```
classdef VertCompletionObj < handle
%VertCompletionObj. Vertical Completion Object
% VertCompletionObj includes the information from the inflow
 performance
% relationships (IPR) in a vertical completion.
%
%   Properties
%    - IPRType         : IPR model
%    - Qsc             : hydrocarbon flow rate @SC             [Sm3/d]
%    - Pws             : reservoir pressure                    [kPa]
%    - Pwf             : bottom-hole pressure                  [kPa]
%    - Twf             : bottom-hole temperature               [C]
%    - JonesA          : Jones' model laminar flow constant    []
%    - JonesB          : Jones' model turbulent flow constant  []
%    - JonesType       : Jones' model phase type
%    - WellPIJ         : well productivity index
%    - WellPIType      : well PI phase type
%    - VogelQmax       : Vogel's model maximum flow rate       [Sm3/d]
%    - FetchovitchQmax : Fetchovitch's model maximum flow rate [Sm3/d]
%    - FetchovitchN    : Fetchovitch's model n coefficient     [-]
%    - BackPressureC   : backpressure model c coefficient      []
%    - BackPressureN   : backpressure model n coefficient      [-]
%    - CalculationType : pressure or flow rate calculation
%
%   Dependent properties
%    - status          : programming property (to be used in further
 versions)
%
%   Methods
%    - QCalculation
%        Calculation of Qsc using the selected IPR model.
%    - PwfCalculation
%        Calculation of Pwf using the selected IPR model.
%    - SolveCompletion
%        Solve the completion depending on the selected model,
 calculating
%        either Pwf or Qsc, according the complation data.
%
 ------------------------------------------------------------------------
% Development
%   By: Ruben Ensalzado
%       2015
%   Rev 00 151216 original release
%   Rev 01 160109 remove Q, Tsc, Psc, Zwf properties
%                 fix bug in error notification for BadArguments error
 type
%                 include AOFPCalculation method

    properties
        IPRType
        Qsc
        Pws
```

```
        Pwf
        Twf
        JonesA
        JonesB
        JonesType
        WellPIJ
        WellPIType
        VogelQmax
        FetchovitchQmax
        FetchovitchN
        BackPressureC
        BackPressureN
        CalculationType
    end

    properties (Dependent = true)
        status
        AOFP
    end

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

        function VC = VertCompletionObj
            VC.IPRType = 'Well PI';
            VC.CalculationType = 'pwf';
            VC.JonesType = 'oil';
            VC.WellPIType = 'oil';
        end
    end

    methods
        % -- error verification methods --

        function set.IPRType(VC, uIPRType)
            if strcmpi(uIPRType, 'well pi') ||
strcmpi(uIPRType, 'vogel') || ...
               strcmpi(uIPRType, 'fetkovitch') ||
strcmpi(uIPRType, 'jones') || ...
               strcmpi(uIPRType, 'backpressure equation')
                VC.IPRType = uIPRType;
            else
                errrm = ['The available methods are:';...
                         '- Well PI                ';...
                         '- Vogel                  ';...
                         '- Fetkovcitch            ';...
                         '- Jones                  '; ...
                         '- BackPressure Equation  '];

                error('VerticalCompletionObject:BadArgument', ...
                      '%s \n\t%s \n\t%s \n\t%s \n\t%s \n\t%s \n', ...
                      errrm(1, :), errrm(2, :), errrm(3, :), ...
                      errrm(4, :), errrm(5, :), errrm(6, :))
            end
```

```
        end

        function set.CalculationType(VC, uType)
            if strcmpi(uType, 'pwf') || strcmpi(uType, 'q')
                VC.CalculationType = uType;
            else
                errrm = ['The available calculations are:';...
                         '- pwf: bottom hole presssure   ';...
                         '- q  : fluid inflow            '];

                error('VerticalCompletionObject:BadArgument', ...
                      '%s \n\t%s \n\t%s \n', ...
                      errrm(1, :), errrm(2, :), errrm(3, :));
            end
        end

        function set.WellPIType(VC, uWellPIType)
            if strcmpi(uWellPIType, 'oil') ||
strcmpi(uWellPIType, 'gas')
                VC.WellPIType = uWellPIType;
            else
                errrm = ['The available types are:';...
                         '- oil                  ';...
                         '- gas                  '];

                error('VerticalCompletionObject:BadArgument', ...
                      '%s \n\t%s \n\t%s \n', ...
                      errrm(1, :), errrm(2, :), errrm(3, :));
            end
        end

        function set.JonesType(VC, uJonesType)
            if strcmpi(uJonesType, 'oil') ||
strcmpi(uJonesType, 'gas')
                VC.JonesType = uJonesType;
            else
                errrm = ['The available types are:';...
