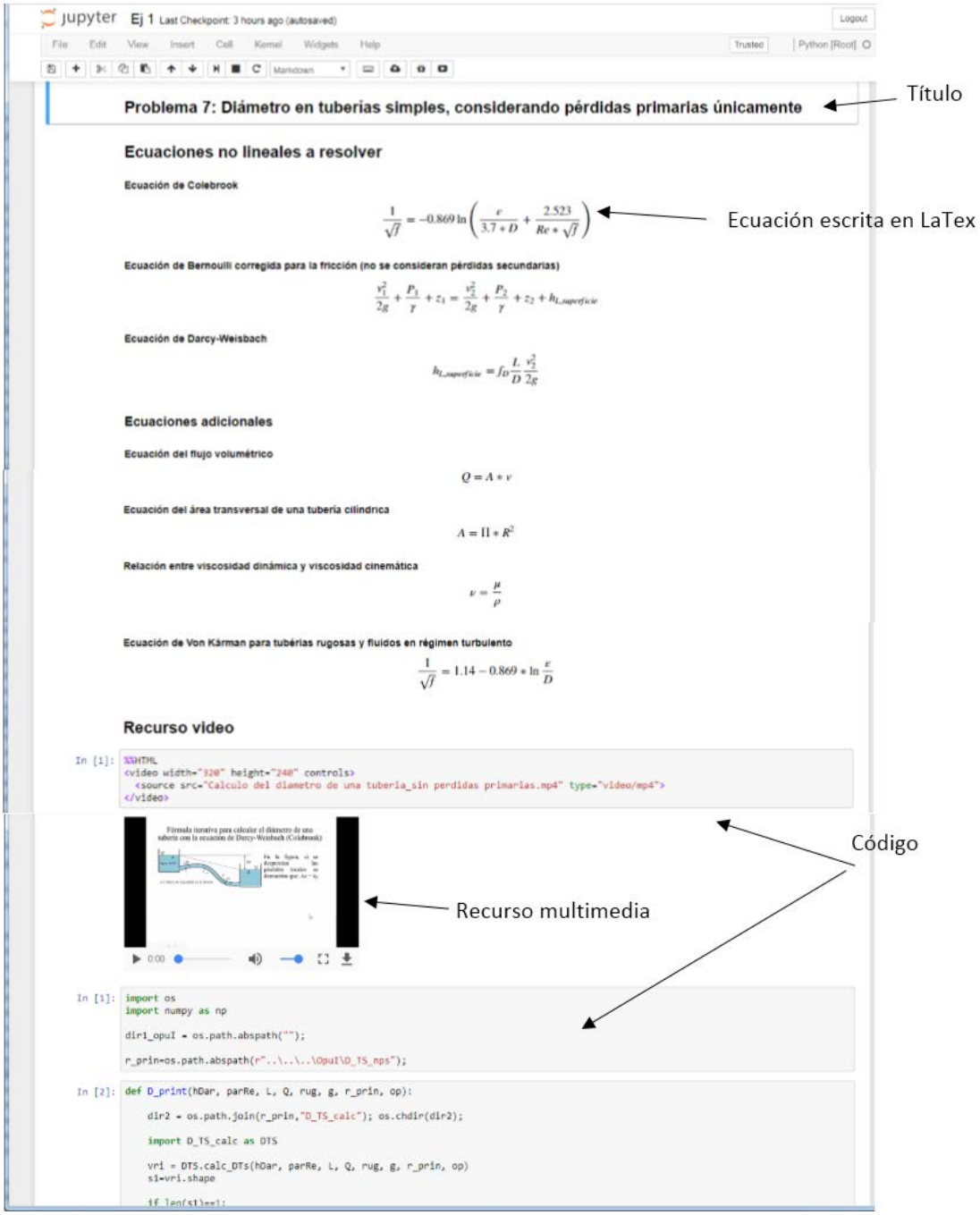


Figura 4.4. Ejemplo de cuaderno electrónico Jupyter elaborado

Termodinámica Química I

Para el cuaderno Jupyter del primer problema, se han escrito las ecuaciones no lineales para el cálculo del factor de compresibilidad y las ecuaciones adicionales para la obtención de los parámetros adimensionales de la ecuación cúbica genérica y de la presión de saturación dada la temperatura y un volumen molar supuesto, en caso se defina la sustancia en estado de saturación. Se construyó la función bat2 que invoca las funciones fv2, desc_region y pres_sat_cubic. Luego se evalúa bat2, incluyendo todos los parámetros necesarios. El cuaderno pide al usuario un único dato: el factor acéntrico. Se ha añadido un video sobre las raíces de la ecuación de Peng-Robinson, "Peng_Robinson_equation.mp4".

Para el cuaderno Jupyter del segundo problema, se han escrito la ecuación no lineal para el cálculo del factor de compresibilidad y las correlaciones tipo Pitzer para el cálculo de los coeficientes del virial en términos del volumen (ecuación de Kammerlingh-Onnes). Una función bat2 invoca la función desc_region y el cuaderno pide al usuario el factor acéntrico. Además deben definirse en el cuaderno los parámetros que desc_region necesita. Se ha incluido un video sobre la ecuación del virial, "Virial equation of state.mp4".

Termodinámica Química II

En el cuaderno Jupyter del tercer problema, se han escrito las ecuaciones no lineales para el cálculo del factor de compresibilidad y las ecuaciones adicionales para la obtención de las propiedades residuales de entalpía, entropía y energía de Gibbso de saturación, y de la presión de saturación dada la temperatura y un volumen molar supuesto, en caso se defina la sustancia en estado de saturación

Se construye la función bat que invoca las funciones fv2 y Hr_Sr_imp. Luego se evalúa bat, incluyendo todos los parámetros necesarios. El cuaderno pide al usuario un único dato: el factor acéntrico.

Para los cuadernos electrónicos del cuarto y quinto problema, se han escrito la ecuaciones no lineales de la formulación Gamma-Phi, del cálculo de la presión de saturación de Reid et al. (1987), Poling et al. (2001), Liley et al. (1999), Poling et al. (2008) y Smith et al. (2007); así

como las ecuaciones utilizadas para el cálculo de los coeficientes de actividad, coeficientes del virial y Phi (únicamente quinto problema).

La función bat_termoII_ELV_Bt invoca la función yBurbT. Se definen las funciones auxiliares det_eco y det_pvf. Deben especificarse en los cuadernos electrónicos los parámetros que yBurbt necesita. Se incluye un video para el cuaderno jupyter del cuarto problema, "Vapor-Liquid Equilibrium Using the Wilson Equation.mp4"

En el cuaderno del sexto problema, se ha incluido la ecuación no lineal de equilibrio de la reacción y las ecuaciones de constante de equilibrio y de las funciones de la capacidad calorífica dadas en Reid et al. (1987), Poling et al. (2001), Liley et al. (1999), Poling et al. (2008) y Smith et al. (2007).

La función bat_termoII_ceq_Rxn invoca las funciones cons_Eq, R_limit, rootRegulaFA_mod. Se define la función auxiliar det_lf_HS y la ecuación a utilizar para el cálculo de las capacidades caloríficas: cpH_Prausnitz_4ed, cpH_Prausnitz_5ed, cpH_Perrry7ed_1eq y cpH_Perrry8ed_2eq y cpH_SVN. Se añade un recurso de video sobre el cálculo de las cantidades molares y composiciones molares de equilibrio, "Equilibrium Composition.mp4".

Operaciones Unitarias I

En el cuaderno Jupyter del séptimo, octavo y noveno problemas, se escribieron las ecuaciones no lineales de Colebrook, Bernoulli corregida para la fricción y Darcy Weisbach para pérdida primarias. En el caso del octavo y noveno problemas también se incluyó la ecuación no lineal del factor K para la determinación de las pérdidas secundarias. Si bien los coeficientes de fricción, factores K de accesorios, longitudes y rugosidades absolutas de un sistema de tuberías (noveno problema) en paralelo pueden variar, la carga del sistema será la misma.

En el cuaderno para el cálculo del diámetro de una tubería, considerando pérdidas primarias únicamente, se construye la función D_print que invoca la función calc_DTS y calcula el diámetro de tubería. Se incluye el video "Calculo del diametro de una tuberia_sin perdidas primarias.mp4".

El cuaderno Jupyter del octavo problema contiene la función fl_vol_print, la cual invoca a la función calc_fl_vol para obtener el flujo volumétrico. Este cuaderno electrónico incluye el video “Velocidad y_o caudal en tuberías_perdidas pr y sec.mp4".

Para el cuaderno Jupyter del noveno problema, se tiene la función fl_vol_par_print que invoca a la función fl_vol_tub_par y calcula los caudales de un sistema de tuberías en paralelo. El cuaderno electrónico incluye los videos "Tuberia en paralelo parte 1.mp4" y "Tuberia en paralelo parte 2.mp4".

Operaciones Unitarias III

Para el cuaderno del décimo problema, la función bat_opuIII_ELV invoca las funciones yBurbt y xRot. Se definen las funciones auxiliares det_eco y det_pvf. Deben especificarse en los cuadernos electrónicos los parámetros que yBurbt y xRot necesitan.

4.2.4. *Manual de usuario elaborado*

Se elaboró un manual del usuario con el fin de proporcionar una guía para la instalación de Scilab y Python y un instructivo para la creación de un cuaderno en una carpeta determinada (sistema operativo Windows). Finalmente se brinda una descripción de los datos requeridos y datos impresos por los programas principales y las funciones elaboradas; también se proporciona, a través de un conjunto de tablas, una esquematización de la distribución de los programas en las carpetas principales y sub-carpetas y de la forma en que se relacionan los distintos archivos generados: programas principales, funciones (veáse Anexo B).

5. Discusión de resultados

La selección de los problemas de cálculo se hizo en base al análisis de los resultados de la encuesta, realizada a los docentes de la Escuela de Ingeniería Química e Ingeniería de Alimentos. El número final de problemas de cálculo iterativo a resolver fue de diez, descartándose dos de los doce problemas originales. La distribución de los problemas fue la siguiente: dos de Termodinámica Química I (TQI-115), cuatro de Termodinámica Química II (TQI-215), tres de Operaciones Unitarias I (OPU-115) y uno de Operaciones Unitarias III (OPU-315). De los métodos numéricos de resolución de ecuaciones no lineales conocidos por los docentes, el método de Newton-Raphson es el preferido.

Los programas desarrollados en Scilab se escribieron en base a la elaboración de estructuras de repetición, de selección y secuenciales y de módulos (programación estructurada y modular); mientras que en Python se utilizaron también clases y objetos (programación orientada a objetos) en los problemas cuarto y quinto de TQI-II y los problemas de OPU-115, porque de esta forma disminuía la cantidad de líneas de código. El proceso de elaboración de los programas de este trabajo ha requerido un periodo alrededor de 6 meses para el diseño, codificación y depuración de los distintos archivos generados. El número de líneas de cada programa ronda entre las 300-450 líneas de código, tomando en cuenta la interfase necesaria para el ingreso de datos por parte del usuario.

Con un porcentaje de error menor al 1%, los programas elaborados para los problemas de TQI-115, y problemas tres de TQI-215 y siete de OPU-115 proporcionan valores más apegados a los presentados en las referencias. Estos problemas junto con los problemas ocho y nueve de OPU-115 comparten la característica de ser relativamente simples en cuanto al número de propiedades físicas requeridas para el funcionamiento de los programas, comparados con los problemas cuatro, cinco y seis de TQI-215 y diez de OPU-315.

Los porcentajes de error obtenidos para los problemas ocho y nueve de OPU-115 son mayores de 1% y esto puede deberse a que los autores de las referencias bibliográficas consultadas combinan un método gráfico y un método numérico para el cálculo de las variables.

Así en el problema ocho de la velocidad y flujo volumétrico a través de una tubería, considerando pérdidas primarias y secundarias, McCabe et al. (2007), combinan el método gráfico del diagrama de Moody para la determinación del factor de fricción y un método de prueba y error para el cálculo de la velocidad y flujo volumétrico. De igual forma, en el problema nueve sobre cálculo de caudales en sistemas de tuberías en paralelo, Valiente Barderas (2002) utiliza el método gráfico de Von Karman para el cálculo de la velocidad y un método de prueba y error para el cálculo de las demás velocidades y caudales.

Los porcentajes de error de los problemas cuatro, cinco y seis de TQI-215 y diez de OPU-315 varían dependiendo en gran medida de la ecuación utilizada para el cálculo de presiones de vapor o de la capacidad calorífica.

En el cuarto problema, los porcentajes de error se mantienen debajo del 2%; en el quinto problema dicho porcentajes son menores al 2% en el caso de ingresar los coeficientes del virial dados en la referencia bibliográfica, pero incrementan hasta valores menores al 3% cuando estos coeficientes se calculan a partir de las temperaturas, presiones, volúmenes y factores de compresibilidad críticos y factores acéntricos. Finalmente el sexto problema, los porcentajes de error son menores al 1% utilizando la ecuación dada por Reid et al. (1987) o por Smith et al. (2007), mientras que estos porcentajes aumentan hasta valores menores al 3% al ocupar la ecuación de Liley et al. (1999) o de Poling et al. (2008). Las constantes ocupadas para el cálculo de las presiones de vapor y capacidades caloríficas son el resultado del ajuste de datos experimentales a un tipo determinado de función, por lo tanto los datos generados por estas funciones podrán ser más apegados a los datos experimentales base para distintas zonas del rango de validez de dichas funciones.

De entre los métodos numéricos utilizados para la resolución de ecuaciones no lineales, el método de la secante es el que se ocupó con mayor frecuencia (7 problemas de cálculo) en un total de tres de las cuatro asignaturas abordadas de Ingeniería Química. La razón de ello es que, tal y como se menciona en la bibliografía, este método tiene mejor convergencia que la mayoría de los demás métodos (excepto el método de Newton) y no necesita del conocimiento de la derivada de la función a resolver ni tampoco de un intervalo que encierre el valor de la solución. Para casos en donde se requiera restringir la respuesta a un intervalo de valores (problema seis con avances de reacción positivos y menores al avance de reacción

máximo del reactivo limitante), el método de regula falsi modificado para una mejor convergencia que el método de regula falsi normal, es adecuado.

Se elaboraron un conjunto de cuadernos electrónicos "Jupyter" para cada asignatura abordada, donde se incluyó código Python, ecuaciones en LaTeX, videos de interés, texto narrativo y títulos para la separación de cada parte del cuaderno. De esta forma, este cuaderno electrónico se convierte en un medio para reunir diversos tipo de contenido y transmitirlos de forma novedosa en las asignaturas de la carrera de Ingeniería Química en general.

Los programas escritos en Scilab y Python para este trabajo de graduación no cuentan con una interfase amigable, por lo que todo el ingreso de las propiedades debe ser de manera secuencial. Esto puede convertirse en una tarea tediosa sobre todo si el usuario comete algún error en la introducción de las propiedades. Esto es cierto para cualquier programa de aplicación futura en las asignaturas de la carrera de Ingeniería Química. Es por esta razón que los programas de esta naturaleza son más apropiados para el uso personal.

A excepción de los programas del cuarto, quinto y décimo problema, que cuentan con una pequeña base de datos para almacenar las contribuciones por grupos del método UNIFAC, estos programas no cuentan con archivos que guarden las constantes para el cálculo de las propiedades físicas de los compuestos: presiones de vapor, capacidades caloríficas, densidades y viscosidades. La introducción de datos por parte del usuario podría volverse menos engorrosa tomando en cuenta este factor.

La importancia de este trabajo de graduación radica en la verificación de las características dadas en la bibliografía, de distintos métodos numéricos aplicados a problemas de cálculo iterativo en asignaturas de Ingeniería Química; y en la inclusión de los métodos numéricos más convenientes en los programas elaborados. En adelante, tanto el estudiante de ingeniería química como los docentes contarán con una guía para abordar esta clase de problemas utilizando a Scilab y/o Python.

La importancia de este trabajo de graduación es que aborda el uso de distintos métodos numéricos para la resolución de problemas de cálculo iterativo en Ingeniería Química. Tras aplicar estos métodos numéricos en la resolución de la mayoría de los problemas expuestos en la Tabla 4.14, se pudo corroborar las ventajas y desventajas en el uso de dichos métodos

numéricos dadas en la bibliografía. La ponderación de las cualidades de distintos métodos numéricos podrá conducir a una mejor utilización del tiempo disponible del estudiante/usuario en la resolución de este tipo de problemas iterativos en Ingeniería Química.

Conclusiones

- La metodología que funcionó durante la selección de los ejercicios de cálculo iterativo consistió en la realización de una encuesta a los docentes con el objetivo de obtener sugerencias sobre este tipo de ejercicios. En el caso de Termodinámica Química II (TQI-215), los problemas sugeridos estaban bien delimitados en el programa de estudio pero dada la diversidad de cálculos involucrados en cada problema se decidió la utilización de sub-problemas. En general, se eligió aquellos problemas con una destacada frecuencia de repetición en las referencias bibliográficas y que implicaran la resolución de ecuaciones no lineales mediante métodos numéricos como los métodos de Regula Falsi, de iteración de punto fijo, de Newton, etc.
- Los programas desarrollados en Scilab y Python para los problemas de cálculo identificados para las asignaturas de Termodinámica Química I (TQI-115), TQI-215, Operaciones Unitarias I (OPU-115) y Operaciones Unitarias III (OPU-315) no difieren mucho en cuanto a extensión (aproximadamente un promedio de 300-450 líneas de código). Para estos programas se ocuparon los paradigmas de programación estructurada y programación modular. Sin embargo en Python también se utilizó el paradigma de programación orientada a objetos. Esta forma de programación permitió la disminución del número de líneas de código en los problemas de punto de burbuja T (TQI-215) y destilación instantánea (OPU-315); y mejoró la claridad del código al poder agrupar y ordenar las funciones creadas en el cálculo del factor de fricción y del cálculo de caudales en sistemas de tuberías en paralelo (OPU-115).

  Se observaron otras semejanzas y diferencias en Scilab y Python: en cuanto a la sintaxis, Scilab y Python no ocupan llaves para abrir y cerrar condicionales, funciones y ciclos while y for; pero además Python tampoco necesita el uso de la palabra “end” o similares al final de las estructuras mencionadas. Como resultado se obtiene una mayor claridad y menor extensión del código en Python. Tanto Scilab como Python permiten la creación de listas y diccionarios (struct en Scilab) y la utilización de funciones como argumentos de otras funciones.

En Scilab se permite el ingreso de vectores, matrices y caracteres, mientras que Python únicamente permite la introducción de caracteres. En ambos lenguajes, la depuración de los programas se facilitó tras permitir el ingreso de un dato a la vez: número o carácter (Scilab) o carácter (Python).

Python cuenta con una mayor disponibilidad de recursos en línea que Scilab en la forma de libros en línea, blogs, documentación en línea de las bibliotecas como Numpy, y las listas de correo de Python. Para Scilab se encuentra información principalmente en blogs, documentación de ayuda incorporada o en línea de Scilab y listas de correo de Scilab.

Python no permite anular ciclos infinitos while, aunque si cuenta con la opción de abrir una terminal Ipython sin necesidad de cerrar Spyder. Scilab si permite anular ciclos while infinitos. Finalmente, cualquier tipo de error en código demasiado extenso para los programas generados en Scilab, genera un mensaje sobre "Stacking problem" (problema de apilamiento) que impide cualquier tipo de acción en Scilab y es necesario cerrar y abrir la consola de dicho software. Python no cuenta con esta dificultad.

- Dada su rápida convergencia, el método de resolución de ecuaciones no lineales más utilizado fue el método de la secante. Dicho método se aplicó a 7 problemas de cálculo: tercer, cuarto, quinto, séptimo, octavo, noveno y décimo problemas.

  Los siguientes métodos más utilizados fueron el método de Newton, el método de Regula-Falsi y el método de iteración de punto fijo. Estos métodos se ocuparon en el primer y segundo problemas de cálculo.

  Finalmente, el método de Regula-Falsi modificado se ocupó en el sexto problema de cálculo. Es importante indicar la conveniencia de un método de resolución de ecuaciones no lineales que combine una rápida convergencia al valor o valores respuesta y/o permita la restricción de dichos valores respuesta. Esta clase de método sería uno de optimización, como los indicados por Chapra y Canale (2007, pp. 363–449).
- El porcentaje de error entre los resultados de los programas desarrollados y de los ejemplos numéricos tomados de las referencias bibliográficas, es menor del 1% para el primer, segundo, tercero, cuarto y séptimo problema de cálculo, en Scilab y Python. Debe también notarse que los programas elaborados para el primer problema presentan error cerca del punto crítico de las sustancias elegidas.

Mientras que en el caso del quinto problema, el porcentaje de error fue menor del 1% para la temperatura calculada pero, dependiendo de la ecuación para el cálculo de la presión de vapor utilizada y del componente, el porcentaje de error para las composiciones de equilibrio podía llegar a ser menor del 1%, entre el 1 y 2% o entre 1 y 3%, tanto en Scilab y en Python en el caso del cálculo de los coeficientes del virial. Para coeficientes del virial dados los porcentajes de error podían menor del 1%, entre el 1 y 1.5% o entre el 2 y 3.5%, tanto en Scilab como en Python.

El cálculo de las composiciones de equilibrio de las fases líquido y vapor de procesos de destilación instantánea (décimo problema) también presentó un comportamiento similar: dependiendo del componente y de la ecuación de la presión de vapor utilizada, el porcentaje de error podía ser menor del 1% o entre 1 y 2%, tanto en Scilab como en Python.

El porcentaje de error en el sexto problema sobre el cálculo de las composiciones de equilibrio en reacciones químicas simples de gases ideales, varió entre el 0.01 y el 4.5%.

Para el octavo problema, el porcentaje de error se mantuvo entre el 2 y el 3%; mientras que para el noveno problema, el porcentaje de error de los resultados obtenidos de los programas con respecto a los ejemplos numéricos, fue entre el 1 y el 3.5%.

De todos estos problemas de cálculo, los problemas cuarto y quinto son más susceptibles a la verificación experimental, puesto que las referencias bibliográficas presentan casos para el ajuste de datos experimentales. Sería una forma adicional de contrastar los resultados obtenidos.

- El cuaderno Jupyter es adecuado para probar porciones medianas de código, resultando ser una herramienta adicional a las propuestas en la metodología (línea de comandos de Python, hojas de cálculo de Excel y la TI CX CAS) para la detección de posibles errores en el código.
- Algunas ideas sobre trabajo futuro relacionado con el área de este trabajo de graduación son: generación de interfaces gráficas de usuario y de bases de datos con las propiedades físicas y químicas de sustancias para los problemas de cálculo tratados en este trabajo de graduación, aplicación de métodos numéricos a distintos problemas de cálculo de Ingeniería Química utilizando software libre como Python o Scilab, por ejemplo la

utilización del método de Runge-Kuta para la resolución de ecuaciones diferenciales en problemas de reactores químicos; evaluación de las respuestas obtenidas al ocupar las rutinas de métodos numéricos disponibles en las bibliotecas Numpy o Scipy, en problemas como el cálculo de las composiciones de equilibrio en reacciones múltiples de mezclas gaseosas; uso de widgets en el cuaderno Jupyter para cambiar el valor numérico de una variable mediante un botón deslizador y apreciar cómo esto modifica un gráfico, aplicable a problemas de ajuste de valores experimentales en termodinámica de soluciones, evaluación de distintos escenarios en problemas de diseño de tuberías simples y sistemas de tuberías (OPU-115), de intercambiadores de calor (OPU-315), de reactores químicos (ingeniería de las reacciones químicas), de torres de destilación, de absorción y de enfriamiento (OPU-315) y de cambio climático (energía y cambio climático).

Recomendaciones

- El software libre como Scilab y Python no presenta ningún costo de uso, volviéndolo una alternativa a las licencias de sitio de software comercial. Se sugiere que la Escuela de Ingeniería Química e Ingeniería de Alimentos, de la Universidad de El Salvador lleve a cabo la comparación de las cualidades y defectos de varios software libre; adopte y promueva la opción más conveniente en las asignaturas de Ingeniería Química. De esta forma el estudiante de ingeniería química podría familiarizarse con estas valiosas herramientas informáticas.
- Se propone la aplicación de métodos gráficos como el método de McCabe Thiele en problemas de destilación de mezclas binarias o el método de Mickley para la obtención de la temperatura de salida del gas proveniente de una torre de enfriamiento utilizando software libre como Scilab o Python;
- Se plantea el uso del cuaderno electrónico Jupyter como un medio para la presentación de distintos elementos: código, títulos escritos en Markdown, texto explicativo, ecuaciones matemáticas escritas en LaTeX, audio y video.
- Se recomienda realizar la actualización de las referencias bibliográficas registradas en los syllabus de las asignaturas, y adquirir los libros de estas referencias para facilitar el acceso por parte de los estudiantes.
- Se aconseja que el estudiante diseñe y revise las veces que sean necesarias la parte metodológica de su futuro trabajo de graduación, puesto que esto ahorra tiempo valioso en el desarrollo del trabajo.
- Se recomienda que el estudiante de ingeniería química profundice en nuevos complicados métodos numéricos, porque de esta manera podrá resolver toda una gama de problemas de cálculo en Ingeniería Química. Un punto de inicio es la lectura de “Métodos numéricos para ingenieros” de Chapra y Canale (2007) o la consulta de videotutoriales de internet.

Referencias Bibliográficas

Abbott, M. M., & Van Ness, H. C. (1975). *Teoría y problemas de Termodinámica* (1a ed.). México D. F., México: McGraw-Hill de México.

Abbott, M. M., & Van Ness, H. C. (2008). Thermodynamics. En D. W. Green (Ed.), *Perry's Chemical Engineers' Handbook* (8a ed.). United States of America: McGraw-Hill Companies.

Ahuja, P. (2010). *Introduction to numerical methods in chemical engineering* (1a ed.). New Delhi, India: PHI Private Learning Limited.

Baker, J., & Sugden, S. (2007). Spreadsheets in Education –The First 25 Years. *Spreadsheets in Education (eJSiE), 1*(1).

Chapra, S. C., & Canale, R. P. (2007). *Métodos numéricos para ingenieros* (5a ed.). México D. F., México: McGraw-Hill/Interamericana Editores.

Chapra, S. E., & Canale, R. P. (2007). *Métodos Numéricos para Ingenieros* (5a ed.). México D. F., México: McGraw-Hill/Interamericana Editores.

Continuum Analytics. (2015, septiembre 21). Download Anaconda now! Recuperado el 31 de julio de 2016, a partir de https://www.continuum.io/downloads

Coulson, J. M., Richardson, J. F., Backhurst, R., & Harker. (1999). *Coulson & Richardson's Chemical Engineering. Fluid flow, heat transfer and mass transfer* (6a ed., Vol. 1). Oxford, Great Britain: Butterworth-Heinemann.

Donal Q. Kern. (1950). *Process heat transfer* (1a ed.). United States of America: Mc-Graw-Hill.

Edgar, T.F. (2006). Enhancing the undergraduate computing experience. *Chemical Engineering Education*, *40*, 231–238.

Faires, V. M., & Simmang, C. M. (1983). *Termodinámica* (6a ed.). México D. F., México: Unión Tipográfica Editorial Hispano Americana.

Ferreira, E. C., Lima, R., & Salcedo, R. (2004). Spreadsheets in Chemical Engineering Education - A Tool in Process Design and Process Integration. *International Journal of Engineering Education*, *20*(6), 928–938.

Foust, A. S., Wenzel, L. A., Clump, C. W., Maus, L., & Andersen, L. B. (2006). *Principios de Operaciones Unitarias* (2a ed.). México, México D. F.: Compañía Editorial Continental.

Geankoplis, C. J. (1998). *Procesos de transporte y operaciones unitarias* (3a ed.). México: Compañía Editorial Continental.

Golub, G. H., & Ortega, J. M. (1992). *Scientific Computing and Differential Equations. An Introduction to Numerical Analysis*. United States of America: Academic Press.

Granger, B. E., & Pérez, F. (2016). Multi-language Data Science with IPython, IJulia, IR, and Friends. Recuperado el 31 de julio de 2016, a partir de http://conferences.oreilly.com/strata/stratany2014/public/schedule/detail/36762

Grulke, E. A. (1986). Using spreadsheets for teaching design. *Chemical Engineering Education*, *20*(3), 128–131, 153.

Guttag, J. V. (2013). *Introduction to Computation and Programming using Python* (1a ed.). Recuperado a partir de https://ia800505.us.archive.org/33/items/IntroductionToComputationAndProgrammingUsingPythonRevisedGuttagJohnV./Introduction%20to%20Computation%20and

%20Programming%20Using%20Python,%20Revised%20-%20Guttag,%20John%20V..pdf

Hill, C. G. (1977). *An introduction to chemical engineering kinetics and reactor design* (1a ed.). United States of America: John Wiley & Sons.

Himmelblau, D. M., & Riggs, J. B. (2004). *Basic Principles and Calculations in Chemical Engineering* (7a ed.). New Jersey, United States of America: Prentice Hall Professional Technical Reference.

Hinestroza, J. P., & Papadopoulos, K. (2003). Using spreadsheets and visual basic application as teaching aids for a unit operations course. *Chemical Engineering Education*, *37*(4), 316–320.

Holland, C. D. (1981). *Fundamentals of multicomponent distillation* (1a ed.). New York, United States: Mc-Graw-Hill.

Hougen, O. A., & Watson, K. M. (1947). *Chemical Process Principles* (1a ed.). New York, United States: Wiley.

Johanssonn, R. (2016). Introduction to Python programming. Recuperado a partir de https://github.com/jrjohansson/scientific-python-lectures/blob/master/Lecture-1-Introduction-to-Python-Programming.ipynb

Levenspiel, O. (2004). *Ingeniería de las Reacciones Químicas* (3a ed.). México: Limusa.

Levine, I. N. (2004). *Físicoquimica* (5a ed., Vol. 1). Madrid, España: McGraw-Hill/Interamericana de España.

Liley, P. E., Thomson, G. H., Friend, D. G., Daubert, T. E., Duck, E., & Green, D. W. (1999). Physical and Chemical Data. En *Perry's Chemical Engineers' Handbook* (7a ed.). United States of America: McGraw-Hill Companies.

Mascarelli, A. (2014). Research tools: Jump off the page. *Nature*, *507*(7493), 523–525. https://doi.org/http://dx.doi.org/10.1038/nj7493-523a

Mataix, C. (1986). *Mecánica de fluidos y máquinas hidráulicas* (2a ed.). Madrid, España: Ediciones del Castillo.

McCabe, W. L., Smith, J. C., & Harriott, P. (2007). *Operaciones unitarias en ingeniería química* (7a ed.). México D. F., México: McGraw-Hill/Interamericana Editores.

Moore, W. J. (1986). *Fisicoquímica Básica* (1a ed.). México D. F., México: Prentice Hall Hispanoamericana.

Moran, M. J., & Shapiro, H. N. (2004). *Fundamentos de Termodinámica Técnica* (2a ed.). Barcelona, España: Reverté.

Mott, R. L. (2006). *Mecánica de fluidos* (6a ed.). México D. F., México: Pearson Educación.

Moura, L. F. (2005, Spring). Microsoft Excel: A simple way of teaching complex tasks. *CACHE News*, *60*.

Nieves Hurtado, A., & Domínguez Sánchez, F. C. (2014). *Métodos numéricos aplicados a la ingeniería* (1a ed.). México D. F., México: Grupo Editorial Patria.

Öhrström, L., Svenson, G., Larson, S., Christie, M., & Niklasson, C. (2005). The Pedagogical Implications of Using MATLAB in Integrated Chemistry and Mathematics Course. *International Journal of Engineering Education*, *21*(4), 683–691.

Perez, F., & Granger, B. E. (2007). IPython: A System for Interactive Scientific Computing. *Computing in Science & Engineering*, *9*(3), 21–29. https://doi.org/10.1109/MCSE.2007.53

Perkel, J. M. (2015a). Programming: Pick up Python. *Nature*, *518*(7537), 125–126.

Perkel, J. M. (2015b). Programming: Pick up Python. *Nature*, *518*(7537), 125–126. https://doi.org/http://dx.doi.org/10.1038/518125a

Poling, B. E., Prausnitz, J. M., & O'Connel, J. P. (2001). *The Properties of Gases and Liquids* (5a ed.). United States of America: McGraw-Hill Companies.

Poling, B. E., Thomson, G. H., Friend, D. G., Rowley, R. E., & Wilding, W. V. (2008). Physical and Chemical Data. En D. W. Green (Ed.), *Perry's Chemical Engineers' Handbook* (8a ed.). United States of America: McGraw-Hill Companies.

Pritchard, P. J., & Mitchell, J. W. (2015). *Fox and McDonald's Introduction to fluid mechanics* (9a ed.). United States of America: John Wiley & Sons.

Reid, R. C., Prausnitz, J. M., & Poling, B. E. (1987). *The Properties of Gases and Liquids* (4a ed.). New York, United States: Mc-Graw-Hill.

Reklaitis, G. V. (1983). *Introduction to Material and Energy Balances* (1a ed.). New York, United States: John Wiley & Sons.

Rosen, E. M., & Partin, L. R. (2000). A Perspective: The Use of the Spreadsheet for Chemical Engineering Computations. *Industrial & Engineering Chemistry Research*, *39*(6), 1612–1613.

Rossant, C. (2016, julio 29). IPython vs Python. Recuperado el 31 de julio de 2016, a partir de https://plot.ly/python/ipython-vs-python/

Sandler, S. I. (2006). *Chemical, biochemical and engineering thermodynamics* (4a ed.). United States of America: John Wiley & Sons.

Scilab Enterprises. (2015). Scilab (Versión 5.5.2) [Software]. Recuperado a partir de http://www.scilab.org/download/latest

SciPy developers. (2016). NumPy Reference. Recuperado a partir de https://docs.scipy.org/doc/numpy/reference/?v=20161029141446

Shacham, M. (1989). An Improved Memory Method for the Solution of a Nonlinear Equation. *Chemical Engineering Science, 44*(7), 1495–1501. Recuperado a partir de ftp://folklore.org.il/shacham/publ_papers/CES_44_1495_89.pdf.

Sharma, N., & Gobbert, M. K. (2010). *A comparative evaluation of Matlab, Octave, Freemat and Scilab for research and teaching* (No. HPCF-2010-7). University of Maryland.

Smith, J. M., Van Ness, H. C., & Abbot, M. M. (2007). *Introducción a la termodinámica en Ingeniería Química* (7a ed.). México D. F., México: McGraw-Hill/Interamericana Editores.

Soni, A. (2008). *An analysis of scientific computing environments: a consumer's view* (Tesis inédita de maestría en Ciencias de la Computación para el Diseño y la Optimización). Massachussets Institute of Technology, United States of America.

Sonntag, R. E., Borgnakke, C., & Van Wylen, G. J. (2003). *Fundamentals of Thermodynamics* (6a ed.). United States of America: John Wiley & Sons.

Streeter, V. L., Wylie, E. B., & Bedford, K. W. (2000). *Mecánica de fluidos* (9a ed.). Santafé de Bogotá, Colombia: McGraw-Hill/Interamericana Editores.

Texas Instrument. (2016). TI-Nspire. TI-Nspire CAS. Guía de Referencia. Recuperado a partir de https://education.ti.com/es/latinoamerica/guidebook/details/es/0AF942A4AC0E47ABA20853691FA6712E/TI-NspireCXCASReferenceGuide

The Dia Developers. (2014). Dia (Versión 0.97.2) [Software]. Recuperado a partir de http://dia-installer.de/

Treybal, R. E. (1986). *Operaciones de transferencia de masa* (2a ed.). México D. F., México: McGraw-Hill de México.

Universidad de El Salvador. (1998). *Plan de Estudio de 1998 Carrera de Ingeniería Química*. San Salvador, El Salvador.

Universidad de El Salvador. (2011). *Catálogo Académico. Pre-Grado. Universidad de El Salvador* (9a ed.). San Salvador, El Salvador: Imprenta Universitaria.

Valiente Barderas, A. (2002). *Problemas de flujo de fluidos* (2a ed.). México D. F., México: Limusa Noriega Editores.

Van Noorden, R. (2014). My digital toolbox: Ecologist Ethan White on interactive notebooks. Recuperado el 30 de julio de 2016, a partir de http://www.nature.com/news/my-digital-toolbox-ecologist-ethan-white-on-interactive-notebooks-1.16015

Welty, J. R., Wicks, C. E., & Wilson, R. E. (2008). *Fundamentos de Transferencia de Momento, Calor y Masa* (2a ed.). México D. F., México: Limusa.

Zotos, K. (2008). Computer Algebra Systems - New Strategies and Techniques. *Applied Mathematics and Computation*, *198*(1), 123–127.

Anexo A Formato de encuesta y Carta a la Escuela

A.1 Formato de la encuesta

Universidad de El Salvador
Facultad de Ingeniería y Arquitectura
Escuela de Ingeniería Química e Ingeniería De Alimentos
Fecha: __ de Diciembre de 2016.

Trabajo de Graduación: Uso de herramientas de computación científica para la resolución de problemas en Ingeniería Química

Encuesta a los docentes de la asignatura de la asignatura[1] para recabar información con respecto a problemas de cálculo iterativo, a resolver mediante herramientas de computación científica.

El objetivo de esta encuesta es determinar los métodos de resolución de ecuaciones no lineales en problemas de cálculo de la asignatura de la asignatura[21], así como establecer los problemas de cálculo, en base a los cuales se elaborarán los programas.

1. ¿Qué tipo de herramientas de software utiliza en su asignatura? ¿Por qué?

_____

_____

_____

_____

_____

[1] Este formato se utilizó en las cuatro asignaturas de interés: Termodinámica Química I, Termodinámica Química II, Operaciones Unitarias I y Operaciones Unitarias III.

2. De los métodos numéricos enlistados a continuación, ¿Cuáles métodos considera más útiles en la realización de cálculos iterativos?

Método de la bisección _____ Método de regula falsi _____

Método de la sustitución sucesiva _____ Método de Newton _____

Método de Newton-Raphson _____ Método de la Secante _____

Método de Ridder _____ Otros _____

3. Si marcó la casilla de otros, especifique.

_____

_____

_____

_____

_____

4. ¿En qué tipo de problemas de cálculo iterativo necesitaría el apoyo o desarrollo de un software?

_____

_____

_____

_____

_____

5. ¿Estaría dispuesto/a a utilizar programas desarrollados en Scilab o Spyder (un ambiente de desarrollo integrado de Python) para la resolución de problemas de cálculo iterativo de su asignatura?

_____

_____

_____

_____

A.2 Carta a la Escuela de Ingeniería Química e Ingeniería de Alimentos sobre la solicitud de información de personal docente activo

UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA QUÍMICA E INGENIERÍA DE ALIMENTOS

REF.EIQIA. 291.2016

Ciudad Universitaria, 23 de noviembre de 2016

Br. Raúl Alejandro Zura Zamora
Estudiante de la carrera de Ingeniería Quimica
Presente.

Estimado Br. Zura:

Reciba un cordial saludo de la Escuela de Ingeniería Química e Ingeniería de Alimentos.

En atención a su nota de fecha 22 de noviembre del presente año, le remito en nota adjunto información solicitada con respecto a los profesores de las asignaturas en la EIQIA-FIA; Operaciones Unitarias I y III, Termodinámica Quimica I y II. Respecto a información sobre la permanencia de los docentes que imparten las asignaturas le sugiero acercarse a cada uno de ellos a consultarles sobre el horario que tienen disponible para atenderlo; pero antes vea los horarios de permanencia colocados en la pared a la entrada de cada cubículo.

Sin otro particular, me despido cordialmente,

"HACIA LA LIBERTAD POR LA CULTURA"

ESCUELA DE INGENIERIA QUIMICA E INGENIERIA DE ALIMENTOS

Ingra. Tania Torres Rivera
Directora

Final Av. Héroes y Mártires del 30 de Julio. El Salvador. C.A.
Apartado Postal 740 Telefax: 2235-5035
Email: ingenieria.quimica@ues.edu.sv / ingenieria.quimica@fia.ues.edu.sv

Termodinámica Química I	Profesora. Delmy del Carmen Rico Peña Instructor, Ing. Fernando Teodoro Ramírez Zelaya
Termodinámica Química II	Profesora. Ingra. Tania Torres Rivera Instructor, Ing. José Aníbal Erazo Cornejo
Operaciones Unitarias I	profesores: Ing. Fernando Teodoro Ramírez Ing. José Aníbal Erazo Cornejo Ing. Miguel Francisco Arévalo
Operaciones Unitarias III	Profesor. Ing. Juan Rodolfo Ramírez Profesora. Ingra. Sara Elizabeth Orellana Claros Instructor, Ing. José Aníbal Erazo Cornejo.

Anexo B Guía de Usuario

B.1 Descarga e Instalación de Scilab 5.5.2 y Anaconda 3

Paso 1. Accedemos a la página web de Scilab en http://www.scilab.org/ y de Anaconda en , en https://www.continuum.io/.

Paso 2. Seleccionamos el archivo de instalación de acuerdo al Sistema Operativo y a la arquitectura de nuestra máquina (por ejemplo Windows 7 64 bits).

Paso 3. Descargamos los instaladores y seguimos las instrucciones de los asistentes de intalación respectivos.

Paso 4. Si la instalación ha sido exitosa se obtendrán los recuadros de las figuras B.1 y B.2

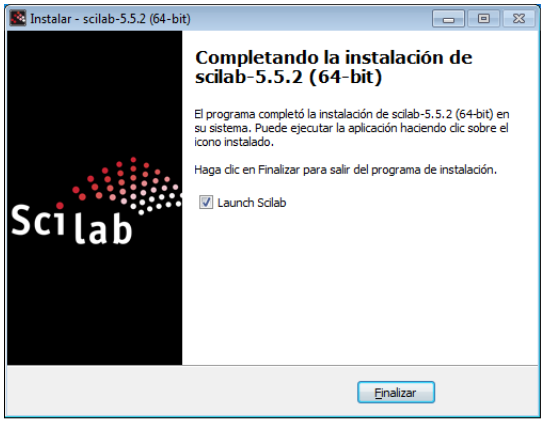


Figura B.1. Finalización de la instalación de Scilab 5.5.2

Figura B.2. Finalización de la instalación de Anaconda 3.

B.2 Creando un nuevo Cuaderno Jupyter

Paso 1. Creamos una carpeta donde almacenaremos nuestros programas de Python. Dentro de dicha carpeta presionamos Shift + clic derecho. En el menú contextual que aparece, seleccionamos con el mouse la opción “Abrir ventana de comandos aquí” .

Paso 2. En la ventana de comandos, digitamos las palabras “jupyter notebook”[2] y presionando enter (Figura B.3), se abre el navegador web predeterminado. A excepción de los enlaces a recursos multimedia en línea, es posible trabajar sin conexión a internet puesto que el navegador es utilizado únicamente para visualizar la carpeta principal y los archivos que este contiene (archivos de texto, cuadernos Jupyter, y otras carpetas) (Figura B.4). Luego en la esquina superior derecha, ubicamos la opción “New” y elegimos Python [Root] para crear un cuaderno Jupyter (Figura B.5). Una vez creado, el archivo recibe automáticamente el nombre de Untitled (Figura B.6).

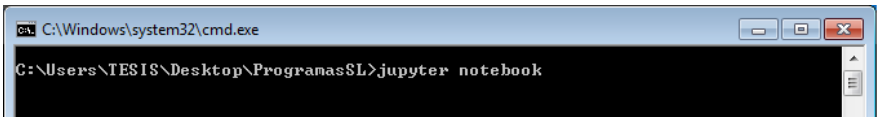


Figura B.3. Ventana de comandos

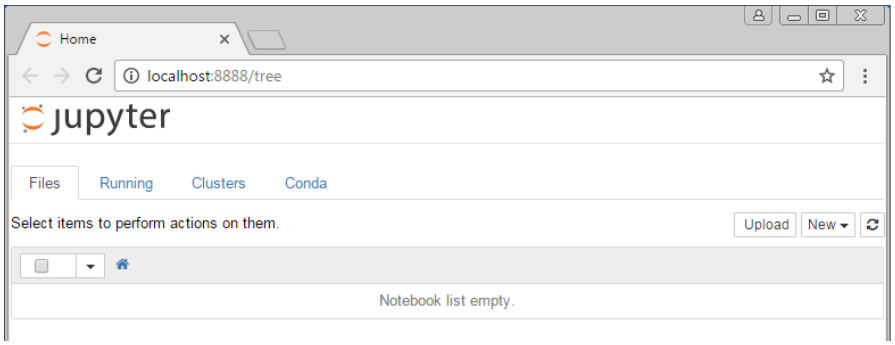


Figura B.4. Directorio principal que mostrará los cuadernos Jupyter de la carpeta creada.

[2] La consola de Windows no diferencia entre mayúsculas y minúsculas

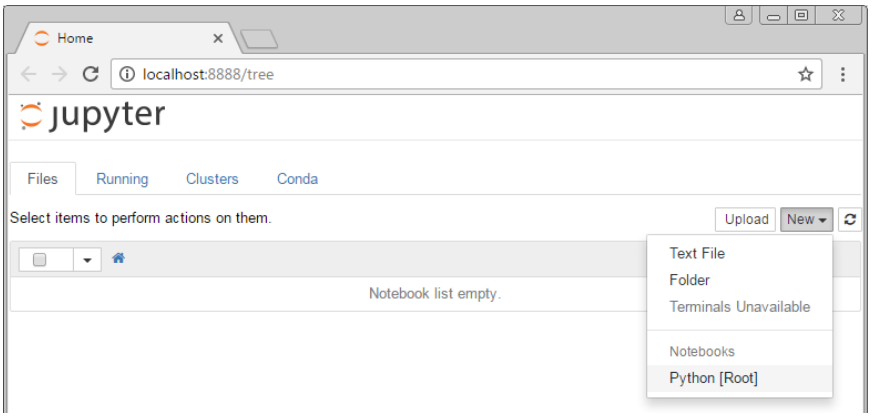


Figura B.5. Creación de un cuaderno Jupyter.

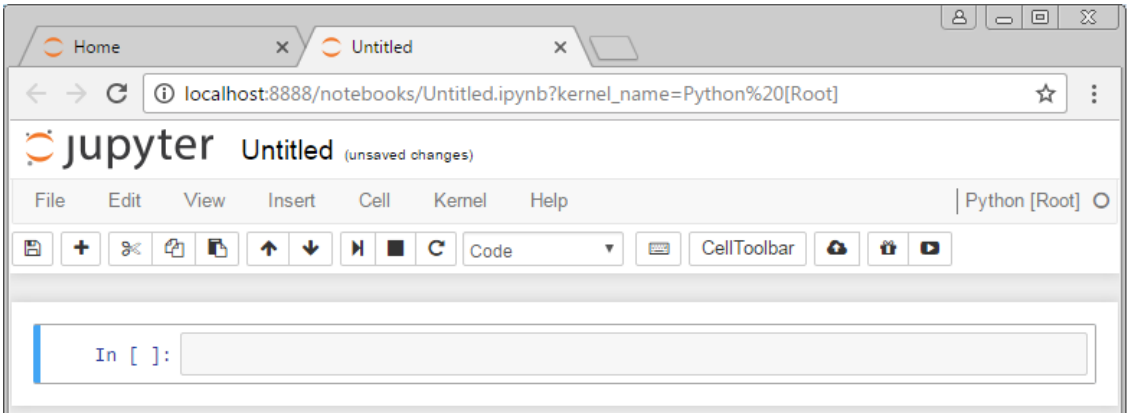


Figura B.6. Nuevo cuaderno Jupyter creado.

B.3 Descripción de programas elaborados en Scilab 5.5.2 y Python 3.5.2

Los programas descritos a continuación son el resultado del trabajo de graduación: "Uso de herramientas de computación científica para la resolución de problemas en Ingeniería Química". Dichos programas piden al usuario un conjunto determinado de datos para la realización de cálculos. Estos datos de ingreso se detallarán para cada programa.

Se brindan además varias tablas que dan a conocer a los programas principales y los sub-programas, así como la distribución de los programas en las carpetas principales y sub-carpetas.

Estos esquemas pretenden además dar una idea general del orden en que el programa principal llama por primera vez a los sub-programas. Es posible que el programa principal realice más de un llamado a los distintos sub-programas o que un sub-programa haga varios llamados de otros sub-programas (como en el caso de BurbT.py con phiCM.py o de Calc_EqC.sce con fncad.sci), pero esto ya no se incluye en los esquemas.

B.3.1 *Termodinámica Química I*

Se diseñaron una serie de programas para los siguientes problemas de Termodinámica Química I:

A. Primer Problema: "Volumen molar en ecuaciones cúbicas de estado, utilizando parámetros de la ecuación de Van der Waals, Redlich-Kwong, Soave-Redlich-Kwong, Peng-Robinson".
B. Segundo Problema: "Volumen molar mediante la ecuación de Kammerlingh-Onnes y las correlaciones para el segundo y el tercer coeficientes del virial.".

Volumen molar en ecuaciones cúbicas de estado, utilizando parámetros de la ecuación de Van der Waals, Redlich-Kwong, Soave-Redlich-Kwong, Peng-Robinson y la ecuación del virial en términos del volumen (Problemas 1° y 2°)

El objetivo de los programas elaborados para estos problemas es el de calcular los volúmenes de líquido y vapor saturado de sustancias puras, utilizando una ecuación de estado cúbica en el volumen o la ecuación de Kammerlingh-Onnes[3] y las correlaciones del segundo y tercer coeficiente del virial. En la Tabla A.1 se esquematiza la forma en que se relacionan los programas escritos en Scilab y Python para el primer y segundo problemas: cal_fz, prueba_sat, y fv2, rootNewton_cubic2 y rootRegulaFA2p2.

Datos que pide cal_fz:

1. Número que identifica a una ecuación cúbica de estado o a la ecuación del virial de Kammerlignh-Onnes para la obtención de volúmenes: 1: Ecuación de Van der Waals, 2: Ecuación de Redlich-Kwong, 3: Ecuación de Soave-Redlich-Kwong, 4: Ecuación de Peng-Robinson y 5: Ecuación de Kammerlingh-Onnes ("correlación de Pitzer del tercer coeficiente del virial").
2. Número que identifica al tipo de método para la resolución de ecuaciones no lineales: 1: método de Regula Falsi, 2: método de sustitución sucesiva y 3: método de Newton-Raphson.
3. Temperatura
4. Presión
5. Número que identifica la opción elegida para definir si las condiciones de la sustancia son de saturación (1) o no (2).
6. Temperatura critica
7. Presión critica

Datos que pide prueba_sat:

1. Factor acéntrico

Sustancia en estado de saturación

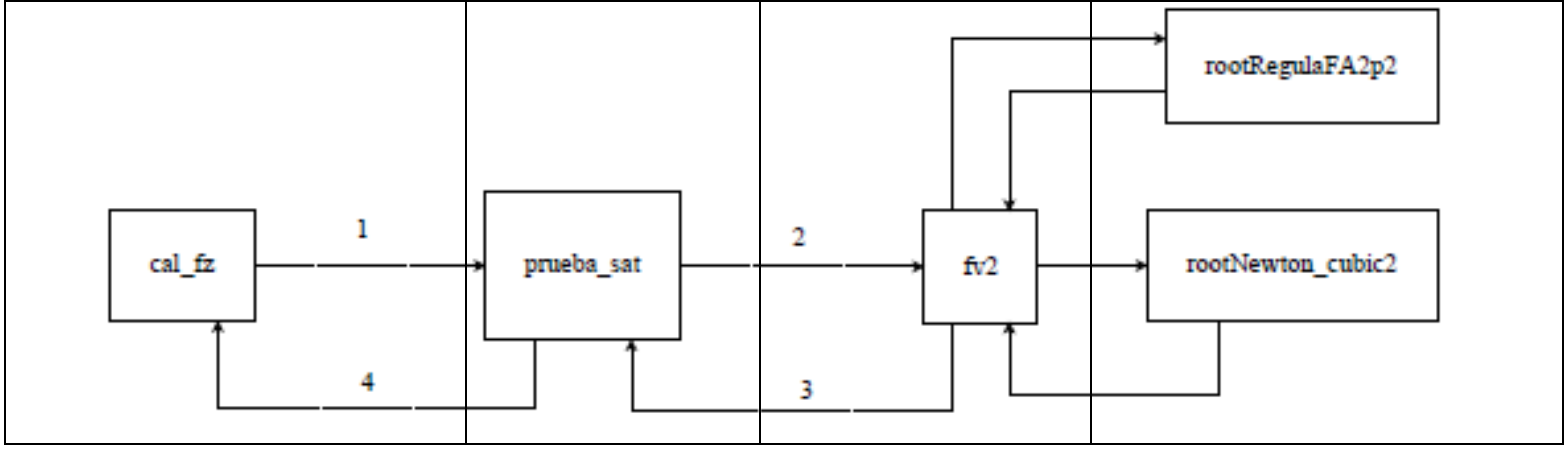
[3] La ecuación de Kammerlingh-Onnes se ocupa únicamente para calcular volúmenes de vapor o gas.

2. Número que identifica la opción elegida para imprimir los factores de compresibilidad de líquido y vapor y los volúmenes de líquido y vapor (1), o solo se imprime las propiedades calculadas de vapor (2), en caso la sustancia se encuentre en estado de saturación.

Datos impresos:

1. Factor de compresibilidad y volumen, para sustancias en estado de saturación o de gas sobrecalentado.

Tabla B.1. Relación entre cal_fz (programa principal) y prueba_sat, fv2, rootRegulaFA2p2 y rootNewton_cubic2 para el problema 1° y 2°.

