THE INVESTIGATION OF APPROPRIATE CONTROL ALGORITHMS FOR THE SPEED CONTROL OF WIND TURBINE HYDROSTATIC SYSTEMS

Magnus Johan Gulstad

Master i fysikk og matematikk

Oppgaven levert:	Juni 2007
Hovedveileder:	Brynjulf Owren, MATH
Biveileder(e):	Peter Chapple, Vannkraftlabratoriet, NTNU

Norges teknisk-naturvitenskapelige universitet
Institutt for matematiske fag

Oppgavetekst

As part of the ongoing programme of work on the application of hydrostatic transmissions to wind power turbines the project was concerned with the simulation of the flow process in long pipes.

The proposed diploma work is concerned with further establishing the flow behaviour in long pipes and evaluating suitable control algorithms for the control of either the speed of the turbine or hydrostatic motor. It is intended to base the work on the turbine application at Kvalsundet which has a pipe length connecting the turbine pump and hydrostatic motor that is in the order of 800m.

The control problem for this application with the generator running at synchronous speed can often be resolved by the use of hydraulic pressure feedback. However where the volume of hydraulic fluid is large or where the pipe lengths are such that the pipe line dynamics are significant then alternative algorithms are required.

The following questions should be considered in the project work:
1. To continue the flow modelling into frequency dependent friction.
2. The use of a simulation of the hydraulic system where pressure feedback provides a satisfactory technique evaluate the relative benefits of using various compensation algorithms which could include:
- Proportional, Integral and derivative systems
- Phase advance or lead-lag type systems

3. To investigate the limits of these techniques when used with pipe lengths that make normal proportional control ineffective.
4. To carry out a literature survey for alternative control techniques that can provide a successful solution to the control problem.

Oppgaven gitt: 17. januar 2007
Hovedveileder: Brynjulf Owren, MATH

THE INVESTIGATION OF APPROPRIATE CONTROL ALGORITHMS FOR THE SPEED CONTROL OF WIND TURBINE HYDROSTATIC SYSTEMS

Magnus Gulstad
Stud.techn.
Norwegian University of Science and Technology

June 28, 2007

0.1 Abstract

This report consists of two chapters. The first is concerned with a new approach to pipe flow modelling and the second has to do with the simulation of the hydrostatic system which will be applied to a wind turbine.

For the pipe flow model, the main focus has been to create a flow model which accounts for the frequency dependent friction, i.e. the fluid friction which occurs at non-steady conditions. The author is convinced that the solution to this problem lies in the velocity profile, as the friction is a direct result of the shear stresses in the pipe. At the same time, it is possible to keep track of the velocity profile in the pipe as the pressure evolves in time and space.

The new model utilizes the continuity equation for pipe flow and the equation of motion for axisymmetrical flow of a Newtonian fluid to find both a pressure distribution in the pipe and velocity profiles throughout the pipe. There are uncertainties whether the approach to find these velocity profiles done in the new model is correct.

The modelling of the hydrostatic transmission to a wind power turbine is done using SIMULINK software. The design of the system and basics of the modelling are described in the second chapter. The motor speed is regulated using a PID-controller and the generator torque is varied based on the pressure drop over the hydraulic motor. The PID-controller for motor speed seems be of good-enough quality and speed deviations are within acceptable limits.

Simulation results are given for one certain case with an initial rotor torque of $20kNm$ and an additional step torque of $20kNm$.

0.2 Acknowledgments

This diploma work has been a co-operation between the Department of Mathematical Sciences (`http://www.math.ntnu.no/english`) and the Water Power Laboratory (`http://www.ivt.ntnu.no/ept/vk/`) at the Norwegian University of Science and Technolog (`http://www.ntnu.no/english`).

Academic supervisors have been Torbjørn Nielsen, Brynjulf Owren and Peter Chapple.

List of Mathematical Symbols and their Units

Δ	Difference operator	$[dimensionless]$
γ	Spesific weight of hydraulic oil	$[kg/m^2s^2]$
μ	Dynamic viscosity	$[kg/ms]$
ν	Kinematic viscosity	$[m^2/s]$
π	Approximately equal to 3.14159	$[dimensionless]$
ρ	Density of hydraulic oil	$[kg/m^3]$
τ'	Dimensionless time$= \nu t/R^2$	$[dimensionless]$
τ''	Dimensionless time$= ta/L$	$[dimensionless]$
τ	Shear stress	$[kg/ms^2]$
τ_0	Shear stress at pipe wall	$[kg/ms^2]$
A	Area of the cross section of the pipe	$[m^2]$
a	Sonic velocity in fluid	$[m/s]$
D	Inner diameter of pipeline	$[m]$
F	Force	$[kg/ms^2]$
g	Acceleration of gravity	$[m/s^2]$
H	Head	$[m]$
h_{fl}	Friction headloss in laminar flow	$[dimensionless]$
h_{ft}	Friction headloss in turbulent flow	$[dimensionless]$

h_f	Friction headloss	$[dimensionless]$
i	Counter in axial direction	$[dimensionless]$
j	Counter in time	$[dimensionless]$
k	Counter in radrial direction	$[dimensionless]$
L	Length of pipe	$[m]$
L'	Characteristic length of pipe	$[m]$
m_i	Mass of plug i	$[kg]$
NR	Number of steps in radial direction	$[dimensionless]$
NT	Number of time steps	$[dimensionless]$
NX	Number of steps in axial direction	$[dimensionless]$
p	Pressure	$[kg/ms^2]$
R	Pipe radius	$[m]$
r	Distance from pipe axis	$[m]$
Re	Reynolds number	$[dimensionless]$
t	Time	$[s]$
V	Average velocity in a cross section of the pipe	$[m/s]$
v	Velocity as a function of x, r and t	$[m/s]$
x	Distance along the x-axis	$[m]$
A	Area of the cross section of the pipe	$[m^2]$
D_{gain}	Derivative gain	$[m/s]$
I_{gain}	Integral gain	$[m]$
P_{gain}	Propotional gain	$[ms]$
C_m	Motor friction coefficient	$[(kg/s)]$
Cd_m	Motor discharge coefficient	$[m^3/s/(N/m^2)]$
Cl_m	Motor leakage coefficient	$[m^3/s/(N/m^2)]$

D_{m-max}	Maximum motor displacement	$[m^3/rad]$
D_{m-min}	Minimum motor displacement	$[m^3/rad]$
$Omega_{m-set}$	Set motor speed	$[rad/s]$
$Omega_{m-start}$	Initial motor speed	$[rad/s]$
$Rotary_{Inertia-m}$	Motor rotary inertia	$[kgm^2]$
Cd_p	Pump discharge coefficient	$[m^3/s/(N/m^2)]$
Cf_p	Pump friction coefficient	$[kg/s]$
Cl_p	Pump leakage coefficient	$[m^3/s/(N/m^2)]$
D_p	Pump displacement	$[m^3/rad]$
$Inertia_{rt}$	Rotor rotary inertia	$[kgm^2]$
$Omega_{p-max}$	Maximum pump speed	$[rad/s]$
$Omega_{p-min}$	Minimum pump speed	$[rad/s]$
$Omega_{p-start}$	Initial pump speed	$[rad/s]$
$Step\_time$	Step time	$[s]$
$Step_{torque-rt}$	Step torque	$[Nm]$
$Torque_{rt-max}$	Maximum rotor torque	$[Nm]$
$Torque_{rt-min}$	Minimum rotor torque	$[Nm]$
$Torque_{rt}$	Rotor torque	$[Nm]$

Contents

Chapter 1

Pipe flow model

1.1 Introduction

Pipe flow with unsteady boundary conditions occurs in hydraulic systems such as water power plants and hydostatic transmissions. For these applications, the frequence dependency of the friction might be of importance to the dynamic behaviour of the systems, and accurate flow models must to be applied when modelling the systems.

One of the main focuses of this report is to develop an appropriate algorithm for the pipe flow which accounts for the frecuency dependent friction in non-steady pipe flow. The fundamentals of transmission line dynamics are given to some extent, and introduced to an appropriate mathematical flow model.

The friction coefficient normally used when modelling pipe flow is quasi steady, i.e. it is only dependent of the rate of fluid flow, and not of how the fluid flow changes with time. It is important to understand that the energy dissipation essentially is not dependent on the rate of flow, rather the shape of the velocity profile, which again is dependent of the history of the pressure gradient. This is because the fluid friction is a result of shear stresses between fluid layers. The application of the steady state friction term to non-steady flow, results in poor approximation to experimental results and the actual pressure decay and energy loss in the pipe are under-estimated.

The starting point of this work has been the modelling done in my student project in the fall semester 2006. The project work describes some of the background theory in detail and computer implementations of the method of characteristics were developped using both steady state and Zielke's friction term. An attempt was done to combine Bratland's [1][1] flow model with a solution of the governing equations (1.1) and (1.2) by means of the method of characteristics.

The main idea behind this thesis, is to compute the velocity profile in the pipe sections and to let the friction term depend on these velocity profiles. Solution methods to the Navier Stokes (1.11) and the continuity equation (1.2) have been developped and this has been the basis of the further analysis.

1.2 Background

The fundamental equations for transient pipe flow are the equation of continuity and the momentum equation. The 1D equations for motion and continuity are written in terms of V and H, the average fluid velocity and head[2] in the crossection of the pipe and yield

$$g\frac{\delta H}{\delta x} + V\frac{\delta V}{\delta x} + \frac{\delta V}{\delta t} + gh_f = 0 \tag{1.1}$$

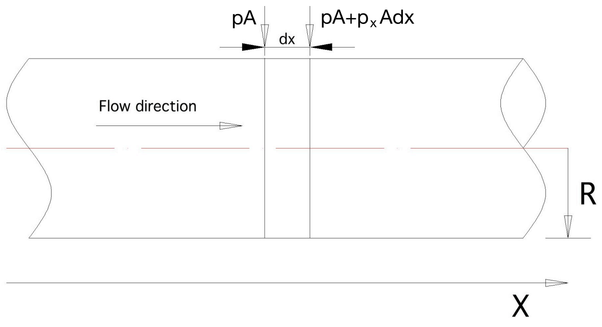
and

$$\frac{a^2}{g}\frac{\delta V}{\delta x} + V\frac{\delta H}{\delta x} + \frac{\delta H}{\delta t} = 0 \tag{1.2}$$

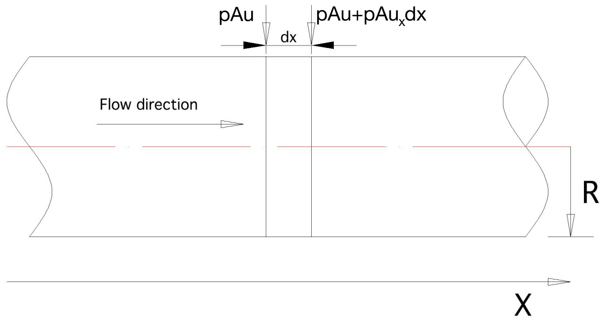
respectively. x and t are independent variables in axial direction and time. h_f designate a friction loss due to wall shear. Both equations are derived in reference [4]. These equations were solved by means of the method of characteristics, also discribed in [4], in my student project, fall 2006. This was done with both steady state friction and Zielke's friction term included.

[1]Numers in brackets designate bibliography references

[2]Meter water column, mWC



(a) Motion



(b) Continuity

Figure 1.1: Control volumes for the governing equations.

For steady state conditions, the headloss term is generally taken to be

$$h_{fl} = \frac{32\nu}{gD^2}V \tag{1.3}$$

and

$$h_{ft} = \frac{f}{2gD}|V|V \tag{1.4}$$

for laminar and turbulent flow respectively [6]. However, for oscillatory flow where the Renold's number changes with time, equations (1.3) and (1.4) do not give satisfactory prediction of the actual headloss. This has to do with the velocity profile in the pipe, which constantly changes with pressure tansients.

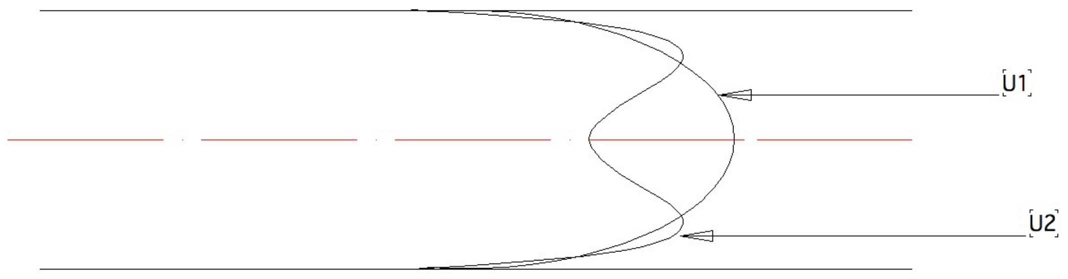


Figure 1.2: Velocity profiles in a cross section of a circular pipe. U1 is a fully developped Hagen-Poiseuille flow, while U2 is some flow profile resulting from variable pressure gradients.

Various approaches have been made to introduce the effect of frequency dependent friction to the flow model. Among others, Zielke [6] bases the friction term on past velocity changes. Bratland [1] divides the pipe into cylindrical shells and sets up a momentum equation for each shell. The headloss is then found based on the velocity differences between the shells [see figure 1.6 for illustration].

1.3 Earlier Models

1.3.1 Steady state friction

In it's most basic form, the analysis of transient pipe flow problems is solved using the equation of motion (1.1) and continuity (1.2). A friction term can be included to the equation of motion. The steady state headloss (1.3) has proved not to give adecuate prediction of the actual energy dissipation for ocillatory flow conditions. The solution to the water hammer problem with steady state friction was implemented using the method of characteristics in my student project fall 2006. The result is given in the figure below.

