```
classdef BOObj < handle
%BOObj. Black Oil Class
% BOObj represents a black oil model. In each property, there is a
% reference of the correlations used for the calculation.
%
%   Properties
%    - P            : pressure                          [psia]
%    - T            : temperature                       [F]
%    - Tbgsc        : bubble point temperature gas @SC  [F]
%    - Tbosc        : bubble point temperature oil @SC  [F]
%    - GOR          : gas in oil ratio                  []
%    - WC           : water cut                         [%]
%    - gSG          : specific gravity gas              [-]
%    - oSG          : specific gravity oil              [-]
%    - wSG          : specific gravity water            [-]
%    - RichardsonO  : Richardson coefficient            [-]
%    - RichardsonW  : Richardson coefficient            [-]
%    - F01          : tuning factor
%    - F02          : tuning factor
%    - F03          : tuning factor
%    - F04          : tuning factor
%    - F05          : tuning factor
%    - F06          : tuning factor
%    - F07          : tuning factor
%
%   Dependent properties
%    - status       : programming property
%    - API          : API grade oil                     [-]
%    - Bg           : formation volume gas              [rcf/stb]
%    - Bo           : formation volume oil              [rbl/stb]
%    - Bw           : formation volume water            [rbl/stb]
%    - Co           : oil compressibily factor          [1/psi]
%    - oCp          : specific heat capacity oil        [BTU/lb.F]
%    - gCp          : specific heat capacity gas        [BTU/lb.F]
%    - wCp          : specific heat capacity water      [BTU/lb.F]
%    - Pb           : bubble point pressure             [psia]
%    - Rs           : gas/oil in solution ratio         [scf/stb]
%    - Z            : compressibility factor            [-]
%    - grho         : density gas                       [lb/ft3]
%    - orho         : density oil                       [lb/ft3]
%    - wrho         : density water                     [lb/ft3]
%    - emu          : dynamic viscosity emulsion        [cP]
%    - gmu          : dynamic viscosity gas             [cP]
%    - omu          : dynamic viscosity oil             [cP]
%    - wmu          : dynamic viscosity water           [cP]
%    - Tsc          : pseudo critic temperature         [R]
%    - Psc          : pseudo critic pressure            [psia]
%    - rhosc2ac     : conversion matrix
%    - qsc2ac       : conversion matrix
%
%   Methods
%    - TuneViscosity
```

```
%    - ClearTuneViscosity
%
 ------------------------------------------------------------------------
% Development
%   By: Ruben Ensalzado
%       2015
%   Rev 00 151216 original release
%   Rev 01 160117 inclusion property: rhosc2ac, qsc2ac
%                 inclusion method: TuneViscosity
%                 inclusion method: ClearTuneViscosity
    properties
        P
        T
        Tbgsc
        Tbosc
        GOR
        WC
        gSG
        oSG
        wSG
        RichardsonO
        RichardsonW
        F01
        F02
        F03
        F04
        F05
        F06
        F07
    end

    properties (Hidden = true)
        pcF05
        pcF06
    end

    properties (Dependent = true)
        status
        API
        Bg
        Bo
        Bw
        Co
        Pb
        Rs
        Z
        grho
        orho
        wrho
        gmu
        omu
        wmu
        emu
        gCp
```

```
        oCp
        wCp
        Psc
        Tsc
        rhosc2ac
        qsc2ac
    end

    methods (Hidden = true)
        % -- initialization method --

        function BO = BOObj
            BO.F01 = 1;
            BO.F02 = 1;
            BO.F03 = 1;
            BO.F04 = 1;
            BO.F05 = 1;
            BO.F06 = 1;
            BO.F07 = 1;

            BO.pcF05 = [];