<table>
<tr><td>Carpeta Principal
prop_volumetricas_fluidos_puros</td><td colspan="2">Sub-Carpeta
Sub-file</td><td>Carpeta Externa
Metodos_numericos</td></tr>
<tr><td>Programa 1</td><td>Programa 2</td><td>Programa 3</td><td>Programas 4</td></tr>
<tr><td colspan="4"> 
</td></tr>
</table>

B.3.2 *Termodinámica Química II*

Se diseñaron varios programas para los siguientes problemas de Termodinámica Química II:

A. Tercer Problema: "Entalpía, entropía y energía de Gibbs residual de un gas puro, mediante la ecuación cúbica genérica"[4]
B. Cuarto Problema: "Punto de burbuja T utilizando la ecuación de Raoult modificada para mezclas binarias, ternarias y cuaternarias"
C. Quinto Problema: "Punto de burbuja T utilizando la formulación gamma-phi para mezclas binarias, ternarias y cuaternarias"
D. Sexto Problema: "Composiciones de equilibrio en reacciones simples para mezclas de gases ideales"

<u>Entalpía, entropía y energía de Gibbs residual de un gas puro, mediante la ecuación cúbica genérica (Problema 3°)</u>

El objetivo de los programas elaborados para este problema es calcular las propiedades residuales de entalpía, entropía y energía de Gibbs de gases reales, haciendo uso de una ecuación de estado cúbica en el volumen. En las Tablas A.2 y A.3 se esquematiza la forma en que se relacionan los programas escritos en Scilab y Python para el tercer problema: Hr_Sr_calc, Hr_Sr_prin, prueba_sat, fv2 (veáse Tabla A.2) y fv2, rootRegulaFa2p2 y rootNewton_cubic2 (véase Tabla A.3).

Datos que pide Hr_Sr_prin:

1. Número que identifica a una ecuación cúbica de estado para la obtención de factores de compresibilidad: 1: Ecuación de Van der Waals, 2: Ecuación de Redlich-Kwong, 3: Ecuación de Soave-Redlich-Kwong y 4: Ecuación de Peng-Robinson.
2. Temperatura
3. Presión

[4] La referencia consultada proporciona los datos de entalpía y entropía residual pero no energía de Gibbs residual)

4. Temperatura critica
5. Presión critica

Datos que pide prueba_sat:

1. Factor acéntrico

Datos impresos:

1. Entalpía, entropía y energía de Gibbs residual

Tabla B.2. Relación entre Hr_Sr_prin (programa principal) y Hr_Sr_calc, prueba_sat y fv2 para el problema 3°

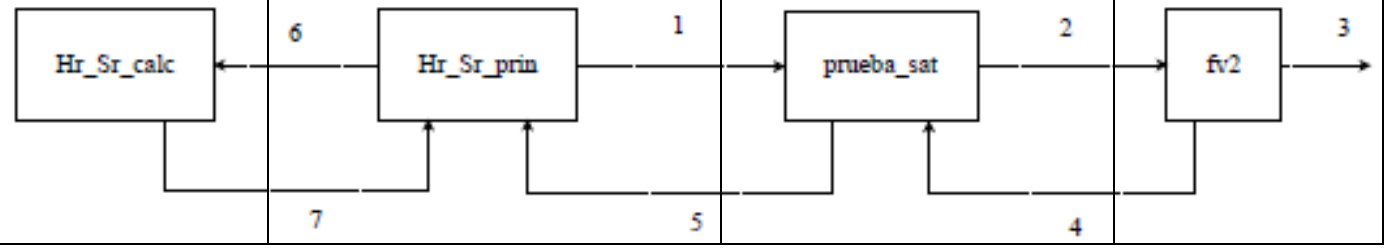
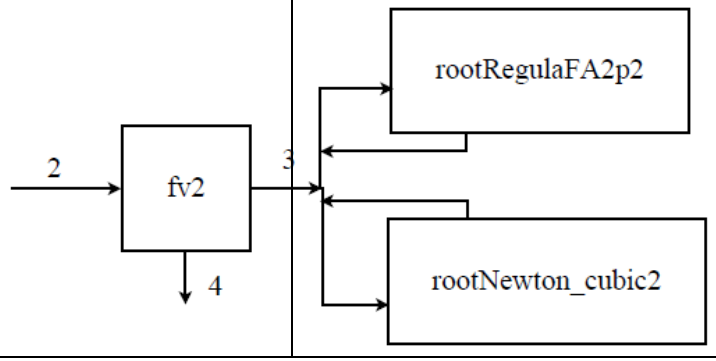
Sub-Carpeta Hr_Sr_calc	Carpeta Principal Prop_resi_cubic	Sub-Carpeta sat_condicion	
Programa 5	Programa 1	Programa 2	Programa 3
			

Tabla B.3. Relación entre fv2 y rootRegulaFA2p2 y rootNewton_cubic2 para el problema 3°

Sub-Carpeta sat_condicion	Carpeta Externa Metodos_numericos
Programa 3	Programas 4
	

Punto de burbuja T utilizando la ecuación de Raoult modificada y la formulación Gamma-Phi para mezclas binarias, ternarias y cuaternarias (Problemas 4° y 5°)

El objetivo de los programas elaborados para estos problemas es el calcular de la temperatura y de las composiciones molares en la fase gaseosa de una mezcla de sustancias (el "punto de burbuja"), utilizando la ecuación de Raoult modificada y la formulación γ - Φ para modelar el comportamiento de los componentes de la mezcla. En las Tablas A.4 – A.9 se esquematiza la forma en que relacionan los programas escritos en Scilab y Python para los problemas cuarto y quinto:

- calc_ELV, opc y cpre_ing (véase Tabla A.4)
- coac, opc y BurbT (véase Tabla A.5)
- Wcomp, NRTLcomp, UNIFAC, coac, BurbT y phiCM (véase Tabla A.6)
- tablas_read, UNIFAC, BurbT, tab_amk, tab_Rk_Qk (véase Tabla A.7)
- rootSecantePlus2Ap2, BurbT, fpres_vap (véase Tabla A.8)
- BurbT, imp_op_tab e imp_tab (véase Tabla A.9)

Datos que pide calc_ELV:

1. Número que identifica el número de componentes de la solución[5] (2, 3 o 4)
2. Número que identifica la ecuación para el cálculo de equilibrio líquido vapor elegida: Ley de Raoult modificada (1) o Formulación Gamma-Phi (2).

Únicamente Formulación Gamma-Phi

   2.1. Número que identifica la opción elegida para definir si se cuenta con coeficientes del virial (1) o no (2), en caso se haya elegido el segundo grado de idealidad.

   2.2. Número que identifica la opción elegida para definir si se puede considerar a la fase vapor como solución ideal (1) o solución real (2), en caso se haya elegido la Formulación Gamma-Phi.

3. Presión
4. Temperatura inicial para el cálculo de la temperatura de burbuja

[5] Los programas para este problema se trabajaron de tal manera que es posible especificar un número de componentes más alto, pero los ejemplos utilizados tratan soluciones con 2, 3 o 4 componentes.

5. Conjunto de composiciones molares de la fase líquida (“x”)
6. Número que identifica la opción elegida para impresión de tablas con las iteraciones realizadas: en la consola de Scilab o Spyder (1), archivos de extensión.CSV (pueden ser abiertos en Excel) (2) o la no impresión de las tablas de iteraciones (se imprimirá únicamente la temperatura de burbuja) (3).

Datos que pide opc:

1. Número que identifica el método de cálculo de coeficientes de actividad: método de Wilson (1), método NRTL (2) o método UNIFAC (3).

Únicamente Formulación Gamma-Phi

Se cuenta con coeficientes del virial

2. Conjunto de coeficientes del virial, en caso se cuenta con dichos coeficientes

No se cuenta con coeficientes del virial

3. Factores acéntricos, temperaturas críticas, volúmenes críticos, factores de compresibilidad críticos y presiones críticas, en caso se haya elegido la formulación Gamma-Phi.

Datos que pide cpre_ing:

1. Número que identifica el libro elegido para la búsqueda de las constantes de las ecuaciones de cálculo de presiones de vapor: “Properties of Gases and Liquids“, cuarta edición; “Properties of Gases and Liquids”, quinta edición; “Chemical Engineers’ Handbook”, séptima u octava edición; o “Introducción a la Termodinámica Química en Ingeniería Química”.
2. Número que identifica la ecuación elegida para el cálculo de presiones de vapor: Si se ha elegido a “Properties of Gases and Liquids”, cuarta edición, elegir entre la ecuación 1 (Wagner), la ecuación 2 (Antoine) o la ecuación 3 (Frost-Kalkwarf-Thodos). Si se ha elegido a “Properties of Gases and Liquids”, quinta edición, elegir entre la ecuación 1 (Antoine), la ecuación 2 (Antoine modifcada), y la ecuación 3 (Wagner modificada). En estos programas, únicamente se define el tipo de ecuación a utilizar, por lo tanto el usuario debe buscar las constantes en los libros referidos.

   Ingresar la temperatura y presión crítica, en caso de elegir la ecuación de Wagner,

Datos que pide coac:

1. Si se eligió el método de Wilson para el cálculo de los coeficientes de actividad, se pide los volúmenes molares de los componentes de la mezcla y los parámetros aij.
2. Si se eligió el método NRTL para el cálculo de los coeficientes de actividad, se pide los parámetros alfa ij y los parámetros bij.

Datos impresos:

1. Temperatura de burbuja
2. Impresión de tablas con los cálculos iterativos de la temperatura de burbuja en las consolas de Scilab y Python o en archivos de extensión .CSV, en caso se haya elegido estas opciones.

Tabla B.4. Relación entre calc_ELV (programa principal) y opc y cpre_ing para los problemas 4° y 5°.

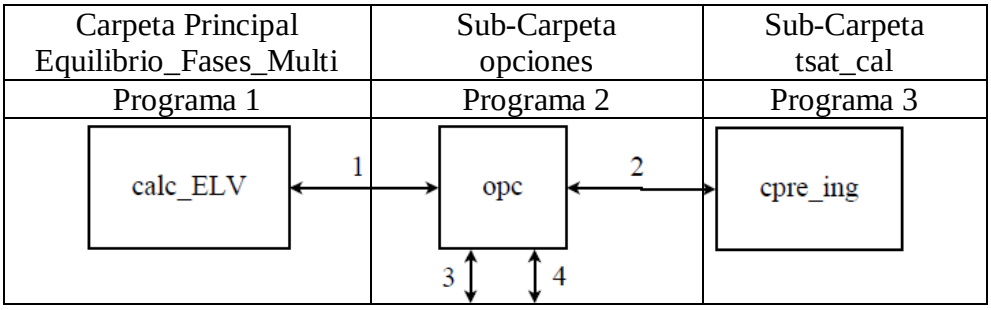


Tabla B.5. Relación entre coac, opc y BurbT para los problemas 4° y 5°.

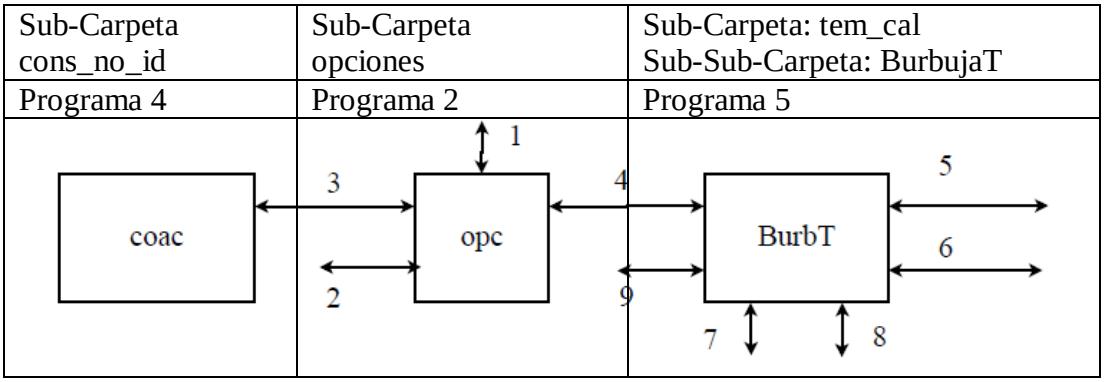
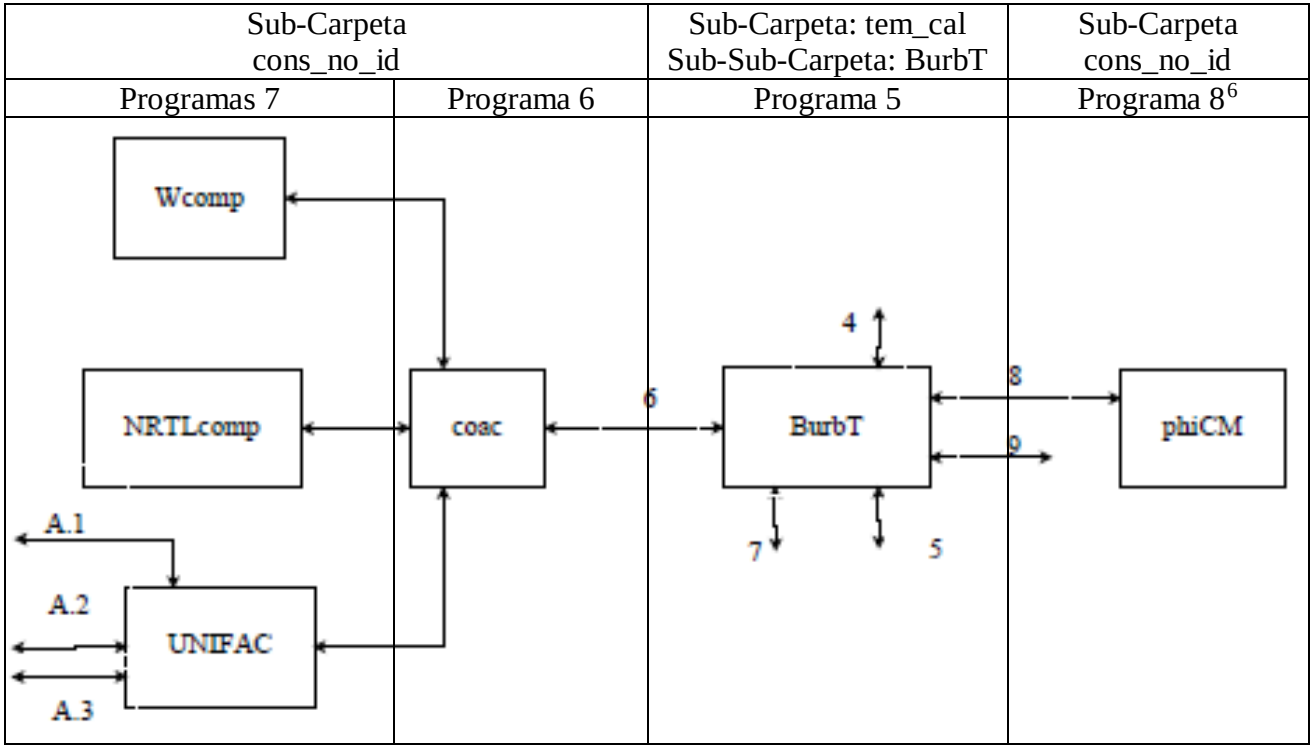


Tabla B.6. Relación entre coac, BurbT, phiCM, Wcomp, NRTLcomp y UNIFAC para los problemas 4° y 5°.



[6] El programa 8 se utilizará si se eligió el segundo grado de idealidad.

Tabla B.7. Relación entre UNIFAC y tablas_read, tab_amk y tab_Rk_Qk para para los problemas 4° y 5°.

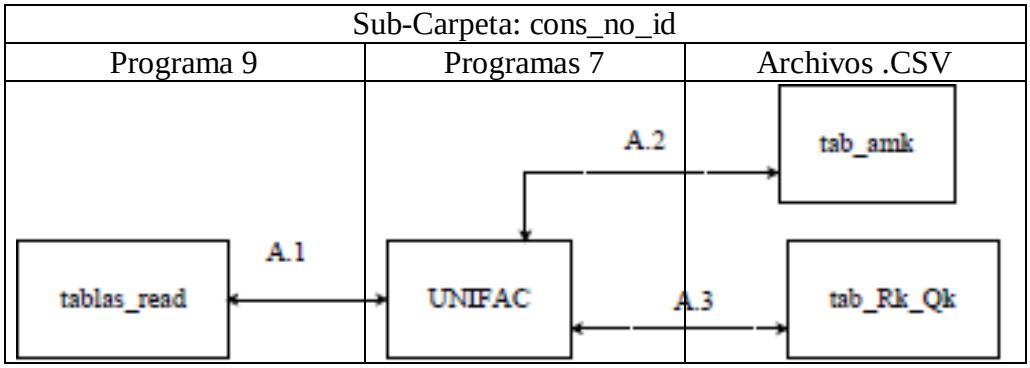


Tabla B.8. Relación entre BurbT y fpres_vap y rootSecantePlus2Ap2 para para los problemas 4° y 5°.

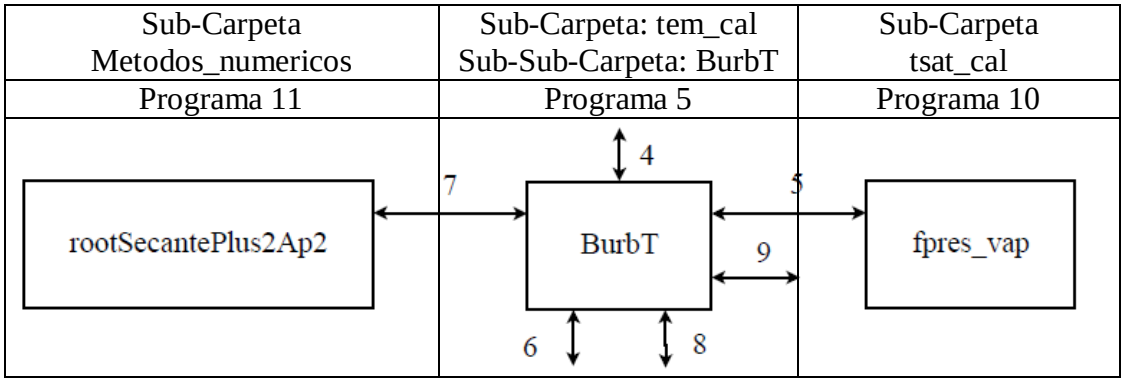
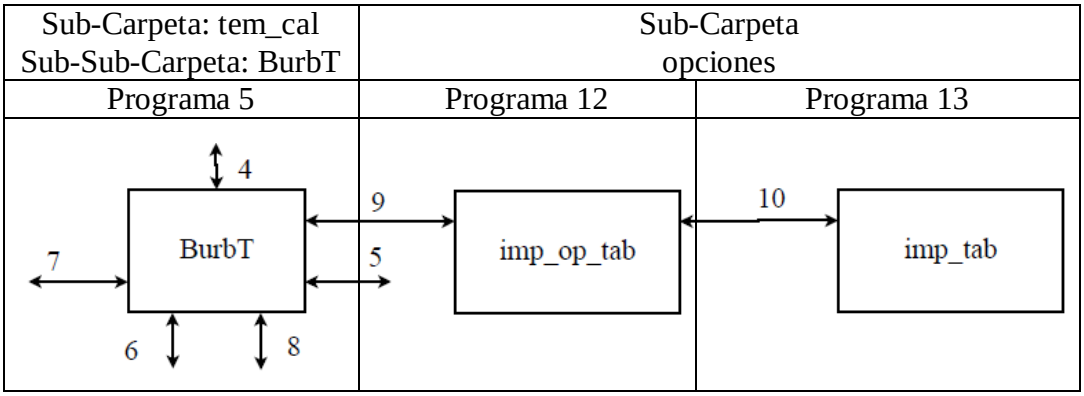


Tabla B.9. Relación entre BurbT, imp_op_tab e imp_tab para los problemas 4° y 5°.



Composiciones de equilibrio en reacciones simples para mezclas de gases ideales (Problema 6°)

El objetivo de los programas elaborados para este problema es calcular las composiciones molares de equilibrio de reactivos y productos considerados como gases ideales (únicamente a temperaturas altas y bajas presiones) en reacciones simples. En las Tablas A.10-A.13 se la forma en que se relacionan los programas escritos en Scilab y Python para el sexto problema:

- func_cad, Calc_EqC, defK (véase Tabla A.10)
- ccp_ing, defK, cK (véase Tabla A.11)
- RomberT, cK, Calc_DH_DS, func_cad (véase Tabla A.12)
- cK, Calc_EqC, rootRegulaFA_mod (véase Tabla A.13)

Datos que pide calc_EqC:

1. Cantidad de reactivos que participan en la reacción
2. Número de moles de los reactivos que participan en la reacción
3. Coeficientes estequiométricos de los reactivos que participan en la reacción
4. Cantidad de productos que participan en la reacción
5. Número de moles de los productos que participan en la reacción
6. Coeficientes estequiométricos de los productos que participan en la reacción

7. Número que identifica la opción elegida para definir si hay presencia de compuestos inertes (1) o no (2)

   Con presencia de compuestos inertes

   7.1. Cantidad de compuestos inertes presentes en el sistema reactivo, en caso se haya establecido la presencia de compuestos inertes

   7.2. Número de moles de los compuestos inertes presentes en el sistema reactivo

8. Temperatura
9. Presión

Datos que pide defK

1. Número que identifica la opción elegida para definir si se cuenta con la constante de equilibrio (1) o no (2)

Se cuenta con la constante de equilibrio

2. Constante de equilibrio, en caso se cuente con dicha constante.

No se cuenta con la constante de equilibrio

3. Número que identifica la opción elegida para establecer si el cambio de entalpia es independiente de la temperatura (1) o no (2) (se calculará la constante de equilibrio en base al valor de dicha constante a la temperatura de referencia).

   Se considera que el cambio de entalpia es independiente de la temperatura

   3.1. Temperatura de referencia, en caso el cambio de entalpia es independiente de la temperatura.

   3.2. Constante de equilibrio a la temperatura de referencia, en caso el cambio de entalpia es independiente de la temperatura.

   3.3. Cambio de entalpía, en caso el cambio de entalpia es independiente de la temperatura.

   No se puede considerar que el cambio de entalpía es independiente de la temperatura

   3.4. Temperatura de referencia (por lo general es 298.15 K)

   3.5. Cambio de entalpía a la temperatura de referencia (por lo general es 298.15 K)

   3.6. Cambio de energía de Gibbs a la temperatura de referencia (por lo general es 298.15 K).

Datos impresos:

1. Número de moles en equilibrio de los reactivos que participan en la reacción
2. Número de moles en equilibrio de los productos que participan en la reacción
3. Composiciones molares de equilibrio de los reactivos que participan en la reacción
4. Composiciones molares de equilibrio de los productos que participan en la reacción

Con presencia de compuestos inertes

5. Número de moles de los compuestos inertes que participan en la reacción
6. Composiciones molares de los compuestos inertes que participan en la reacción

Tabla B.10. Relación entre Cal_EqC (programa principal) y func_cad y defK para el problema 6°.

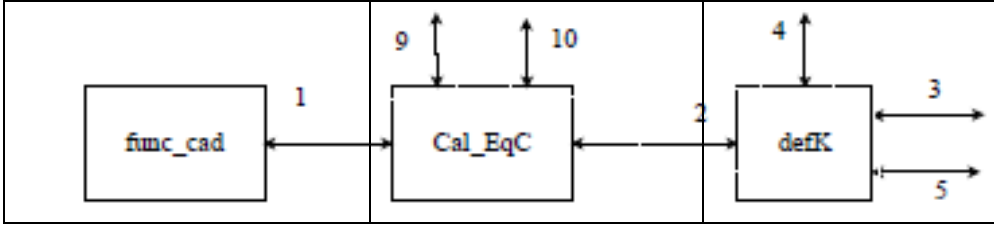
Sub-Carpeta opcionesRxn	Carpeta Principal Equilibrio_rxns	Sub-Carpeta opcionesRxn
Programa 2	Programa 1	Programa 3
		

Tabla B.11. Relación entre defK, ccp_ing y cK para el problema para el problema 6°.

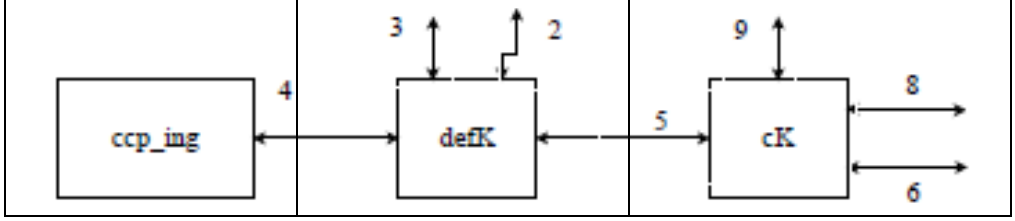
Sub-Carpeta opcionesRxn		Sub-Carpeta cal_HSK
Programa 4	Programa 3	Programa 5
		

Tabla B.12. Relación entre cK, RomberT, Calc_DH_DS y func_cad para el problema 6°.

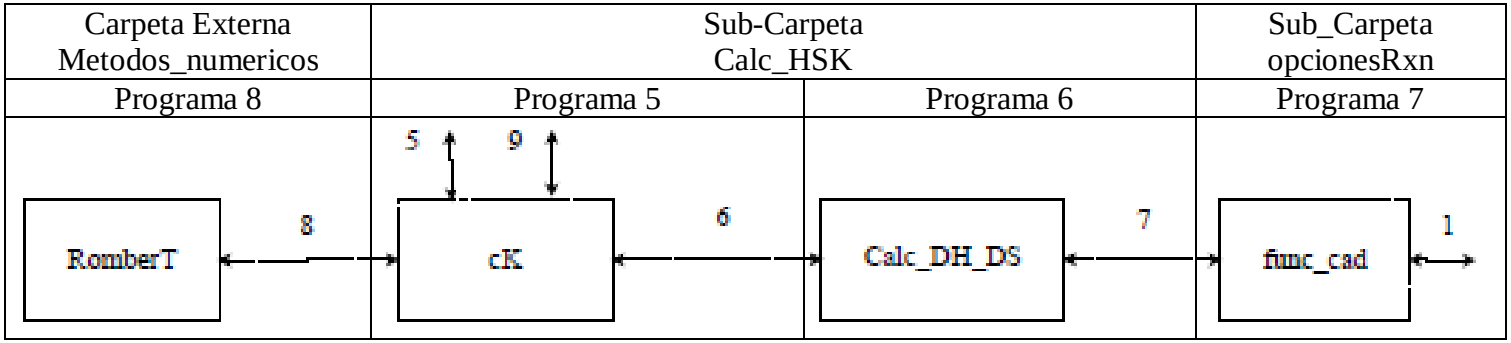
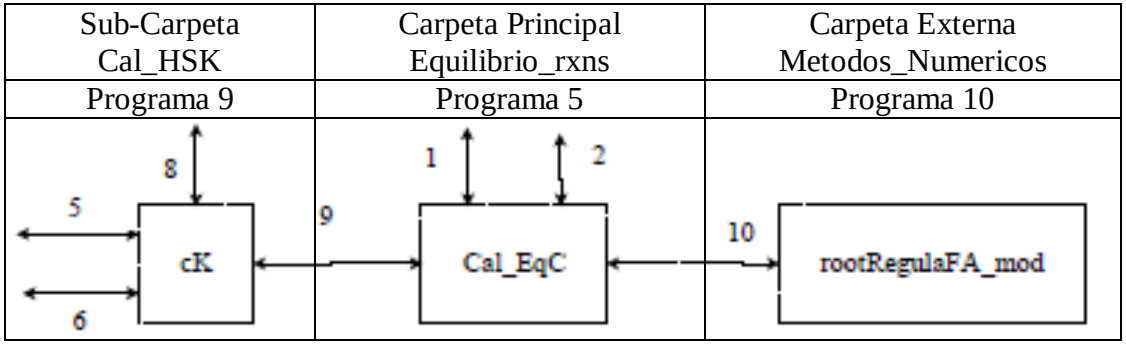


Tabla B.13. Relación entre cK, Cal_EqC y rootSecantePlus2Ap2 para el problema 6°.



.

B.3.3 *Operaciones Unitarias I*

Se diseñaron varios programas para los siguientes problemas de Operaciones Unitarias I:

A. Séptimo Problema: “Diámetro en tuberías simples, considerando pérdidas primarias únicamente”
B. Octavo Problema: “Velocidad y caudal en tuberías simples considerando pérdidas primarias y secundarias”
C. Noveno Problema: “Caudales para las tuberías de un sistema en paralelo

Diámetro en tuberías simples, considerando pérdidas primarias únicamente (Problema 6°)

El objetivo de los programas elaborados para este problema es calcular el diámetro de una tubería simple, considerando únicamente pérdidas debidas a la fricción superficial. En las Tablas A.14 y A.15 se esquematiza como se relacionan los programas escritos en Scilab y Pyton para el séptimo problema: CalcD, D_TS_calc y f_fric (véase Tabla A.14) y D_TS_calc, rootSecantePlus2Ap2 (véase Tabla A.15).

Datos que pide Prin_2_D_c:

1. Rugosidad de la tubería
2. Longitud de la tubería
3. Número que identifica la opción elegida para definir si se cuenta con las pérdidas del sistema (carga o cabeza) en m (1) o no (2)

Se cuenta con las pérdidas del sistema en m

4. Número que identifica la opción elegida para definir si se cuenta con datos de la viscosidad dinámica y densidad (1) o con datos de la viscosidad cinemática (2)

Se cuenta con la viscosidad dinámica y densidad

4.1. Densidad
4.2. Viscosidad dinámica

Se cuenta con la viscosidad cinemática

4.3. Viscosidad cinemática

No se cuenta con las pérdidas del sistema en m

5. Presión inicial y final de la tubería
6. Nivel de altura inicial y final de la tubería
7. Densidad
8. Número que identifica la opción elegida para definir si se cuenta con datos de la viscosidad dinámica (1) o con datos de la viscosidad cinemática (2)

Se cuenta con la viscosidad dinámica y densidad

8.1. Viscosidad dinámica

Se cuenta con la viscosidad cinemática

8.2. Viscosidad cinemática

9. Caudal que circula a través de la tubería

Datos impresos:

1. Diámetro de la tubería

Tabla B.14. Relación entre Calc_D (programa principal) y D_TS_calc y f_fric para el problema 7°

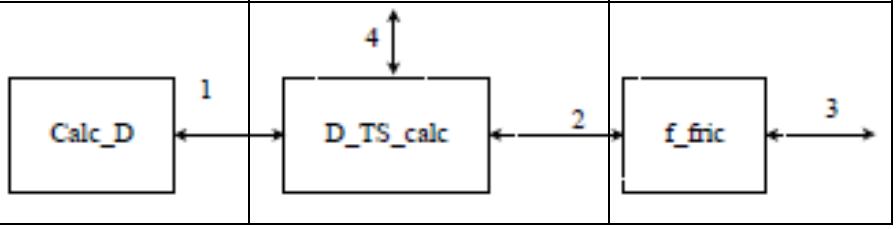
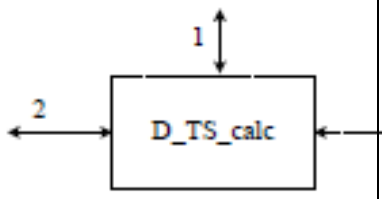
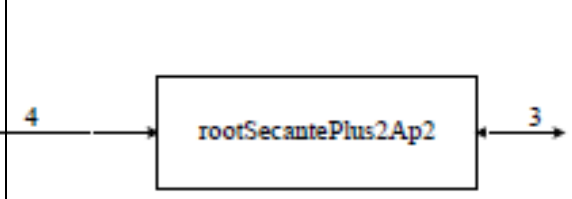
Carpeta Principal D_TS_nps	Sub-Carpeta D_TS_calc	Carpeta Externa f_fric
Programa 1	Programa 2	Programa 3
		

Tabla B.15. Relación entre D_TS_calc y rootSecante_Plus2Ap2 para el problema 7°.

Carpeta Principal D_TS_calc	Carpeta Externa Metodos_numericos
Programa 2	Programa 4
	

Velocidad y caudal en tuberías simples considerando pérdidas primarias y secundarias (Problema 8°)

El objetivo de los programas elaborados para este problema es calcular el flujo volumétrico a través de una tubería simple, considerando pérdidas debidas a la fricción superficial y a la fricción de forma. En las Tablas A.16-A.17 se esquematiza el funcionamiento de los programas escritos en Scilab y Python para el octavo problema.

Datos que pide Prin2_fl_vol:

1. Rugosidad de la tubería
2. Longitud de la tubería
3. Suma de los coeficientes K de los accesorios de la tubería
4. Número que identifica la opción elegida para definir si se cuenta con las pérdidas del sistema (carga o cabeza) en m (1) o no (2)

   Se cuenta con las pérdidas del sistema en m

   4.1. Número que identifica la opción elegida para definir si se cuenta con datos de la viscosidad dinámica y densidad (1) o con datos de la viscosidad cinemática (2)

   Se cuenta con la viscosidad dinámica y densidad

   4.1.1. Densidad

   4.1.2. Viscosidad dinámica

   Se cuenta con la viscosidad cinemática

   4.1.3. Viscosidad cinemática

No se cuenta con las pérdidas del sistema en m

4.2. Presión inicial y final de la tubería

4.3. Nivel de altura inicial y final de la tubería

4.4. Densidad

4.5. Número que identifica la opción elegida para definir si se cuenta con datos de la viscosidad dinámica (1) o con datos de la viscosidad cinemática (2)

Se cuenta con la viscosidad dinámica y densidad

4.5.1. Viscosidad dinámica

Se cuenta con la viscosidad cinemática

4.5.2. Viscosidad cinemática

5. Diámetro de la tubería

Datos impresos:

1. Flujo volumétrico a través de la tubería

Tabla B.16. Relación entre Calc_fl_vol (programa principal) y Fl_vol_TubS_calc para el problema 8°

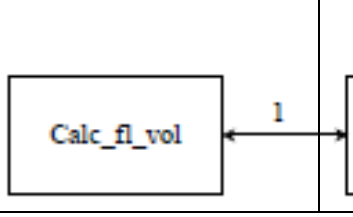
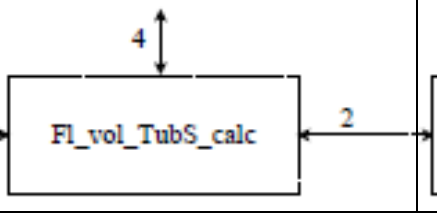
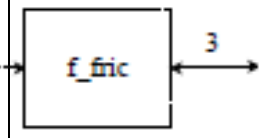
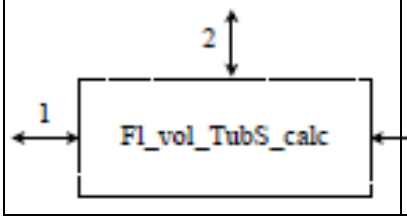
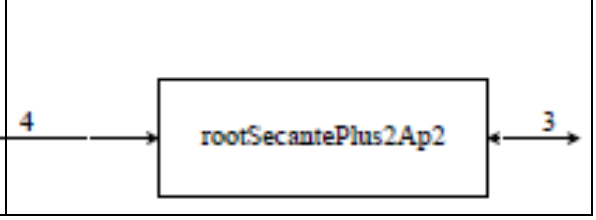
Carpeta Principal Fl_TS_pps	Sub-Carpeta Fl_vol_TubS_calc	Carpeta Externa f_fric
Programa 1	Programa 2	Programa 3
		

Tabla B.17. Relación entre Fl_vol_TubS_calc y rootSecantePlus2Ap2 para para el problema 8°

Carpeta Principal Fl_vol_TubS_calc	Carpeta Externa Metodos_numericos
Programa 2	Programa 4
	

Caudales para las tuberías de un sistema en paralelo (Problema 9°)

El objetivo de los programas elaborados para este problema es calcular los caudales a través de un sistema de tuberías en paralelo, considerando pérdidas debidas a la fricción superficial y a la fricción de forma. En las Tablas A.18-A.20 se esquematiza la forma en que se relacionan los programas desarrollados en Scilab y Python para el noveno problema:

- Calc_tub_par, FL_vol_par_calc y f_fric (véase Tabla A.18)
- FL_vol_par_calc, imp_op_tab e imp_tab (véase Tabla A.19)
- f_fric y root_SecantePlus2Ap2 (véase Tabla A.20)

Datos que pide Prin2_tub_par:

1. Número de tuberías en paralelo
2. Diámetro de cada tubería
3. Longitud de cada tubería
4. Rugosidad de cada tubería
5. Suma de los coeficientes K de los accesorios de cada tubería
6. Número que identifica la opción elegida para definir si se cuenta con datos de la viscosidad dinámica y densidad (1) o con datos de la viscosidad cinemática (2)

   Se cuenta con la viscosidad dinámica y densidad

   6.1. Densidad

   6.2. Viscosidad dinámica

Se cuenta con la viscosidad cinemática

6.3. Viscosidad cinemática

7. Caudal total del sistema de tuberías en paralelo.

Datos impresos:

1. Tres tablas con los datos de los factores de fricción, velocidades y caudales de las tuberías, generados para cada iteración.
2. Caudal de cada tubería

Tabla B.18. Relación entre Calc_tub_par (programa principal) y FL_vol_par_calc y f_fric para el problema 9°.

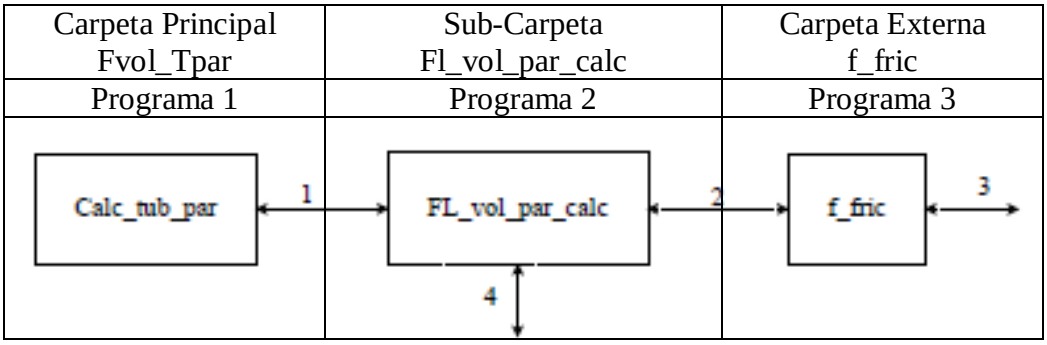


Tabla B.19. Relación entre FL_vol_par_calc, imp_op_tab e imp_tab para el problema 9°

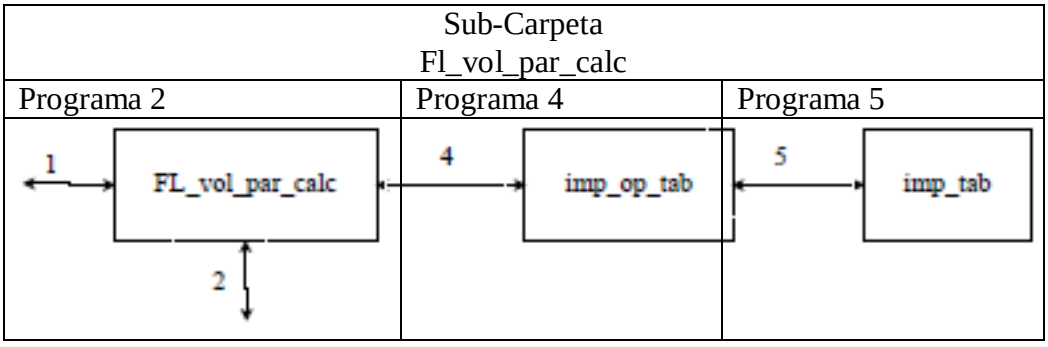
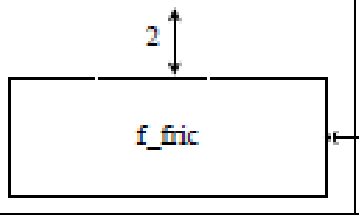
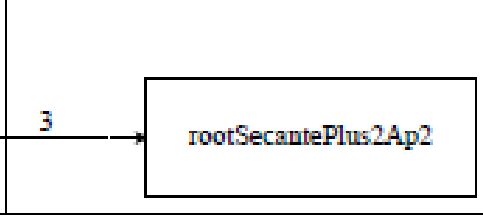


Tabla B.20. Relación entre f_fric y rootSecantePlus2Ap2 para el problema 9°.

Carpeta Externa f_fric	Carpeta Externa Metodos_numericos
Programa 2	Programa 4
	

B.3.4 *Operaciones Unitarias III*

Se consideraron los siguientes problemas para Operaciones Unitarias I:

A. Décimo Problema: "Composiciones de los productos de líquido y vapor, en procesos de destilación instantánea de mezclas multicomponentes"

Composiciones de los productos de líquido y vapor, en procesos de destilación instantánea de mezclas multicomponentes (Problema 10°)

El objetivo de los programas elaborados para este problema es calcular las composiciones en la fase líquida y vapor para una mezcla de compuestos, en procesos de vaporización instantánea o destilación en el equilibrio (operación de destilación de una sola etapa). En las Tablas A.21 – A.36 se esquematiza la forma en que se relacionan los programas escritos en Scilab y Python para el décimo problema:

- opdK, Kdatos, opcK y cpre_ing (véase Tabla A.21)
- coac, opcK y BurbP (véase Tabla A.22)
- Wcomp, NRTLcomp, UNIFAC, coac, BurbP y phiCM (véase Tabla A.23)
- tablas_read, UNIFAC, tab_amk y tab_Rk_Qk (véase Tabla A.24)
- imp_tab, imp_op_tab, BurbP y fpres_vap (véase Tabla A.25)
- opcK y RoP (véase Tabla A.26)
- Wcomp, NRTLcomp, UNIFAC, coac, RoP y phiCM (véase Tabla A.27)

- imp_tab, imp_op_tab, RoP y fpres_vap (véase Tabla A.28)
- opcK y BurbT (véase Tabla A.29)
- Wcomp, NRTLcomp, UNIFAC, coac, BurbT y phiCM (véase Tabla A.30)
- rootSecantePlus2Ap2, BurbT y fpres_vap (véase Tabla A.31)
- BurbT, imp_op_tab e imp_tab (véase Tabla A.32)
- opcK y RoT (véase Tabla A.33)
- Wcomp, NRTLcomp, UNIFAC, coac, RoT y phiCM (véase Tabla A.30)
- rootSecantePlus2Ap2, RoT y fpres_vap (véase Tabla A.31)
- RoT, imp_op_tab e imp_tab (véase Tabla A.32)

Datos que pide Kdatos:

1. Número que identifica el número de componentes de la solución[7] (2, 3 o 4)
2. Número que identifica el tipo de cálculo a realizar: cálculos de burbuja y rocío P (1) o cálculos de burbuja y rocío T (2)
3. Número que identifica la ecuación de equilibrio líquido vapor utilizada: Ley de Raoult (1), Ley de Raoult modificada o Formulación Gamma-Phi (3).

   Únicamente Formulación Gamma-Phi

   3.1. Número que identifica la opción elegida para definir si se cuenta con coeficientes del virial (1) o no (2), en caso se haya elegido el segundo grado de idealidad.

   3.2. Número que identifica la opción elegida para definir si se puede considerar a la fase vapor como solución ideal (1) o solución real (2), en caso se haya elegido la Formulación Gamma-Phi.
4. Temperatura del sistema
5. Presión del sistema
6. Conjunto de composiciones molares globales (“z”)
7. Número que identifica la opción elegida para impresión de tablas con las iteraciones realizadas: en la consola de Scilab o Spyder (1), archivos de extensión.CSV (pueden

[7] Los programas para este problema se trabajaron de tal manera que es posible especificar un número de componentes más alto, pero los ejemplos utilizados tratan soluciones con 2, 3 o 4 componentes.

ser abiertos en Excel) (2) o la no impresión de las tablas de iteraciones (se imprimirá únicamente la temperatura de burbuja) (3).

Datos que pide opc:

1. Número que identifica el método de cálculo de coeficientes de actividad: método de Wilson (1), método NRTL (2) o método UNIFAC (3).

Formualción Gamma-Phi

Se cuenta con coeficientes del virial

2. Conjunto de coeficientes del virial, en caso se cuenta con dichos coeficientes

No se cuenta con coeficientes del virial

3. Factores acéntricos, temperaturas críticas, volúmenes críticos, factores de compresibilidad críticos y presiones críticas, en caso se haya elegido el segundo grado de idealidad.

Datos que pide cpre_ing:

1. Número que identifica el libro elegido para la búsqueda de las constantes de las ecuaciones de cálculo de presiones de vapor: “Properties.of Gases and Liquids“, cuarta edición; “Properties of Gases and Liquids”, quinta edición; “Chemical Engineers’ Handbook”, séptima u octava edición; o “Introducción a la Termodinámica Química en Ingeniería Química”.
2. Número que identifica la ecuación elegida para el cálculo de presiones de vapor: Si se ha elegido a “Properties of Gases and Liquids”, cuarta edición, elegir entre la ecuación 1 (Wagner), la ecuación 2 (Antoine) o la ecuación 3 (Frost-Kalkwarf-Thodos). Si se ha elegido a “Properties of Gases and Liquids”, quinta edición, elegir entre la ecuación 1 (Antoine), la ecuación 2 (Antoine modifcada), y la ecuación 3 (Wagner modificada). Aquí únicamente se define el tipo de ecuación a utilizar, por lo tanto el usuario debe buscar las constantes en los libros referidos.
3. Ingresar la temperatura y presión crítica, en caso de elegir la ecuación de Wagner.

Tabla B.21. Relación entre Kdatos (programa principal) y opcK, opdK y cpre_ing para el problema 10°.

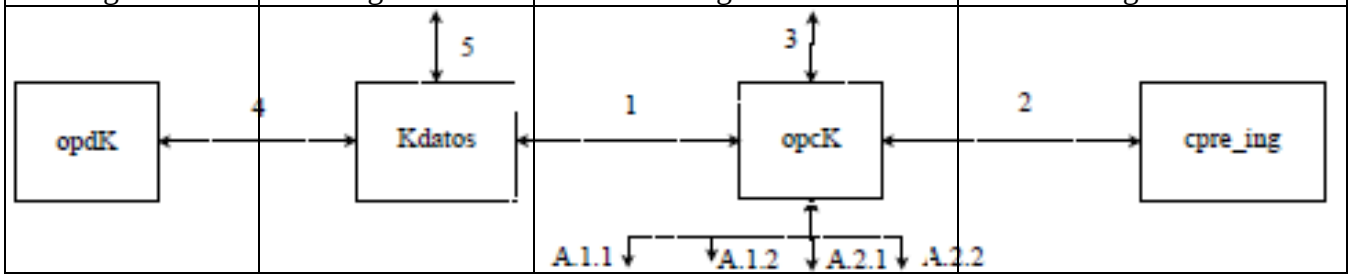
Sub-Carpeta opciones	Carpeta Principal Destilación	Sub-Carpeta opcionesK	Sub-Carpeta tsat_cal
Programa 4	Programa 1	Programa 2	Programa 3
			

Tabla B.22. Relación entre opcK, coac y BurP para el problema 10°

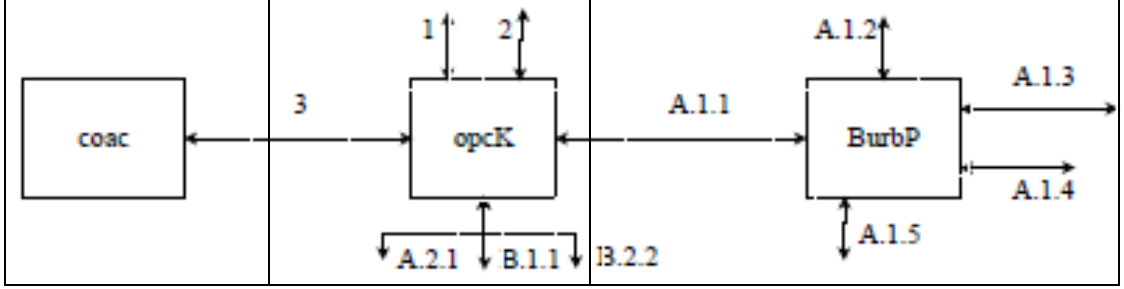
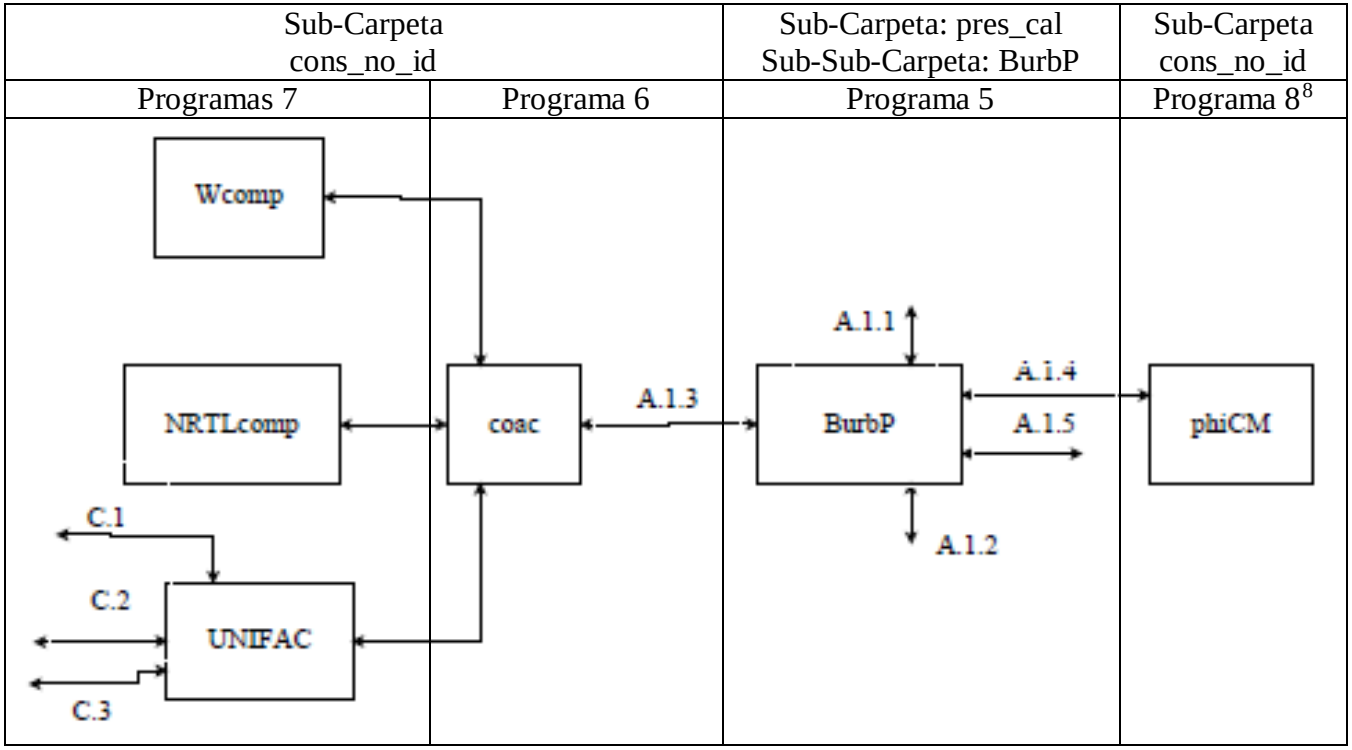
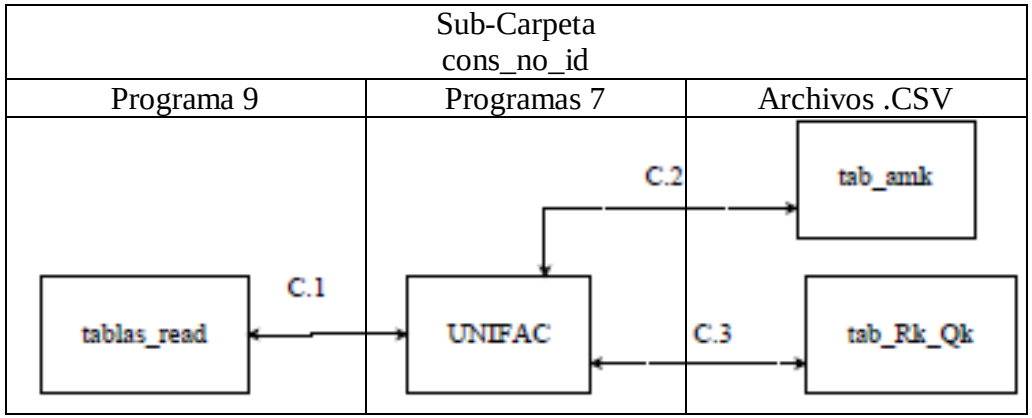
Sub-Carpeta cons_no_id	Sub-Carpeta opciones	Sub-Carpeta: pres_cal Sub-Sub-Carpeta: BurbujaP
Programa 6	Programa 2	Programa 5
		

Tabla B.23. Relación entre coac, Wcomp, NRTLcomp, UNIFAC, BurP y phiCM para el problema 10°



[8] El programa 8 se utilizará si se eligió el segundo grado de idealidad.

Tabla B.24. Relación entre tablas_read, UNIFAC, tab_amk y tab_Rk_Qk para el problema 10° [9]



[9] La relación entre el programa UNIFAC, tablas_read, tab_amk y tab_Rk_Qk sin importar si el programa que invoca a UNIFAC es BurbP, RoP, BurbT y RoT.

