



Universidad Nacional Autónoma de México

Facultad de Ingeniería

División de Ciencias de la Tierra

Departamento de Ingeniería Petrolera

Ingeniería de Yacimientos de Gas

Proyecto 1

"Análisis Composicional"

Presenta: Ramírez Salazar Isaura

Semestre 2021-1

Fecha de entrega: 17 de noviembre de 2020

El análisis composicional del gas permite conocer sus propiedades, tales como viscosidad, factor de volumen del gas, compresibilidad, factor de desviación, temperaturas y presiones críticas, densidad relativa, etc.

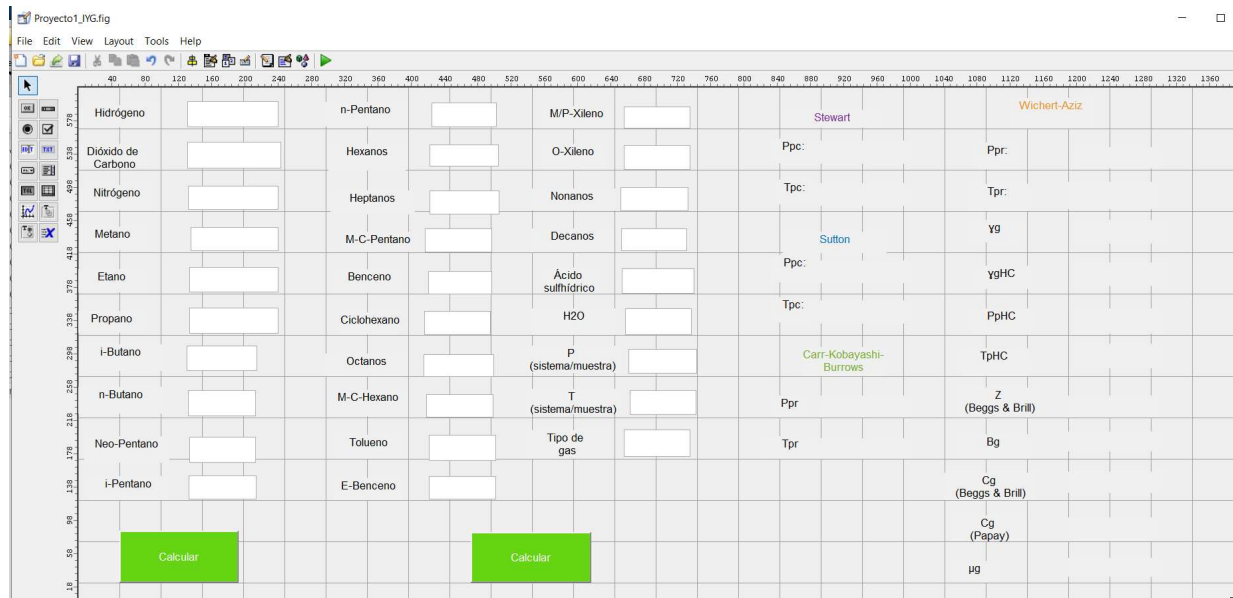
A lo largo de la historia, se han desarrollado diferentes métodos, tanto gráficos como numéricos, para llevar a cabo cálculos que nos permitan conocer dichas propiedades.

En este trabajo se utilizarán las correlaciones de Carr-Kobayashi-Burrows, Wichert-Aziz, Sutton, Lee , Stewart, Burkhard y Voo, Papay y Beggs y Brill para calcular y corregir temperaturas y presiones pseudocríticas, pseudoreducidas, el cálculo de la viscosidad del gas, densidad del gas, el factor de volumen del gas, el factor de desviación Z y la viscosidad, así como los cálculos intermedios necesarios para el análisis composicional del pozo Nejo-17.

Las propiedades de presión y temperatura utilizadas fueron las indicadas en el archivo "Análisis Composicional de Gas", las cuales son 150[psig] y 29.0°C, respectivamente, del mismo archivo fueron tomados los valores de peso molar y fracción mol.