            BO.pcF06 = [];

            BO.gSG = 0.64;
            BO.oSG = 141.5/(30 + 131.5);
            BO.wSG = 1.02;

            BO.WC = 0;

            BO.RichardsonO = 3.215;
            BO.RichardsonW = 3.089;
        end
    end

    methods
        % -- dependent properties --

         function status = get.status(BO)
            %Status. Independent property verification

            req = zeros(7, 1);

            req(1) = isempty(BO.P);
            req(2) = isempty(BO.T);
            req(3) = isempty(BO.GOR);
            req(4) = isempty(BO.WC);
            req(5) = isempty(BO.gSG);
            req(6) = isempty(BO.oSG);
            req(7) = isempty(BO.wSG);

            if all(~req)
                status = true;
            else
                status = false;
```

```
    end
end

function API = get.API(BO)
    %APIGravity. Definition

    API = 141.5/BO.oSG - 131.5;
end

function Bg = get.Bg(BO)
    %GasFormationVolume. Mass balance
    %Units
    % T   : F
    % P   : psia
    % Bg  : rcf/scf

    Bg = (14.7/520)*BO.Z*(BO.T + 460)/BO.P;
end

function Bo = get.Bo(BO)
    %OilFormationVolume. Frick correlation
    %Units
    % T   : F
    % P   : psia
    % Pb  : psia
    % Bo  : rbl/stb
    % Co  : 1/psi
    % GOR : scf/stb

    SGr = BO.gSG/BO.oSG;

    if BO.Rs <= BO.GOR
        F = BO.Rs*sqrt(SGr) + 1.25*BO.T;
        Bo = 0.972*BO.F02 + 1.47e-4*BO.F03*F^1.175;
    else
        F = BO.GOR*sqrt(SGr) + 1.25*BO.T;
        Boc = 0.972*BO.F02 + 1.47e-4*BO.F03*F^1.175;
        Bo = Boc*exp(BO.Co*(BO.Pb - BO.P));
    end
end

function Bw = get.Bw(BO)
    %WaterFormationVolume. Frick correlation
    %Units
    % T   : F
    % P   : psia
    % Bw  : rbl/stb

    DP = BO.P - 14.7;
    DT = BO.T - 60;

    Bw = (1 + 1.25e-4*DT + 9.88e-7*DT^2)*(1 - 3e-6*DP);
end
```

```
        function Co = get.Co(BO)
            %OilCompressibility. Beggs and Vasquez correlation
            %Units
            % T   : F
            % P   : psia
            % Pb  : psia
            % GOR : scf/stb
            % Co  : 1/psi

            if BO.P <= BO.Pb
                Co = 0;
            else
                gSGb = BO.gSG*(1 +
5.912e-5*BO.oSG*60*log10(14.7/114.7));
                Co = BO.F04*(-1433 + 5*BO.GOR + 17.2*BO.T -
(1180*gSGb) + 12.61*BO.API)/(1e5*BO.P);
            end
        end

        function gCp = get.gCp(BO)
            %HeatCapacity. (gas) Watson-Nelson (1933)
            %Units
            % T   : F
            % Tb  : F
            % gCp : BTU/lb.F

            Tb = BO.Tbgsc + 460;
            K = Tb^(1/3)/BO.oSG;

            gCp = ((4 - BO.gSG)/6450)*(BO.T + 670)*(0.12*K - 0.41);
        end

        function oCp = get.oCp(BO)
            %HeatCapacity. (oil) Watson-Nelson (1933)
            %Units
            % T   : F
            % Tb  : F
            % oCp : BTU/lb.F

            % Tb: bubble point pressure @ 14.7 psia

            Tb = BO.Tbosc;
            K = Tb^(1/3)/BO.oSG;

            A1 = 0.055*K + 0.35;
            A2 = 0.6811 - 0.308*BO.oSG;
            A3 = 8.15e-4 - 3.06e-4*BO.oSG;

            oCp = A1*(A2 + A3*BO.T);
        end

        function wCp = get.wCp(BO)
            %HeatCapacity. (water) Miller (1976)
            %Units
```

```
            % T   : F
            % wCp : BTU/lb.F

            M = 18.0153;
            R = 1.986/M;

            A1 = 8.712;
            A2 = 1.25e-3;
            A3 = -0.18e-6;

            wCp = R*(A1 + A2*((BO.T + 460)/1.8) + A3*((BO.T +
460)/1.8)^2);
        end

        function Pb = get.Pb(BO)
            %BubblePointPressure. Fick correlation
            %Units
            % T   : F
            % GOR : scf/stb
            % Pb  : psia

            A = 9.1e-4*BO.T - 1.25e-2*BO.API;
            Pb = 18*BO.F01*(BO.GOR/BO.gSG)^0.83*10^A;
        end

        function Rs = get.Rs(BO)
            %InSolutionOGRatio. Standing correlation
            %Units