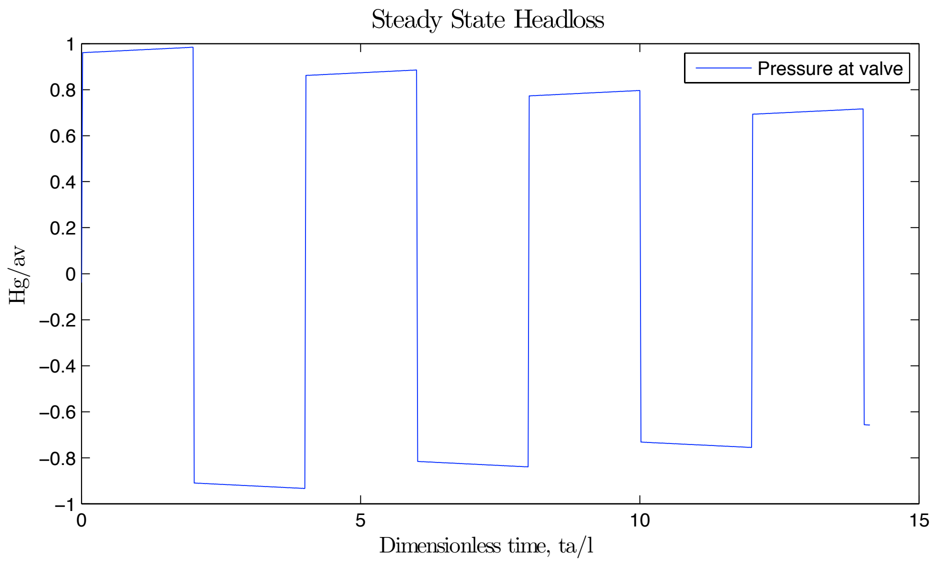


Figure 1.3: Pressure fluctuations after instantaneous valve closure using the steady state friction term. The dimensionless time is $\tau'' = ta/L$

1.3.2 Zielke's friction term

Zielke was the first who succeeded at including the effect of frequency dependent friction to a flow model. His model assumes constant pressure throughout the pipe and solves

$$g\frac{\delta H}{\delta x} + V\frac{\delta V}{\delta x} + \frac{\delta V}{\delta t} + gh_{fz} = 0 \tag{1.5}$$

and

$$\frac{a^2}{g}\frac{\delta V}{\delta x} + V\frac{\delta H}{\delta x} + \frac{\delta H}{\delta t} = 0, \tag{1.6}$$

also by means of the method of characteristics. Zielke's friction term, h_{fz}, is dependent on the history of the velocity at a cross section [6]. Zielke finds the wall shear stress given by

$$\tau_0(t) = \frac{4\nu\rho}{R}V(t) + \frac{2\nu\rho}{R}\int_0^t \frac{\delta V}{\delta t}W(t-u)\delta u, \tag{1.7}$$

where $W(t)$ is a weighing function. The equation shows that the wall shear stress is given by the instantainious shear plus a term in which weights are given to the past velocity changes. Now the headloss can be found by

$$h_{fz} = \frac{4\tau_0}{\gamma D} \tag{1.8}$$

1.3.3 Bratland's cylinder shell model

In stead of solving the governing equations (1.1) and (1.2), Bratland devides the pipe into plugs and sylindrical shells as in figure 1.6. The plugs represent the fluid and the interveaning spaces represent the pressure acting on the fluid. For this discretization the equation of forces yields

$$\frac{dv_i}{dt} = \frac{1}{m_i}\left(\frac{\pi d^2}{4}(p_{i-1} - p_i) - F_{Ri}\right), \tag{1.9}$$

where i is the counter in axial direction. The plugs are devided into cylindrical shells which

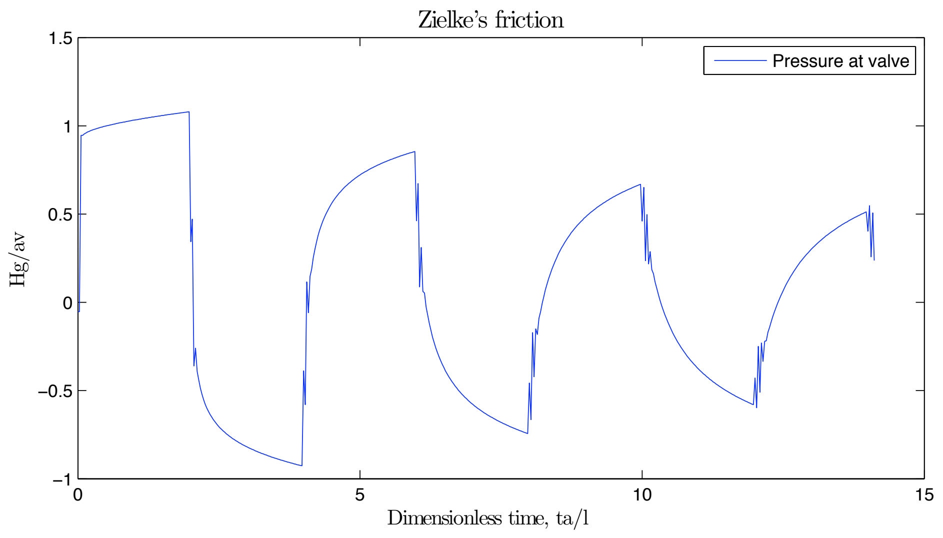


Figure 1.4: Pressure fluctuations after instantaneous valve closure using Zielke's friction term. The dimensionless time is $\tau'' = ta/L$

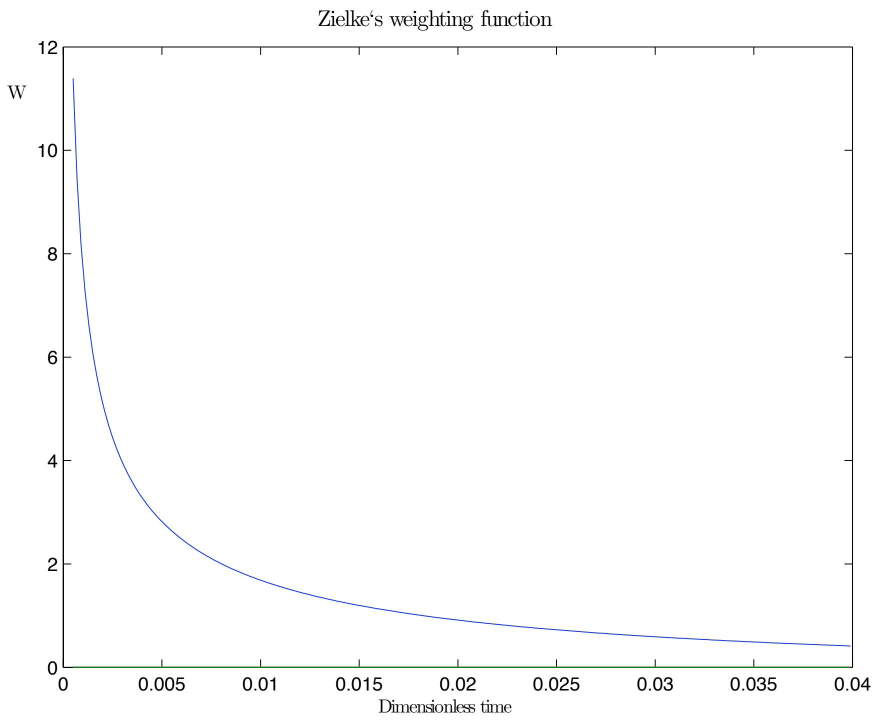


Figure 1.5: Zielkes weighting function as a function of the dimensionless time $\tau' = \nu t/R^2$.

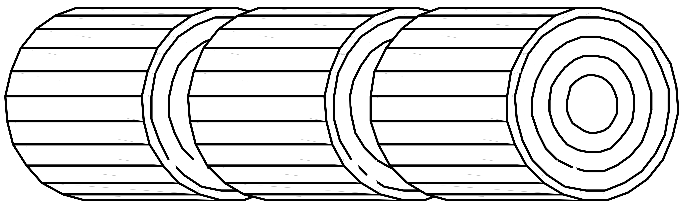


Figure 1.6: Bratland's plugs divided into cylindrical shells.

can move independently of each other, only effected by the fluicd friction which occurs between them. The friction force between two shells is found by

$$F_{i,k} = 2\pi\nu\rho r_k \Delta x \frac{v_{i,k+1} - v_{i,k}}{\Delta r}, \tag{1.10}$$

where r_k is the distance between shell number k and $k+1$.

1.4 New model

1.4.1 Solving the Navier Stokes equation

In the matlab code used, the Navier Stokes equation

$$\frac{\delta v}{\delta t} = -\frac{1}{\rho}\frac{\delta p}{\delta x} + \nu\frac{1}{r}\frac{\delta v}{\delta r} + \nu\frac{\delta^2 v}{\delta r^2} \tag{1.11}$$

is solved by means of a semi-discretization in `sd.m`, Appendix A.1.5. The function is called by v and ΔH, where v designates the velocity profile in the cross section of the pipe and ΔH is the pressure difference over the plug we are looking at. The finite-difference approximations to the space derivatives are

$$\frac{1}{r}\frac{\delta v}{\delta r}\bigg|_k^j \approx \frac{u_{k+1}^j + u_{k-1}^j}{2\Delta r(k-1)\Delta r} \tag{1.12}$$

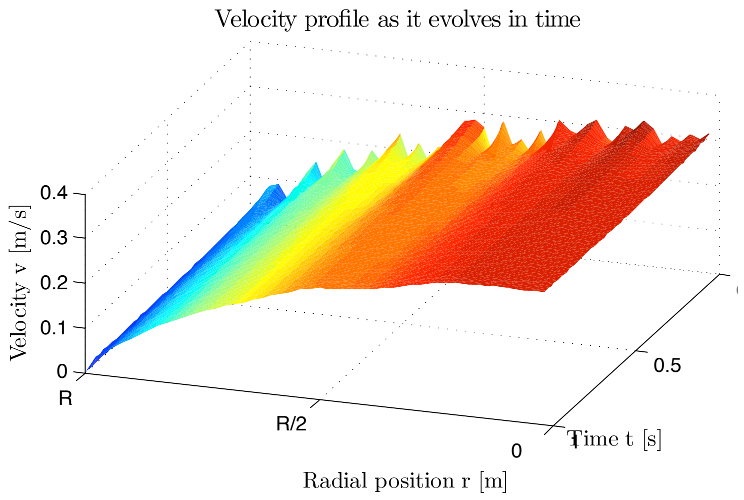
and

$$\frac{\delta^2 v}{\delta r^2}\bigg|_k^j \approx \frac{u_{k+1}^j - 2u_k^j + u_{k-1}^j}{\Delta r^2} \tag{1.13}$$

for the convective term and the laplacian respectively. k denotes the counter in radial direction and j is the current time step.

The matlab `ode45` solver, which handles ordinary differential equations is then applied to solve this system up until time $t^{j+1} = t^j + \Delta t$.

Figure 1.7 goes to show how this semi-discretization works in time and space.



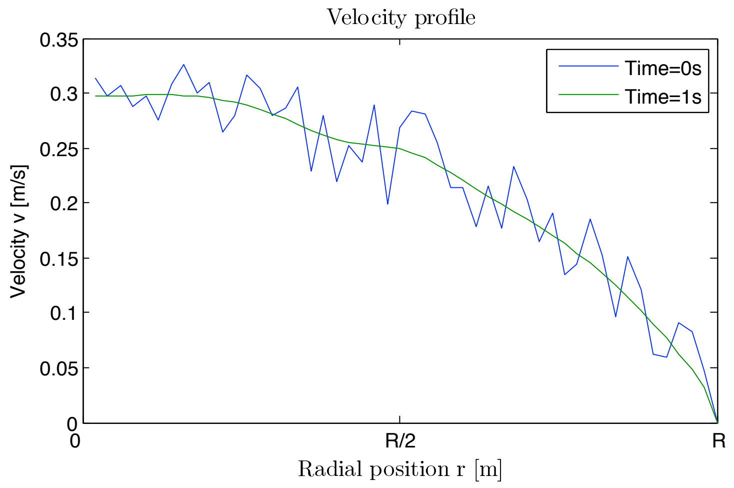


Figure 1.7: Solving the Navier Stokes equation in time and space.

1.4.2 Solving the continuity equation

The continuity equation is solved using the 1D Lax-Wendroff scheme [3]. Also here a semi-discretization has been tried, but as the result turned out to be quite oscillatory more solution methods have been tried. The Lax-Wendroff gave exactly the same answer as a semi-discretization. Various ordinary differential equation solvers have been applied to see if a semi-discretization could give acceptable results, without success. For the further analysis the Lax-Wendroff scheme has been used.

The equation to be solved is

$$\frac{\delta H}{\delta t} = -\frac{a^2}{g}\frac{\delta V}{\delta x} - V\frac{\delta H}{\delta x}. \tag{1.14}$$

Applying the Lax Wendroff scheme, the discretization yields

$$\begin{aligned} H_i^{j+1} = H_i^j - \frac{V_i^j \Delta t}{2\Delta x}\left(H_{i+1}^j - H_{i-1}^j\right) + \frac{\left(V_i^j\right)^2 \Delta t^2}{2\Delta x^2}\left(H_{i+1}^j - H_i^j + H_{i-1}^j\right) \\ \frac{a^2}{g}\left(-\frac{\Delta t}{2\Delta x}\left(V_{i+1}^j - V_{i-1}^j\right) + \frac{\Delta t^2}{2\Delta x^2}\left(V_{i+1}^j - V_i^j + V_{i-1}^j\right)\right). \end{aligned} \tag{1.15}$$

The Lax-Wendroff scheme is implemented in `LaxWendroff.m`, Appendix A.1.7.

1.4.3 Combining the two governing equations

Due to the fact that the Navier Stokes equation (1.11) is two-dimensional and the continuity equation (1.14) is one-dimensional, some manipulations to the variables need to be done. v and h are variables in radial and axial direction while H and V are constant throughout the cross section and has x and t as independent variables.

To make the jump between these two, the average velocity of the cross section is found for use in the continuity equation. This manipulation is quite drastic, and it is hard to predict the result and whether it will be acceptable or not.

The matlab function `average.m` in Appendix A.1.9 takes on the velocity profile and returns the average velocity:

$$
\begin{aligned}
V &= \sum_{i=1}^{NR} v_i \frac{A_i}{A} \\
&= \sum_{i=1}^{NR} v_i \frac{2\pi i \Delta r \Delta r}{\pi (NR\Delta r)^2} \qquad (1.16) \\
&= 2\sum_{i=1}^{NR} v_i \frac{i}{(NR)^2}.
\end{aligned}
$$

The matlab scripts `main.m` and `main2.m`, Appendices A.1.3 and A.1.4, solves a transient flow problem and a water hammer problem respectively. They both initialize the same velocity profile and pressure throughout the pipe. `main.m` predicts the flow and pressure response to a sinusoidal pressure at the pipe outlet. `main2.m` calculates the response to a water hammer. In the same fashon, initial values for pressure and flow are intialized, only with 0 velocity at the valve.

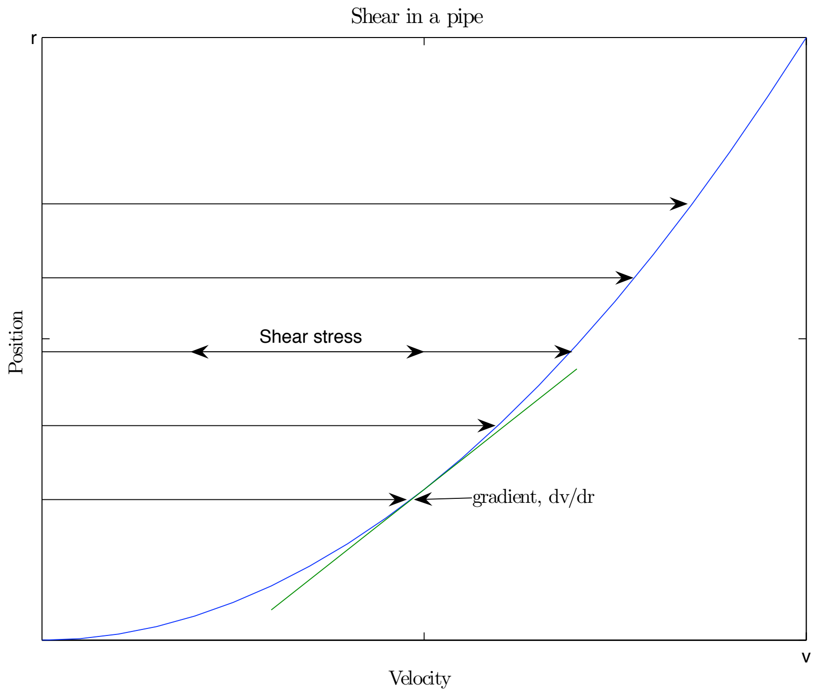


Figure 1.8: Shear stress for fluid flow in a pipe.