Primero se realizó la interfaz de usuario, se insertaron los "edit text" donde el usuario puede ingresar el valor de la fracción mol, los "text" a modo de etiqueta para indicar al usuario a qué componente corresponde cada casilla de "edit text", se cambiaron las etiquetas con las que se identifica cada *edit text* para que fuera más sencilla su manipulación, posteriormente se agregaron dos botones, los cuales permiten calcular las condiciones pseudo críticas por el Método de Sutton y por el método de Stewart, Burkhard y Voo.

Se añadió una casilla en la cual, el usuario ingresará el tipo de gas, haciendo referencia a la correlación de Sutton, las opciones que se pueden teclear son "natural", "condensado" y "natural Sutton", se evalúa la entrada por medio de expresiones regulares y de este modo se utiliza la ecuación pertinente para el cálculo de P_{pHC} y T_{pHC} , que serán utilizados para calcular P_{pcm} y T_{pcm} .



Una vez lista la interfaz se prosiguió a la programación del código con las correlaciones necesarias.

Se declararon las variables para manejar los datos ingresados en cada botón para que cada uno pudiera manejar los datos ingresados y así realizara los cálculos correspondientes.

```

Editor - D:\Isaura\IYG_Programas\Proyecto1\Proyecto1_IYG.m
aes_demo.m x Proyecto1_IYG.m x +
467 % --- Executes on button press in SuttonButton.
468 function SuttonButton_Callback(hObject, eventdata, handles)
469 % hObject    handle to SuttonButton (see GCBO)
470 % eventdata  reserved - to be defined in a future version of MATLAB
471 % handles    structure with handles and user data (see GUIDATA)
472
473 H2 = str2double(get(handles.H2e, 'string')); %
