```
classdef ESPObj < handle
%ESPObj. Electro Submersible Pump Object
% ESPObj represents a single stage centrifugal electrosubmersible pump
% (ESP) to lift production fluid.
%
%   Properties
%    - Q          : capacity                          [m3/h]
%    - CurveHQ    : head-capacity data                [m, m3/h]
%    - CurvePQ    : power-capacity data               [kW, m3/h]
%    - CurveNQ    : efficiency-capacity data          [-, m3/h]
%    - CurveHQvc  : head-capacity data @VA            [m, m3/h]
%    - CurvePQvc  : power-capacity data @VA           [kW, m3/h]
%    - CurveNQvc  : efficiency-capacity data @VA      [-, m3/h]
%    - CurveHQnc  : head-capacity data @FA            [m, m3/h]
%    - CurvePQnc  : power-capacity data @FA           [kW, m3/h]
%    - FCurveHQvc : head-capacity curve @VA           [m, m3/h]
%    - FCurveHQnc : head-capacity curve @FA           [m, m3/h]
%    - BEP        : best efficiency point (HQ)        [m, m3/h]
%    - N          : design frequency                  [rpm]
%    - Nc         : adjusted frequency                [rpm]
%    - vnu        : fluid viscosity (until 4000 cSt)  [cSt]
%
%   Dependent properties
%    - status     : programming property (to be used in further
 versions)
%    - H          : differential head                 [m]
%
%   Methods
%    - ViscosityAdjustment
%        Adjustment of pump performance data with water considering
 the
%        fluid real viscosity.
%    - FrequencyAdjustment
%        Adjustment of pump performance data with water considering
 the
%        value Nc.
%    - PlotHQvAdjustment
%        Plotting pump performance including the original data (water)
%        and the viscous fluid.
%    - PlotHQnAdjustment
%        Plotting pump performance including the viscosity corrected
 curve
%        at the design frequency and the curve at the adjusted
 frequency.
%
 ---------------------------------------------------------------------
% Development
%   By: Ruben Ensalzado
%       2015
%   Rev 00 151216 original release
%   Rev 01 160108 bug fixing: error notification BadArguments error
 type
```

```
%   Rev 02 160306 bug fixing: updating performance curve when Nc
 changes
%                 definition: include code for B <= 1

    properties
        Q
        CurveHQ
        CurvePQ
        CurveNQ
        CurveHQvc
        CurvePQvc
        CurveNQvc
        CurveHQnc
        CurvePQnc
        FCurveHQvc
        FCurveHQnc
        BEP
        N
        vnu
        SG
    end

    properties (SetObservable)
        Nc
    end

    properties (Dependent)
        H
        status
    end

    properties (Constant)
        g = 9.81;
    end

    properties (Hidden, SetAccess = immutable)
        Pf
    end

    methods (Hidden)
        % -- initialization method --

        function ESP = ESPObj
            ESP.CurveHQ = zeros(10, 2);
            ESP.BEP = [0 0];
            ESP.SG = 0.895;
            ESP.Pf = @(Qvis, Hvis, N, Nvis) Qvis.*Hvis*N./(367*Nvis);
            addlistener(ESP, 'Nc', 'PostSet',
 @ESPObj.ESPPropertyChange);
        end
    end

    methods
        % -- error verification methods --
```

```
        function set.CurveHQ(ESP, CurveHQ)
            if size(CurveHQ, 2) == 2
                ESP.CurveHQ = CurveHQ;
            else
                error('ESPObj:BadArgument', ...
                      'Dimensions of HQ matrix are nx2')
            end
        end

        function set.CurvePQ(ESP, CurvePQ)
            if size(CurvePQ, 2) == 2
                ESP.CurvePQ = CurvePQ;
            else
                error('ESPObj:BadArgument', ...
                      'Dimensions of PQ matrix are nx2')
            end
        end

        function set.BEP(ESP, BEP)
            if size(BEP, 1) == 1 && size(BEP, 2) == 2
                ESP.BEP = BEP;
            else
                error('ESPObj:BadArgument', ...
                      'BEP should be a row vector 1x2')
            end
        end
    end

    methods
        % -- dependent properties --

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

            req = zeros(6, 1);

            req(1) = isempty(ESP.Q);
            req(2) = isempty(ESP.CurveHQ);
            req(3) = isempty(ESP.BEP);
            req(4) = isempty(ESP.N);
            req(5) = isempty(ESP.Nc);
            req(6) = isempty(ESP.vnu);

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

        function H = get.H(ESP)
            %DifferentialHead.