For each time step, the actual pressure drop is computed for all the plugs throughout the pipe using the matlab function `shear.m` given in Appendix A.1.8. A graphical illustration of how shear stresses result from the velocity profile is given in figure 1.8

The shear stress

$$\tau = -\rho\nu \frac{v(k+1) - v(k)}{\Delta r} \tag{1.17}$$

results in a force

$$f_k = -\rho\nu \frac{v(k+1) - v(k)}{\Delta r} \cdot 2\pi k \Delta r \Delta x \tag{1.18}$$

which acts on the respective interveaning space between the plugs and sum up to yeld a total force of

$$F = -\rho\nu \sum_{k=1}^{NR-1} \frac{v(k+1) - v(k)}{\Delta r} \cdot 2\pi k \Delta r \Delta x \cdot \frac{2\pi k \Delta r \Delta r}{\pi R^2} \tag{1.19}$$

hence a pressure drop given by

$$\Delta P = \frac{F}{\pi R^2} = -4\rho\nu \frac{\Delta r^2 \Delta x}{R^4} \sum_{k=1}^{NR-1} k^2 [v(k+1) - v(k)]. \tag{1.20}$$

1.4.4 Initial values and boundary conditions

I have assumed laminar flow and no-slip condition at the pipe walls for the flow model. Values for velocity are initialized for steady state conditions. For these assumptions, the Hagen-Poiseuille relation [5] yields:

$$v(r) = \left(\frac{\Delta p}{L}\right) \frac{1}{4\mu} (R^2 - r^2), \tag{1.21}$$

where Δp is the pressure drop over the length of the entire pipe L. The maximum velocity is found in the centerline at $r = 0$:

$$v_{max} = \left(\frac{\Delta p}{L}\right)\frac{1}{4\mu}R^2. \tag{1.22}$$

The average velocity is then foud to be

$$\begin{aligned} V_0 &= \frac{1}{\pi R^2}\int_0^R 2\pi r v(r)dr \\ &= \frac{1}{\pi R^2}\int_0^R 2\pi r\left(\frac{\Delta p}{L}\right)\frac{1}{4\mu}(R^2 - r^2)dr \\ &= \frac{\Delta p}{R^2 2\nu L}\left[\frac{R^2}{2}r^2 - \frac{1}{4}r^4\right]_0^R \\ &= \left(\frac{\Delta p}{L}\right)\frac{1}{8\mu}R^2 \\ &= \frac{1}{2}v_{max}. \end{aligned} \tag{1.23}$$

We can now initialize the velocity profiles by means of the average velocity V_0. This is done in the matlab script `initial.m` in Appendix A.1.2.

For both `main.m` and `main2.m` in Appendicies A.1.3 and A.1.4, the head at the reservoir is set to $70 meters$. Using the same matlab functions, `sd.m` and `LaxWendroff.m`, `main.m` and `main2.m` solves the pipe flow problem for variable head at the outlet and a water hammer respectively.

1.5 Results of Simulation

The calculation of the velocity profile is working quite well. Fluctuant pressures have been forced upon parabolic velocity profiles and the results show that the pressure gradient, which is the driving force of the fluid, affects the flow in the inner region of the pipe and the boundary layer close to the wall differently. The viscous effects are concentrated in a layer close to the pipe wall. At the same time, the inertial forces here are small so that the velocity

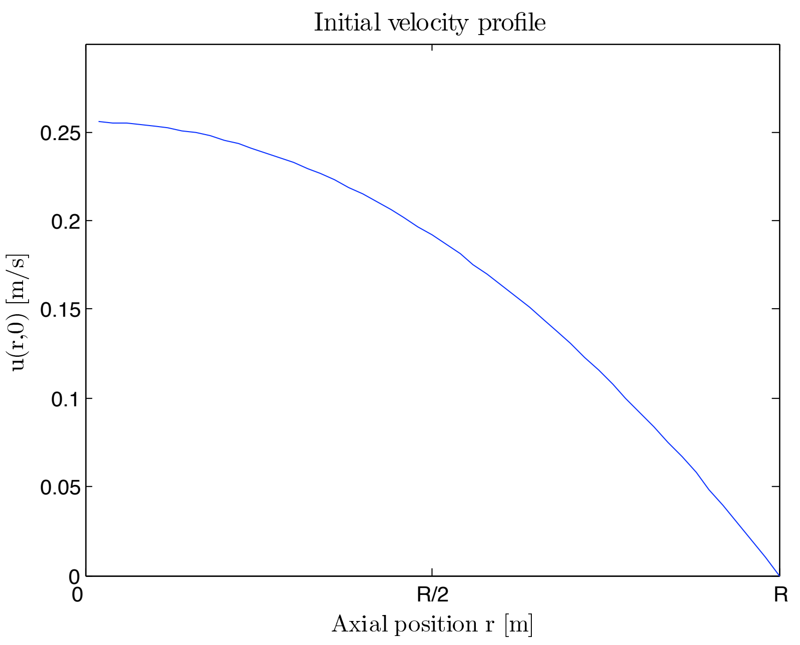
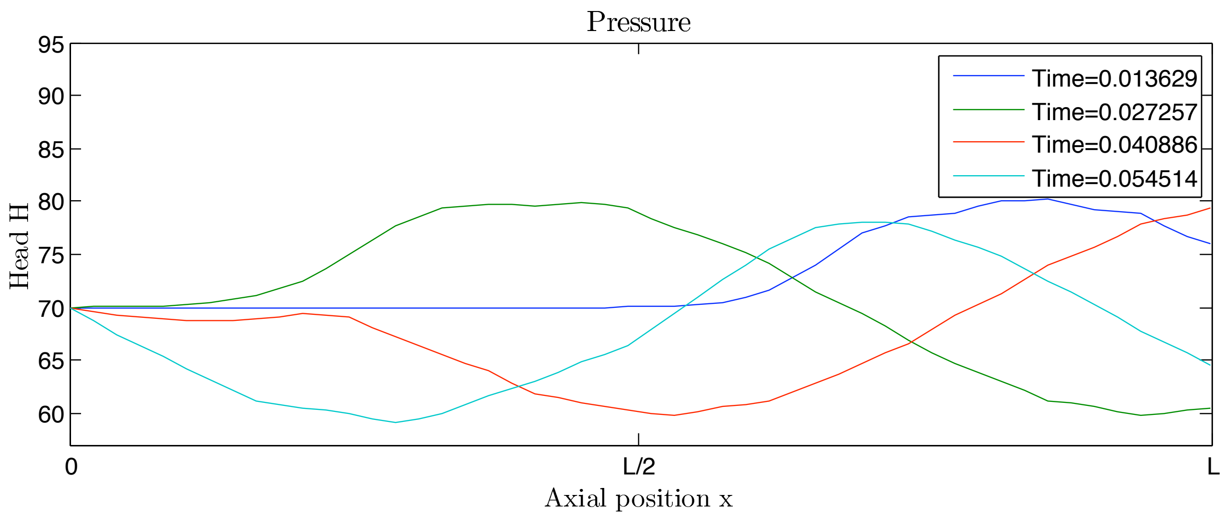


Figure 1.9: Initialized velocity profile for steady-state laminar flow.

is in phase with the pressure gradient. This means that the boundary layer close to the wall is first effected by the pressure gradient. For the inner layer which is not so much effected by wall shear, but mostly dominated by inertial forces, the acceleration of the flow is in phase with the pressure gradient.

The results tor the pipe flow where variable pressure at the pipe outlet was enforced, has been reasonable, see figures 1.10, 1.11, 1.12 and 1.13. The pressure waves propagate through the pipe with the right speed and amplitude. The acceleration of the main velocity changes in phase with the pressure gradient. However, it is hard to tell whether the results are reliable without any experimental verifications. The simulation run with $\nu = 0.00396$ shows a much higher damping and that the boundary layer for shear forces in the cross section of the pipe is much thicker than for a lower viscosity. This is in accordance with the theory[5].

Various solution methods have been tried to solve the equation of continuity (1.2) for the water hammer problem. Both the solution from applying the Lax Wendroff scheme and



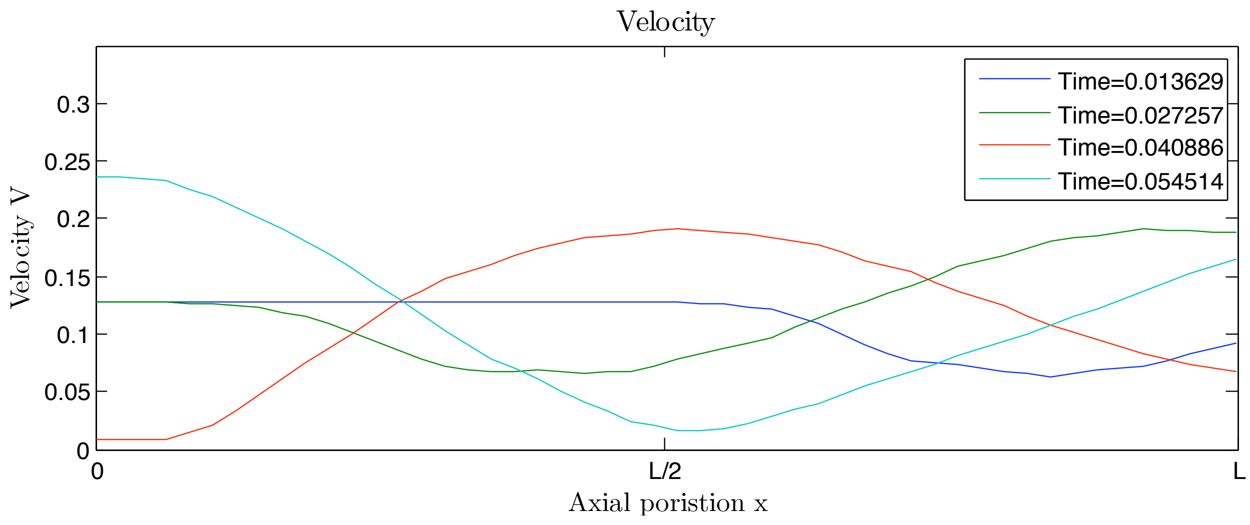


Figure 1.10: Velocity and pressure distribution throughout the pipe with variable pressure at pipe outlet.

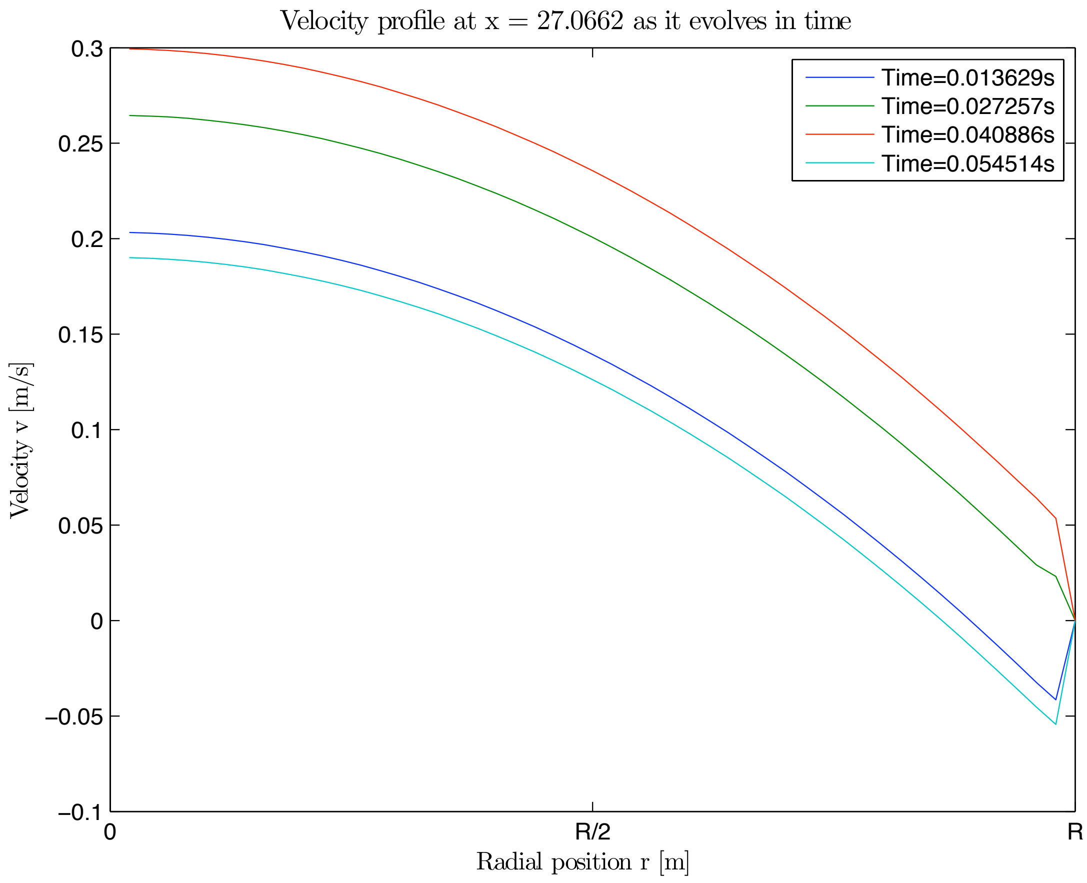
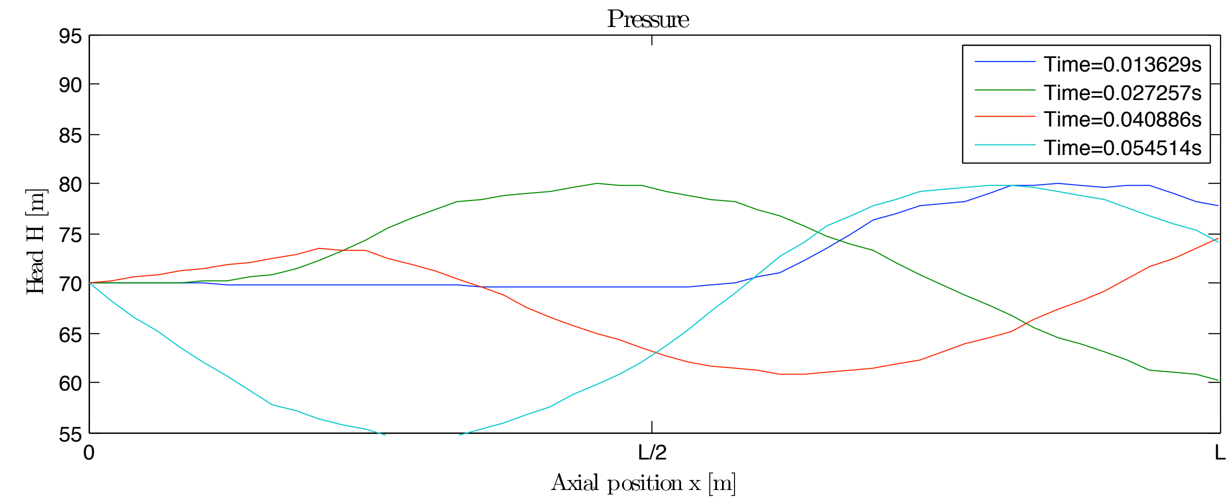


Figure 1.11: Velocity profile as it evolves due to variable pressure at pipe outlet.