                         '- oil                  ';...
                         '- gas                  '];

                error('VerticalCompletionObject:BadArgument', ...
                      '%s \n\t%s \n\t%s \n', ...
                      errrm(1, :), errrm(2, :), errrm(3, :));
            end
        end
    end

    methods
        % -- dependent properties --

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

            req = zeros(5, 1);
```

```
        req(1) = isempty(VC.Qsc);
        req(2) = isempty(VC.Pws);

        switch lower(VC.IPRType)
            case 'well pi'
                req(3) = isempty(VC.WellPIType);
                req(4) = isempty(VC.WellPIJ);
            case 'vogel'
                req(3) = isempty(VC.VogelQmax);
            case 'fetkovitch'
                req(3) = isempty(VC.FetchovitchQmax);
                req(4) = isempty(VC.FetchovitchN);
            case 'jones'
                req(3) = isempty(VC.JonesType);
                req(4) = isempty(VC.JonesA);
                req(5) = isempty(VC.JonesB);
            case 'backpressure equation'
                req(3) = isempty(VC.BackPressureC);
                req(4) = isempty(VC.BackPressureN);
        end

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

    function AOFP = get.AOFP(VC)
        %AOFP. Absolute open flow performance
        % Well performance when Pwf = 0.

        CP0 = VC.CalculationType;

        switch CP0
            case 'q'
                CP1 = VC.Pws;
                CP2 = VC.Pwf;
            case 'pwf'
                CP1 = VC.Qsc;
                CP2 = VC.Pwf;
        end

        VC.CalculationType = 'q';
        VC.Pwf = 0;
        VC.SolveCompletion
        AOFP = VC.Qsc;

        switch CP0
            case 'q'
                VC.CalculationType = 'q';
                VC.Pws = CP1;
                VC.Pwf = CP2;
```

```
                case 'pwf'
                    VC.CalculationType = 'pwf';
                    VC.Qsc = CP1;
                    VC.Pwf = CP2;
            end
        end
    end

    methods
        % -- calculation methods --

        function CalculateQ(VC)
            %CalculateFlowRate. Q calculation depending on the IPR
model

            switch lower(VC.IPRType)
                case 'well pi'
                    VC.QWellPI
                case 'vogel'
                    VC.QVogel
                case 'fetkovitch'
                    VC.QFetkovitch
                case 'jones'
                    VC.QJones
                case 'backpressure equation'
                    VC.QBackPressure
            end

            if VC.Qsc < 0
                VC.Qsc = 0;
                lastwarn('Pwf may be greater than Pws; Q was set to
zero!')
            end
        end

        function CalculatePwf(VC)
            %CalculateBottomHolePressure. Pwf calculation depending on
the
            %IPR model

            switch lower(VC.IPRType)
                case 'well pi'
                    VC.PwfWellPI
                case 'vogel'
                    VC.PwfVogel
                case 'fetkovitch'
                    VC.PwfFetkovitch
                case 'jones'
                    VC.PwfJones
                case 'backpressure equation'
                    VC.PwfBackPressure
            end

            if any(~isreal(VC.Pwf))
```

```
                VC.Pwf = 0;
                lastwarn('Pwf may be greater than Pws; Q was set to
zero!')
