ESP Lifted Oil Field: Basic Model for Control, and Comparison of Simulation Tools

Bernt Lie *

Posted Date: 19 December 2023

doi: 10.20944/preprints202312.1383.v1

Keywords: Oil Production; ESP lift; Dimensionless model; Dynamic model; Simulation tool; Modelica; ModelingToolkit

Article

ESP Lifted Oil Field: Basic Model for Control, and Comparison of Simulation Tools

Bernt Lie [1,*]

[1] University of South-Eastern Norway, Norway; Bernt.Lie@usn.no
* Correspondence: Bernt.Lie@usn.no

Abstract: Optimal operation of petroleum production is important in a transition from energy systems based on fossil fuel to sustainable systems. One sub-process in petroleum production deals with transport from the (subsea) well-bore to a topside separator. A simple model in Sharma and Glemmestad [1] of Electric Submersible Pump [ESP] lifted production was previously streamlined into a dynamic model suitable for illustration of the dynamics of oil transport, as well for control studies, with some comparison of two popular modeling languages: Modelica, and ModelingToolkit for Julia, Lie [2]. Here, the discussion on dimensionless equipment models goes into more detail, and the comparison between Modelica and ModelingToolkit is significantly expanded upon with code comparison, numerical performance, and more experiments. Some added possibilities with ModelingToolkit and Julia wrt. sensitivity analysis and control design is included.

Keywords: oil production; ESP lift; dimensionless model; dynamic model; simulation tool; modelica; ModelingToolkit

1. Introduction

1.1. Background

Petroleum products have been key energy carriers for more than a century. Current focus on climate[1] implies a change towards sustainable energy carriers. To succeed in this change, a transition period from the use of fossil fuel is necessary. In the transition, improved operation of petroleum production through model based optimal operation will be necessary. Petroleum production entails slow (reservoir; months) and fast (reservoir-to-separator; seconds) subsystems; this is a focus of on-going research project "DigiWell"[2]. Vertical transport of petroleum from oil well to surface requires sufficient pressure to counteract gravitational and friction forces. If the oil-well heel pressure is insufficient for such transport, either (i) gas is injected in the vertical pipe to "blow" the petroleum fluids to the surface ["gas lifted"], or (ii) an Electric Submersible Pump [ESP] is installed in the vertical pipe to increase the pressure ["ESP lifted"] sufficiently. Here, the focus is on the dynamics of transport from the reservoir formation to a surface manifold via an ESP, and further horizontal transport from the manifold to a separator.

Industrial simulation tools typically put main emphasis on the dynamics of the *reservoir* (time constant: months) and use steady state models for the reservoir-to-surface transport. This emphasis is inadequate for daily operation and control. Here, a simplified, yet complete, dynamic model for oil transport from reservoir to separator is discussed. The model provides an understanding of the dynamic behavior of such systems, and is suitable for industrial control design, as well as for control and petroleum production studies. Emphasis is put on a simple, yet stringent model development, while avoiding variable *unit* complexities.

[1] https://sdgs.un.org/goals
[2] DigiWell: see Funding.

 doi:10.20944/preprints202312.1383.v1

1.2. Previous work

Sharma and Glemmestad [1] (see also Sharma [3]) provide a dynamic model of oil transport from reservoir to separator suitable for control design. Binder *et al.* [4] discuss an older model; other models typically are CFD models, which are too complex for control design.

Sharma's model considers a case with 4 vertical pipes from oil reservoirs to a single manifold, with 2 horizontal pipes from the manifold to a single separator. Each vertical pipe has an ESP and a choke valve at a common manifold entrance; the pump speeds can be manipulated individually. The horizontal pipes have booster pumps to counteract friction effects. The original ESP model includes induction motors, but the dynamics of the pump actuator is fast, and is neglected in later work. Sharma and Glemmestad [1] provide a novel ESP model, a simple model for a booster pump, and use a valve model based on on the ANSI/ISA S75.01 standard[3]. The model in Sharma and Glemmestad [1] was re-structured and simplified in Lie [2], emphasizing dimensionless equipment models, and thereby eliminating some level of complexity in common industry models.

The model with ESP in Sharma and Glemmestad [1] is mainly relevant for the production of heavy oil. Several papers use this model in advanced industrial control studies, Krishnamoorthy *et al.* [5], Delou *et al.* [6], Santana *et al.* [7].

Mixtures of liquid oil and water form an emulsion when stirred (e.g., in a multi-stage ESP); for such emulsions, the viscosity — and hence the friction — varies dramatically with water content, Justiniano and Romero [8]. Sharma and Glemmestad [1] assume an unrealistic linear viscosity dependence on water fraction.

1.3. Structure of paper

Section 2 gives an overview of the transport system from oil reservoir via manifold to a separator, and key equipment models. Section 3 develops a simple mechanistic model of the system. Section 4 contrasts two modeling languages for simulation: Modelica and Julia's ModelingToolkit. Section 5 illustrates model behavior and the use of modeling/simulation tools for analysis and control. Finally, Section 6 provides some conclusions.

2. System description

Production of a mixture of water and crude oil in liquid phase is considered, where evaporation of liquids is assumed negligible.

2.1. System topology

Oil production *systems* merge several boreholes from the same or different reservoirs through vertical pipes into a manifold. Normally, more than one horizontal transport pipe is needed from the manifold to a separator for sufficient transport capacity. Water is commonly injected into the manifold to reduce friction loss in the horizontal pipes; the added water is typically recycled from the separator, and is at close to production temperature. Figure 1 shows a system with n_w wells/vertical pipes via a common manifold to n_t transportation/horizontal pipes leading to the separator.

---

[3] http://integrated.cc/cse/ISA_750101_SPBd.pdf

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

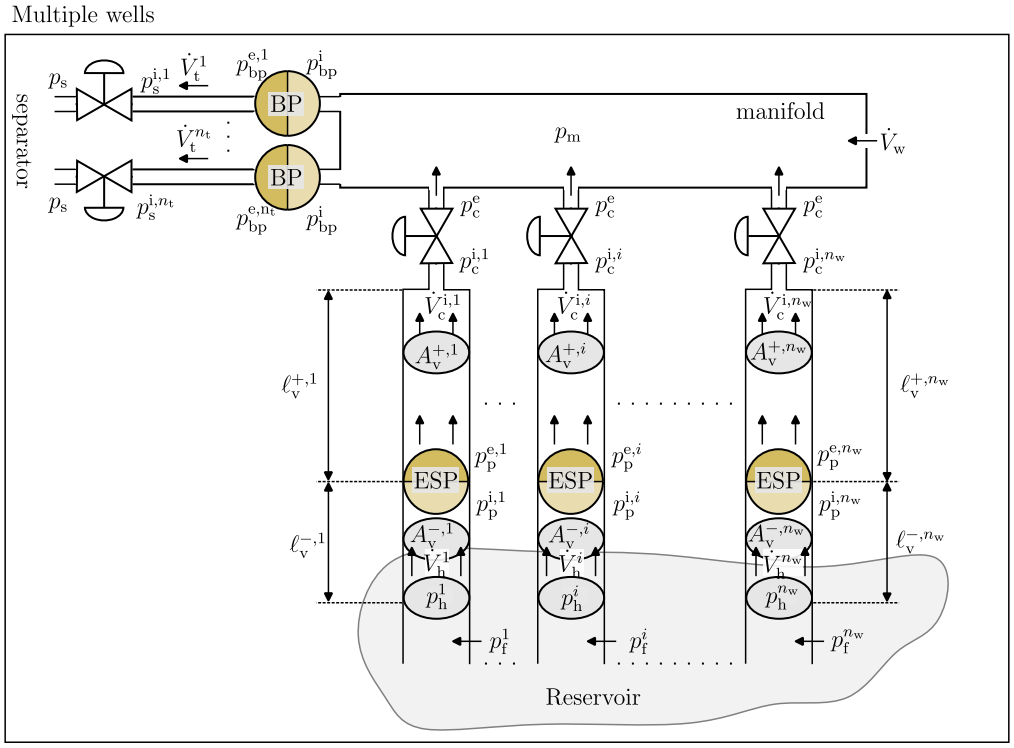

Figure 1. Multiple well system with n_w wells — possibly coming from different reservoirs, and n_t transport pipes to the separator; from Lie [2] and based on Sharma and Glemmestad [1].

In Figure 1, p_f^j is formation pressure for well j, p_h^j is bore-hole heel pressure, $\dot{V}_h^j$ is volumetric heel flow rate, $A_v^{-,j}$ is vertical cross sectional area below the ESP pump, $\ell_v^{-,j}$ is the length of the vertical pipe below the ESP pump, $p_p^{i,j}$ is inlet pressure to the (ESP) pump, $p_p^{e,j}$ is effluent pressure after the (ESP) pump, $A_v^{+,j}$ is vertical cross sectional area above the ESP pump, $\ell_v^{+,j}$ is the length of the vertical pipe above the ESP pump, $\dot{V}_c^{i,j}$ is volumetric flow rate at the inlet to the choke valve, $p_c^{i,j}$ is pressure at the inlet to the choke valve, $p_c^{e,j}$ is pressure at the effluent from the choke valve. Next, p_m is manifold pressure, while $\dot{V}_w$ is water added to the manifold to reduce viscosity. At the outlet from the manifold, $p_{bp}^{i,j}$ is the influent pressure to the booster pump (BP), $p_{bp}^{e,j}$ is effluent pressure after the booster pump, $\dot{V}_t^j$ is volumetric flow rate in a transport pipe from manifold to separator, $p_s^{i,j}$ is pressure at the inlet to the valve into the separator, and $p_s^{e,j} = p_s$ is the separator pressure.

For simplicity, it is assumed that $A_v^{-,j} = A_v^{+,j} = A_v$. All vertical pipes are assumed connected to the same manifold pressure p_m; hence *effluent choke pressure* satisfies $p_c^{e,j} = p_c^e = p_m$ for all j. The influent pressure to the booster pumps, $p_{bp}^{i,j}$ are all assumed to be equal to the outlet pressure from the manifold, and have the same value, $p_{bp}^{i,j} = p_{bp}^i = p_m$. Likewise, all transport pipes end up in the same separator: $p_s^{e,j} = p_s$ for all j.

2.2. Fluid properties

The petroleum fluid properties are important. Density ρ varies with pressure p and temperature T, $\rho(p, T)$. Neglecting temperature dependence, and assuming constant *isothermal compressibility* β_T,[4] $\rho(p)$ is given as

$$\rho = \rho_0 \exp(\beta_T (p - p_0)) \tag{1}$$

[4] Isothermal compressibility is the inverse of bulk modulus.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

where (ρ_0, p_0) is some reference state.

Defining water cut χ_w as $\chi_w \triangleq \dot{V}_w/\dot{V}$: volumetric flow rate of water divided by total flow rate of the fluid, total density ρ can be expressed as

$$\rho = \chi_w \rho_w + (1 - \chi_w)\,\rho_o; \tag{2}$$

here, ρ_w and ρ_o are constant densities of pure water and crude oil, respectively.

In reality, water and crude oil have different isothermal compressibilities. Here, we simplify and assume an overall value for β_T. Using data in Appendix A1, density ρ varies ca. 10 kg/m^3 with pressure variation in the range 25–225 bar, Figure 2.

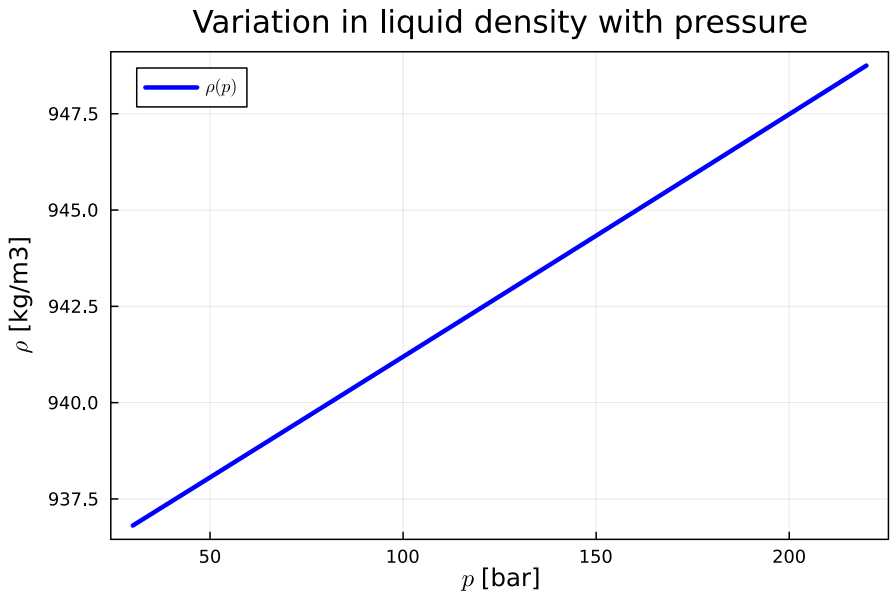

Figure 2. Typical variation in density for *production fluid* in the pressure range of interest.

We thus assume constant density in pipes, but a pressure-dependent density will be assumed in the manifold.

Sharma and Glemmestad [1] propose a simple linear mixing rule for *kinematic* viscosity ν:

$$\nu = \chi_w \nu_w + (1 - \chi_w)\,\nu_o. \tag{3}$$

With ν known, *dynamic* viscosity μ can be computed (if needed) as

$$\mu = \nu\rho.$$

The linear interpolation model of Eq. 3 is used here to ease comparison with results in Sharma and Glemmestad [1], even though it is not physically realistic [8].

2.3. Well-bore production

Total production from the reservoir (formation pressure p_f) relates volumetric petroleum fluid rate $\dot{V}_h$ at the well-bore heel as $\dot{V}_h \propto p_f - p_h$, where p_h is heel pressure and the proportionality constant C_{pi} is the *productivity index*,

$$\dot{V}_h = C_{pi} \cdot (p_f - p_h)\,;$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

C_{pi} is unit-dependent. Here, we instead propose a dimensionless form,

$$\dot{V}_{\text{h}} = \dot{V}^{\text{c}}_{\text{pi}} \frac{p_{\text{f}} - p_{\text{h}}}{p^{\varsigma}_{\text{pi}}} \tag{4}$$

where $\dot{V}^{\text{c}}_{\text{pi}}$ is the productivity index *capacity* in the same unit as $\dot{V}_{\text{h}}$, and $p^{\varsigma}_{\text{pi}}$ is scaling pressure with the same unit as $p_{\text{f}}, p_{\text{h}}$.

2.4. Pump models

Electric Submersible Pump

Pump models are often given as

$$\Delta p_{\text{p}} = \rho g h_{\text{p}}; \tag{5}$$

here, $h_{\text{p}} = h_{\text{p}}\left(\dot{V}, f_{\text{p}}\right)$ is pump *head* with volumetric flow rate $\dot{V}$ and control input f_{p} — rotational pump frequency Hz.