            %Units
```

```
            % Nc : rpm
            % N  : rpm
            % Q  : m3/h
            % H  : m

            if isempty(ESP.Nc)
                ESP.Nc = ESP.N;
            end

            if isempty(ESP.FCurveHQnc)
                ESP.PerformanceCurve('frequency')
            end

            H = polyval(ESP.FCurveHQnc, ESP.Q);
        end
    end

    methods
        % -- calculation methods --

        function ViscosityAdjustment(ESP)
           %ViscosityAdjustment. Pump performance curve adjustment
           % Viscosity adjustment according to ANSI/HI 9.6.7-2010

           if isempty(ESP.N) || isempty(ESP.vnu) || isempty(ESP.BEP)
               error('ESPObj:NotEnoughArguments', ...
                     'N, BEP and viscosity have to be defined first.')
           end

           Np = ESP.N;
           nuvis = ESP.vnu;
           BEP_H = ESP.BEP(1);
           BEP_Q = ESP.BEP(2);

           B = 16.5*sqrt(nuvis)*(BEP_H)^(0.0625)/
(BEP_Q^0.375*Np^0.25);

           if B > 1
               Cq = 2.71^(-0.165*log10(B)^3.15);
               Cb = Cq;
               Chf = @(Qw) 1 - ((1 - Cb)*(Qw/BEP_Q).^0.75);
               Ch = Chf(ESP.CurveHQ(:,2));
               Cn = B^(-0.0547*B^0.69);
           else
               Cq = 1;
               Ch = 1;
               Cn = 1;
           end


           ESP.CurveHQvc = [Ch.*ESP.CurveHQ(:, 1) Cq*ESP.CurveHQ(:,
 2)];

           if ~isempty(ESP.CurveNQ) && ~isempty(ESP.CurvePQ)
```

```
                ESP.CurveNQvc = [Ch.*ESP.CurveNQ(:, 1)
Cn*ESP.CurveNQ(:, 2)];
                ESP.CurvePQvc(:, 1) = Ch.*ESP.CurvePQ(:, 1);
                ESP.CurvePQvc(:, 2) = ESP.Pf(ESP.CurvePQvc(:, 1), ...
                                             ESP.CurveHQvc(:,1), Np, ...
                                             ESP.CurveNQvc(:,2));
            end
        end

        function PerformanceCurve(ESP, type)
            %Performance Curve. Adjust pump performance curve to a
polynom
            % Polynomic adjustment, considering the less possible
grade and
            % the R2 as an fitness parameter. If R2 differs in less
than 5%
            % then, thhe lower grade polynom is selected.

            if isempty(ESP.CurveHQnc) || isempty(ESP.CurveHQvc)
                return
            end

            if  ~(strcmpi(type, 'viscosity') ||
strcmpi(type, 'frequency'))

                errrm = ['The available calculation type are:';...
                         '- viscosity                        ';...
                         '- frequency                        '];

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

            fr2 = @(y, f) max(0, 1 - sum((y - f).^2)/sum((y -
mean(y)).^2));

            switch lower(type)
                case 'viscosity'
                    x = ESP.CurveHQvc(:, 2);
                    y = ESP.CurveHQvc(:, 1);
                case 'frequency'
                    x = ESP.CurveHQnc(:, 2);
                    y = ESP.CurveHQnc(:, 1);
            end

            opoly = polyfit(x, y, 2);
            or2 = fr2(y, polyval(opoly, x));

            for i = 3:9
                 npoly = polyfit(x, y, i);
                 nr2 = fr2(y, polyval(npoly, x));

                 if abs(or2 - nr2)/nr2 <= 0.05
```

```
                    apoly = opoly;
                    break
                end

                opoly = npoly;
                or2 = nr2;
            end

            switch lower(type)
                case 'viscosity'
                    ESP.FCurveHQvc = apoly;
                case 'frequency'
                    ESP.FCurveHQnc = apoly;
            end

        end

        function FrequencyAdjustment(ESP)
            %FrequencyAdjustment. Pump performance curve adjustment
            % Frequency correction using affinity laws

            if isempty(ESP.Nc)|| isempty(ESP.CurveHQvc)
               error('ESPObj:NotEnoughArguments', ...