            % T   : F
            % P   : psia
            % Pb  : psia
            % Rs  : scf/stb

            if BO.P <= BO.Pb
                PRs = BO.P;
            else
                PRs = BO.Pb;
            end
            Rs = BO.gSG*(10^(1.25e-2*BO.API - 9.1e-4*BO.T)*PRs/(18
*BO.F01))^(1/0.83);
        end

        function Psc = get.Psc(BO)
            %PseudoCriticalPressure. Standing correlation
            %Units
            % Psc : psia

            Psc = 677 + 15*BO.gSG - 37.5*BO.gSG^2;
        end

        function Tsc = get.Tsc(BO)
            %PseudoCriticalTemperature. Standing correlation
            %Units
            % Tsc : R
```

```
            Tsc = 168 + 325*BO.gSG - 12.5*BO.gSG^2;
        end

        function Z = get.Z(BO)
            %CompressibilityFactor. Hall & Yarborough correlation
(1973/1974)
            %Units
            % T   : F
            % P   : psia
            % Tsc : R
            % Psc : psia

            Tri = BO.Tsc/(BO.T + 460);
            Prn = BO.P/BO.Psc;

            A = 6.125e-2*Tri*exp(-1.2*(1 - Tri)^2);
            B = 14.76*Tri - 9.76*Tri^2 + 4.58*Tri^3;
            C = 90.7*Tri - 242.2*Tri^2 + 42.4*Tri^3;
            D = 2.18 + 2.82*Tri;

            fy = @(y) -A*Prn + (y + y^2 + y^3 - y^4)/(1 - y)^3 - B*y^2
+ C*y^D;
            dfy = @(y) (1 + 4*y + 4*y^2 - 4*y^3 + y^4)/(1 - y)^4 -
2*B*y + C*D*y^(D-1);

            y0 = 1e-3;
            err = 1;

            while err > 1e-8
                y = y0 - fy(y0)/dfy(y0);
                err = abs(y - y0);
                y0 = y;
            end

            Z = A*Prn/y;
        end

        function emu = get.emu(BO)
            %ViscosityCalculation. (emulsion) Richardson
            %Units
            % emu : cP

            if BO.WC < 60
                emu = BO.omu*exp(BO.RichardsonO*BO.WC/1e2);
            else
                emu = BO.wmu*exp(BO.RichardsonW*BO.WC/1e2);
            end
        end

        function gmu = get.gmu(BO)
            %ViscosityCalculation. (gas) Lee correlation
            %Units
            % T    : F
```

```
            % grho : lb/ft3
            % grhoc: kg/m3
            % gmu  : cP

            grhoc = BO.grho*16.0186;

            Tr = BO.T + 460;
            m = 28.97*BO.gSG;
            dlg = grhoc/1000;

            k = (9.4 + 0.02*m)*Tr^1.5/(209 + 9*m + Tr);
            x = 3.5 + 986/Tr + 0.01*m;
            y = 2.4 - 0.2*x;
            gmu = 1e-4*k*exp(x*dlg^y);
        end

         function omu = get.omu(BO)
            %ViscosityCalcalculation. (oil) Beggs and Robinson
correlation
            %Units
            % T   : F
            % P   : psia
            % Pb  : psia
            % Rs  : scf/stb
            % GOR : scf/stb
            % omu : cP

            y = 3.0324 - 2.023e-2*BO.API;
            x = 10^y*BO.T^(-1.163);
            visd = 10^x - 1;

            if BO.Rs < BO.GOR
                vRs = BO.Rs;

                if ~isempty(BO.pcF05) && ~isempty(BO.pcF06)
                    BO.F05 = polyval(BO.pcF05, vRs);
                    BO.F06 = polyval(BO.pcF06, vRs);
                end

                A = 10.715*(vRs + 100)^(-0.515);
                B = 5.44*(vRs + 150)^(-0.338);
                omu = A*BO.F05*visd^(BO.F06*B);
            else

               if ~isempty(BO.pcF05) && ~isempty(BO.pcF06)
                    BO.F05 = polyval(BO.pcF05, BO.GOR);
                    BO.F06 = polyval(BO.pcF06, BO.GOR);
                end

                A = 10.715*(BO.GOR + 100)^(-0.515);
                B = 5.44*(BO.GOR + 150)^(-0.338);
                visb = A*BO.F05*visd^(BO.F06*B);
                z = BO.F07*(2.4*visb^1.6 + 3.8*visb^0.56)/1e5;
                omu = visb + z*(BO.P - BO.Pb);
```

```
            end
        end

        function wmu = get.wmu(BO)
            %ViscosityCalcalculation. (water) Brown and Beggs
correlation