            end
        end

        function SolveCompletion(VC)
            %SolveCompletion. Calculate Pwf or Qsc depeding on
calculation
            %method

            switch lower(VC.CalculationType)
                case 'pwf'
                    VC.CalculatePwf
                case 'q'
                    VC.CalculateQ
            end
        end
    end

    methods (Hidden = true)
        % -- IPR models : flow rate --

        function QWellPI(VC)
            %WellProductivityIndexFlowRate. Well PI IPR model
            %Units
            % J   : Sm3/d.kPa or Sm3/d.kPa2
            % Pws : kPa
            % Pwf : kPa
            % Qsc : Sm3/d

            J = VC.WellPIJ;

            switch lower(VC.WellPIType)
                case 'oil'
                    VC.Qsc = J*(VC.Pws - VC.Pwf);
                case 'gas'
                    VC.Qsc = J*(VC.Pws^2 - VC.Pwf^2);
            end
        end

        function QVogel(VC)
            %Vogel. (flow rate) Vogel IPR model
            %Units
            % qmax : Sm3/d
            % Pws  : kPa
            % Pwf  : kPa
            % Qsc  : Sm3/d

            Pwf_Pws = VC.Pwf/VC.Pws;
            qmax = VC.VogelQmax;

            VC.Qsc = qmax*(1 - 0.2*(Pwf_Pws) - 0.8*(Pwf_Pws)^2);
        end
```

```
        function QFetkovitch(VC)
            %Fetkovitch. (flow rate) Fetkovitch IPR model
            %Units
            % qmax : Sm3/d
            % Pws  : kPa
            % Pwf  : kPa
            % Qsc  : Sm3/d

            Pwf_Pws = VC.Pwf/VC.Pws;
            qmax = VC.FetchovitchQmax;
            n = VC.FetchovitchN;

            VC.Qsc = qmax*(1 - (Pwf_Pws)^2)^n;
        end

        function QJones(VC)
            %Jones. (flow rate) Jones IPR model
            %Units
            % JonesA : kPa.d/Sm3 or kPa2.d/Sm3
            % JonesB : kPa.d2/(Sm3)2 or kPa2.d2/(Sm3)2
            % Pws    : kPa
            % Pwf    : kPa
            % Qsc    : Sm3/d

            C1 = VC.JonesB;
            C2 = VC.JonesA;

            switch lower(VC.JonesType)
                case 'oil'
                    C3 = VC.Pws - VC.Pwf;
                case 'gas'
                    C3 = VC.Pws^2 - VC.Pwf^2;
            end

            VC.Qsc = max(roots([C1 C2 C3]));
        end

        function QBackPressure(VC)
            %BackPressure. (flow rate) Backpressure IPR model
            %Units
            % C   : Sm3/(kPa)2n
            % Pws : kPa
            % Pwf : kPa
            % Qsc : Sm3/d

            C = VC.BackPressureC;
            n = VC.BackPressureN;

            VC.Qsc = C*(VC.Pws^2 - VC.Pwf^2)^n;
        end
    end

    methods (Hidden = true)
```

```
        % -- IPR models : bottom hole pressure --

        function PwfWellPI(VC)
            %WellProductivityIndex. (bottomhole pressure) Well PI IPR
    model
            %Units
            % J   : Sm3/d.kPa or Sm3/d.kPa2
            % q   : Sm3/d
            % Pws : kPa
            % Pwf : kPa

            J = VC.WellPIJ;
            pws = VC.Pws;
            q = VC.Qsc;

            switch lower(VC.WellPIType)
                case 'oil'
                    VC.Pwf = pws - q/J;
                case 'gas'
                    VC.Pwf = sqrt(pws^2 - q/J);
            end
        end

        function PwfVogel(VC)
            %Vogel. (bottomhole pressure) Vogel IPR model
            %Units
            % qmax : Sm3/d
            % q    : Sm3/d
            % Pws  : kPa
            % Pwf  : kPa

            pws = VC.Pws;
            q = VC.Qsc;
            qmax = VC.VogelQmax;

            C1 = 0.8;
            C2 = 0.2;
            C3 = q/qmax - 1;

            VC.Pwf = max(roots([C1 C2 C3]))*pws;
        end

        function PwfFetkovitch(VC)
            %Fetkovitch. (bottomhole pressure) Fetkovitch IPR model
            %Units
            % qmax : Sm3/d
            % q    : Sm3/d
            % Pws  : kPa
            % Pwf  : kPa

            pws = VC.Pws;
            q = VC.Qsc;
            qmax = VC.FetchovitchQmax;
            n = VC.FetchovitchN;
```

```
            VC.Pwf = pws*sqrt(1 - (q/qmax)^(1/n));
        end

        function PwfJones(VC)
            %Jones. (bottomhole pressure) Jones IPR model
            %Units
            % JonesA : kPa.d/Sm3 or kPa2.d/Sm3
            % JonesB : kPa.d2/(Sm3)2 or kPa2.d2/(Sm3)2
            % q      : Sm3/d
            % Pws    : kPa
            % Pwf    : kPa

            A = VC.JonesB;
            B = VC.JonesA;
            pws = VC.Pws;
            q = VC.Qsc;

            switch lower(VC.JonesType)
                case 'oil'
                    VC.Pwf = pws - A*q - B*q^2;
                case 'gas'
                    VC.Pwf = sqrt(pws^2 - A*q - B*q^2);
            end
        end

        function PwfBackPressure(VC)
            %BackPressure. (flow rate) Backpressure IPR model
            %Units
            % C   : Sm3/(kPa)2n
            % q   : Sm3/d
            % Pws : kPa
            % Pwf : kPa

            pws = VC.Pws;
            q = VC.Qsc;
            C = VC.BackPressureC;
            n = VC.BackPressureN;

            VC.Pwf = sqrt(pws^2 - q^(1/n)/C);
        end
    end

end
```

Published with MATLAB® R2015a