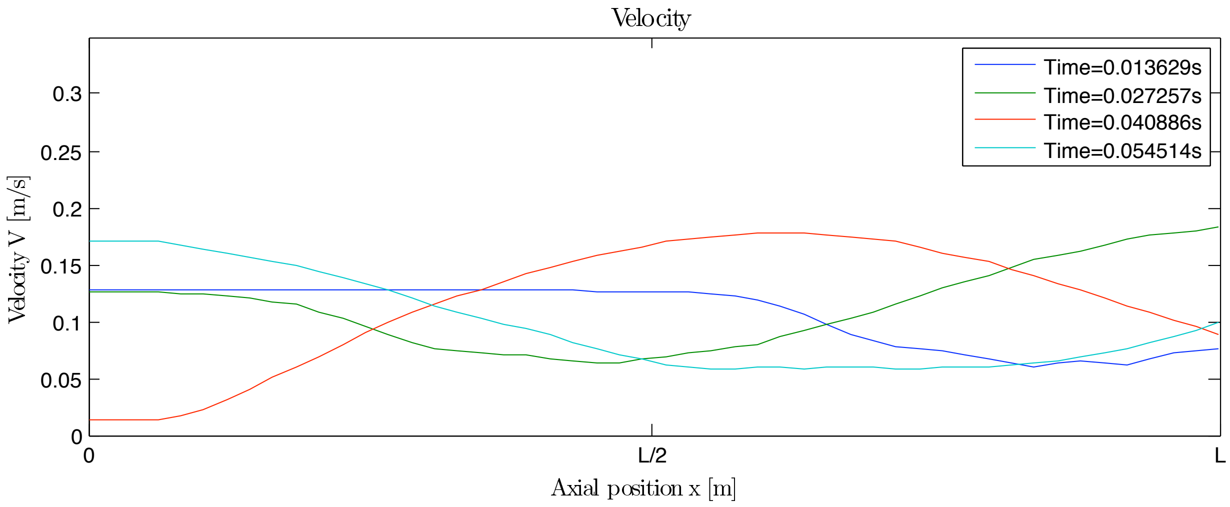


Figure 1.12: Velocity and pressure distribution throughout the pipe with variable pressure at pipe outlet. ν is here 0.00396.

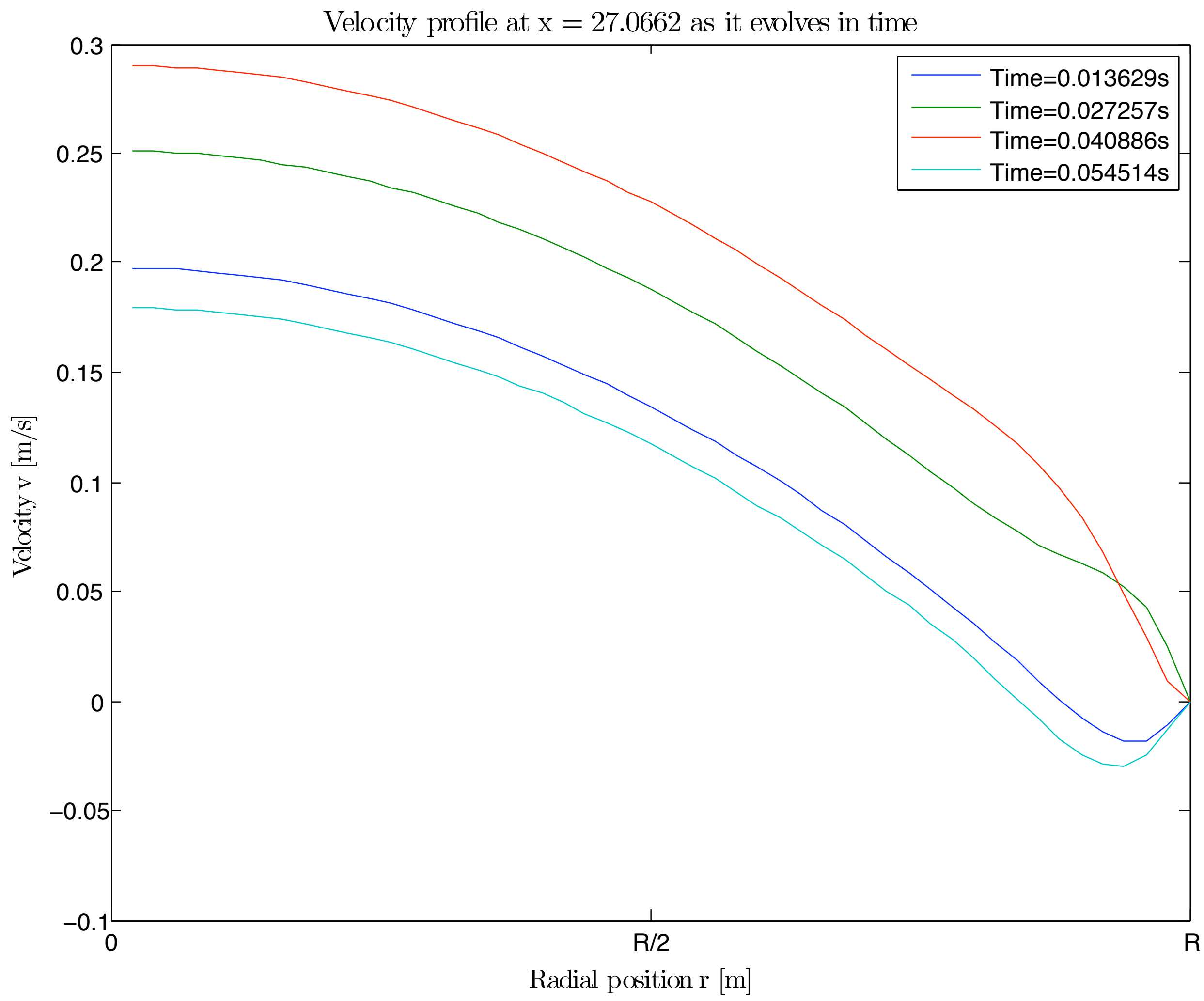


Figure 1.13: Velocity profile as it evolves due to variable pressure at pipe outlet. ν is here 0.00396.

a semi-descretizaion are given in figure 1.14. Numerous solution methods could be tried, but I do not think that the oscillations are a solution method problem as such, rather that the discretization of the physical system cause the oscillations. At the same time, it is hard to tell whether the equations solved, the Navier Stokes (1.11) and the continuity equation (1.2), would give a correct answer - regardless of solution method. The pressure process at the valve does show the right shape, but overestimates the pressure slightly. The oscillations botch up the results, so it is hard to tell whether the matlab scripts handle the frequency dependent friction correctly.

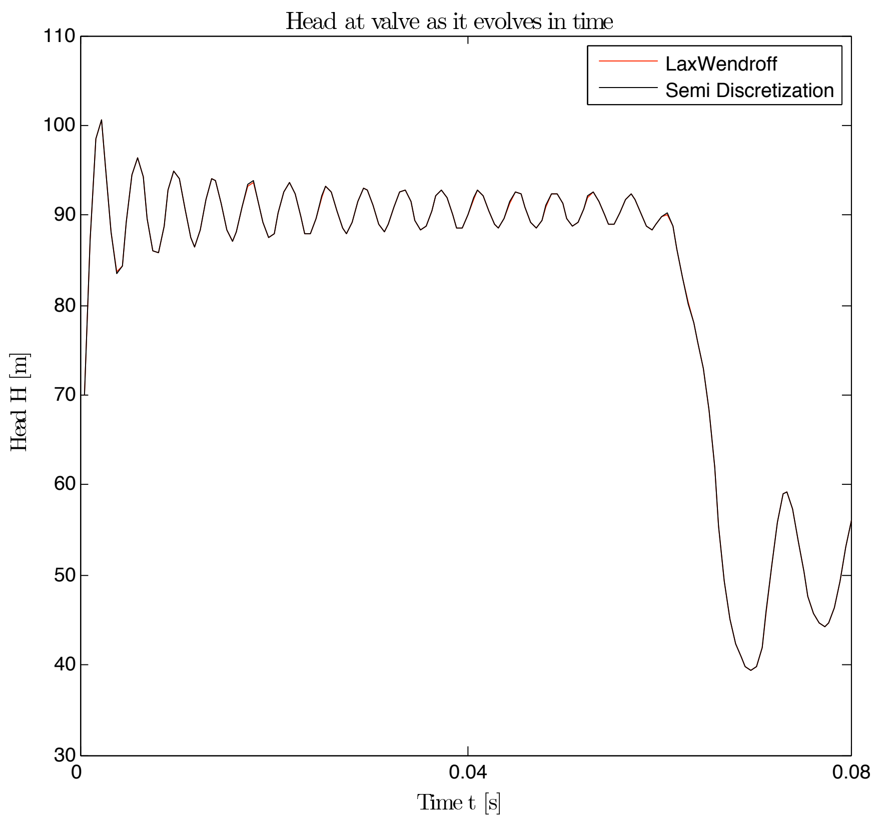
1.6 Conclusion

It is an exciting thought to be able to predict the velocity profile in the pipe and to see how it evolves in time for different flow conditions. The velocity profile gives information in regards to shear stresses and energy dissipation in the pipe. Besides, the radial variation of kinetic energy can be of importance in some applications. Braltand [1] showed that much kinetic energy was still in the pipe after immediate closure of a valve, such that not all of the fluid's kinetic energy was obsorbed.

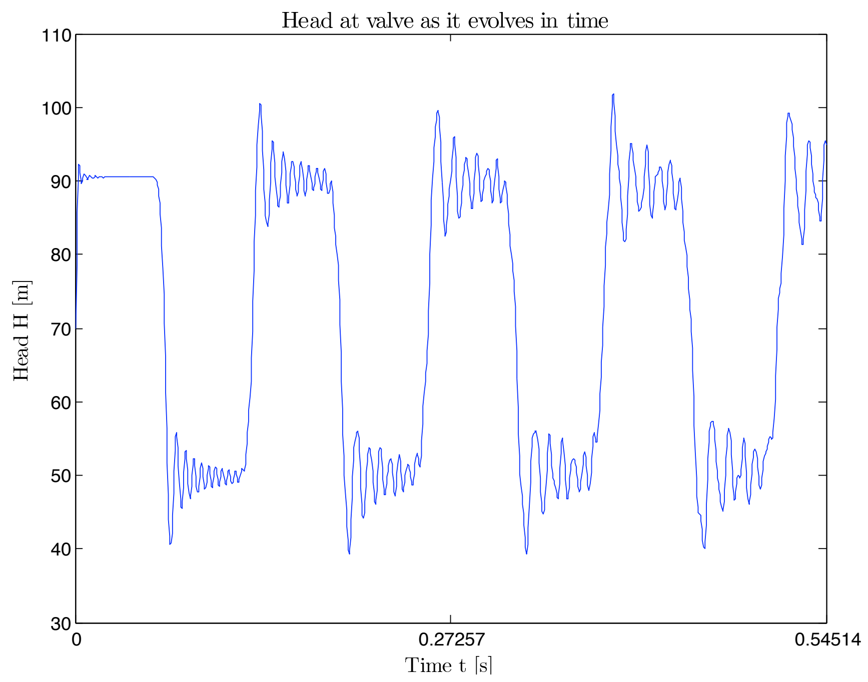
Jumping between the one dimensional continuity equation (1.2) and two dimensional Navier Stokes equation (1.11) seems not have been such a great idea. The one dimensional equations of momenutm (1.1) and continuity (1.2) both shear the same assumptions regarding equally distributed fluid velocity and pressure throughout the cross section of the pipe. And therefore the control volumes 1.1 satisfy the principlpes of momentum and continuity. Violations to this has been done in this work.

One other violation that can be mentioned is the pressure losses which are a result of shear stress. Pressure losses due to shear forces are, really, for specific cylinder shells, and should not be averaged out over the cross section.

The solution method to find the velocity profiles seems to be working very well. However, also here some violations are done. The velocity in the outer shell (closest to the pipe wall) is constantly held to zero. I am uncertain whether this is the actual case or not, but the



(a) Lax Wendroff scheme and semi-discretization



(b) Lax Wendroff

Figure 1.14: Pressure at valve after immediate closure.

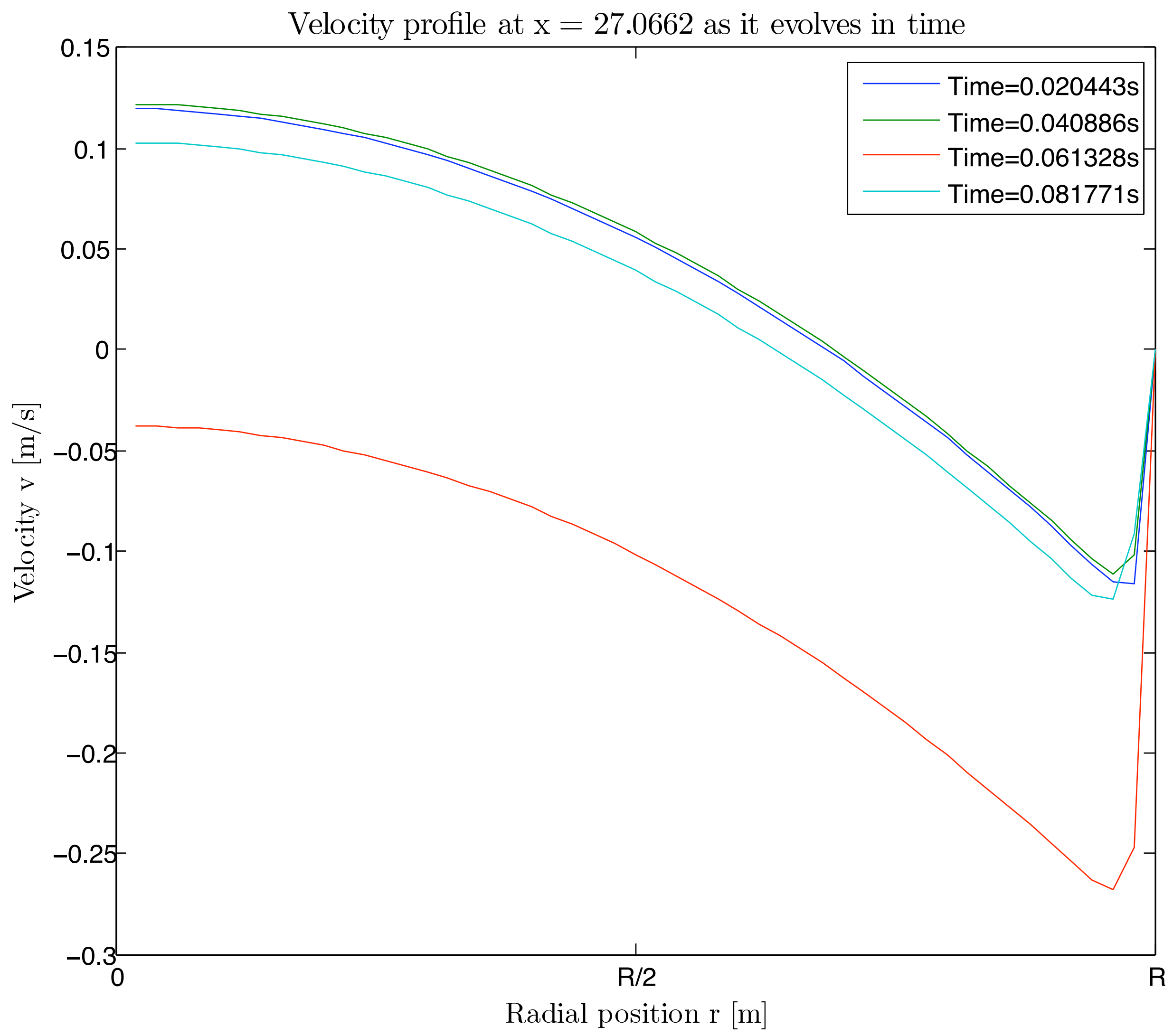
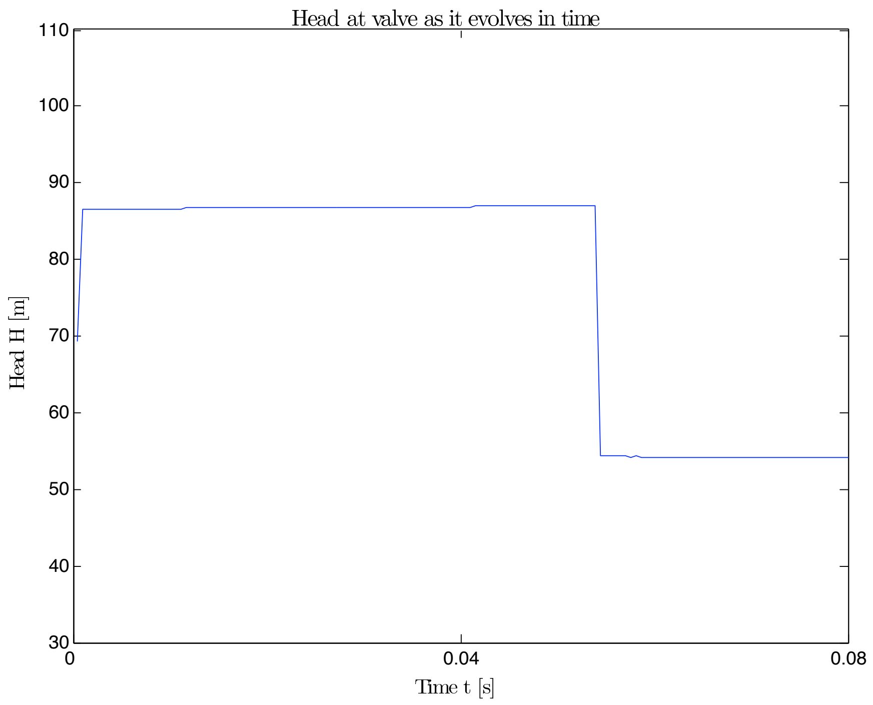
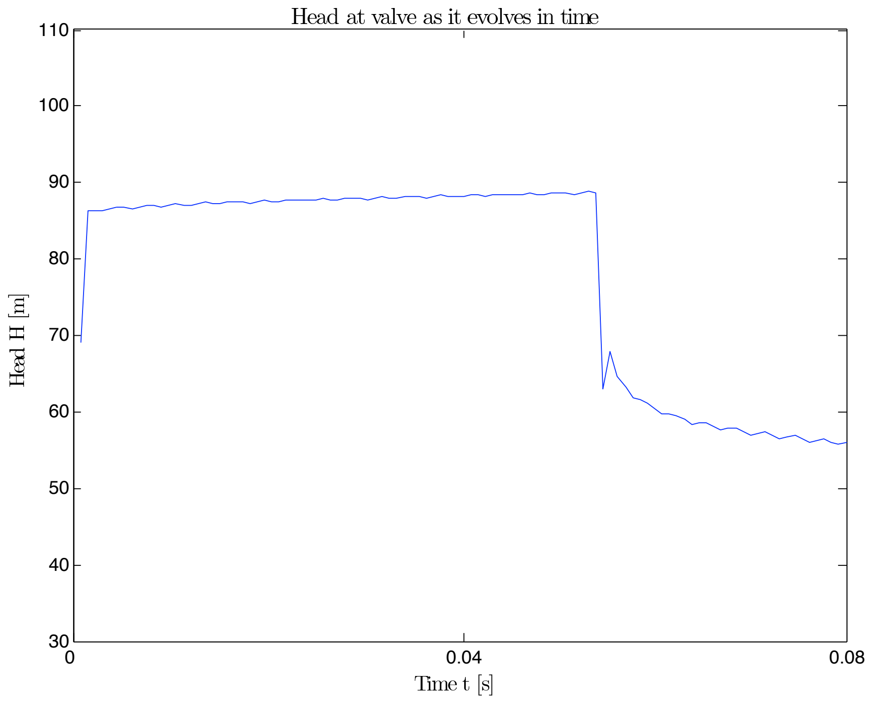