Tabla B.25. Relación entre imp_op_tab, imp_tab, BurP y fpres_vap para el problema 10°

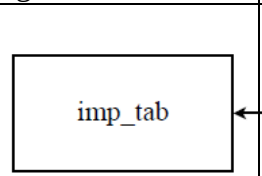
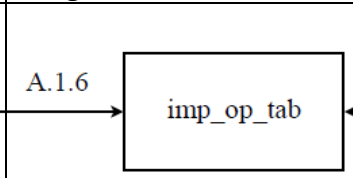
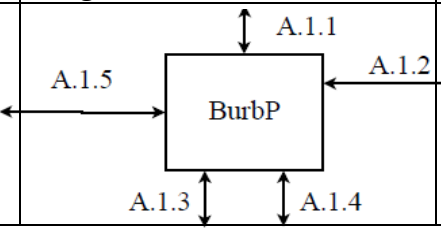
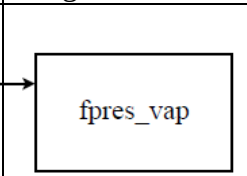
Sub-Carpeta opcionesK		Sub-Carpeta: pres_cal Sub-Sub-Carpeta: BurbujaP	Sub-Carpeta tsat_cal
Programa 13	Programa 12	Programa 5	Programa 11
			

Tabla B.26. Relación entre opcK y RoP para el problema 10°

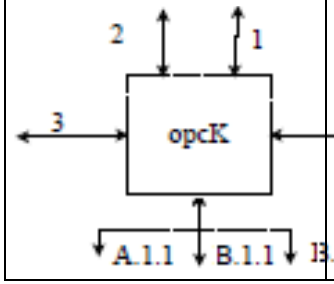
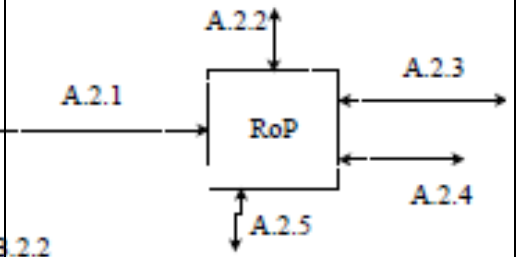
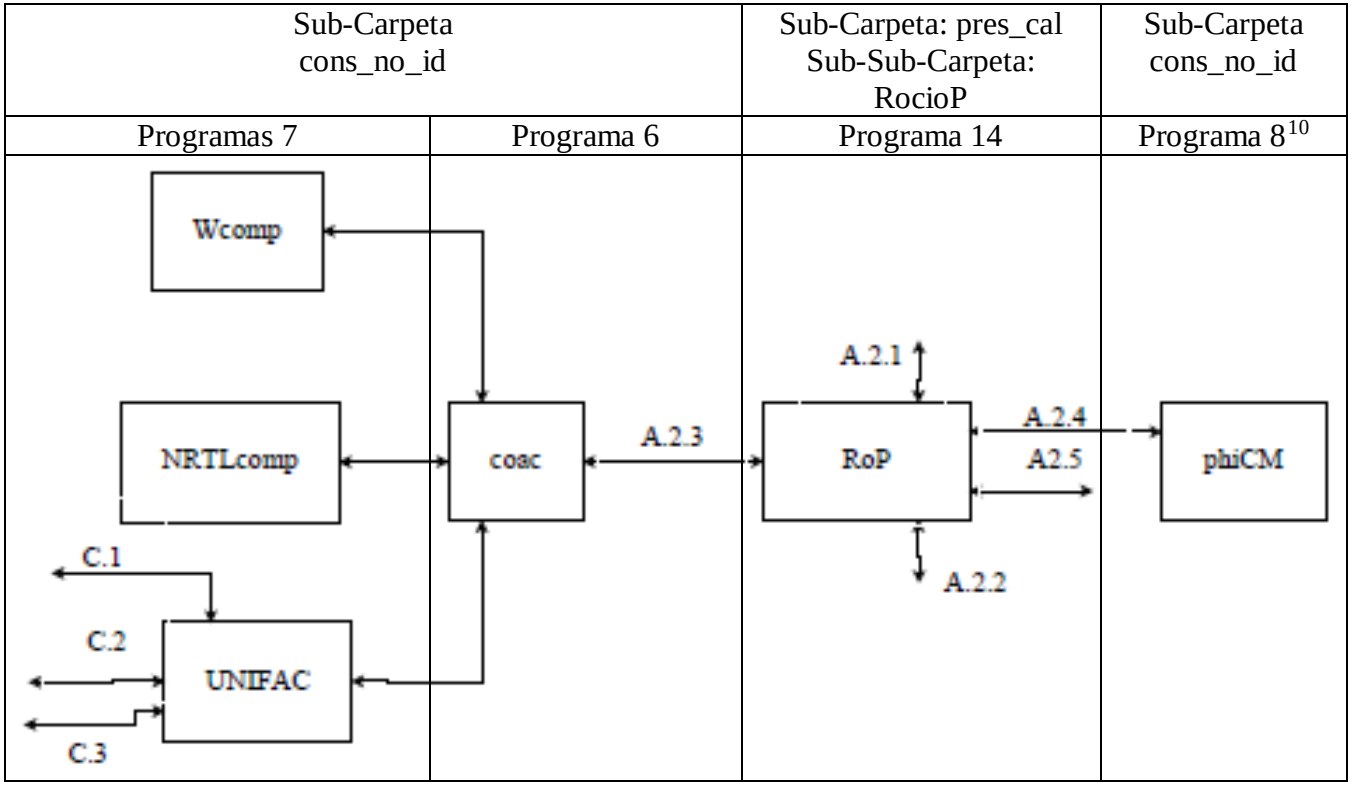
Sub-Carpeta opcionesK	Sub-Carpeta: pres_cal Sub-Sub-Carpeta: RocioP
Programa 2	Programa 14
	

Tabla B.27. Relación entre coac, Wcomp, NRTLcomp, UNIFAC, RoP y phiCM para el problema 10°



[10] El programa 8 se utilizará si se eligió el segundo grado de idealidad.

Tabla B.28. Relación entre imp_op_tab, imp_tab, RoP y fpres_vap para el problema 10°

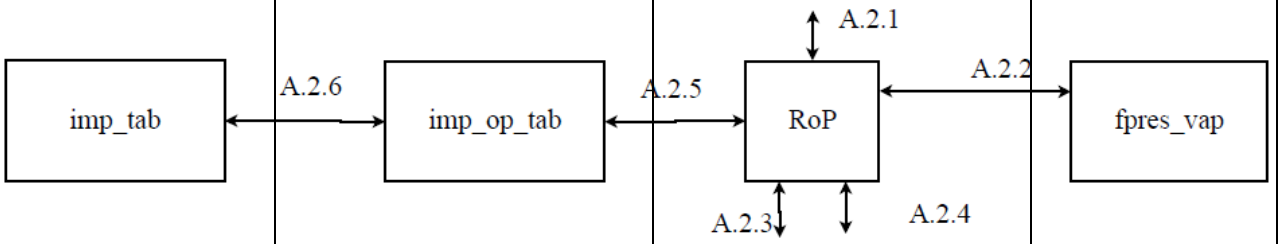
Sub-Carpeta opcionesK		Sub-Carpeta: pres_cal Sub-Sub-Carpeta: RocioP	Sub-Carpeta tsat_cal
Programa 13	Programa 12	Programa 14	Programa 11
			

Tabla B.29. Relación entre opcK y BurbT para el problema 10°

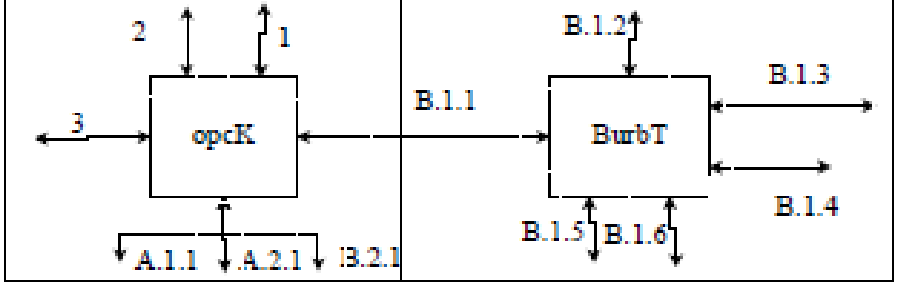
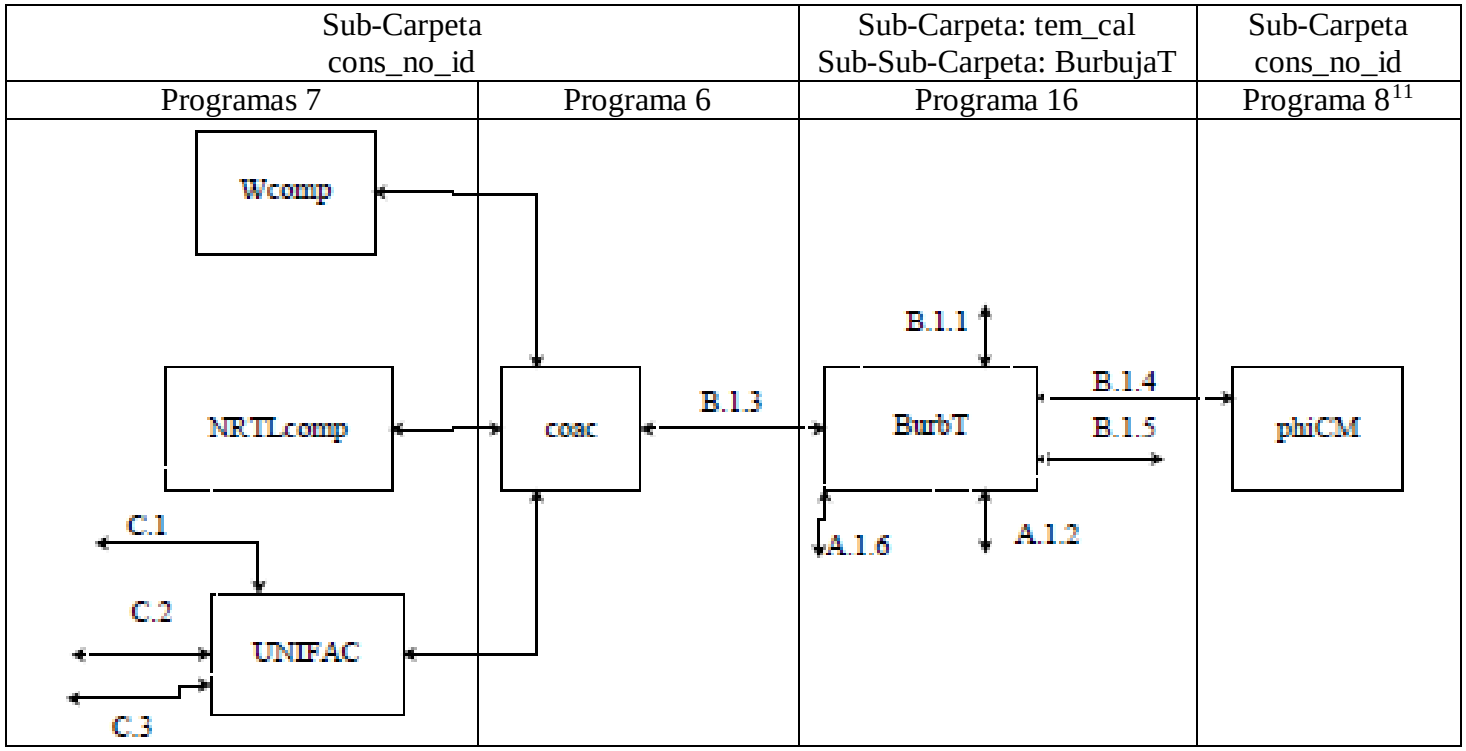
Sub-Carpeta opcionesK	Sub-Carpeta: tem_cal Sub-Sub-Carpeta: BurbujaT
Programa 2	Programa 16
	

Tabla B.30. Relación entre coac, Wcomp, NRTLcomp, UNIFAC, BurbT y phiCM para el problema 10°



[11] El programa 8 se utilizará si se eligió el segundo grado de idealidad.

Tabla B.31. Relación entre BurbT, rootSecantePlus2Ap2 y fpres_vap para el problema 10°

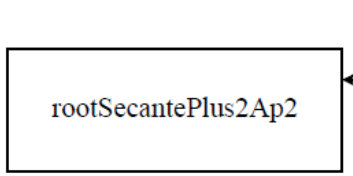
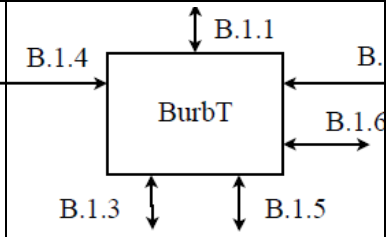
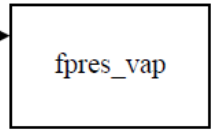
Sub-Carpeta Metodos_numericos	Sub-Carpeta: tem_cal Sub-Sub-Carpeta: BurbujaT	Sub-Carpeta tsat_cal
Programa 17	Programa 16	Programa 11
		

Tabla B.32. Relación entre BurbT, imp_op_tab e imp_tab para el problema 10°

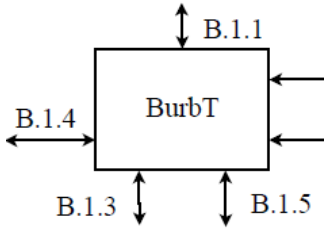
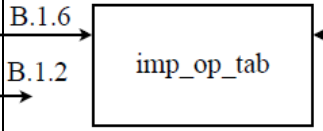
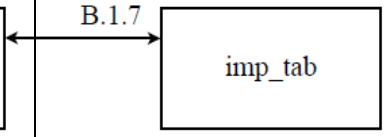
Sub-Carpeta: tem_cal Sub-Sub-Carpeta: BurbujaT	Sub-Carpeta opcionesK	
Programa 5	Programa 12	Programa 13
		

Tabla B.33. Relación entre opcK y RoT para el problema 10°

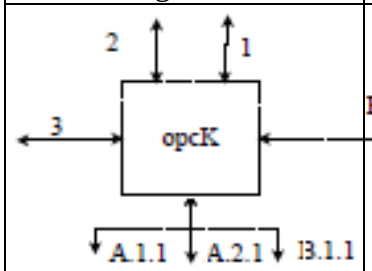
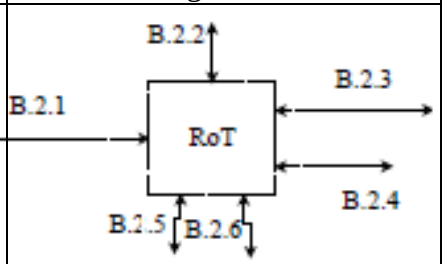
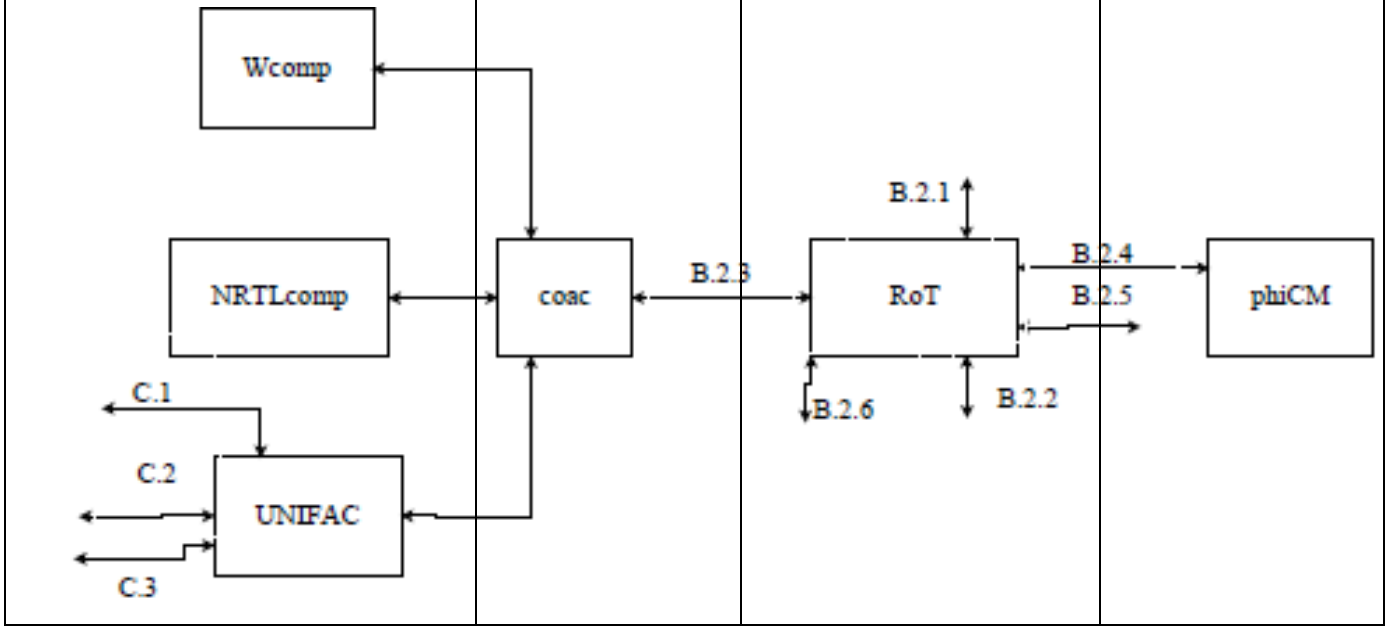
Sub-Carpeta opcionesK	Sub-Carpeta: tem_cal Sub-Sub-Carpeta: RocioT
Programa 2	Programa 18
	

Tabla B.34. Relación entre coac, Wcomp, NRTLcomp, UNIFAC, RoT y phiCM para el problema 10°

Sub-Carpeta cons_no_id		Sub-Carpeta: tem_cal Sub-Sub-Carpeta: RocioT	Sub-Carpeta cons_no_id
Programas 7	Programa 6	Programa 18	Programa 8[12]



[12] El programa 8 se utilizará si se eligió el segundo grado de idealidad.

Tabla B.35. Relación entre RoT, rootSecantePlus2Ap2 y fpres_vap para el problema 10°.

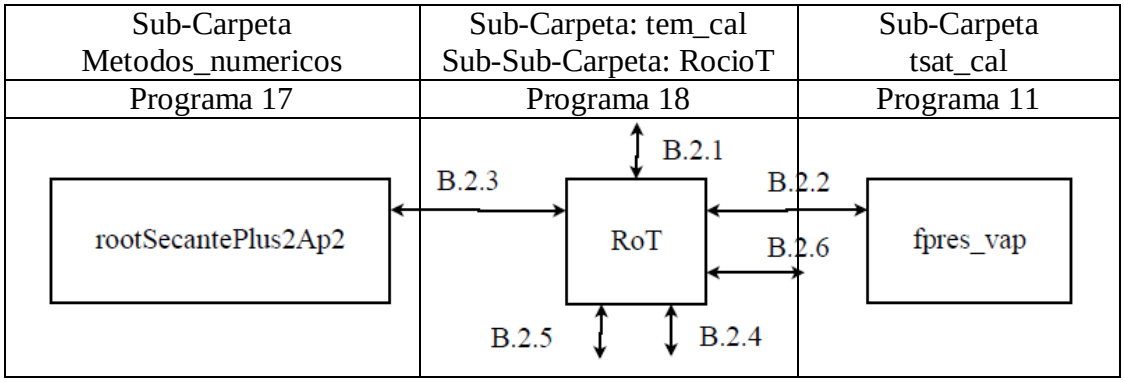
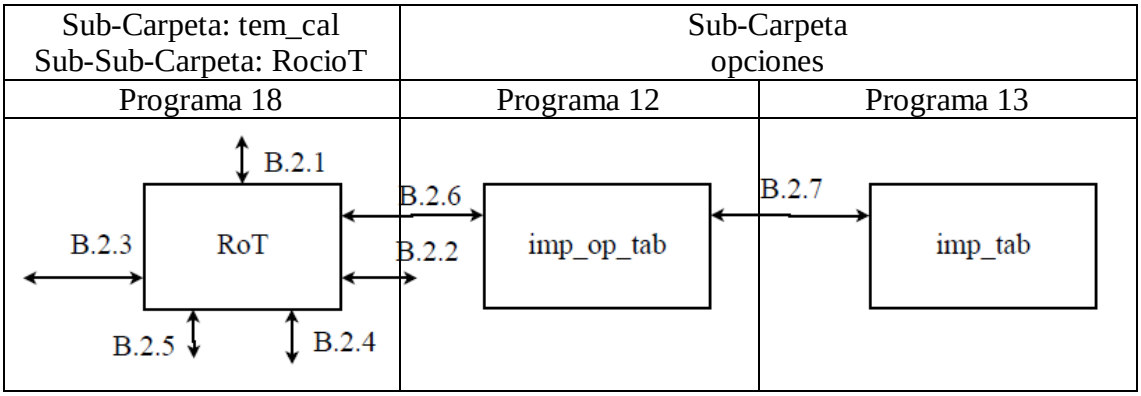


Tabla B.36. Relación entre RoT, imp_op_tab e imp_tab para el problema 10°.



Anexo C Seudocódigos para los programas elaborados en Scilab y Python

C.1 Métodos numéricos

RomberT

FUNCTION **TrapEq(nit**, **numc**, **a**, **b**, **lfun_pri)**

Asigna **lfun_pri**(1) a **fun**; Asigna **lfun_pri**(2) a **lfun_sec**; Asigna **lfun_pri**(3) a **pathDir**

Calcula (**b**-**a**)/**nit** y asigna el resultado a **h**
Asigna a **x** el valor de **a**
Evalúa **fun(x**, **lfun_sec**, **numc,pathDir)** y asigna el resultado a **suma**

FOR 1 < **i** < **nit** - 1[13]
Calcula **x** + **h** y asigna el resultado a **x**
Evalúa 2***fun(x**, **lfun_sec**, **numc**, **pathDir)** + **suma** y asigna el resultado a **suma**.
END FOR

Evalúa **fun(b**, **lfun_sec**, **numc, pathDir)** + **suma** y asigna el resultado a **suma**.
Calcula **suma** * **h/2**, y asigna el resultado a **res**
Devuelve **res**

END FUNCTION

FUNCTION **Romberg(numc**, **a**, **b**, **lfun**, **itermax**, **tol)**

Realiza **ZEROS(10,10)** y asigna el resultado a **I**
Asigna a **nit** el valor de 1

Evalúa **TrapEq(nit**, **numc**, **a**, **b, lfun)**, y asigna el resultado a la celda de **I**(1,1)
Asigna a **iter** el valor de 0
Asigna a **ea** el valor de 10; Calcula **tol** * 100 y asigna el resultado a **es**

WHILE **ea** >= **es**
Calcula 1 + **iter**, y asigna el resultado **iter**

[13] En Python, se suele ocupar 0 como el primer valor de la variable utilizada para iterar en ciclos while y for, hacer una extracción (de una lista) o referenciar la posición del primer elemento. También las variables utilizadas como contador tendrán un valor inicial de 0.

Calcula 2 ^ **iter** y asigna el resultado a **nit**
Evalúa **TrapEq(nit**, **numc**, **a**, **b, lfun)**, y asigna el resultado a la celda **I**(**iter** + 1, 1).

FOR 2< **k** < **it** + 1
Calcula 2 - **k** + **iter**, y asigna el resultado a **j**

Calcula (4^(**k**-1)***I**(**j**+1,**k**-1)-I(**j**,**k**-1))/(4^(**k**-1)-1) y asigna el resultado a la celda **I**(**j**, **k**) (fórmula de integración de Romberg).
END FOR

Calcula **ABS((I**(1,**iter**+1)-**I**(1,**iter**))/**I**(1,**iter**+1)**)***100, y asigna el resultado a **ea**.

IF **it** >= **itermax**
Salir del ciclo
END IF
END WHILE
Asigna el valor de la celda de **I**(1, **iter** + 1) a **ultI**
Forma una lista con los valores de **ultI** e **I** y asigna el resultado a **res**
Devuelve **res**

END FUNCTION

rootNewton_cubic2

FUNCTION **rootNewton_cubic2(x0, f, fp, param, dparam, tol, imax, store)**

Guarda 0 en **ve**(1); Asigna 1 a **iterv**
Guarda **x0** en **vx**(1); Almacena **vx**(**iterv**) en **rv**; Guarda **iterv** en **vi**(1)
Asigna **x0** a **x**

Evalúa **rootNewton_cubic_2_evalf(f**, **x, param)**, y asigna el resultado a **fx0**
Guarda **fx0** en **f_x**(1)
Evalúa **rootNewton_cubic_2_evalf(fp**, **x, dparam)** y asigna el resultado a **fpv**

Evalúa **x0** - **fx0**/ **fpv** y asigna el resultado a **r** (fórmula de Newton).
Suma 1 a **iterv** y guarda el resultado a **iterv**

Guarda **r** en **vx**(**iterv**); Almacena **vx**(**iterv**) en **rn;** Guarda **iterv** en **vi**(**iterv**)

IF **rn** ~= 0[14]
Calcula **ABS((rv** – **rn**)/**rn)***100, y lo guarda en **ve**(**iterv**)

[14] En Python, "distinto de" es !=.

ELSE
  Guarda 0 en **ve**(**iterv**)
END IF

Asigna 1 a **op**
WHILE la **ABS(r – x)** > **tol** AND **iterv** <= **imax**
   Asigna el valor de **r** a **x**; Asigna **fr** a **frw**
   Evalúa **rootNewton_cubic2_evalf(f**, **param**, **x),** y asigna el resultado a **fr**
   Evalúa **rootNewton_cubic2_evalf(fp**, **dparam**, **x)**, y asigna el resultado a **fpv**

   Evalúa **x0** - **fx0**/ **fpv** y asigna el resultado a **r** (fórmula de Newton).
   Suma 1 a **iterv** y guarda el resultado a **iterv**

   Agrega **r** a **vx(iterv);** Almacena **vx**(**iterv**) en **rn;** Almacena **vx**(**iterv**-1) en **rv**
   Guarda **iterv** en **vi**(**iterv**); Guarda **fr** en **f_x**(**iterv**)

   IF **rv** ~= 0
      Calcula **ABS((rv** – **rn)**/**rn)***100, y guarda en **ve**(**iterv**)
   ELSE
      Almacena **ve**(**iterv**-1) en **ve**(**iterv**)
   END IF

   IF **r-x** da como resultado un valor real
      IF **ABS(r-x)** < **tol**
         Asigna **ABS(frw** – **fr)** a **dif_fx**
         IF **dif_fx** > **tol**
            Asigna 1 a **op**
         ELSE
            Asigna 2 a **op**
         END IF
      END IF
   ELSE
      PRINT “se ha encontrado una raíz no real”
   END IF

   IF **iterv** >= **imax**
      Sale del ciclo
   END IF
END WHILE

IF **store** == **%T**[15]
   Crea una matriz con un número de filas igual a **iterv** y 4 columnas, y asigna el resultado a **vm**

[15] En Python, el valor de verdadero es **True**

Asigna **vi** a **vm**(:,1); Asigna **vx** a **vm(**:,2**);** Asigna **f_x** a **vm**(:,3);
Asigna **v_e** a **vm**(:,4)
ELSE
Agrupa a **iterv**, **vx**(**iterv**), **f_x**(**iterv**), **ve**(**iterv**) y asigna el resultado a **vm**
END IF
Devuelve **vm**

END FUNCTION

FUNCTION **rootNewton_cubic2_evalf(f, x, param)**

Crea una cadena de caracteres con el fin de evaluar posteriormente **f(x**, **param**(1:$)**)**, y asigna el resultado a **y**
Ejecuta el código de la cadena de caracteres **y** y guarda el resultado en **err**

IF **ierr** ~= 0
PRINT "error"
END IF
Devuelve **y**

END FUNCTION

rootBracket_3p3

Function **rootBracket_3(fun, x0, param, h)**

Asigna el valor de **x0** a **xl**
Evalúa **rootBracket_evalf(fun**, **param**,**xl)**, y asigna el resultado a **fxl**
Calcula **h** + **xl** y asigna el resultado a **xu**
Evalúa **rootBracket_evalf(fun**, **param, xu)** y asigna el resultado a **fxu**

Establece **SIGN(fxl)** != **SIGN(fxu)** y almacena el resultado en **done**

IF **done** ~= **%T**
IF el **ABS(fxu)** > **ABS(fxl)**
Asigna –**h** a **h**
Asigna **xl** a **xu**
Guarda **fxl** en **fxu**
END IF
END IF

WHILE **done** ~= **%T**

```
        Guarda el valor de xu en xl
        Asigna a fxl, el valor de fxu
        Almacena 2*h en h

       Suma h a xl y guarda el resultado en xu
       Evalúa rootBracket_evalf(fun, param, xu) y asigna el resultado a fxu
       Establece SIGN(fxl) != SIGN(fxu), son distintos y almacena el resultado en done

        IF ABS(fxu) > ABS(fxl) AND done ~= %T
           Lanza un mensaje de error
       END IF
   END WHILE

   IF xu < 0
      Asigna el valor de 0 a xu
   ELSEIF xl < 0
      Asigna el valor de 0 a xl
   END IF
   Devuelve xl y xu

END FUNCTION
```

```
FUNCTION rootBracket_evalf(fun, param, x)

   IF LENGTH(param) ~= 0[16]
     Crea una cadena de caracteres con el fin de evaluar posteriormente fun(x, param), y
     asigna el resultado a y

   ELSEIF param ==0
     Crea una cadena de caracteres con el fin de evaluar posteriormente fun(x) y asigna el
     resultado a y
   END IF

   Ejecuta el código de y y guarda el resultado en err
   IF err ~= 0
      Lanza un mensaje de error
   END IF
   Devuelve y

END FUNCTION
```

[16] Para los programas elaborados en Python, se ocupa el valor “None” en lugar del 0.

rootRegulaFA2p2

FUNCTION **rootRegulaFA2(x0, h, fun, param, tol, imax, store, pathDir)**

Utiliza **pathDir** para establecer el directorio de rootBracket_3p3, y asigna el resultado a **pathBracket**;
Utilizando **pathBracket**, llama al espacio de trabajo, a **rooBracket_3** desde rootBracket_3p3.
Evalúa **rootBracket_3(fun**, **param**, **x0**, **h)**, esto genera dos valores que son asignados a **xl** y **xu**

Guarda 0 en **ve**(1); Asigna 1 a **iterv**

Evalúa **rootRegulaFA2_evalf(fun**, **param**, **xl)**, y asigna el resultado a **fxl**
Evalúa **rootRegulaFA2_evalf(fun, param, xu)**, y asigna el resultado a **fxu**

Calcula **xu** – **fxu** * (**xl** - **xu**)/(**fxl** - **fxu**) (fórmula del método de Regula-Falsi) y asigna el resultado a **x**
Evalúa **rootRegulaFA2_evalf(fun, param, x)**, y asigna el resultado a **fx**

IF **SIGN(fxl)** == **SIGN(fxu)**
Asigna **x** a **xl;** Asigna **fx** a **fxl**

ELSEIF **SIGN(fx)** == **SIGN(fxu)**
Asigna **x** a **xu**; Asigna **fx** a **fxu**

ELSE
Asigna **x** a **xl;** Asigna **x** a **xu**
END IF

Guarda **iterv** en **vi**(1); Guarda **x** en **vx**(1); Guarda **fx** en **f_x**(1)
Suma 1 a i**terv** y asigna el resultado a **iterv**

IF **fxl** * **fxu** > 0
Lanza un error
END IF

WHILE **ABS((xu** – **xl)/2)** > **tol** AND **iterv** < **imax**
Calcula **xu** – **fxu** * (**xl** - **xu**)/(**fxl** - **fxu**) (fórmula del método de Regula-Falsi) y asigna el resultado a **x**
Evalúa **rootRegulaFA2_evalf(fun**, **param, x)**, y asigna el resultado a **fx**

IF **SIGN(fxl) == SIGN(fxu)**

Asigna **x** a **xl;** Asigna **fx** a **fxl**

ELSEIF el signo de **fx** y el signo de **fxu** son iguales
Asigna **x** a **xu**; Asigna **fx** a **fxu**

ELSE
Asigna **x** a **xl;** Asigna **x** a **xu**
END IF

Guarda **iterv** en **vi**(**iterv**); Guarda **x** en **vx**(**iterv**)
Guarda **fx** en **f_x**(**iterv**)
Almacena **vx**(**iterv**) en **rn**
Almacena **vx**(**iterv**– 1) en **rv**

IF **iterv** >= 2
IF **rv** ~= 0
Calcula **ABS(**100*(**rv** – **rn**)/**rn)**, y lo guarda en **ve**(**iterv**)
ELSE
Almacena **ve**(**iterv** – 1) y lo almacena en **ve**(**iterv**)
END IF
END IF
Suma 1 a i**terv** y asigna el resultado a **iterv**
END WHILE

IF **store** == **%T**
Crea una matriz con un número de filas igual a **iterv** y 4 columnas, y asigna el resultado a **vm**
Asigna **vi** a **vm**(:,1)**;** Asigna **vx** a **vm**(:,2); Asigna **f_x** a **vm**(:,3);
Asigna **v_e** a **vm**(:,4)
ELSE
Agrupa **iterv, vx**(**iterv**), **f_x**(**iterv**), y **ve**(**iterv**), y asigna el resultado a **vm**.
END IF
Devuelve **vm**

END FUNCTION

FUNCTION **rootRegulaFA2_evalf(fun, x, param)**

IF **param** ~= 0
Crea una cadena de caracteres con el fin de evaluar posteriormente **fun(x, param)**, y asigna el resultado a **y**
ELSEIF **param** ==0
Crea una cadena de caracteres con el fin de evaluar posteriormente **fun(x)**, y asigna el resultado a **y**

```
   ELSE
      PRINT “error”
   END IF
   Ejecuta el código de la cadena de caracteres y y guarda el resultado en err

   IF err ~= 0
      Lanza un mensaje de error
   END IF
   Devuelve y

END FUNCTION
```

rootRegulaFA_mod

```
FUNCTION rootRegulaFA_mod(x0, fun, param, tol, imax, store, bracket)

   Asigna el primer elemento de bracket a xl; Asigna el segundo elemento de bracket a
   xu
   Guarda 0 en ve(1); Asigna 1 a iterv

   Evalúa rootRegulaFA2_evalf(fun, param, xl), y asigna el resultado a fxl
   Evalúa rootRegulaFA2_evalf(fun, param, xu), y asigna el resultado a fxu

   Calcula xu – fxu * (xl - xu)/( fxl - fxu) (fórmula del método de Regula-Falsi) y asigna
   el resultado a x
   Evalúa rootRegulaFA2_evalf(fun, param, x), y asigna el resultado a fx

   IF SIGN(fxl) == SIGN(fxu)
      Asigna x a xl; Asigna fx a fxl
   ELSEIF SIGN(fx) == SIGN(fxu)
     Asigna x a xu; Asigna fx a fxu
  ELSE
     Asigna x a xl
     Asigna x a xu
   END IF

   Guarda iterv en vi(1); Guarda x en vx(1); Guarda fx en f_x(1)
   Suma 1 a iterv y asigna el resultado a iterv

   IF fxl * fxu >= 0
      PRINT “error”
   END IF
   Asigna 0 a iu; Asigna 0 a il
```

```
Asigna 1 a op
WHILE op == 1
   Calcula xu – fxu * (xl - xu)/( fxl - fxu) (fórmula del método de Regula-Falsi) y
   asigna el resultado a x

    Asigna fx a fxw
    Evalúa rootRegulaFA_mod_evalf(fun, param, x), y asigna el resultado a fx

    IF SIGN(fxl) == SIGN(fxu)
       Asigna x a xl; Asigna fx a fxl
       Calcula 1 + iu y asigna el resultado a iu
       IF iu >= 2
          Calcula fxu /2 y asigna el resultado a fxu
        END

   ELSEIF SIGN(fx) == SIGN(fxu)
       Asigna x a xu; Asigna fx a fxu
       Calcula 1 + il y asigna el resultado a il
       IF il >= 2
          Calcula fxl/2 y asigna el resultado a fxl
        END

   ELSE
       Asigna x a xl; Asigna x a xu
   END IF
   Guarda iterv en vi(iterv); Guarda x en vx(iterv); Guarda fx en f_x(iterv)
   Almacena vx(iterv) en rn; Almacena vx(iterv – 1) en rv

   IF ABS((xu – xl)/2) es real
      IF ABS((xu – xl)/ 2) < tol
         Calcula ABS(fxw – fx) y asigna el resultado a dif_fx

         IF dif_x > tol
            Asigna 1 a op
         ELSE
            Asigna 2 a op
         END IF
      END IF
   ELSE
      PRINT “error”
      Sale del ciclo
   END IF
   IF iterv >= 2
      IF rv ~= 0
```

```
            Calcula ABS((rv – rn )/rn)*100, y agrega el resultado como elemento de ve
          ELSE
            Almacena ve(iterv – 1) en ve(iterv)
          END IF
       END IF
       Calcula 1 + iterv y asigna el resultado a iterv
       IF iterv > imax
          Sale del ciclo
       END IF
    END WHILE

    IF store es igual al valor %T
       Crea una matriz con un número de filas igual a iterv y 4 columnas, y asigna el
       resultado a vm
       Asigna vi a vm(:,1); Asigna vx a vm(:,2); Asigna f_x a vm(:,3)
       Asigna v_e a vm(:,4)
    ELSE
       Agrupa iterv, vx(iterv), f_x(iterv), y ve(iterv) y asigna el resultado a vm
    END IF
    Devuelve vm

END FUNCTION
```

```
FUNCTION rootRegulaFA_mod_evalf(fun, param, x)

    IF param ~= 0
       Crea una cadena de caracteres con el fin de evaluar posteriormente fun(x, param),
       y asigna el resultado a y
    ELSEIF param ==0
       Crea una cadena de caracteres con el fin de evaluar posteriormente fun(x), y asigna
       el resultado a y
    ELSE
       PRINT "error"
    END IF

    Ejecuta el código de la cadena de caracteres y y guarda el resultado en err

    IF err ~= 0
       PRINT "error"
    END IF
    Devuelve y

END FUNCTION
```

rootSecantePlus2Ap2

FUNCTION **fp(x, f, param)**

Asigna 0.001 a **h**
Agrupa **x**-2*__h__, **x**-**h**, **x**+**h**, **x**+2*__h__ y asigna el resultado a **xv**.
Asigna 1 a **i**

WHILE **i** < 4
Evalúa **rootSecantePlus3A_evalf(f**, el elemento número **i** de **xv, param)**, y asigna el
resultado a **f_xv**; Guarda **f_xv** en **fxv**(**i**)
Calcula 1 + **i** y asigna el resultado a **i**
END WHILE

Calcula (-**fxv**(4)+8*__fxv__(3)-8*__fxv__(2)+**fxv**(1))/(12*__h__) y asigna el resultado a **fun**;
Devuelve **fun**

END FUNCTION

FUNCTION **rootSecantePlus2A(x0, fun, param, tol, imax, store)**

Guarda 0 en **ve**(1); Asigna 1 a **iterv;** Guarda **x0** en **vx**(1)
Almacena **vx**(1) en **rv;** Suma 1 a **iterv** y agrega el resultado a **vi**

Evalúa **rootSecantePlus2A_evalf(fun**, **x0, param)** y asigna el resultado a **fx**

Evalúa **fp(fun, x0, param)**, y asigna el resultado a **fx_p**
Guarda **fx** en **f_x**(**iterv**); Asigna **x0** a **x**

Evalúa **x0** - **fx0**/**f_xp** y asigna el resultado a **r** (fórmula de Newton con derivada estimada)

Calcula 1 + **iterv** y guarda el resultado a **iterv**

Guarda **r** en **vx**(**iterv**); Almacena **vx**(**iterv**) en **rn;** Guarda **iterv** en **vi**(**iterv**)
Evalúa **rootSecantePlus2A_evalf(fun**, **r, param)**, y asigna el resultado a **fx;**
Evalúa **fp(fun, r, param)**, y asigna el resultado a **fx_p**
Guarda **fx** en **f_x**(**iterv**)

IF **r** < 0
Asigna 0 a **r**
END IF

```
IF rn ~= 0
   Calcula ABS((rv – rn)/rn)*100 y lo guarda en ve(iterv)
ELSE
  Guarda 0 en ve(iterv)
END IF

Asigna 1 a op
WHILE op ==1
     Asigna r a x
     Calcula x0 - fx/fx_p, luego guarda este valor en r
     Suma 1 a iterv, y asigna el resultado a iterv
    Agrega r a vx; Almacena vx(iterv) en rn; Almacena vx(iterv – 1) en rv
    Suma 1 a iterv y agrega el resultado a vi

    Asigna fx a fxw
    Evalúa rootSecantePlus2A_evalf(fun, r, param), y asigna el resultado a fx
    Evalúa fp(fun, r, param), y asigna el resultado a fx_p
    Guarda fx en f_x(iterv)

    IF rn ~= 0
        Calcula ABS((rv – rn)/rn)*100, y lo guarda en ve(iterv)
    ELSE
        Guarda ve(iterv – 1) en ve(iterv)
   END IF

   IF r – x es real
      IF ABS(r – x) > tol
         Calcula ABS(fxw – fx), y asigna el resultado a dif_fx
         IF dif_fx > tol
             Asigna 1 a op
         ELSE
             Asigna 2 a op
        END IF
      END IF
   ELSE
      PRINT “error”; Sale del ciclo
   END IF

   IF iterv >= imax
      Sale del ciclo WHILE
   END IF
END WHILE

IF store == %T
```

```
      Asigna vi a vm(:,1); Asigna vx a vm(:,2); Asigna f_x a vm(:,3)
      Asigna v_e a vm(:,4)
   ELSE
      Agrupa iterv, vx(iterv), f_x(iterv) y ve(iterv), y asigna el resultado a vm
   END IF

   Devuelve vm
END FUNCTION
```

```
FUNCTION rootSecantePlus2A_evalf(fun, x, param)

   IF param ~=0
      Crea una cadena de caracteres con el fin de evaluar posteriormente fun(x, param),
      y asigna el resultado a y

   ELSEIF param ==0
      Crea una cadena de caracteres donde se evalúa fun(x), y asigna el resultado a y

   ELSE
      PRINT "error"
   END IF

   Ejecuta el código de la cadena de caracteres y y guarda el resultado en err

   IF err ~= 0
      PRINT "error"
   END IF
   Devuelve y

END FUNCTION
```

C.2 Asignatura: Termodinamica Quimica I

Cálculo del volumen molar en ecuaciones cúbicas de estado, utilizando parámetros de la ecuación de Van der Waals, Redlich-Kwong, Soave-Redlich-Kwong, Peng-Robinson; y del calculo del volumen molar mediante la ecuación de Kammerlingh-Onnes y las correlaciones para el segundo y el tercer coeficientes del virial (Problemas 1° y 2°).

fv2

FUNCTION **fcg(z, pr, tr, Bs, Cs)**

Calcula **pr**/**tr** y lo asigna a **facpt**
Calcula **z** – 1 – **Bs** * **facpt**/**z** – **Cs** * (**facpt**/**z**)^2 y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **dfcg(z, pr, tr, Bs, Cs)**

Calcula **pr**/**tr** y lo asigna a **facpt**
Calcula 1+**Bs** * **facpt**/**z****2 – 2 * **Cs** * (**facpt**/**z**)^3 y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **ffz(z, fact1, fact2, fact3)**

Calcula **z**^3 + **fact1** * **z**^2 + **fact2** * **z** + **fact3** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **dffz(z, fact1, fact2)**

Calcula 3 * **z**^2 + 2 * **fact1** * **z** + **fact2** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **fv2(t, tc, p, pc, op, opn, op_extra, r_prin, fac)**

Calcula **t**/**tc** y guarda el resultado en **tr**
Calcula **p**/**pc** y guarda el resultado en **pr**

IF **fac** == –100[17]
SELECT **opn**
CASE 1[18]
Asigna 1 a **alf;** Asigna 1/8 a **ome;** Asigna 27/64 a **psi**
Asigna 0 a **sig;** Asigna 0 a **epsi**
CASE 2
Calcula **tr**^-0.5 y asigna el resultado a **alf**
Asigna 0.08664 a **ome;** Asigna 0.42748 a **psi;** Asigna 1 a **sig;** Asigna 0 a **epsi**
END SELECT

ELSEIF **fac** ~= –100
SELECT **opn**
CASE 3
Calcula 0.48 + 1.574 * **fac** - 0.176***fac**^2 (ecuación de Soave-Redlich-Kwong) y lo asigna a **facf**
Calcula (1+**facf***(1 - **tr**^0.5))^2 y lo asigna a **alf**
Asigna 0.08664 a **ome;** Asigna 0.42748 a **psi;** Asigna 1 a **sig;** Asigna 0 a **epsi**
CASE 4
Calcula 0.37464 + 1.5422 * **fac** - 0.26992 * **fac**^2 (ecuación de Peng-Robinson) y lo asigna a **facf**
Calcula (1+**facf** * (1 - **tr**^0.5))^2 y lo asigna a **alf**
Asigna 0.0778 a **ome;** Asigna 0.45724 a **psi**
Calcula 1 + 2^0.5 y asigna el resultado a **sig**
Calcula 1 - 2^0.5 y asigna el resultado a **epsi**
CASE 5
Calcula 0.083 - 0.422/(**tr**^1.6) y lo asigna **B0**
Calcula 0.139 - 0.172/(**tr**^4.2) y lo asigna a **B1**
Calcula **B0** + **fac** * **B1** y lo asigna a **Bs**

[17] En Python, en lugar de -100 se ocupó None
[18] Para los programas de Python se ocupó una estructura anidad de IF, ELSEIF porque no cuenta con la estructura de selección CASE.

```
      Calcula 0.01407 + 0.02432/tr - 0.00313/(tr^10.5) y lo asigna C0
      Calcula -0.02676 + 0.05539/(tr^2.7) - 0.00242/(tr^10.5) y lo asigna a C1
      Calcula C0 + fac * C1 y lo asigna a Cs

      Calcula B0 + fac * B1 y lo asigna a Bs
      Agrupa a pr, tr, Bs y Cs y guarda el resultado en dat
      Asigna dat a ddat
   END SELECT
END IF

Asigna 1 a z0
Asigna 0.00001 a tol
Asigna 100 a imax

IF opn ==5
   IF op == 1 OR op ==3
      IF op ==1
         Utilizando a r_prin, establece el directorio de rootRegulaFA2p2 y lo asigna a
         r4
         Utilizando r4, llama al espacio de trabajo a rootRegulaFA2, desde
         rootRegulaFA2p2
         Utilizando a r_prin, establece el directorio de "\Metodos numéricos" y lo
         asigna a regBracket
         Utilizando regBracket, llama al espacio de trabajo a rootBracket_3, desde
         rootBracket_3p3

         Asigna 0.0005 a h
         Evalua rootRegula2(z0, h, fcg, dat, tol, imax, store = False, regBracket) y
         asigna el resultado a vm
         Almacena vm(2) y lo asigna a lz
      ELSE
         Utilizando a r_prin, establece el directorio de rootNewton_cubic2 y lo asigna a
         r3
         Utilizando r3, llama al espacio de trabajo a rootNewont_cubic2, desde
         rootNewton_cubic2
         Evalua rootNewton_cubic2(z0, fcg, dfcg, dat, ddat, tol, imax, store = False)
         y asigna el resultado a vm
      END IF

   ELSE
      Calcula pr/tr y asigna el resultado a facpt
      Calcula 1 + Bs * facpt/z0 + Cs * (facpt/z0)^2 y asigna el resultado a z
      Calcula ABS(z – z0) y asigna el resultado a dif
```

WHILE **dif** > 0.00001
Asigna **z0** a **z**
Calcula **pr**/**tr**, y asigna el resultado a **facpt**
Calcula 1 + **Bs** * **facpt**/**z0** + **Cs** * (**facpt**/**z0**)^2 y asigna el resultado a **z**
Calcula **ABS(z – z0)** y asigna el resultado a **dif**
END WHILE
Asigna **z** a **lz**
END IF

ELSE
Calcula **ome** * **pr**/**tr** y el resultado lo asigna **bet**
Calcula **psi** * **alf**/(**ome** * **tr**), y el resultado lo asigna a **q**
Calcula **epsi** * **bet** + **sig** * **bet** – 1 - **bet** y asigna el resultado a **fact1**
Calcula **epsi*****sig*****bet**^2-**epsi*****bet**-**sig*****bet**-**epsi*****bet**^2-**sig*****bet**^2+**q*****bet** y asigna el resultado a **fact2**
Calcula -**epsi*****sig*****bet**^2-**epsi*****sig*****bet**^3-**q*****bet**^2 y asigna el resultado a **fact3**
Agrupa a **fact1**, **fact2** y **fact3** y asigna el resultado a **cof**

Asigna 1 a **z0v**
IF **op** ==1 OR **op** ==3
IF **op** ==1
Utilizando a **r_prin**, establece el directorio de rootRegulaFA2p2 y lo asigna a **r4**
Utilizando **r4**, llama al espacio de trabajo a **rootRegulaFA2**, desde rootRegulaFA2p2
Utilizando a **r_prin**, establece el directorio de "\Metodos numéricos" y lo asigna a **regBracket**
Utilizando **regBracket**, llama al espacio de trabajo a **rootBracket_3**, desde rootBracket_3p3

Asigna 0.0005 a **h**
Evalua **rootRegula2(z0v**, **ffz**, **cof**, **h**, **tol**, **imax**, **r_prin**, **store** = **False)** y asigna el resultado a **vm**
Almacena **vm**(2) y lo asigna a **zv**
Asigna **zv** a **lz**

ELSE
Utilizando a **r_prin**, establece el directorio de rootNewton_cubic2 y lo asigna a **r3**
Utilizando **r3**, llama al espacio de trabajo a **rootNewont_cubic2**, desde rootNewton_cubic2
Evalua **rootNewton_cubic2(z0v**, **ffz**, **dffz**, **cof**, **dcof**, **tol**, **imax**, y **store** = **False)** y asigna el resultado a **vm**
Almacena **vm**(2) en **zv;** Asigna **zv** a **lz**

```
    END IF

  ELSE
    Calcula (z0v-bet)/((z0v+epsi*bet)*(z0v+sig*bet)) y asigna el resultado a fracv
    Calcula 1+bet-q*bet*fracv y asigna el resultado a zv
    Calcula ABS(zv – zv0) y asigna el resultado a dif
    Asigna 100 a itermax; Asigna 0 a it
    WHILE dif > 0.00001
      Asigna z0v a zv
      Calcula (z0v-bet)/((z0v+epsi*bet)*(z0v+sig*bet)) y asigna el resultado a
      fracv
      Calcula 1+bet-q*bet*fracv y asigna el resultado a zv
      Calcula ABS(zv – zv0) y asigna el resultado a dif
      Suma 1 a it y el resultado lo asigna a it

      IF it > itermax
         Sale del ciclo WHILE
      END IF
    END WHILE

    IF it > itermax
      Asigna 0 a lz
    ELSE
     Asigna zv a lz
  END IF

  Asigna It a it1
  IF op_extra ==1
    Asigna bet a z0l
    IF op ==1 OR op ==3
      IF op ==1
        Utilizando a r_prin, establece el directorio de rootRegulaFA2p2 y lo asigna
        a r4
        Utilizando r4, llama al espacio de trabajo a rootRegulaFA2, desde
        rootRegulaFA2p2
        Utilizando a r_prin, establece el directorio de "\Metodos numéricos" y lo
        asigna a regBracket
        Utilizando regBracket, llama al espacio de trabajo a rootBracket_3, desde
        rootBracket_3p3
        Evalua rootRegula2(z0l, ffz, cof, h, tol, imax, r_prin, store = %F)[19] y
        asigna el resultado a vm
        Almacena vm(2) en zl
        Agrupa a zv, zl, bet, q, sig y epsi y lo asigna a lz
```

[19] En Python, el valor de Falso es **False**

```
          ELSE
             Utilizando a r_prin, establece el directorio de rootNewton_cubic2 y lo
             asigna a r3
             Utilizando r3, llama al espacio de trabajo a rootNewont_cubic2, desde
             rootNewton_cubic2

             Evalua rootNewton_cubic2(z0l, ffz, dffz, cof, dcof, tol, imax, store = %F)
             y asigna el resultado a vm
             Almacena vm(2) en zl
             Agrupa zv, zl, bet, q, sig y epsi y lo asigna a lz
          END IF

       ELSE
          Calcula (1+bet-z0l)/(q*bet) y asigna el resultado a fracl
          Calcula bet+(z0l+epsi*bet)*(z0l+sig*bet)*fracl y asigna el resultado a zl
          Calcula ABS(zl – z0l) y asigna el resultado a dif

          Asigna 100 a itermax; Asigna 0 a it2
          WHILE dif > 0.00001
             Asigna z0l a zl
             Calcula (1+bet-z0l)/(q*bet) y asigna el resultado a fracl
             Calcula bet+(z0l+epsi*bet)*(z0l+sig*bet)*fracl y asigna el resultado a zl

             Calcula ABS(zl – z0l) y asigna el resultado a dif
             Suma 1 a it2 y el resultado lo asigna a it2
             IF it2 > itermax
                Sale del ciclo WHILE
             END IF
          END WHILE

          IF it1 > itermax AND it2 > itermax
             Agrupa zv, zl, bet, q, sig y epsi en una lista y asigna el resultado a lz
          ELSE
             Asigna 0 a lz
          END IF
       END IF
    END IF
 END IF
 Devuelve lz

END FUNCTION
```

prueba_sat

FUNCTION **calc_phi_lv_cubic(zcat, opn)**

Asigna **zcat**(1) a **zv**
Asigna **zcat**(2) a **zl**
Asigna **zcat**(3) a **bet**
Asigna **zcat**(4) a **q**
Asigna **zcat**(5) a **sig**
Asigna **zcat**(6) a **epsi**

IF **opn** ==1 OR **opn** == 2 OR **opn** == 3
LOG((zv+sig*bet)/(zv+epsi*bet))*1/(**sig-epsi**) y lo asigna a **I_v**
LOG((zl+sig*bet)/(zl+epsi*bet))*1/(**sig-epsi**) y lo asigna a **I_l**

ELSEIF **opn** ==1
Calcula **bet/zv** y lo asigna a **I_v**
Calcula **bet/zl** y lo asigna a **I_l**
END IF

Calcula **zv**-1-**LOG(zv-bet)-q*I_v** y lo asigna a **lnphi_v**; Calcula **EXP(lnphi_v)** y lo asigna a **phi_v**
Calcula **zl**-1-**LOG(zl-bet)-q*I_l** y lo asigna a **lnphi_l**; Calcula **EXP(lnphi_l)** y lo asigna a **phi_l**
Agrupa a **phi_v** y **phi_l** y asigna el resultado a **tab**
Devuelve **tab**

END FUNCTION

FUNCTION **calc_pres(t, tc, pold, pc, op, opn, op_extra, r_prin, fac, zcat0)**

Establece la carpeta “sub_file” como el directorio actual de trabajo, y asigna el resultado a **dir1**;
Utilizando **dir1**, llama al espacio de trabajo, a **fv2**.