Sharma and Glemmestad [1] provide values for minimal, maximal, and best-efficiency-point flow rates,

$$\frac{\dot{V}_{\text{min}}}{\dot{V}_{\text{min},0}} = \frac{f_{\text{p}}}{f_{\text{p},0}} \tag{6}$$

$$\frac{\dot{V}_{\text{max}}}{\dot{V}_{\text{max},0}} = \frac{f_{\text{p}}}{f_{\text{p},0}} \tag{7}$$

$$\frac{\dot{V}_{\eta}}{\dot{V}_{\eta,0}} = \frac{f_{\text{p}}}{f_{\text{p},0}}. \tag{8}$$

In Sharma and Glemmestad [1], a comprehensive model for the pump head of a *multi-stage ESP* is developed. To ease change of units, their model is here rewritten in dimensionless form

$$\frac{h_{\text{p}}\left(\dot{V}, f_{\text{p}}\right)}{h_{\text{p},0}} = \left(\frac{f_{\text{p}}}{f_{\text{p},0}}\right)^2 + \sum_{j=1}^{3} a_j \left(\frac{f_{\text{p}}}{f_{\text{p},0}}\right)^{2-j} \left(\frac{\dot{V}}{\dot{V}^{\varsigma}}\right)^j. \tag{9}$$

In Eq. 9, $h_{\text{p},0}$ is a nominal scaling head, f_{p} is the pump rotational frequency in the same unit as that of the nominal rotational frequency $f_{\text{p},0}$, $\dot{V}$ is the actual volumetric flow rate out of the pump, $\dot{V}^{\varsigma}$ a scaling flow rate, and $a_1, \ldots, a_3$ are dimensionless model parameters[5]. Sharma and Glemmestad [1] include a head curve plot; the result in Figure 3 based on a dimensionless model is identical to their plot.

---

[5] Here, a_j is dimensionless, while in Sharma [3] his parameters a_j have dimensions. This implies that the values of a_j here are different from those of a_j in Sharma and Glemmestad [1].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

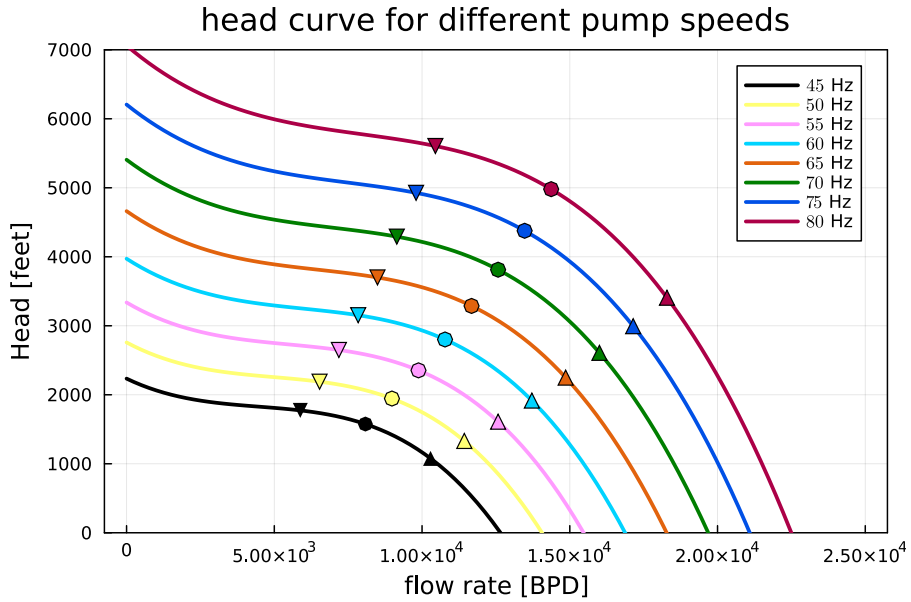

Figure 3. ESP Pump head in ft as a function of volumetric flow rate in bbl/d for selected pump speeds. Lower, best-efficiency-point, and maximum flow rates are indicated.

In addition, Sharma and Glemmestad [1] provide a model for the mechanical power requirement $\dot{W}_{\mathrm{p}}^{\mathrm{m}} = \dot{W}_{\mathrm{p}}^{\mathrm{m}}\left(\dot{V}, f_{\mathrm{p}}\right)$ for operating the pump[6], again rewritten in dimensionless form,

$$\frac{\dot{W}_{\mathrm{p}}^{\mathrm{m}}}{\dot{W}_{\mathrm{p,0}}^{\mathrm{m}}} = \left(\frac{f_{\mathrm{p}}}{f_{\mathrm{p,0}}}\right)^{3} + \sum_{j=1}^{4} b_j \left(\frac{f_{\mathrm{p}}}{f_{\mathrm{p,0}}}\right)^{3-j} \left(\frac{\dot{V}}{\dot{V}^{\varsigma}}\right)^{j}. \tag{10}$$

In Eq. 10, $\dot{W}_{\mathrm{p,0}}^{\mathrm{m}}$ is a nominal scaling power consumption to operate the pump, $b_1, \ldots, b_4$ are dimensionless model parameters, while f_{p} and $\dot{V}$ are as above.

The actual power added to the fluid is

$$\dot{W}_{\mathrm{p}} = \Delta p_{\mathrm{p}} \dot{V}, \tag{11}$$

which gives the efficiency as

$$\eta = \frac{\dot{W}_{\mathrm{p}}}{\dot{W}_{\mathrm{p}}^{\mathrm{m}}} = \frac{\Delta p_{\mathrm{p}} \dot{V}}{\dot{W}_{\mathrm{p}}^{\mathrm{m}}} \tag{12}$$

where it is assumed that $\dot{W}_{\mathrm{p}}$ and $\dot{W}_{\mathrm{p}}^{\mathrm{m}}$ have the same units. Sharma and Glemmestad [1] include an efficiency curve plot; the result in Figure 4 based on a dimensionless model is identical to their plot.

---

[6] "Brake Horse Power", BHP, in the original publication.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

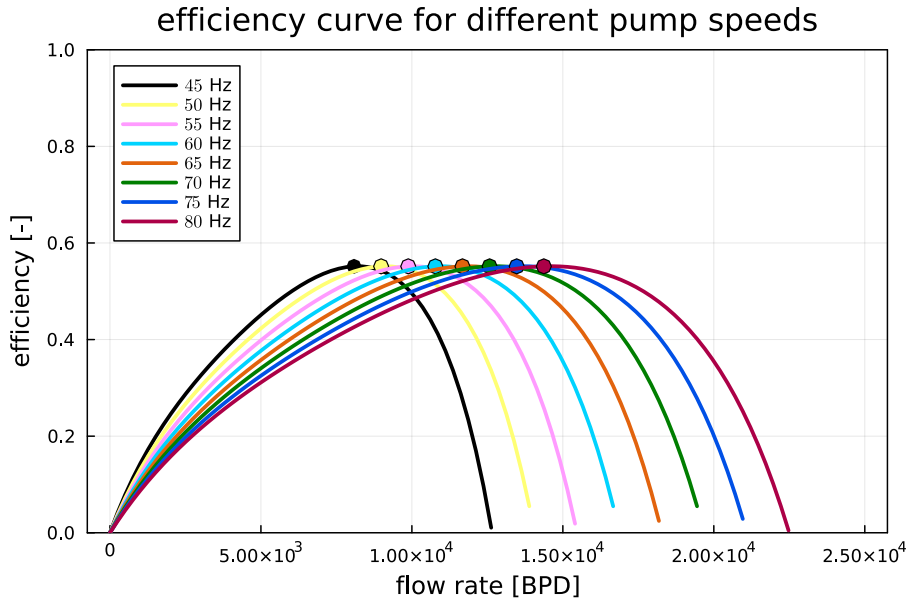

Figure 4. ESP Pump efficiency curve as a function of volumetric flow rate in bbl/d for selected pump speeds. Best-efficiency-point flow rates are indicated, and they match the peaks in the $\eta\left(\dot{V}; f_{\mathrm{p}}\right)$ plots.

Booster pump

For the *booster pump* in horizontal pipes, a simpler model is suggested in Sharma and Glemmestad [1], rewritten in dimensionless form as

$$\frac{\Delta p_{\mathrm{bp}}\left(f_{\mathrm{bp}}\right)}{\Delta p_{\mathrm{bp},0}} = \left(\frac{f_{\mathrm{bp}}}{f_{\mathrm{bp},0}}\right)^2 \tag{13}$$

Here, $\Delta p_{\mathrm{bp}}\left(f_{\mathrm{bp}}\right)$ is the pressure increase at the given pump frequency/speed f_{bp}, in the same unit as $\Delta p_{\mathrm{bp},0}$ — which is the pressure increase at the nominal pump frequency $f_{\mathrm{bp},0}$.

Pump control input

In reality, the pump speed (f_{p}, f_{bp}) is not a control input. Instead, a motor is used to control the torque applied to the aggregate of motor and pump.

In Sharma [3], a model of the induction motor driving the ESP is developed. The experience [3] is that the motor dynamics is much faster than that of the mechanical system; hence in most of his work, Sharma [3] neglects the motor dynamics. However, it is not clear whether Sharma [3] considers the mechanical dynamics of accelerating the pump itself. This dynamics would be described by the kinetic energy balance in rotational form (the "swing equation"), which can be written as

$$\frac{\mathrm{d}K}{\mathrm{d}t} = P_{\mathrm{i}} - \dot{W}_{\mathrm{p}}^{\mathrm{m}}$$

with kinetic energy K

$$K = \frac{1}{2} J_{\mathrm{t}} \omega_p^2,$$

J_{t} is the total moment of inertia for the pump, the motor, and a possible flywheel, while P_{i} is the input power from the motor (control input), and $\dot{W}_{\mathrm{p}}^{\mathrm{m}}$ is as in Eq. 10.

In Sharma [3], the moment of inertia is approximately given as $J \approx 71\,\mathrm{kg\,m^2}$. It is not clear whether this is the motor moment of inertia or the total moment of inertia. However, using such a moment of inertia leads to a pump time constant which is still much faster than the dynamics of the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

flow $\dot{V}_v$, etc., hence the pump/motor dynamics is neglected here, and for simplicity, it is assumed that f_p is a control input.

2.5. Valve models

Sharma and Glemmestad [1] base their valve models on the ANSI/ISA S75.01 standard[7]. Here, instead a dimensionless description is proposed with extension to a control input,

$$\dot{m} = \dot{m}_v^c \cdot f(u_v) \frac{\rho_i}{\rho_e} \sqrt{\frac{(p_i - p_e)/p^\varsigma}{\rho_i/\rho^\varsigma}} \tag{14}$$

where $\dot{m}_v^c$ is the valve mass flow rate capacity, $u_v \in [0,1]$ is the valve control signal, $f : [0,1] \to [0,1]$ is the valve characteristics, ρ_i, ρ_e are influent and effluent densities, respectively, p_i, p_e are influent and effluent pressures, respectively, while $\rho^\varsigma, p^\varsigma$ are scaling density and pressure, respectively.

2.6. Friction loss

The friction drop along the pipe can be given by the Darcy-Weisbach model[8] as

$$\frac{\Delta p_f}{\ell} = f_D \frac{\rho}{2} \frac{v^2}{D} \tag{15}$$

where f_D is Darcy's friction factor given by Colebrook's[9] implicit expression. One explicit *approximation* to Colebrook's expression is due to Swamee and Jain [9],

$$\frac{1}{\sqrt{f_D}} = -2 \cdot \log_{10}\left(\frac{5.74}{N_{Re}^{0.9}} + \frac{\epsilon/D}{3.7}\right), \tag{16}$$

where N_{Re} is the Reynolds number,

$$N_{Re} = \frac{\rho v D}{\mu} = \frac{vD}{\nu}, \tag{17}$$

μ is *dynamic* viscosity, ν is *kinematic* viscosity, and ϵ is the "roughness height" of the pipe internal surface. Linear velocity v is related to volumetric flow rate $\dot{V}$ by

$$\dot{V} = vA \tag{18}$$

where A is the cross-sectional area of the pipe.

2.7. Why dimensionless models?

Example: ESP pump model

As a first example, consider the ESP model in Eq. 9. In the original formulation in Sharma and Glemmestad [1],[10]

$$h_p = \bar{a}_0 \frac{f_p}{f_{p,0}} + \bar{a}_1 \frac{f_p}{f_{p,0}} \dot{V} + \bar{a}_2 \dot{V}^2 + \bar{a}_3 \frac{f_{p,0}}{f_p} \dot{V}^3, \tag{19}$$

---

7 `http://integrated.cc/cse/ISA_750101_SPBd.pdf`
8 E.g., https://en.wikipedia.org/wiki/ Darcy%E2%80%93Weisbach_equation
9 The Colebrook equation, or sometimes known as the Colebrook-White equation.
10 Slight change in notation.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

where parameters $\bar{a}_j$ have rather complicated units and the equation is hard-coded to assume[11] $[\dot{V}] = \text{gal/min}$, while $[h_\text{p}] = \text{ft}$. In practice, either the rest of the model has to be posed using these units, or one has to operate with several copies of variables, e.g., $\dot{V}_{\text{g/m}}$ and $\dot{V}$, and remember to correctly convert between these versions of the flow rate. Both of these approaches are error-prone, and also require several versions of variables.

A much better solution is to write the model in dimensionless form. The simplest way to do this for the model in Eq. 19, is as

$$\frac{h_\text{p}}{h_{\text{p},0}} = \tilde{a}_0 \frac{f_\text{p}}{f_{\text{p},0}} + \tilde{a}_1 \frac{f_\text{p}}{f_{\text{p},0}} \frac{\dot{V}}{\dot{V}^\varsigma} + \tilde{a}_2 \left(\frac{\dot{V}}{\dot{V}^\varsigma}\right)^2 + \tilde{a}_3 \frac{f_{\text{p},0}}{f_\text{p}} \left(\frac{\dot{V}}{\dot{V}^\varsigma}\right)^3 . \tag{20}$$

If we choose $h_{\text{p},0} \equiv 1\,\text{ft}$ and $\dot{V}^\varsigma \equiv 1\,\text{gal/min}$, then $\tilde{a}_j \equiv \bar{a}_j$. Suppose we want to generate the plot in Figure 3. Because that figure plots h_p in ft (the "native" unit), while the flow rate $\dot{V}$ is given in bbl/d ("native" unit is gal/min), this result is produced by choosing $\dot{V}^\varsigma = 1\,\text{gal/min} = 34.29\,\text{bbl/d}$, which can easily be found using the WolframAlpha app[12], Figure 5.