Figure 1.15: Velocity profile in a cross section of the pipe after immediate closure of valve.



(a) Steady state friction



(b) Zielke's friction term

Figure 1.16: Pressure at valve after immediate closure using the method of characteristics and (1.1) and (1.2) as governing equations.

non-slip condition is a normal assumption to make.

Chapter 2

Wind turbine hydrostatic gear system

2.1 Introduction

The main reason why it is interesting to introduce a hydrostatic transmission to wind turbines is the offshore installations which are being planned. First of all, a reduction of the weight at the nacelle is an absolute necessity and the hydrostatic pump and motor gives the possiblility to place the heavy power geneartor at ground/water level. Besides, there will no longer be use for the the mechanical gear which has been the weak link in chain so far.

The dynamic characteristics of a closed loop hydraulic system connecting a pump and a motor, is a complicated area of research. This report aims at describing and evaluating some of the dynamic properties of a system wich will be applied in a wind power turbine.

The dynamics of the hydrostatic transmission in a wind power tubine is related to the wind speed which can vary much. With variable wind speeds the rotor torque changes and therefor also the pump speed will vary. The speed of the hydraulic motor is controlled to be constant so that the generator runs at syncronous speed. The motor displacement is regulated by means of a PID[1]-controlling system which assures constant motor speed. The generator torque is adjusted according to the pressure drop over the hydraulic motor in order to take out as much electric power as possible.

[1]Proportional-Integral-Derivative control

I assume that the reader has some basic knowledge about control theory, and will therefore not go into detail on the PID-controlling system. The whole SIMULINK model is given in appendix B.

2.2 Hydraulic design

The hydraulic system which will be applied is sketched in figure 2.1. The hydraulic pump is a fixed displacement unit with displacement $D_p = 1.800679 \cdot 10^{-3} m^3/rad$. The pump is connected directly to the rotor rod. The motor has variable displacement rating from $D_{m-min} = 2.0 \cdot 10^{-5} m^3/rad$ to $D_{m-max} = 8.0 \cdot 10^{-5} m^3/rad$.

A boost pump is included in the hydraulic circuit to assure that the pressure from the pump, $P1$, does not drop below $80 \cdot 10^5 Pascal$, and that the pressure in the pipe from the motor back to the pump, $P2$, is always more than $40 \cdot 10^5 Pascal$.

To make sure that the pressure differences over the motor and pump does not exceed $310 \cdot 10^5 Pascal$, a pressure relief valve connects the hydraulic transmission lines. Based on

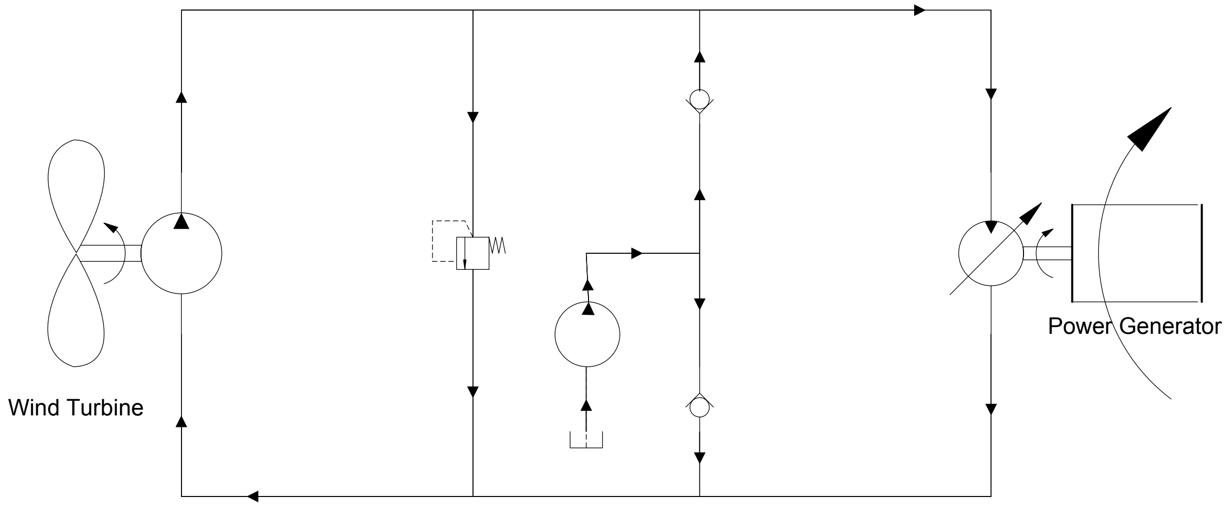


Figure 2.1: Design of the hydraulic gear to the wind turbine.

the pressures given above, the vind turbine should be able to operate at wind conditions resulting in rotor torques from about

$$Torque_{rt\_min} = \Delta P_{min} \cdot D_p = 7.2 \cdot 10^3 Nm \tag{2.1}$$

to

$$Torque_{rt\_max} = \Delta P_{max} \cdot D_p = 55.8 \cdot 10^3 Nm, \tag{2.2}$$

where $\Delta P_{min} = 40 \cdot 10^5 Pascal$ and $\Delta P_{max} = 310 \cdot 10^5 Pascal$.

The hydraulic motor is connected to the power generator which is runned at syncrounous speed around $157 rad/s$. Without taking the effects of friction and leakage into account, the pressure drops over the hydraulic motor and its displacement range, result in a motor power rating from

$$Power_{min} = \Delta P_{min} \cdot D_{m\_min} \cdot Omega_{m\_set} = 12.56 kW \tag{2.3}$$

to

$$Power_{max} = \Delta P_{max} \cdot D_{m\_max} \cdot Omega_{m\_set} = 389.36 kW. \tag{2.4}$$

2.3 SIMULINK model

The hydraulic system has been implemented in SIMULINK. The model is limited to rotor, hydraulic gear which includes the pump and motor and the torque of the power generator. All parameters and simulation block diagrams are given in Appendix B. Only in the extent necessary, block diagrams are included in this chapter.

The reader should not be confused by the use of $P1/Q1$ and $P1_f/Q2_f$ and also $P2/Q2$

and $P2_f/Q2_f$. The P1's and P2's and Q1's and Q2's are essensially the same[2].

Since the driving force to this system is the rotor torque, the authour has found it natural to base the analysis of this hydraulic system on sudden changes in rotor torque. The SIMULINK system takes on rotor torque plus some additional step torque and predicts the system's dynamic responce. A PID-controller regulates the motor displacement in order to maintain constant motor speed and the generator torque is varied according to the motor torque. This assures that as much electric power as possible is extracted from the system, without stalling the wind turbine that is.

A list of hydrualic system coefficients is given in table 2.1

Abbrevation	**Value**	**Description**	**Unit**
L	30.0	Pipe length	$[m]$
A	0.05	Cross section of the pipe	$[m^2]$
B	$1.5 \cdot 10^9$	Fluid bulk modulus	$[kg/ms^2]$
Cl_p	$7.5 \cdot 10^{-12}$	Pump leakage coefficient	$[(m^3/s)/(N/m^2)]$
Cf_p	0.01	Pump friction coefficient	$[(kg/s)]$
Cd_p	$7.5 \cdot 10^{-12}$	Pump discharge coefficient	$[(m^3/s)/(N/m^2)]$
Cl_m	$7.5 \cdot 10^{-12}$	Motor leakage coefficient	$[(m^3/s)/(N/m^2)]$
Cd_m	$7.5 \cdot 10^{-12}$	Motor discharge coefficient	$[(m^3/s)/(N/m^2)]$
C_m	0.005	Motor friction coefficient	$[(kg/s)]$

Table 2.1: Constants used in the SIMULINK model.

2.3.1 Governing dynamic properties of the transmission

I have summarized the equations for the pump parameters below. The same principles apply also for the motor, and the motor equations are therefore not stated in this chapter. However, they can easily be derived from the block diagrams in Appendix B. The driving force to this system is the rotor torque wich varies with the wind speed. When the rotor torque changes,

[2]The f-variables were ment for a pipe flow model that could be introduced with an embedded matlab function.

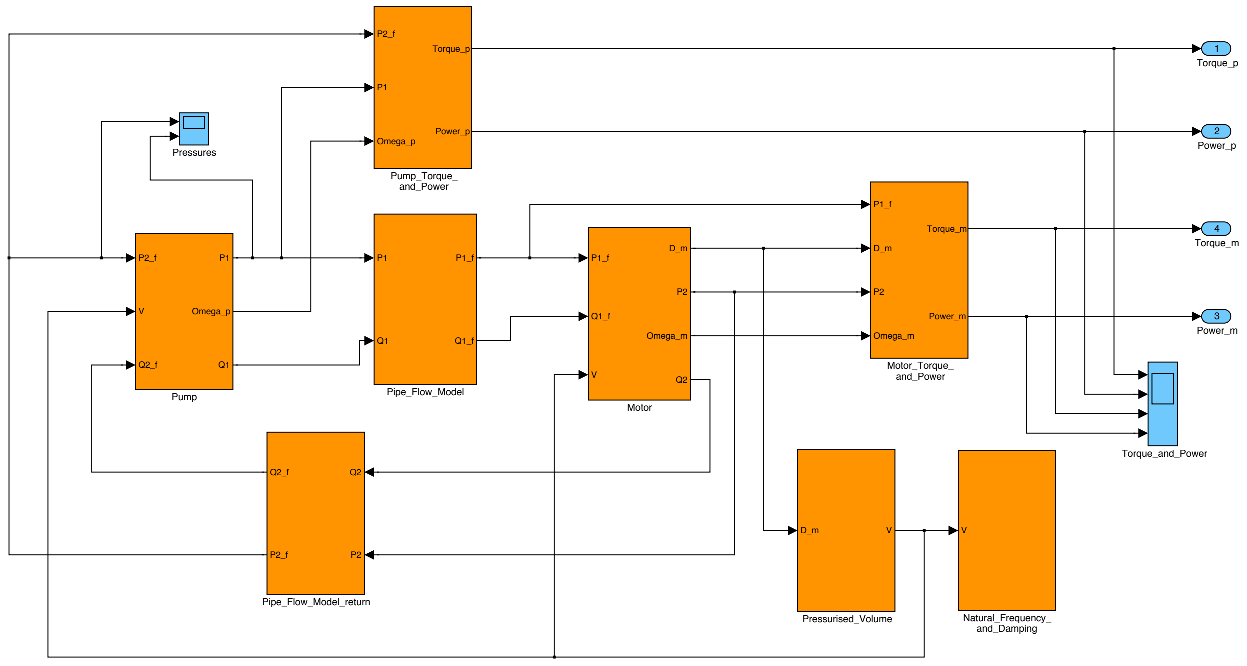


Figure 2.2: Overview of the hydraulic system simultaion block diagram.

the pump either accelerates or slows down according to

$$\frac{\delta(Omega_p)}{\delta t} = \frac{(P2 - P1) \cdot D_p - Omega_p \cdot Cf_p + Torque_{rt}}{Inertia_{rt}}. \tag{2.5}$$

Simulink integrates this term over the last time step and computes the new pump velocity. The integral is given a start value equal to $Omega_{p\_start}$ and maximum and minimum values of the integral are $Omega_{p\_max}$ and $Omega_{p\_min}$, respectively. $Omega_p \cdot Cf_p$ is a mechanical loss due to friction.

The change of pressure from the pump is given by

$$\frac{\delta(P1)}{\delta t} = \frac{B}{V}\left[Omega_p \cdot D_p - (P1 - P2) \cdot Cl_p - (P1 + P2) \cdot Cd_p - Q2\right], \tag{2.6}$$

where $Omega_p \cdot D_p$ is the flow from the pump, $(P1 - P2) \cdot Cl_p$ is a loss due to leakage and $(P1 + P2) \cdot Cd_p$ is the discharge. $Q2$ is the flow into the pump and B/V is the bulk modulus divided by the pressurised volume. In the same fashon as for the pump speed, also this integral is given a start value and maximum and minimum values. The maximum and minimum values are according to $P1_{max}$ and $P1_{min}$.