474 CO2 = str2double(get(handles.CO2e, 'string'));
475 N2 = str2double(get(handles.N2e, 'string'));
476 C1 = str2double(get(handles.C1e, 'string'));
477 C2 = str2double(get(handles.C2e, 'string'));
478 C3 = str2double(get(handles.C3e, 'string'));
479 iC4 = str2double(get(handles.iC4e, 'string'));
480 nC4 = str2double(get(handles.nC4e, 'string'));
481 C5 = str2double(get(handles.C5e, 'string'));
482 iC5 = str2double(get(handles.iC5e, 'string'));
483 nC5 = str2double(get(handles.nC5e, 'string'));
484 C6 = str2double(get(handles.C6e, 'string'));
485 C7 = str2double(get(handles.C7e, 'string'));
486 MC5 = str2double(get(handles.MC5e, 'string'));
487 Benceno = str2double(get(handles.Bencenoe, 'string'));
488 Ciclohexano = str2double(get(handles.Ciclohexanoe, 'string'));
489 C8 = str2double(get(handles.C8e, 'string'));
490 MC6 = str2double(get(handles.MC6e, 'string'));
491 Tolueno = str2double(get(handles.Toluenoe, 'string'));
492 C9 = str2double(get(handles.C9e, 'string'));
493 EBenceno = str2double(get(handles.EBencenoe, 'string'));
494 MPX = str2double(get(handles.MPXe, 'string'));
495 OX = str2double(get(handles.OXe, 'string'));
496 C10 = str2double(get(handles.C10e, 'string'));

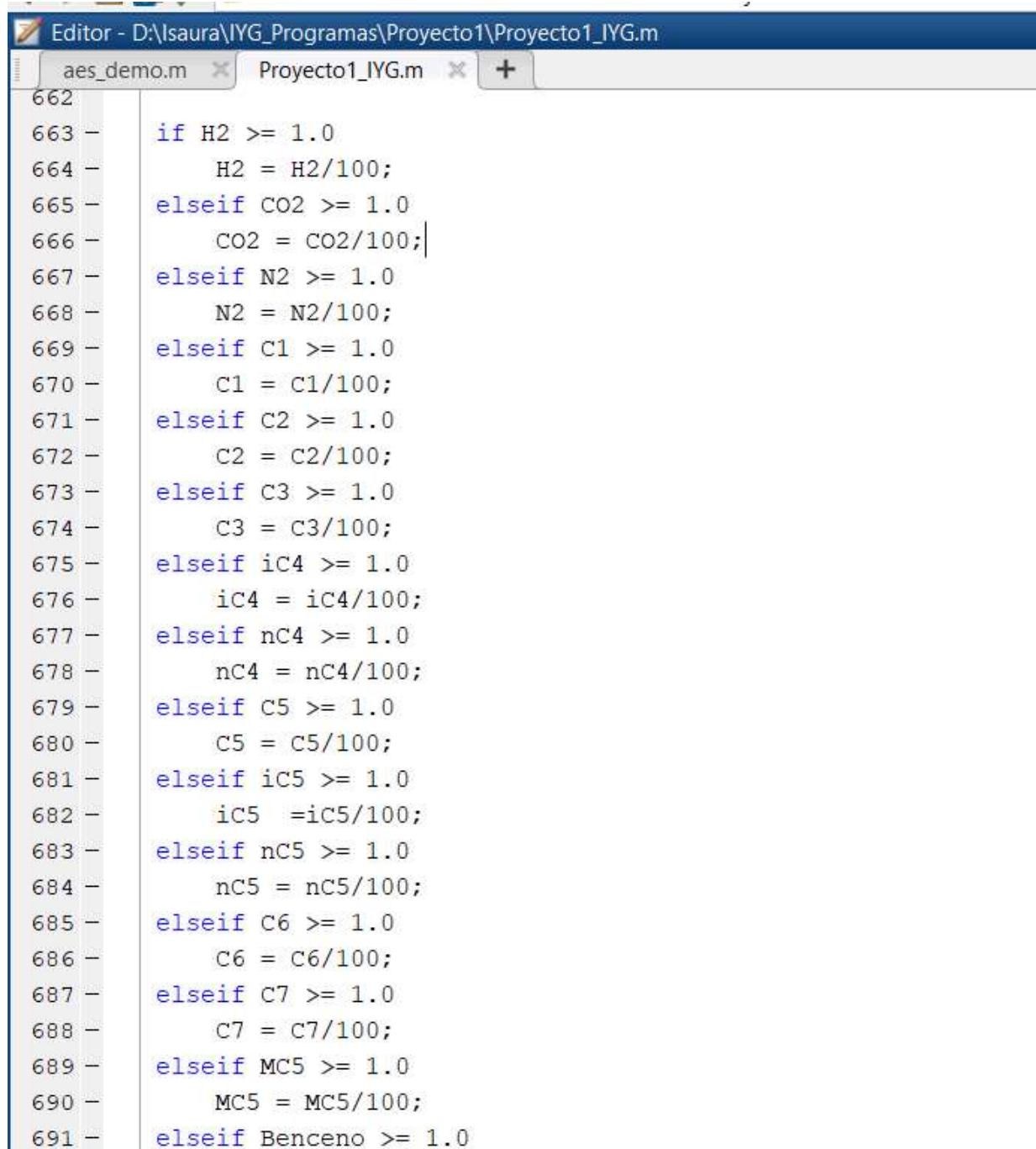
```

```

497 - H2S = str2double(get(handles.H2Se , 'string'));
498 - P = str2double(get(handles.Psme , 'string'));
499 - T = str2double(get(handles.Tsme , 'string'));
500 - H2O = str2double(get(handles.H2Oe , 'string'));
501 - TDG = get(handles.TDGe, 'string');
502
503

```

Se comprueba que las fracciones molares ingresadas sean menores a uno (no estén dadas en porcentaje), si son mayores, se asume que se están ingresando en porcentaje y son divididas entre 100.