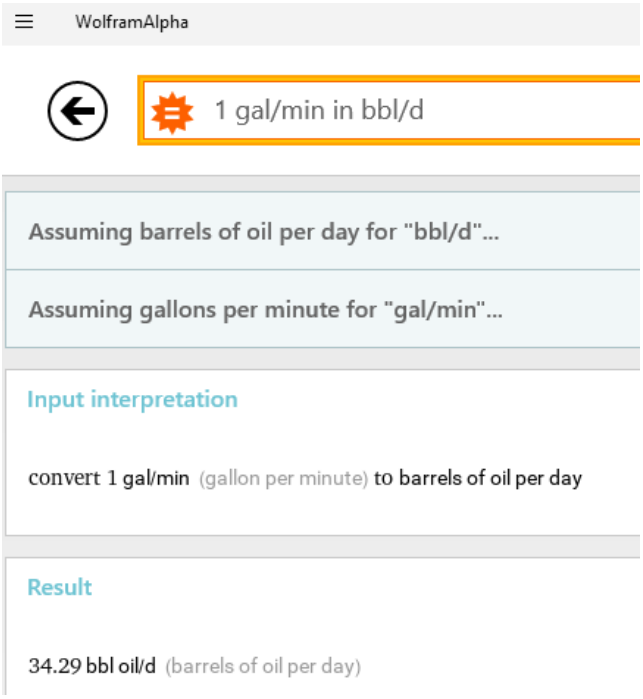

Figure 5. Converting scaling flow rate $\dot{V}^\varsigma$ from 1 gal/min to bbl/d.

In practice, it may be better to choose a more natural scaling unit, e.g., SI units. In that case, it is necessary to change parameters $\bar{a}_j$; for the parameters in Eq. 9, the scaling parameters are in SI units, where $\bar{a}_j \to a_j$ and a_j is given in Table A2. To find a_j, choose $h_{\text{p},0} = 1\,\text{ft} = 0.3048\,\text{m}$, $\dot{V}^\varsigma = 1\,\text{gal/min} = 6.309 \cdot 10^{-5}\,\text{m}^3/\text{s}$, and write Eq. 20 as

$$\underbrace{\frac{h_\text{p}}{h_{\text{p},0}\tilde{a}_0}}_{\to h_{\text{p},0}} = \frac{f_\text{p}}{f_{\text{p},0}} + \underbrace{\frac{\tilde{a}_1}{\tilde{a}_0 \dot{V}^\varsigma}}_{a_1} \frac{f_\text{p}}{f_{\text{p},0}} \dot{V} + \underbrace{\frac{\tilde{a}_2}{\tilde{a}_0 \left(\dot{V}^\varsigma\right)^2}}_{a_2} \dot{V}^2 + \underbrace{\frac{\tilde{a}_3}{\tilde{a}_0 \left(\dot{V}^\varsigma\right)^3}}_{a_3} \frac{f_{\text{p},0}}{f_\text{p}} \dot{V}^3,$$

[11] With quantity x, $[x]$ is the unit of the quantity.
[12] E.g., Microsoft Store

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

where the new $h_{p,0}$ is given in unit m, a_1, a_2, a_3 are the new, dimensionless parameters in Eq. 9, while new scaling flow rate $\rightarrow \dot{V}^{\varsigma} = 1\,\mathrm{m}^3/\mathrm{s}$. With this modified correlation $(a_1, \ldots, a_3)$ for h_p using SI units, the dimensionless form is as in Eq. 9.

Example: control valve

The ANSI/ISA S75.01 standard[13] for *compressible* (i.e., $\dot{m}_i = \dot{m}_e = \dot{m}$, $\rho_i \neq \rho_e$), non-choked fluids without fitting is

$$C = \frac{\dot{m}}{N_6 \frac{\rho_i}{\rho_e} \sqrt{\frac{p_i - p_e}{\rho_i}}}. \tag{21}$$

Here, C is the valve coefficient, $\dot{m}$ is the mass flow rate through the valve, ρ_i is the influent density, ρ_e is the effluent density, p_i is the influent pressure, p_e is the effluent pressure, N_6 is used to handle unit conversion. Typically, tabular values for N_6 are given which are valid for different combinations of units for $\dot{m}$, ρ, and p. This makes change of units rather complicated. A dimensionless formulation as in Eq. 14 greatly simplifies the use of the valve model in different units, and also includes a control valve characteristic.

3. Dynamic model

3.1. Balance laws

The model is based on the total mass balance (manifold) and the linear momentum balance (pipes). The total mass balance is expressed as

$$\frac{\mathrm{d}m}{\mathrm{d}t} = \dot{m}_i - \dot{m}_e \tag{22}$$

where m is accumulated mass in the system, t is time, $\dot{m}$ is mass flow rate, and indices (i, e) denote influent and effluent, respectively.

The linear momentum balance is

$$\frac{\mathrm{d}\mathfrak{m}}{\mathrm{d}t} = \dot{\mathfrak{m}}_i - \dot{\mathfrak{m}}_e + F, \tag{23}$$

where $\mathfrak{m}$ is linear momentum given as $\mathfrak{m} = mv$ with linear velocity v, $\dot{\mathfrak{m}}$ is momentum flow rate given as $\dot{\mathfrak{m}} = \dot{m}v$, and F is total force. With constant fluid density, $\dot{\mathfrak{m}}_i = \dot{\mathfrak{m}}_e$, and the momentum balance reduces to Newton's law, $\frac{\mathrm{d}\mathfrak{m}}{\mathrm{d}t} = F$.

3.2. Vertical pipes with ESP

We assume constant density in the pipes, causing volumetric vertical flow rate $\dot{V}_v$ to be the same everywhere: $\dot{V}_h = \dot{V}_c^i = \dot{V}_v$. Furthermore, Eq. 23 reduces to Newton's law. Momentum is given as $\mathfrak{m} = \dot{m}v$ with $\dot{m} = \rho\dot{V}_v$, and v related to $\dot{V}_v$ by Eq. 18. The total force is $F = F_p + F_b - F_f - F_g$, with

- Pressure forces at inlet and outlet of the pipe,

$$F_p = p_h A - p_c^i A \tag{24}$$

- Possible pressure boost due to a pump,

$$F_b = \Delta p_p A, \tag{25}$$

with Δp_p given by Eqs. 5, 9,

[13] http://integrated.cc/cse/ISA_750101_SPBd.pdf

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

- Friction loss,

$$F_\mathrm{f} = \Delta p_\mathrm{f} A, \tag{26}$$

  with Δp_f given by Eqs. 15, 16, 17, 18,
- Flow against gravity, with a vertical height h,

$$F_\mathrm{g} = \Delta p_\mathrm{g} A, \tag{27}$$

  with

$$\Delta p_\mathrm{g} = \rho_\mathrm{v} g h. \tag{28}$$

In addition, we need information about how flow rate $\dot{V}_\mathrm{v}$ relates to the bottom hole pressure via the productivity index, Eq. 4, and how the flow rate $\dot{V}_\mathrm{v}$ relates to the choke valve flow, Eq. 14.

The most structured formulation would be to pose the momentum balance (here: Newton's law) as the differential equation, and add all necessary algebraic equations. However, the OpenModelica DAE solver struggles with such a formulation: the valve equation Eq. 14 is implicit in pressure difference; in the iteration to find $\Delta p_\mathrm{v} = p_\mathrm{i} - p_\mathrm{e}$, if Δp_v becomes negative, the square root gives a complex number, and the simulation crashes.[14] Instead, the differential variable has been changed to $\dot{V}_\mathrm{v}$; then the valve equation can be inverted and expressed as $\Delta p_\mathrm{v} \propto \dot{V}_\mathrm{v}^2$.

The following formulation is used in OpenModelica and ModelingToolkit:

$$\frac{\mathrm{d}\dot{V}_\mathrm{v}}{\mathrm{d}t} = \frac{p_\mathrm{h} - p_\mathrm{i}^\mathrm{c} + \Delta p_\mathrm{p} - \Delta p_\mathrm{f} - \Delta p_\mathrm{g}}{\rho_\mathrm{v} \ell / A} \tag{29}$$

$$\rho_\beta^0 = \chi_\mathrm{w} \rho_\mathrm{w} + (1 - \chi_\mathrm{w})\, \rho_\mathrm{o} \tag{30}$$

$$\nu = \chi_\mathrm{w} \nu_\mathrm{w} + (1 - \chi_\mathrm{w})\, \nu_\mathrm{o} \tag{31}$$

$$\mu = \rho_\beta^0 \nu \tag{32}$$

$$\rho_\mathrm{v} = \rho_\beta^0 \exp\left(\beta_T \left(p_\mathrm{c}^\mathrm{i} - p_\beta^0\right)\right) \tag{33}$$

$$p_\mathrm{h} = p_\mathrm{f} - p_\mathrm{pi}^\varsigma \frac{\dot{V}_\mathrm{v}}{\dot{V}_\mathrm{pi}^\mathrm{c}} \tag{34}$$

$$\dot{m}_\mathrm{v} = \rho_\mathrm{v} \dot{V}_\mathrm{v} \tag{35}$$

$$p_\mathrm{i}^\mathrm{c} = p_\mathrm{m} + p_\mathrm{v}^\varsigma \frac{\rho_\mathrm{v}}{\rho_\mathrm{v}^\varsigma} \left(\frac{\dot{m}_\mathrm{v}}{\dot{m}_\mathrm{v}^\mathrm{c}}\right)^2 \frac{1}{f_\mathrm{c}^2\,(u_\mathrm{v})} \tag{36}$$

$$\begin{aligned} h_\mathrm{p} = h_{\mathrm{p},0} \Bigg(& \left(\frac{f_\mathrm{p}}{f_{\mathrm{p},0}}\right)^2 + a_1 \frac{f_\mathrm{p}}{f_{\mathrm{p},0}} \frac{\dot{V}_\mathrm{v}}{\dot{V}^\varsigma} \\ & + a_2 \left(\frac{\dot{V}_\mathrm{v}}{\dot{V}^\varsigma}\right)^2 + a_3 \frac{f_{\mathrm{p},0}}{f_\mathrm{p}} \left(\frac{\dot{V}_\mathrm{v}}{\dot{V}^\varsigma}\right)^3 \Bigg) \end{aligned} \tag{37}$$

$$\Delta p_\mathrm{p} = \rho_\mathrm{v} g h_\mathrm{p} \tag{38}$$

[14] It was not tested whether ModelingToolkit can handle this implicit algebraic equation.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

$$v_{\mathrm{v}} = \frac{\dot{V}_{\mathrm{v}}}{A} \tag{39}$$

$$N_{\mathrm{Re}} = \frac{v_{\mathrm{v}} d_{\mathrm{v}}}{\nu_{\mathrm{v}}} \tag{40}$$

$$f_{\mathrm{D}}^{\mathrm{v}} = \frac{1}{4\left(\log_{10}\left(\frac{5.74}{N_{\mathrm{Re}}^{0.9}} + \frac{\epsilon_{\mathrm{v}}/d_{\mathrm{v}}}{3.7}\right)\right)^2} \tag{41}$$

$$\Delta p_{\mathrm{f}} = \ell \cdot f_{\mathrm{D}}^{\mathrm{v}} \frac{\rho_{\mathrm{v}}}{2} \frac{v_{\mathrm{v}}^2}{d_{\mathrm{v}}} \tag{42}$$

$$\Delta p_{\mathrm{g}} = \rho_{\mathrm{v}} g h. \tag{43}$$

If we only consider the model of a single vertical pipe, we need to specify (i) initial state (e.g., $\dot{V}_{\mathrm{v}}$), (ii) all "input" variables, i.e., p_{f}, f_{p}, p_{m}, and possibly water cut χ_{w}, and (iii) all parameters, i.e., ρ_{w}, ρ_{o}, ν_{w}, ν_{o}, p_{β}^{0}, ℓ, A, $p_{\mathrm{pi}}^{\varsigma}$, $\dot{V}_{\mathrm{pi}}^{\mathrm{c}}$, $p_{\mathrm{v}}^{\varsigma}$, $\rho_{\mathrm{v}}^{\varsigma}$, $\dot{m}_{\mathrm{v}}^{\mathrm{c}}$, $h_{\mathrm{p},0}$, $f_{\mathrm{p},0}$, $\dot{V}^{\varsigma}$, a_1,a_2, a_3, g, d_{v}, ν_{v}, ϵ_{v}, h.

3.3. Manifold

We assume a perfectly mixed manifold. Assuming constant manifold volume V_{m}, and adding water at flow rate $\dot{V}_{\mathrm{w}}$ to dilute the fluid to a specified manifold water cut $\chi_{\mathrm{w}}^{\mathrm{m}}$, thus reducing friction loss in the pipe towards separator, $\dot{V}_{\mathrm{w}}$ must be approximately

$$\dot{V}_{\mathrm{w}} = \frac{\chi_{\mathrm{w}}^{\mathrm{m}} - \chi_{\mathrm{w}}}{1 - \chi_{\mathrm{w}}^{\mathrm{m}}} \dot{V}_{\mathrm{v}}. \tag{44}$$

Total mass balance for the manifold can then be expressed as

$$\frac{\mathrm{d}p_{\mathrm{m}}}{\mathrm{d}t} = \frac{1}{\rho_{\mathrm{m}} V_{\mathrm{m}} \beta_T} \left(\rho_{\mathrm{v}} \dot{V}_{\mathrm{v}} + \rho_{\mathrm{w}} \dot{V}_{\mathrm{w}} - \rho_{\mathrm{m}} \dot{V}_{\mathrm{t}}\right) \tag{45}$$

$$\rho_{\beta}^{0} = \chi_{\mathrm{w}}^{\mathrm{m}} \rho_{\mathrm{w}} + \left(1 - \chi_{\mathrm{w}}^{\mathrm{m}}\right) \rho_{\mathrm{o}} \tag{46}$$

$$\rho_{\mathrm{m}} = \rho_{\beta}^{0} \exp\left(\beta_T \left(p_{\mathrm{m}} - p_{\beta}^{0}\right)\right) \tag{47}$$

$$\dot{V}_{\mathrm{w}} = \frac{\chi_{\mathrm{w}}^{\mathrm{m}} - \chi_{\mathrm{w}}}{1 - \chi_{\mathrm{w}}^{\mathrm{m}}} \dot{V}_{\mathrm{c}}^{\mathrm{i}} \tag{48}$$

In practice, the water cut χ_{w} and flow rate $\dot{V}_{\mathrm{c}}^{\mathrm{i}}$ are not known perfectly, and it is necessary to use a feedback control system to manipulate $\dot{V}_{\mathrm{w}}$ instead of using Eq. 44.

For the manifold model, we must know (i) the initial manifold pressure, (ii) the vertical inflow $\dot{V}_{\mathrm{v}}$ and the horizontal transport flow $\dot{V}_{\mathrm{t}}$ from manifold to separator, as well as manifold water cut $\chi_{\mathrm{w}}^{\mathrm{m}}$, and (iii) parameters.

3.4. Transport pipe

For simplicity, we will neglect the separator inlet valve, and assume that $p_{\mathrm{s}}^{\mathrm{i},j} \equiv p_{\mathrm{s}}$. It is straightforward to reverse this assumption.