Evalúa **calc_phi_lv_cubic(zcat0** y **opn)**, y asigna el resultado a **tab**
Almacena **tab**(1) en **phi_v**;
Almacena **tab**(2) en **phi_l**

Calcula **pold** * **phi_l/phi_v** y asigna el resultado a **pnew**
Calcula **ABS(pnew – pold)** y asigna el resultado a **difp;** Asigna **pnew** a **pold**

Asigna 0 a **itp**; Asigna 100 a **itmax**

```
   WHILE difp > 0.0001
      Evalúa fv2(t, tc, pnew, pc, op, opn, op_extra, r_prin, fac) y asigna el resultado a
      zcat
      Evalúa calc_phi_lv_cubic(zcat, opn), y asigna el resultado a tab
      Almacena tab(1) en phi_v;
      Almacena tab(2) en phi_l

      Calcula pold * phi_l/phi_v y asigna el resultado a pnew
      Calcula ABS(pnew – pold) y asigna el resultado a difp; Asigna pnew a pold
      Suma 1 a itp y asigna el resultado a itp
      IF itp > itmax
         Sale del ciclo WHILE
      END IF
   END WHILE
   Devuelve pnew

END FUNCTION
```

```
FUNCTION pres_sat_cubic(t, tc, pold, pc, op, opn, op_extra, r_prin, zcat0, fac)

   Establece la carpeta "sub_file" como el directorio actual de trabajo, y asigna el
   resultado a dir1;
   Utilizando dir1, llama al espacio de trabajo a fv2

   IF zcat ~= 0
      Almacena zcat0(1) en zv0;
      Almacena zcat0(2) en zl0
      Calcula (ABS(zv0 – zl0))^2, y asigna el resultado a difz0

      IF difz0 <= 0.0001
         Asigna 3 a zeval
         WHILE zeval ~= 1
            Calcula 0.95 * pold y el resultado lo asigna a pnew; Asigna pnew a pold
            Evalúa la función fv2(t, tc, pnew, pc, op, opn, op_extra, r_prin, fac) y asigna
            el resultado a zcat0

            IF zcat0 ~= 0
               Almacena zcat0(1) en zv0;
               Almacena zcat0(2) y lo asigna a zl0;
               Calcula (ABS(zv0 – zl0))^2 y asigna el resultado a difz0
               IF difz0 > 0.0001
                  Asigna 1 a zeval
               END IF
            END IF
```

```
      END WHILE
      Asigna pnew a pr
    ELSE
      Asigna pold a pr
    END IF
    Evalúa calc_pres(t, tc, pr, pc, op, opn, op_extra, r_prin, fac, zcat0) y lo asigna a
    pnew

  ELSE
    Asigna 3 a zeval
    WHILE zeval ~= 1
      Calcula 0.95 * pold y el resultado lo asigna a pnew; Asigna pnew a pold
      Evalúa fv2(t, tc, pnew, pc, op, opn, op_extra, r_prin, fac) y asigna el resultado
      a zcat0

      IF zcat0 ~= 0
        Almacena zcat0(1) y lo asigna a zv0; Almacena zcat0(2) y lo asigna a zl0;
        Calcula (ABS(zv0 – zl0))^2 y asigna el resultado a difz0

        IF difz0 > 0.0001
          Asignar 1 a zeval
        END IF
      END IF
    END WHILE
    Asigna pnew a pr
    Evalúa calc_pres(t, tc, pr, pc, op, opn, op_extra, r_prin, fac, zcat0), y asigna el
    resultado a pnew
  END IF
  Devuelve pnew

END FUNCTION
```

```
FUNCTION desc_region_cubic(t, tc, p, pc, op, opn, consat, r_prin)

  Establece la carpeta “sub_file” como el directorio actual de trabajo, y asigna el
  resultado a dir1;
  Utilizando dir1, llama al espacio de trabajo a fv2

  Asigna 83.14 a r
  IF opn ~= 5
    IF consat ==1
      IF t > tc
        Evalúa desc_region_cubic_first(t, tc, p, pc, op, opn, r_prin)
      END IF
```

```
      ELSEIF consat ==2
         Evalúa desc_region_cubic_second(t, tc, p, pc, op, opn, r_prin)
      END IF

   ELSE
      Asigna 3 a op_extra; Asigna 83.14 a r

      INPUT factor acéntrico y asigna el resultado a fac
      Evalúa fv2(t, tc, p, pc, op, opn, op_extra, r_prin, fac), y asigna el resultado a zcat
      Calcula zcat * r * t/p, y asigna el resultado a vol_vap
      PRINT zcat y vol_vap
   END IF

END FUNCTION
```

```
FUNCTION desc_region_cubic_first(t, tc, p, pc, op, opn, r_prin)

   Establece la carpeta "sub_file" como el directorio actual de trabajo, y asigna el
   resultado a dir1;
   Utilizando dir1, llama al espacio de trabajo a fv2

   Asigna 83.14 a r
   Asigna 1 a op_extra

   IF opn ==3 OR opn ==4
      INPUT factor acéntrico y asigna el resultado a fac

   ELSEIF opn ==1 OR opn ==2
      Asigna  – 100  a fac
   END IF

   Evalúa fv2(t, tc, p, pc, op, opn, op_extra, r_prin, fac), y asigna el resultado a zcat0

   IF zcat0 ~= 0
      Almacena zcat0(1) y lo asigna a zv;
      Almacena zcat0(2) y lo asigna a zl
      Calcula (ABS(zv – zl))^2 y asigna el resultado a difz

      IF difz > 0.0001
         PRINT "ambos volúmenes de saturación (líquido y vapor) o sólo el de vapor
                saturado? 1. Si  2. No"
         INPUT número entre 1 y 2 y asigna el resultado a op_sat

         IF op_sat ==1
```

```
            Calcula zv * r * t/p y lo asigna a vol_vap
            Calcula zl * r * t/p y lo asigna a vol_liq
            PRINT zv, zl, vol_vap y vol_liq

         ELSEIF op_sat == 2
            Calcula zv * r * t/p y lo asigna a vol_vap
            PRINT zv y vol_vap
         END IF
      ELSE
         PRINT "sustancia esta en la región de liquido comprimido"
      END IF
   ELSE
      PRINT ""sustancia esta en la región de liquido comprimido"
   END IF

END FUNCTION
```

```
FUNCTION desc_region_cubic_second(t, tc, p, pc, op, opn, r_prin)

   Asigna 83.14 a r
   Asigna 1 a op_extra

   Establece la carpeta "sub_file" como el directorio actual de trabajo, y asigna el
   resultado a dir1;
   Utilizando dir1, llama al espacio de trabajo a fv2

   IF op ~= 2
      IF opn ==3 OR opn ==4
         INPUT factor acéntrico y asigna el resultado a fac

      ELSEIF opn ==1 OR opn ==2
         Asigna  –100  a fac
      END IF

   END IF

   IF t > tc
     IF op ~= 2
        Evalúa fv2(t, tc, p, pc, op, opn, op_extra, r_prin, fac), y asigna el resultado a
        zcat0
         Almacena zcat0(1) y lo asigna a zv;
         Almacena zcat0(2) y lo asigna a zl
         Asigna p a pold
```

Evalua **pres_sat_cubic(t**, **tc**, **pold**, **pc**, **op**, **opn**, **op_extra**, **fac**, **r_prin**, **zcat0)** y asigna el resultado a **pr**
Evalua **calc_phi_lv_cubic(zcat0, opn)** y asigna el resultado a **tab**

Almacena **tab**(1) y lo asigna a **phi_v**
Almacena **tab**(2) y lo asigna a **phi_l**
Calcula **p*****phi_v** – **pr*****phi_l** y lo asigna a **dif_fug**
IF **dif_fug** > 0.001
  IF **p** > **pr**
    PRINT "sustancia se encuentra en la región de liquido comprimido"
  ELSEIF **p** > **pr**
    PRINT "sustancia se encuentra en la región de vapor sobrecalentado"
    Calcula **zv** * **r** * **t**/**p** y lo asigna a **vol_vap**.
    PRINT **zv** y **vol_vap**
  ELSE
    Calcula **zv** * **r** * **t**/**p** y lo asigna a **vol_vap**
    PRINT **zv** y **vol_vap**
  END IF
END IF
ELSEIF **op** == 2
  Evalúa **fv2(t**, **tc**, **p**, **pc**, **op**, **opn**, **op_extra**, **r_prin**, **fac),** y asigna el resultado a **zcat0**

  IF **zcat** ~= 0
    Almacena **zcat0**(1) y lo asigna a **zv**;
    Almacena **zcat0**(2) y lo asigna a **zl**
    Asigna **p** a **pold**
    Evalua **pres_sat_cubic(t**, **tc**, **pold**, **pc**, **op**, **opn**, **op_extra**, **fac**, **r_prin**, **zcat0)** y asigna el resultado a **pr**
    Evalua **calc_phi_lv_cubic(zcat0, opn)** y asigna el resultado a **tab**
    Almacena **tab**(1) y lo asigna a **phi_v**
    Almacena **tab**(2) y lo asigna a **phi_l**

    Calcula el producto **p*****phi_v** y el producto **pr*****phi_l**, calcula la diferencia **p*****phi_v**– **pr*****phi_l** y lo asigna a **dif_fug**

    IF **dif_fug** > 0.001
      IF **p** > **pr**
        PRINT "sustancia se encuentra en la región de liquido comprimido"
      ELSEIF **p** > **pr**
        PRINT "sustancia se encuentra en la región de vapor sobrecalentado"
        Calcula **zv** * **r** * **t**/**p** y asigna el resultado a **vol_vap**.
        PRINT **zv** y **vol_vap**
      ELSE

```
                PRINT "ambos volúmenes de saturación (líquido y vapor) o sólo el de
                      vapor saturado?"
                INPUT número entre 1 y 2 y asigna el resultado a op_sat

                IF op_sat ==1
                   Calcula zv * r * t/p y lo asigna a vol_vap
                   Calcula zl * r * t/p y lo asigna a vol_liq
                   PRINT zv, zl, vol_vap y vol_liq

                ELSEIF op_sat ==2
                   Calcula zv * r * t/p y lo asigna a vol_vap
                   PRINT zv y vol_vap
                END IF
             END IF
          END IF

       ELSEIF zcat ==0
          Asigna p a pold
          PRINT "la sustancia se encuentra en la region de liquido comprimido"
       END IF
    ELSE
       Asigna 2 a op_extra
       Evalúa fv2(t, tc, p, pc, op, opn, op_extra, r_prin, fac), y asigna el resultado a
       zcat0
       Calcula zv * r * t/p y asigna el resultado a vol_vap
       PRINT zv y vol_vap
    END IF
 END IF

END FUNCTION
```

cal_fz

```
Establece la carpeta "sub_file" como el directorio actual de trabajo, y asigna el
resultado a r2;
Utilizando r2, llama al espacio de trabajo a desc_region

PRINT "ecuación cubica de estado a utilizar"
INPUT número entre 1, 2, 3, 4 o 5 y asigna el resultado a opn

WHILE %T
   IF opn ~= [1, 2, 3, 4, 5]
```

```
      PRINT "ecuación cubica de estado a utilizar"
      INPUT número entre 1, 2, 3, 4 o 5 y asigna el resultado a opn
   ELSE
      Sale del ciclo
   END IF
END WHILE

PRINT "método de resolución para la ecuación cúbica del volumen"

INPUT número entre 1, 2 y 3 y asigna el resultado a op
WHILE %T
   IF op ~= [1, 2, 3]
      PRINT "ecuación cubica de estado a utilizar"
      INPUT número entre 1, 2 o 3 y asigna el resultado a op
   ELSE
      Sale del ciclo
   END IF
END WHILE

INPUT temperatura del fluido y asigna el resultado a t
WHILE %T
   IF t > 0
      Sale del ciclo
   ELSE
      INPUT temperatura del fluido y asigna el resultado a t
   END IF
END WHILE

INPUT presión del fluido y asigna el resultado a t
WHILE %T
   IF p > 0
      Sale del ciclo
   ELSE
      INPUT presión del fluido y asigna el resultado a p
   END IF
END WHILE

PRINT "temperatura y presión corresponden a condiciones de saturación 1. Si, 2. No?"

INPUT número entre 1 y 2 y se asigna el resultado a consat
WHILE %T
   IF consat ~= [1,2]
      PRINT "temperatura y presión corresponden a condiciones de saturación 1. Si  2.
            No?"
```

```
      INPUT número entre 1 y 2 y se asigna el resultado a consat
   ELSE
      Sale del ciclo
   END IF
END WHILE

INPUT temperatura crítica y se asigna el resultado a tc
WHILE %T
   IF tc > 0
      Sale del ciclo
   ELSE
      INPUT temperatura crítica y se asigna el resultado a tc
   END IF
END WHILE
INPUT presión crítica y se asigna el resultado a pc
WHILE %T
   IF pc > 0
      Sale del ciclo
   ELSE
      INPUT presión crítica y se asigna el resultado a pc
   END IF
END WHILE

Evalua desc_region_cubic(t, tc, p, pc, op, opn, consat, r_prin)
```

C.3 Asignatura: Termodinamica Quimica II

Cálculo de la entalpía, entropía y energía de Gibbs residual de un gas puro, mediante la ecuación cúbica genérica ((Problema 3°).

fv2

FUNCTION **ffz(z, fact1, fact2, fact3)**

z^3+**fact1*****z**^2+**fact2*****z**+**fact3** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **dffz(z, fact1, fact2)**

3***z**^2+2***fact1*****z**+**fact2** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **fv2(t, tc, p, pc, opn, r_prin, fac)**

Establece la carpeta "Metodos numericos" como el directorio actual de trabajo, y asigna el resultado a **r**;
Utilizando **r**, llama al espacio de trabajo, a **rootNewton_cubic2** y **rootRegulaFA2**.

Calcula **t**/**tc** y guarda el resultado en **tr**
Calcula **p**/**pc** y guarda el resultado en **pr**

IF **fac** ==–100
IF **opn** ==1
Asigna 1 a **alf;** Asigna 1/8 a **ome;** Asigna 27/64 a **psi;** Asigna 0 a **sig**
Asigna 0 a **epsi**

ELSEIF **opn** ==2
Calcula **tr**^-0.5 y asigna el resultado a **alf**
Asigna 0.08664 a **ome;** Asigna 0.42748 a **psi;** Asigna 1 a **sig;** Asigna 0 a **epsi**
END IF
ELSEIF **fac** ~= –100

```
    IF opn ==3
      Calcula 0.48 + 1.574 * fac - 0.176*fac^2 (ecuación de Soave-Redlich-Kwong) y
      lo asigna a facf
      Calcula (1+facf*(1 - tr^0.5))^2 y lo asigna a alf
      Asigna 0.08664 a ome; Asigna 0.42748 a psi; Asigna 1 a sig; Asigna 0 a epsi

    ELSEIF opn ==4
      Calcula 0.37464 + 1.5422 * fac - 0.26992 * fac^2 (ecuación de Peng-Robinson) y
      lo asigna a facf
      Calcula (1+facf * (1 - tr^0.5))^2 y lo asigna a alf
      Asigna 0.0778 a ome; Asigna 0.45724 a psi;
      Calcula 1+2^0.5, y asigna el resultado a sig
      Calcula 1-2^0.5, y asigna el resultado a epsi
    END IF
  END IF

  Asigna 0.00001 a tol; Asigna 100 a imax
  Calcula ome * pr/tr y el resultado lo asigna bet
  Calcula psi * alf/(ome * tr), y el resultado lo asigna a q
  Calcula epsi * bet + sig * bet – 1 - bet y asigna el resultado a fact1
  Calcula epsi*sig*bet^2-epsi*bet-sig*bet-epsi*bet^2-sig*bet^2+q*bet y asigna el
  resultado a fact2
  Calcula -epsi*sig*bet^2-epsi*sig*bet^3-q*bet^2 y asigna el resultado a fact3
  Agrupa a fact1, fact2 y fact3 y asigna el resultado a cof

  Asigna 1 a z0v
  Evalua rootNewton_cubic2(z0v, ffz, dffz, cof, dcof, tol, imax, store = False) y
  asigna el resultado a vm
  Almacena vm(2) en zv
  Agrupa a zv, zl, bet, q, sig y epsi y el resultado lo asigna a lz

  IF t > tc
    Asigna bet a z0l
     Evalua rootNewton_cubic2(z0l, ffz, dffz, cof, dcof, tol, imax,store = False) y
     asigna el resultado a vm
     Almacena vm(2) en zl
     Agrupa a zv, zl, bet, q, sig y epsi y el resultado lo asigna a lz
  END IF
  Devuelve lz

END FUNCTION
```

prueba_sat

FUNCTION **calc_phi_lv_cubic(zcat, opn)**

Asigna **zcat**(1) a **zv;** Asigna **zcat**(2) a **zl;** Asigna **zcat**(3) a **bet;** Asigna **zcat**(4) a **q**
Asigna **zcat**(5) a **sig;** Asigna **zcat**(6) a **epsi**

IF **opn** ==2 OR **opn** ==3 OR **opn** ==4
LOG((zv+sig*bet)/(zv+epsi*bet))*1/(**sig-epsi**) y lo asigna a **I_v**
LOG((zl+sig*bet)/(zl+epsi*bet))*1/(**sig-epsi**) y lo asigna a **I_l**

ELSEIF **opn** ==1
Calcula **bet/zv** y lo asigna a **I_v;** Calcula **bet/zl** y lo asigna a **I_l**
END

Calcula **zv**-1-**LOG(zv-bet)-q*I_v** y lo asigna a **lnphi_v**; Calcula **EXP(lnphi_v)** y lo asigna a **phi_v**
Calcula **zl**-1-**LOG(zl-bet)-q*I_l** y lo asigna a **lnphi_l**; Calcula **EXP(lnphi_l)** y lo asigna a **phi_l**
Agrupa a **phi_v** y **phi_l** y asigna el resultado a **tab**
Devuelve **tab**

END FUNCTION

FUNCTION **calc_pres(t, tc, pold, pc, opn, fac, r_prin, zcat0)**

Establece la carpeta "sat_condicion" como el directorio actual de trabajo, y asigna el resultado a **dir1**;
Utilizando **dir1**, llama al espacio de trabajo, a **fv2**.
Asigna 0 a **itp**; Asigna 100 a **itmax**
Evalúa **calc_phi_lv_cubic(zcat0**, **opn)**, y asigna el resultado a **tab**
Almacena **tab**(1) y lo asigna a **phi_v**; Almacena **tab**(2) y lo asigna a **phi_l**

Calcula **pold** * **phi_l/phi_v** y asigna el resultado a **pnew**
Calcula **ABS(pnew – pold)** y asigna el resultado a **difp;** Asigna **pnew** a **pold**
Asigna 0 a **itp**; Asigna 100 a **itmax**

WHILE **difp** > 0.0001
Evalúa **fv2(t**, **tc**, **pnew**, **pc**, **opn**, **r_prin** y **fac)** y asigna el resultado a **zcat**
Evalúa **calc_phi_lv_cubic(zcat**, **opn)**, y asigna el resultado a **tab**
Almacena **tab**(1) y lo asigna a **phi_v**; Almacena **tab**(2) y lo asigna a **phi_l**
Calcula **pold** * **phi_l/phi_v** y asigna el resultado a **pnew**
Calcula **ABS(pnew – pold)** y asigna el resultado a **difp;** Asigna **pnew** a **pold**

```
    Suma 1 a itp y asigna el resultado a itp
    IF itp > itmax
      Sale del ciclo WHILE
    END IF
  END WHILE
  Devuelve pnew

END FUNCTION
```

```
FUNCTION pres_sat_cubic(t, tc, pold, pc, opn,  fac, r_prin, zcat0)

  Establece la carpeta "sat_condicion" como el directorio actual de trabajo, y asigna el
  resultado a dir1;
  Utilizando dir1, llama al espacio de trabajo a fv2

  IF zcat ~= 0
    Almacena zcat0(1) y lo asigna a zv0; Almacena zcat0(2) y lo asigna a zl0
    Calcula (ABS(zv0 – zl0))^2 y asigna el resultado a difz0

    IF difz0 <= 0.0001
      Asigna 3 a zeval
      WHILE zeval ~= 1
        Calcula 0.95 * pold y el resultado lo asigna a pnew; Asigna pnew a pold
        Evalúa fv2(t, tc, pnew, pc, opn, r_prin, fac) y asigna el resultado a zcat0

        IF zcat0 ~= 0
          Almacena zcat0(1) y lo asigna a zv0;
          Almacena zcat0(2) y lo asigna a zl0;
          Calcula (ABS(zv0 – zl0))^2 y asigna el resultado a difz0

          IF difz0 > 0.0001
            Asigna 1 a zeval
          END IF
        END IF
      END WHILE
      Asigna pnew a pr

    ELSE
      Asigna pold a pr
    END IF
    Evalúa calc_pres(t, tc, pr, pc, opn, r_prin, fac, zcat0) y lo asigna a pnew
  ELSE
    Asigna 3 a zeval
```

```
    WHILE zeval ~= 1
      Calcula 0.95 * pold y el resultado lo asigna a pnew; Asigna pnew a pold
      Evalúa fv2(t, tc, pnew, pc, opn, r_prin, fac) y asigna el resultado a zcat0

      IF zcat0 ~= 0
        Almacena zcat0(1) y lo asigna a zv0;
        Almacena zcat0(2) y lo asigna a zl0;
        Calcula (ABS(zv0 – zl0))^2 y asigna el resultado a difz0

        IF difz0 > 0.0001
          Asignar 1 a zeval
        END IF
      END IF
    END WHILE

    Asigna pnew a pr
    Evalúa calc_pres(t, tc, pr, pc, opn, r_prin, fac, zcat0), y asigna el resultado a
    pnew
  END IF
  Devuelve pnew

END FUNCTION
```

```
FUNCTION desc_region_cubic(t, tc, p, pc, op, opn, consat, r_prin)

  Establece la carpeta "sub_file" como el directorio actual de trabajo, y asigna el
  resultado a dir1;
  Utilizando dir1, llama al espacio de trabajo a fv2

  IF opn ==1 OR opn ==4
    INPUT factor acéntrico y asigna el resultado a fac

  ELSEIF opn ==1 OR opn ==2
    Asigna –100  a fac
  END IF

  Evalua fv2(t, tc, p, pc, opn, r_prin, fac) y asigna el resultado a zcat0

 IF t > tc
    Asigna p a pold
    Evalua pres_sat_cubic(t, tc, pold, pc, opn, fac, r_prin, zcat0) y asigna el resultado
    a pr
    Evalua calc_phi_lv_cubic(zcat0, opn) y asigna el resultado a tab
    Almacena tab(1) y lo asigna a phi_v
```

```
    Calcula p*phi_v – pr*phi_l y lo asigna a dif_fug

    IF dif_fug > 0.001
      IF p > pr
        PRINT "sustancia se encuentra en la región de liquido comprimido"
        Asigna 0 a lz
      ELSEIF p > pr
        PRINT "sustancia se encuentra en la región de vapor sobrecalentado"
        Agrupa a zcat0 y fac, y asigna el resultado a lz
      ELSE
        Agrupa a zcat0 y fac, y asigna el resultado a lz
      END IF
    END IF

  ELSE
    Agrupa a zcat0 y fac, y asigna el resultado a lz
  END IF
  Devuelve lz

END FUNCTION
```

Hr_Sr_calc

```
FUNCTION l_alf_evalf(lf_alf, tr, param)

  Evalúa lf_alf(tr, param) y asigna el resultado a y
  Devuelve y

END FUNCTION
```

```
FUNCTION fp(tr, fun_alf, param)

  Asigna 0.001 a h
  Agrupa tr-2*h, tr-h, tr+h, tr+2*h y asigna el resultado a xv.
  Crea un vector vacío y asigna el resultado a fxv
  Asigna 0 a i

  WHILE i <= 3
    Evalúa l_alf_evalf(fun_alf, trv(i), param) y asigna el resultado a f_xv;
    Agrega f_xv como elemento de fxv
    Suma 1 a i y asigna el resultado a i
  END WHILE
```

```
   Calcula (-fxv(4)+8*fxv(3)-8*fxv(2)+fxv(1))/(12*h) y asigna el resultado a fun;
   Devuelve fun

END FUNCTION
```

```
FUNCTION ln_alf(tr, opn, fac)

   IF fac ==–100
      IF opn ==1
         Asigna 1 a alf
      ELSEIF opn ==2
         Eleva tr^–0.5 y asigna el resultado a alf
      END IF

   ELSE
      IF opn ==3
         Calcula 0.48+1.574*fac-0.176*fac^2 (ecuación de Soave-Redlich-Kwong) y
         asigna el resultado a facf
         Calcula (1+facf*(1-tr^0.5))^2 y asigna el alf

      ELSEIF opn ==4
         Calcula 0.37464 + 1.5422 * fac - 0.26992 * fac^2 (ecuación de Peng-Robinson) y
         lo asigna a facf
         Calcula (1+facf * (1 - tr^0.5))^2 y lo asigna a alf
      END IF
   END IF
   Calcula LOG(alf) y lo asigna a lf
   Devuelve lf

END FUNCTION
```

```
FUNCTION calcl(opn, zv, bet, sig, epsi)

   IF opn ==2 OR opn ==3 OR opn ==4
      LOG((zv+sig*bet)/(zv+epsi*bet))*1/(sig-epsi) y lo asigna a fI

   ELSEIF opn ==1
      Calcula bet/zv y lo asigna a I_v y asigna el resultado a fI
   END IF
   Devuelve fI

END FUNCTION
```

FUNCTION **calcHr(zv, dlalf, t, tc, q, l, Rcons)**

Calcula **t**/**tc** y asigna el resultado a **tr**
Calcula (**zv**-1+(**tr*****dlalf**-1)***q*****I**)***Rcons*****t** y asigna el resultado a **Hr**

Devuelve **Hr**
END FUNCTION

FUNCTION **calcSr(zv, dlalf, t, tc, bet, q, l, Rcons)**

Calcula **t**/**tc** y lo asigna a **tr**
Calcula (**LOG(zv-bet)+tr*****dlalf*****q*****I**)***Rcons** y asigna el resultado a **Sr**
Devuelve **Sr**

END FUNCTION

FUNCTION **calcGr(Hr, Sr, t)**

Calcula **Hr-t*****Sr** y asigna el resultado a **Gr**
Devuelve **Gr**

END FUNCTION

FUNCTION **Hr_Sr_imp(lz, t, tc, p, pc, opn)**

IF **lz** ~= 0
Asigna **lz**(2) a **fac**
Agrupa a **opn** y **fac** y asigna el resultado a **param**
Asigna **lz**(1) a **zcat**

IF **LENGTH(lz)** ==5
Asigna **zcat**(1) a **zv**; Asigna **zcat**(2) a **bet**; Asigna **zcat**(3) a **q**;
Asigna **zcat**(4) a **sig**; Asigna **zcat**(5) a **epsi**;
ELSEIF **LENGTH(lz)** ==6
Asigna **zcat**(1) a **zv**; Asigna **zcat**(3) a **bet**; Asigna **zcat**(4) a **q**;
Asigna **zcat**(5) a **sig**; Asigna **zcat**(6) a **epsi**;
END IF

Calcula **t**/**tc** y asigna el resultado a **tr**
Evalua **calcl(opn**, **zv**, **bet**, **sig**, **epsi)**, y asigna el resultado a **I**
Evalua **fp(tr**, **ln_alf** y **param)** y asigna el resultado a **dlalf**

Asigna 8.314 a **Rcons**

```
      Evalua calcHr(zv, dlalf, t, tc, q, I y Rcons)
      Evalua calcSr(zv, dlalf, t, tc, bet, q, I y Rcons)
      Agrupa a Hr y Sr, y asigna el resultado a res

   ELSE
      Asigna 0 a res
   END IF
   Devuelve res

END FUNCTION
```

Hr_Sr_prin

```
Establece la ruta absoluta del directorio de desc_region_cubic, y asigna el resultado a
r_prin;
Utilizando r_prin, establece la ruta del directorio de prueba_sat, y asigna el resultado a
r2
Utilizando r2, llama al espacio de trabajo a desc_region_cubic
Utilizando r_prin, establece la ruta del directorio de Hr_Sr_calc, y asigna el resultado a
r3
Utilizando r3, llama al espacio de trabajo a Hr_Sr_imp
PRINT "ecuación cubica de estado a utilizar"

INPUT número entre 1, 2 3 o 4 y asigna el resultado a opn
WHILE %T
   IF opn ~= [1, 2, 3, 4]
      PRINT "ecuación cubica de estado a utilizar"
      INPUT número entre 1, 2, 3 o 4 y asigna el resultado a opn
   ELSE
      Sale del ciclo
   END IF
END WHILE

INPUT temperatura y asigna el resultado a t
WHILE %T
   IF t > 0
      Sale del ciclo
   ELSE
      INPUT temperatura y asigna el resultado a t
   END IF
END WHILE
```

```
INPUT presión y asigna el resultado a p
WHILE %T
   IF p > 0
      Sale del ciclo
   ELSE
      INPUT presión y asigna el resultado a p
   END IF
END WHILE

INPUT temperatura crítica y se asigna el resultado a tc
WHILE %T
   IF tc > 0
      Sale del ciclo
   ELSE
      INPUT temperatura crítica y se asigna el resultado a tc
   END IF
END WHILE

INPUT presión crítica y se asigna el resultado a pc
WHILE %T
   IF pc > 0
      Sale del ciclo
   ELSE
      INPUT presión crítica y se asigna el resultado a pc
   END IF
END WHILE

Evalua desc_region_cubic(t, tc, p, pc, opn, r_prin) y asigna el resultado a lz
Evalua Hr_Sr_imp(lz, t, tc, p, pc, opn) y asigna el resultado a res

IF res ~= 0
   Almacena res(1) a Hr; Almacena res(2) a Sr; Almacena res(3) en Gr
   PRINT "propiedades residuales Hr, Sr y Gr: "
   PRINT Hr; PRINT Sr; PRINT Gr

ELSE
   PRINT "Introduzca nuevas condiciones de termperatura y presión"
END IF
```

Cálculo del punto de burbuja T utilizando la ecuación de Raoult modificada y la formulación gamma-phi para mezclas binarias, ternarias y cuaternarias (Problemas 4° y 5°).

cpre_ing

```
FUNCTION tab_c(bk, eco)

   IF bk == 1
      IF eco == 1
         Agrupa los caracteres “A”, “B”, “C” y “D” y asigna el resultado a car
      ELSEIF eco == 2
         Agrupa los caracteres “A”, “B”, “C” y “D” y asigna el resultado a car
      ELSEIF eco == 3
         Agrupa los caracteres “A”, “B” y “C” y asigna el resultado a car
      END IF
   ELSEIF bk == 2
      IF eco == 1
         Agrupa los caracteres “A”, “B” y “C” y asigna el resultado a car
      ELSEIF eco == 2
         Agrupa los caracteres “A”, “B”, “C”, “Tc(K)”, “t0”, “n”, “E” y “F” y asigna el
         resultado a car
      ELSEIF eco == 3
         Agrupa los caracteres “Tc(K)”, “A”, “B”, “C”, “D” y “Pc(bar)” y asigna el
         resultado a car
      END IF
   ELSEIF bk ==3
      Agrupa los caracteres "C1","C2","C3", “C4” y "C5" y asigna el resultado a car
   ELSEIF bk ==4
      Agrupa los caracteres “A”, “B” y “C” y asigna el resultado a car
   END IF
   Devuelve car

END FUNCTION
```

```
FUNCTION ingreso_constantes(bk,eco,st1)

   Evalúa tab_c(bk, eco) y asigna el resultado a car
   Calcula LENGTH(car) y asigna el resultado a n
   Asigna 2 a ind

   WHILE ind == 2
```

```
    FOR 1 < i < n
      INPUT valor numérico correspondiente a car(i) y asigna el resultado a cpre(i)
    END FOR

    PRINT "Constantes para el cálculo de las capacidades caloríficas correctas? 1. Si  2.
          No"
    INPUT número entre 1 y 2 y asigna el resultado a ind
  END WHILE
  Devuelve cpre

END FUNCTION
```

```
FUNCTION cpre_ing(pathDir, p, t, st1, bk)

  IF bk ==1
    PRINT "tipo de ecuación de presión de vapor?"
    INPUT número entre 1, 2 y 3 y asigna el resultado en eco
    WHILE %T
      IF eco ~=[1, 2, 3]
        PRINT "tipo de ecuación de la capacidad calorífica?"
        INPUT número entre 1, 2 y 3 y asigna el resultado en eco
      ELSE
        Sale del ciclo
      END IF

  ELSEIF bk ==2
    PRINT "tipo de ecuación de presión de vapor?"
    INPUT número entre 1, 2 y 3 y asigna el resultado en eco
    WHILE %T
      IF eco ~=[1, 2, 3]
        PRINT "tipo de ecuación de la capacidad calorífica?"
        INPUT número entre 1, 2 y 3 y asigna el resultado en eco
      ELSE
        Sale del ciclo
      END IF
  END IF

  IF bk == 1
    IF eco == 1
      Asigna 2 a ind
      WHILE ind == 2
        INPUT temperatura crítica y asigna el resultado a tc
        WHILE %T
          IF tc < 0
```

```
            INPUT temperatura crítica y asigna el resultado a tc
          ELSE
            Sale del ciclo
          END IF
        END WHILE

        INPUT presión crítica y asigna el resultado a pc
        WHILE %T
          IF pc < 0
            INPUT presión crítica y asigna el resultado a pc
          ELSE
            Sale del ciclo
          END IF
        END WHILE
        PRINT "temperatura crítica y presión crítica correctas? 1. Si  2. No"
        INPUT número entre 1 y 2 y asigna el resultado a ind
      END WHILE
      Agrupa cpre, tc y pc y asigna el resultado a cpre

    ELSEIF eco ==2
      Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a cpre
    ELSEIF eco ==3
      Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a cpre
    END IF

  ELSEIF bk ==2
    IF eco == 1
      Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a cpre
    ELSEIF eco ==2
      Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a cpre
    ELSEIF eco ==3
      Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a cpre
    END IF
  ELSEIF bk ==3
    Asigna 0 a eco
    Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a cpre
  ELSEIF bk ==4
    Asigna 0 a eco
    Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a cpre
  END IF

  IF bk == 1 OR bk == 2
    Agrupa a bk y eco y asigna el resultado a bec
  ELSEIF bk == 3 OR bk == 4
```

Coloca a **bk** en una lista y asigna el resultado a **bec**
END IF
Agrupa a **p**, **t**, **bec**, **cpre** y **pathDir** y asigna el resultado a **catg1**

END FUNCTION

fpres_vap

FUNCTION **ant_van_ness(t, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**
Calcula **EXP(vpa-vpb**/(**t**-273.15+**vpc**)**)** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **ant_prausnitz_p_4ed(t, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**
Calcula **EXP(vpa-vpb**/(**t**+**vpc**)**)** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **ant_prausnitz_p_5ed(t, cpre)**

Guarda **cpre(1)** en **vpa**; Guarda **cpre(2)** en **vpb;** Guarda **cpre(3)** en **vpc**
Calcula 10^(**vpa-vpb**/(**t**-273.15+**vpc**)) y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **wag_p(xp, pc, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc;**
Guarda **cpre**(4) en **vpd**
Calcula **EXP(**((1-**xp**)^-1)*(**vpa***xp+**vpb***xp^1.5+**vpc***xp^3+**vpd***xp^6)**)***pc** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **antmod_p(t, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**;
Guarda **cpre**(4) en **vpe**
Guarda **cpre**(5) en **vpn**; Guarda **cpre**(6) en **vpf**; Guarda **cpre**(7) en **vpt0**;
Guarda **cpre**(8) en **vptc**

Calcula (**t**-**vpt0**-273.15)/**vptc** y asigna el resultado a **x**
Calcula **vpa**-**vpb**/(**t**+**vpc**-273.15)+0.43429***x*^**vpn**+**vpe***x*^8+**vpf***x*^12 y asigna el resultado a **log_10_pv**
Calcula 10^**log_10_pv** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **wagmod_p(t, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**
Guarda **cpre**(4) en **vpd**; Guarda **cpre**(5) en **tc**; Guarda **cpre**(6) en **pc**
Calcula **pc***EXP((**tc**/**t**).*(**vpa***(1-**t**/**tc**)+**vpb***(1-**t**/**tc**).^1.5+**vpc***(1-**t**/**tc**).^2.5+**vpd***(1-**t**/**tc**).^5)) y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **riedel_p(t, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**;
Guarda **cpre**(4) en **vpd;** Guarda **cpre**(5) en **vpe**
Calcula **EXP(vpa**+**vpb**/**t**+**vpc***LOG(**t**)+**vpd***(**t**)^**vpe)** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **ant_prausnitz_t_4ed(p, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**
Calcula **vpb**/(**vpa**-**LOG(p_bar)**)-**vpc** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **ant_prausnitz_t_5ed(p, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**
Calcula **vpb**/(**vpa**-**LOG10(p)**)-**vpc**+273.15 y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **ant_van_ness_t(p, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**
Calcula **vpb**/(**vpa**-**LOG(p)**)-**vpc** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **wag_t(xp, p, pc, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**;
Guarda **cpre**(4) en **vpd**
Calcula **LOG(p/pc)**-((1-**xp**).^-1)*(**vpa***xp+**vpb***xp.^1.5+**vpc***xp.^3+**vpd***xp.^6) y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **antmod_t(t, p, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**
Guarda **cpre**(4) en **vpe**; Guarda **cpre**(5) en **vpn**; Guarda **cpre**(6) en **vpf**
Guarda **cpre**(7) en **vpt0**; Guarda **cpre**(8) en **vptc**

Calcula **LOG10(p)**-**vpa**+**vpb** ./(**t**+**vpc**-273.15)-0.43429*((**t**-**vpt0**-273.15)/**vptc**)^**vpn**-**vpe***((**t**-**vpt0**-273.15)/**vptc**)^8-**vpf***((**t**-**vpt0**-273.15)/**vptc**)^12 y asigna el resultado a

fun
Devuelve **fun**

END FUNCTION

FUNCTION **wag_mod_t(t, p, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**
Guarda **cpre**(4) en **vpd**; Guarda **cpre**(5) en **tc**; Guarda **cpre**(6) en **pc**

Calcula -**LOG(p**/**pc)**+(**tc**/**t**).*(**vpa***(1-**t**/**tc**)+**vpb***(1-**t**/**tc**).^1.5+**vpc*** (1-**t**/**tc**).^2.5+**vpd***(1-**t**/**tc**).^5) y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **riedel_t(t, p, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**;
Guarda **cpre**(4) en **vpd;** Guarda **cpre**(5) en **vpe**
Calcula **LOG(p)-vpa-vpb**/**t-vpc***LOG(t)-**vpd***(**t**)^**vpe** y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **fkt_p(p, t, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**;
Guarda **cpre**(4) en **vpd**
Calcula **LOG(p)-vpa**+**vpb**/**t-vpc***LOG(t)-**vpd***p /(**t**^2) y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **fkt_t(t, p, cpre)**

Guarda **cpre**(1) en **vpa**; Guarda **cpre**(2) en **vpb**; Guarda **cpre**(3) en **vpc**;
Guarda **cpre**(4) en **vpd**
Calcula **LOG(p)-vpa**+**vpb**/**t-vpc***LOG(t)-**vpd***p /(**t**^2) y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **t_pv(p, catg1)**

Guarda **catg1**(3) en **bec**; Guarda **catg1**(4) a **cpre**; Guarda **catg1**(5) a **pathDir**;
Guarda **catg1**(2) a **t0**;

IF **LENGTH(bec)** ==2
Gurada **bec**(1) a **bk**; Guarda **bec**(2) a **eco**
ELSEIF **LENGTH(bec)** ==1
Gurada **bec**(1) a **bk**;
END IF

Utilizando **pathDir**, establece la ruta del directorio de rootSecantePlus2Ap2 y guarda el resultado en **dir1_1**
Utilizando **dir1_1**, llama al espacio de trabajo a rootSecantePlus2A
Asigna 0.00001 a **tol**; Asigna 100 a **imax**

IF **bk** ==1
IF **eco** ==1
Guarda **cpre**(1) en **cpre2**; Guarda el **cpre**(2) en **tc**; Guarda **cpre**(3) en **pc**
Calcula **p**/100 y asigna el resultado a **p_bar**; Asigna **pc** a **pc_bar**
Agrupa a **p_bar**, **pc_bar** y **cpre2**, y asigna el resultado a **l**
Evalua **rootSecantePlus2A(x0**, **wag_t**, **l, tol**, **imax**, **store** = **%T)**, y asigna el resultado a **vm**
Calcula las dimensiones de **vm** y asigna el resultado a **s**
Guarda **s**(1) en **r**
Almacena **vm**(r,2) y asigna el resultado a **x**
Calcula (1-**x**)***tc** y asigna el resultado a **t**

ELSEIF **eco** ==2
Calcula **p**/100 y asigna el resultado a **p_bar**;
Agrupa a **p_bar cpre**, y asigna el resultado a **l**
Evalúa **rootSecantePlus2A(t0, fkt_t, l, tol**, **imax**, **store** = **%T)**, y asigna el resultado a **vm**
Calcula las dimensiones de **vm** y asigna el resultado a **s**
Guarda **s**(1) en **r**
Almacena **vm**(r,2) en **t**

ELSEIF **eco** ==3
Calcula **p**/100 y asigna el resultado a **p_bar**;
Evalúa **ant_prausnitz_t_4_ed(p_bar,cpre)**, y asigna el resultado a **t**
END IF

ELSEIF **bk** ==2

```
      IF eco ==1
         Calcula p/100 y asigna el resultado a p_bar;
         Evalúa ant_prausnitz_t_5_ed(p_bar,cpre), y asigna el resultado a t

      ELSEIF eco ==2
         Calcula p/100 y asigna el resultado a p_bar;
         Agrupa a p_bar y cpre, y asigna el resultado a l
         Evalúa rootSecantePlus2A(t0, ant_mod_t, l, tol, imax, store = %T), y asigna el
         resultado a vm
         Calcula las dimensiones de vm y asigna el resultado a s
         Guarda s(1) en r
         Almacena vm(r,2) en t
      END IF

   ELSEIF bk ==3
      Calcula p*1000 y asigna el resultado a p_pa;
      Agrupa a p_pa y cpre, y asigna el resultado a l
      Evalúa rootSecantePlus2A(t0, riedel_t, l, tol, imax, store = %T), y asigna el
      resultado a vm
      Calcula las dimensiones de vm y asigna el resultado a s
      Guarda s(1) en r
      Almacena vm(r,2) en t

   ELSEIF bk ==4
      Evalua ant_van_ness_t(p, cpre) y asigna el resultado a tC
      Suma 273.15 a tC y asigna el resultado a t
   END IF
   Devuelve t

END FUNCTION
```

```
FUNCTION Psat(t, catg1)

   Asigna catg1(3) a bec;  Asigna catg1(4) a cpre;
   IF LENGTH(bec) ==2
      Guarda bec(1) a bk; Guarda bec(2) a eco
   ELSEIF LENGTH(bec)==1
      Guarda bec(1) a bk;
   END IF

   IF bk ==1
      IF eco ==1
         Guarda cpre(1) a cpre2; Guarda cpre(2) a tc; ; Guarda cpre(3) a pc
         Asigna pc a pc_bar
```

```
        Calcula 1-t/tc y asigna el resultado a x0
        Evalúa wag_p(x0,pc_bar,cpre2) y asigna el resultado a p_bar
        Calcula p_bar*100 y asigna el resultado a pv
      ELSEIF eco ==2
        Guarda catg1(1) en p0; Guarda catg1(5) en pathDir;
        Calcula p0/100 y asigna el resultado a p0_bar
        Utilizando pathDir, establece la ruta del directorio de rootSecantePlus2Ap2 y
        guarda el resultado en dir1_1
        Asigna 0.00001 a tol
        Asigna 100 a imax
        Agrupa t y cpre y asigna el resultado a l
        Evalúa rootSecantePlus2A(p0_bar, fkt_p, l, tol, imax, store = %T) y asigna el
        resultado a vm
        Calcula las dimensiones de vm y asigna el resultado a s
        Guarda s(1) en r
        Guarda vm(r,2) en pv_bar
        Calcula pv_bar*100 y asigna el resultado a pv
      ELSEIF eco ==3
        Evalua ant_prausnitz_p_4ed(t, cpre), y asigna el resultado a p_bar
        Calcula p_bar*100 y asigna el resultado a pv
      END IF

    ELSEIF bk ==2
      IF eco ==1
        Evalua ant_prausnitz_p_5ed(t, cpre), y asigna el resultado a p_bar
        Calcula p_bar*100 y asigna el resultado a pv
      ELSEIF eco ==2
        Evalua antmod_p(t, cpre), y asigna el resultado a p_bar
        Calcula p_bar*100 y asigna el resultado a pv
      ELSEIF eco ==3
        Evalua wagmod_p(t, cpre), y asigna el resultado a p_bar
        Calcula p_bar*100 y asigna el resultado a pv
      END IF

    ELSEIF bk ==3
      Evalua riedel_p(t, cpre), y asigna el resultado a p_pa
      Calcula p_pa*100 y asigna el resultado a pv

    ELSEIF bk ==4
      Evalua ant_van_ness_p(t, cpre), y asigna el resultado a pv
    END IF
    Devuelve pv

END FUNCTION
```

FUNCTION **te2(tpr,x,pt,cof,phicf,cat_mayor,n,pathDir)**

Utilizando **pathDir**, establece la ruta del directorio de fpres_vap y guarda el resultado en **dir1**
Utilizando **dir1**, llama al espacio de trabajo a **Psat**, desde fpres_vap
Realiza **ZEROS(n)** y lo asigna a **sum1**;
Realiza **ZEROS(n)** y lo asigna a **presat**

FOR 1 < i < n
Evalúa **Psat(tpr,cat_mayor(i))** y asigna el resultado a **presat(i)**
Calcula **x(i)*cof(i)*presat(i)/phicf(i)** y asigna el resultado a **sum1(i)**
END FOR
Calcula **SUM(sum1)** y asigna el resultado a **sum2**
Calcula **–pt** + **sum2** y asigna el resultado a **funct**

END FUNCTION

FUNCTION **yBurbt(x,p,cat_mayor,opa,prop,op_gid,pathDir,n,imp,param)**

Utilizando a **pathDir**, establece la ruta del directorio de coac y guarda el resultado en **dir1_1**
Utilizando **dir1_1**, llama al espacio de trabajo a **coac**, desde coac
Utilizando a **pathDir**, establece la ruta del directorio de phiCM y guarda el resultado en **dir2_1**
Utilizando **dir2_1**, llama al espacio de trabajo a **phiCM**, desde phiCM
Utilizando a **pathDir**, establece la ruta del directorio de rootSecantePlu2Ap2 y guarda el resultado en **dir3**
Utilizando **dir3**, llama al espacio de trabajo a **rootSecantePlus2A**, desde rootSecantePlus2Ap2
Utilizando a **pathDir**, establece la ruta del directorio de fpres_vap y guarda el resultado en **dir4_1**
Utilizando **dir4_1**, llama al espacio de trabajo a **t_pv**, desde fpres_vap
Utilizando a **pathDir**, establece la ruta del directorio de cpre_ing y guarda el resultado en **dir4_2**
Utilizando **dir4_2**, llama al espacio de trabajo a **cre_ing**, desde cpre_ing
Utilizando a **pathDir**, establece la ruta del directorio de imp_op_tab y guarda el resultado en **dir5_1**
Utilizando **dir5_1,** llama al espacio de trabajo a **imp_op_tab**, desde imp_op_tab