The screenshot shows a MATLAB editor window titled "Editor - D:\Isaura\IYG_Programas\Proyecto1\Proyecto1_IYG.m". The code is as follows:

```

662
663 - if H2 >= 1.0
664 -     H2 = H2/100;
665 - elseif CO2 >= 1.0
666 -     CO2 = CO2/100;
667 - elseif N2 >= 1.0
668 -     N2 = N2/100;
669 - elseif C1 >= 1.0
670 -     C1 = C1/100;
671 - elseif C2 >= 1.0
672 -     C2 = C2/100;
673 - elseif C3 >= 1.0
674 -     C3 = C3/100;
675 - elseif iC4 >= 1.0
676 -     iC4 = iC4/100;
677 - elseif nC4 >= 1.0
678 -     nC4 = nC4/100;
679 - elseif C5 >= 1.0
680 -     C5 = C5/100;
681 - elseif iC5 >= 1.0
682 -     iC5 = iC5/100;
683 - elseif nC5 >= 1.0
684 -     nC5 = nC5/100;
685 - elseif C6 >= 1.0
686 -     C6 = C6/100;
687 - elseif C7 >= 1.0
688 -     C7 = C7/100;
689 - elseif MC5 >= 1.0
690 -     MC5 = MC5/100;
691 - elseif Benceno >= 1.0

```

Se agrupan los valores correspondientes a C7+, C8+ y C9+

```

Editor - D:\Isaura\IYG_Programas\Proyecto1\Proyecto1_IYG.m
aes_demo.m x Proyecto1_IYG.m x +
713 - end
714
715 - C7p = MC5 + Benceno + Ciclohexano + C7;
716 - C8p = MC6 + Tolueno + C8;
717 - C9p = EBenceno + MPX + OX + C9;

```

Se crea un vector que contiene todas las fracciones, llamado yi

```

719 - yi(:,1) = [H2,CO2,H2S,N2,C1,C2,C3,iC4,nC4,C5,iC5,nC5,C6,C7p,C8p,C9p,C10];

```

Se crean dos archivos de texto plano con extensión ".txt". Uno contendrá los valores de temperatura y presión crítica de cada componente, y el otro alojará los valores de peso molecular.

```

710 %Se añaden txt's con Tc,Pc y Peso molecular
711 - TyP = load('TyPcríticas.txt');
712 - Tc = TyP(:,1);
713 - Pc = TyP(:,2);
714 - save('TyPcríticas','Tc','Pc');
715 - load('TyPcríticas','Tc','Pc');
716
717 - global PM
718
719 - Pmol = load('PesoMol.txt');
720 - PM = Pmol(:,1);
721 - save('PesoMol','PM');
722 - load('PesoMol','PM');

```

Para utilizar el método de Stewart, Burkhard y Voo

$$J = \frac{1}{3} \left[\sum_{i=1}^n y_i \left(\frac{T_c}{P_c} \right)_i + \frac{2}{3} \left[\sum_{i=1}^n y_i \left(\frac{T_c}{P_c} \right)_i^{\frac{1}{2}} \right]^2 \right]$$

$$K = \sum_{i=1}^n y_i \left(\frac{T_c}{\sqrt{P_c}} \right)_i$$

$$T_{pc} = \frac{K^2}{J}$$

$$P_{pc} = \frac{T_{pc}}{J}$$

Se crearon variables para no escribir las ecuaciones completas de manera repetitiva a las que llamaremos V1, V2 y V3, para llevar a cabo las sumas, se crearon variables de nombre Vs, Vs2 y Vs3


```

724 %Variables para almacenar la suma de V1, V2 y V3
725 Vs = 0;
726 Vs2 = 0;
727 Vs3 = 0;
728 for i = 1:length(yi)
729     V1(i,1) = (yi(i,1).*(Tc(i,1)./(Pc(i,1))));
730     V2(i,1) = (yi(i,1).*((Tc(i,1)./(Pc(i,1))).^(1./2)));
731     V3(i,1) = (yi(i,1).*(Tc(i,1)./(Pc(i,1).^(1./2))));
732     Vs = V1(i,1) + Vs;
733     Vs2 = V2(i,1) + Vs2;
734     Vs3 = V3(i,1) + Vs3;
735     M = Pmol(i,1)*yi(i,1);
736 end
737 disp(Vs)
738 disp(Vs2)
739 disp(Vs3)
740 %Método Stewart, Bukhard y Voo
741 %Se calcula J y K
742 J = ((1/3).*Vs) + (2/3).*(Vs2).^2;
743 K = Vs3