The model of the horizontal pipe from manifold to separator is almost identical to the vertical pipe from reservoir to manifold. The essential differences are (i) no gravity pressure drop, (ii) simpler

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

booster pump model, (iii) neglecting pressure drop from pipe into separator, (iv) no need for a production index model. The complete model is

$$\frac{\mathrm{d}\dot{V}_\mathrm{t}}{\mathrm{d}t} = \frac{p_\mathrm{m} - p_\mathrm{s} + \Delta p_\mathrm{bp} - \Delta p_\mathrm{f}^\mathrm{t}}{\rho_\mathrm{t}\ell_\mathrm{t}/A_\mathrm{t}} \tag{49}$$

$$\rho_\beta^{0,\mathrm{t}} = \chi_\mathrm{w}^\mathrm{m}\rho_\mathrm{w} + (1-\chi_\mathrm{w}^\mathrm{m})\,\rho_\mathrm{o} \tag{50}$$

$$\nu_\mathrm{t} = \chi_\mathrm{w}^\mathrm{m}\nu_\mathrm{w} + (1-\chi_\mathrm{w}^\mathrm{m})\,\nu_\mathrm{o} \tag{51}$$

$$\mu_\mathrm{t} = \rho_\beta^{0,\mathrm{t}}\nu_\mathrm{t} \tag{52}$$

$$\rho_\mathrm{t} = \rho_\beta^0 \exp\left(\beta_T\left(p_\mathrm{m} - p_\beta^0\right)\right) \tag{53}$$

$$\Delta p_\mathrm{bp} = \Delta p_{\mathrm{bp},0}\left(\frac{f_\mathrm{bp}}{f_{\mathrm{bp},0}}\right)^2 \tag{54}$$

$$v_\mathrm{t} = \frac{\dot{V}_\mathrm{t}}{A_\mathrm{t}} \tag{55}$$

$$N_{\mathrm{Re,t}} = \frac{v_\mathrm{t} d_\mathrm{t}}{\nu_\mathrm{t}} \tag{56}$$

$$f_\mathrm{D}^\mathrm{t} = \frac{1}{4\left(\log_{10}\left(\frac{5.74}{N_{\mathrm{Re,t}}^{0.9}} + \frac{\epsilon_\mathrm{t}/d_\mathrm{t}}{3.7}\right)\right)^2} \tag{57}$$

$$\Delta p_\mathrm{f}^\mathrm{t} = \ell_\mathrm{t}\cdot f_\mathrm{D}^\mathrm{t}\frac{\rho_\mathrm{t}}{2}\frac{v_\mathrm{t}^2}{d_\mathrm{t}}. \tag{58}$$

Again, we need to know the initial condition of the differential variable ($\dot{V}_\mathrm{t}$), the inputs ($\chi_\mathrm{w}^\mathrm{m}$, f_bp, p_m, p_s), and the parameters.

3.5. Combined model

For illustration, we use two vertical pipes, one manifold, and one horizontal transport pipe from manifold to separator; Sharma and Glemmestad [1] use 4 vertical pipes, one manifold, and two horizontal transport pipes. Both Modelica and Julia's ModelingToolkit have support for building classes/reusable models. Because of the similarity between the models for vertical and horizontal pipes, it would be possible to collect these in the same class/constructor and just differentiate between them with a function argument. The manifold model should be a separate class, though.

With re-usability of such classes/constructors, modeling of the combined system simply consists of (i) instantiating one model per unit (2 vertical pipes, one horizontal transport pipe, and the manifold), and (ii) connecting the various instances. Specifically, the vertical pipes should see the same manifold pressure p_m, the vertical transport pipe should have the same inlet pressure as the manifold pressure p_m, the influent volumetric flows to the manifold should be the sum of the flows from the vertical pipes and the viscosity diluting water feed $\dot{V}_\mathrm{w}$ now being

$$\dot{V}_\mathrm{w} = \frac{\sum_{i=1}^{2}\left(\chi_\mathrm{w}^\mathrm{m} - \chi_\mathrm{w}^i\right)\dot{V}_\mathrm{v}^i}{1-\chi_\mathrm{w}^\mathrm{m}}; \tag{59}$$

the effluent volumetric flow from the manifold is still $\dot{V}_\mathrm{t}$.

For a proper re-usable implementation, connections should be done using *connectors* (supported by both Modelica and ModelingToolkit). Connectors are not implemented here.

4. Simulation tools

Modelica is a mature language dating back to the 1990s; ModelingToolkit [MTK] is some 4–5 years old and is still evolving rapidly. MTK is more general than Modelica, and is also integrated in

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

the larger eco-system of Julia. Currently, MTK does not support a graphical flow-sheeting tool, and it is unclear whether MTK allows for as large models as OpenModelica. Both tools have extensive support for building libraries.

The combined model has been solved using the free languages/tools OpenModelica [10,11] and ModelingToolkit [12] for Julia. To illustrate the similarity between OpenModelica code and a current formulation using ModelingToolkit, the following listing shows parts of the Modelica code for the reservoir heel–to–manifold; to save space, *description* of quantities is only included for constant π to illustrate how it is done:

```
model Reservoir_2_Manifold
    // Model of Reservoir-to-Manifold
   //
   // Model constants
   constant Real PI = 3.151592654 "pi";
   constant Real g = 9.81;
   ...
   // Model parameters
   parameter Real ell_m = 100;
   parameter Real ell_p = 2000;
   ...
   // Initial state parameters
   parameter Real Vd_v0 = 23.15e-3;
   //
   // Declaring variables
   // -- differential variables
   Real Vd_v(start = Vd_v0, fixed = true);
   // -- depending on inputs
   Real rho_beta_0;
   ...
   Real p_c__i;
   Real p_h;
   ...
   // -- input variables
   input Real p_f;
   ...
  // Equations constituting the model
  equation
   // Balance equations
   der(Vd_v) = A*(p_h - p_c__i + Dp_p - Dp_f - rho_v*g*h)/(rho_v*ell);
   // Algebraic equations
   // -- depending on inputs
   rho_beta_0 = chi_w*rho_w + (1-chi_w)*rho_o;
   ...
   //
  end Reservoir_2_Manifold;
```

Next, the following listing shows similar parts of the ModelingToolkit code for the reservoir heel–to–manifold:

```
# Reservoir-to-manifold pipe
@mtkmodel Reservoir_2_Manifold begin
    # Model of Reservoir-to-Manifold
    #
    # Model "constants" and parameters
    @parameters begin
        # -- constants
        PI=3.141592654 , [description="pi"]
        g=9.81
        ...
        # -- parameters
        ell_m=100
        ell_p=2_000
        ...
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1