                     'Nc, and the adjusted performance pump curve have
to be defined first.')
            end

            N1 = ESP.N;
            N2 = ESP.Nc;

            ESP.CurveHQnc(:, 1) = ESP.CurveHQvc(:, 1)*(N2/N1)^2;
            ESP.CurveHQnc(:, 2) = ESP.CurveHQvc(:, 2)*(N2/N1);

            if ~isempty(ESP.CurvePQvc)
                ESP.CurvePQnc(:, 1) = ESP.CurvePQvc(:, 1)*(N2/N1)^3;
                ESP.CurvePQnc(:, 2) = ESP.CurvePQvc(:, 2)*(N2/N1);
            end
        end

    end

    methods
       % -- information and plotting methods --

       function PlotHQvAdjustment(ESP)
           %PlotHQvAdjustment. Plot performace curve - viscosity
           % Plotting water performace curve vs viscous fluid

           if isempty(ESP.CurveHQvc)
               error('ESPObj:NotEnoughArguments', ...
                     'There is not HQvc curve to plot')
           end

           figure
```

```
            axes
            hold on
            grid on
            plot(ESP.CurveHQ(:,2), ESP.CurveHQ(:,1), ...
                 '-o', 'MarkerFaceColor', 'b')
            plot(ESP.CurveHQvc(:,2), ESP.CurveHQvc(:,1), ...
                 '-o', 'MarkerFaceColor', 'r')
            set(gca, 'XLim', [0 round(max(ESP.CurveHQvc(:,
2))*1.1)], ...
                     'YLim', [0 round(max(ESP.CurveHQvc(:,
1))*1.1)], ...
                     'FontName', 'Segoe UI')
            xlabel(gca, 'Capacity (m^3/h)')
            ylabel(gca, 'Head (m)')
            title(gca, 'HQ pump curve - Viscosity Adjustment')
            legend('Pump performance curve', ...
                   'Pump adjusted performance curve', ...
                   'Location', 'southwest');
            hold off
        end

        function PlotHQnAdjustment(ESP)
            %PlotHQnAdjustment. Plot performace curve - frequency
            % Plotting base frequency performance vs new frequency

            if isempty(ESP.CurveHQnc)
                error('ESPObj:NotEnoughArguments', ...
                      'There is not HQnc curve to plot')
            end

            figure
            axes
            hold on
            grid on
            plot(ESP.CurveHQvc(:,2), ESP.CurveHQvc(:,1), ...
                 '-o', 'MarkerFaceColor', 'b')
            plot(ESP.CurveHQnc(:,2), ESP.CurveHQnc(:,1), ...
                 '-o', 'MarkerFaceColor', 'r')
            set(gca, 'XLim', [0 round(max(ESP.CurveHQvc(:,
2))*1.1)], ...
                     'YLim', [0 round(max(ESP.CurveHQvc(:,
1))*1.1)], ...
                     'FontName', 'Segoe UI')
            xlabel(gca, 'Capacity (m^3/h)')
            ylabel(gca, 'Head (m)')
            title(gca, 'HQ pump curve - Frequency Adjustment')
            legend(['Pump performance curve' ' @ ' num2str(ESP.N) '
rpm'], ...
                   ['Pump adjusted performance curve' ' @ '
num2str(ESP.Nc) ' rpm'], ...
                   'Location', 'southwest');
            hold off
        end
```

```
        function PlotpQvAdjustment(ESP)
            %PlotpQnAdjustment. Plot performace curve - viscosity
            % Plotting base frequency performance vs new frequency

            if isempty(ESP.CurveHQnc)
                error('ESPObj:NotEnoughArguments', ...
                      'There is not HQnc curve to plot')
            end

            figure
            axes
            hold on
            grid on
            plot(ESP.CurveHQvc(:,2),
 ESP.CurveHQvc(:,1)*ESP.SG*ESP.g, ...
                 '-o', 'MarkerFaceColor', 'b')
            plot(ESP.CurveHQnc(:,2),
 ESP.CurveHQnc(:,1)*ESP.SG*ESP.g, ...
                 '-o', 'MarkerFaceColor', 'r')
            set(gca, 'XLim', [0 round(max(ESP.CurveHQvc(:,
 2))*1.1)], ...
                     'YLim', [0 round(max(ESP.CurveHQvc(:,
 1)*ESP.SG*ESP.g)*1.1)], ...
                     'FontName', 'Segoe UI')
            xlabel(gca, 'Capacity (m^3/h)')
            ylabel(gca, 'Pressure (kPa)')
            title(gca, 'pQ pump curve - Viscosity Adjustment')
            legend('Pump performance curve @ 1 cSt', ...
                   ['Pump adjusted performance curve @ '
 num2str(ESP.vnu) ' cSt'], ...
                    'Location', 'southwest');
            hold off
        end

    end

    methods (Static)
      function ESPPropertyChange(ESPsource, ESPevent)
          switch ESPsource.Name
              case 'Nc'

 ESPevent.AffectedObject.PerformanceCurve('frequency')
          end
      end
    end

end
```

Published with MATLAB® R2015a