```

Una vez obtenidas J y K, se calculan las propiedades pseudo críticas y se muestran en la interfaz.

```

749 Tpc = K.^2./J
750 Ppc = Tpc./J
751 set(handles.PpcRes, 'string', Ppc)
752 set(handles.TpcRes, 'string', Tpc)
753

```

Se realiza la correlación de Sutton para corregir los parámetros J y K.

```

756 %Correlación de Sutton
757 FJ = (1./3).*(C7p+C8p+C9p).*(Tc(14,1)./(Pc(14,1))) + (2./3).*((C7p+C8p+C9p).*(Tc(14,1)./(Pc(14,1))).^(1./2)).^2
758 epsj = (0.6081.*FJ) + (1.1325.*(FJ.^2)) - (14.004.*FJ.*(C7p+C8p+C9p)) + (664.434.*FJ.*(C7p+C8p+C9p).^2)
759 epsk = (Tc(14,1)./(Pc(14,1).^(1./2))).*(0.3129.*(C7p+C8p+C9p)) - (4.8156.*((C7p+C8p+C9p).^2)) + (27.3751.*((C7p+C8p+C9p).^3))
760 Jp = J - epsj
761 Kp = K - epsk
762 Tpc = (Kp.^2)/Jp
763 Ppc = Tpc/Jp
764

```

Se corrigen las propiedades pseudocríticas por el método de Carr-Kobayashi-Burrows

```

764
765 %Carr-Kobayashi-Burrows
766 Tppc = Tpc - (80.*yi(2,1)) + (130.*yi(3,1)) - (250.*(yi(4,1)));
767 Pppc = Ppc + (440.*yi(2,1)) + (600.*yi(3,1)) - (170.*yi(4,1));
768

```

Se calculan las propiedades pseudoreducidas con las propiedades pseudocríticas corregidas

```

770 - Ppr = P./Pppc
771 - Tpr = T./Tppc
772

```

Se calcula el factor de desviación Z con la correlación de Beggs y Brill

```

773 %Correlación por método de Beggs & Brill para Z
774 - Ab = (1.39.*((Tpr-0.92).^0.5)) - (0.36.*Tpr) - 0.101;
775 - Eb = 9.*(Tpr-1)
776 - Bb = ((0.62-(0.23.*Tpr))*Ppr)+(((0.066)./(Tpr-0.86))-0.037)*(Ppr.^2)+((0.32.*(Ppr.^6))/(10.^Eb))
777 - Cb = 0.132-0.32.*log10(Tpr)
778 - Fb = 0.3106-(0.49.*Tpr)+(0.1824*(Tpr.^2))
779 - Db = 10.^Fb;
780 - Zb = Ab+((1-Ab)./(exp(Bb)))+(Cb.*(Ppr).^Db)
781 - set(handles.ZCKBRes, 'string', Zb)
782

```

En el segundo botón, se llevan a cabo los cálculos de la correlación de Wichert-Aziz, donde se corrigen de modo distinto las propiedades pseudo críticas de gases con impurezas.

```

506 %Método de Wichert-Azis corrección de propiedades pc
507 - A = H2S + CO2;
508 - B = H2S;
509 - eps = 120.*(A.^0.9)-A.^1.6)+15.*(B.^0.5)-(B.^4.0));
510 - Tprimpc = Tpcc - eps; %Tpcc
511 - Pprimpc = (Ppcc.*Tprimpc)/(Tpcc+H2S.*(1-H2S).*eps); %Ppcc
512 |
513 - Pprw = P./Pprimpc;
514 - Tprw = T./Tprimpc;