```
    end
    # Dependent variables
    @variables begin
        # -- differential variable
        Vd_v(t)=23.15e-3
        # -- algebraic variables
      rho_beta_0(t)
        ...
        p_c__i(t)=58.5e5
        p_h(t)
        ...
    end
    # Equations
    @equations begin
      # Balance equation
        Dt(Vd_v) ~ A*(p_h - p_c__i + Dp_p - Dp_f - rho_v*g*h)/(rho_v*ell)
      # Algebraic equations
      # -- depending on inputs
        rho_beta_0 ~ chi_w*rho_w + (1-chi_w)*rho_o
        ...
    end
end
```

These listings show that Modelica code and ModelingToolkit code have a high degree of similarity. A few things to note:

1. In Modelica, the independent temporal variable has a fixed name (`time`), and the time differentiation operator has a fixed name (`der`). In ModelingToolkit, both of these can be freely named by the user. In order to make unit models work together (e.g., in a standard library), it is, however, necessary to standardize on a name for time (commonly `t`); differentiation can be given a name as, e.g., `Dt = Differential(t)` or similar.
2. In Modelica, quantities need to be specified with a type (e.g., `Real`), and are prepended with a qualifier (e.g., `constant`, `parameter`) — except for variables. For Julia and ModelingToolkit, the data type is inferred, unless explicitly stated. In the code above, quantities in MTK are grouped within `begin...end` blocks in *macros* (identifiers prepended by `@`, e.g., `@parameters`).
3. Modelica has a simple way to handle implicit algebraic equations, and in many cases an initial guess of the algebraic variable is not required (see variable `p_c__i` in the Modelica code). In ModelingToolkit, initial values for unknowns after structural simplification ("states") must be provided with numeric values (see variable `p_c__i` in the MTK code).
4. In ModelingToolkit, initial values of differential variables can be changed outside of the code, hence default values can be written as `Vd_v(t)=23.15e-3`. In Modelica, only *parameters* can be changed outside of the code (after compilation), hence a parameter has been defined to hold the default initial value `Vd_v(start = Vd_v0, fixed = true`.
5. Modelica uses symbol = for mathematical equality; MTK uses symbol ~ since Julia already uses symbol = for assignment.

The default solver in OpenModelica is excellent, although here it struggled if the model is posed as a DAE formulation with *momentum* as differential variable. ModelingToolkit can use solvers from the large, high quality DifferentialEquations.jl package [13]. With ModelingToolkit, more thought is currently required when choosing solver, accuracies, etc., compared to OpenModelica. Also, OpenModelica handles step-changes in inputs well, while for the DifferentialEquations.jl solvers, it is often necessary to specify the time points where step changes occur. On the other hand, the solutions from ModelingToolkit include interpolation functions, which yields smooth solutions with considerably fewer data points than for Modelica.

Results presented in Section 5 compare numerical solutions for the Reservoir heel–to–Manifold system for ModelingToolkit vs. OpenModelica.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

OpenModelica's support for linearization and plotting can be accessed from Julia via the OMJulia API [14]. ModelingToolkit is integrated in the Julia eco-system, with support for linearization, plotting, control systems analysis, random variables, etc., and has overall more possibilities that OpenModelica if further analysis is required.

OpenModelica is currently reported to handle models up to approximately 10^6 variables/equations; various conference presentations indicate that ModelingToolkit currently can solve models of up to approximately 10^5 variables/equations.[15] The model above (`Reservoir_2_Manifold`) is reported by OpenModelica to have 15 variables (differential+algebraic) and 15 equations. In ModelingToolkit, linear equations with "observed" variables are stripped off from the model (function `structural_simplify()`) before solving the model. The above `Reservoir_2_Manifold` model in the listing is reported to have 2 "states" and 2 equations by ModelingToolkit. It is not clear whether the ModelingToolkit claim of 10^5 variables/equations is before or after the "observed" variables are stripped off.

Other commonly used languages for scientific computing are MATLAB (commercial) and Python (free). Compared to both of these languages, Julia (free) has a more extensive set of differential equation solvers[16]. Neither MATLAB nor Python offer equation based modeling languages with library/re-use support such as Modelica or ModelingToolkit; MathWorks do offer Simscape[17] (commercial) with MATLAB integration for such use, though.

5. Results

5.1. Reservoir heel to manifold

Parameters, initial conditions, and system inputs are given in Appendix A. Figure 6 shows the input variation for the Reservoir heel–to–manifold (`R2M`) case.

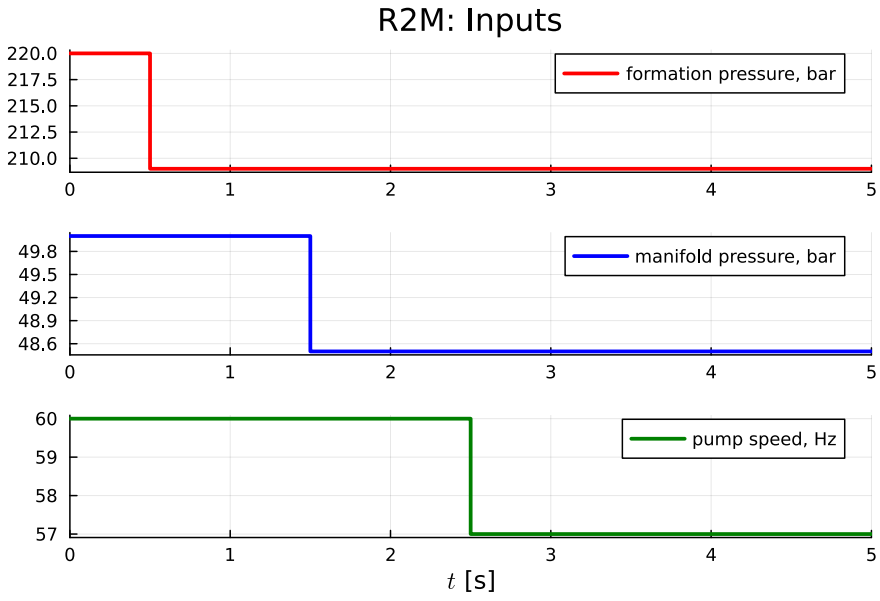

Figure 6. Input variation in experiment.

A step change in the formation pressure (red curve) as in Figure 6 is not very realistic; such changes are normally slow. The manifold pressure (blue curve) is not normally an input function,

[15] On-going work on a JuliaSimCompiler.jl for a commercial extension of Julia will increase the possible system size.
[16] Julia's DifferentialEquations.jl package can be accessed from Python and R.
[17] https://se.mathworks.com/products/simscape.html

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

but rather a dependent variable in the overall system as in Section 5.2, and thus also normally varies slowly. The pump speed (green curve) is, however, a control variable, and can change fast. Still, the inputs in Figure 6 will help provide useful information about time constants in the system.

Figure 7 shows the response in (vertical) volumetric flow rate $\dot{V}_v$, with comparison between Julia (red, solid) and OpenModelica (blue, dash-dot).

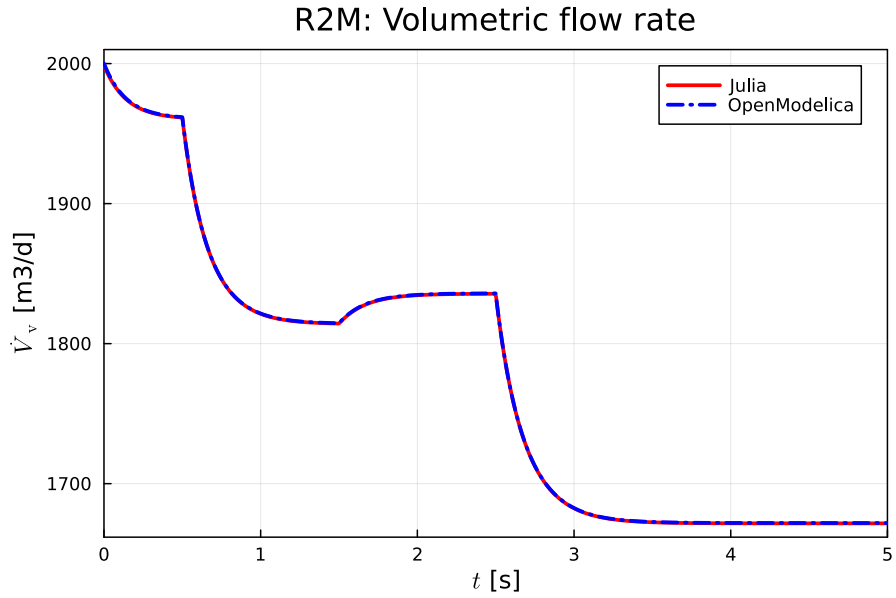

Figure 7. Response in volumetric flow rate to step inputs.

An important observation is that the volumetric flow rate is *continuous* under the step changes in Figure 6. This makes sense, in that the momentum of the fluid (oil-water) is substantial. Time constants are in the range of 0.2 –0.5 s.

Figure 8 shows the response in the choke valve inlet pressure, p_c^i.

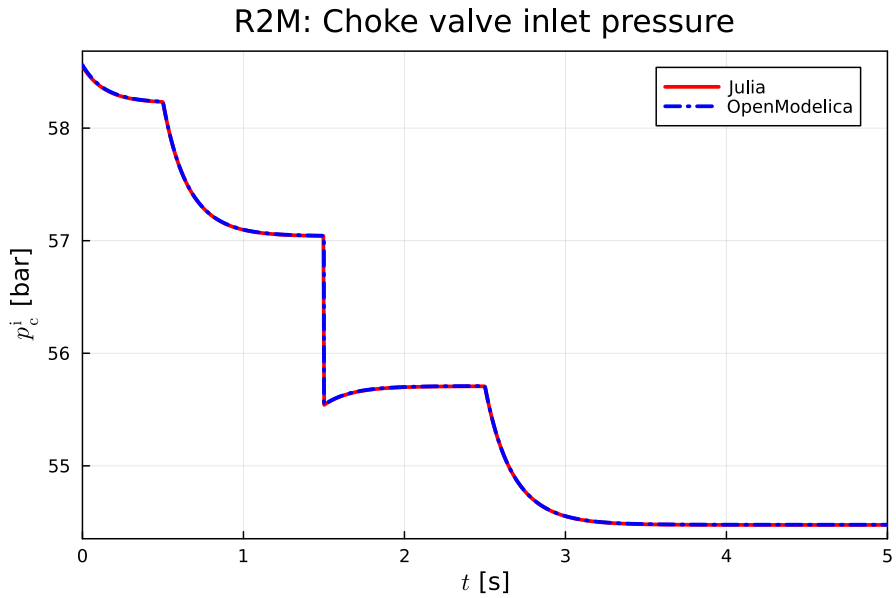

Figure 8. Response in choke valve inlet pressure to step inputs.

Observe that the choke valve inlet pressure normally is continuous under step changes, but that it changes *discontinuously* upon a step change in manifold pressure at $t = 1.5$ s. Again, this makes

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

sense: when the manifold pressure drops, Figure 6, the choke valve inlet pressure must also drop in proportion so that the flow through the valve changes continuously, see Eq. 14.