FOR 1 < **i** < **n**
Evalúa **t_pv(p,cat_mayor(i))** y asigna el resultado a **tsat(i)**

```
END FOR

Asigna 0 a t0
FOR 1 < i < n
  Calcula t0+x(i)*tsat(i) y asigna el resultado a t0
END FOR

Asigna t0 a t0K
Realiza ZEROS(n) y asigna el resultado a cof
Realiza ZEROS(n) y asigna el resultado a phicf
Realiza ZEROS(n) y asigna el resultado a ps
Realiza ZEROS(n) y asigna el resultado a y

FOR 1 < i < n
   Asigna 1.0000001 a cof(i); Asigna 1.0000001 a phicf(i);
   Evalúa Psat(t0K,cat_mayor(i)) y asigna el resultado a ps(i)
   Calcula 1/n y asigna el resultado a y(i);
END FOR

Realiza ZEROS(n) y asigna el resultado a coetma1
Realiza ZEROS(n) y asigna el resultado a coetma2
FOR 1 < i < 2
   IF i == 1
      Asigna 1 a coetma1(i); Asigna 1 a coetma2(i)
   ELSEIF i == 2
      Asigna t0K a coetma1(i); Asigna t0K a coetma2(i)
   END IF
END FOR

Almacena coetma1(1,3:2+n) en y; Almacena coetma2(1,3:2+n) en ps;
IF op_gid ~= 0
   Evalúa coac(opa,prop,t0K,x,n,pathDir) y asigna el resultado a cof
END IF

Realiza ZEROS(n) y asigna el resultado a coetma3
Realiza ZEROS(n) y asigna el resultado a coetma4

FOR 1 < i < 2
   IF i == 1
      Asigna 1 a coetma3(i); Asigna 1 a coetma4(i)
   ELSEIF i == 2
      Asigna t0K a coetma3(i); Asigna t0K a coetma4(i)
   END IF
END FOR
```

```
Almacena coetma3(1,3:2+n) a cof; Almacena coetma4(1,3:2+n) a phicf;
Agrupa x, p, cof, phicf, cat_mayor, n y pathDir y asigna el resultado a l
Evalúa rootSecantePlus2A(t0K,te2,l,0.001,1,store=%T) y asigna el resultado a vri
Calcula las dimensiones de vri y asigna el resultado a s1
Almacena s1(1) en f1
Almacena vri(f1,2) en tK1
Asigna 3 en delt
Asigna 35 en imaxt
Asigna 1 en it

WHILE delt > 0.0001
   FOR 1< i < n
      Evalúa Psat(tK1,cat_mayor(i)) y asigna el resultado a ps(i)
   END FOR
   Almacena coetg2(it,3:2+n) en ps
   Asigna 40 en imaxc
   Realiza ZEROS(n) y asigna el resultado a yte(i)
   FOR 1 < i < n
      Calcula yte(i)/sumat y asigna el resultado a yte(i)
   END FOR
   Guarda coetg1(it,3:2+n) en y
   Calcula p/100 y asigna el resultado en p0phi

   IF op_gid ~= 0
      Evalúa coac(opa,prop,tK1,x,n,pathDir) y asigna el resultado a cof

      IF op_gid == 2
         Guarda param(1) en B; Guarda param(2) en opi; Guarda param(3) en opB
         Calcula phiCM(tK1,p0phi,cat_mayor,B,opi,opB,n,pathDir,y) y asigna el
         resultado a phicf
      END IF
   END IF
  Almacena coetma1(1,3:2+n) en cof; Almacena coetma2(1,3:2+n) en phicf;

  FOR 1 < i < n
      IF i ==1
         Calcula it + 1 y asigna el resultado a coetg1(it,i);
         Calcula it + 1 y asigna el resultado a coetg2(it,i)
         Calcula it + 1 y asigna el resultado a coetg3(it,i)
         Calcula it + 1 y asigna el resultado a coetg4(it,i)
      ELSEIF i == 2
         Asigna tK1 y asigna el resultado a coetg1(it,i);
         Asigna tK1 y asigna el resultado a coetg2(it,i);
         Asigna tK1 y asigna el resultado a coetg3(it,i);
```

```
            Asigna tK1 y asigna el resultado a coetg4(it,i);
          END IF
          Agrupa x, p, cof, phicf, cat_mayor, n y pathDir y asigna el resultado a l
          Evalúa rootSecantePlus2A(tK1,te2,l,0.001,2,store=%F) y asigna el resultado a
          vri
          Calcula las dimensiones de vri y asigna el resultado a s1
          Guarda s1(1) en f1
          Guarda vri(f1,2) en tK1p
          Calcula ABS(tK1p – tK1) y asigna el resultado a delt
          Asigna tK1p a tK1
          Calcula it + 1 y asigna el resultado a it

          IF it > imaxt
            Sale del ciclo
          END IF
        END FOR

        Calcula p/100 y asigna el resultado p0phi
        IF op_gid ~= 0
          Evalúa coac(opa,prop,tK1,x,n,pathDir) y asigna el resultado a cof

          IF op_gid == 2
            Guarda param(1) en B; Guarda param(2) en opi; Guarda param(3) en opB
            Calcula phiCM(tK1,p0phi,cat_mayor,B,opi,opB,n,pathDir,y) y asigna el
            resultado a phicf
          END IF
        END IF
      END WHILE

      Une coetma1 y coetg1 y asigna el resultado a coetf1;
      Une coetma2 y coetg2 y asigna el resultado a coetf2
      Une coetma3 y coetg3 y asigna el resultado a coetf3
      Une coetma4 y coetg4 y asigna el resultado a coetf4

      Agrupa coetf1, coetf2, coetf3 y coetf4 y asigna el resultado a coetf_gru
      Asigna “T” a car1; Asigna “y” a car2; Asigna “Tem” a car3; Asigna “\\Burbuja” a
      nfile
      Asigna “Burbuja T” a titulo
      Evalúa imp_op_tab(coetf_gru,n,car1,car2,car3,pathDir,imp,nfile,titulo)
      Agrupa "burbuja", p, t, x, y, cof y phicf y asigna el resultado a catalog
      Devuelve catalog

END FUNCTION
```

```
FUNCTION mat_val(n,car)

   Asigna 2 a ind
   WHILE ind == 2
      Realiza ZEROS(n, n) y asigna el resultado a vij
      Crea un vector vacío y asigna el resultado a v_stv
      Crea una lista vacía y asigna el resultado a sub_l
      Asigna 1 a k

      FOR 1 < i < n
         FOR 1 < j < n
            IF j ~= i
               INPUT parámetro de interacción correspondiente a car y asigna el resultado
                     a vij(i,j)
               Asigna vij(i,j) a v_stv; Agrupa i y j y asigna el resultado a sub_l(k)
               Calcula k + 1 y asigna el resultado a k

            ELSEIF j == 1
               Asigna 0 a vij(i,j)
            END
         END FOR
      END FOR

      Calcula LENGTH(v_stv) y asigna el resultado a ni
      PRINT "parámetros de interacción correctos? 1. Si  2. No"
      FOR 1 < k < ni
         Almacena sub_l(k) en ij; Guarda ij(1) en i; Guarda ij(2) en j
         PRINT "Parámetro de interacción car(i,j)"
         PRINT v_st(k)
      END FOR
      INPUT número de entre 1 y 2 y asigna el resultado a ind
   END WHILE
   Devuelve vij

END FUNCTION
```

```
FUNCTION mat_car(n, car)

   FOR 1 < i < n
      Crea una cadena de caracteres vacía y asigna el resultado a st2A(i)
      FOR 1 < i < n
```

```
        Crea una cadena de caracteres vacía y asigna el resultado a stA
        IF j ~= i
          IF j ~= n
            Utiliza car, i y j para crear una cadena de caracteres y asigna el resultado a
            stA
          ELSEIF j == n
            Utiliza car, i y j para crear una cadena de caracteres y asigna el resultado a
            stA
          END IF

        ELSEIF j == i
          IF j ~= n
            Utiliza car, i y j para crear una cadena de caracteres y asigna el resultado a
            stA
          ELSEIF j == n
            Utiliza car, i y j para crear una cadena de caracteres y asigna el resultado a
            stA
          END IF
        END IF
        Une stA y st2A(i) y asigna el resultado a st2A
      END FOR
    END FOR

    FOR 1 < i < n
      FOR 1 < j < n
        Asigna una cadena de caracteres vacía a st2B(i,j)
      END FOR
    END FOR

    FOR 1 < i < n
      Calcula STRSPLIT(st2A(i), ",") y asigna el resultado a st2A_aux
      FOR 1 < j < n
        Almacena st2A_aux(j) a st2B(i,j)
      END FOR
    END FOR
    PRINT st2B

END FUNCTION
```

```
FUNCTION coac(opa, prop, Tsis, x, n, pathDir)

    IF opa ~= 0
      IF opa ==1
        Utilizando pathDir, establece el directorio de Wcomp y asigna el resultado a l1
```

```
      Utilizando l1, llama al espacio de trabajo a Wcomp

      Asigna prop(1) a v; Asigna prop(2) a aij
      Evalúa Wcomp(v, aij, Tsis, x, n) y asigna el resultado a cof

   ELSEIF opa ==2
      Utilizando pathDir, establece el directorio de NRTLcomp y asigna el resultado a
      l1
      Utilizando l1, llama al espacio de trabajo a NRTLcomp
      Asigna prop(1) a alf; Asigna prop(2) a bij
      Evalua NRTLcomp(alf, bij, Tsis, x, n) y asigna el resultado a cof

   ELSEIF opa ==3
      Utilizando pathDir, establece el directorio de UNIFAC y asigna el resultado a l1
      Utilizando l1, llama al espacio de trabajo a UNIFAC
      Asigna prop(1) a alf; Asigna prop(2) a bij
      Evalua UNIFAC(Tsis, x, n, pathDir, prop) y asigna el resultado a cof
      Utilizando pathDir, establece el directorio de Wcomp.py y asigna el resultado a
      dir3
      Utilizando dir3, llama al espacio de trabajo a UNIFAC
   END IF

ELSEIF opa == 0
   Realiza ONES(LENGTH(n)) y asigna el resultado a cof
END IF
Devuelve cof

END FUNCTION
```

```
FUNCTION in_dat_cof(opa, n, name)

   IF opa ==1
      PRINT "volúmenes molares de los componentes de la mezcla"
      Crea una cadena de caracteres vacía y asigna el resultado a st2
      FOR 1 < i < n
         IF i ~= n
            Utilizando name(i), elabora una cadena de caracteres y asigna el resultado a
            st(i)
         ELSEIF i == n
            Utilizando name(i), elabora una cadena de caracteres y asigna el resultado a
            st(i)
         END IF
         Une st2 y st(i) y asigna el resultado a st2
      END FOR
```

```
    Calcula STRSPLIT(st2, ",") y asigna el resultado a st2
    Realiza ZEROS(LENGTH(n)) y asigna el resultado a v
    Asigna 2 a ind
    WHILE ind == 2
      FOR 1 < i < n
        PRINT st2(i)
        INPUT propiedad definida en st2(i) y asigna el resultado a v(i)
      END FOR
      PRINT "volúmenes molares correctos? 1. Si  2. No"
      INPUT número entre 1 y 2 y asigna el resultado a ind
    END WHILE

    PRINT "matriz de los parámetros de interacción a de Wilson?"
    Asigna "a" a car
    Evalúa mat_car(n,car) y asigna el resultado a st2B
    Evalúa mat_val(n,car) y asigna el resultado a aij
    Agrupa v y aij y asigna el resultado a prop

  ELSEIF opa == 2
    PRINT "parámetro(s) de interacción alfa de NRTL"
    Asigna "α" a car
    Evalúa mat_car(n,car) y asigna el resultado a st2B

    Asigna 2 a ind
    WHILE ind == 2
      Realiza ZEROS(n, n) y asigna el resultado a alf
      Crea un vector vacío y asigna el resultado a alftem_v
      Crea una lista vacía y asigna el resultado a sub_l
      Asigna 1 a k

      FOR 1 < i < n
        FOR 1 < j < n
          IF j > i
            INPUT parámetro de interacción correspondiente a car y asigna el
                  resultado a alf_tem
            Asigna alf_tem a alftem_v(k); Agrupa i y j y asigna el resultado a
            sub_l(k)
            Calcula k + 1 y asigna el resultado a k
            Almacena alftem en alf(i,j); Almacena alf(i,j) en alf(j,i)
          END
        END FOR
      END FOR

      Calcula LENGTH(alftem_v) y asigna el resultado a ni
```

```
      PRINT "parámetros de interacción correctos? 1. Si  2. No"

      FOR 1 < k < ni
        Almacena sub_l(k) en ij; Guarda ij(1) en i; Guarda ij(2) en j
        PRINT "Parametro de interacción car(i,j):"
        PRINT v_st(k)
      END FOR
      INPUT número de entre 1 y 2 y asigna el resultado a ind
    END WHILE

    PRINT "matriz de los parámetros de interacción b de NRTL?"
    Asigna "b" a car
    Evalúa mat_car(n,car) y asigna el resultado a st2B
    Evalúa mat_val(n,car) y asigna el resultado a bij
    Agrupa alf y bij y asigna el resultado a prop

  ELSEIF opa == 3
    Asigna n(1) a pathDir; Asigna n(2) a nv
    Utilizando pathDir, establece el directorio de tablas_read y asigna el resultado a
    path3
    Utilizando path3, llama a path3 al espacio de trabajo
    Evalúa dict_mapping(path2) y asigna el resultado a lmaps
    Almacena lmaps(1) en map_sg_g; Almacena lmaps(2) en tab_amk1;
    Almacena lmaps(3) en lookmap_RQ;
    Guarda map_sg_g(1) en mapping_subg; Guarda map_sg_g(2) en mapping_g;
    Asigna mapping_subg(1) en lcomplookupt;
    Crea una lista vacía y la asigna a comp_i

    Asigna 2 a ind2
    WHILE ind2 ==2
      FOR 1 < i < nv
        INPUT Cantidad de subgrupos SIN REPETIR del name(i) y asigna el
              resultado a num_sub
        WHILE %T
          IF num_sub < 0
            INPUT Cantidad de subgrupos SIN REPETIR del name(i) y asigna el
                  resultado a num_sub
          ELSE
            Sale del ciclo
          END IF
        END WHILE
        Crea una lista vacía y la asigna a sgd
        Crea un vector y asigna un sub
        PRINT "subgrupos SIN REPETIR del name(i) y frecuencia de repetición"
```

```
    FOR 1 < j < num_sub
      INPUT subgrupo j SIN REPETIR del name(i) y asigna el resultado a subgt
      Asigna 1 a ind
      WHILE ind == 1
        FOR 1 < k < LENGTH(lcomplookup)
          IF subgt == lcomplookup(k)
            Sale del ciclo
          END IF
        END FOR

        IF ALL(subgt ~= lcomplookup)
          PRINT "no puede hallarse el subgrupo en table UNIFAC"
          INPUT subgrupo j SIN REPETIR del name(i) y asigna el resultado a
                subgt
        ELSE
          Asigna 2 a ind
        END IF
      END WHILE
      Asigna subgt a subg(j)
      INPUT frecuencia de repetición del subgrupo j del name(i) y asigna el
            resultado a numsg(i)
    END FOR
    Asigna subg a sg
    FOR 1 < j < num_sub
      Asigna 0 a ind
      IF sg(j) == "CHO"
        PRINT "subgrupo j del compuesto sg(j) del compuesto i"
        PRINT "1. grupo secundario 20"
        PRINT "2. grupo secundario 26"
        INPUT número entre 1 y 2 y asigna el resultado a sgdt
        IF sgdt == 1
          Asigna 10 a ns; Asigna 20 a npr; Asigna "CHO" a g
        ELSEIF sgdt == 1
          Asigna 13 a ns; Asigna 26 a npr; Asigna "CHO" a g
        END IF
        Agrupa npr, ns y g y asigna el resultado a sgd(j)
      ELSE
        Crea una lista vacía y asigna el resultado a sgd(j)
      END IF
    END FOR

    Agrupa sg, numsg en una lista, se agrupa esta lista y sgd en una nueva lista y
    asigna el resultado a comp_i(i)
  END FOR
```

```
        Crea un vector con elementos entre 1 y LENGTH(comp_i) y asigna el resultado
        a clv
        FOR 1 < i < LENGTH(clv)
           Almacena clv(i) a clave(i)
        END FOR
        Crea un struct vacío[20] y asigna el resultado a l_c(i)

        FOR 1 < i < LENGTH(clv)
           Almacena comp_i(i) como l_c(clv(i))
        END FOR

        Evalúa tabs_RQ_av(path2,l_c,clave) y asigna el resultado a prop
        PRINT "Nombres de compuestos, subgrupos y frecuencia de subgrupos para
                estos compuestos, correctos?"
        INPUT número entre 1 y 2 y asigna el resultado a ind2
      END WHILE
   ENDIF
   Devuelve prop

END FUNCTION
```

Wcomp

```
FUNCTION Wcomp(v, aij, Tsis, x, n)

   Asigna 1.987 a R
   Realiza ZEROS(n, n) y asigna el resultado a lamb
   FOR 0 < i < n
      FOR 0 < j < n
         Calcula v(j)*EXP(-aij(i,j)/(R*Tsis))/v(i) y asigna el resultado a lamb(i,j)
      END FOR
   END FOR

   Realiza ZEROS(LENGTH(n)), y lo asigna a s1
   Realiza ZEROS(LENGTH(n)), y lo asigna a sk
   Realiza ZEROS(LENGTH(n)), y lo asigna a ln_coef_act
   FOR 0 < i n
      Realiza ZEROS(LENGTH(n)) y lo asigna a p1
      FOR 0 < i < n
         Calcula x(j)*lamb(i,j) y asigna el resultado a p1(j)
      END FOR
```

[20] En Python, la función del "struct" es cumplida por el "diccionario".

```
      Calcula 1 - LOG(SUM(p1)) y asigna el resultadoa a s1(i)
      Realiza ZEROS(LENGTH(n)) y lo asigna a rk

      FOR k se encuentra entre 0 y n
         Realiza ZEROS(LENGTH(n)) y lo asigna a p3
         FOR 1 < j < n
            Calcula x(k)*lamb(k,i)/SUM(p3) y asigna el resultado a rk(k)
         END FOR
         Calcula x(k)*lamb(k,i)/SUM(p3) y asigna el resultado a rk(k)
      END FOR

      Calcula -SUM(rk) y asigna el resultado a sk(i)
      Calcula s1(i) + sk(i) y asigna el resultado a ln_coef_act(i)
   END FOR

   Realiza ZEROS(LENGTH(n)) y asigna el resultado a Cof
   FOR 0 < i < n
      Calcula EXP(ln_coef_act(i)) y asigna el resultado a Cof(i)
   END FOR
   Devuelve Cof

END FUNCTION
```

NRTLcomp

```
FUNCTION NRTLcomp(alf, bij, Tsis, x, n)

   Asigna 1.987 a R
   Realiza ZEROS(n, n) y asigna el resutlado a tauij
   Realiza ZEROS(n, n) y asigna el resutlado a G

   FOR 0 < i < n
      FOR 0 < j < n
         Calcula bij(i,j)/(R * Tsis) y asigna el resultado a tauij(i,j)
         Calcula EXP(alf(i,j) *tauij(i,j)) y asigna el resultado a G(i,j)
      END FOR
   END FOR

   Realiza ZEROS(LENGTH(n)) y asigna el resultado a frac3
   Realiza ZEROS(LENGTH(n)) y asigna el resultado a ln_coef_act
   Realiza ZEROS(LENGTH(n)) y asigna el resultado a sfrac2

   FOR 0 < i < n
```

```
    Realiza ZEROS(LENGTH(n)) y asigna el resultado a frac
    Realiza ZEROS(LENGTH(n)) y asigna el resultado a frac2

    FOR 0 < j < n
      Realiza ZEROS(LENGTH(n)) y asigna el resultado a p11
      Realiza ZEROS(LENGTH(n)) y lo asigna a p12

      FOR 0 < k < n
        Calcula x(k) * tauij(k,j) * G(k,j) y asigna el resultado en p11(k)
        Calcula G(k,j) * x(k) y asigna el resultado en p12(k)
      END FOR

      Calcula tauij(i,j) – SUM(p11)/SUM(p12) y asigna el resultado a frac(j)
      Calcula x(j) * G(i,j) * frac(j)/SUM(p12), y asigna el resultado a frac2(j)
    END FOR

    Calcula SUM(frac2) y asigna el resultado en sfrac2(i)
    Realiza ZEROS(LENGTH(n)) y guarda el resultado en p31
    FOR 0 < j < n
      Calcula tauij(j,i) * G(j,i) * x(j) y asigna el resultado a p31
    END FOR

    Realiza ZEROS(LENGTH(n)) y guarda el resultado en p32
    FOR 0 < k < n
      Calcula G(k,i) * x(k) y asigna el resultado p32(k)
    END FOR

    Calcula SUM(p31)/SUM(p32) y asigna el resultado a frac3(i)
    Calcula frac3(i) + sfrac2(i) y asigna el resultado a ln_coef_act(i)
  END FOR

  Realiza ZEROS(LENGTH(n)) y asigna el resultado a Cof
  FOR 0 < i < n
    Calcula EXP(ln_coef_act(i)) y asigna el resultado a Cof(i)
  END FOR
  Devuelve Cof

END FUNCTION
```

UNIFAC

```
FUNCTION UNIFAC(Tsis, x, n, dir3, prop)

   Almacena prop(1) en tabRk_Qk; Almacena prop(2) en amk;
   Almacena prop(3) en vki; Almacena prop(4) en sv;

   IF LENGTH(sv) ~= 0
      PRINT "Pruebe otro método para el cálculo de los γ_i"

      FOR 1 < i < LENGTH(sv)
         PRINT sv(i)
      END FOR
      Realiza ONES(LENGTH(vki)) y asigna el resultado a coeti
   ELSE
      Calcula las dimensiones de tabRk_Qk y asigna el resultado a vdfil
      Almacena vdfil(1) en dfil
      Calcula 3 + n y asigna el resultado a dcol
      Guarda dfil en nk
      Almacena tabRk_Qk(:,1) en Rk; Almacena tabRk_Qk(:,2) en Qk;
      Guarda ni en n
      Crea una lista vacía y la asigna a ri; Crea una lista vacía y la asigna a qi

      FOR 1 < i < ni
         Crea una lista vacia y la asigna a sri; Crea una lista vacia y la asigna a sqi
         FOR 1 < k < nk
            Calcula vki(k,i) * Rk(k) y asigna el resultado a sri(k)
            Calcula vki(k,i) * Qk(k) y asigna el resultado a sqi(k)
         END FOR
         Calcula SUM(sri) y asigna el resultado a ri(i)
         Calcula SUM(sqi) y asigna el resultado a qi(i)
      END FOR

      Realiza ZEROS(nk, ni) y asigna el resultado a eki
      FOR 1 < k < nk
         FOR 1 < i < ni
            Calcula vki(k,i) * Qk(k)/qi(i) y asigna el resultado a eki(k,i)
         END FOR
      END FOR

      Realiza ZEROS(nk, nk) y asigna el resultado a tau_mk
      FOR 1 < m < nk
         Asigna amk(m) a amk_m
```

```
   FOR 1 < k < nk
      Calcula EXP(-amk(k)/Tsis) y asigna el resultado a tau_amk(m,k)
   END FOR
END FOR

Realiza ZEROS(ni, nk), y asigna el resultado a bik
FOR 1 < i < ni
   FOR 1 < k < nk
      Crea una lista vacia y la asigna smk
      FOR 1 < m < nk
         Calcula eki(m,i) * tau_mk(m,k), y asigna el resultado a smk(m)
      END FOR
      Almacena SUM(smk) en bik(i,k)
   END FOR
END FOR

Crea una lista vacia y la asigna a tek
FOR 1 < k < nk
   Crea una lista vacia y la asigna a sn
   FOR 1 < i < ni
      Calcula x(i) * q(i) * eki(k,i), y el resultado lo asigna a sn
   END FOR

   Calcula SUM(sn) y el resultado lo asigna a num
   Crea una lista vacia y la asigna a sd
   FOR 1 < i < ni
      Calcula x(i) * q(i), y el resultado lo asigna a sd
   END FOR
   Calcula num/SUM(sd), y asigna el resultado a tek
END FOR

Crea una lista vacia y la asigna a sk
FOR 1 < k < nk
    Crea una lista vacia la asigna a suk

   FOR m se encuentra entre 0 y nk
      Calcula tek(m) * tau_mk(m,k) y asigna el resultado a suk(m)
   END FOR
   Calcula SUM(suk) y asigna el resultado a sk(k)
END FOR

Realiza ZEROS(LENGTH(ni)) y lo asigna a Ji
FOR 1 < i < ni
   Crea una lista vacia y la asigna a sd
```

```
      FOR 1 < j < ni
        Calcula ri(j) * x(j) y asigna el resultado a sd(j)
      END FOR

      Calcula ri(i)/SUM(sd), y asigna el resultado a Ji(i)
    END FOR

    Realiza ZEROS(LENGTH(ni)) y asigna el resultado a Li
    FOR 1 < i < ni
      Crea una lista vacia y la asigna a sd
      FOR j se encuentra entre 0 y ni
        Calcula qi(j) * x(j), y asigna el resultado a sd(j)
      END FOR
      Calcula qi(i)/SUM(sd) y asigna el resultado a Li(i)
    END FOR

    Crea una lista vacia y la asigna a ln_coetiC; Crea una lista y la asigna a ln_coetiR
    Crea una lista y la asigna a fc

    FOR 1 < i < ni
      1-Ji(i)+LOG(Ji(i))-5*qi(i)*(1-Ji(i)/Li(i)+LOG(Ji(i)/Li(i))) y asigna el resultado
      a ln_coetiC(i)
      Crea una lista vacia y la asigna a sfi
      FOR 1 < k < nk
        Calcula tek(k)*bik(i,k)/sk(k)-eki(k,i)*LOG(bik(i,k)/sk(k)) y asigna el
        resultado a sfi(k)
      END FOR
      Calcula SUM(sfi) y asigna el resultado a fc(i)
      Calcula qi(i) * (1-fc(i)) y asigna el resultado a ln_coetiR(i)
    END FOR

    Calcula ln_coetiC + ln_coetiR y asigna el resultado a ln_coeti
    Calcula EXP(ln_coeti) y el resultado lo asigna a coeti
  END IF
  Devuelve coeti

END FUNCTION
```

phiCM

```
FUNCTION phisat(catg1, Bii, Tsis, pathDir)

  Utilizando a pathDir, establece la ruta del directorio de fpres_vap y asigna el
```

resultado a **dir2_1**
Utilizando **dir2_1**, llama al espacio de trabajo a **Psat** desde **fpres_vap**

Evalua **Psat(Tsis, catg1)**, y asigna el resultado a **PsatKpa**
Calcula **PsatKpa**/100 y asigna el resultadoa a **Psatbar**
Calcula **EXP(Bii** * **Psatbar**/(**Tsis** * 83.14)**)**, y luego asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **phiCM(Tsis, Psis, cat_mayor, B, opi, opB, n, pathDir, param)**

IF **opB** es igua a 1
Asigna **B**(:,1) a **tc;** Asigna **B**(:,2) a **pc;** Asigna **B**(:,3) a **vc;** Asigna **B**(:,4) a **zc;**
Asigna **B**(:,5) a **w**
Elimina **B**

Realiza **ONES(LENGTH(n))** * **Tsis** y asigna el resultado a **T**
Calcula **T**/**Tc** y asigna el resultado a **tr**
Realiza **ZEROS(n,n)** y asigna el resultado a **wm;**
Realiza **ZEROS(n,n)** y asigna el resultado a **tcm**
Realiza **ZEROS(n,n)** y asigna el resultado a **vcm**
Realiza **ZEROS(n,n)** y asigna el resultado a **zcm**
Realiza **ZEROS(n,n)** y asigna el resultado a **trm**
Realiza **ZEROS(n,n)** y asigna el resultado a **pcm**

FOR 1 < **i** < **n**
FOR 1 < **j** < **n**
IF **i** ==**j**
Almacena **w**(**j**) en **wm**(**i**,**j**)
Almacena **tc**(**j**) en **tcm**(**i**,**j**)
Almacena **vc**(**j**) en **vcm**(**i**,**j**)
Almacena **zc**(**j**) en **zcm**(**i**,**j**)
Almacena **pc**(**j**) en **pcm**(**i**,**j**)
Almacena **tr**(**j**) en **trm**(**i**,**j**)

IF **i** ~= **j**
Calcula 0.5*(**w**(**i**)+**w**(**j**)) y asigna el resultado a **wm**(**i**,**j**);
Almacena **wm**(i,j) en **wm**(j,i)
Calcula (**tc**(**i**)+**tc**(**j**))^0.5 y asigna el resultado a **tcm**(**i**,**j**);
Almacena **tcm**(i,j) en **tcm**(j,i)
Calcula (((**vc**(**i**))^(1/3)+(**vc**(**j**))^(1/3))/2)^3 y asigna el resultado a **vcm**(**i**,**j**);
Almacena **vcm**(i,j) en **vcm**(j,i)
Calcula 0.5*(**zc**(**i**)+**zc**(**j**)) y asigna el resultado a **zcm**(**i**,**j**);

```
            Almacena zcm(i,j) en zcm(j,i)
            Calcula zcm(i,j)*83.14*tcm(i,j)/vcm(i,j) y asigna el resultado a pcm(i,j);
            Almacena pcm(i,j) en pcm(j,i)
          END IF
        END FOR
      END FOR

      Realiza ZEROS(n, n), y asigna el resultado a Bn_r
      Realiza ZEROS(n), y asigna el resultado a Bn_i
      Realiza ZEROS(n), y asigna el resultado a B0
      Realiza ZEROS(n), y asigna el resultado a B1

      FOR 1 < i < n
        FOR 1 < j < n
          IF opi ~=1
            Calcula 0.083-0.422/((trm(i,j))^1.6) y asigna el resultado en B0(j)
            Calcula 0.139-0.172/((trm(i,j))^4.2) y asigna el resultado en B1(j)
            Calcula (B0(j)+B1(j)*wm(i,j))*83.14*tcm(i,j)/pcm(i,j) y asigna el
            resultado en Bn_r(i,j)
            Almacena Bn_r(i,j) en Bn_r(j,i)

          ELSEIF opi ==1
            IF i ==j
              Calcula 0.083-0.422/((trm(i,j))^1.6) y asigna el resultado a B0(j)
              Calcula 0.139-0.172/(trm(i,j)^4.2) y asigna el resultado a B1(j)
              Calcula (B0(j)+B1(j)*wm(i,j))*83.14*tcm(i,j)/pcm(i,j) y asigna el
              resultado a Bn_i(j)
            END IF
          END IF
        END FOR
      END FOR
    END

    IF opi == 1
      IF opB == 1
        Asigna Bn_i a B
      END IF

      Realiza ZEROS(n) y asigna el resultado a ln_phi_solid
      Realiza ZEROS(n) y asigna el resultado a phi_solid
      Realiza ZEROS(n) y asigna el resultado a phis
      Realiza ZEROS(n) y asigna el resultado a phiM

      FOR 1 < i < n
```

```
      Calcula Psis * B(j)/(83.14 * Tsis) y asigna el resultado a ln_phi_solid
      Calcula EXP(ln_phi_solid(j)) y asigna el resultado a phi_solid(j)
      Evalúa phisat(cat_mayor(j),B(j),Tsis) y asigna el resultado a phis(j)
    END FOR
    Asigna phiM a tabl_calc;
  ELSEIF opi ~=1
    IF opB ==1
      Asigna Bn_r a B
    END IF

    Asigna param a y
    Realiza ZEROS(n, n) y asigna el resultado a delta_ji
    Realiza ZEROS(n, n) y asigna el resultado a delta_jk
    Realiza ZEROS(n, n) y asigna el resultado a pro
    Realiza ZEROS(n), y asigna el resultado a sp_k
    Realiza ZEROS(n), y asigna el resultado a sp_i
    Realiza ZEROS(n), y asigna el resultado a expr_corch
    Realiza ZEROS(n), y asigna el resultado a ln_phi_sol
    Realiza ZEROS(n), y asigna el resultado a phi_sol
    Realiza ZEROS(n), y asigna el resultado a phis
    Realiza ZEROS(n), y asigna el resultado a phiM

    FOR 1 < i < n
      FOR 1 < j < n
        IF j ==i
          Asigna 0 a delta_ji(j,j)
        ELSEIF j > i
          Calcula 2 * B(j,i) – B(j,j) – B(i,i) y asigna el resultado delta_ji(j,i)
          Asigna delta_ji(j,i) a delta_ji(i,j)
        END IF
      END FOR
    END FOR
    FOR 1 < j < n
      FOR 1 < k < n
        IF k ==j
          Asigna 0 a delta_jk(k,j)
        ELSEIF k > j
          Calcula 2 * B(j,k) – B(j,j) – B(k,k)
          Asigna delta_ji(j,k) a delta_ji(k,j)
        END IF
      END FOR
    END FOR

    FOR 1 < i < n
```

```
        FOR 1 < j < n
          FOR 1 < k < n
            Calcula y(j)*y(k)*(2*delta_ji(j,i)-delta_jk(j,k)) y asigna el resultado a
            pro(j,k)
          END FOR
          Calcula SUM(pro(j,:)) y asigna el resultado a sp_k(j)
        END FOR
        Calcula SUM(sp_k) y asigna el resultado a sp_i(i)
        Calcula B(i,i) + 0.5 *sp_i(i), y asigna el resultado a expr_corch(i)
        Calcula Psis * (expr_corch(i)/(83.14 * Tsis)) y asigna el resultlado a
        ln_phi_sol(i)
        Calcula EXP(ln_phi_sol(i)) y asigna el resultado a phi_sol
        Evalua phisat(cat_mayor(i), B(i,i), Tsis, pathDir), y asigna el resultado a
        phis(i)
        Calcula phi_sol(i)/phis(i) y asigna el resultado a phiM(i)
      END FOR
      Asigna phiM a tabl_calc
    END IF
    Devuelve tabl_calc

END FUNCTION
```

tablas_read

```
FUNCTION tab_split(tab)

    Calcula las dimensiones de tab y asigna el resultado a vltab
    Guarda vltab(1) en ltab
    Crea una lista vacía y la asigna a tab2
    FOR 1 < i < ltab
      Realiza STRSPLIT(tab(i),",") y asigna el resultado a tab_tem
      Almacena tab_tem en tab2(i,:)
    END FOR
    Devuelve tab2

END FUNCTION
```

```
FUNCTION tab_eval(tab)

    Crea un vector vacío y asigna el resultado a tab_res
    Calcula las dimensiones de tab y asigna el resultado a vltab
    Almacena vltab(1) en ltab; Almacena vltab(2) en coltab
```

```
   Almacena tab(1:ltab,1:2) en th1; Almacena tab(1:ltab,4:coltab) en th2;

   FOR 1 < i < ltab
      Asigna th2(i,:) en taux; Calcula LENGTH(taux) y asigna el resultado a ltaux
      Crea un vector vacío y lo asigna a fil_aux1; Crea un vector vacío y lo asigna a
      fil_aux2;

      FOR 1 < j < 2
         Realiza EVSTR(th1(i,j)) y asigna el resultado a fil_aux1(j)
      END FOR

      FOR 1 < j < ltaux
         IF th2(i,j) ~= "NA"
            Realiza EVSTR(th2(i,j)) y asigna el resultado a fil_aux2(j)
         END IF
      END FOR

      Une fil_aux1 y fil_aux2 y asigna el resultado a tab_res(i,:)
   END FOR
   Devuelve tab_res

END FUNCTION
```

```
FUNCTION col_extract(tab, idc)

   Crea un vector vacío y asigna el resultado a col
   Calcula las dimensiones de ltab y asigna el resultado a vltab; Asigna vltab(1) a ltab
   FOR 1 < i < ltab
      Asigna tab(i,idc) a col(i)
   END FOR
   Devuelve col

END FUNCTION
```

```
FUNCTION take_once_adjacent_dups2(xs)

   Crea un vector vacío y lo asigna a result; Crea un vector vacío y lo asigna a el
   Asigna "None" a most_recent_elem
   Calcula LENGTH(xs) y asigna el resultado a lxs
   Asigna 1 a e_aux
   FOR 1 < e < lxs
      IF xs(e) ~= most_recent_elem
         Almacena xs(e) en result(e_aux); Asigna xs(e) en most_recent_elem;
```

```
        Calcula e_aux + 1 y asigna el resultado en e_aux; Asigna 0 a c
      ELSEIF xs(e) == most_recent_elem
        Calcula c + 1 y asigna el resultado a c
        IF c == 1
          Asigna xs(e) en el
        END IF
      END IF
    END FOR
    Agrupa result y el y asigna el resultado a tab_res
    Devuelve tab_res

END FUNCTION
```

```
FUNCTION sg_mod(ln_comp, map_subg, cl, idtsec, idtpriml)

    Calcula LENGTH(cl) y asigna el resultado a le_cl
    Evalúa dict_create(map_subg(2),map_subg(1)) y asigna el resultado a map_nsg
    FOR 1 < i < le_cl
      Alamacena ln_comp(cl(i)) en com_i
      Almacena com_i(1) en sg; Almacena com_i(2) en sgd
      Almacena sg(1) en ingcomp_el; Almacena idtsecl(EVSTR(cl(i)) en idtsec_i
      Almacena idtpriml(EVSTR(cl(i))) en idtprim_i

      IF LENGTH(sgd) ~= 0
        FOR 1 < j < LENGTH(incomp_el)
          IF sgd(j) ~= lista vacía
            Asigna sgd(j) a sgd_j
            IF sgd_j(3) == search_value(map_nsg,idtsec_i(j))
              Asigna sgd_j(1) a idtsec_i(j); Asigna sgd_j(2) a idtprim_i(j)
            END IF
          END IF
        END FOR
        Asigna idtsec_i a idtsecl(i); Asigna idtprim_i a idtpriml(i)
      END IF
    END FOR
    Crea una lista vacía y asigna el resultado a idtsecn
    Crea una lista vacía y asigna el resultado a idtprimn

    FOR 1 < i < LENGTH(idtsecl)
      Une idtsecn e idtsecl(i) y asigna el resultado a idtsecn
      Une idtprimn e idtpriml(i) y asigna el resultado a idtprimn
    END FOR
    Agrupa idtsecn, idtprimn, idtsecl, idtprimI y asigna el resultado a lsp
    Devuelve lsp
```

END FUNCTION

FUNCTION **id_comp_Rk_Qk(map_subg,map_g,l_gcomp,ln_comp,cl)**
Crea una lista vacía y asigna el resultado a **idtsecl**
Crea una lista vacía y asigna el resultado a **idtpriml**

FOR 1 < **i** < **LENGTH(l_gcomp)**
Asigna **l_gcomp**(**i**) a **l_gcomp_i**
Crea una lista vacía y asigna el resultado a **filaux1**;
Crea una lista vacía y asigna el resultado a **filaux2**;

FOR 1 < **j** < **LENGTH(l_gcomp_i)**
Evalúa **search_value(map_subg,l_gcomp_i(j))** y asigna el resultado a **filaux1**(**j**)
Evalúa **search_value(map_g,l_gcomp_i(j))** y asigna el resultado a **filaux2**(**j**)
END FOR
Asigna **filaux1** a **idtsecl**(**i**); Asigna **filaxu2** a **idtpriml**(**i**);
END FOR

Evalúa **sg_mod(ln_comp,map_subg,cl,idtsecl,idtprimI)** y asigna el resultado a **lsp**
Asigna **lsp**(1) a **idtsec**; Asigna **lsp**(2) a **idtprim**; Asigna **lsp**(3) a **idtsecI**
Evalúa **SORT(idtsec)** y asigna el resultado a **idtsec**
Evalúa **SORT(idtprim)** y asigna el resultado a **idtprim**
Evalúa **dict_create(map_subg**(2)**,map_g**(2)**)** y asigna el resultado a **map_itdsp**
Evalúa **take_once_adjacement_dups2(idtsec)** y asigna el resultado a **idtsec_el**
Asigna **idtsec_el**(1) a **idtsec_ab**

FOR 1 < **i** < **LENGTH(idtsec_ab)**
Evalúa **search_value(map_idtsp,idtsec_ab(i))** y asigna el resultado a **idtprim_nab**(**i**)
END FOR

Evalúa **dict_create(map_subg**(2)**, map_subg**(1)**)** y asigna el resultado a **map_nsg**
Crea una lista vacía y asigna el resultado a **lsc_ab**
FOR 1 < **i** < **LENGTH(idtsec)**
Evalúa **search_value(map_nsg,idtsec**(**i**)**)** y asigna el resultado a **lsc_ab**(**i**)
END FOR
Agrupa a **idtprim_nab**, **idtsec_ab**, **lsc_ab** e **idtsecl** y asigna el resultado a **res_lpsc**
Devuelve **res_lpsc**

END FUNCTION

```
FUNCTION val_amk(lkp_amk,idt)

   Calcula LENGTH(idt) y asigna el resultado a n
   Crea una lista vacía y asigna el resultado a tab_amk
   FOR 1 < i < n
      Crea un vector vacío y asigna el resultado a tab_amk_aux
      FOR 1 < j < n
         Asigna idt(j) a c; Asigna idt(i) a f
         Asigna lkp_amk(f,c) a tab_amk_aux(j)
      END FOR
      Asigna tab_amk_aux a tab_amk(i)
   END FOR

   Crea una lista vacía y la asigna a tab2; Crea un vector vacío y lo asigna a loc
   FOR 1 < i < LENGTH(tab_amk)
      Asigna tab_amk(i) a taux; Crea un vector vacío y lo asigna a fil_aux
      FOR 1 < j < ltaux
         IF taux(j) ~= "NA"
            Asigna EVSTR(taux(j)) y asigna el resultado a fil_aux(j)
         ELSE
            Agrupa en una lista a i y j y asigna el resultado a loc
         END IF
      END FOR
      Asigna fil_aux a tab2(i)
   END FOR

   Asigna tab2 a tab_amk
   Calcula LENGTH(tab_amk(1)) y asigna el resultado a lf
   Crea un vector vacío y asigna el resultado a lfs
   Calcula las dimensiones de tab_amk y asigna el resultado a vltab
   Asigna vltab(1) a ltab; Asigna LENGTH(tab_amk) a ltab
   FOR 1 < i < ltab
      Calcula LENGTH(tab_amk(i)) y asigna el resultado a lfs(i)
   END FOR

   IF AND(lfs) == lf
      Asigna 1 a amk_vl
   ELSE
      Asigna 2 a amk_vl
   END IF
   Agrupa tab_amk, amk_vl y loc y asigna el resultado a t_amk_vl_loc
   Devuelve t_amk_vl_loc

END FUNCTION
```

FUNCTION **dict_create(field_col, value_col)**

Agrupa **field_col** y **value_col** y asigna el resultado a **map**
Devuelve **map**

END FUNCTION

FUNCTION **dict_mapping(pathDir)**

Utilizando **pathDir**, establece la carpeta de trabajo
Utilizando **pathDir**, establece el directorio de **tab_Rk_Qk.csv** y asigna el resultado a **direc2**
Utilizando **pathDir**, establece el directorio de **tab_amk.csv** y asigna el resultado a **direc3**
Realiza **csvRead(direc2)**[21] y asigna el resultado a **tabRk_Qk0**
Realiza **csvRead(direc3)** y asigna el resultado a **tab_amk0**
Evalúa **tab_split(tabRk_Qk0)** y asigna el resultadoa **tabRk_Qk1**
Evalúa **tab_eval(tabRk_Qk1)** y asigna el resultadoa **lookupRk_Qk**
Evalúa **tab_split(tab_amk0)** y asigna el resultadoa **tab_amk1**
Evalúa **col_extract(tabRk_Qk1, 3)** y asigna el resultado a **sub_g**
Evalúa **col_extract(lookup_Rk_Qk, 1)** y asigna el resultado a **num_prim**
Evalúa **col_extract(lookup_Rk_Qk, 2)** y asigna el resultado a **num_sec**
Crea un vector con elementos entre 1 y 108 y asigna el resultado a **num_sec2**
Evalúa **dict_create(num_sec,num_sec2)** y asigna el resultado a **mapping_RQ**
Evalúa **dict_create(sub_g,num_sec)** y asigna el resultado a **mapping_subg**
Evalúa **dict_create(sub_g,num_prim)** y asigna el resultado a **mapping_g**
Agrupa **mapping_subg** y **mapping_g** y asigna el resultado a **map_sg_g**
Agrupa **lookupRk_Qk** y **mapping_RQ** y asigna el resultado a **lookmap_RQ**
Agrupa **map_sg**, **tab_amk1** y **lookmap_RQ** y asigna el resultado a **map3**

END FUNCTION

FUNCTION **search_value(tab,key)**

Asigna **tab**(1) y asigna el resultado a **field_value_col**;
Asigna **tab**(2) y asigna el resultado a **value_col**;
Calcula **LENGTH(value_col)** y asigna el resultado a **longt**
FOR 1 < **i** < **longt**
IF **key** ==**field_value_col**(**i**)

[21] En Python, fue necesaria la utilización de una función adicional para la lectura de los archivos csv tab_Rk_Qk.csv y tab_amk.csv

```
        Asigna value_col(i) a value
      END IF
   END FOR

END FUNCTION
```

```
FUNCTION tabs_RQ_av(pathDir, l2_comp,cl)

   Crea una lista vacía y asigna el resultado a ingcomp;
   Crea una lista vacía y asigna el resultado a nt; Crea una lista vacía y asigna el
   resultado a sgd

   FOR 1 < i < LENGTH(cl)
      Asigna l2_comp(cl(i)) a comp_i
      Asigna comp_i(1) a sg; Asigna comp_i(2) a sgd(i)
      Asigna sg(1) a ingcomp(i); Asigna sg(2) a nt(i)
   END IF

   Evalúa dict_mapping(pathDir) y asigna el resultado a map3
   Asigna map3(1) a map_sg_g; Asigna map3(2) a tab_amk1;
   Asigna map3(3) a lookup_Rk_Qk
   Asigna lookmap_RQ(1) a lookup_Rk_Qk; Asigna lookmap_RQ(2) a mapping_RQ
   Asigna map_sg_g(1) a mapping_subg; Asigna map_sg_g(2) a mapping_g
   Evalúa id_comp_Rk_Qk(mapping_subg,mapping_g,ingcomp,l2_comp,cl) y asigna
   el resultado a lpsg
   Asigna lpsg(1) a l_prim_nab; Asigna lpsg(2) a l_sec_ab;
   Asigna lpsg(3) a g_ab; Asigna lpsg(4) a num_gru;

   Realiza ZEROS(LENGTH(l_sec_ab),LENGTH(ingcomp)) y asigna el resultado a
   vki
   FOR 1 < i < LENGTH(l_sec_ab)
      FOR 1 < k < LENGTH(cl)
         Asigna num_gru(k) a num_gru_k; Asigna nt(k) a nt_k
         FOR 1 < j < LENGTH(num_gru_k)
            IF l_sec_ab(i) == num_gru_k(j)
               Asigna nt_k(j) a vki(i,k)
            END IF
         END FOR
      END FOR
   END FOR

   Evalúa val_amk(tab_amk1, l_prim_nab) y asigna el resultado a t_amk_vl_loc
   Asigna t_amk_vl_loc(1) y asigna el resultado a tab_amk2
   Asigna t_amk_vl_loc(2) y asigna el resultado a amk_vl
```

```
   Asigna t_amk_vl_loc(3) y asigna el resultado a loc
   Crea un vector vacío y lo asigna a sv

   IF amk_vl == 2
      Calcula LENGTH(loc) y asigna el resultado a l
      FOR 1 < i < l
         Asigna loc(i) a ub; Asigna ub(1) a ubf; Asigna ub(2) a ubc;
         Asigna "El parámetro a(ubf,ubc) no se encuentra disponible" a s1
         Asigna ",ubf,ubc" a s2; Agrupa a s1 y s2 y asigna el resultado a sv(i)
      END FOR
      Asigna "Pruebe otro metodo para el calculo de los coeficientes de actividad" a
      tab_amk
   END IF

   FOR 1 < i < LENGTH(l_sec_ab)
      Evalúa search_value(mapping_RQ,l_sec_ab(i)) y asigna el resultado a idl
      Asigna lookup_Rk_Qk(idl,3:$) a tab_Rk_Qk
   END FOR
   Agrupa tab_Rk_Qk, tab_amk2, vki y sv y asigna el resultado a l_tabs

END FUNCTION
```

opc

```
FUNCTION mat_carB(n,car,opi)

   FOR 1 < i < n
      Crea una cadena de caracteres vacía y la asigna a st2A(i)
      FOR 1 < j < n
        Crea una cadena de caracteres vacía y la asigna a stA
        Utilizando stA, crea una cadena de caracteres a ser impresa
        Une stA y st2A(i) y guarda el resultado a en st2A(i)
     END FOR
   END FOR

   FOR 1 < i < n
      FOR 1 < j < n
         Crea una cadena de caracteres vacía y la asigna a st2B(i,j)
      END FOR
   END FOR

   FOR 1 < i < n
      Calcula STRSPLIT(st2A(i), ", ") y asigna el resultado a st2A_aux
```

```
      FOR 1 < j < n
         Almacena st2A_aux(j) en st2B(i,j)
      END FOR
   END FOR
   PRINT st2B

END FUNCTION
```

```
FUNCTION ingreso_constantesB(car,car2,st1,tpc)

   Calcula LENGTH(car) y asigna el resultado a n
   Asigna 2 a ind
   WHILE ind == 2
      FOR 1 < i < n
         IF tpc ~= 0
            IF i == 1
               Asigna tpc(1) a prop_c(i)
            ELSEIF i == 2
               Asigna tpc(2) a prop_c(i)
            ELSE
               INPUT propiedad correspondiente a car(i) del compuesto st1 y asigna el
                     resultado a prop_c(i)
            END IF
         ELSEIF tpc ==0
               INPUT propiedad correspondiente a car(i) del compuesto st1 y asigna el
                     resultado a prop_c(i)
         END
      END FOR
      PRINT "propiedades correctas del compuesto st1? 1. Si  2. No"
      PRINT "car(i) = prop_c(i)?"
      INPUT número entre 1 y 2 y asigna el resultado a ind
   END WHILE

END FUNCTION
```

```
FUNCTION mat_valB(n,car,opi)

   Asigna 2 a ind
   WHILE ind == 2
      Realiza ZEROS(n,n) y asigna el resultado a Bcof
      Crea un vector vacío y asigna el resultado a B_stv
      Crea una lista vacía y asigna el resultado a sub_l
      Asigna 1 a k
```

```
      FOR 1 < i < n
        FOR 1 < j < n
          IF j == i
            INPUT coeficiente del virial B(i,j) y asigna el resultado a B_stv(k)
            Agrupa i y j y asigna el resultado a sub_l(k)
            Calcula k + 1 y asigna el resultado a k
            Asigna B_stv(k) a Bcof(i,j); Asigna Bcof(i,j) a Bcof(j,i)
          END IF
          IF opi == 2
            IF j > i
              INPUT coeficiente del virial B(i,j) y asigna el resultado a B_stv(k)
              Agrupa i y j y asigna el resultado a sub_l(k)
              Calcula k + 1 y asigna el resultado a k
              Asigna B_stv(k) a Bcof(i,j); Asigna Bcof(i,j) a Bcof(j,i)
            END IF
          END IF
        END FOR
      END FOR

      Calcula LENGTH(ni) y asigna el resultado a ni
      PRINT "Datos correctos? 1. Si  2. No"
      FOR 1 < k < ni
        Guarda sub_l(k) y asigna el resultado a ij; Asigna ij(1) a i; Asigna ij(1) a j;
        PRINT "Coeficiente del virial B(i,j)"
        PRINT B_stv(k)
      END FOR
      INPUT número entre 1 y 2 y asigna el resultado a ind
    END WHILE

END FUNCTION
```

```
FUNCTION B_tab(n, name, opi, opB, catalog)

    IF opB ==1
      Asigna "La temperatura crítica (tc) en K" a cad(1)
      Asigna "La presión crítica (pc) en bar" a cad(2)
      Asigna "El volumen crítico (vc) en cm^3/mol" a cad(3)
      Asigna "El factor de compresibilidad crítico (zc)" a cad(4)
      Asigna "El factor acéntrico (ω)" a cad(5)
      Asigna "(Tc K)" a cad(1)
      Asigna "(Pc bar)" a cad(2)
      Asigna "(Vc en cm^3/mol)" a cad(3)
      Asigna "?" a cad(4)
      Asigna "?" a cad(5)
```

```
      FOR 1 < i < n
         Asigna catalog(i) a catalog_i
         Asigna catalog_i(3) a bec

         IF LENGTH(bec) == 2
            Asigna bec(1) a bk; Asigna bec(2) a eco
            IF bk == 1
               IF eco == 1
                  Asigna catalog_i(4) a cpre
                  Asigna cpre(2) a tc; Asigna cpre(3) a pc
                  Agrupa a tc y pc y asigna el resultado a tpc
                  Evalúa ingreso_constantesB(cad,cad2,name(i),tpc) y asigna el resultado
                  a prBt
               END IF
            END IF
         ELSE
            Asigna 0 a tpc
            Evalúa ingreso_constantesB(cad,cad2,name(i),tpc) y asigna el resultado a
            prBt
         END IF
         Asigna prB(i,:) a prBt
      END FOR
       Asigna prB(:,:) a B

   ELSEIF opB == 2
      IF opi == 1
         Asigna "B" a car
         Evalúa mat_carB(n,car,opi)
         Evalúa mat_valB(n,car,opi) y asigna el resultado a B
      ELSEIF opi == 2
         Asigna "B" a car
         Evalúa mat_carB(n,car,opi)
         Evalúa mat_valB(n,car,opi) y asigna el resultado a B
      END IF
   END
   Devuelve B

END FUNCTION
```

```
FUNCTION opc(op_gid,n,names,pathDir,p,t,x,imp,param_sr)

   Utilizando pathDir, establece el directorio de coac y asigna el resultado a dir1
   Utilizando a dir1, llama al espacio de trabajo a coac, desde coac
   Utilizando pathDir, establece el directorio de cpre_ing y asigna el resultado a dir2_1
```

Utilizando a **dir2_1**, llama al espacio de trabajo a **cpre_ing**, desde cpre_ing

Asigna 2 a **ind**
WHILE ind == 2
  FOR 1 < **i** < **n**
    PRINT "Libro a ocupar en la búsqueda de las constantes para el cálculo de las presiones de vapor, del compuesto **names**(**i**)"
    INPUT número entre 1, 2, 3 ó 4 y asigna el resultado a **bk**(**i**)
  END FOR

  PRINT "Opción correcta para la búsqueda de las constantes para el cálculo de las presiones de vapor"
  INPUT número entre 1 y 2 y asigna el resultado a **ind**
END WHILE
Crea una lista vacía y asigna el resultado a **catalog**

FOR 1 < **i** < **n**
  PRINT "Ingreso de constantes para el compuesto **names**(**i**), opción **bk**(**i**)"
  Evalúa **cpre_ing(pathDir,p,t,names(i),bk(i))** y asigna el resultado a **catalog**(**i**)
END FOR

IF **op_gid** ~= 0
  PRINT "método para el cálculo de los coeficientes de actividad"
  INPUT número entre 1, 2 o 3 y asigna el resultado a **opa**

  IF **opa** ~= 3
    Evalúa **in_dat_cof(opa,n,names)** y asigna el resultado a **prop**
  ELSEIF **opa** == 3
    Agrupa **pathDir** y **n** y asigna el resultado a **dirnum**
    Evalúa **in_dat_cof(opa,dirnum,names)** y asigna el resultado a **prop**
  END IF
ELSE
  Asigna 0 a **opa**; Asigna 0 a **prop**
END IF

Utilizando **pathDir**, establece el directorio de BurbT y asigna el resultado **dir1_1**
IF **op_gid** == 1
  Asigna 0 a **param**
ELSEIF **op_gid** == 2
  Asigna **param_sr**(1) a **opi**; Asigna **param_sr** a **opB**
  Evalúa **B_tab(n,names,opi,opB,catalog)** a **B**; Agrupa **B**, **opi**, **opB** y asigna el resultado a
  param
END IF

yBurbt(x,p,catalog,opa,prop,op_gid,pathDir,n,imp,param) y asigna el resultado a **cat1**

END FUNCTION

imp_tab

FUNCTION **phitab_imp(n, car1, car3)**

Crea una cadena de caracteres vacia y la asigna a **st4**
Crea una cadena de caracteres vacia y la asigna a **st3**
Crea una lista vacia y la asigna a **st1**
FOR 1 < **i** < **n** + 2
Utiliza **car1** e **i** para construir parte de una cadena de caracteres a imprimir y la asigna a **st1**(**i**)

Une **st1**(**i**) y **st3** en una cadena de caracteres a imprimir y asigna el resultado a **st3**
END FOR

Utiliza **car3** en una cadena de caracteres a imprimir y la asigna a **st4**
Agrupa **st3** y **st4** en una lista y devuelve dicha lista

END FUNCTION

FUNCTION **coactab_imp(n, car1, car3)**

Crea una cadena de caracteres vacia y la asigna a **st4**
Crea una cadena de caracteres vacia y la asigna a **st3**
Crea una lista vacia y la asigna a **st1**

FOR 1 < **i** < **n** + 2
Utiliza **car1** e **i** para construir parte de una cadena de caracteres a imprimir y la asigna a **st1**(**i**)
Une **st1**(**i**) y **st3** en una cadena de caracteres a imprimir y asigna el resultado a **st3**
END FOR

Utiliza **car3** en una cadena de caracteres a imprimir y la asigna a **st4**
Agrupa **st3** y **st4** en una lista y devuelve dicha lista

END FUNCTION

FUNCTION **pstab_imp(n, car1, car3)**

Crea una cadena de caracteres vacia y la asigna a **st4**
Crea una cadena de caracteres vacia y la asigna a **st3**
Crea una lista vacia y la asigna a **st1**

FOR 1 < **i** < **n** + 2
Utiliza **car1** e **i** para construir parte de una cadena de caracteres a imprimir y la asigna a **st1**(**i**)
Une **st1**(**i**) y **st3** en una cadena de caracteres a imprimir y asigna el resultado a **st3**
END FOR

Utiliza **car3** en una cadena de caracteres a imprimir y la asigna a **st4**
Agrupa **st3** y **st4** en una lista y devuelve dicha lista

END FUNCTION

FUNCTION **comptab_imp(n, car1, car2 ,car3)**

Crea una cadena de caracteres vacia y la asigna a **st4**
Crea una cadena de caracteres vacia y la asigna a **st3**
Crea una lista vacia y la asigna a **st1**

FOR 1 < **i** < **n** + 2
Utiliza **car1** y **car2** e **i** para construir parte de una cadena de caracteres a imprimir y la asigna a **st1**(**i**)
Une **st1**(**i**) y **st3** en una cadena de caracteres a imprimir y asigna el resultado a **st3**
END FOR