```

Para poder calcular las densidades, es necesario conocer el resultado de multiplicar la fracción molar por el peso molecular, debido a que, tanto el peso molar como la fracción se encuentran en vectores, dentro de un ciclo *for* se realiza la multiplicación elemento por elemento y posteriormente la suma de esos resultados.

```

523 - SM = 0;
524
525 - for j = 1:length(PM)
526 -     M(j,1) = (PM(j,1).*yi(j,1))
527 -     SM = M(j,1) + SM;
528 - end
529

```

Posteriormente se realiza el cálculo de las densidades.

```

533 - Gammagm = SM./28.96;
534 - disp("Gammagm")
535 - disp(Gammagm)
536 - set(handles.GammagRes, 'string', Gammagm)
537
538 - GammaHC = (Gammagm-(1.1767.*H2S)-(1.5196.*CO2)-(0.9672.*N2)-(0.622.*H2O))/(1-H2S-CO2-N2-H2O);
539 - set(handles.GammaghcRes, 'string', GammaHC)
540

```

Se hace uso de las expresiones regulares y sentencias *if* y *elseif* para poder decidir qué ecuaciones deben ser utilizadas para el cálculo de la presión y temperatura

```

Editor - D:\Isaura\IYG_Programas\Proyecto1\Proyecto1_IYG.m
aes_demo.m x Proyecto1_IYG.m x Untitled2* x Untitled3* x Untitled4* x +
539 set(handles.BgRes, 'string', GammaHC);
540
541 %Se prueba el tipo de gas
542 - gasec = 'condensado';
543 - gasen = 'natural';
544 - gasens = 'natural Sutton';
545 - startIndex = regexp(TDG, gasec);
546 - startIndex2 = regexp(TDG, gasen);
547 - startIndex3 = regexp(TDG, gasens);
548
549 %Gas condensado
550 - if startIndex >= 1
551 -     PpcHC = 706 + (51.7.*GammaHC) - (11.1.*((GammaHC).^2));
552 -     TpcHC = 187 + (330.*GammaHC) - (71.5.*((GammaHC).^2));
553 -     disp(TDG)
554
555 %Gas natural
556 - elseif startIndex2 >= 1
557 -     PpcHC = 677 + (15.*GammaHC) - (37.5.*((GammaHC).^2));
558 -     TpcHC = 168 + (325.*GammaHC) - (12.5.*((GammaHC).^2));
559 -     disp(TDG)
560
561 %Gas natural(Sutton)
562 - elseif startIndex3 >= 1
563 -     PpcHC = 765.8 + (131.0.*GammaHC) - (3.6.*((GammaHC).^2));
564 -     TpcHC = 169.2 + (349.5.*GammaHC) - (74.0.*((GammaHC).^2));
565 -     disp(TDG)
566 - end
567

```

Y una vez que se han elegido y resuelto dichas ecuaciones, se corrigen

```

573
574 %Concentraciones
575 - PpcM = (1-H2S-CO2-N2-H2O).*PpcHC + (1306.*H2S) + (1071.*CO2) + (493.1.*N2) + (3200.1.*H2O);
576 - TpcM = (1-H2S-CO2-N2-H2O).*TpcHC + (672.35.*H2S) + (547.58.*CO2) + (227.16.*N2) + (1164.9.*H2O);
577

```

Se realiza el cálculo del factor de volumen del gas y se muestra en la interfaz.

```

585 %Cálculo de Bg
586 - Bg = (0.02829.*Zb.*T)/(P)
587
588 - set(handles.BgRes, 'string', Bg);
589