The flow from the pump is given by the equation

$$Q1 = Omega_p \cdot D_p - (P1 - P2) \cdot Cl_p, \tag{2.7}$$

which can be understood quite easily. $(P1-P2)\cdot Cl_p$ is a loss due to leakage. The increased pump flow needs to be compensated for by regulating the motor displacement and/or increasing the generator torque at the same time.

Pump torque and power are well known to engineers as

$$Torque_p = (P1 - P2) \cdot D_p \tag{2.8}$$

and

$$Power_p = Torque_p \cdot Omega_p. \tag{2.9}$$

2.3.2 PID-Controller

To assure that the generator runs at syncronous speed, the motor displacement is varied according to deviations from the set motor speed. The PID-controller in figure 2.3 takes on the difference between actual motor speed and set speed. Proportional, integral and derivative gains are found using the build-in SIMULINK parameter estimator "Ziegler-Nichols". It is worth mentioning that both integral and derivative terms are based on the most recent time-step only.

The saturation block keeps the output of the PID-controller within the values of D_{m-min} and D_{m-max}.

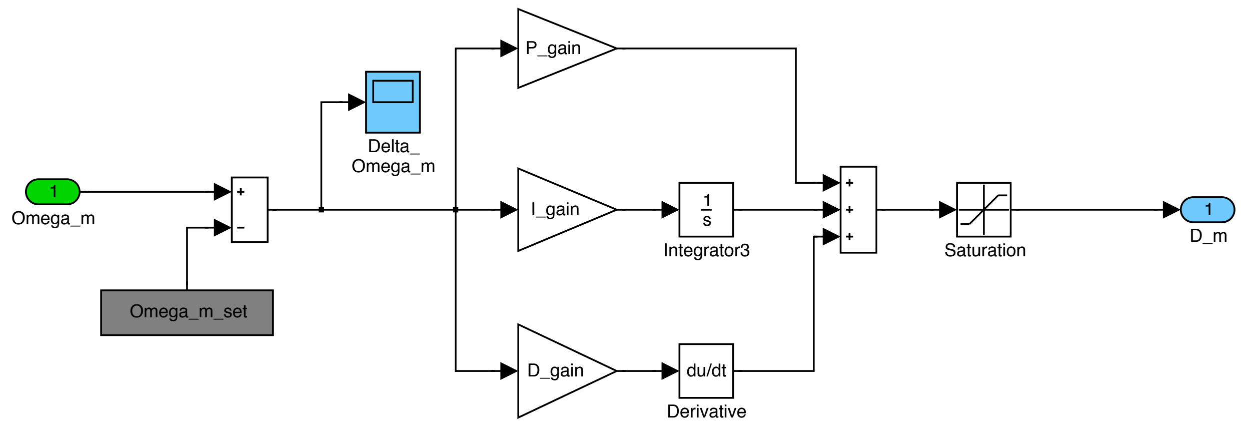


Figure 2.3: PID-control applied to regulate the motor displacement.

2.3.3 Generator torque

To assure maximum effect from the power generator, it's torque is adjusted based on the pressure drop over the hydraulic motor and the motor displacement, i.e. the motor torque. The relationship between the generator torque and motor torque can not be linear due to the losses from friction and leakage. The gains found are based on experimental using the SIMULINK model. As it turned out, the system stalled fast if the gains were set to high.

2.4 Results of Simulation

Different combinations of initial torque and step torque have been applied to see the systems reaction. For the simulations run, pipe pressures, pump and motor speed and motor displacement were initialized in the same manner. For the results given in this report, the pump starts off with an initial rotor torque of $20kNm$. After 500 seconds an additional step torque hits in and the pump is run by a total torque of $40kNm$. The adjustment of the generator torque seems to work quite well in this model, without stalling the pump.

The problem we are left with, is that the pump speed goes up to it's maximum value. There after the speed adjustment of the motor is fairly simple as the flow remains constant.

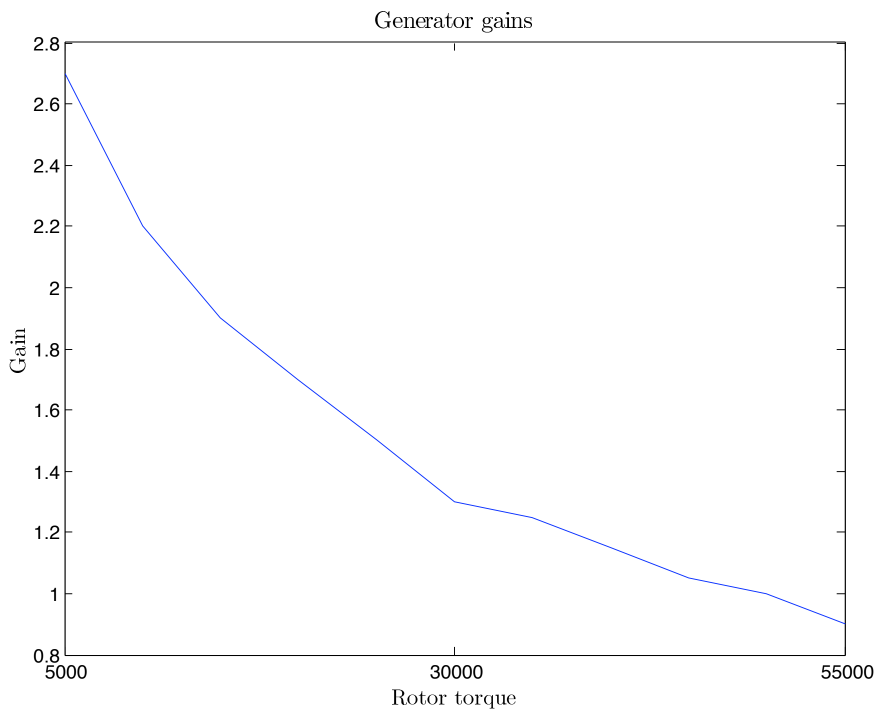


Figure 2.4: Generator gains.

2.5 Conclusion

The gains of the PID controllers are working quite well for this SIMULINK model. However, the friction, leakage and discharge coefficients are not measured and they are of importance for the dynamic response of the system, and therfore the PID gains need to be adjusted on site.

As it turns out, this model might be a bit to limited to actually simulate the characteristics of the rotor and pump. As it appears from the results, the pump will, under the given conditions, accelerate up till the maximum pump speed. However, the rotor torque is very much speed-dependent and the torque should decrease with increasing rotor speed. Hence, another which also could be added to the motor torque is the pump speed. If the pump speed goes up towards it's upper limit, the generator torque should be increased - and vice versa in the case where the pump speed approaches it's minimum.

To be better able to measure the strength of this PID-controller, a step input to the pump

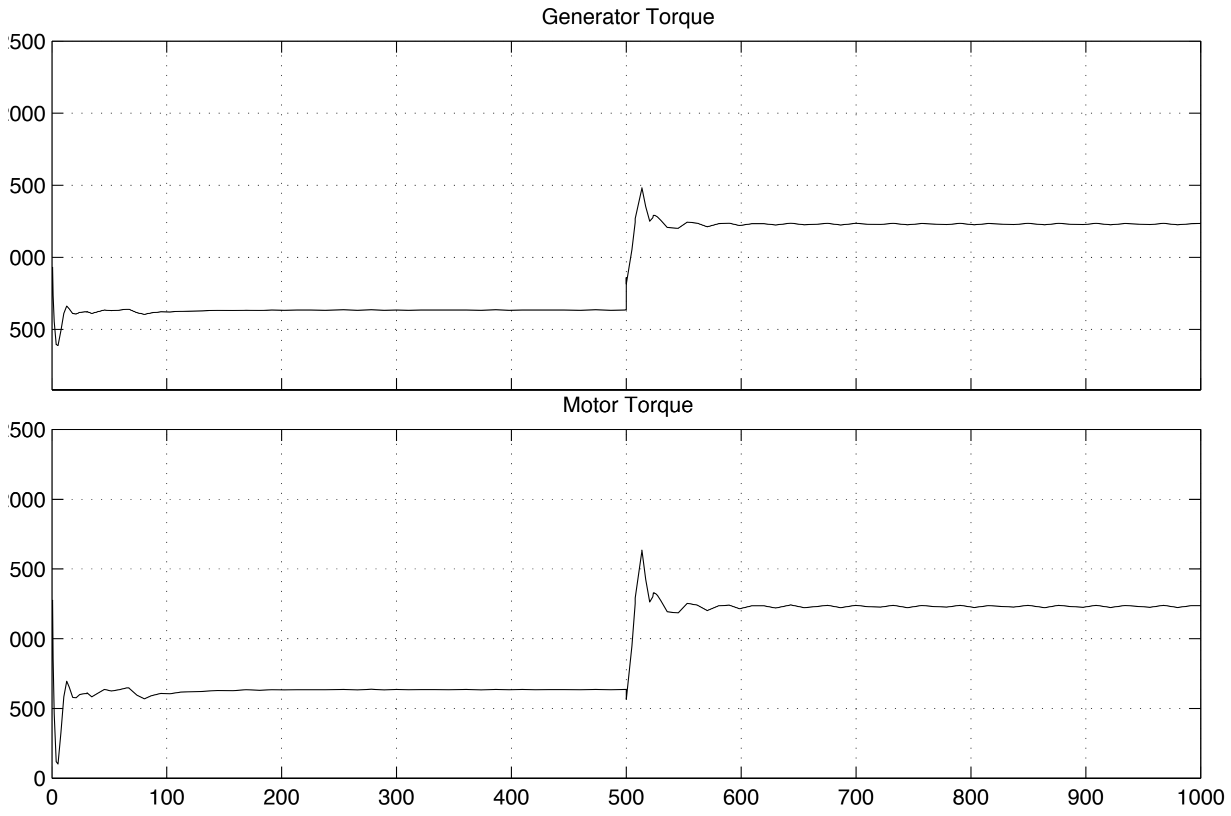


Figure 2.5: Generator and motor torque.

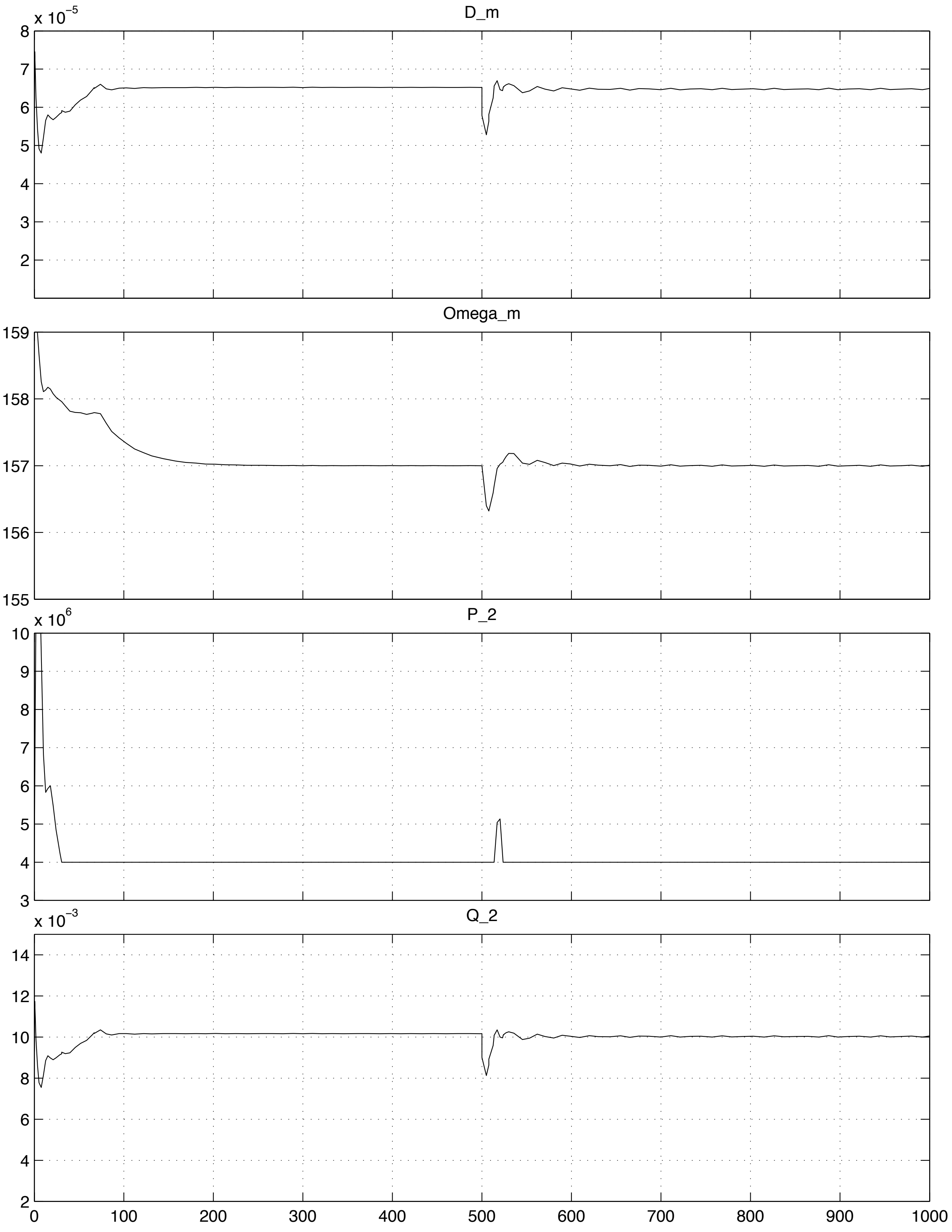


Figure 2.6: Motor variables.

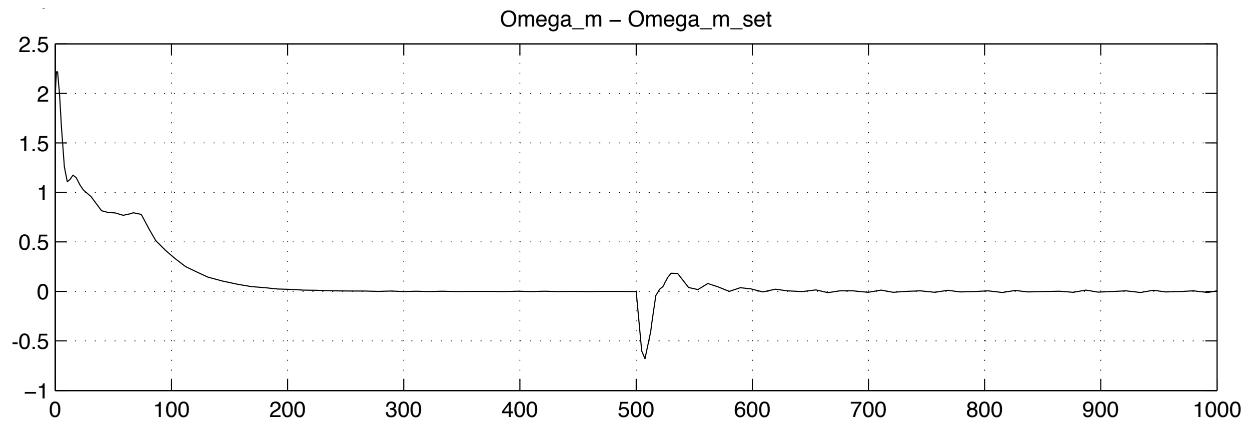


Figure 2.7: Deviation between actual motor speed and set speed.