Figure 9 shows the response in the reservoir pressure heel, p_{h}.

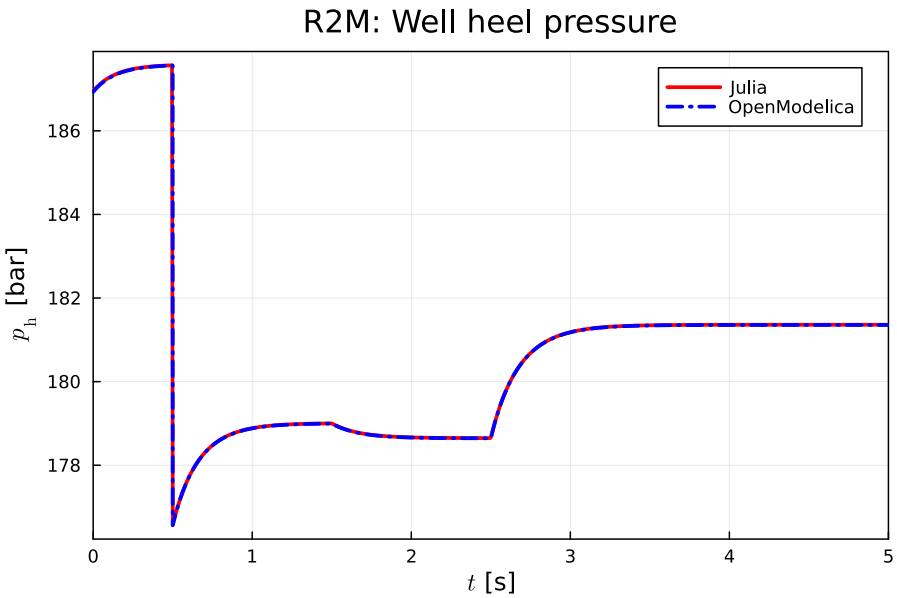

Figure 9. Response in well heel pressure to step inputs.

As noted, the formation pressure can not change in a step, but if it does, the well heel pressure must also change in proportion (i.e., discontinuously) to maintain continuity in the flow rate.

Figure 10 shows the response in the ESP pump pressure head, Δp_{p}.

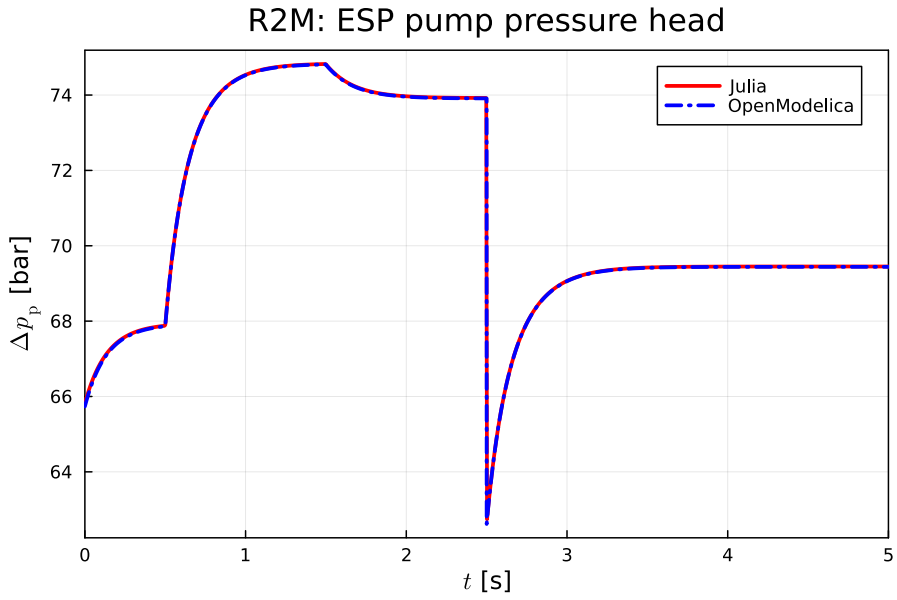

Figure 10. Response in ESP pressure increase to step inputs.

Again, the discontinuous change in pump pressure head is due to the step change in the manifold pressure, and the result makes sense.

So far, ModelingToolkit/Julia and OpenModelica have given (seemingly) identical simulation results, Figures 7–10. Figure 11 shows the pipe friction loss, Δp_{f}.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

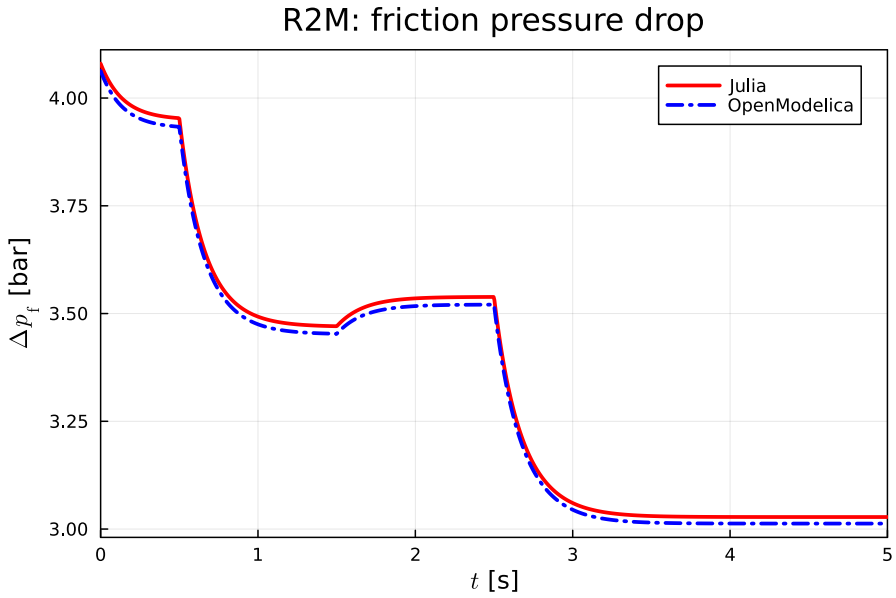

Figure 11. Response in pipe friction pressure drop.

It is interesting to contrast the magnitude of the friction loss in Figure 11 and how much smaller it is than the pressure boost in the ESP, Figure 10. Obviously, if a more realistic viscosity model had been used, and in particular if emulsification occurs [8], the friction pressure drop might increase considerable with an unfortunate mixing fraction of oil and water.

Apart from this, it is interesting to observe a slight discrepancy between the result from ModelingToolkit/Julia and OpenModelica in this case. This discrepancy is maintained during a multitude of tests with different solvers and accuracy for the DifferentialEquation.jl solvers [13] and the solvers supported by OpenModelica. It seems like there is a minor discrepancy at $t = 0$ for Δp_f, which propagates throughout the solution. Because the codes and initial conditions are the same for both implementations, a natural suspicion is that the difference is due to different handling of the implicit algebraic equation at initial time, and that Δp_f is rather sensitive to such an inaccuracy.

5.2. Reservoir heel to separator

Parameters, initial conditions, and system inputs are given in Appendix A. For vertical pipe #2, scaling pump head $h_{p,0}$ is set to 80% of the value suggested in Appendix A. For this more complete system (2 vertical pipes, one horizontal transport pipe), there is no observed difference between the solution from ModelingToolkit/Julia and OpenModelica.

Figure 12 shows the input variation for the Reservoir heel–to–*separator* (`R2S`) case; because of slower dynamics for this larger system, the locations of the step changes have been changed, and the step change in the manifold pressure (Figure 6) has been replaced by a step change in the separator pressure (Figure 12).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

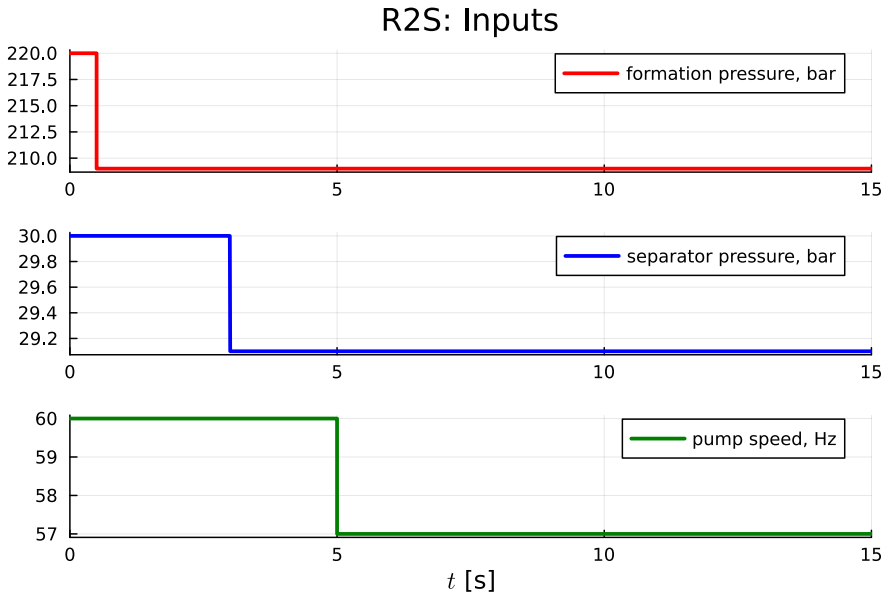

Figure 12. Input variation in experiment.

For comments on formation pressure and pump speed inputs, see Section 5.1. Although the separator pressure is not normally an input function, there may be action applied to the separator that may create relatively quick changes in the separator pressure.

Figure 13 shows the pressures in front of the choke valves for the vertical pipes, as well as the manifold pressure.

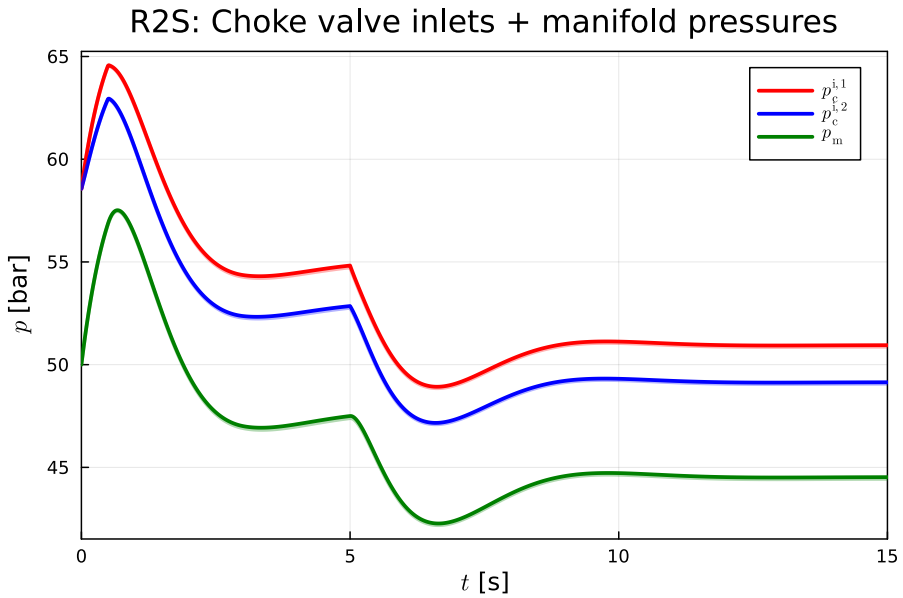

Figure 13. Pressures in front of choke valve into manifold for vertical pipes (red, blue) and manifold pressure (green).

Figure 13 demonstrates the positive pressure drop over the choke valves, and that they are different for the two valves ($\Delta p_c^j = p_c^{i,j} - p_m$). Therefore, one should expect different flows through the two valves. Because the manifold pressure is a dependent (dynamic) variable in this case, there is no discontinuity in the pressures of Figure 13.

Figure 14 displays the pressure increase over the ESP pumps in the two vertical pipes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

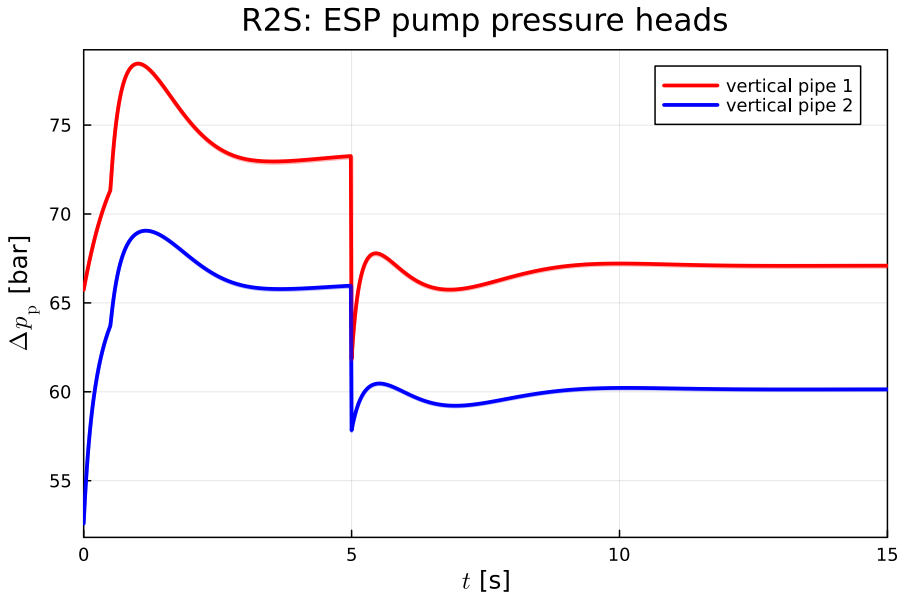

Figure 14. Electric Submersible Pump pressure heads in vertical pipes (red, blue) to compensate for gravity and friction loss.

In this figure, the sudden drop in Δp_p is due to a sudden change in the pump speed f_p, and is thus realistic.

Figure 15 shows the friction pressure drops in the two vertical pipes and in the horizontal pipe.

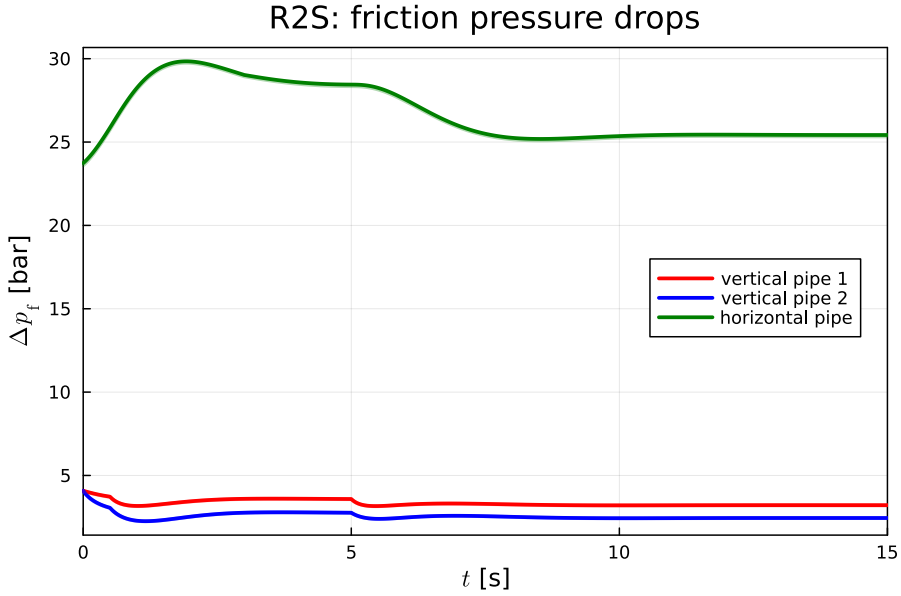

Figure 15. Friction pressure drops in vertical pipes (red, blue) from bore-well to manifold, and in horizontal pipe (green) from manifold to separator.

Here, the interesting thing is that there is no visible difference between OpenModelica and Julia solution of Δp_f for the three pipes; confer Figure 11. Of course, this could be due to the zoomed-out view, but also on close inspection of Δp_f for the initial few seconds for vertical pipe 1, there is virtually no difference.

The resulting time constants and overall behavior in Figures 13–15 are similar to those in Sharma [3]. Particularly in Figures 13 and 14, some oscillatory behavior/overshoot is noticeable. This is to be

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

expected due to the elasticity of the oil/water mixture with the given non-zero value of isothermal compressibility β_T.

Figure 16 shows vertical flow rates from reservoir to manifold in the two pipes, as well as the flow from manifold to separator (thick, solid lines), and the effect of uncertain productivity indices in Well 1, $\dot{V}_{\text{pi}}^{\text{c},1} \sim \mathcal{N}\left(7 \cdot 10^{-4}, 10^{-4}\right)$, and uncertain isothermal compressibility in the petroleum fluid, $\beta_T \sim \mathcal{U}_{[0.3/1.5 \cdot 10^9, 3/1.5 \cdot 10^9)]}$.

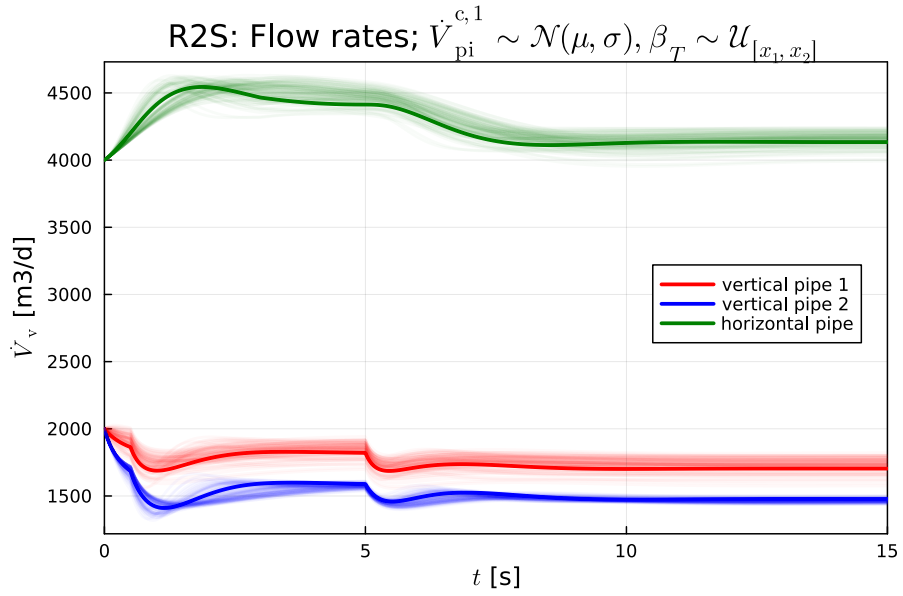

Figure 16. Vertical flow rates (red, blue) from bore-well into manifold, and horizontal flow rate (green) from manifold to separator, with uncertainty productivity capacity and isothermal compressibility.

ModelingToolkit has support for efficient Monte Carlo studies; this is comparatively more complicated using Modelica + OMJulia.

5.3. Linearized model

ModelingToolkit.jl has good support for linearization of models. To linearize a system, it is necessary to provide a model where the inputs have not been defined (`sys_p` in Figure 17), a vector of input variables (`sys_in`), a vector of output variables (`sys_in`), and an operating point (keyword `op`, value `op_0`). If the ModelingToolkitStandardLibrary.jl (similar to Modelica's Standard Library) is used, alternatively, some "virtual" inputs/outputs can be added in the form of *Analysis Points* which simplifies linearization; this is not discussed further here.

In Figure 17, linearization is performed using the `named_ss` function in ControlSystemsMTK.jl, which has similar arguments as function `linearize` in ModelingToolkit.jl.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1