Utiliza **car3** en una cadena de caracteres a imprimir y la asigna a **st4**
Agrupa **st3** y **st4** en una lista y devuelve dicha lista

END FUNCTION

imp_op_tab

FUNCTION **imp_op_tab(coetf_g2, n, car1, car2, car3, pathDir, imp, nfile, name1)**

IF **LENGTH(imp)** ==2
Asigna **imp**(1) a **op_imp**; Asigna **imp**(2) a **datn**
ELSE
Asigna **imp**(1) a **op_imp**

END IF

Utilizando **pathDir**, establece la ruta del directorio de imp_tab y asigna el resultado a **dir1**
Utilizando **dir1**, llama al espacio de trabajo a **phitab_imp**, **coactab_imp**, **pstab_imp** y **comptab_imp** desde **imp_tab**
Evalua **comptab_imp(n**, **car1**, **car2, car3)**, y asigna el resultado a **tab1**
Evalua **pstab_imp(n**, **car1, car3)**, y asigna el resultado a **tab2**
Evalua **coactab_imp(n**, **car1**, **car3)**, y asigna el resultado a **tab3**
Evalua **phitab_imp(n**, **car1**, **car3)**, y asigna el resultado a **tab4**

Guarda **tab1**(1) y asigna el resultado a **st3c**; Guarda **tab2**(1) y asigna el resultado a **st3p**
Guarda **tab3**(1) y asigna el resultado a **st3a**; Guarda **tab4**(1) y asigna el resultado a **st3f**
Guarda **tab1**(2) y asigna el resultado a **st4c**; Guarda **tab2**(2) y asigna el resultado a **st4p**
Guarda **tab3**(2) y asigna el resultado a **st4a**; Guarda **tab4**(2) y asigna el resultado a **st4f**
Agrupa **st3c**, **st3p**, **st3a** y **st3f** en una lista y la asigna a **st3_g**
Agrupa **st4c**, **st4p**, **st4a** y **st4f** en una lista y la asigna a **st4_g**

IF **op_imp** ==1
  PRINT "Tablas de cálculo del punto de **name1**"

  PRINT **st3c**; PRINT **st4c**; PRINT **coetf_g2**(1);
  PRINT **st3p**; PRINT **st4p**; PRINT **coetf_g2**(2)
  PRINT **st3a**; PRINT **st3a**; PRINT **coetf_g2**(3)
  PRINT **st3f**; PRINT **st3f**; PRINT **coetf_g2**(4)

ELSEIF **op_imp** ==2
  Utilizando **pathDir**, crea o utiliza la carpeta "Archivos_usuario" y guarda el resultado en **dir1**
  Utiliza **dir1**, **datn** y **nfile** para formar una ruta especifica según **nfile**, y guarda el resultado en **dir2**
  Utilizando **dir2**, crea una carpeta
  Agrupa "comp", "Psat", "coet_act" y "Phi" y la guarda en **cad**
  FOR 1 < **i** < 4
    Utiliza **cad**(**i**) y **datn** para elaborar parte del nombre de un archivo .csv y lo guarda en **name2**
    Utiliza **dir2** y **name2** para formar la ruta completa del archivo csv donde se guardaran los datos y lo almacena en **filename**
    Utiliza **tabB**(**i**) y de **tabC**(**i**) y **coetf_g2**(i) para crear títulos y escribir los datos de **coetf_g2**(**i**) en el archivo **filename**

```
    END FOR
  END IF
  PRINT "los archivos del punto de name1 están guardados en:"
  PRINT dir2

END FUNCTION
```

calc_ELV

```
FUNCTION names(n,st)

  FOR 1 < i < n
    INPUT nombre del compuesto st y asigna el resultado a idn(i)
  END FOR
  Devuelve idn

END FUNCTION
```

```
FUNCTION cad_comp_imp(car,n,idn)

  Crea un vector vacío y asigna el resultado a st1;
  Crea un vector vacío y asigna el resultado a st2
  Crea una cadena de caracteres vacía y asigna el resultado a st3
  Crea una cadena de caracteres vacía y asigna el resultado a st4

  FOR 1 < i < n
    Utiliza car e i para construir parte de una cadena de caracteres a imprimir y la
    asigna st1(i)
    Utiliza car e i para construir parte de una cadena de caracteres a evaluar y la asigna
    al st2(i)
    Une st3 y st1(i) en una cadena de caracteres a imprimir y asigna el resultado a st3
    Une st4 y st2(i) en una cadena de caracteres a evaluar y asigna el resultado a st4
  END FOR

END FUNCTION
```

```
FUNCTION com_ing(n, name, car, sf_aux1, sf_aux2, sf_aux3)

  Crea una cadena de caracteres vacia y la asigna a st2
  Crea un vector vacío y lo asigna a st

  FOR 1 < i < n
```

```
      Utiliza car e i para construir parte de una cadena de caracteres a imprimir y la
      asigna a st(i)
      Une st(i) y st2 en una cadena de caracteres a imprimir y asigna el resultado a st2
   END FOR
   Realiza STRSPLIT(st2, " ") y asigna el resultado a st2
   IF n == 2
      PRINT sf_aux1(1); PRINT sf_aux1(2);
   ELSEIF n == 3
      PRINT sf_aux1(1); PRINT sf_aux1(2); PRINT sf_aux1(3);
   ELSEIF n == 4
      PRINT sf_aux1(1); PRINT sf_aux1(2); PRINT sf_aux1(3); PRINT sf_aux1(4)
   END IF

   Asigna 1 a ind
   WHILE ind == 1
      FOR 1 < i < n-1
         INPUT propiedad correspondiente a st2(i) y asigna el resultado a c_st(i)
      END FOR

      Calcula 1 – SUM(c_st) y asigna el resultado a c_st
      IF c_comp > 0
         Asigna 2 a ind
      ELSEIF c_comp <= 0
         Asigna 1 a ind
         PRINT sf_aux3
      END IF
   END WHILE

   Une c_st y c_comp y asigna el resultado a c_st
   Asigna c_st(:) y asigna el resultado a c(:)
   Devuelve c

END FUNCTION
```

```
FUNCTION Calc_ELV(r_prin)

   Asigna 2 a ind
   WHILE ind == 2
      PRINT "Selección del número de componentes: 2, 3 o 4"
      INPUT número entre 2, 3 o 4 y asigna el resultado a opC
      PRINT "Selección de la ecuación para el cálculo ELV"
      INPUT número entre 1 o 2 y asigna el resultado a op_gid
      PRINT "Ingreso de los nombres de los compuestos"
      Asigna "compuesto" a st
```

```
   Evalúa names(opC,st) y asigna el resultado a nameco
   IF op_gid ==2
      PRINT "Cuenta con coeficientes del virial? 1. Si  2. No"
      INPUT número entre 1 y 2 y asigna el resultado a opB
      IF opB == 1
         Asigna 2 a opB
      ELSEIF opB == 2
         Asigna 1 a opB
      END IF
      PRINT "fase vapor en estado ideal? 1. Si  2. No"
      INPUT número entre 1 y 2 y asigna el resultado a opi
   ELSE
      Asigna 1 a opB; Asigna 1 a opi
   END IF

    PRINT "número y nombres de componentes, ecuación a utilizar e información
           ingresada de lo coeficientes del virial, correcta?"
    INPUT número entre 1 y 2 y asigna el resultado a ind
END WHILE

Asigna 2 a ind
WHILE ind == 2
   INPUT Presión en Kpa y asigna el resultado a p
   WHILE %T
      IF p < 0
         INPUT Presión en Kpa y asigna el resultado a p
      ELSE
         Sale del ciclo
      END IF
   END WHILE

   INPUT Temperatura en K y asigna el resultado a t
   WHILE %T
      IF t < 0
         INPUT Presión en Kpa y asigna el resultado a t
      ELSE
         Sale del ciclo
      END IF
   END WHILE

   Utilizando r_prin, establece el directorio de opc y asigna el resultado a r1
   Utilizando r1, llama al espacio de trabajo a opc desde opc

   Asigna "x" a car
```

IF **n** ==2
Asigna “Ingrese la composición molar **car** del **nameco**(1)” a **sf_aux1**
ELSEIF **n** ==3
Asigna “Ingrese la composición molar **car** del **nameco**(1) y **nameco**(2)” a **sf_aux1**
ELSEIF **n** ==4
Asigna “Ingrese la composición molar **car** del **nameco**(1), **nameco**(2) y **nameco**(3)” a **sf_aux1**
END IF

Asigna “Ingrese una cantidad numérica positiva” y asigna el resultado a **sf_aux2**
Asigna “La suma de las composiciones **car** debe ser igual a 1” y asigna el resultado a **sf_aux3**
Evalúa **com_ing(n,nameco,car,sf_aux1,sf_aux2,sf_aux3)** y asigna el resultado a **x**
PRINT “Presión, temperatura y composiciones correctas? 1. Si 2. No”
INPUT número entre 1 y 2 y asigna el resultado a **ind**
END WHILE

PRINT “imprimir los resultados iterativos del cálculo de punto de burbuja”
PRINT “1. Consola de Scilab 2. Archivo .csv 3. Últimos cálculos realizados”
INPUT número entre 1, 2 o 3 y asigna el resultado a **op_imp**

IF **op_imp** == 2
Calcula **INT(RAND())***10000 y asigna el resultado a **nw**
ELSE
Coloca a **op_imp** en una lista y asigna el resultado a **imp**
END IF
IF **op_gid** == 0 **OR op_gid** == 1
Asigna 0 a **param**
Evalúa **opc(op_gid,n,nameco,r_prin,p,t,x,imp,param)** y asigna el resultado a **ct**
ELSEIF **op_gid** ==2
Agrupa a **opi** y **opB** y asigna el resultado a **param**
Evalúa **opc(op_gid,n,nameco,r_prin,p,t,x,imp,param)** y asigna el resultado a **ct**
END IF

IF op_imp == 1
FOR 1 < **i** < **n**
PRINT “el componente **i** es **nameco**(**i**)”
END FOR
END IF

Asigna **ct**(3) a **t**; Asigna **ct**(5) a **y**
PRINT “Temperatura del punto de Burbuja T”; PRINT **t**

Asigna “y” a **car**; Evalúa cad_comp_imp(car,n,nameco) y asigna el resultado a una lista formada por **stycar** y **styimp**
PRINT “Composiciones y son: ”;
PRINT **stycar** y **EVSTR(styimp)**

END FUNCTION

FUNCTION **rep(r_prin)**

Asigna 1 a **flag**
WHILE **flag** == 1
Evalúa **Calc_ELV(r_prin)**
PRINT “continuación de los cálculos de composiciones de equilibrio? 1 Si 2 2. No”
INPUT número entre 1 y 2 y asigna el resultado a **flag**
END WHILE
END FUNCTION

Establece la ruta del directorio de Calc_ELV y guarda el resultado en **r_prin**

Evalúa **rep(r_prin)**

Composición de equilibrio en reacciones para mezclas de gases ideales (Problema 6°).

CalcDH_DS

FUNCTION **cpH_SVN(t, l1)**

Almacena **l1**(1) y lo asigna a **ccp;** Almacena **l1**(2) y lo asigna a **Rcons**
Almacena **ccp**(1) y lo asigna a **A;** Almacena **ccp**(2) y lo asigna a **B**
Almacena **ccp**(3) y lo asigna a **C;** Almacena **ccp**(4) y lo asigna a **D**
Calcula (**A**+**B*****t**+**C*****t**^2+**D*****t**^-2)***Rcons** y lo asigna a y
Devuelve **y**

END FUNCTION

FUNCTION **cpH_Prausnitz_4ed(t, l1)**

Almacena **l1**(1) y lo asigna a **A;** Almacena **l1**(2) y lo asigna a **B**

Almacena **l1**(3) y lo asigna a **C;** Almacena **l1**(4) y lo asigna a **D**
Calcula **A**+**B*****t**+**C*****t**^2+**D*****t**^3 y lo asigna a **y**
Devuelve **y**

END FUNCTION

FUNCTION **cpH_Prausnitz_5ed(t, l1)**

Almacena **l1**(1) y lo asigna a **ccp;** Almacena **l1**(2) y lo asigna a **Rcons**
Almacena **ccp**(1) y lo asigna a **a0;** Almacena **ccp**(2) y lo asigna a **a1**
Almacena **ccp**(3) y lo asigna a **a2;** Almacena **ccp**(4) y lo asigna a **a3**
Almacena **ccp**(5) y lo asigna a **a4**
Calcula (**a0**+**a1*****t**+**a2*****t**^2+**a3*****t**^3+**a4*****t**^4)***Rcons** y asigna el resultado a **y**
Devuelve **y**

END FUNCTION

FUNCTION **cpH_Perry7_1(t, l1)**

Almacena **l1**(1) y lo asigna a **C1;** Almacena **l1**(2) y lo asigna a **C2**
Almacena **l1**(3) y lo asigna a **C3;** Almacena **l1**(4) y lo asigna a **C4**
Almacena **l1**(5) y lo asigna a **C5**
Calcula (**C1**+**C2***((**C3**/**t**)/SINH(**C3**/**t**))^2+**C4***((**C5**/**t**)/**COSH**(**C5**/**t**))^2)/1000 y asigna el resultado a **y**
Devuelve **y**

END FUNCTION

FUNCTION **cpH_Perry7_2(t, l1)**

Almacena **l1**(1) y lo asigna a **C1;** Almacena **l1**(2) y lo asigna a **C2**
Almacena **l1**(3) y lo asigna a **C3;** Almacena **l1**(4) y lo asigna a **C4**
Almacena **l1**(5) y lo asigna a **C5**
Calcula (**C1**+**C2*****t**+**C3*****t**^2+**C4*****t**^3+**C5*****t**^4)/1000 y asigna el resultado a **y**
Devuelve **y**

END FUNCTION

FUNCTION **cpH_Perry7_3(t, l1)**

Almacena **l1**(1) y lo asigna a **C1;** Almacena **l1**(2) y lo asigna a **C2**
Almacena **l1**(3) y lo asigna a **C3;** Almacena **l1**(4) y lo asigna a **C4**

Calcula (**C1**+**C2*****LOG(t)**+**C3**/**t**+**C4*****t**)/1000 y asigna el resultado a **y**
Devuelve **y**

END FUNCTION

FUNCTION **cpH_Perry8_1(t, l1)**

Almacena **l1**(1) y lo asigna a **C1;** Almacena **l1**(2) y lo asigna a **C2**
Almacena **l1**(3) y lo asigna a **C3;** Almacena **l1**(4) y lo asigna a **C4**
Almacena **l1**(5) y lo asigna a **C5**
Calcula (**C1**+**C2*****t**+**C3*****t**^2+**C4*****t**^3+**C5*****t**^4)/1000 y asigna el resultado a **y**
Devuelve **y**

END FUNCTION

FUNCTION **cpH_Perry8_2(t, l1)**

Almacena **l1**(1) y lo asigna a **C1;** Almacena **l1**(2) y lo asigna a **C2**
Almacena **l1**(3) y lo asigna a **C3;** Almacena **l1**(4) y lo asigna a **C4**
Almacena **l1**(5) y lo asigna a **C5**
Calcula (**C1**+**C2***((**C3**/**t**)/**SINH(C3**/**t)**)^2+**C4***((**C5**/**t**)/**COSH**(**C5**/**t**))^2)/1000 y asigna el resultado a **y**
Devuelve **y**

END FUNCTION

FUNCTION **Int_cpH_evalf(t, lf_cpH, numc, pathDir)**

IF **numc** ==1
Evalua **lf_cpH**(1)**(t, lf_cpH**(2)**)** y asigna el resultado a **y**

ELSEIF **numc** > 1
Utilizando **pathDir**, llama al espacio de trabajo a **func_def** y a **func_cad_impH** desde func_cad
Asigna “func” a **car**
Asigna **lf_cpH**(1) a **lf**; Asigna **lf_cpH**(2) a **lc;** Asigna **lf_cpH**(3) a **lv**

Evalua **func_def(car**, **numc)** y asigna el resultado a **vf**
Ejecuta **vf**
Evalua **func_cad_impH(car**, **lv** y **numc)**, y asigna el resultado a **sum_fun**
Une “y = “ y **sum_fun** y el resultado lo asigna a **instr**
Evalua **instr**
END IF

Devuelve **y**

END FUNCTION

FUNCTION **Int_cpS_evalf(t, lf_cpS, numc, pathDir)**

IF **numc** ==1
Evalua **lf_cpS**(1)(**t, lf_cpS**(2)/**t** y asigna el resultado a **y**

ELSEIF **numc** > 1
Utilizando **pathDir**, llama al espacio de trabajo a **func_def** y a **func_cad_impH** desde func_cad
Asigna "func" a **car**
Almacena **lf_cpS**(1) en **lf**; Almacena **lf_cpS**(2) a **lc**
Almacena **lf_cpS**(3) a **lv**
Evalua **func_def(car**, **numc)** y asigna el resultado a **vf**
Ejecuta **vf**
Evalua **func_cad_impH(car**, **lv**, **numc)**, y asigna el resultado a **sum_fun**
Une "y = " y **sum_fun** y el resultado lo asigna a **instr**
Evalua **instr**

END IF
Devuelve **y**

END FUNCTION

cK

FUNCTION **consEq(numc, T, T0, lf_cpH, lf_cpS, pathDir, itemRom, delH0, delG0)**

Utilizando **pathDir**, establece la ruta del directorio de RomberT, y asigna el resultado a **dir2**
Utilizando **dir2**, llama al espacio de trabajo a **Romberg** desde RomberT
Asigna 8.314 a **Rcons**
Agrupa a **Int_cpH_evalf**, **lf_cpH** y **pathDir** y asigna el resultado a **lf_cpH_evalf**
Agrupa a **Int_cpS_evalf**, **lf_cpS** y **pathDir** y asigna el resultado a **lf_cpS_evalf**
Asigna 0.0001 a **tol**
Evalua **Romberg(numc**, **T0**, **T**, **lf_cpH_evalf**, **itemRom**, **tol)**, y asigna el resultado a **IntH**
Calcula **IntH**(1)/**Rcons** y asigna el resultado a **delH**
Evalua **Romberg(numc**, **T0**, **T**, **lf_cpS_evalf**, **itemRom** y **tol),** y asigna el resultado a **IntS**

Calcula **IntH**(2)/**Rcons** y asigna el resultado a **delS**
Calcula **EXP(-delG0**/(**Rcons***T0**))** y asigna el resultado a **K0**
Calcula **EXP(delH0***(1-**T0**/**T**)/(**Rcons***T0**))** y asigna el resultado a **K1**
Calcula **EXP(-delH**/**T**+**delS)** y asigna el resultado a **K2**
Calcula **K0***K1***K2** y asigna el resultado a **K**
Devuelve **K**

END FUNCTION

FUNCTION **comEq(e, mol_ini, lv, K, P)**

Calcula **LENGTH(lv)** y la asigna a **n**
Crea un vector vacio y la asigna a **molv**
FOR 1 < **i** < **n**
Calcula **molv**(**i**)=**mol_ini**(**i**)+**lv**(**i**)***e**
END FOR

Calcula **SUM(molv)** y el resultado lo asigna a **molT**
Crea un vector vacio y la asigna a **compv**; Asigna 1 a **fc**
FOR 1 < **i** < **n**
Calcula **molv**(i)/**molT**, y asigna el resultado a **compv**(**i**)

IF **lv**(**i**) ~= 0
Calcula (**compv**(**i**))^**lv**(i) * **fc**
ELSEIF **lv**(**i**) ==0
Calcula **fc** * 1, y el resultado lo asigna a **fc**
END IF
END FOR

Calcula **SUM(lv)** y lo asigna a **v**

Calcula -**K***(**P**^(-**v**))+**fc** y asigna el resultado a **fun**.
Devuelve **fun**

END FUNCTION

FUNCTION **Rlimit(mol_ini, lv, K)**

Calcula **LENGTH(mol_ini)** y la asigna a **n**
Asigna 0 a **cont**
Crea un vector vacio y la asigna a **e_limit**
Crea un vector vacio y la asigna a **iv**

```
  FOR 1 < i < n
    IF lv(i) > 0
      IF ABS(lv(i)) > 0
        Suma 1 a cont y asigna el resultado a cont
        Calcula (0-mol_ini(i))/(lv(i)) y guarda el resultado en e_limit(cont)
        Guarda i en iv(cont)
      END IF
    END IF
  END FOR

  Asigna 0 a cont
  Calcula LENGTH(e_limit) y lo asigna a lnr
  Realiza ZEROS(lnr,n) y asigna el resultado a mol_rxn
  Crea un vector vacio y lo asigna a e_res
  Crea una vector vacio y lo asigna a iv2

  FOR 1 < i < n
    IF ANY(i == iv)
      Suma 1 a cont y asigna el resultado a cont

      FOR 1 < j < n
        Calcula mol_ini(j)+lv(j)*e_limit(cont) y asigna el resultado a
        mol_rxn(cont,j)
        (cont equivale al numero de reactivos y/o productos y j al numero de especies)
      END FOR

      IF ALL(mol_rxn(cont,:) > 0
        Calcula 1 + cont2 y asigna el resultado a cont2
        Almacena e_limit(cont) en e_res(cont2)
        Almacena cont2 en iv2(cont)
      END IF
    END IF
  END FOR
  Almacena e_res(iv2(1)) en e_max
  Devuelve fun

END FUNCTION
```

defK

```
FUNCTION ing_delHG(op_Kri,T0,v,names)

  Calcula LENGTH(v) y asigna el resultado a n
```

```
   Asigna 2 a ind
   WHILE ind ==2

      FOR 1 < i < n
        IF v(i)~=0
           INPUT calor de formación de  names(i) y asigna el resultado a delHf(i)

           IF op_Kri == 2
              INPUT calor de formación de names(i) y asigna el resultado a delGf(i)
           END IF

        END IF
      END FOR

      PRINT "calores de formación y energías de Gibbs de formación correctos? 1. Si  2.
               No"
      INPUT número entre 1 y 2, y asigna el resultado a ind
   ENDWHILE

   Asigna 0 a delHrxn y a delGrxn
   FOR 1 < i < n
      IF v(i)~=0
         Calcula delHrxn0 + v(i)*delHf(i) y asigna el resultado a delHrxn0

         IF op_Kri == 2
           Calcula delGrxn0 + v(i)*delGf(i) y asigna el resultado a delGrxn0
         END IF
      END IF
   END FOR
   Agrupa delHrxn0 y delGrxn0 y asigna el resultado a delHGrxn0
   Devuelve delHGrxn0

ENDFUNCTION
```

```
FUNCTION defConsEq(datc, T, P, v, pathDir)

   Calcula LENGTH(v) y asigna el resultado a n
   PRINT "disponibilidad de la constante de equilibrio: Si (1) No(2)?"
   INPUT número entre 1 y 2 y asigna el resultado a op_vK

   IF op_vK == 1
      INPUT constante de equilibrio y asigna el resultado a K
      WHILE %T
         IF K > 0
```

```
        Sale del ciclo
      ELSE
        INPUT constante de equilibrio y asigna el resultado a K
      END IF
    END WHILE
  ELSEIF op_vK == 2
    PRINT "cambio de entalpía, función independiente de la temperatura? 1. Si  2. No"
    INPUT número entre 1 y 2 y asigna el resultado a op_Kri

    IF op_Kri == 1
      INPUT temperatura de referencia T0 y asigna el resultado a T0
      WHILE %T
        IF T0 > 0
          Sale del ciclo
        ELSE
          INPUT temperatura de referencia T0 y asigna el resultado a T0
        END IF
      END WHILE

      INPUT constante de equilibrio a la temperatura de referencia T0 y asigna el
            resultado a K0
      WHILE %T
        IF K0 > 0
          Sale del ciclo
        ELSE
          INPUT constante de equilibrio a la temperatura de referencia T0 y asigna el
                resultado a K0
        END IF
      END WHILE

      INPUT cambio de entalpía de reacción a la temperatura de referencia T0 y asigna
            el resultado a delHrxn0

      Evalúa ing_delHG(op_Kri,T0,v,names) y asigna el resultado a delHGrxn0
      Asigna delHGrxn0(1) a delHrxn0
      Asigna 8.314 a Rcons
      Calcula K0*EXP(-(delHrxn0/Rcons)*(1/T-1/T0)) y asigna el resultado a K

    ELSEIF op_Kri == 2
      Asigna 2 a ind
      WHILE ind ==2
        Utilizando pathDir, establece la ruta del directorio de ccp_ing y lo asigna a
        ring
        Utilizando ring, llama al espacio de trabajo a ccp_ing, desde ccp_ing
```

```
   Establece la ruta del directorio de cK y lo asigna a rcK
   Utilizando rcK, llama al espacio de trabajo a consEq, desde cK

   INPUT temperatura de referencia T0 y asigna el resulato a T0
   WHILE %T
      IF T0 > 0
         Sale del ciclo
      ELSE
         INPUT temperatura de referencia T0 y asigna el resulato a T0
      END IF
   END WHILE

   INPUT cambio de entalpía de reacción a T0 y asigna el resultado a delHrxn0
   Evalúa ing_delHG(op_Kri,T0,v,names) y asigna el resultado a delHGrxn0
   Asigna delHGrxn0(1) a delHrxn0; Asigna delHGrxn0(2) a delGrxn0;
   Asigna 8.314 a Rcons
   Asigna datc(1) a numR; Asigna datc(2) a numP; Asigna datc(3) a numI
   Calcula numR + numP y asigna el resultado a numP2
   Calcula numP2 + numI y asigna el resultado a numI2
   Crea una lista vacia y la asigna a bk
   PRINT "libro a utilizar para el cálculo de las capacidades caloríficas …"

   FOR 1 < i < n
      if v(i)~=0
         PRINT "compuesto names(i)? "
         INPUT número entre 1, 2, 3, 4 o 5 y asigna el resultado a bk(i)
   END FOR

   PRINT "temperatura de referencia T0 y libros a usar correctos?"
   INPUT número entre 1 y 2 y asigna el resultado a ind
END WHILE

Crea una lista vacía y la asigna a catalog

FOR 1 < i < n
   IF v(i) ~= 0
      PRINT "constantes para el cálculo del cp del compueto names(i)"
      Evalúa ccp_ing(pathDir,names(i),bk(i),Rcons) y asigna el resultado a
      catalog(i)
   END IF
END FOR

Crea una lista vacía y asigna el resultado a vn
FOR 1 < i < n
```

```
            IF v(i) ~= 0
               Almacena v(i) en vn(i)
            ENDIF
         END FOR

         Calcula LENGTH(vn) y asigna el resultado a nw
         Crea una lista vacia y la asigna a lc; Crea una lista vacía y la asigna a lf

         FOR 1 < i < nw
            Guarda catalog(i) en compt; Guarda compt(1) en valc; Guarda compt(2) en
            valf
            IF LENGTH(compt) == 3
               Agrupa a valc y Rcons y guarda el resultado en valc
            END IF
            Almacena valc en lc(i); Almacena valf en lf(i)
         END FOR

         Agrupa lf, lc y vn en lf_cpH; Agrupa lf, lc y vn en lf_cpS
         Asigna 10 a itemRom
         Evalúa consEq(nw,T,T0,lf_cpH,lf_cpS,pathDir,itemRom,delHrxn0,
         delGrxn0) y asigna el resultado a K
      END IF
   END IF
   Devuelve a K

END FUNCTION
```

ccp_ing

```
FUNCTION tab_c(bk, eco)

   IF bk == 1
      Agrupa los caracteres “A”, “B”, “C” y “D” y asigna el resultado a car
   ELSEIF bk ==2
      Agrupa los caracteres "a0","a1","a2","a3" y"a4" y asigna el resultado a car

   ELSEIF bk ==3
      IF eco == 1
         Agrupa los caracteres "C1","C2","C3","C4" y"C5" y asigna el resultado a car
      ELSEIF eco ==2
         Agrupa los caracteres "C1","C2","C3","C4" y"C5" y asigna el resultado a car
      ELSEIF eco ==3
         Agrupa los caracteres "C1","C2","C3" y "C4" y asigna el resultado a car
```

```
    END IF
  ELSEIF bk ==4
    IF eco == 1
      Agrupa los caracteres "C1","C2","C3","C4" y"C5" y asigna el resultado a car
    ELSEIF eco ==2
      Agrupa los caracteres "C1","C2","C3","C4" y"C5" y asigna el resultado a car
    END IF

  ELSEIF bk ==5
      Agrupa los caracteres "A","B","C" y"D" y asigna el resultado a car
  END IF
  Devuelve fun

END FUNCTION
```

```
FUNCTION ingreso_constantes(bk,eco,st1)

  Evalúa tab_c(bk, eco) y asigna el resultado a car
  Calcula LENGTH(car) y asigna el resultado a n
  Asigna 2 a ind

  WHILE ind == 2
    FOR 1 < i < n
      INPUT valor numérico correspondiente a car(i) y asigna el resultado a ccp(i)
    END FOR

    PRINT "Constantes para el cálculo de las capacidades caloríficas correctas? 1. Si  2.
          No"
    INPUT número entre 1 y 2 y asigna el resultado a ind
  END WHILE
  Devuelve ccp

END FUNCTION
```

```
FUNCTION ccp_ing(pathDir, st1, bk, Rcons)

  Establece la ruta del directorio de CalcDH_DS.sci y lo guarda en dir1
  Utilizando dir1 llama al espacio de trabajo a CalcDH_DS

  IF bk ==3
    PRINT "tipo de ecuación de la capacidad calorífica?"

    INPUT número entre 1, 2 y 3 y asigna el resultado en eco
```

```
    WHILE %T
      IF eco ~=[1, 2, 3]
        PRINT "tipo de ecuación de la capacidad calorífica?"
        INPUT número entre 1, 2 y 3 y asigna el resultado en eco
      ELSE
        Sale del ciclo
      END IF

  ELSEIF bk ==4
    PRINT "tipo de ecuación de la capacidad calorífica?"

    INPUT número entre 1, 2 y 3 y asigna el resultado en eco
    WHILE %T
      IF eco ~=[1, 2]
        PRINT "tipo de ecuación de la capacidad calorífica?"
        INPUT número entre 1, 2 y 3 y asigna el resultado en eco
      ELSE
        Sale del ciclo
      END IF
    END WHILE
  END IF

  IF bk ==1
    Asigna 0 a eco
    Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a ccp
    Agrupa a ccp y cpH_Prausnitz_4ed y asigna el resultado a catg1

  ELSEIF bk ==2
    Asigna 0 a eco
    Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a ccp
    Agrupa a ccp y cpH_Prausnitz_5ed y asigna el resultado a catg1

  ELSEIF bk ==3
    IF eco ==1
      Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a ccp
      Agrupa a ccp y cpH_Perry7_1 y asigna el resultado a catg1

    ELSEIF eco ==2
      Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a ccp
      Agrupa a ccp y cpH_Perry7_2 y asigna el resultado a catg1

    ELSEIF eco ==3
      Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a ccp
      Agrupa a ccp y cpH_Perry7_3 y asigna el resultado a catg1
```

```
      END IF
   ELSEIF bk ==4
      IF eco ==1
         Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a ccp
         Agrupa a ccp y cpH_Perry8_1 y asigna el resultado a catg1

      ELSEIF eco ==2
         Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a ccp
         Agrupa a ccp y cpH_Perry8_2 y asigna el resultado a catg1
     END IF

   ELSEIF bk ==5
      Asigna 0 a eco
      Evalúa ingreso_constantes(bk,eco,st1) y asigna el resultado a ccp
      Agrupa a ccp, cpH_SVN y Rcons y asigna el resultado a catg1
   END IF

   Devuelve catg1

END FUNCTION
```

func_cad

```
FUNCTION lab_imp(name,car)

   Calcula LENGTH(name) y asigna el resultado a ln
   FOR 1 < i < ln
      Une car y name(i) y asigna el resultado a lcar(i)
   END FOR
   Devuelve lcar

END FUNCTION
```

```
FUNCTION names(n,st)

   FOR 1 < i < n
      INPUT nombre del compuesto st y asigna el resultado a idn(i)
   END FOR
   Devuelve idn

END FUNCTION
```

FUNCTION **cad_comp_imp(car, car2, car3, n)**

Evalúa **lab_imp(car2,car3)** y asigna el resultado a **car2**
Crea un vector vacio y la asigna a **st1**
Crea un vector vacio y la asigna a **st2**
Crea una cadena de caracteres vacia y la asigna a **st3**
Crea una cadena de caracteres vacia y la asigna a **st4**

FOR 1 < **i** < **n**
Utiliza **car2**(**i**) para construir parte de una cadena de caracteres a imprimir y la asigna **st1**(**i**)

Utiliza **i** y **car** para construir parte de una cadena de caracteres a evaluar y la asigna al **st2**(**i**)

Une **st3** y **st1**(**i**) en una cadena de caracteres a imprimir y asigna el resultado a **st3**
Une **st4** y **st2**(**i**) en una cadena de caracteres a evaluar y asigna el resultado a **st4**
END FOR
Agrupa a **st3** y **st4** y devuelve el resultado

END FUNCTION
(En esta funcion se toman en cuenta la posibilidad de que **n** sea igual o mayor de 2 elementos, o que **i** se encuentre al principio, medio o final de la cadena de caracteres final)

FUNCTION **com_ing(n, car, car2, sf_aux1, sf_aux2, sf_auxextra)**

Crea una cadena de caracteres vacia y la asigna a **st2**
FOR **i** se encuentra entre 1 y **n**
Utiliza **car** y **car2**(**i**) para construir parte de una cadena de caracteres a imprimir y la asigna al **st**(**i**)
Une **st**(**i**) y **st2** en una cadena de caracteres a imprimir y asigna el resultado a **st2**
END FOR

PRINT **sf_aux1**
IF **LENGTH(sf_auxextra)** ~= 0
FOR 1 < **i** < **LENGTH(sf_auxextra)**
PRINT **sf_auxextra**(**i**)
END FOR
END IF
FOR 1 < **i** < **n**

```
      PRINT st2(i)

      INPUT valores numéricos y asigna el resultado c(i)
      WHILE %T
         IF c(i) < 0
            PRINT sf_aux2
            PRINT sf_aux1
            PRINT st2(i)
            INPUT valores numéricos y asigna el resultado c(i)
          ELSE
            Sale del ciclo
          END IF
      END WHILE
   END FOR
   Devuelve c

END FUNCTION
```

```
FUNCTION func_def(car, n)

   Crea una cadena de caracteres vacia y la asigna a st4
   FOR 1 < i < n
      Utiliza car e i para construir parte de una cadena de caracteres a imprimir y la
      asigna a st2(i)
      Une st2(i) y st4 en una cadena de caracteres a evaluar y asigna el resultado a st4
   END FOR
   Devuelve st4

END FUNCTION
```

```
FUNCTION func_cad_impH(car, lv, n)

   Crea una cadena de caracteres vacia y la asigna a st4

   FOR 1 < i < n
      Utiliza car, lv e i para construir parte de una cadena de caracteres a evaluar y la
      asigna a st2(i)
      Une st2(i) y st4 en una cadena de caracteres a evaluar y asigna el resultado a st4
   END FOR
   Devuelve st4

END FUNCTION
```

FUNCTION **func_cad_impS(car, lv, n)**

Crea una cadena de caracteres vacia y la asigna a **st4**
Crea una lista vacia y la asigna a **st2**

FOR 1 < **i** < **n**
Utiliza **car**, **lv** e **i** para construir parte de una cadena de caracteres a evaluar y la asigna a **st2**(**i**)
Une **st2**(**i**) y **st4** en una cadena de caracteres a evaluar y asigna el resultado a **st4**
END FOR
Devuelve **st4**

END FUNCTION

Calc_EqC

FUNCTION **Calc_EqC(r_prin)**

Establece la ruta del directorio de cK y guarda el resultado en **dir1**
Utilizando a **dir1**, llama al espacio de trabajo a **defConsEq**, desde cK
Establece la ruta del directorio de func_cad y guarda el resultado en **dir2**
Utilizando a **dir2**, llama al espacio de trabajo a **names**, **com_ing** y **cad_comp_imp**, desde func_cad

Establece la ruta del directorio de defK y guarda el resultado en **dir3**
Utilizando a **dir3**, llama al espacio de trabajo a **defConsEq, comEq** y **Rlimit**, desde defK
Establece la ruta del directorio de rootRegulaFA_mod y guarda el resultado en **dir4**
Utilizando a **dir4**, llama al espacio de trabajo a **rootRegulaFA_mod**, desde rootRegulaFA_mod

Asigna 2 a **ind**
WHILE **ind** == 2
INPUT número de reactivos y asigna el resultado a **numR**
WHILE **%T**
IF **numR** < 0
INPUT número de reactivos y asigna el resultado a **numR**
ELSE
Sale del ciclo
END
END WHILE

Asigna “reactivos” a **st**

Evalúa **names(numR,st)** y asigna el resultado a **nameR**
Asigna "**n0_**" a **car**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux1**
Crea una lista vacía y la asigna a **sf_auxextra**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux2**
Evalúa **com_ing(numR,car,nameR,sf_aux1,sf_aux2,sf_auxextra)** y asigna el resultado a **CantR**

Asigna "**v_**" a **car**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux1**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux1B**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux1C**
Agrupa **sf_aux1B** y **sf_aux1C** y asigna el resultado a **sf_auxextra**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux2**
Evalúa **com_ing(numR,car,nameR,sf_aux1,sf_aux2,sf_auxextra)** y asigna el resultado a **vR**
Calcula **vR** * -1 y asigna el resultado a **vR**
PRINT "número de reactivos, nombres, cantidades molares iniciales y números estequiométricos correctos? 1. Si 2. No"
INPUT número entre 1 y 2 y asigna el resultado a **ind**
END WHILE

Asigna 2 a **ind**
WHILE **ind** == 2
INPUT número de productos y asigna el resultado a **numP**
WHILE **%T**
IF **numP** < 0
INPUT número de productos y asigna el resultado a **numP**
ELSE
Sale del ciclo
END
END WHILE

Asigna "productos" a **st**
Evalúa **names(numP,st)** y asigna el resultado a **nameP**
Asigna "**n0_**" a **car**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux1**
Crea una lista vacía y la asigna a **sf_auxextra**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux2**
Evalúa **com_ing(numP,car,nameP,sf_aux1,sf_aux2,sf_auxextra)** y asigna el resultado a **CantP**

Asigna "**v_**" a **car**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux1**

Crea una cadena de caracteres a imprimir y la asigna a **sf_aux1B**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux1C**
Agrupa **sf_aux1B** y **sf_aux1C** y asigna el resultado a **sf_auxextra**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux2**
Evalúa **com_ing(numP,car,nameP,sf_aux1,sf_aux2,sf_auxextra)** y asigna el resultado a **vP**

PRINT "número de productos, nombres, cantidades molares iniciales y números estequiométricos correctos? 1. Si 2. No"
INPUT número entre 1 y 2 y asigna el resultado a **ind**
END WHILE

Une **nameR** y **nameP** y asigna el resultado a **namT**
Une **CantR** y **CantP** y asigna el resultado a **Cant**
Une **vR** y **vP** y asigna el resultado a **v**
PRINT "presencia de compuestos inertes? 1. Si 2. No"
INPUT número entre 1 y 2 y asigna el resultado a **CmI**

Asigna 0 a **numI**
IF **CmI** == 0
WHILE **ind** == 2
INPUT número de compuestos inertes y asigna el resultado **numI**
Asigna "compuesto inerte" a **st**
Evalúa **names(numI,st)** y asigna el resultado a **nameI**

Asigna "**n_**" a **car**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux1**
Crea una lista vacía y la asigna a **sf_auxextra**
Crea una cadena de caracteres a imprimir y la asigna a **sf_aux2**
Evalúa **com_ing(numP,car,nameP,sf_aux1,sf_aux2,sf_auxextra)** y asigna el resultado a **CantIn**

Realiza **ZEROS(LENGTH(numI))** y asigna el resultado a **vI**
Une **namT** y **nameI** y asigna el resultado a **namT**
Une **Cant** y **CantIn** y asigna el resultado a **Cant**
PRINT "número, nombres y cantidades molares de compuestos inertes correctos? 1. Si 2. No"
INPUT número entre 1 y 2 y asigna el resultado a **ind**
END WHILE
END IF

Calcula **LENGTH(Cant)** y asigna el resultado a **n**
Asigna 2 a **ind**
WHILE **ind** == 2

INPUT temperature de reacción y asigna el resultado a **T**
INPUT presión de reacción y asigna el resultado a **P**

PRINT "temperatura y presión correctas? 1. Si 2. No"
INPUT número entre 1 y 2 y asigna el resultado a **ind**
END WHILE

Agrupa **numR**, **numP** y **numI** y asigna el resultaldo a **datc**
Evalúa **defConsEq(datc,T,P,v,r_prin,namT)** y asigna el resultado a **K**
Evalúa **Rlimit(Cant,v,K)** y asigna el resultado a **e0**

Asigna 0.00001 a **tol**; Asigna 100 a **imax**;
Agrupa **Cant**, **v**, **K** y **P** y asigna el resultado a **param**
Asigna 0.0000005 a **h**; Calcula **e0** – **h** y asigna el resultado a **e0**;
Agrupa 0+**h** y **e0** y asigna el resultado **bracket**
Evalúa **rootRegulaFA_mod(e0,comEq,param,tol,imax,store=%T,bracket)** y asigna el resultado a **vm1**
Calcula las dimensiones de **vm1** y asigna el resultado a **s**; Guarda **s**(1) a **fil**
Guarda **vm1**(**fil**,2) en **e_eq**

Realiza **ZEROS(n)** y asigna el resultado a **molv**;
Realiza **ZEROS(n)** y asigna el resultado a **compv**;
FOR 1 < **i** < **n**
Calcula **Cant**(**i**)+**v**(**i**)*__e_eq__ y almacena el resultado en **molv**(**i**)
END FOR
Calcula **SUM**(**molv**) y asigna el resultado a **molT**

FOR 1 < **i** < **n**
Calcula **molv**(**i**)/**molT** y almacena el resultado en **compv**(**i**)
END FOR

Calcula **numR** + **numP** y asigna el resultado a **numP2**
Almacena **molv**(1:**numR**) en **R**; Almacena **molv**(**numR**+1:**numP2**) en **P**
Almacena **compv**(1:**numR**) en **cR**; Almacena **compv**(**numR**+1:**numP2**) en **cP**

PRINT constante de equilibrio
PRINT grado de avance de reacción

Asigna "R" a **car**; Asigna "n_" a **car3**
Evalúa **cad_comp_imp(car,nameR,car3,numR)** y asigna el resultado a una lista formada por **stRcar** y **stReval**
Utiliza **stRcar** para crear una cadena de caracteres y asigna el resultadoa a **stRc**

Asigna "P" a **car**; Asigna "n_" a **car3**

Evalúa **cad_comp_imp(car,nameP,car3,numP)** y asigna el resultado a una lista formada por **stPcar** y **stPeval**
Utiliza **stRcar** para crear una cadena de caracteres y asigna el resultadoa a **stPc**

PRINT “cantidades molares de reactivos: “; PRINT **stRc** y **EVSTR(stReval)**
PRINT “cantidades molares de productos: “; PRINT **stPc** y **EVSTR(stPeval)**

Asigna “cR” a **car**; Asigna “c_” a **car3**
Evalúa **cad_comp_imp(car,nameR,car3,numR)** y asigna el resultado a una lista formada por **stRcar** y **stReval**
Utiliza **stRcar** para crear una cadena de caracteres y asigna el resultadoa a **stRc**

Asigna “cP” a **car**; Asigna “c_” a **car3**
Evalúa **cad_comp_imp(car,nameP,car3,numP)** y asigna el resultado a una lista formada por **stPcar** y **stPeval**
Utiliza **stPcar** para crear una cadena de caracteres y asigna el resultadoa a **stPc**

PRINT “composiciones molares de reactivos: “; PRINT **stRc** y **EVSTR(stReval)**
PRINT “composiciones molares de productos: “; PRINT **stPc** y **EVSTR(stPeval)**

IF **CmI** == 1
  Calcula **numP2** + **numI** y asigna el resultado a **numI2**
  Almacena **molv**(**numP2**+1:**numI2**) en **I**; Almacena **compv**(**numP2**+1:**numI2**) en **cI**
  Asigna “I” a **car**; Asigna “n_” a **car3**
  Evalúa **cad_comp_imp(car,nameI,car3,numI)** y asigna el resultado a una lista formada por **stIcar** y **stIeval**
  Utiliza **stIcar** para crear una cadena de caracteres y asigna el resultadoa a **stIc**

  PRINT “cantidades molares de compuestos inertes: “; PRINT **stIc** y **EVSTR(stIeval)**

  Asigna “cI” a **car**; Asigna “c_” a **car3**
  Evalúa **cad_comp_imp(car,nameI,car3,numI)** y asigna el resultado a una lista formada por **stIcar** y **stIeval**
  Utiliza **stIcar** para crear una cadena de caracteres y asigna el resultadoa a **stIc**

  PRINT “composiciones molares de compuestos inertes: “; PRINT **stIc** y **EVSTR(stIeval)**
END IF

END FUNCTION

FUNCTION **rep(r_prin)**

Asigna 1 a **flag**
WHILE **flag** == 1
Evalúa **Calc_EqC(r_prin)**
PRINT “continuar con los cálculos de ”
PRINT “continuación de los cálculos de composiciones de equilibrio? 1 Si 2 2. No”
INPUT número entre 1 y 2 y asigna el resultado a **flag**
END WHILE

END FUNCTION

Establece la ruta del directorio de Cal_EqC y guarda el resultado en **r_prin**
Evalúa **rep(r_prin)**

C.4 Asignatura: Operaciones Unitarias I

f_fric

FUNCTION **FD_flujo_laminar(Re)**

Calcula 64/**Re**, y asigna el resultado a **fd**
return **fd**

END FUNCTION

FUNCTION **FD_flujoturc_tub_rug_VK(D, rug)**

Calcula 1.14 -0.869 * **LOG**(**rug**/**D**) (ecuación de Von Karmán) y asigna el resultado a **res_par**
Calcula (**res_par****-1)**2 y asigna el resultado a **fd**
Devuelve **fd**

END FUNCTION

FUNCTION **FD_flujotr_Col(fd, rug, Re, D)**

Calcula - 1/(**fd****0.5)-0.869 * **LOG**(**rug**/(3.7***D**)+2.523/(**Re***(**fun****0.5))) y asigna el resultado a **fun**
Devuelve **fun**

END FUNCTION

FUNCTION **Reynolds(v, D, vis_c)**

Calcula **v** * **D**/ **vis_c**, y asigna el resultado a **Re**
Devuelve **Re**

END FUNCTION

FUNCTION **vel(Q, D)**

Calcula 4 * Q/(π * **D**^2), y lo asigna a **v**.
Devuelve **v**

END FUNCTION

FUNCTION **Q_calc(veloc, D)**

Calcula 0.25 * π * **D**^2 * **veloc**, y asigna el resultado a **q**
Devuelve **q**

END FUNCTION

FUNCTION **calc_fdarcy_col(parRe, Q, D, rug, fdir, op)**

Utilizando **fdir**, establece la ruta del directorio de rootSecantePlus2Ap2, y asigna el resultado a **dir_metnum**;
Utilizando **dir_metnum**, llama al espacio de trabajo, a **rooSecantePlus2A** desde rooSecantePlus2Ap2.

IF la **LENGTH(op)** == 2
Asigna **op**(1) a **oph**, y **op**(2) a **opv**

IF **oph** ==2
IF **opv** ==1
Asigna **parRe**(1) a **ro**, y **parRe**(2) a **visd**
Calcula **visd**/**ro**, y asigna el resultado a **vis_c**
ELSEIF **opv** ==2
Asigna **parRe**(1) a **ro**, y **parRe**(2) a **vis_c**
END IF

ELSEIF **oph** ==1
IF **opv** ==1
Asigna **parRe**(1) a **ro**, y **parRe**(2) a **visd**
Calcula **visd**/**ro**, y asigna el resultado a **vis_c**
ELSEIF **opv** ==2
Asigna **parRe** a **vis_c**
END IF
END IF

ELSE
Asigna **op** a **opv**

IF **opv** ==1
Asigna **parRe**(1) a **ro**, y **parRe**(2) a **visd**
Calcula **visd**/**ro**, y asigna el resultado a **vis_c**
ELSEIF **opv** ==2

```
        Asigna parRe a vis_c
      END IF
   END IF

   Evalúa vel(Q, D), y asigna el resultado a veloc.
   Evalúa Reynolds(veloc, D, vis_c), y asigna el resultado a Re.

   IF Re < 2300
      Evalúa FD_flujo_laminar(Re), y el resultado lo asigna a f1D.
      Devuelve f1D

   ELSEIF Re >= 2300
      Evalúa FD_flujoturc_tub_rug_VK(D, rug), y asigna el resultado a f0
      Agrupa rug, Re y D y asigna el resultado a param_fD;
      Asigna 20 a imax

      Evalúa rootSecantePlus2A(f0, FD_flujotr_col, param_fD, tol = 0.001, imax = 20,
      store = %T), y asigna el resultado a vri
      Calcula LENGTH(vri), y asigna el resultado a s1;
      Almacena s1(1) y lo asigna a f1; Almacena vri(f1,2) en f1D
      Devuelve a f1D
   END IF

END FUNCTION
```

Cálculo del diámetro de tubería (Problema 7°)

D_TS_calc

```
FUNCTION D_fun(D, parRe, L, par_calcd, Q, rug, g, oph, dir1)

   Utilizando dir1, establece la ruta del directorio de f_fric, y asigna el resultado
   a dir_fric;
   Utilizando dir_fric, llama al espacio de trabajo a vel y a calc_fdarcy_col desde f_fric

   IF op ~= None
      Asigna op(1) a oph; Asigna op(2) a opv
      IF oph ==2
         IF opv ==1
            Asigna parRe(1) a ro, y parRe(2) a visd
         END IF
         IF opv ==2
```

```
            Asigna parRe(1) a ro, y parRe(2) a vis_c
          END IF

          Asigna par_calcd(1) a p1;
          Asigna par_calcd(2) a p2;
          Asigna par_calcd(3) a z1
          Asigna par_calcd(4) a z2
          Calcula (p1/(ro * g) - p2/(ro * g)) + (z1 - z2) y asigna el resultado a hDar

       ELSEIF oph ==1
          Asigna par_calcd a hDar
       END IF
    END IF
    Evalúa calc_fdarcy_col(parRe, Q, D, rug, dir1, op), y asigna el resultado a f1D.
    Evalúa vel(Q, D), y asignando el resultado a v2
    Calcula -hDar + f1D * (L/valor inicial de D) * (v2^2)/(2*g) y asigna el resultado a
    fun.
    Devuelve fun

END FUNCTION
```

```
FUNCTION calc_DTs(hDar, parRe, L, Q, rug, g, dir1, op)

    Utilizando dir1, establece la ruta del directorio de rootSecantePlus2Ap2, y asigna el
    resultado a dir_metnum;
    Utilizando dir_metnum, llama al espacio de trabajo a rootSecantePlus2A desde
    rootSecantePlus2Ap2
    Asigna hDar a par_calcd
    Asigna 20 a imax
    Agrupa a parRe, L, par_calcd, Q, rug, g, dir1 y op en un vector y asigna el resultado
    a param_hDar

    Asigna 2.54E-2 a D0
    Evalúa rootSecantePlus2A(D0, D_fun, param_hDar, tol = 0.001, imax, store =
     %T), y asigna el resultado a vri

    Devuelve vri

END FUNCTION
```

Calc_D

Establece la ruta del directorio de Calc_D, y asigna el resultado a **dir1**

Asigna 1 a **flag**

WHILE **flag** == 1
  Establece la ruta del directorio de D_TS_calc, y asigna el resultado a **dir2**
  Utilizando a **dir2**, llama al espacio de trabajo a **calc_DTs**, desde D_TS_calc
  INPUT nombre del fluido y asigna el resultado a **nameco**

  Asigna 2 a **ind**
  WHILE **ind** == 2
    INPUT rugosidad de la tubería, y asigna el resultado a **rug**
    WHILE **%T**
      IF **rug** > 0
        Sale del ciclo
      ELSE
        INPUT rugosidad de la tubería, y asigna el resultado a **rug**
      END IF
    END WHILE

    INPUT longitud de la tubería, y asigna el resultado a **L**
    WHILE **%T**
      IF **L** > 0
        Sale del ciclo
      ELSE
        INPUT longitud de la tubería, y asigna el resultado a **L**
      END IF
    END WHILE
    PRINT "rugosidad y longitud de la tubería están correctos? 1. Si 2. No"
    INPUT número entre 1 y 2 y asigna el resultado a **ind**

  ENDWHILE

  PRINT "hay disponibilidad pérdidas del sistema equivalente en metros? 1. Si 2. No"
  INPUT número entre 1 y 2 y asigna el resultado a **oph**

  IF **oph** == 1
    Asigna 2 a **ind**
    WHILE **ind** == 2
      INPUT pérdidas del sistema equivalente en metros y asigna el resultado a **hDar**

```
WHILE %T
   IF hDar > 0
      Sale del ciclo
   ELSE
      INPUT pérdidas del sistema equivalente en metros, y asigna el resultado a
            hDar
   END IF
END WHILE

PRINT "hay disponibilidad de la viscosidad dinámica y de la densidad o de
       viscosidad cinemática? 1. Si  2. No"
INPUT número entre 1 y 2 y asigna el resultado a opv

IF opv == 1
   INPUT densidad del fluido, y asigna el resultado a ro
   WHILE %T
      IF ro > 0
         Sale del ciclo
      ELSE
         INPUT densidad del fluido y asigna el resultado a ro
      END IF
   END WHILE

   INPUT viscosidad dinámica y asigna el resultado a visd
   WHILE %T
      IF visd > 0
         Sale del ciclo
      ELSE
         INPUT viscosidad dinámica y asigna el resultado a visd
      END IF
   END WHILE
   Agrupa a ro y visd, y asigna el resultado a parRe
ELSEIF opv == 2
   INPUT viscosidad cinemática del fluido y asigna el resultado a vis_c
   WHILE %T
      IF vis_c > 0
         Sale del ciclo
      ELSE
         INPUT viscosidad dinámica y asigna el resultado a vis_c
      END IF
   END WHILE
   Asigna vis_c a parRe
END IF
```

```
      PRINT "pérdidas del sistema correctas? 1. Si  2. No"
      IF opv ==1
        PRINT "viscosidad dinámica y densidad correctas"
      ELSEIF opv ==2
        PRINT "viscosidad cinemática correcta"
      ENDIF
      INPUT número entre 1 y 2 y asigna el resultado a ind
    END WHILE

ELSEIF oph ==2
  ind = 2
  WHILE ind == 2
    Crea un vector de 2 columnas y asigna el resultado a p
    PRINT "presión inicial, p(1) y presión final, p(2)? "

    INPUT presión en Kpa para cada columna de p
    WHILE %T
      IF los valores de p > 0
        Sale del ciclo
      ELSE
        PRINT "presión inicial, p(1) y presión final, p(2)? "
        INPUT presión en Kpa para cada columna de p
      END IF
    END WHILE

    Calcula p * 1000, para convertir las presiones en Kpa a Pa y asigna el resultado a
    p
    Asigna el primer elemento de p a p1; Asigna el segundo elemento de p a p2
    Crea un vector de 2 columnas y asigna el resultado a z
    PRINT "nivel de altura inicial, z(1) y nivel de altura final, z(2)? "

    INPUT nivel de altura en metros para cada columna de z
    WHILE %T
      IF los valores de z > 0
        Sale del ciclo
      ELSE
        PRINT "nivel de altura inicial, z(1) y nivel de altura final, z(2)? "
        INPUT nivel de altura en metros para cada columna de z
      END IF
    END WHILE

    Asigna el primer elemento de z a z1; Asigna el segundo elemento de z a z2
    Agrupa a p1, p2, z1, z2 en un vector y asigna el resultado a prop_fl
```

```
INPUT densidad y asigna el resultado a ro
WHILE %T
   IF ro > 0
      Sale del ciclo
   ELSE
      INPUT densidad y asigna el resultado a ro
   END IF
END WHILE

PRINT "hay disponibilidad de la viscosidad dinámica (1) o de viscosidad
         cinemática(2)?"
INPUT número entre 1 y 2 y asigna el resultado a opv

IF opv == 1
   INPUT viscosidad dinámica y asigna el resultado a visd
   WHILE %T
      IF visd > 0
         Sale del ciclo
      ELSE
         INPUT viscosidad dinámica y asigna el resultado a visd
      END IF
   END WHILE
   Agrupa a ro y visd, y asigna el resultado a parRe

ELSEIF opv == 2
   INPUT viscosidad cinemática y asigna el resultado a vis_c
   WHILE %T
      IF vis_c > 0
         Sale del ciclo
      ELSE
         INPUT viscosidad cinemática y asigna el resultado a vis_c
      END IF
   END WHILE
   Agrupa a ro y vis_c y asigna el resultado a parRe
END IF

PRINT "presión inicial (p1), presión final (p2) correctas?"
PRINT "altura inicial (z1), altura final (z2) correctas?"
IF opv ==1
   PRINT "viscosidad dinámica correcta?"
ELSEIF opv ==2
   PRINT "viscosidad cinemática correcta?"
ENDIF
INPUT número entre 1 y 2 y asigna el resultado a ind
```

```
    END WHILE
  ENDIF
  Asigna 9.8 a g

  INPUT caudal del fluido, y asigna el resultado a Q
  WHILE %T
    IF Q > 0
      Sale del ciclo
    ELSE
      INPUT caudal del fluido y asigna el resultado a Q?
    END IF
  END WHILE
  Agrupa a oph y opv y asigna el resultado a op

  Evalúa calc_DTs(prop_fl, parRe, L, Q, rug, g, dir1, op), y asigna el resultado a vri
  Calcula el número de filas y columnas de vri, y asigna el resultado a s1;
  Almacena s1(1) en f1;

  Almacena vri(f1, 2) y asigna el resultado a D
  PRINT "diámetro de tubería: "
  PRINT D
  PRINT "continuación de los cálculos del diámetro de tubería? 1 Si  2 No"
  INPUT número entre 1 y 2 y asigna el resultado a flag
END WHILE
```

Cálculo del flujo volumétrico a través de una tubería (Problema 8°).