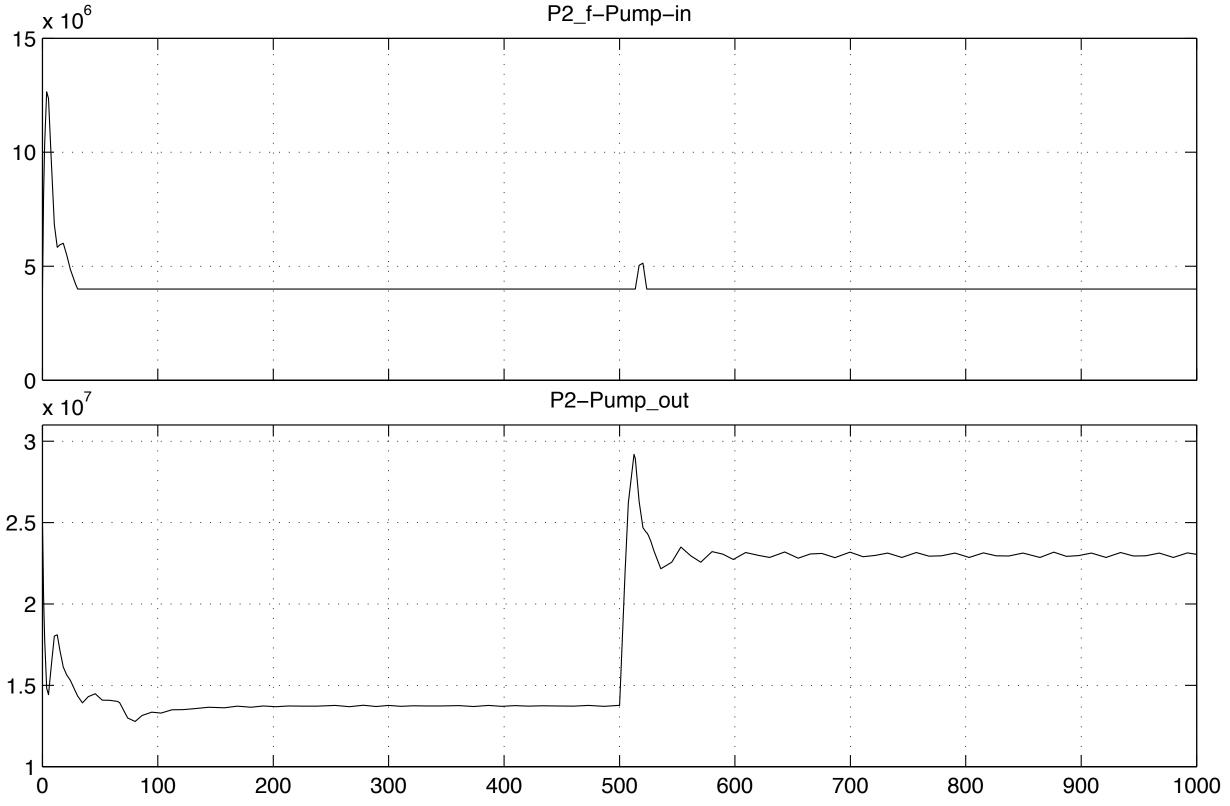


Figure 2.8: Pressures at pump inlet and pipe outlet.

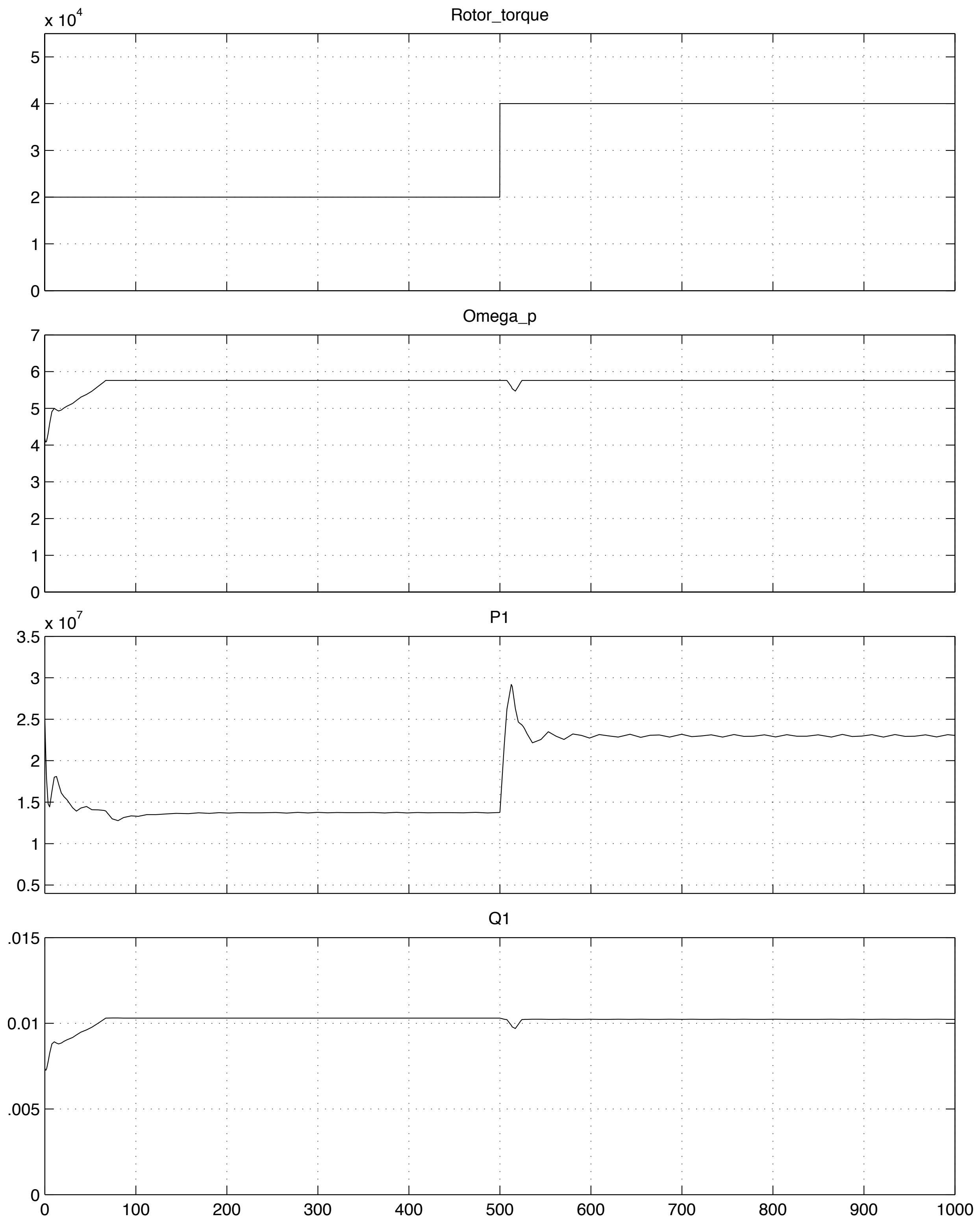


Figure 2.9: Pump parameters.

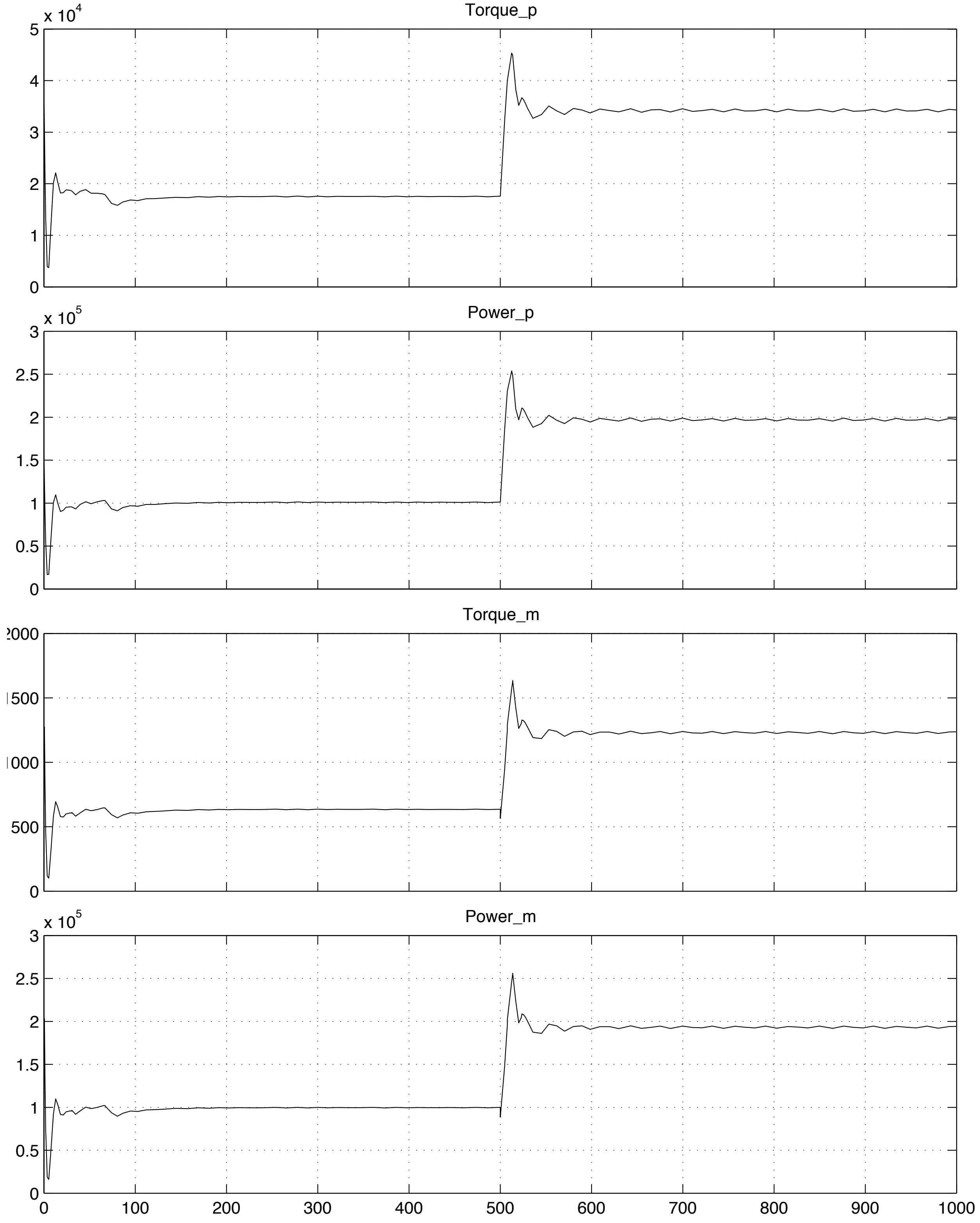


Figure 2.10: Pump and motor torque and power.

speed might be used in stead. Not too many re-arrangements in the SIMULINK model are necessary to do this.

The leakage terms used in the SIMULINK model are only dependent of pressure differences over the motor and pump. However, the viscosity of the hydraulic oil varies with the temperature and this might also play an important role. The same argumentation yields for the friction coefficients.

Bibliography

[1] O. Bratland. Frequency-dependent Friction and Radial Kinetic Energy Variation in Transient Pipe Flow. *5th International Conference on Pressure Surges, Hannover, F.R. Germany*, September 1986.

[2] L.F. Shampine. *Solving Hyperbolic PDEs in MATLAB*. Mathematics Department, Southern Methodist University, 2005.

[3] V. Streeter and E. B. Wylie. *Fluid Transients*. McGraw-Hill, 1978.

[4] F. M. White. *Fluid Mechanics*. McGraw-Hill, 2003.

[5] W. Zielke. *Frequency Dependent Friction in Transient Pipe Flow*. PhD thesis, The University of Michigan, University of Michigan, Department of Civil Engineering, December 1966.

List of Figures

Appendix A

Flow Model

A.1 Matlab Code

A.1.1 parameters.m

```
rho = 950;
g = 9.81;
nu = 0.00003967;
a = 1324;
L = 36.08832;
NX = 50;
dt = (L/(NX*a));
T = 0.08385;
NT = round(T/dt);
NR = 50;
dx=L/NX;
D = 0.254;
R = D/2;
A = pi*R^2;
dr = R/NR;
t = linspace(0,T,NT);
r = dr:dr:R;
x = linspace(0,L,NX);
V0 = 0.128;
H0 = 70;
HL = H0-(32*nu/(D^2*g))*V0*L;
```

A.1.2 initial.m

```
function vstart=initialflow

```

```
% The function "initialflow" computes the initial Hagen−Poiseuille flow
% profile (from r=0 to R) in the pipe assuming a no−slip condition at the
% wall and axi−symmetric flow steady state flow.

parameters;

for k = 1:NR
    vstart(k) = (−2*V0/R^2)*(r(k))^2+2*V0; % +0.1*rand(1);
end
vstart(NR)=0;

% −−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−

% figure1 = figure('PaperSize',[20.98 29.68]);
% axes('Parent',figure1,'XTickLabel',{'0   ','R/2','R'},...
%     'XTick',[0 0.0635 0.127]);
% box('on');
% hold('all');
% plot(r,vstart)
% title('Initial velocity profile','Interpreter','latex','FontSize',12);
% xlabel('Axial position r [m]','Interpreter','latex','FontSize',11);
% ylabel('u(r,0) [m/s]','Interpreter','latex','FontSize',11);
```