```
# Linearization using ControlSystemsMTK
linsys = named_ss(sys_p, sys_in, sys_out; op=op_0)
```

```
1  dampreport(linsys)
✓ 0.0s
```

Pole	Damping Ratio	Frequency (rad/sec)	Frequency (Hz)	Time Constant (sec)
-9.77e-17	1	9.77e-17	1.55e-17	1.02e+16
+4.62e-16	-1	4.62e-16	7.35e-17	-2.17e+15
-0.518 ± 0.925im	0.489	1.06	0.169	1.93
-6.68	1	6.68	1.06	0.15
-7.27	1	7.27	1.16	0.138

```
1  tzeros(linsys)
✓ 0.0s
```

```
3-element Vector{Float64}:
  2.4528640144466883e-16
 -7.2707090955266835
 -3.9948619151982376e-16
```

Figure 17. Use of ControlSystemsMTK.jl and ControlSystems.jl for linearization and analysis.

Figure 17 suggests 6 poles/states in the system, and 3 transmission zeros. Obviously, the system has 4 states/differential variables (flow rates $\dot{V}_v$ for each of the vertical pipes, pressure p_m for the manifold, and flow rate $\dot{V}_t$ for the horizontal transport pipe). Comparing poles and zeros, one might suspect that two spurious/"infinitesimal" poles have been added together with two spurious/"infinitesimal" zeros, and that canceling out the tiny poles and zeros should give the correct transfer function. Observe also that there is a *finite* zero at -7.27 that cancels out one of the 4 true eigenvalues.

It is possible to write one's own linearization code using a (symbolic) Jacobian function applicable to ModelingToolkit.jl models. If one assumes that the original model is a DAE of index 0 or 1, this will indeed give the correct transfer function with 4 states (and one zero that cancels out one of the eigenvalues). Why does ModelingToolkit.jl produce spurious additional poles/zeros? Possibly because the linearization algorithm in ModelingToolkit.jl makes no assumption of the index of the DAE, thus producing the two spurious "infinitesimal" poles/zeros.

Figure 18 shows the Bode plot of the transfer function from ESP rotational speed (f_p)[18] to flow rate into the separator ($\dot{V}_t$), as found using Julia + ModelingToolkit.jl based on the transfer function provided by the system in Figure 17. Package ControlSystems.jl also has a function `balance_statespace()` which in combination with minimal realization provides a transfer function with 3 poles or 1 time constant and one damped resonator,

$$P(s) \approx \frac{1.094 \cdot 10^{-3}}{(1+0.15s)\left(1+2\cdot 0.489\frac{s}{1.06}+\left(\frac{s}{1.06}\right)^2\right)}; \tag{60}$$

see pale curves in Figure 18.

[18] It is assumed that the same speed is used for both ESP:s in the vertical pipes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

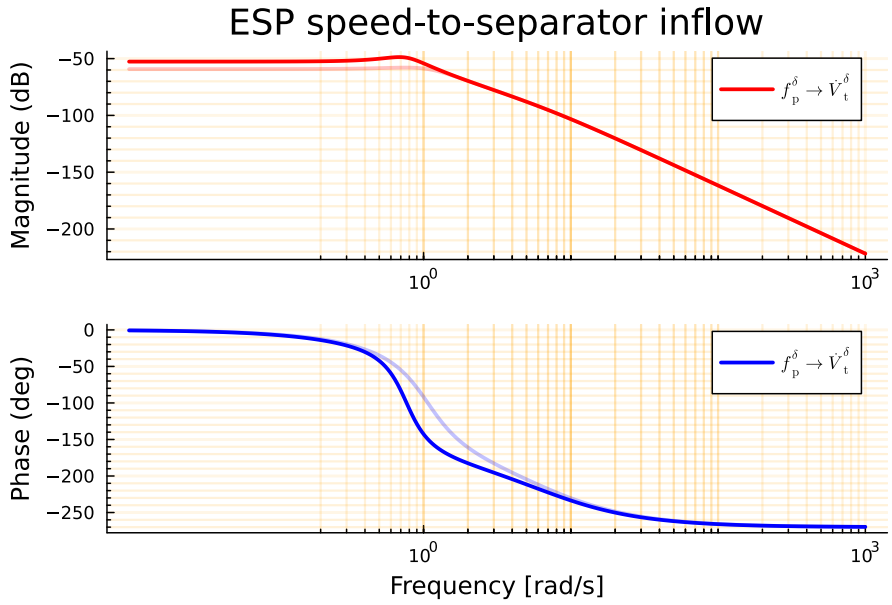

Figure 18. Bode plot of linearized model from ESP speed f_p^δ to volumetric flow $\dot{V}_t^\delta$ from manifold to separator.

It is possible to instead use Modelica+OMJulia for linearization, and then use ControlSystems.jl for Julia [similar capabilities as MATLAB's Control Toolbox] for plotting and analysis/design. However, control analysis and design is simpler to do if a Julia set-up is used also for modeling and simulation.

5.4. Single-loop controller tuning

Figure 19 shows a unit step response for the linearized model using convenience function `step()` in package ControlSystems.jl.

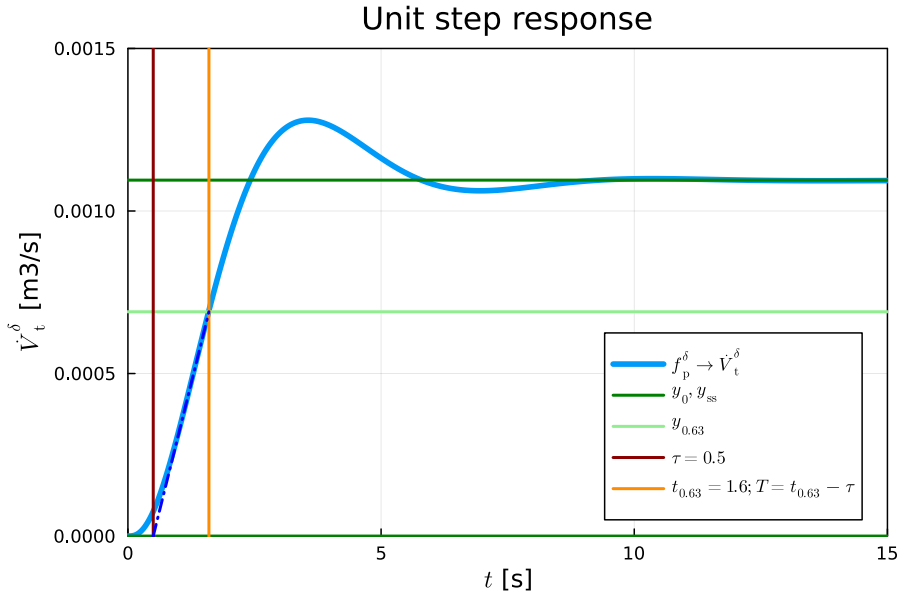

Figure 19. Response in $\dot{V}_t^\delta$ after a unit step in f_p^δ.

A crude approximation of the system is read off Figure 19 as the plant transfer function $P_\approx(s)$

$$P_\approx(s) = K\frac{\exp(-\tau s)}{s} \tag{61}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

with

$$K \approx 6.27 \cdot 10^{-3}$$
$$\tau \approx 0.5.$$

A PI controller

$$C(s) = K_{\mathrm{p}} \frac{1 + T_{\mathrm{i}} s}{T_{\mathrm{i}} s} \tag{62}$$

based on Skogestad's method [15–17] is used with

$$T_\ell = 1.3 \cdot \tau \tag{63}$$
$$\kappa = 4 \tag{64}$$
$$K_{\mathrm{p}} = \frac{1}{K(\tau + T_\ell)} \tag{65}$$
$$T_{\mathrm{i}} = \kappa(\tau + T_\ell); \tag{66}$$

where tuning parameters are: T_ℓ, closed loop time constant; κ, integral time modifier.

Figure 20 shows a unit reference step response for the closed loop system [blue line]; here, utility function `feedback(P*C)` has been used to construct the closed-loop system. Observe that the closed loop time constant T_ℓ is not achieved, and that the resulting system is rather oscillatoric.

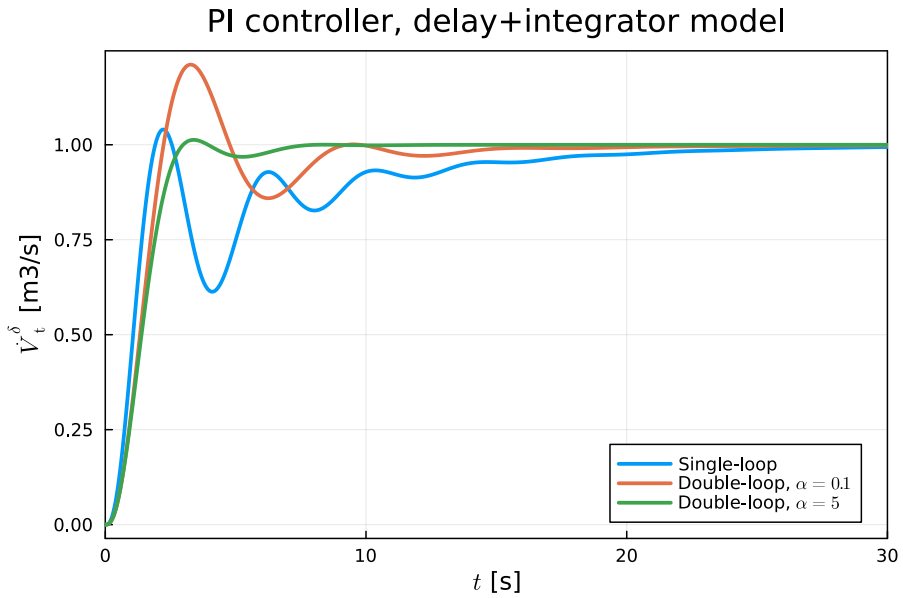

Figure 20. Response in $\dot{V}_{\mathrm{t}}^{\delta}$ after a unit reference step assuming delay + integrator process model, using Skogestad's method: single_loop [blue], double-loop with $\alpha = 0.1$ [red], and double-loop with $\alpha = 5$ [green].

The reason for the oscillatoric behavior is that single-loop tuning methods are not designed for oscillatoric systems as the one in Eq. 60.

5.5. Double-loop controller tuning

In order to better handle the oscillatoric behavior of the system, which has generic form

$$P_\zeta(s) = \frac{K}{(1 + T_1 s)\left(1 + 2\zeta \frac{s}{\omega_0} + \left(\frac{s}{\omega_0}\right)^2\right)}, \tag{67}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

an idea of Nandong [18] is pursued, where the control signal is split into an "inner" control signal

$$u_{\mathrm{i}} = -C_{\mathrm{i}}(s)\, y \tag{68}$$

and an "outer" control signal u_{o},

$$u = u_{\mathrm{i}} + u_{\mathrm{o}} \tag{69}$$

where first C_{i} is designed to reduce the oscillations in the system.

Consider the following proper inner controller

$$C_{\mathrm{i}}(s) = K_{\mathrm{c}}^{\mathrm{i}} \frac{1 + T_1 s}{1 + \alpha T_1 s}; \tag{70}$$

Inserting controller Eq. 68 into $y = P_\zeta(s)\, u$ with u as in Eq. 69 and $P_\zeta(s)$ as in Eq. 67 leads to,

$$\begin{aligned} y = P_\zeta(s)\, u &= P_\zeta(s)\left(-C_{\mathrm{i}}(s)\, y + u_{\mathrm{o}}\right) \\ &\Downarrow \\ \left(1 + P_\zeta(s)\, C(s)\right) y &= P(s)\, u_{\mathrm{o}} \end{aligned}$$

which after some re-arrangement gives:

$$\left(1 + \frac{K K_{\mathrm{c}}^{\mathrm{i}}}{1 + \alpha T_1 s} + 2\zeta \frac{s}{\omega_0} + \left(\frac{s}{\omega_0}\right)^2\right) y = \frac{K}{1 + T_1 s} u_{\mathrm{o}}.$$

For *small* values of α, $\alpha \to 0$:

$$\begin{aligned} \left(1 + K K_{\mathrm{c}}^{\mathrm{i}}\right) + 2\zeta \frac{s}{\omega_0} + \left(\frac{s}{\omega_0}\right)^2 = \left(1 + K K_{\mathrm{c}}^{\mathrm{i}}\right) \times \\ \cdots \times \left(1 + 2\zeta_{\mathrm{i}} \frac{s}{\omega_{\mathrm{i}}} + \left(\frac{s}{\omega_{\mathrm{i}}}\right)^2\right) \end{aligned}$$

where closed loop damping ζ_{i} given by

$$\begin{aligned} \zeta_{\mathrm{i}} &= \zeta / \sqrt{1 + K K_{\mathrm{c}}^{\mathrm{i}}} \\ &\Downarrow \\ K_{\mathrm{c}}^{\mathrm{i}} &= \frac{(\zeta/\zeta_{\mathrm{i}})^2 - 1}{K}. \end{aligned} \tag{71}$$

A design procedure could thus be:

1. Specify inner loop damping, $\zeta_{\mathrm{i}} \geq 1$ to provide (over-) damping.
2. Compute inner gain $K_{\mathrm{c}}^{\mathrm{i}}$ from Eq. 71.
3. Choose a "small" value for α to make the above design valid.

We choose $\zeta_{\mathrm{i}} = 1$, and compute $K_{\mathrm{c}}^{\mathrm{i}}$ from Eq. 71. Next, closing the loop from u_{o} to y where we allow for $\alpha > 0$ to have a realizable controller, the step response is as in Figure 21.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

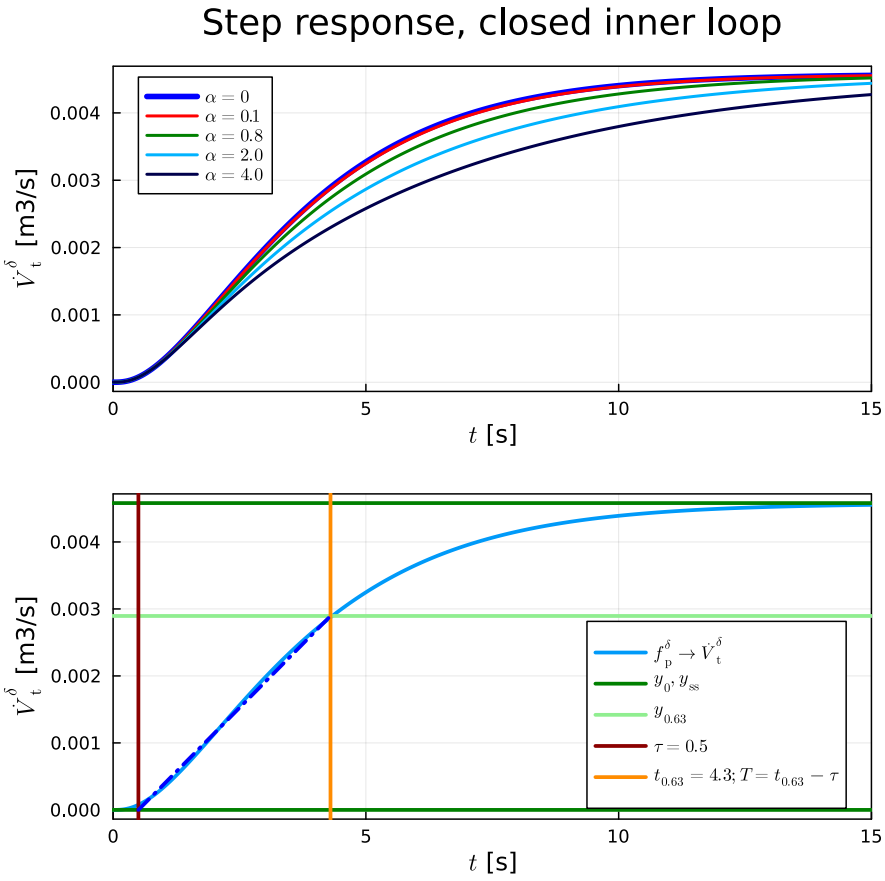

Figure 21. Response in $\dot{V}_t^\delta$ after a unit step in u_o. Upper plot: effect of varying α. Lower plot: indicating steady state and 63% rule for time constant with $\alpha = 0.1$.

Based on closed inner loop with $\alpha = 0.1$ as in the lower plot of Figure 21, we find model parameters in the model of Eq. 61 to be:

$$K \approx 0.762 \cdot 10^{-3}$$
$$\tau \approx 0.5.$$

Tuning an outer loop PI controller with Skogestad's method with an integrator + delay approximation, this time with $T_\ell = 2 \cdot \tau$, the result is as in Figure 20, red curve. Although the result is considerable faster than what is achieved with the single-loop controller (same figure, blue curve), the result is surprisingly oscillatoric.

It is interesting to observe that with the same value for $T_\ell = 2 \cdot \tau$, and changing α from $\alpha = 0.1$ to $\alpha = 5$, this gives a considerably better response, Figure 20, green curve.

5.6. Controller implementation with nonlinear model

The inner loop controller of Eq. 68 with $C_i(s)$ as in Eq. 70 admits the following state space realization

$$\frac{dz_i}{dt} = -\frac{1}{\alpha T_1} z_i + y$$
$$u_i = \frac{K_c^i}{\alpha^2 T_1}(1-\alpha) z_i - \frac{K_c^i}{\alpha} y,$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

where we must require $\alpha > 0$. The PI controller for the outer loop as in Eq. 62 admits the following state space realization:

$$\begin{aligned} e &= r - y \\ \frac{\mathrm{d}z_\mathrm{o}}{\mathrm{d}t} &= \frac{1}{T_\mathrm{i}^\mathrm{o}} e \\ u_\mathrm{o} &= K_\mathrm{p}^\mathrm{o} \left(e + z_\mathrm{o} \right) . \end{aligned}$$

With $y \leftarrow \dot{V}_\mathrm{t}$ as input to the controller together with a reference value $r \leftarrow \dot{V}_\mathrm{t}^\mathrm{ref}$, the controller produces output $u = u_\mathrm{i} + u_\mathrm{o} \rightarrow f_\mathrm{p}$ and is straightforward to implement in ModelingToolkit (or Modelica) and connect with the reservoir-to-separator model. Using the inner-outer model with $\alpha = 5$, starting in steady state and injecting a step change in $\dot{V}_\mathrm{t}^\mathrm{ref}$ gives the response as in Figure 22.

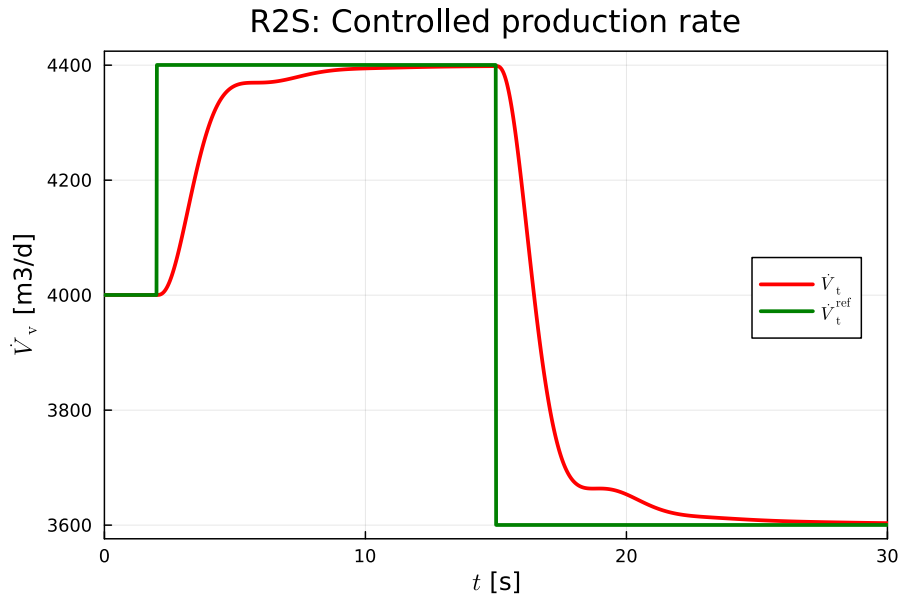

Figure 22. Response in $\dot{V}_\mathrm{t}$ after a unit step in $\dot{V}_\mathrm{t}^\mathrm{ref}$, with $\alpha = 5$ in Eq. 70.

The response in Figure 22 [red curve] is similar to the corresponding response [green curve] in Figure 20.

The response in Figure 22 is achieved with an ESP pump speed as in Figure 23.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

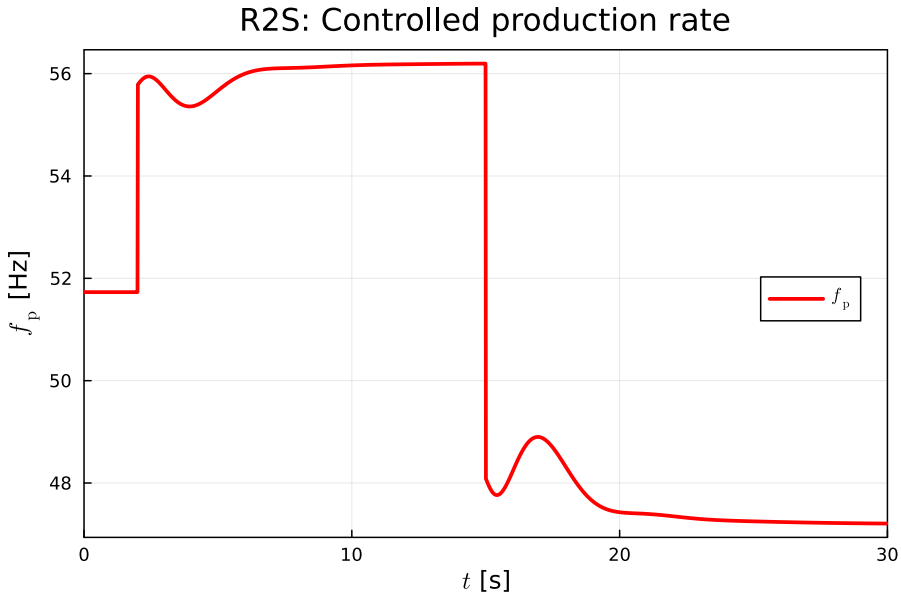

Figure 23. ESP pump speed f_p, with $\alpha = 5$ in Eq. 70.

In reality, it is not possible to "require" more production than the reservoir can produce. To handle this correctly, a more complex interaction with the reservoir than the productivity index model is required.

6. Conclusions

Sharma and Glemmestad [1], Sharma [3] provided a novel model of an ESP-lifted oil production system from reservoir to separator. In Lie [2], the model of Sharma and Glemmestad [1] was provided with a more structured model development, and with dimensionless equipment models. Also, some information that is difficult to find in the original publications (e.g., ESP model parameters) was included. A basic comparison of equation based modeling languages Modelica and ModelingToolkit for Julia was given.

Here, the work in Lie [2] is extended in several directions: (i) a couple of model typos in [2] are corrected, (ii) some more fluid details are given, (iii) more details of the ESP model is provided for reference (max/min flow rates, power consumption, efficiency), (iv) an extended discussion of pump control inputs is included, (v) a more thorough discussion is given of advantages with dimensionless models, (vi) a more detailed comparison of model implementation in Modelica vs. ModelingToolkit is provided, (vii) with a comparison of simulation results using OpenModelica vs. ModelingToolkit with Julia, (viii) more simulation results are included for reference, (ix) model linearization using ModelingToolkit with ControlSystems.jl for Julia is discussed, (x) controller tuning for the linearized model is illustrated using Skogestad's method applied to a simple integrator+time delay approximation, as well as a double-loop design to dampen oscillations, (xi) the double-loop controller is applied to the nonlinear ModelingToolkit model to confirm the controller tuning. Model parameters, inputs, and initial states are provided in Appendix A.