```

Se realiza el cálculo de la compresibilidad del gas tanto con la correlación de Papay como con la de Beggs y Brill.


```

589
590 %Compresibilidad por correlación de Papay
591 - dzdpP = -(3.52./(10.^(0.9813.*Tprw))) + ((0.548.*Pprw)./(10.^(0.8157.*Tprw)));
592 - CgP = (1./Pprw) - ((1./Zb).*dzdpP);
593 - set(handles.CgPRes, 'string', CgP);
594
595 %Compresibilidad por Beggs&Brill
596 - dzdpB = (-((1-Ab)./(((0.62-0.23.*Tprw)+(0.132./(Tprw-0.86))-0.74).*Pprw + ((1.92.*(Pprw.^5)./(10.^(9.*(Tprw-1)))))).*exp(Bb)))+(Cb.*Db.*(Pprw.^(Db-1))));
597 - CgB = (1./Pprw) - ((1./Zb).*dzdpB);
598 - set(handles.CgBRes, 'string', CgB);
599

```

Y finalmente se realiza el cálculo de la viscosidad del gas por medio de la correlación de Lee.

```

599
600 %Viscosidad del gas por correlación de Lee
601 - Rom = (1.4935.*(10.^-3)).*(P.*SM)./(Zb.*T);
602 - Km = (9.379+(0.01607.*SM).*(T.^1.5))/(209.2+(19.26.*SM)+T);
603 - Xm = 3.448+(986.4/T)+(0.01009.*SM);
604 - Ym = 2.447 - (0.2224.*Xm);
605 - mugg = (1.*(10.^-4)).*Km.*exp(Xm.*(Rom.^Ym));
606 - set(handles.MugRes, 'string', mugg);
607

```

Ingresando los datos dados en el análisis del pozo Nejo-17, recordando convertir la temperatura a Rankine ($29^{\circ}\text{C} = 84.2^{\circ}\text{F} = 544.2\text{R}$) y presión en PSI (150), convirtiendo el porcentaje molar a fracción molar y considerando el gas como condensado se obtuvo:

Composición del Gas			Propiedades de los Componentes		Correlaciones	
Hidrógeno	0	n-Pentano	0.0042	M/P-Xileno	0	Stewart
Dióxido de Carbono	0.31	Hexanos	0.0040	O-Xileno	0	Wichert-Aziz
Nitrógeno	0.1094	Heptanos	0.0012	Nonanos	0.0002	
Metano	0.8078	M-C-Pentano	0.0003	Decanos	0	Sutton
Etano	0.0329	Benceno	0.0001	Ácido sulfhídrico	0	
Propano	0.0178	Ciclohexano	0.0005	H2O	0	
i-Butano	0.055	Octanos	0.0006	P (sistema/muestra)	150	Carr-Kobayashi-Burrows
n-Butano	0.0084	M-C-Hexano	0.0002	T (sistema/muestra)	544.2	
Neo-Pentano	0.0001	Tolueno	0	Tipo de gas	condensado	
i-Pentano	0.0037	E-Benceno	0			

Propiedad	Valor
Ppc:	477.435
Tpc:	455.985
Ppr:	0.332572
Tpr:	1.26173
Yg:	1.2552
YgHC:	1.1683
PpHC:	751.25
TpHC:	474.947
Z (Beggs & Brill):	0.969366
Bg:	0.099492
Cg (Beggs & Brill):	6.23987
Cg (Papay):	3.19915
μg:	0.000530871

Conclusión

El análisis de un gas real es complicado y programarlo aún más si no se tiene cuidado con las variables, paréntesis, sintaxis, etc, al momento de programar, es importante ya que gracias a este análisis podemos conocer propiedades importantes del gas con el que se tratará y esto permitirá que se tomen mejores decisiones en cuanto a la producción y recuperación del hidrocarburo.

Mi aprendizaje con este proyecto ha sido que nosotros como petroleros podemos desarrollar software que nos sea útil, que hay muchas correlaciones y es complicado saber cuál es más conveniente aplicar o saber dónde aplicarlas.

Para mi habría sido de utilidad el saber mejor qué datos son los que deben usarse en los txt y comprender un poco más el cómo manipular la información correspondiente a los C7+, ya que generalmente en los ejercicios (no solo de ésta materia) se nos da como un solo dato, no desglosado, entonces saber qué presión y temperatura crítica tomar fue algo que me costó trabajo.

Gracias por su tiempo para resolver dudas profesor.