Fl_vol_TubS_calc

```
FUNCTION fl_vol_fun(v2, parRe, par_calcd, prop_tub, g, dir1, op)

  Utilizando dir1, establece la ruta del directorio de f_fric, y asigna el resultado
  a dir_fric;
  Utilizando dir_fric, llama al espacio de trabajo a Q_calc y a calc_fdarcy_col desde
  f_fric

  Asigna prop_tub(1) a D; Asigna prop_tub(2) a rug; Asigna prop_tub(3) a L;
  Asigna prop_tub(4) a sumK

  IF LENGTH(op) ==2
    Asigna op(1) a oph; Asigna op(2) a opv
    IF oph ==2
```

```
        IF opv ==1
           Asigna parRe(1) a ro, y parRe(2) a visd
        IF opv ==2
           Asigna parRe(1) a ro
        END

        Asigna prop_fl(1) a p1; Asigna prop_fl(2) a p2; Asigna prop_fl(3) a z1
        Asigna prop_fl(4) a z2; Asigna prop_fl(5) a v1
        Calcula (p1/(ro * g) - p2/(ro * g)) + (z1 - z2) y asigna el resultado a hDar

    ELSEIF oph ==1
       Asigna hDar a prop_fl
    END IF
  END IF

  Evalua Q_calc(v2, D) y asigna el resultado a caudal
  Evalúa calc_fdarcy_col(parRe, caudal, D, rug, dir1, op), y  asigna el resultado a
  f1D.

  Calcula -hDar + f1D * (L/valor inicial de D + sumK) * (v2^2)/(2*g) y asigna el
  resultado a fun.
  Devuelve fun

END FUNCTION
```

```
FUNCTION calc_fl_vol(parRe, prop_fl, prop_tub, g, dir1, op)

  Utilizando dir1, establece la ruta del directorio de rootSecantePlus2Ap2, y asigna el
  resultado a dir_metnum;
  Utilizando dir_metnum, llama al espacio de trabajo a rootSecantePlu2A desde
  rootSecantePlus2Ap2
  Asigna prop_tub(1) a D; Asigna prop_tub(2) a rug;
  Asigna prop_tub(3) a L; Asigna prop_tub(4) a sumK

  IF LENGTH(op) ==2
    Asigna op(1) a oph; Asigna op(2) a opv

    IF oph ==2
      Asigna prop_fl(3) a z1; Asigna prop_fl(4) a z2; Asigna prop_fl(5) a v1
      IF z1 > z2
        Calcula (2 * 9.8*(z1 - z2) + v1^2)^0.5 y asigna el resultado a v20
      END IF

     ELSE IF oph ==1
```

```
        Asigna prop_fl a hDar
        Calcula (2 * 9.8*hDar)^0.5 y asigna el resultado a v20
      END IF
   END IF
   Agrupa a parRe, prop_fl, prop_tub, g, dir1, op y asigna el resultado a
   param_hDar; Asigna 20 a imax

   Evalúa rootSecantePlus2A(v20, fl_vol_fun, param_hDar, tol = 0.001, imax, store =
   %T) y  asigna el resultado a vri
   Devuelve vri

END FUNCTION
```

Calc_fl_vol

```
Establece la ruta del directorio de Calc_fl_vol, y asigna el resultado a dir1

Asigna 1 a flag
WHILE flag == 1
   Utilizando dir1, establece la ruta del directorio de Fl_vol_TS_calc, y asigna el
   resultado a dir2
   Utilizando dir2, llama al espacio de trabajo a calc_fl_vol, desde Fl_vol_TS
   INPUT nombre del fluido y asigna el resultado a nameco

   Asigna 2 a ind
   WHILE ind == 2
      INPUT rugosidad de la tubería, y asigna el resultado a rug
      WHILE %T
         IF rug > 0
            Sale del ciclo
         ELSE
            INPUT rugosidad de la tubería, y asigna el resultado a rug
         END IF
      END WHILE

      INPUT longitud de la tubería, y asigna el resultado a L
      WHILE %T
         IF L > 0
            Sale del ciclo
         ELSE
            INPUT longitud de la tubería, y asigna el resultado a L
         END IF
```

```
   END WHILE

   INPUT suma de los coeficientes K de pérdidas secundarias, y asigna el resultado a
         sumK
   WHILE %T
      IF sumK > 0
         Sale del ciclo
      ELSE
        INPUT suma de los coeficientes K de pérdidas secundarias, y asigna el
              resultado a sumK
      END IF
   END WHILE

   PRINT "rugosidad y longitud de la tubería y suma de los coeficientes K de pérdidas
         secundarias de la tubería están correctos? 1. Si  2. No"
   INPUT número entre 1 y 2 y asigna el resultado a ind
END WHILE

PRINT "hay disponibilidad pérdidas del sistema equivalente en metros? 1. Si  2. No?"
INPUT número entre 1 y 2 y asigna el resultado a oph

IF oph == 1
   Asigna 2 a ind
   WHILE ind == 2
      INPUT pérdidas del sistema equivalente en metros y asigna el resultado a
            prop_fl
      WHILE %T
         IF prop_fl > 0
            Sale del ciclo
         ELSE
            INPUT pérdidas del sistema equivalente en metros, y asigna el resultado a
                  prop_fl
         END IF
      END WHILE

      PRINT "hay disponibilidad de la viscosidad dinámica y de la densidad o de
            viscosidad cinemática? 1. Si  2. No"
      INPUT número entre 1 y 2 y asigna el resultado a opv

      IF opv == 1
         INPUT densidad del fluido, y asigna el resultado a ro
         WHILE %T
            IF ro > 0
               Sale del ciclo
```

```
            ELSE
               INPUT densidad del fluido y asigna el resultado a ro
            END IF
         END WHILE

         INPUT viscosidad dinámica y asigna el resultado a visd
         WHILE %T
            IF visd > 0
               Sale del ciclo
            ELSE
               INPUT viscosidad dinámica y asigna el resultado a visd
            END IF
         END WHILE
         Agrupa en un vector a ro y visd, y asigna el resultado a parRe

      ELSEIF opv == 2
         INPUT viscosidad cinemática del fluido y asigna el resultado a vis_c
         WHILE %T
            IF vis_c > 0
               Sale del ciclo
            ELSE
               INPUT viscosidad cinemática y asigna el resultado a vis_c
            END IF
         END WHILE
         Asigna vis_c a parRe
      END IF

      PRINT "pérdidas del sistema correctas?"

      IF opv ==1
         PRINT "viscosidad dinámica y densidad correctas? 1. Si  2. No"
      ELSEIF opv ==2
         PRINT "viscosidad cinemática correcta? 1. Si  2. No"
      ENDIF
      INPUT número entre 1 y 2 y asigna el resultado a ind
   END WHILE

ELSEIF oph ==2
   Asigna 2 a ind

   WHILE ind == 2
      Crea un vector de 2 columnas y asigna el resultado a p
      PRINT "presión inicial, p(1) y presión final, p(2)? "
```

```
INPUT presión en Kpa para cada columna de p
WHILE %T
   IF los valores de p > 0
      Sale del ciclo
   ELSE
      PRINT "presión inicial, p(1) y presión final, p(2)? "
      INPUT presión en Kpa para cada columna de p
   END IF
END WHILE

Calcula p * 1000, para convertir las presiones en Kpa a Pa y asigna el resultado a
p
Asigna el primer elemento de p a p1; Asigna el segundo elemento de p a p2
Crea un vector de 2 columnas y asigna el resultado a z
PRINT "nivel de altura inicial, z(1) y nivel de altura final, z(2)? "

INPUT nivel de altura en metros para cada columna de z
WHILE %T
   IF los valores de z > 0
      Sale del ciclo
   ELSE
      PRINT "nivel de altura inicial, z(1) y nivel de altura final, z(2)? "
      INPUT nivel de altura en metros para cada columna de z
   END IF
END WHILE
Asigna el primer elemento de z a z1; Asigna el segundo elemento de z a z2

INPUT velocidad inicial y asigna el resultado a v1
WHILE %T
   IF v1 > 0
      Sale del ciclo
   ELSE
     INPUT velocidad inicial y asigna el resultado a v1
   END IF
END WHILE
Agrupa a p1, p2, z1, z2 y v1 y asigna el resultado a prop_fl

INPUT densidad y asigna el resultado a ro
WHILE %T
   IF ro > 0
      Sale del ciclo
   ELSE
      INPUT densidad y asigna el resultado a ro
   END IF
```

```
      END WHILE
      PRINT "hay disponibilidad de la viscosidad dinámica o de viscosidad
            cinemática? 1. Si  2. No"
      INPUT número entre 1 y 2 y asigna el resultado a opv
      IF opv == 1
        INPUT viscosidad dinámica y asigna el resultado a visd
        WHILE %T
          IF visd > 0
            Sale del ciclo
          ELSE
            INPUT viscosidad dinámica y asigna el resultado a visd
          END IF
        END WHILE
        Agrupa a ro y visd, y asigna el resultado a parRe

      ELSEIF opv == 2
        INPUT viscosidad cinemática y asigna el resultado a vis_c
        WHILE %T
          IF vis_c > 0
            Sale del ciclo
          ELSE
            INPUT viscosidad cinemática y asigna el resultado a vis_c
          END IF
        END WHILE
        Agrupa a ro y vis_c y asigna el resultado a parRe
      END IF

      PRINT "presión inicial (p1), presión final (p2) correctas?"
      PRINT "altura inicial (z1), altura final (z2) correctas?"

      IF opv ==1
        PRINT "viscosidad dinámica y densidad correctas? 1. Si  2. No"
      ELSEIF opv ==2
        PRINT "viscosidad cinemática? 1. Si  2. No"
      END IF
      INPUT número entre 1 y 2 y asigna el resultado a ind
    ENDWHILE
ENDIF

Asigna 9.8 a g

INPUT diámetro de la tubería y asigna el resultado a D
WHILE %T
  IF D > 0
```

```
        Sale del ciclo
      ELSE
        INPUT Diámetro de la tubería yasigna el resultado a D
      END IF
   END WHILE
   Agrupa a D, rug, L, sumK y asigna el resultado a prop_tub
   Agrupa a oph y opv y asigna el resultado a op
   Evalúa calc_fl_vol(parRe, prop_fl, prop_tub, g, dir1, op), y asigna el resultado a vri
   Calcula el número de filas y columnas de vri, y asigna el resultado a s1;
   Almacena s1(1) y lo asigna a f1;
   Almacena vri(f1, 2) y asigna el resultado a v2

   Calcula 0.25 * π * D ^2 * v2 y asigna el resultado a fl_vol
   PRINT "flujo volumétrico: "
   PRINT fl_vol

   PRINT "continuación de los cálculos del diámetro de tubería? 1 Si  2 No"
   INPUT número entre 1 y 2 y asigna el resultado a flag
END WHILE
```

Cálculo de los flujos volumétricos a través de tuberías en paralelo (Problema 9°).

Fl_vol_par_calc

```
FUNCTION ftot(fd, L, D, sumK)

   Calcula fd * (L/D) + sumK (el coeficiente de fricción total = pérdidas primarias más
   secundarias) y asigna el resultado a res
   Devuelve res

END FUNCTION
```

```
FUNCTION h(fric_t, v, g)

   Calcula fric_t.*(v.^2)/(2*g) y asigna el resultado h_v
   Devuelve h_v

END FUNCTION
```

FUNCTION **desvh(h_v)**

Promedia los elementos de **h** y asigna el resultado a **h_prom**
Calcula **SUM((h_v - h_prom**).^2**)** y asigna el resultado a **desv**
Calcula (**desv**/(**LENGTH(h_v)**))**0.5 y asigna el resultado a **dh**
Devuelve **dh**

END FUNCTION

FUNCTION **vel_hf(g, h_prom, fric_t, D)**

Calcula (2 * **g** * **h_prom** ./(**fric_t**)) .^0.5 y asigna el resultado a **vel_hfv**
Devuelve **vel_hfv**

END FUNCTION

FUNCTION **tab(i, fd, veloc_v2, Qs)**

Define **tabs1**, **tabs2** y **tabs3** como variables globales
IF **i** ==1
Agrupa **i** y **fd** y asigna el resultado a **tabs1**
Agrupa **i** y **veloc_v2** y asigna el resultado a **tabs2**
Agrupa **i** y **Qs** y **SUM(Qs)** y lo asigna a **tabs3**
ELSEIF **i** > 1
Agrupa **i** y **fd** y el resultado lo asigna a **tabs1t**; Agrega **tabs1t** a **tabs**
Agrupa **i** y **veloc_v2** y el resultado lo asigna a **tabs2t**;
Agrega **tabs2t** a **tabs2;** Agrupa **i**, **Qs** y **SUM(Qs)**, y los agrega a **tabs3t**
Agrega **tabs3t** a **tabs3**
END IF
Agrupa a **tabs1**, **tabs2**, **tabs3** en una lista y la devuelve

END FUNCTION

FUNCTION **Qs_calc(Di, L, sumK, rug, caudal, parRe, g, dir1, op, imp)**

Utilizando **dir1**, establece la ruta del directorio de imp_op_tab, y asigna el resultado a **dir2**
Utilizando **dir2**, llama al espacio de trabajo a **imp_op_tab** desde imp_op_tab
Utilizando **dir1**, establece la ruta del directorio de f_fric, y asigna el resultado a **dir_fric**
Utilizando **dir_fric**, llama al espacio de trabajo a **vel** y a **calc_fdarcy_col**, desde f_fric
Asigna 20 a **n_max**
Calcula **caudal/LENGTH(Di)**, y asigna el resultado a **Q0**

Almacena el valor de **Q0** en cada uno de los elementos de un vector de longitud igual al de **Di** y asigna el resultado a **Qs**

FOR 0 < **i n_max**
  Realiza **ZEROS(LENGTH(Di))** y asigna el resultado a **fd**
  Realiza **ZEROS(LENGTH(Di))** y asigna el resultado a **veloc_v1**
  FOR 0< **j** <**LENGTH(Di)**
    Evalúa **vel(Qs**(**j**)**, Di**(**j**) y asigna el resultado a **veloc_v1**(**j**)
    Evalúa **calc_fdarcy_col(parRe**, **Qs**(**j**), **D**(**j**), **rug**(**j**)**, dir**, **op)**, y asigna el resultado al elemento **fd**(**j**)
  END FOR

  Evalua **ftot(fd, L, Di, sumK)**, y asigna el resultado a **fric_t**
  Evalua **h(fric_t**, **veloc_v1**, **g)** y asigna el resultado a **h_tub**
  Calcula el promedio de los elementos de **h_tub** y asigna el resultado a **h_prom**
  Evalua **desh(h_tub)** y asigna el resultado a **delh**
  Evalua **vel_hf(g**, **h_prom**, **fric_t, Di)** y asigna el resultado a **veloc_v2**

  FOR 0 < **j** < **LENGTH(Di)**
    Evalua **Q_calc(veloc_v2**(**j**), **Di**(**j**)) y asigna el resultado a **Qs**(**j**)
  END FOR

  Calcla **SUM(Qs)** y lo asigna a **Q_t**
  Calcula **ABS(Q_t – caudal)**, y lo asigna a **delQ**
  Evalua **tablas(i**, **fd**, **veloc_v2** y **Qs)** y asigna el resultado a **tabs1**, **tabs2** y a **tabs3**

  IF **delQ** > 0.001 y **delh** > 0.001
    Sale del ciclo FOR
  ELSE
    FOR 0 < **j** < **LENGTH(Di)**
      Calcula (**Qs**(**j**) / **Q_t)** * **caudal**
    END FOR

    IF **i** == **n_max**
      Sale del ciclo FOR
    END IF
  END IF
END FOR

Obtiene **LENGTH(Di)** y asigna el resultado a **n**
Agrupa a **tabs1**, **tabs2** y **tabs3** y asigna el resultado a **tab_g2**
Asigna "\Sistema_Paralelo" a **nfile**
Evalúa **imp_op_tab(tab_g2**, **n**, **dir1**, **imp**, **nfile)**
Agrupa **tabs1**, **tabs2**, **tabs3**, **tabs4** y asigna el resultado a **lt**

END FUNCTION

imp_tab

FUNCTION **fac_fric_imp(n)**

Crea una cadena de caracteres vacia y la asigna a **st3**
Crea una lista vacia y la asigna a **st1**
FOR 0 < **i** < **n** + 1
Construye parte de una cadena de caracteres a imprimir y la asigna a **st1**
Une el elemento **i** de **st1** y **st3** en una cadena de caracteres a imprimir y asigna el resultado a **st3**
END FOR
Devuelve **st3**

END FUNCTION

FUNCTION **vel_imp(n)**

Crea una cadena de caracteres vacia y la asigna a **st3**
Crea una lista vacia y la asigna a **st1**
FOR 0 < **i** < **n** + 2
Construye parte de una cadena de caracteres a imprimir y la asigna a **st1**
Une el elemento **i** de **st1** y **st3** en una cadena de caracteres a imprimir y asigna el resultado a **st3**
END FOR
Devuelve **st3**

END FUNCTION

FUNCTION **caudales_imp(n)**

Crea una cadena de caracteres vacia y la asigna a **st3**
Crea una lista vacia y la asigna a **st1**

FOR 0 < **i** < **n** + 2
Construye parte de una cadena de caracteres a imprimir y la asigna a **st1**
Une el elemento **i** de **st1** y **st3** en una cadena de caracteres a imprimir y asigna el resultado a **st3**
END FOR
Devuelve **st3**

END FUNCTION

imp_op_tab

FUNCTION **imp_op_tab(tab_g2, n, pathDir, imp, nfile)**

IF **LENGTH(imp)** ==2
Asigna **imp**(1) a **op_imp**; Asigna **imp**(2) a **datn**
ELSE
Asigna **imp**(1) a **op_imp**
END IF

Utilizando **pathDir**, establece la ruta del directorio de imp_tab y asigna el resultado a **dir1**
Utilizando **dir1**, llama al espacio de trabajo a **fac_fric_imp**, **vel_imp**, **caudales_imp** desde **imp_tab**
Evalua **fac_fric_imp(n)**, y asigna el resultado a **st3A**
Evalua **vel_imp(n)**, y asigna el resultado a **st3B**
Evalua **caudales_imp(n)**, y asigna el resultado a **st3C**

Agrupa **st3A**, **st3B**, **st3C** en una lista y la asigna a **st3_g**

IF **op_imp** ==1
PRINT **st3A**; PRINT **tab_g2**(1);
PRINT **st3B**; PRINT **tab_g2**(2)
PRINT **st3C;** PRINT **tab_g2**(3)

ELSEIF **op_imp** ==2
Utilizando **pathDir**, crea o utiliza la carpeta “Archivos_usuario” y guarda el resultado en **dir3**
Utiliza **dir3**, **datn** y **nfile** para formar una ruta especifica según **nfile**, y guarda el resultado en **dir4**
Utilizando **dir4**, crea una carpeta
Agrupa “fac_fric”, “veloc” y “caudales” y la guarda en **cad**
FOR 1 < **i** < 4
Utiliza **cad**(**i**) y **datn** para elaborar parte del nombre de un archivo .csv y lo guarda en **name2**
Utiliza **dir4** y **name2** para formar la ruta completa del archivo csv donde se guardaran los datos y lo almacena en **filename**
Utiliza **tab_g2**(**i**) para escribir los datos de **tab_g2**(**i**) en el archivo **filename**
END FOR
END IF
PRINT **dir4**

END FUNCTION

Calc_tub_par

Establece la ruta del directorio de Calc_tub_par, y asigna el resultado a **dir1**
Asigna 1 a **flag**

WHILE **flag** == 1
  Establece la ruta del directorio de Fl_vol_par_calc, y asigna el resultado a **dir2**
  Utilizando **dir2**, llama al espacio de trabajo a **Qs_calc**, desde Fl_vol_par_calc
  INPUT nombre del fluido y asigna el resultado a **nameco**

  Asigna 2 a **ind**
  WHILE **ind** == 2
    INPUT número de tuberías, y asigna el resultado a **num**
    WHILE **%T**
      IF **num** > 0
        Sale del ciclo
      ELSE
        INPUT número de tuberías, y asigna el resultado a **num**
      END IF
    END WHILE

    Realiza **ZEROS(num)** y el resultado lo asigna a **D**
    FOR 0 < **i** < **num**
      INPUT diámetro de tubería (i) y asigna el resultado a **D(i)**
      WHILE **%T**
        IF **D(i)** > 0
          Sale del ciclo
        ELSE
          INPUT diámetro de tubería (i) y asigna el resultado a **D(i)**
        END IF
      END WHILE
    END FOR

    FOR 0 < **i** < **num**
      INPUT longitud de tubería (i) y asigna el resultado a **L(i)**
      WHILE **%T**
        IF **L(i)** > 0
          Sale del ciclo
        ELSE
          INPUT longitud de tubería (i) y asigna el resultado a **L(i)**

```
        END IF
      END WHILE
   END FOR

   PRINT "número, diámetro y longitud de tuberías están correctos? 1. Si  2. No"
   INPUT número entre 1 y 2 y asigna el resultado a ind
ENDWHILE

Asigna 2 a ind
WHILE ind == 2
   FOR 0 < i < num
     PRINT "rugosidad de tubería (i)?"

     INPUT rugosidad de tubería (i) y asigna el resultado a rug(i)
      WHILE %T
        IF rug(i) > 0
          Sale del ciclo
        ELSE
          PRINT "rugosidad de tubería (i)?"
          INPUT rugosidad de tubería (i) y asigna el resultado a rug(i)
        END IF
      END WHILE
   END FOR

   Realiza ZEROS(num) y el resultado lo asigna a sumK
   FOR 0 < i < num
     INPUT suma de los coeficientes K de pérdidas secundarias, y asigna el resultado
           a sumK

     WHILE %T
        IF sumK(i) > 0
          Sale del ciclo
        ELSE
          INPUT suma de los coeficientes K de pérdidas secundarias, y asigna el
                resultado a sumK
        END IF
     END WHILE
   END FOR
   PRINT "rugosidad y longitud de la tubería y suma de los coeficientes K de pérdidas
          secundarias de la tubería están correctos? 1. Si  2. No"
   INPUT número entre 1 y 2 y asigna el resultado a ind
ENDWHILE
PRINT "hay disponibilidad de la viscosidad dinámica y de la densidad o de viscosidad
        cinemática? 1. Si  2. No"
```

```
INPUT número entre 1 y 2 y asigna el resultado a opv

IF opv ==1
   Asigna 2 a ind

   WHILE ind == 2
      INPUT densidad del fluido, y asigna el resultado a ro
      WHILE %T
         IF ro > 0
            Sale del ciclo
         ELSE
            INPUT densidad del fluido y asigna el resultado a ro
         END IF
      END WHILE

      INPUT viscosidad dinámica y asigna el resultado a visd
      WHILE %T
         IF visd > 0
            Sale del ciclo
         ELSE
            INPUT viscosidad dinámica y asigna el resultado a visd
         END IF
      END WHILE

      Agrupa a ro y visd, y asigna el resultado a parRe

     PRINT "viscosidad dinámica y densidad correctas? 1. Si  2. No"
     INPUT número entre 1 y 2 y asigna el resultado a ind
  ENDWHILE

ELSEIF opv == 2
   Asigna 2 a ind

   WHILE ind == 2
      INPUT viscosidad cinemática y asigna el resultado a visd
      WHILE %T
         IF vis_c > 0
            Sale del ciclo
         ELSE
            INPUT viscosidad cinemática y asigna el resultado a vis_c
         END IF
      END WHILE
      Asigna vis_c a parRe
```

```
        PRINT "viscosidad cinemática correcta? 1. Si  2. No"
        INPUT número entre 1 y 2 y asigna el resultado a ind
      ENDWHILE
   END IF

   INPUT caudal del fluido, y asigna el resultado a Q
   WHILE %T
      IF Q > 0
         Sale del ciclo
      ELSE
         INPUT caudal del fluido y asigna el resultado a Q?
      END IF
   END WHILE

   Asigna 9.8 a g; Asigna opv a op

   PRINT "Imprimir resultados: 1. en la consola, 2. archivos.csv, 3 imprimir última
            iteración"
   INPUT número entre 1, 2 o 3 y asigna el resultado a op_imp

   Evalúa a Qs_calc(D, L, sumK, rug, caudal, parRe, g, dir1, op, imp) y guarda el
   resultado en cau_tubs.
   PRINT "caudales para las tuberías en paralelo: "
   PRINT cau_tubs

   PRINT "continuación de los cálculos de caudal en sistemas de tuberías en paralelo? 1
            Si  2 No"
   INPUT número entre 1 y 2 y asigna el resultado a flag

ENDWHILE
```

C.5 Asignatura: Operaciones Unitarias III

Cálculo de las composiciones de los productos de líquido y vapor, en procesos de destilación instantánea de mezclas multicomponentes (Problema 10°).

Algunos de los programas elaborados para el problema de cálculo del punto de burbuja T utilizando la ecuación de Raoult modificada y la formulación gamma-phi (mezclas binarias, ternarias y cuaternarias, problema 4° y 5°) fueron reutilizados y fueron:

- cpre_ing
- fpres_vap
- BurbT
- coac
- Wcomp
- NRTLcomp
- UNIFAC
- phiCM
- tablas_read
- opc
- imp_tab
- imp_op_tab

Los seudocódigos descritos en esta sección corresponden a:

- BurbP
- RoP
- RoT
- opcK
- opdK
- Vapor_ins
- imp_op_tab_Kcal
- Kdatos

BurbP

FUNCTION **pres(x,tpr,cof,phicf,cat_mayor,n,pathDir)**
Utilizando **pathDir**, establece la ruta del directorio de fpres_vap y guarda el resultado

en **dir1**
Utilizando **dir1**, llama al espacio de trabajo a **Psat**, desde fpres_vap

Realiza **ZEROS(n)**
FOR 1 < **i** < **n**
Calcula **x**(**i**)*__cof__(**i**)*__Psat(tpr__,__cat_mayor__(**i**))/__phicf__(**i**)
END FOR
Evalúa **SUM(p_par)** y asigna el resultado a **pt**

END FUNCTION

FUNCTION **yBurbp(x,t,cat_mayor,opa,prop,op_gid,pathDir,n,imp,param)**

Utilizando **pathDir**, establece el directorio de coac y asigna el resultado a **dir1_1**
Utilizando **pathDir**, establece el directorio de phiCM y asigna el resultado a **dir2_1**
Utilizando **pathDir**, establece el directorio de fpres_vap y asigna el resultado a **dir3_1**
Utilizando **pathDir**, establece el directorio de cre_ing y asigna el resultado a **dir3_2**
Utilizando **pathDir**, establece el directorio de imp_op_tab y asigna el resultado a **dir4_1**

Asigna **t** a **tK**
FOR 1 < **i** < **n**
Asigna 1.0000001 a **cof**(**i**); Asigna 1.0000001 a **phicf**(**i**);
Asigna **Psat(tK**,**cat_mayor**(**i**)**)** y asigna el resultado a **ps**(**i**)
Calcula 1**n** y asigna el resultado a **y**
END FOR

IF **op_gid** ~= 0
Evalúa **coac(opa,prop,tK,x,n,pathDir)** y asigna el resultado a **cof**
END IF

Evalúa **pres(x,tK,cof,phicf,cat_mayor,n)** y asigna el resultado a **p0**
Realiza **ZEROS(n)** y asigna el resultado a **coetma1**
Realiza **ZEROS(n)** y asigna el resultado a **coetma2**
Realiza **ZEROS(n)** y asigna el resultado a **coetma3**
Realiza **ZEROS(n)** y asigna el resultado a **coetma4**
Realiza **ZEROS(n)** y asigna el resultado a **coetg1**
Realiza **ZEROS(n)** y asigna el resultado a **coetg2**
Realiza **ZEROS(n)** y asigna el resultado a **coetg3**
Realiza **ZEROS(n)** y asigna el resultado a **coetg4**

FOR 1 < **i** < 2
IF **i** == 1
Asigna 1 a **coetma1**(**i**); Asigna 1 a **coetma2**(**i**)

```
      Asigna 1 a coetma3(i); Asigna 1 a coetma4(i)

   ELSEIF i == 2
      Asigna p0 a coetma1(i); Asigna p0 a coetma2(i)
      Asigna p0 a coetma3(i); Asigna p0 a coetma4(i)
   END IF
END FOR

Almacena coetma1(1,3:2+n) en y; Almacena coetma2(1,3:2+n) en ps;
Almacena coetma3(1,3:2+n) en cof; Almacena coetma4(1,3:2+n) en phicf;
Asigna 3 a delp; Asigna 40 a imaxp; Asigna 1 a ip

WHILE delp > 0.000001
   Realiza ZEROS(n) y asigna el resultado a yte
   FOR 1 < i < n
      Calcula x(i)*cof(i)*ps(i)/(phicf(i)*p0) y asigna el resultado a yte(i)
   END FOR
   Evalúa SUM(yte) y asigna el resultado a sumat

   FOR 1 < i < n
      Calcula yte(i)/sumat y asigna el resultado a yte(i)
   END FOR
   Asigna yte a y; Elimina yte

   Calcula p0/100 y asigna el resultado a p0phi

   IF op_gid == 2
      Guarda param(1) en B; Guarda param(2) en opi; Guarda param(3) en opB
      Calcula phiCM(tK1,p0phi,cat_mayor,B,opi,opB,n,pathDir,y) y asigna el
      resultado a phicf
   END IF
   Evalúa pres(x,tK,cof,phicf,cat_mayor,n) y asigna el resultado a p0
   Evalúa ABS(p0p-p0) y asigna el resultado a delp

   FOR 1 < i < 2
      IF i == 1
         Asigna ip + 1 a coetg1(ip + 1); Asigna ip + 1 a coetg2(i)
         Asigna ip + 1 a coetg3(ip + 1); Asigna ip + 1 a coetg4(i)

      ELSEIF i == 2
         Asigna p0 a coetg1(i); Asigna p0 a coetg2(i)
         Asigna p0 a coetg3(i); Asigna p0 a coetg4(i)
      END IF
   END FOR
```

Almacena **coetg1**(1,3:2+**n**) en **y**; Almacena **coetg2**(1,3:2+**n**) en **ps**;
Almacena **coetg3**(1,3:2+**n**) en **cof**; Almacena **coetg4**(1,3:2+**n**) en **phicf**;
Calcula **ip** + 1 a **ip**
IF **ip** > **imaxp**
Sale del ciclo
END IF
END WHILE

Calcula **p0**/100 y asigna el resultado a **p0phi**
IF **op_gid** == 2
Guarda **param**(1) en **B**; Guarda **param**(2) en **opi**; Guarda **param**(3) en **opB**
Calcula **phiCM(tK1,p0phi,cat_mayor,B,opi,opB,n,pathDir,y)** y asigna el resultado a **phicf**
END IF
Calcula las dimensiones de **coetg1** y asigna el resultado a **sc1**;
Asigna **sc1**(1) a **filsc1**; Asigna sc1(2) a **colsc1**
Almacena **coetma1** en **coetf1**(1,1:**colsc1**)
Almacena **coetg1** en **coetf1**(2:**filsc1**+1,1:**colsc1**)
Almacena **coetma2** en **coetf2**(1,1:**colsc1**)
Almacena **coetg2** en **coetf2**(2:**filsc1**+1,1:**colsc1**)
Almacena **coetma3** en **coetf3**(1,1:**colsc1**)
Almacena **coetg3** en **coetf3**(2:**filsc1**+1,1:**colsc1**)
Almacena **coetma4** en **coetf4**(1,1:**colsc1**)
Almacena **coetg4** en **coetf4**(2:**filsc1**+1,1:**colsc1**)

Agrupa **coetf1**, **coetf2**, **coetf3** y **coetf4** y asigna el resultado a **coetf_gru**
Asigna “P” a **car1**; Asigna “y” a **car2**; Asigna “Pre” a **car3**; Asigna “\\Burbuja” a **nfile**
Asigna “Burbuja P” a **name**
Evalúa **imp_op_tab(coetf_gru,n,car1,car2,car3,pathDir,imp,nfile,name)**
Agrupa "burbuja", **p0**, **tK**, **x**, **y**, **cof** y **phicf** y asigna el resultado a **catalog**
Devuelve **catalog**

END FUNCTION

RoP

FUNCTION **pres(x,tpr,cof,phicf,cat_mayor,n,pathDir)**

Utilizando **pathDir**, establece la ruta del directorio de fpres_vap y guarda el resultado en **dir1**
Utilizando **dir1**, llama al espacio de trabajo a **Psat**, desde fpres_vap
Realiza **ZEROS(n)**

```
   FOR 1 < i < n
      Calcula x(i)*cof(i)*Psat(tpr,cat_mayor(i))/phicf(i)
   END FOR
   Evalúa SUM(p_par) y asigna el resultado a pt

END FUNCTION
```

```
FUNCTION xRop(y,t,cat_mayor,opa,prop,op_gid,pathDir,n,imp,param)

   Utilizando pathDir, establece el directorio de coac y asigna el resultado a dir1_1
   Utilizando dir1_1, llama al espacio de trabajo a coac, desde coac
   Utilizando pathDir, establece el directorio de phiCM y asigna el resultado a dir2_1
   Utilizando dir2_1, llama al espacio de trabajo a phiCM, desde phiCM
   Utilizando pathDir, establece el directorio de fpres_vap y asigna el resultado a dir3_1
   Utilizando dir3_1, llama al espacio de trabajo a Psat, desde fpres_vap
   Utilizando pathDir, establece el directorio de cre_ing y asigna el resultado a dir3_2
   Utilizando dir3_2, llama al espacio de trabajo a cre_ing, desde cpre_ing
   Utilizando pathDir, establece el directorio de imp_op_tab y asigna el resultado a
   dir4_1
   Utilizando dir4_1, llama al espacio de trabajo a imp_op_tab, desde imp_op_tab

   Asigna t a tK
   FOR 1 < i < n
      Asigna 1.0000001 a cof(i); Asigna 1.0000001 a phicf(i);
      Asigna Psat(tK,cat_mayor(i)) y asigna el resultado a ps(i)
      Calcula 1\n y asigna el resultado a x
   END FOR

   Evalúa pres(x,tK,cof,phicf,cat_mayor,n) y asigna el resultado a p0
   Realiza ZEROS(n) y asigna el resultado a coetma1
   Realiza ZEROS(n) y asigna el resultado a coetma2
   Realiza ZEROS(n) y asigna el resultado a coetma3
   Realiza ZEROS(n) y asigna el resultado a coetma4
   Realiza ZEROS(n) y asigna el resultado a coetm1
   Realiza ZEROS(n) y asigna el resultado a coetm2
   Realiza ZEROS(n) y asigna el resultado a coetm3
   Realiza ZEROS(n) y asigna el resultado a coetm4

   FOR 1 < i < 2
      IF i == 1
         Asigna 1 a coetma1(1,i); Asigna 1 a coetma2(1,i)
         Asigna 1 a coetma3(1,i); Asigna 1 a coetma4(1,i)

      ELSEIF i == 2
```

```
      Asigna p0 a coetma1(1,i); Asigna p0 a coetma2(1,i)
      Asigna p0 a coetma3(1,i); Asigna p0 a coetma4(1,i)
   END IF
END FOR

Almacena coetma1(1,3:2+n) en x; Almacena coetma2(1,3:2+n) en ps;
Almacena coetma3(1,3:2+n) en cof; Almacena coetma4(1,3:2+n) en phicf;

Realiza ZEROS(n) y asigna el resultado a xte
FOR 1 < i < n
   Calcula y(i)*phicf(i)*p0/(cof(i)*ps(i)) y asigna el resultdo a xte
END FOR
Calcula SUM(xte) y asigna el resultado a sumat
FOR 1 < i < n
   Calcula xte(i)/sumat y asigna el resultado a xte
END FOR
Asigna xte a x; Elimina a xte

IF op_gid ~= 0
   Evalúa coac(opa,prop,tK,x,n,pathDir) y asigna el resultado a cof
ELSE
   Asigna cof a cofp
END IF
Evalúa pres(x,tK,cof,phicf,cat_mayor,n) y asigna el resultado a p0
Evalúa ABS(p0p-p0) y asigna el resultado a delp

FOR 1 < i < n
   Evalúa ABS(cof(i)-cofp(i)) y asigna el resultado a delcoet(i)
END FOR
Calcula NORM(delcoet) y asigna el resultado a delc

FOR 1 < i < n
   Asigna cofp(i) a cof(i)
END FOR

FOR 1 < i < 2
   IF i == 1
      Asigna 2 a coetma1(2,i); Asigna 2 a coetma2(2,i)
      Asigna 2 a coetma3(2,i); Asigna 2 a coetma4(2,i)

   ELSEIF i == 2
      Asigna p0 a coetma1(2,i); Asigna p0 a coetma2(2,i)
      Asigna p0 a coetma3(2,i); Asigna p0 a coetma4(2,i)
   END IF
```

END FOR

Almacena **coetma1**(2,3:2+**n**) en **x**; Almacena **coetma2**(2,3:2+**n**) en **ps**;
Almacena **coetma3**(2,3:2+**n**) en **cof**; Almacena **coetma4**(2,3:2+**n**) en **phicf**;

Asigna 50 a **imaxp**; Asigna 1 a **ip**

WHILE **delp** > 0.000001
  Calcula **p0**/100 y asigna el resultado a **p0phi**

  IF **op_gid** == 2
    Guarda **param**(1) en **B**; Guarda **param**(2) en **opi**; Guarda **param**(3) en **opB**
    Calcula **phiCM(tK1,p0phi,cat_mayor,B,opi,opB,n,pathDir,y)** y asigna el resultado a **phicf**
  END IF

  Realiza **ZEROS(n)** y asigna el resultado a **coetg1**
  Realiza **ZEROS(n)** y asigna el resultado a **coetg2**
  Realiza **ZEROS(n)** y asigna el resultado a **coetg3**
  Realiza **ZEROS(n)** y asigna el resultado a **coetg4**
  Asigna 40 a **imaxc**; Asigna 1 a **i;** Asigna 1 a **delc**
  WHILE **delc** > 0.001
    Realiza **ZEROS(n)** y asigna el resultado a **xte**
    FOR 1 < **i** < **n**
      Calcula **y**(**i**)*__phicf__(**i**)*__p0__/(__cof__(**i**)*__ps__(**i**)) y asigna el resultado a **xte**(**i**)
    END FOR
    Evalúa **SUM(xte)** y asigna el resultado a sumat

    FOR 1 < **i** < **n**
      Calcula **xte**(**i**)/**sumat** y asigna el resultado a **xte**(**i**)
    END FOR
    Asigna **xte** a **x**; Elimina **xte**

    IF **op_gid** ~= 0
      Evalúa **coac(opa,prop,tK,x,n,pathDir)** y asigna el resultado a **cof**
    ELSE
      Asigna **cof** a **cofp**
    END IF

    FOR 1 < **i** < **n**
      Evalúa **ABS(cof**(**i**)-**cofp**(**i**)**)** y asigna el resultado a **delcoet**(**i**)
    END FOR
    Calcula **NORM(delcoet)** y asigna el resultado a **delc**
    FOR 1 < **i** < **n**

```
        Asigna cofp(i) a cof(i)
      END FOR

      Almacena coetg1(i,2:1+n) en y; Almacena coetg2(i,2:1+n) en ps;
      Almacena coetg3(i,2:1+n) en cof; Almacena coetg4(i,2:1+n) en phicf;
      Almacena i en coetg1(i,1) en x; Almacena coetg2(i,1) en ps;
      Almacena i en coetg3(i,1) en cof; Almacena coetg4(i,n) en phicf;
      Calcula i + 1 y asigna el resultado a i

      IF i > imaxc
        Sale del ciclo
      END IF
    END WHILE

    Calcula las dimensiones de coetg1 y asigna el resultado a sc1
    Asigna sc1(1) a filsc1; Asigna sc1(2) a colsc1
    FOR 1 < i < 2
      IF i == 1
        Asigna ip + 2 a coetm1(ip,i); Asigna 2 a coetma2(ip,i)
        Asigna ip + 2 a coetm3(ip,i); Asigna 2 a coetma4(ip,i)

      ELSEIF i == 2
        Asigna p0 a coetma1(ip,i); Asigna p0 a coetma2(ip,i)
        Asigna p0 a coetma3(ip,i); Asigna p0 a coetma4(ip,i)
      END IF
    END FOR

    Almacena coetg1(filsc1,2:colsc1) en coetm1(ip,3:colsc1+1)
    Almacena coetg2(filsc1,2:colsc1) en coetm2(ip,3:colsc1+1)
    Almacena coetg3(filsc1,2:colsc1) en coetm3(ip,3:colsc1+1)
    Almacena coetg4(filsc1,2:colsc1) en coetm4(ip,3:colsc1+1)

    Evalúa pres(x,tK,cof,phicf,cat_mayor,n) y asigna el resultado a p0
    Evalúa ABS(p0p-p0) y asigna el resultado a delp
    Realiza ZEROS(n) a coetg1; Realiza ZEROS(n) a coetg2
    Realiza ZEROS(n) a coetg3; Realiza ZEROS(n) a coetg4

    IF i > imaxc
      Sale del ciclo
    END IF
  END WHILE

  IF op_gid ~= 0
    Evalúa coac(opa,prop,tK,x,n,pathDir) y asigna el resultado a cof
```

IF **op_gid** == 2
Guarda **param**(1) en **B**; Guarda **param**(2) en **opi**; Guarda **param**(3) en **opB**
Calcula **phiCM(tK1,p0phi,cat_mayor,B,opi,opB,n,pathDir,y)** y asigna el resultado a **phicf**
END IF
END IF

Calcula las dimensiones de **coetm1** y asigna el resultado a **sc3**;
Asigna **sc3**(1) a **filsc3**; Asigna **sc3**(2) a **colsc3**
Almacena **coetma1** a **coetf1**(1:2,1:**colsc3**)
Almacena **coetm1** a **coetf1**(3:**filcs3**+2,1:**colsc3**)
Almacena **coetma2** a **coetf2**(1:2,1:**colsc3**)
Almacena **coetm2** a **coetf2**(3:**filcs3**+2,1:**colsc3**)
Almacena **coetma3** a **coetf3**(1:2,1:**colsc3**)
Almacena **coetm3** a **coetf3**(3:**filcs3**+2,1:**colsc3**)
Almacena **coetma4** a **coetf4**(1:2,1:**colsc3**)
Almacena **coetm4** a **coetf4**(3:**filcs3**+2,1:**colsc3**)

Agrupa **coetf1**, **coetf2**, **coetf3** y **coetf4** y asigna el resultado a **coetf_gru**
Asigna "P" a **car1**; Asigna "x" a **car2**; Asigna "Pre" a **car3**; Asigna "\\RocioP" a nfile
Asigna "Rocio P" a name
Evalúa **imp_op_tab(coetf_gru,n,car1,car2,car3,pathDir,imp,nfile,name)**
Agrupa "rocio", **p0**, **tK**, **x**, **y**, **cof** y **phicf** y asigna el resultado a **catalog**
Devuelve **catalog**

END FUNCTION

RoT

FUNCTION **te2(tpr,x,pt,cof,phicf,cat_mayor,n,pathDir)**

Utilizando **pathDir**, establece la ruta del directorio de fpres_vap y guarda el resultado en **dir3**
Utilizando **dir3**, llama al espacio de trabajo a **Psat**, desde fpres_vap
Realiza **ZEROS(n)** y lo asigna a **sum1**;
Realiza **ZEROS(n)** y lo asigna a **presat**

FOR 1 < **i** < **n**
Evalúa **Psat(tpr,cat_mayor(i))** y asigna el resultado a **presat**(**i**)
Calcula **y**(**i**)*__phicf__(**i**)**x**(**i**)/(**cof**(**i**)*__presat__(**i**)) y asigna el resultado a **sumte**(**i**)
END FOR
Calcula **SUM(sumte)** y asigna el resultado a **sum2**

Calcula –**pt** + **sum2** y asigna el resultado a **funct**

END FUNCTION

FUNCTION **xRot(y,p,cat_mayor,opa,prop,op_gid,pathDir,n,imp,param)**

Utilizando **pathDir**, establece el directorio de coac y asigna el resultado a **dir1_1**
Utilizando **dir1_1**, llama al espacio de trabajo a **coac**, desde coac
Utilizando **pathDir**, establece el directorio de phiCM y asigna el resultado a **dir1_2**
Utilizando **dir1_2**, llama al espacio de trabajo a **phiCM**, desde phiCM
Utilizando **pathDir**, establece el directorio de fpres_vap y asigna el resultado a **dir2_1**
Utilizando **dir2_1**, llama al espacio de trabajo a **Psat**, desde fpres_vap
Utilizando **pathDir**, establece el directorio de rootSecantePlus2Ap2 y asigna el resultado a **dir3_1**
Utilizando **dir3_1**, llama al espacio de trabajo a **rootSecantePlus2A**, desde rootSecantePlus2Ap2
Utilizando **pathDir**, establece el directorio de imp_op_tab y asigna el resultado a **dir4_1**
Utilizando **dir4_1**, llama al espacio de trabajo a **imp_op_tab**, desde imp_op_tab
Realiza **ZEROS(n)** y asigna el resultado a **tsat**
FOR 1 < **i** < **n**
  Evalúa a **t_pv(p**,**cat_mayor**(**i**)**)** y asigna el resultado a **tsat**(**i**)
END FOR

Asigna 0 a **t0**
FOR 1 < **i** < **n**
  Calcula **t0**+**y**(**i**)*****tsat**(**i**) y asigna el resultado a **t0**
END FOR

Asigna **t0** a **t0K**
Realiza **ZEROS(n)** y asigna el resultado a **cof**
Realiza **ZEROS(n)** y asigna el resultado a **phicf**
Realiza **ZEROS(n)** y asigna el resultado a **ps**
Realiza **ZEROS(n)** y asigna el resultado a **xte**
Asigna 1.0000001 a **cof**(**i**); Asigna 1.0000001 a **phicf**(**i**);

FOR 1 < **i** < **n**
  Asigna **Psat(tK**,**cat_mayor**(**i**)**)** y asigna el resultado a **ps**(**i**)
END FOR

Realiza **ZEROS(n)** y asigna el resultado a **coetma1**
Realiza **ZEROS(n)** y asigna el resultado a **coetma2**
Realiza **ZEROS(n)** y asigna el resultado a **coetma3**
Realiza **ZEROS(n)** y asigna el resultado a **coetma4**

FOR 1 < **i** < 2
  IF **j** == 1
    Asigna 1 a **coetma1**(1,**j**); Asigna 1 a **coetma2**(1,**j**)
    Asigna 1 a **coetma3**(1,**j**); Asigna 1 a **coetma4**(1,**j**)

  ELSEIF **j** == 2
    Asigna **t0K** a **coetma1**(1,**j**); Asigna **t0K** a **coetma2**(1,**j**)
    Asigna **t0K** a **coetma3**(1,**j**); Asigna **t0K** a **coetma4**(1,**j**)
  END IF
END FOR

Almacena **coetma1**(1,3:2+**n**) en **cx**; Almacena **coetma2**(1,3:2+**n**) en **ps**;
Almacena **coetma3**(1,3:2+**n**) en **cof**; Almacena **coetma4**(1,3:2+**n**) en **phicf**;
Agrupa **y**, **p**, **cof**, **phicf**, **cat_mayor**, **n** y **pathDir** y asigna el resultado a **l**
Evalúa **rootSecantePlus2A(t0K, te2,1,0.001,2,store=%T)** y asigna el resultado a **vri**
Calcula las dimensiones de **vri** y asigna el resultado a **s1**
Asigna **s1**(1) a **f1**;
Asigna **vri**(f1,2) a **tK1**

Realiza **ZEROS(n)** y asigna el resultado a **ps**
FOR 1 < **i** < **n**
  Evalúa **Psat(tK1**,**cat_mayor**(**i**)**)** y asigna el resultado a **ps**(**i**))
END FOR

Calcula **p**/100 y asigna el resultado a **p0phi**
IF **op_gid** == 2
  Guarda **param**(1) en **B**; Guarda **param**(2) en **opi**; Guarda **param**(3) en **opB**
  Calcula **phiCM(tK1,p0phi,cat_mayor,B,opi,opB,n,pathDir,y)** y asigna el resultado a **phicf**
END IF

Realiza **ZEROS(n)** y asigna el resultado a **xte**
FOR 1 < **i** < **n**
  Calcula **y**(**i**)*__phicf__(**i**)*__p0__/(__cof__(**i**)*__ps__(**i**)) y asigna el resultado a **xte**(**i**)
END FOR
Evalúa **SUM(xte)** y asigna el resultado a **sumat**