A key difference between the model description here and the original one [1] is the focus on dimensionless models, which greatly reduces the chance of errors. The inclusion of the ESP power consumption model and flow rate constraints/best-efficiency-point for reference, makes it possible with realistic studies of constraints in controller design. Although not implemented here, an extension of more realistic pump control inputs by inclusion of the pump-motor aggregate moment of inertia is discussed. For some systems, such an extension may be of interest.

Both Modelica and ModelingToolkit are suitable and free languages for structured model development. Modelica is a mature language and a good choice, while ModelingToolkit is a more

general language with an extensive eco-system. Model implementation is very similar for the two languages; ModelingToolkit does support more general model classes such as distributed models, stochastic models, etc., and has an extensive tool-set for control design, optimization, model fitting, etc. The numeric solutions are virtually identical for the two implementations; a minor discrepancy in friction pressure loss in one case is most likely due to a difference in handling implicit algebraic equations.

Model linearization is provided by both OpenModelica and ModelingToolkit for Julia; in the case of ModelingToolkit, the eco-system of Julia for further analysis and design can directly be used, while for OpenModelica, the linearization can be carried out via a script language API (e.g., from Julia, Python, or MATLAB) and analyzed in the host language.

For illustration purpose, linearization of the system from ESP pump speed to volumetric separator flow using ModelingToolkit/ControlSystems.jl is carried out in Julia. Then, several linear controllers are developed based on Skogestad's method for an integrator+time delay model approximation, as well for a double-loop controller to dampen oscillations in the system caused by the compressible fluid in the manifold. A re-tuning of the double-loop controller with Skogestad's method in the outer loop gives best performance. Implementation of the best controller in ModelingToolkit confirms that the linear controller works well with the nonlinear reservoir-to-separator system under the simulated experimental conditions.

The model development and discussion indicates a number of possible extensions such as (a) more realistic properties (density, viscosity), (b) allowing for distributed density along pipes[19], (c) adding a more realistic system for water dilution in the manifold, (d) inclusion of valves in manifold–separator pipes, (e) integration with reservoir models, (f) use for advanced control design with constraints, (g) use for optimization, (h) integration with a reservoir model to study how to handle stiffness, and more Such extensions will give more insight into the industrial usefulness of the model.

Funding: The economic support from The Research Council of Norway and Equinor ASA through Research Council project "308817 —Digital wells for optimal production and drainage" (DigiWell) is gratefully acknowledged.

Acknowledgments: Information/data for the original paper [1] provided by dr. Roshan Sharma and Ph.D. student Kushila Jayamanne, both at University of South-Eastern Norway, is gratefully acknowledged.

Conflicts of Interest: The author declares no conflict of interest. The background information for the study predates the funded project, and is openly available. Although the study is relevant to the funded project [DigiWell], the funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CFD	Computational Fluid Dynamics
ESP	Electric Submersible Pump
MTK	ModelingToolkit
R2M	Reservoir heel-to-Manifold
R2S	Reservoir heel-to-Separator

Appendix A. Parameters and Operating Conditions

Parameters for petroleum fluid, nominal vertical pipes, and nominal manifold+horizontal pipe are given in Tables A1–A3. Initial states are given in Table A4, while input functions are given in Table A5.

[19] ModelingToolkit for Julia has support for automatic discretization of PDEs.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 December 2023 doi:10.20944/preprints202312.1383.v1

Table A1. Parameters: petroleum liquid.

Parameter
$\beta_T = \frac{1}{1.5 \cdot 10^9} \approx 6.67 \cdot 10^{-10}\,\mathrm{Pa}^{-1}$
$p_0 = 1\,\mathrm{bar}$
$\rho_\mathrm{o} = 900\,\mathrm{kg/m^3}$
$\rho_\mathrm{w} = 1000\,\mathrm{kg/m^3}$
$\chi_\mathrm{w} = 0.35$
$\rho_0 = \chi_\mathrm{w}\rho_\mathrm{w} + (1-\chi_\mathrm{w})\,\rho_\mathrm{o}$
$\chi_\mathrm{w}^\mathrm{m} = 0.5$
$\rho_0^\mathrm{m} = \chi_\mathrm{w}^\mathrm{m}\rho_\mathrm{w} + (1-\chi_\mathrm{w}^\mathrm{m})\,\rho_\mathrm{o}$
$\nu_\mathrm{o} = 100\,\mathrm{cSt} = 100 \cdot 10^{-6}\,\mathrm{m^2/s}$
$\nu_\mathrm{w} = 1\,\mathrm{cSt} = 10^{-6}\,\mathrm{m^2/s}$

Table A2. Parameters: vertical pipe.

Parameter
$\ell^- = 100\,\mathrm{m}$
$\ell^+ = 2000\,\mathrm{m}$
$d = 0.1569\,\mathrm{m}$
$\epsilon = 0.0018\,\mathrm{inch} = 45.7^-\mathrm{m} = 45.7 \cdot 10^{-6}\,\mathrm{m}$
$\dot{V}_{\mathrm{min},0} = 14.43 \cdot 10^{-3}\,\mathrm{m^3/s}$
$\dot{V}_{\eta,0} = 19.83 \cdot 10^{-3}\,\mathrm{m^3/s}$
$\dot{V}_{\mathrm{max},0} = 25.24 \cdot 10^{-3}\,\mathrm{m^3/s}$
$h_{\mathrm{p},0} = 1210.6\,\mathrm{m}$
$f_{\mathrm{p},0} = 60\,\mathrm{Hz}$
$\dot{V}^\varsigma = 1\,\mathrm{m^3/s}$
$a_1 = -37.57$
$a_2 = 2.864 \cdot 10^3$
$a_3 = -8.668 \cdot 10^4$
$\dot{W}_{\mathrm{p},0}^\mathrm{m} = 167.733\,\mathrm{kW}$
$b_1 = 52.12$
$b_2 = -768.7$
$b_3 = 38.544 \cdot 10^3$
$b_4 = -1.534 \cdot 10^6$
$\dot{m}_\mathrm{v}^\mathrm{c} = 25.9 \cdot 10^3\,\mathrm{kg/h}$
$f(u_\mathrm{v}) = \begin{cases} 0, & u_\mathrm{v} \le 0.05 \\ \frac{11.1u_\mathrm{v}-0.556}{30}, & 0.05 < u_\mathrm{v} \le 0.5 \\ \frac{50u_\mathrm{v}-20}{30}, & 0.5 < u_\mathrm{v} \le 1 \end{cases}$
$p^\varsigma = 1\,\mathrm{bar}$
$\rho^\varsigma = 1000\,\mathrm{kg/m^3}$
$\dot{V}_\mathrm{pi}^\mathrm{c} = 7 \cdot 10^{-4}\,\mathrm{m^3/s}$

Table A3. Parameters: manifold+horizontal pipe.

Parameter
$\ell_\mathrm{m} = 500$
$d_\mathrm{m} = 0.1569$
$\ell_\mathrm{t} = 4000\,\mathrm{m}$
$d_\mathrm{t} = 0.1569\,\mathrm{m}$
$\epsilon = 0.0018\,\mathrm{inch} = 45.7^-\mathrm{m} = 45.7 \cdot 10^{-6}\,\mathrm{m}$
$\Delta p_{\mathrm{bp},0} = 10\,\mathrm{bar}$
$f_{\mathrm{bp},0} = 60\,\mathrm{Hz}$

Table A4. Nominal initial states.

Variable
$\dot{V}_\mathrm{v}(t=0) = 2000\,\mathrm{m^3/d} \approx 0.02315\,\mathrm{m^3/s}$
$p_\mathrm{m}(t=0) = 50\,\mathrm{bar} = 50 \cdot 10^5\,\mathrm{Pa}$
$\dot{V}_\mathrm{t}(t=0) = 2000\,\mathrm{m^3/d} \approx 0.02315\,\mathrm{m^3/s}$

 doi:10.20944/preprints202312.1383.v1

Table A5. Nominal inputs.

Variable
$p_{\mathrm{f}}(t) = \begin{cases} 220\,\mathrm{bar}, & t < 0.5\,\mathrm{s} \\ 0.95 \cdot 220\,\mathrm{bar}, & t \geq 0.5\,\mathrm{s} \end{cases}$
$p_{\mathrm{s}}(t) = \begin{cases} 30\,\mathrm{bar}, & t < 3\,\mathrm{s} \\ 0.97 \cdot 30\,\mathrm{bar}, & t \geq 3\,\mathrm{s} \end{cases}$
$f_{\mathrm{p}}(t) = \begin{cases} 60\,\mathrm{Hz}, & t < 5\,\mathrm{s} \\ 0.95 \cdot 60\,\mathrm{Hz}, & t \geq 5\,\mathrm{s} \end{cases}$
$u_{\mathrm{v}}(t) = 1.0$
$f_{\mathrm{bp}} = 60\,\mathrm{Hz}$

References

1. Sharma, R.; Glemmestad, B. Modeling and Simulation of an Electric Submersible Pump Lifted Oil Field. *International Journal of Petroleum Science and Technology* **2014**, *8*, 39–68.
2. Lie, B. ESP Lifted Oil Field: Core Model, and Comparison of Simulation Tools. *Scandinavian Simulation Society* **2023**, pp. 159–166. doi:10.3384/ecp200021.
3. Sharma, R. Optimal Operation of Gas Lifted and ESP Lifted Oil Fields: An Approach Based on Modeling, Simulation, Optimization and Control. PhD thesis, University of South-Eastern Norway, Kjølnes Ring 56, N-3918 Porsgrunn, Norway, 2014.
4. Binder, B.J.T.; Pavlov, A.; Johansen, T.A. Estimation of Flow Rate and Viscosity in a Well with an Electric Submersible Pump Using Moving Horizon Estimation. *IFAC-PapersOnLine* **2015**, *48–6*, 140–146.
5. Krishnamoorthy, D.; Bergheim, E.M.; Pavlov, A.; Fredriksen, M.; Fjalestad, K. Modelling and Robustness Analysis of Model Predictive Control for Electrical Submersible Pump Lifted Heavy Oil Wells. *IFAC-PapersOnLine* **2016**, *49–7*, 544–549.
6. Delou, P.d.A.; de Azevedo, J.P.A.; Krishnamoorthy, D.; Jr, M.B.d.S.; Secchi, A.R. Model Predictive Control with Adaptive Strategy Applied to an Electrical Submersible Pump in a Subsea Environment. *IFAC PapersOnLine* **2019**, *52–1*, 784–789.
7. Santana, B.A.; Fontes, R.M.; Schnitman, L.; Martins, M.A.F. An Adaptive Infinite Horizon Model Predictive Control Strategy Applied to an ESP-lifted Oil Well System. *IFAC PapersOnLine* **2021**, *54–3*, 176–181.
8. Justiniano, M.; Romero, O.J. Inversion Point of Emulsions as a Mechanism of Head Loss Reduction in Onshore Pipeline Heavy Oil Flow. *Brazilian Journal of Petroleum and Gas* **2021**, *15*, 13–24.
9. Brkić, D. Review of Explicit Approximations to the Colebrook Relation for Flow Friction. *Journal of Petroleum Science and Engineering* **2011**, *77*, 34–48.
10. Fritzson, P. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3: A Cyber-Physical Approach*, second ed.; Wiley-IEEE Press: Piscataway, NJ, 2015.
11. Fritzson, P.; Pop, A.; Asghar, A.; Bachmann, B.; Braun, W.; Braun, R.; Buffoni, L.; Casella, F.; Castro, R.; Danós, A.; Franke, R.; Gebremedhin, M.; Lie, B.; Mengist, A.; Moudgalya, K.; Ochel, L.; Palanisamy, A.; Schamai, W.; Sjölund, M.; Thiele, B.; Waurich, V.; Östlund, P. The OpenModelica Integrated Modeling, Simulation and Optimization Environment. Proceedings of the 1st American Modelica Conference; LIU Electronic Press, www.ep.liu.se: Cambridge, MA, USA, 2018.
12. Ma, Y.; Gowda, S.; Anantharaman, R.; Laughman, C.; Shah, V.; Rackauckas, C. ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling. *arXiv* **2021**. doi:10.48550/arXiv.2103.05244.
13. Rackauckas, C.; Nie, Q. DifferentialEquations.Jl — A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. *Journal of Open Research Software* **2017**, *5*. doi:10.5334/jors.151.
14. Lie, B.; Palanisamy, A.; Mengist, A.; Buffoni, L.; Sjölund, M.; Asghar, A.; Pop, A.; Fritzson, P. OMJulia: An OpenModelica API for Julia-Modelica Interaction. Proceedings of the 13th International Modelica Conference, 2019, pp. 699–708. doi:10.3384/ecp19157.
15. Skogestad, S. Simple Analytic Rules for Model Reduction and PID Controller Tuning. *Journal of Process Control* **2003**, *13*, 291–309. doi:10.1016/S0959-1524(02)00062-8.

16. Skogestad, S. Simple Analytic Rules for Model Reduction and PID Controller Tuning. *Modeling, Identification and Control: A Norwegian Research Bulletin* **2004**, *25*, 85–120. doi:10.4173/mic.2004.2.2.
17. Haugen, F. Comparing PI Tuning Methods in a Real Benchmark Temperature Control System. *Modeling, Identification and Control* **2010**, *31*, 79–91.
18. Nandong, J. Double-Loop Control Structure for Oscillatory Systems: Improved PID Tuning via Multi-Scale Control Scheme. 2015 10th Asian Control Conference (ASCC). IEEE, 2015. doi:10.1109/ascc.2015.7244476.