            %Units
            % T   : F
            % wmu : cP

            wmu = exp(1.003 - 1.479e-2*BO.T + 1.982e-5*BO.T^2);
        end

        function grho = get.grho(BO)
            %DensityCalculation. (gas) Definition
            %Units
            % Bg   : rcf/scf
            % grho : lb/ft3

            grho = (1.223/16.0185)*BO.gSG/BO.Bg;
        end

        function orho = get.orho(BO)
            %DensityCalculation. (oil)
            %Units
            % Rs   : scf/stb
            % GOR  : scf/stb
            % orho : lb/ft3

            % Rsc  : m3/m3
            % GORc : m3/m3

            Rsc = BO.Rs/5.615;
            GORc = BO.GOR/5.615;

            if Rsc < GORc
                orho = (1e3*BO.oSG + 1.223*Rsc*BO.gSG)/BO.Bo;
            else
                orho = (1e3*BO.oSG + 1.223*GORc*BO.gSG)/BO.Bo;
            end
            orho = orho/16.0186;
        end

        function wrho = get.wrho(BO)
            %DensityCalculation. (water)
            %Units
            % Sw   : ppm
            % Bw   : rbl/stb
            % wrho : lb/ft3

            % wSG = 1 + 0.695*1e-6*Sw;
            % Sw : solids in formation water (ppm)
            % Collins (1987)
```

```
            wrho = BO.wSG*62.37/BO.Bw;
        end

        function qsc2ac = get.qsc2ac(BO)
            %StandardToActual. Flow rate
            %Units
            % Bg: rcf/stb
            % Bo: rbl/stb
            % Bw: rbl/stb
            % Rs: scf/stb

            Bgc = BO.Bg;
            Boc = BO.Bo;
            Bwc = BO.Bw;
            Rsc = BO.Rs/5.615;

            qsc2ac = [Bgc  -Bgc*Rsc  0; ...
                      0     Boc      0; ...
                      0     0        Bwc];
        end

        function rhosc2ac = get.rhosc2ac(BO)
            %StandardToActual. Density
            %Units
            % Bg: rcf/stb
            % Bo: rbl/stb
            % Bw: rbl/stb
            % Rs: scf/stb

            Bgc = BO.Bg;
            Boc = BO.Bo;
            Bwc = BO.Bw;
            Rsc = BO.Rs;

            rhosc2ac = [1/Bgc    0       0; ...
                        Rsc/Boc  1/Boc   0; ...
                        0        0       1/Bwc];
        end
    end

    methods

        function TuneViscosity(BO, mrmu, mrT)
            %TuneViscosity. Tuning factors for viscosity calculation
            % TuneViscosity provide with the fitting parameters F05
and F05
            % to adjust the viscosity calculation of the model with
            % experimental data.

            T0 = BO.T;
            P0 = BO.P;

            n = length(mrT);
            BO.F05 = 1;
```

```
            BO.F06 = 1;
            BO.F07 = 1;

            A = zeros(n, 1);
            B = zeros(n, 1);
            C = zeros(n, 1);
            domu = zeros(n, 1);
            BO.P = 14.7;

            for i = 1:n
                BO.T = mrT(i);

                y = 3.0324 - 2.023e-2*BO.API;
                x = 10^y*mrT(i)^(-1.163);
                domu(i) = 10^x - 1;

                if BO.Rs < BO.GOR
                    C(i) = BO.Rs;
                else
                    C(i) = BO.GOR;
                end

                A(i) = 10.715*(C(i) + 100)^(-0.515);
                B(i) = 5.44*(C(i) + 150)^(-0.338);
            end

            pcmu = polyfit(log(domu), log(mrmu), 1);
            vF05 = exp(pcmu(2))./A;
            vF06 = pcmu(1)./B;

            BO.pcF05 = polyfit(C, vF05, 1);
            BO.pcF06 = polyfit(C, vF06, 1);

            BO.T = T0;
            BO.P = P0;
        end

        function ClearTuneViscosity(BO)
            %ClearTuneViscosity. Clear viscosity tuning paramenters
            % ClearTuneViscosity allows to clear the functions
 developed
            % for adjusting the parameters of the viscosity
 calculation.

            BO.pcF05 = [];
            BO.pcF06 = [];
        end
    end
end
```

Published with MATLAB® R2015a