FOR 1 < **i** < **n**
  Calcula **xte**(**i**)/**sumat** y asigna el resultado a **xte**(**i**)
END FOR
Asigna **xte** a **x**; Elimina **xte**
FOR 1 < **j** < 2
  IF **j** == 1

```
      Asigna 2 a coetma1(2,j); Asigna 2 a coetma2(2,j)
      Asigna 2 a coetma3(2,j); Asigna 2 a coetma4(2,j)

   ELSEIF j == 2
      Asigna tK1 a coetma1(2,j); Asigna tK1 a coetma2(2,j)
      Asigna tK1 a coetma3(2,j); Asigna tK1 a coetma4(2,j)
   END IF
END FOR

Almacena coetma1(2,3:2+n) en xte; Almacena coetma2(2,3:2+n) en ps;
Almacena coetma3(2,3:2+n) en cof; Almacena coetma4(2,3:2+n) en phicf;

IF op_gid ~= 0
   Evalúa coac(opa,prop,tK,cx,n,pathDir) y asigna el resultado a cof
ELSE
   Asigna cof a cofp
END IF

Agrupa y, p, cof, phicf, cat_mayor, n y pathDir y asigna el resultado a l
Evalúa rootSecantePlus2A(t0K, te2,1,0.001,2,store=%T) y asigna el resultado a vri
Calcula las dimensiones de vri y asigna el resultado a s1
Asigna s1(1) a f1;
Asigna vri(f1,2) a tK1p
Evalúa ABS(tK1p-tK1) y asigna el resultado a delt; Asigna tK1p a tK1
Realiza ZEROS(n) y asigna el resultado a delcoet
FOR 1 < i < n
   Evalúa ABS(cof(i)-cofp(i)) y asigna el resultado a delcoet(i)
END FOR

FOR 1 < j < 2
   IF j == 1
      Asigna 3 a coetma1(3,j); Asigna 3 a coetma2(3,j)
      Asigna 3 a coetma3(3,j); Asigna 3 a coetma4(2,j)

   ELSEIF j == 2
      Asigna tK1 a coetma1(3,j); Asigna tK1 a coetma2(3,j)
      Asigna tK1 a coetma3(3,j); Asigna tK1 a coetma4(3,j)
   END IF
END FOR

Almacena coetma1(3,3:2+n) en x; Almacena coetma2(3,3:2+n) en ps;
Almacena coetma3(3,3:2+n) en cof; Almacena coetma4(3,3:2+n) en phicf;
Evalúa NORM(delcoet) y asigna el resultado a delc; Asigna cofp a cof
Asigna 60 a imaxt; Asigna 1 a it
```

Realiza **ZEROS(n)** y asigna el resultado a **coetm1**
Realiza **ZEROS(n)** y asigna el resultado a **coetm2**
Realiza **ZEROS(n)** y asigna el resultado a **coetm3**
Realiza **ZEROS(n)** y asigna el resultado a **coetm4**

WHILE **delp** > 0.000001
  Calcula **p0**/100 y asigna el resultado a **p0phi**

  IF **op_gid** == 3
    Guarda **param**(1) en **B**; Guarda **param**(2) en **opi**; Guarda **param**(3) en **opB**
    Calcula **phiCM(tK1,p0phi,cat_mayor,B,opi,opB,n,pathDir,y)** y asigna el resultado a **phicf**
  END IF

  Realiza **ZEROS(n)** y asigna el resultado a **ps**
  FOR 1 < **i** < **n**
    Evalúa **Psat(tK1**,**cat_mayor**(**i**)**)** y asigna el resultado a **ps**(**i**))
  END FOR

  Realiza **ZEROS(n+2)** y asigna el resultado a **coetg1**
  Realiza **ZEROS(n+2)** y asigna el resultado a **coetg2**
  Realiza **ZEROS(n+2)** y asigna el resultado a **coetg3**
  Realiza **ZEROS(n+2)** y asigna el resultado a **coetg4**

  Asigna 20 a **imaxc**; Asigna 1 a **it2;** Asigna 1 a **delc**
  WHILE **delc** > 0.001
    Realiza **ZEROS(n)** y asigna el resultado a **xte**
    FOR 1 < **i** < **n**
      Calcula **y**(**i**)*__phicf__(**i**)*__p0__/(__cof__(**i**)*__ps__(**i**)) y asigna el resultado a **xte**(**i**)
    END FOR
    Evalúa **SUM(xte)** y asigna el resultado a **sumat**

    FOR 1 < **i** < **n**
      Calcula **xte**(**i**)/**sumat** y asigna el resultado a **xte**(**i**)
    END FOR
    Asigna **xte** a **x**; Elimina **xte**

    IF **op_gid** ~= 0
      Evalúa **coac(opa,prop,tK,x,n,pathDir)** y asigna el resultado a **cof**
    ELSE
      Asigna **cof** a **cofp**
    END IF

    FOR 1 < **i** < **n**

```
   Evalúa ABS(cof(i)-cofp(i)) y asigna el resultado a delcoet(i)
END FOR
Calcula NORM(delcoet) y asigna el resultado a delc

FOR 1 < i < n
   Asigna cofp(i) a cof(i)
END FOR

Realiza ZEROS(n+2) y asigna el resultado a coetg1t
Realiza ZEROS(n+2) y asigna el resultado a coetg2t
Realiza ZEROS(n+2) y asigna el resultado a coetg3t
Realiza ZEROS(n+2) y asigna el resultado a coetg4t

FOR 1 < i < 2
   IF it2 == 1
      IF i == 1
         Asigna 3 a coetg1(1,i); Asigna 3 a coetg2(1,i)
         Asigna 3 a coetg3(1,i); Asigna 3 a coetg4(1,i)

      ELSEIF i == 2
         Asigna tK1 a coetg1(1,i); Asigna tK1 a coetg2(1,i)
         Asigna tK1 a coetg3(1,i); Asigna tK1 a coetg4(1,i)
      END IF
   IF it2 > 1
      IF i == 1
         Asigna 3 a coetg1t(1,i); Asigna 3 a coetg2t(1,i)
         Asigna 3 a coetg3t(1,i); Asigna 3 a coetg4t(1,i)

      ELSEIF i == 2
         Asigna tK1 a coetg1t(1,i); Asigna tK1 a coetg2t(1,i)
         Asigna tK1 a coetg3t(1,i); Asigna tK1 a coetg4t(1,i)
      END IF
   END IF
END FOR

IF it2 == 1
   Almacena coetg1(1,3:2+n) en y; Almacena coetg2(1,3:2+n) en ps;
   Almacena coetg3(1,3:2+n) en cof; Almacena coetg4(1,3:2+n) en phicf;
ELSEIF it2 == 1
   Almacena coetg1t(1,3:2+n) en y; Almacena coetg2t(1,3:2+n) en ps;
   Almacena coetg3t(1,3:2+n) en cof; Almacena coetg4t(1,3:2+n) en phicf;
   Une coetg1 y coetg1t y asigna el resultado a coetg1
   Une coetg2 y coetg2t y asigna el resultado a coetg2
   Une coetg3 y coetg3t y asigna el resultado a coetg3
```

```
      Une coetg4 y coetg4t y asigna el resultado a coetg4
   END

   Calcula it2 + 1 y asigna el resultado a it2

   IF it2 > imaxc
      Sale del ciclo
   END IF
END WHILE

Calcula las dimensiones de coetg1 y asigna el resultado a sc1
Asigna sc1(1) a fsc1

IF fsc1 == 1
   Realiza ZEROS(n+2) y asigna el resultado a coetm1t
   Realiza ZEROS(n+2) y asigna el resultado a coetm2t
   Realiza ZEROS(n+2) y asigna el resultado a coetm3t
   Realiza ZEROS(n+2) y asigna el resultado a coetm4t

   FOR 1 < i < 2
      IF it == 1
         IF i == 1
            Asigna it + 4 a coetm1(1,i); Asigna it + 4 a coetm2(1,i)
            Asigna it + 4 a coetm3(1,i); Asigna it + 4 a coetm4(1,i)

         ELSEIF i == 2
            Asigna tK1 a coetm1(1,i); Asigna tK1 a coetm2(1,i)
            Asigna tK1 a coetm3(1,i); Asigna tK1 a coetm4(1,i)
         END IF
      IF it2 > 1
         IF i == 1
            Calcula it + 4 y asigna el resultado a coetm1t(1,i);
            Calcula it + 4 y asigna el resultado a coetm2t(1,i);
            Calcula it + 4 y asigna el resultado a coetm3t(1,i);
            Calcula it + 4 y asigna el resultado a coetm4t(1,i);
         ELSEIF i == 2
            Asigna tK1 a coetm1t(1,i); Asigna tK1 a coetm2t(1,i)
            Asigna tK1 a coetm3t(1,i); Asigna tK1 a coetm4t(1,i)
         END IF
      END IF
   END FOR
   Asigna coetg1(fsc1,3:n+2) a coetg1; Asigna coetg2(fsc1,3:n+2) a coetg2;
   Asigna coetg3(fsc1,3:n+2) a coetg3; Asigna coetg4(fsc1,3:n+2) a coetg4;
END
```

```
    IF it == 1
       Asigna coetg1 a coetm1(1,3:n+2); Asigna coetg2 a coetm2(1,3:n+2);
       Asigna coetg1 a coetm3(1,3:n+2); Asigna coetg4 a coetm4(1,3:n+2);

    ELSEIF it>=2
       Asigna coetg1 a coetm1t(1,3:n+2); Asigna coetg2 a coetm2t(1,3:n+2);
       Asigna coetg1 a coetm3t(1,3:n+2); Asigna coetg4 a coetm4t(1,3:n+2);
       Une coetm1 y coetm1t y asigna el resultado a coetm1
       Une coetm2 y coetm2t y asigna el resultado a coetm2
       Une coetm3 y coetm3t y asigna el resultado a coetm3
       Une coetm4 y coetm4t y asigna el resultado a coetm4
    END IF

    Agrupa y, p, cof, phicf, cat_mayor, n y pathDir y asigna el resultado a l
    Evalúa rootSecantePlus2A(t0K, te2,1,0.001,2,store=%T) y asigna el resultado a
    vri
    Calcula las dimensiones de vri y asigna el resultado a s1
    Asigna s1(1) a f1;
    Asigna vri(f1,2) a tK1p
    Evalúa ABS(tK1p-tK1) y asigna el resultado a delt; Asigna tK1p a tK1
    Calcula it + 1 y asigna el resultado a it

    IF it > imaxt
       Sale del ciclo
    END IF
END WHILE

Calcula p/100 y asigna el resultado a p0phi

IF op_gid ~= 0
    Evalúa coac(opa,prop,tK,x,n,pathDir) y asigna el resultado a cof
    IF op_gid == 2
       Guarda param(1) en B; Guarda param(2) en opi; Guarda param(3) en opB
       Calcula phiCM(tK1,p0phi,cat_mayor,B,opi,opB,n,pathDir,y) y asigna el
       resultado a phicf
    END IF
END IF

Calcula las dimensiones de coetm1 y asigna el resultado a sc3;
Asigna sc3(1) a fsc3;
IF fsc3 == 1
    Realiza ZEROS(1:3,1:n+2) y asigna el resultado a coetf1
    Realiza ZEROS(1:3,1:n+2) y asigna el resultado a coetf2
    Realiza ZEROS(1:3,1:n+2) y asigna el resultado a coetf3
```

Realiza **ZEROS(**1:3,1:**n**+2**)** y asigna el resultado a **coetf4**

Almacena **coetma1** a **coetf1**(1:3,1:**n**+2)
Almacena **coetma2** a **coetf2**(1:3,1:**n**+2)
Almacena **coetma3** a **coetf3**(1:3,1:**n**+2)
Almacena **coetma4** a **coetf4**(1:3,1:**n**+2)

Almacena **coetm1** a **coetf1**(4,1:**n+2**); Almacena **coetm2** a **coetf2**(4,1:**n+2**)
Almacena **coetm3** a **coetf3**(4,1:**n+2**); Almacena **coetm4** a **coetf4**(4,1:**n+2**)

ELSEIF **fsc3** >= 2
Realiza **ZEROS(fsc3**+3,1:**n**+2**)** y asigna el resultado a **coetf1**
Realiza **ZEROS(fsc3**+3,1:**n**+2**)** y asigna el resultado a **coetf2**
Realiza **ZEROS(fsc3**+3,1:**n**+2**)** y asigna el resultado a **coetf3**
Realiza **ZEROS(fsc3**+3,1:**n**+2**)** y asigna el resultado a **coetf4**

Almacena **coetma1** a **coetf1**(1:3,1:**n**+2)
Almacena **coetma2** a **coetf2**(1:3,1:**n**+2)
Almacena **coetma3** a **coetf3**(1:3,1:**n**+2)
Almacena **coetma4** a **coetf4**(1:3,1:**n**+2)

Almacena **coetm1** a **coetf1**(4:**fsc3**+3,1:**n+2**);
Almacena **coetm2** a **coetf2**(4:**fsc3**+3,1:**n+2**)
Almacena **coetm3** a **coetf3**(4:**fsc3**+3,1:**n+2**);
Almacena **coetm4** a **coetf4**(4:**fsc3**+3,1:**n+2**)
END

Agrupa **coetf1**, **coetf2**, **coetf3** y **coetf4** y asigna el resultado a **coetf_gru**
Asigna “t” a **car1**; Asigna “x” a **car2**; Asigna “Tem” a **car3**; Asigna “\\RocioT” a **nfile**

Asigna “Rocio T” a **name**
Evalúa **imp_op_tab(coetf_gru,n,car1,car2,car3,pathDir,imp,nfile,name)**
Agrupa "rocio", **p**, **t, xte**, **y**, **cof** y **phicf** y asigna el resultado a **catalog**
Devuelve **catalog**

END FUNCTION

opcK

FUNCTION **mat_carB(n,car,opi)**

FOR 1 < **i** < **n**

```
      Crea una cadena de caracteres vacía y la asigna a st2A(i)
      FOR 1 < j < n
        Crea una cadena de caracteres vacía y la asigna a stA
        Utilizando stA, crea una cadena de caracteres a ser impresa
        Une stA y st2A(i) y guarda el resultado a en st2A(i)
      END FOR
   END FOR

   FOR 1 < i < n
      FOR 1 < j < n
         Crea una cadena de caracteres vacía y la asigna a st2B(i,j)
      END FOR
   END FOR

   FOR 1 < i < n
      Calcula STRSPLIT(st2A(i), ", ") y asigna el resultado a st2A_aux
      FOR 1 < j < n
         Almacena st2A_aux(j) en st2B(i,j)
      END FOR
   END FOR
   PRINT st2B

END FUNCTION
```

```
FUNCTION ingreso_constantesB(car,car2,st1,tpc)

   Calcula LENGTH(car) y asigna el resultado a n
   Asigna 2 a ind
   WHILE ind == 2
      FOR 1 < i < n
         IF tpc ~= 0
            IF i == 1
               Asigna tpc(1) a prop_c(i)
            ELSEIF i == 2
               Asigna tpc(2) a prop_c(i)
            ELSE
               INPUT propiedad correspondiente a car(i) del compuesto st1 y asigna el
                     resultado a prop_c(i)
            END IF
         ELSEIF tpc ==0
               INPUT propiedad correspondiente a car(i) del compuesto st1 y asigna el
                     resultado a prop_c(i)
         END
      END FOR
```

```
    PRINT "propiedades correctas del compuesto st1? 1. Si  2. No"
    PRINT "car(i) = prop_c(i)?"
    INPUT número entre 1 y 2 y asigna el resultado a ind
  END WHILE

END FUNCTION
```

```
FUNCTION mat_valB(n,car,opi)

  Asigna 2 a ind
  WHILE ind == 2
    Realiza ZEROS(n,n) y asigna el resultado a Bcof
    Crea un vector vacío y asigna el resultado a B_stv
    Crea una lista vacía y asigna el resultado a sub_l
    Asigna 1 a k
    FOR 1 < i < n
      FOR 1 < j < n
        IF j == i
          INPUT coeficiente del virial B(i,j) y asigna el resultado a B_stv(k)
          Agrupa i y j y asigna el resultado a sub_l(k)
          Calcula k + 1 y asigna el resultado a k
          Asigna B_stv(k) a Bcof(i,j); Asigna Bcof(i,j) a Bcof(j,i)
        END IF
        IF opi == 2
          IF j > i
            INPUT coeficiente del virial B(i,j) y asigna el resultado a B_stv(k)
            Agrupa i y j y asigna el resultado a sub_l(k)
            Calcula k + 1 y asigna el resultado a k
            Asigna B_stv(k) a Bcof(i,j); Asigna Bcof(i,j) a Bcof(j,i)
          END IF
        END IF
      END FOR
    END FOR

    Calcula LENGTH(ni) y asigna el resultado a ni
    PRINT "Datos correctos? 1. Si  2. No"
    FOR 1 < k < ni
      Guarda sub_l(k) y asigna el resultado a ij; Asigna ij(1) a i; Asigna ij(1) a j;
      PRINT "Coeficiente del virial B(i,j)"
      PRINT B_stv(k)
    END FOR
    INPUT número entre 1 y 2 y asigna el resultado a ind
  END WHILE
```

```
END FUNCTION
```

```
FUNCTION B_tab(n, name, opi, opB, catalog)

  IF opB ==1
    Asigna "La temperatura crítica (tc) en K" a cad(1)
    Asigna "La presión crítica (pc) en bar" a cad(2)
    Asigna "El volumen crítico (vc) en cm^3/mol" a cad(3)
    Asigna "El factor de compresibilidad crítico (zc)" a cad(4)
    Asigna "El factor acéntrico (ω)" a cad(5)
    Asigna "(Tc K)" a cad(1)
    Asigna "(Pc bar)" a cad(2)
    Asigna "(Vc en cm^3/mol)" a cad(3)
    Asigna "?" a cad(4)
    Asigna "?" a cad(5)

    FOR 1 < i < n
      Asigna catalog(i) a catalog_i
      Asigna catalog_i(3) a bec

      IF LENGTH(bec) == 2
        Asigna bec(1) a bk; Asigna bec(2) a eco
        IF bk == 1
          IF eco == 1
            Asigna catalog_i(4) a cpre
            Asigna cpre(2) a tc; Asigna cpre(3) a pc
            Agrupa a tc y pc y asigna el resultado a tpc
            Evalúa ingreso_constantesB(cad,cad2,name(i),tpc) y asigna el resultado
            a prBt
          END IF
        END IF
      ELSE
        Asigna 0 a tpc
        Evalúa ingreso_constantesB(cad,cad2,name(i),tpc) y asigna el resultado a
        prBt
      END IF
      Asigna prB(i,:) a prBt
    END FOR
     Asigna prB(:,:) a B

  ELSEIF opB == 2
    IF opi == 1
      Asigna "B" a car
      Evalúa mat_carB(n,car,opi)
```

```
      Evalúa mat_valB(n,car,opi) y asigna el resultado a B
    ELSEIF opi == 2
      Asigna "B" a car
      Evalúa mat_carB(n,car,opi)
      Evalúa mat_valB(n,car,opi) y asigna el resultado a B
    END IF
  END
  Devuelve B

END FUNCTION
```

```
FUNCTION opcK(op_gid,op_p,n,names,pathDir,p,t,z,imp,param_sr)

  Utilizando pathDir, establece el directorio de coac y asigna el resultado a dir1
  Utilizando a dir1, llama al espacio de trabajo a coac, desde coac
  Utilizando pathDir, establece el directorio de cpre_ing y asigna el resultado a dir2_1
  Utilizando a dir2_1, llama al espacio de trabajo a cpre_ing, desde cpre_ing

  FOR  1 < i < n
    PRINT "Libro a ocupar en la búsqueda de las constantes para el cálculo de las
           presiones de vapor, del compuesto names(i)"
    INPUT número entre 1, 2, 3 ó 4 y asigna el resultado a bk(i)
  END FOR

  Crea una lista vacía y asigna el resultado a catalog

  FOR 1 < i < n
    PRINT "Ingreso de constantes para el compuesto names(i), opción bk(i)"
    Evalúa cpre_ing(pathDir,p,t,names(i),bk(i)) y asigna el resultado a catalog(i)
  END FOR

  IF op_gid ~= 0
    PRINT "método para el cálculo de los coeficientes de actividad"
    INPUT número entre 1, 2 o 3 y asigna el resultado a opa

    IF opa ~= 3
      Evalúa in_dat_cof(opa,n,names) y asigna el resultado a prop
    ELSEIF opa == 3
      Agrupa pathDir y n y asigna el resultado a dirnum
      Evalúa in_dat_cof(opa,dirnum,names) y asigna el resultado a prop
    END IF
  ELSE
    Asigna 0 a opa; Asigna 0 a prop
  END IF
```

Asigna **z** a **x**; Asigna **z** a **y**
IF **op_p** == 1
Utilizando **pathDir**, establece el directorio de BurbP y asigna el resultado **dir1_1**
Utilizando **pathDir**, establece el directorio de RoP y asigna el resultado **dir2_1**

IF **op_gid** == 0 OR **op_gid** == 1
Asigna 0 a **param**
Utilizando **dir1_1**, llama al espacio de trabajo a **yBurbp**, desde BurbP
Evalúa **yBurbp(x,t,catalog,opa,prop,op_gid,pathDir,n,imp,param)** y asigna el resultado a **cat1**

Utilizando **dir2_1**, llama al espacio de trabajo a **xRop**, desde RoP
Evalúa **xRop(y,t,catalog,opa,prop,op_gid,pathDir,n,imp,param)** y asigna el resultado a **cat2**

Agrupa **cat1**, **cat2**, **catalog**, **opa** y **prop** y asigna el resultado a **catalog2**

ELSEIF **op_gid** == 2
Asigna **param_sr**(1) a **opi**; Asigna **param_sr** a **opB**
Evalúa **B_tab(n,names,opi,opB,catalog)** a **B**; Agrupa **B**, **opi**, **opB** y asigna el resultado a **param**

Utilizando **dir1_1**, llama al espacio de trabajo a **yBurbp**, desde BurbP
Evalúa **yBurbp(x,t,catalog,opa,prop,op_gid,pathDir,n,imp,param)** y asigna el resultado a **cat1**

Utilizando **dir2_1**, llama al espacio de trabajo a **xRop**, desde RoP
Evalúa **xRop(y,t,catalog,opa,prop,op_gid,pathDir,n,imp,param)** y asigna el resultado a **cat2**

Agrupa **cat1**, **cat2**, **catalog**, **opa**, **prop** y **B** y asigna el resultado a **catalog2**
END IF

ELSEIF **op_p** == 2
Utilizando **pathDir**, establece el directorio de BurbP y asigna el resultado **dir1_1**
Utilizando **pathDir**, establece el directorio de RoP y asigna el resultado **dir2_1**

IF **op_gid** == 0 OR **op_gid** == 1
Asigna 0 a **param**
Utilizando **dir1_1**, llama al espacio de trabajo a **yBurbt**, desde Burbt
Evalúa **yBurbt(x,t,catalog,opa,prop,op_gid,pathDir,n,imp,param)** y asigna el resultado a **cat1**

Utilizando **dir2_1**, llama al espacio de trabajo a **xRot**, desde Rot
Evalúa **xRot(y,t,catalog,opa,prop,op_gid,pathDir,n,imp,param)** y asigna el resultado a **cat2**

Agrupa **cat1**, **cat2**, **catalog**, **opa** y **prop** y asigna el resultado a **catalog2**

ELSEIF **op_gid** == 2
Asigna **param_sr**(1) a **opi**; Asigna **param_sr** a **opB**
Evalúa **B_tab(n,names,opi,opB,catalog)** a **B**; Agrupa **B**, **opi**, **opB** y asigna el resultado a **param**

Utilizando **dir1_1**, llama al espacio de trabajo a **yBurbt**, desde BurbT
Evalúa **yBurbt(x,t,catalog,opa,prop,op_gid,pathDir,n,imp,param)** y asigna el resultado a **cat1**

Utilizando **dir2_1**, llama al espacio de trabajo a **xRot**, desde RoT
Evalúa **xRot(y,t,catalog,opa,prop,op_gid,pathDir,n,imp,param)** y asigna el resultado a **cat2**

Agrupa **cat1**, **cat2**, **catalog**, **opa, prop** y **B** y asigna el resultado a **catalog2**
END IF
END IF

END FUNCTION

opdK

FUNCTION **opdK(op_gid, ,op_p, ct)**

Asigna **ct**(1) a **lbur**; Asigna **ct**(2) a **lroc**
IF **op_gid** == 0
IF **op_p** ==1
Asigna **lbur**(2) a **pbur**; Asigna **lroc**(2) a **proc**
Agrupa **pbur** y **proc** y asigna el resultado a **vp**
Asigna **ct**(3) a **cat_mayor**; Agrupa **vp** y **cat_mayor** y asigna el resultado a **lpK**
ELSEIF **op_p** == 2
Asigna **lbur**(2) a **tbur**; Asigna **lroc**(2) a **troc**
Agrupa **tbur** y **troc** y asigna el resultado a **vt**
Asigna **ct**(3) a **cat_mayor**; Agrupa **vt** y **cat_mayor** y asigna el resultado a **lpK**
END IF
ELSEIF **op_gid** == 1
IF **op_p** ==1
Asigna **lbur**(2) a **pbur**; Asigna **lroc**(2) a **proc**

Agrupa **pbur** y **proc** y asigna el resultado a **vp**
Asigna **ct**(3) a **cat_mayor**;
Asigna **lbur**(6) a **cofbur**; Asigna **lroc**(6) a **cofroc**
Agrupa **cofbur** y **cofroc** y asigna el resultado a **vcof**
Asigna **ct**(4) a **lopa_cact**; Asigna **ct**(5) a **lprop_cact**;
Agrupa **lopa_cact** y **lprop_cact** y asigna el resultado a **lpar_cact**

Agrupa **vt**, **cat_mayor, lpar_cact y vcof** y asigna el resultado a **lpK**

ELSEIF **op_p** == 2
Asigna **lbur**(2) a **tbur**; Asigna **lroc**(2) a **troc**
Agrupa **tbur** y **troc** y asigna el resultado a **vt**
Asigna **ct**(3) a **cat_mayor**;
Asigna **lbur**(6) a **cofbur**; Asigna **lroc**(6) a **cofroc**
Agrupa **cofbur** y **cofroc** y asigna el resultado a **vcof**
Asigna **ct**(4) a **lopa_cact**; Asigna **ct**(5) a **lprop_cact**;
Agrupa **lopa_cact** y **lprop_cact** y asigna el resultado a **lpar_cact**

Agrupa **vt**, **cat_mayor, lpar_cact y vcof** y asigna el resultado a **lpK**

END IF

ELSEIF **op_gid** == 2
IF **op_p** ==1
Asigna **lbur**(2) a **pbur**; Asigna **lroc**(2) a **proc**
Agrupa **pbur** y **proc** y asigna el resultado a **vp**
Asigna **ct**(3) a **cat_mayor**;
Asigna **lbur**(6) a **cofbur**; Asigna **lroc**(6) a **cofroc**
Agrupa **cofbur** y **cofroc** y asigna el resultado a **vcof**
Asigna **ct**(4) a **lopa_cact**; Asigna **ct**(5) a **lprop_cact**;
Agrupa **lopa_cact** y **lprop_cact** y asigna el resultado a **lpar_cact**
Agrupa **phibur** y **phiroc** y asigna el resultado a **vphi**
Asigna **ct**(6) a **lpar_phi**;

Agrupa **vt**, **cat_mayor, lpar_cact, vcof, lpar_phi** y **vphi** y asigna el resultado a **lpK**

ELSEIF **op_p** == 2
Asigna **lbur**(2) a **tbur**; Asigna **lroc**(2) a **troc**
Agrupa **tbur** y **troc** y asigna el resultado a **vt**
Asigna **ct**(3) a **cat_mayor**;
Asigna **lbur**(6) a **cofbur**; Asigna **lroc**(6) a **cofroc**
Agrupa **cofbur** y **cofroc** y asigna el resultado a **vcof**
Asigna **ct**(4) a **lopa_cact**; Asigna **ct**(5) a **lprop_cact**;
Agrupa **lopa_cact** y **lprop_cact** y asigna el resultado a **lpar_cact**
Agrupa **phibur** y **phiroc** y asigna el resultado a **vphi**

```
        Asigna ct(6) a lpar_phi;

        Agrupa vt, cat_mayor, lpar_cact, vcof, lpar_phi y vphi y asigna el resultado a
        lpK
      END IF
   END IF
END FUNCTION
```

Vaporz_ins

```
FUNCTION vpsat(t,cat_mayor, pathDir)

   Utilizando pathDir, establece el directorio de fpres_vap y asigna el resultado a dir1
   Utilizando dir1, llama al espacio de trabajo a Psat, desde fpres_vap
   Calcula las dimensiones de cat_mayor y asigna el resultado a s1
   Asigna 1 a s1

   WHILE i <= s1
      Evalúa Psat(t,cat_mayor(i)) y asigna el resultado a psat(i)
      Calcula i + 1 y asigna el resultado a i
   END WHILE

END FUNCTION
```

```
FUNCTON cx(z,K,v)

   Calcula la dimensiones de z y asigna el resultado a s1
   Asigna s1(2) a col; Asigna 1 a i

   Calcula 1 + v. *(K-1) y asigna el resultado a denx
   Calcula z ./denx y asigna el resultado a valorx

   Evalúa SUM(valorx) y asigna el resultado a sum_x
   FOR 1 < i < col
      Calcula valorx(i)/sum_x y asigna el resultado a valorx
   END FOR
END FUNCTION
```

```
FUNCTION valini(op_gid,prsis,prburb,prroc,param)

   Calcula (prsis-prburb)/(prroc-prburb) y asigna el resultado a frac; Asigna frac a v
   IF op_gid == 0
```

Coloca en una lista a **v** y asigna el resultado a **est_ini**
ELSEIF **op_gid** == 1
Asigna **param**(1) a **coets**; Asigna **coets**(1) a **coet_burb**; Asigna **coets**(2) a **coet_roc**
Calcula **frac**.*(**coet_roc-coet_burb**)+**coet_burb** y asigna el resultado a **coini**
Agrupa **v**, **coini** y **phini** y asigna el resultado a **est_ini**
ELSEIF **op_gid** == 2
Asigna **param**(1) a **coets**; Asigna **coets**(1) a **coet_burb**; Asigna **coets**(2) a **coet_roc**
Asigna **param**(1) a **phis**; Asigna **phis**(1) a **phi_burb**; Asigna **phis**(2) a **phi_roc**
Calcula **frac**.*(**coet_roc-coet_burb**)+**coet_burb** y asigna el resultado a **coini**
Calcula **frac**.*(**phi_roc-phi_burb**)+**phi_burb** y asigna el resultado a **phi_burb**
Agrupa **v**, **coini** y **phini** y asigna el resultado a **est_ini**
END IF

END FUNCTION

FUNCTION **obtK(op_gid,psis,tsis,psat,n,pathDir,param)**

Utilizando **pathDir**, establece el directorio de coac y asigna el resultado a **dir1_1**
Utilizando **pathDir**, llama al espacio de trabajo a **coac**, desde coac

Asigna **param**(1) a **lpar_act**
Asigna **lpar_act**(1) a **opa**; Asigna **lpar_act**(2) a **prop**; Asigna **lpar_act**(3) a **valorx**

IF **op_gid** == 1
Agrupa a **opa**, **prop** y **xant** y asigna el resultado a **lpar_cat2**
Evalúa **coac(opa,prop,tsis,valorx,n,pathDir)** y asigna el resultado a **cof**
Calcula **cof**.***psat**/**psis** y asigna el resultado a **K**;

ELSEIF **op_gid** == 2
Utilizando **pathDir**, establece el directorio de phiCM y asigna el resultado a **dir2_1**
Utilizando **dir2_1**, llama al espacio de trabajo a **phiCM**, desde phiCM
Asigna **param**(2) a **lpar_phi**
Asigna **lpar_phi**(1) a **cat_mayor**; Asigna **lpar_phi**(2) a **B**; Asigna **lpar_phi**(3) a **opi**
Asigna **lpar_phi**(4) a **opB**; Asigna **lpar_phi**(5) a **y**
Evalúa **coac(opa,prop,tsis,valorx,n,pathDir)** y asigna el resultado a **cof**
Calcula **psis**/100 y asigna el resultado a **p0phi**
Evalúa **phiCM(tsis,p0phi,cat_mayor,B,opi,opB,n,pathDir,y)** y asigna el resultado a **phi**
Calcula **cof**.***psat**/(**phi***psis**)** y asigna el resultado a **K**;
END IF

END FUNCTION

```
FUNCTION fK(v,op_gid,K,z)

   Asigna 1 a i
   WHILE i <= col
      IF i == 1
         Calcula 1+v*(K(i) – 1) y asigna el resultado a den
         Calcula z(i)*(K(i) – 1) y asigna el resultado a num
         Calcula num/den y asigna el resultado a funK
      ELSE
         Calcula 1+v*(K(i) – 1) y asigna el resultado a den
         Calcula z(i)*(K(i) – 1) y asigna el resultado a num
         Calcula num/den y asigna el resultado a funK
      END IF
      Calcula i + 1 y asigna el resultado a i
   END WHILE

END FUNCTION
```

```
FUNCTION vi(op_gid,op_p,psis,tsis,z,lpk,n,pathDir,imp,param)

   Asigna tsis a tsisK
   Asigna lpk(1) a vprop; Asigna vprop(1) a vprbur; Asigna vprop(2) a vprroc
   Asigna lpk(2) a cat_mayor

   IF op_gid == 0
      IF op_p == 1
         Asigna vprbur a pbur; Asigna vprroc a proc
         IF psis < pbur
            IF psis>proc
               Asigna 0 a param1
               Evalúa valini(op_gid,psis,pbur,proc,param1) y asigna el resultado a
               est_ini
               Asigna 0 a param2
               Evalúa calc_int(op_gid,psis,tsisK,z,cat_mayor,est_ini,n,pathDir,
               imp,param2) y asigna el resultado a res
            ELSE
               PRINT "El sistema existe como vapor sobrecalentado"
               Asigna 0 a res
            END IF
         ELSE
            PRINT "El sistema existe como liqudio subenfriado"
            Asigna 0 a res
         END IF
      ELSEIF op_p == 2
```

```
      Asigna vprbur a tbur; Asigna vprroc a troc
      IF tsisK > tbur
        IF tsisK < troc
          Asigna 0 a param1
          Evalúa valini(op_gid,tsisK,tbur,troc,param1) y asigna el resultado a
          est_ini
          Asigna 0 a param2
          Evalúa calc_int(op_gid,psis,tsisK,z,cat_mayor,est_ini,n,pathDir,
          imp,param2) y asigna el resultado a res
        ELSE
          PRINT "El sistema existe como vapor sobrecalentado"
          Asigna 0 a res
        END IF
      ELSE
        PRINT "El sistema existe como liqudio subenfriado"
        Asigna 0 a res
      END IF
    END IF

  ELSEIF op_gid == 1
    Asigna lpk(3) a lpar_act; Asigna lpk(4) a vcof;
    Coloca a vcof en una lista y asigna el resultado a param1
    Coloca a lpar_act en una lista y asigna el resultado a param2
    IF op_p == 1
      Asigna vprbur a pbur; Asigna vprroc a proc
      IF psis < pbur
        IF psis>proc
          Evalúa valini(op_gid,psis,pbur,proc,param1) y asigna el resultado a
          est_ini
          Evalúa calc_int(op_gid,psis,tsisK,z,cat_mayor,est_ini,n,pathDir,
          imp,param2) y asigna el resultado a res
        ELSE
          PRINT "El sistema existe como vapor sobrecalentado"
          Asigna 0 a res
        END IF
      ELSE
        PRINT "El sistema existe como liqudio subenfriado"
        Asigna 0 a res
      END IF
    ELSEIF op_p == 2
      Asigna vprbur a tbur; Asigna vprroc a troc
      IF tsisK > tbur
        IF tsisK < troc
          Evalúa valini(op_gid,tsisK,tbur,troc,param1) y asigna el resultado a
```

est_ini
Evalúa **calc_int(op_gid,psis,tsisK,z,cat_mayor,est_ini,n,pathDir, imp,param2)** y asigna el resultado a **res**
ELSE
PRINT "El sistema existe como vapor sobrecalentado"
Asigna 0 a **res**
END IF
ELSE
PRINT "El sistema existe como liqudio subenfriado"
Asigna 0 a **res**
END IF

END IF
END IF

ELSEIF **op_gid** == 2
Asigna **lpk**(3) a **lpar_act**; Asigna **lpk**(4) a **vcof**; Asigna **lpk**(5) a **lpar_phi**
Asigna **lpk**(5) a **vphi**; Asigna **param**(1) a **opi**; Asigna **param**(2) a **opB**
Agrupa **lpar_phi**, **opi** y **opB** y asigna el resultado a **lpar_phi2**
Agrupa **vcof y vphi** y asigna el resultado a **param1**
Agrupa **lpar_act** y **lpar_phi2** y asigna el resultado a **param2**
IF **op_p** == 1
Asigna **vprbur** a **pbur**; Asigna **vprroc** a **proc**
IF **psis** < **pbur**
IF **psis**>**proc**
Evalúa **valini(op_gid,psis,pbur,proc,param1)** y asigna el resultado a **est_ini**
Evalúa **calc_int(op_gid,psis,tsisK,z,cat_mayor,est_ini,n,pathDir, imp,param2)** y asigna el resultado a **res**
ELSE
PRINT "El sistema existe como vapor sobrecalentado"
Asigna 0 a **res**
END IF
ELSE
PRINT "El sistema existe como liqudio subenfriado"
Asigna 0 a **res**
END IF
ELSEIF **op_p** == 2
Asigna **vprbur** a **tbur**; Asigna **vprroc** a **troc**
IF **tsisK** > **tbur**
IF **tsisK** < **troc**
Evalúa **valini(op_gid,tsisK,tbur,troc,param1)** y asigna el resultado a **est_ini**
Evalúa **calc_int(op_gid,psis,tsisK,z,cat_mayor,est_ini,n,pathDir,**

```
            imp,param2) y asigna el resultado a res
          ELSE
            PRINT "El sistema existe como vapor sobrecalentado"
            Asigna 0 a res
          END IF
        ELSE
          PRINT "El sistema existe como liqudio subenfriado"
          Asigna 0 a res
        END IF

      END IF
    END IF
END FUNCTION
```

```
FUNCTION calc_int(op_gid,psis,tsisK,z,cat_mayor,estini,n,pathDir,imp,param)

  Utilizando pathDir, establece el directorio de rootSecantePlus2Ap2 y asigna el
  resultado a dir2
  Utilizando dir2, llama al espacio de trabajo a rootSecantePlus2A, desde
  rootSecantePlus2Ap2
  Utilizando pathDir, establece el directorio de imp_op_tab_Kcal y asigna el
  resultado a dir2
  Utilizando dir2, llama al espacio de trabajo a imp_op_tab_Kcal, desde
  imp_op_tab_Kcal

  Calcula las dimensiones de z y asigna el resultado a s1
  Realiza ONES(col) y asigna el resultado a delx;
  Realiza ONES(col) y asigna el resultado a dely
  Evalúa ONES(col) y asigna el resultado a delKg

  IF op_gid ~= 0
    Asigna param(1) a lpar_act; Asigna lpar_act(1) a opa; Asigna lpar_act(2) a prop

    IF op_gid == 2
      Asigna param(2) a lpar_phi; Asigna lpar_phi(1) a B; Asigna lpar_phi(2) a opB
    END IF
  END IF
  Evalúa vpsat(tsisK,cat_mayor,pathDir) y asigna el resultado a psat
  Asigna 1 a iw; Asigna 1 a opw
  WHILE opw == 1
    IF iw == 1
      Asigna estini(1) a v; Asigna 1 a delv
      IF op_gid == 0
        Realiza ONES(n) y asigna el resultado a coet
```

```
      Realiza ONES(n) y asigna el resultado a phi
    ELSEIF op_gid ~= 0
      Asigna estini(2) a coet; Asigna ONES(n) a phi
      IF op_gid == 2
        Asigna estini(3) a phi
      END IF
    END IF
    Calcula psat.*coet./(phi*psis) y asigna el resultado a Knuevo
    Evalúa cx(z,Knuevo,v) y asigna el resultado a xnuevo
    Calcula Knuevo.*xnuevo y asigna el resultado a ynuevo
    Asigna xnuevo a xant; Asigna ynuevo a yant; Asigna Knuevo a Kant
    Une iw, v, xnuevo y asigna el resultado a tabx
    Une iw, v, ynuevo y asigna el resultado a taby
    Une iw, v y Knuevo y asigna el reultado a tabk
  ELSE
    IF op_gid == 0
      Calcula psat/psis y asigna el resultado a Knuevo
    ELSEIF op_gid == 1
      Agrupa opa, prop y xant y asigna el resultado a param
      Evalúa obtK(1,psis,tsisK,psat,n,pathDir,param) y asigna el resultado a
      Knuevo
    ELSEIF op_gid == 2
      Agrupa opa, prop y xant y asigna el resultado a lpar_cat2
      Agrupa cat_mayor, B, opi, opB y yant y asigna el resultado a lpar_phi2
      Evalúa obtK(2,psis,tsisK,psat,n,pathDir,param) y asigna el resultado a
      Knuevo
    END IF
    Agrupa a op_gid, Knuevo y z y asigna el resultado a l1; Asigna 0.0001 a tol
    Asigna 30 a imax
    Evalúa rootSecantePlus2A(v,fK,l1,tol,imax,store=%T) y asigna el resultado a
    vri1
    Calcula las dimensiones de vri1 y asigna el resultado a s1
    Asigna s1(1) a f1; Almacena vri1(f1,2) a v1
    Evalúa ABS(v1-v) y asigna el resultado a delv; Asigna v1 a v
    Evalúa cx(z,Knuevo,v) y asigna el resultado a xnuevo
    Calcula xnuevo – xant y asigna el resultado a delx
    Calcula Knuevo.*xnuevo y asigna el resultado a ynuevo
    Calcula ynuevo – yant y asigna el resultado a dely
    Calcula Knuevo – Kant y asigna el resultado a delK
    Evalúa NORM(delx) y asigna el resultado a delxg;
    Evalúa NORM(dely) y asigna el resultado a delyg
    Evalúa NORM(delK) y asigna el resultado a delKg
    Asigna xnuevo a xant; Asigna ynuevo a yant; Asigna Knuevo a Kant
    Une iw, v, xnuevo y asigna el resultado a tabx
```

```
      Une iw, v, ynuevo y asigna el resultado a taby
      Une iw, v y Knuevo y asigna el reultado a tabk
    END IF
    IF v > 0 & v < 1
      IF delxg > 0.0001 AND delxg > 0.0001 AND delxg > 0.0001 AND delxg >
        0.0001
        Asigna 1 a opw
      ELSE
        Asigna 2 a opw
      END IF
    END IF
    Calcula iw + 1 y asigna el resultado a iw
    IF iw > 50
      Sale del ciclo
    END IF
  END WHILE
  Agrupa tabx, taby y tabk y asigna el resultado a tab_gru
  Asigna "V" a car1; Asigna "x" a car2_1; Asigna "y" a car2_2; Asigna "K" a car2_3
  Asigna "\VapInst" a nfile; Asigna "Vap Inst" a name
  Evalúa imp_op_tab_Kcal(tab_gru,n,car1,car2_1,car2_2,pathDir,imp,nfile,name)
  Agrupa v, xnuevo, ynuevo y Knuevo y asigna el resultado a results

END FUNCTION
```

imp_op_tab_Kcal

```
FUNCTION
imp_op_tab_Kcal(tab_g2,n,car1,car2_1,car2_2,car2_3,car3,pathDir,imp,nfile,
name1)

  IF LENGTH(imp) ==2
    Asigna imp(1) a op_imp; Asigna imp(2) a datn
  ELSE
    Asigna imp(1) a op_imp
  END IF
  Utilizando pathDir, establece la ruta del directorio de opcionesK y asigna el resultado
  a dir1

  Utilizando dir1, llama al espacio de trabajo a comptab_impK desde imp_tab
  Evalua comptab_impK(n, car1, car2_1, car3), y asigna el resultado a una lista
  formada por tab1A y tab1C
  Evalua comptab_impK(n, car1, car2_2, car3), y asigna el resultado a una lista
  formada por tab2A y tab2C
```

Evalua **comptab_impK(n**, **car1**, **car2_3, car3)**, y asigna el resultado a una lista formada por **tab3A** y **tab3C**

Agrupa **tab1A**, **tab2A** y **tab3A** y asigna el resultado a **tabA**
Agrupa **tab1C**, **tab2C** y **tab3C** y asigna el resultado a **tabC**

IF **op_imp** == 1
PRINT "Tablas de cálculo del punto de **name1**" a **st_titulo**
PRINT **tab1A**, **tab1C;** PRINT **tab_g2**(1)
PRINT **tab2A**, **tab2C;** PRINT **tab_g2**(2)
PRINT **tab3A**, **tab3C;** PRINT **tab_g2**(3)

ELSEIF **op_imp** == 2
Utilizando **pathDir**, crea o utiliza la carpeta "Archivos_usuario" y guarda el resultado en **dir1**
Utiliza **dir1**, **datn** y **nfile** para formar una ruta especifica según **nfile**, y guarda el resultado en **dir2**
Utilizando **dir2**, crea una carpeta

Agrupa "valores_x","valores_y","valores_K" y asigna el resultado a **cad**

FOR 1 < **i** < 4
Utiliza **dir2** y **name2** para formar la ruta del archivo csv donde se guardaran los datos y lo almacena en **filename**
Utiliza **tabB**(**i**) y de **tabC**(**i**) y **coetf_g2**(i) para crear títulos y escribir los datos de **coetf_g2**(**i**) en el archivo **filename**
END IF
PRINT "los archivos del punto de **name1** están guardados en:"
PRINT **dir2**

END FUNCTION

Kdatos

FUNCTION **names(n,st)**

FOR 1 < **i** < **n**
INPUT nombre del compuesto **st** y asigna el resultado a **idn**(**i**)
END FOR
Devuelve **idn**

END FUNCTION

FUNCTION **cad_comp_imp(car,n,idn)**

Crea un vector vacío y asigna el resultado a **st1**;
Crea un vector vacío y asigna el resultado a **st2**
Crea una cadena de caracteres vacía y asigna el resultado a **st3**
Crea una cadena de caracteres vacía y asigna el resultado a **st4**

FOR 1 < **i** < **n**
Utiliza **car** e **i** para construir parte de una cadena de caracteres a imprimir y la asigna **st1**(**i**)
Utiliza **car** e **i** para construir parte de una cadena de caracteres a evaluar y la asigna al **st2**(**i**)
Une **st3** y **st1**(**i**) en una cadena de caracteres a imprimir y asigna el resultado a **st3**
Une **st4** y **st2**(**i**) en una cadena de caracteres a evaluar y asigna el resultado a **st4**
END FOR

END FUNCTION

FUNCTION **com_ing(n, name, car, sf_aux1, sf_aux2, sf_aux3)**

Crea una cadena de caracteres vacia y la asigna a **st2**
Crea un vector vacío y lo asigna a **st**

FOR 1 < **i** < **n**
Utiliza **car** e **i** para construir parte de una cadena de caracteres a imprimir y la asigna a **st**(**i**)
Une **st**(**i**) y **st2** en una cadena de caracteres a imprimir y asigna el resultado a **st2**
END FOR
Realiza **STRSPLIT(st2, " ")** y asigna el resultado a **st2**
IF **n** == 2
PRINT **sf_aux1**(1); PRINT **sf_aux1**(2);
ELSEIF **n** == 3
PRINT **sf_aux1**(1); PRINT **sf_aux1**(2); PRINT **sf_aux1**(3);
ELSEIF **n** == 4
PRINT **sf_aux1**(1); PRINT **sf_aux1**(2); PRINT **sf_aux1**(3); PRINT **sf_aux1**(4)
END IF

Asigna 1 a **ind**
WHILE **ind** == 1
FOR 1 < **i** < **n**-1
INPUT propiedad correspondiente a **st2**(**i**) y asigna el resultado a **c_st**(**i**)
END FOR

Calcula 1 – **SUM(c_st)** y asigna el resultado a **c_st**

```
      IF c_comp > 0
         Asigna 2 a ind
      ELSEIF c_comp <= 0
         Asigna 1 a ind
         PRINT sf_aux3
      END IF
   END WHILE

   Une c_st y c_comp y asigna el resultado a c_st
   Asigna c_st(:) y asigna el resultado a c(:)
   Devuelve c

END FUNCTION
```

```
FUNCTION Kdatos(r_prin)

   Asigna 2 a ind
   WHILE ind == 2
      PRINT "Selección del número de componentes: 2, 3 o 4"
      INPUT número entre 2, 3 o 4 y asigna el resultado a opC
      PRINT "Selección del tipo de cálculo a realizar: 1. Puntos de Burbuja y de Rocío P
                2. Puntos de Burbuja y Rocío T"
      INPUT número entre 2, 3 o 4 y asigna el resultado a op_p
      PRINT "Selección de la ecuación para el cálculo de vaporización instantánea"
      INPUT número entre 1 o 2 y asigna el resultado a op_gid
      PRINT "Ingreso de los nombres de los compuestos"
      Asigna "compuesto" a st
      Evalúa names(opC,st) y asigna el resultado a nameco
      Calcula op_gid – 1 y asigna el resultado a op_gid

      IF op_gid ==2
         PRINT "Cuenta con coeficientes del virial? 1. Si  2. No"
         INPUT número entre 1 y 2 y asigna el resultado a opB
         IF opB == 1
            Asigna 2 a opB
         ELSEIF opB == 2
            Asigna 1 a opB
         END IF
         PRINT "fase vapor en estado ideal? 1. Si  2. No"
         INPUT número entre 1 y 2 y asigna el resultado a opi
      ELSE
         Asigna 1 a opB; Asigna 1 a opi
      END IF
```

```
    PRINT "número y nombres de componentes, ecuación a utilizar e información
          ingresada de lo coeficientes del virial, correcta?"
    INPUT número entre 1 y 2 y asigna el resultado a ind
END WHILE

Asigna 2 a ind
WHILE ind == 2
   INPUT Presión en Kpa y asigna el resultado a p
   WHILE %T
      IF p < 0
         INPUT Presión en Kpa y asigna el resultado a p
      ELSE
         Sale del ciclo
      END IF
   END WHILE

   INPUT Temperatura en K y asigna el resultado a t
   WHILE %T
      IF t < 0
         INPUT Presión en Kpa y asigna el resultado a t
      ELSE
         Sale del ciclo
      END IF
   END WHILE

   Asigna "z" a car
   IF n ==2
      Asigna "Ingrese la composición molar car del nameco(1)" a sf_aux1
   ELSEIF n ==3
      Asigna "Ingrese la composición molar car del nameco(1) y nameco(2)" a
      sf_aux1
   ELSEIF n ==4
      Asigna "Ingrese la composición molar car del nameco(1), nameco(2) y
      nameco(3)" a sf_aux1
   END IF

   Asigna "Ingrese una cantidad numérica positiva" y asigna el resultado a sf_aux2
   Asigna "La suma de las composiciones car debe ser igual a 1" y asigna el resultado
   a sf_aux3
   Evalúa com_ing(n,nameco,car,sf_aux1,sf_aux2,sf_aux3) y asigna el resultado a z
   PRINT "Presión, temperatura y composiciones correctas? 1. Si  2. No"
   INPUT número entre 1 y 2 y asigna el resultado a ind
END WHILE
```

PRINT “imprimir los resultados iterativos del cálculo de punto de burbuja”
PRINT “1. Consola de Scilab 2. Archivo .csv 3. Últimos cálculos realizados”
INPUT número entre 1, 2 o 3 y asigna el resultado a **op_imp**

IF op_imp == 2
  Calcula **INT(RAND())***10000 y asigna el resultado a **nw**
ELSE
  Coloca a **op_imp** en una lista y asigna el resultado a **imp**
END IF

Utilizando **r_prin**, establece el directorio de opcK y asigna el resultado a **r1**
Utilizando **r1**, llama al espacio de trabajo a **opcK** desde opcK

IF **op_gid** == 0 **OR op_gid** == 1
  Asigna 0 a **param**
ELSEIF **op_gid** ==2
  Agrupa a **opi** y **opB** y asigna el resultado a **param**
END IF

Evalúa **opcK(op_gid,op_p,n,nameco,r_prin,p,t,z,imp,param)** y asigna el resultado a **ct**

Utilizando **r_prin**, establece el directorio de opdK y asigna el resultado a **r3**
Utilizando **r3**, llama al espacio de trabajo a **opdK** desde opdK
Evalúa **opdK(op_gid,op_p,ct)** y asigna el resultado a **lpk**

Utilizando **r_prin**, establece el directorio de Vaporz_ins y asigna el resultado a **r4**
Utilizando **r4**, llama al espacio de trabajo a **vi** desde Vaporz_ins

Evalúa **vi(op_gid,op_p,p,t,z,lpk,n,r_prin,imp,param)** y asigna el resultado a **resultados**
IF **TYPE(resultados)** == 1
  PRINT “Pruebe diferentes condiciones de presiones y temperaturas”
ELSE
  Asigna **resultados**(1) a **v**; Asigna **resultados**(2) a **x**; Asigna **resultados**(3) a **y**
  Asigna **resultados**(4) a **K**
  PRINT “Las moles de vapor son”; PRINT **v**

  Asigna “x” a **car**
  Evalúa **cad_comp_imp(car,n,nameco)** y asigna el resultado a una lista formada por **stxcar** y **stximp**
  Utilizando **stxcar**, construye una cadena de caracteres y asigna el resultado a **stxc**
  Asigna “y” a **car**
  Utilizando **stxcar**, construye una cadena de caracteres y asigna el resultado a **styc**

```
      Evalúa cad_comp_imp(car,n,nameco) y asigna el resultado a una lista formada por
      stycar y styimp
      Utilizando stkcar, construye una cadena de caracteres y asigna el resultado a stkc
      Asigna "K" a car
      Utilizando stkcar, construye una cadena de caracteres y asigna el resultado a stkc
      Evalúa cad_comp_imp(car,n,nameco) y asigna el resultado a una lista formada por
      stkcar y stkimp
      Utilizando stkcar, construye una cadena de caracteres y asigna el resultado a styc

      PRINT stxc, EVSTR(stximp)
      PRINT styc, EVSTR(styimp)
      PRINT stkc, EVSTR(stkimp)
   END IF

END FUNCTION
```