A.1.3 main.m

```
function H=main

% "main.m" simulates a pressure flow process where the pressure at the
% outlet varies with time. H, V and v are initialized for steady state
% conditions and the function "sd.m" is used to find velocity profiles
% according to the pressure distribution found by means of "ssd.m". The
% result of "average.m" is V. To save storage and computational time, the
% variable v should be cleared for long simulations. Figure 2 must in so
% case be commented out.

parameters;

H(1:NX,1) = linspace(H0,HL,NX)
V(1:NX,1) = 0.1280;
% V(NX,1) = 0;
for i=1:NX
    v(i,:,1)=initialflow;
end


for j=2:NT

    j
    % H(:,j) = ssd(H(:,j-1)',V(:,j-1)');
    H(:,j) = LaxWendroff(H(:,j-1)',V(:,j-1)');
    H(NX,j)=70+10*sin(0.09*j);
    H(1,j)=H0;
    v(1,:,j) = sd(H(2,j)-H(1,j),v(1,:,j-1));
    V(1,j) = average(v(1,:,j));

    for i=2:NX-1
        H(i,j)=H(i,j)+shear(v(i,:,j));
        v(i,:,j) = sd((H(i+1,j)-H(i-1,j))/2,v(i,:,j-1));
        V(i,j) = average(v(i,:,j));
    end
    v(NX,:,j) = sd((H(NX,j)-H(NX-1,j)),v(NX,:,j-1));
    V(NX,j)=average(v(NX,:,j));
    % clear v(:,:,j-1)
end

% ---------------------------------------------------------------------------

figure1 = figure('PaperType','a4letter','PaperSize',[20.98 29.68]);
subplot1 = subplot(2,1,1,'Parent',figure1,'XTickLabel',{'0','L/2','L'},...
    'XTick',[0 18 36.1]);
box('on')
hold('all')
plot1 = plot(x,H(:,round(NT/4)),x,H(:,round(NT/2)),...
x,H(:,round(3*NT/4)),x,H(:,round(NT)));
title('Pressure','Interpreter','latex','FontSize',12);
xlabel('Axial position x [m]','Interpreter','latex',...
'HorizontalAlignment','center','FontSize',11);
ylabel('Head H [m]','Interpreter','latex','HorizontalAlignment',...
    'center','FontSize',11);
set(plot1(1),'DisplayName',['Time=',num2str(dt*NT/4),'s']);
set(plot1(2),'DisplayName',['Time=',num2str(dt*NT/2),'s']);
set(plot1(3),'DisplayName',['Time=',num2str(dt*3*NT/4),'s']);
set(plot1(4),'DisplayName',['Time=',num2str(dt*NT),'s']);
legend(subplot1,'show');

subplot2 = subplot(2,1,2,'Parent',figure1,'XTickLabel',{'0','L/2','L'},...
    'XTick',[0 18 36.1]);
box('on')
hold('all')
plot2 = plot(x,V(:,round(NT/4)),x,V(:,round(NT/2)),...
    x,V(:,round(3*NT/4)),x,V(:,round(NT)));
set(plot2(1),'DisplayName',['Time=',num2str(dt*NT/4),'s']);
set(plot2(2),'DisplayName',['Time=',num2str(dt*NT/2),'s']);
set(plot2(3),'DisplayName',['Time=',num2str(dt*3*NT/4),'s']);
set(plot2(4),'DisplayName',['Time=',num2str(dt*NT),'s']);
title('Velocity','Interpreter','latex','FontSize',12);
xlabel('Axial position x [m]','Interpreter','latex','FontSize',11);
ylabel('Velocity V [m/s]','Interpreter','latex','HorizontalAlignment',...
    'center','FontSize',11);
```

```
legend(subplot2,'show');

figure2 = figure('PaperType','a4letter','PaperSize',[20.98 29.68]);
axes3 = axes('Parent',figure2,'XTickLabel',{'0','R/2','R'},...
    'XTick',[0 0.0635 0.127]);
box('on');
hold('all');
plot3 = plot(r,v(round(3*NX/4),:,round(NT/4)),...
    r,v(round(3*NX/4),:,round(NT/2)),...
    r,v(round(3*NX/4),:,round(3*NT/4)),...
    r,v(round(3*NX/4),:,round(NT)));
title(['Velocity profile at x = ',num2str(3*L/4),...
    ' as it evolves in time'],'Interpreter','latex',...
    'FontSize',12);
xlabel('Radial position r [m]','Interpreter','latex',...
    'HorizontalAlignment','center','FontSize',11);
ylabel('Velocity v [m/s]','Interpreter','latex',...
    'HorizontalAlignment','center','FontSize',11);
set(plot3(1),'DisplayName',['Time=',num2str(dt*NT/4),'s']);
set(plot3(2),'DisplayName',['Time=',num2str(dt*NT/2),'s']);
set(plot3(3),'DisplayName',['Time=',num2str(dt*3*NT/4),'s']);
set(plot3(4),'DisplayName',['Time=',num2str(dt*NT),'s']);
legend(axes3,'show');
```

A.1.4 main2.m

```
function H = main2

% "main2.m" simulates a pressure flow process where a valve at the pipe
% outlet is shout immediately. H, V and v are initialized for steady state
% conditions and the function "sd.m" is used to find velocity profiles
% according to the pressure distribution found by means of "ssd.m". The
% result of "average.m" is V. To save storage and computational time, the
% variable v should be cleared for long simulations. Figure 2 must in so
% case be commented out.

parameters;

H(1:NX,1) = linspace(H0,HL,NX)
V(1:NX,1) = 0.1280;
V(NX,1) = 0;
for i=1:NX-1
    v(i,:,1)=initialflow;
end
v(NX,:,1)=0*initialflow;

for j=2:NT

```

```
    j
    % H(:,j) = ssd(H(:,j-1)',V(:,j-1)');
    H(:,j) = LaxWendroff(H(:,j-1)',V(:,j-1)');
    H(1,j)=H0;
    v(1,:,j) = sd(H(2,j)-H(1,j),v(1,:,j-1));
    V(1,j) = average(v(1,:,j));
    for i =1:NX
        H(i,j)=H(i,j)+shear(v(i,:,j));
    end

    for i=2:NX-1
        % v(i,:,j) = sd((H(i+1,j)-H(i-1,j))/2,v(i,:,j-1));
        v(i,:,j) = sd((H(i+1,j)-H(i-1,j))/2,v(i,:,j-1));

        V(i,j) = average(v(i,:,j));
    end
    v(NX,:,j) = 0*initialflow;
    V(NX,j)=0;
    % H(NX,j) = ssd(H(:,j-1),V(:,j-1))
end

% -------------------------------------------------------------------------

figure1 = figure('PaperType','a4letter','PaperSize',[20.98 29.68]);
subplot1 = subplot(2,1,1,'Parent',figure1,'XTickLabel',{'0','L/2','L'},...
    'XTick',[0 18 36.1]);
box('on')
hold('all')
plot1 = plot(x,H(:,round(NT/4)),x,H(:,round(NT/2)),...
    x,H(:,round(3*NT/4)),x,H(:,round(NT)));
title('Pressure','Interpreter','latex','FontSize',12);
xlabel('Axial position x [m]','Interpreter','latex',...
    'HorizontalAlignment','center','FontSize',11);
ylabel('Head H [m]','Interpreter','latex',...
    'HorizontalAlignment','center','FontSize',11);
set(plot1(1),'DisplayName',['Time=',num2str(dt*NT/4),'s']);
set(plot1(2),'DisplayName',['Time=',num2str(dt*NT/2),'s']);
set(plot1(3),'DisplayName',['Time=',num2str(dt*3*NT/4),'s']);
set(plot1(4),'DisplayName',['Time=',num2str(dt*NT),'s']);
legend(subplot1,'show');

subplot2 = subplot(2,1,2,'Parent',figure1,'XTickLabel',{'0','L/2','L'},...
    'XTick',[0 18 36.1]);
box('on')
hold('all')
plot2 = plot(x,V(:,round(NT/4)),x,V(:,round(NT/2)),...
    x,V(:,round(3*NT/4)),x,V(:,round(NT)));
set(plot2(1),'DisplayName',['Time=',num2str(dt*NT/4),'s']);
set(plot2(2),'DisplayName',['Time=',num2str(dt*NT/2),'s']);
set(plot2(3),'DisplayName',['Time=',num2str(dt*3*NT/4),'s']);
set(plot2(4),'DisplayName',['Time=',num2str(dt*NT),'s']);
```

```
title('Velocity','Interpreter','latex','FontSize',12);
xlabel('Axial position x [m]','Interpreter','latex','FontSize',11);
ylabel('Velocity V [m/s]','Interpreter','latex','HorizontalAlignment',...
    'center','FontSize',11);
legend(subplot2,'show');

figure2 = figure('PaperType','a4letter','PaperSize',[20.98 29.68]);
axes3 = axes('Parent',figure2,'XTickLabel',{'0','R/2','R'},...
    'XTick',[0 0.0635 0.127]);
box('on');
hold('all');
plot3 = plot(r,v(round(3*NX/4),:,round(NT/4)),...
    r,v(round(3*NX/4),:,round(NT/2)),...
    r,v(round(3*NX/4),:,round(3*NT/4)),...
    r,v(round(3*NX/4),:,round(NT)));
title(['Velocity profile at x = ',num2str(3*L/4),...
    ' as it evolves in time'],'Interpreter','latex','FontSize',12);
xlabel('Radial position r [m]','Interpreter','latex',...
    'HorizontalAlignment','center','FontSize',11);
ylabel('Velocity v [m/s]','Interpreter','latex',...
    'HorizontalAlignment','center','FontSize',11);
set(plot3(1),'DisplayName',['Time=',num2str(dt*NT/4),'s']);
set(plot3(2),'DisplayName',['Time=',num2str(dt*NT/2),'s']);
set(plot3(3),'DisplayName',['Time=',num2str(dt*3*NT/4),'s']);
set(plot3(4),'DisplayName',['Time=',num2str(dt*NT),'s']);
legend(axes3,'show');

figure3 = figure('PaperType','a4letter','PaperSize',[20.98 29.68]);
axes4 = axes('Parent',figure3,'XTickLabel',...
    {num2str(0*dt*NT),num2str(dt*NT/2),num2str(dt*NT)},...
    'XTick',[0 NT/2 NT]);
box('on');
hold('all');
plot4 = plot(1:NT,H(NX,:));
title('Head at valve as it evolves in time','Interpreter','latex',...
    'FontSize',12);
xlabel('Time t [s]','Interpreter','latex','HorizontalAlignment','center',...
    'FontSize',11);
ylabel('Head H [m]','Interpreter','latex','HorizontalAlignment','center',...
    'FontSize',11);
```

A.1.5 sd.m

```
function Vdt=sd(H,u)

% The function "sd(H,u)" takes on a pressure drop and a velocity
% profile. It solves the equation of motion for parallel
% axisymmetric flow for an incompressible fluidfor this velocity
% profile and the associated pressure drop axisymmetric flow for
% an incompressible fluid. The convective term is found using
% "conv(u)" and the laplacian is found using "lap(u)". The dynamic
% system is then solved using ode45 from time 0 to time dt.

% ---------------------------------------------------------------

global dr
global nu
global rho
global NX
global dx
global px

parameters;

px = H*rho*g;

[t,Vt] = ode45(@mfun,[0 dt],u);

n = length(Vt(:,1));
Vt(:,end)=0;
Vdt = Vt(end,:);
Vdt(end)=0;

% ---------------------------------------------------------------

 figure1 = figure('PaperType','a4letter','PaperSize',[20.98 29.68]);

 subplot1 = subplot(2,1,1,'Parent',figure1,'XTickLabel',...
     {'0   ','R/2','R'},'XTick',[0 0.0635 0.127]);
 view([-160.5 42]);
 grid('on');
 hold('all');

 surf(r,t,Vt,'Parent',subplot1,'EdgeColor','none');

 title('Velocity profile as it evolves in time','Interpreter','latex',...
     'FontSize',12);
 xlabel('Radial position r [m]','Interpreter','latex',...
     'HorizontalAlignment','right','FontSize',11);
 ylabel('Time t [s]','Interpreter','latex',...
     'HorizontalAlignment','right','FontSize',11);
 zlabel('Velocity v [m/s]','Interpreter','latex',...
     'HorizontalAlignment','right','FontSize',11);

 subplot2 = subplot(2,1,2,'Parent',figure1,'XTickLabel',...
     {'0   ','R/2','R'},'XTick',[0 0.0635 0.127]);
 box('on');
 hold('all');

 plot1 = plot(r,u,r,Vdt);

 set(plot1(1),'DisplayName',['Time=',num2str(dt*0),'s']);
 set(plot1(2),'DisplayName',['Time=',num2str(dt*1),'s']);

 title('Velocity profile','Interpreter','latex','FontSize',11);
 xlabel('Radial position r [m]','Interpreter','latex','FontSize',11);
 ylabel('Velocity v [m/s]');
 legend(subplot2,'show');

% ----------------------------------------------------------------------

function fv = mfun(t,u)
global nu
global rho
global NX
global px

fv = u*0;
fv = -px/rho+nu*conv(u)+nu*lap(u);

% ----------------------------------------------------------------------

function Cv = conv(u);
global dr
n=length(u);

Cv = u*0;
for i = 2:n-1
    Cv(i) = (u(i+1)-u(i-1))/(2*dr*(i-1)*dr);
end
Cv(n) = (-u(n-1))/(2*dr*(n-1)*dr);

% ----------------------------------------------------------------------

function Av = lap(u)
global dr
n = length(u);


Av = u*0;
Av(1) = 2*(u(2)-u(1))/dr^2;
for i = 2:n-1
    Av(i) = (u(i-1)-2*u(i)+u(i+1))/dr^2;
end
Av(n) = 2*(-u(n-1))/dr^2;
```

A.1.6 ssd.m

```
function hdt=sdd(h,v)

% The function "sdd(h,v)" takes on the pressure and flow distribution
% in the axial direction of the pipe and solves the equation of
% continuity by means of ode24. The spacial derivatives are found
% using the function "der(u)".
global NX
global dx
global u
global dvdx
global x
u=v;
parameters;
dvdx=(a^2/g)*(deru(u));


[t,ht] = ode45(@mfun,[0 dt],h);

hdt = (ht(end,:))';
hdt(1)=H0;
% hdt(end)=h(end);

function fv = mfun(t,h)
parameters;
global u
global dvdx
fv = h*0;
fv= -uderh(h,u)-dvdx';

% ------------------------------------------------------------------------

function Cv = deru(u);
parameters;
n=length(u);

Cv = u*0;
Cv(1) = (u(2)-u(1))/dx;
for i = 2:n-1
    Cv(i) = (u(i+1)-u(i-1))/(2*dx);
end
Cv(n) = (u(n)-u(n-1))/(dx);

% ------------------------------------------------------------------------

function vh = uderh(h,u);
parameters;
n=length(u);

vh = h*0;
vh(1) = u(1)*(h(2)-h(1))/(dx);
for i = 2:n-1
```

```
    vh(i) = u(i)*(h(i+1)-h(i-1))/(2*dx);
end
vh(n) = u(n)*(h(n)-h(n-1))/(dx);

% ------------------------------------------------------------------
```

A.1.7 LaxWendroff.m

```
function Hdt = LaxWendroff(h,u)

parameters;

Hdt = h-derh(h,u)+flux(h,u)+(a^2/g)*(-derua(u)+derub(u));




% ------------------------------------------------------------------

function dua = derua(u);
parameters;
n=length(u);

dua = u*0;
dua(1) = (dt/(dx))*(u(2)-u(1));
for i = 2:n-1
    dua(i) = (dt/(2*dx))*(u(i+1)-u(i-1));
end
dua(n) = (dt/(dx))*(u(n)-u(n-1));

% ------------------------------------------------------------------

function dub = derub(u);
parameters;
n=length(u);

dub = u*0;
dub(1) = (dt^2/(dx^2))*(u(2)-u(1));
for i = 2:n-1
    dub(i) = (dt^2/(2*dx^2))*(u(i+1)-2*u(i)+u(i-1));
end
dub(n) = (dt^2/(dx^2))*(u(n)-u(n-1));

% ------------------------------------------------------------------

function fl = flux(h,u);
parameters;
n=length(u);
```

```

fl = h*0;
fl(1) = ((u(1)^2*dt^2)/(2*dx^2))*2*(h(2)-h(1));
for i = 2:n-1
    fl(i) = u(i)^2*dt^2/(2*dx^2)*(h(i+1)-2*h(i)+h(i-1));
end
fl(n) = (u(n)^2*dt^2/(2*dx^2))*(h(n)-h(n-1));

% ---------------------------------------------------------------------------

function udh = derh(h,u);
parameters;
n=length(u);

udh = h*0;
udh(1) = (u(1)*dt/dx)*(h(2)-h(1));
for i = 2:n-1
    udh(i) = ((u(i)*dt)/(2*dx))*(h(i+1)-h(i-1));
end
udh(n) = (u(n)*dt/dx)*(h(n)-h(n-1));
```

A.1.8 shear.m

```
function Δh = shear(u)

% The function "shear(u)" takes on a velocity profile and
% computes the pressure drop over each sylinder shell by means
% of the shear stress between the shells. The average pressure
% drop/headloss for the whole crossection of the pipe is found
% after computing the contribution from each sylinder shell.

parameters;

n = length(u);

Δp=0;
for k=1:NR-1
    Δp=Δp+4*nu*rho*(dr^2*dx/R^4)*k^2*(u(k+1)-u(k));
end

Δh=Δp/(rho*g);
```

A.1.9 average.m

```
function AvVdt = average(Vdt)
```

```

parameters;

n = length(Vdt);

AvVdt=0;
for k = 1:n
    AvVdt=AvVdt+2*(k/n^2)*Vdt(k);
end
```

Appendix B

Simulink Model

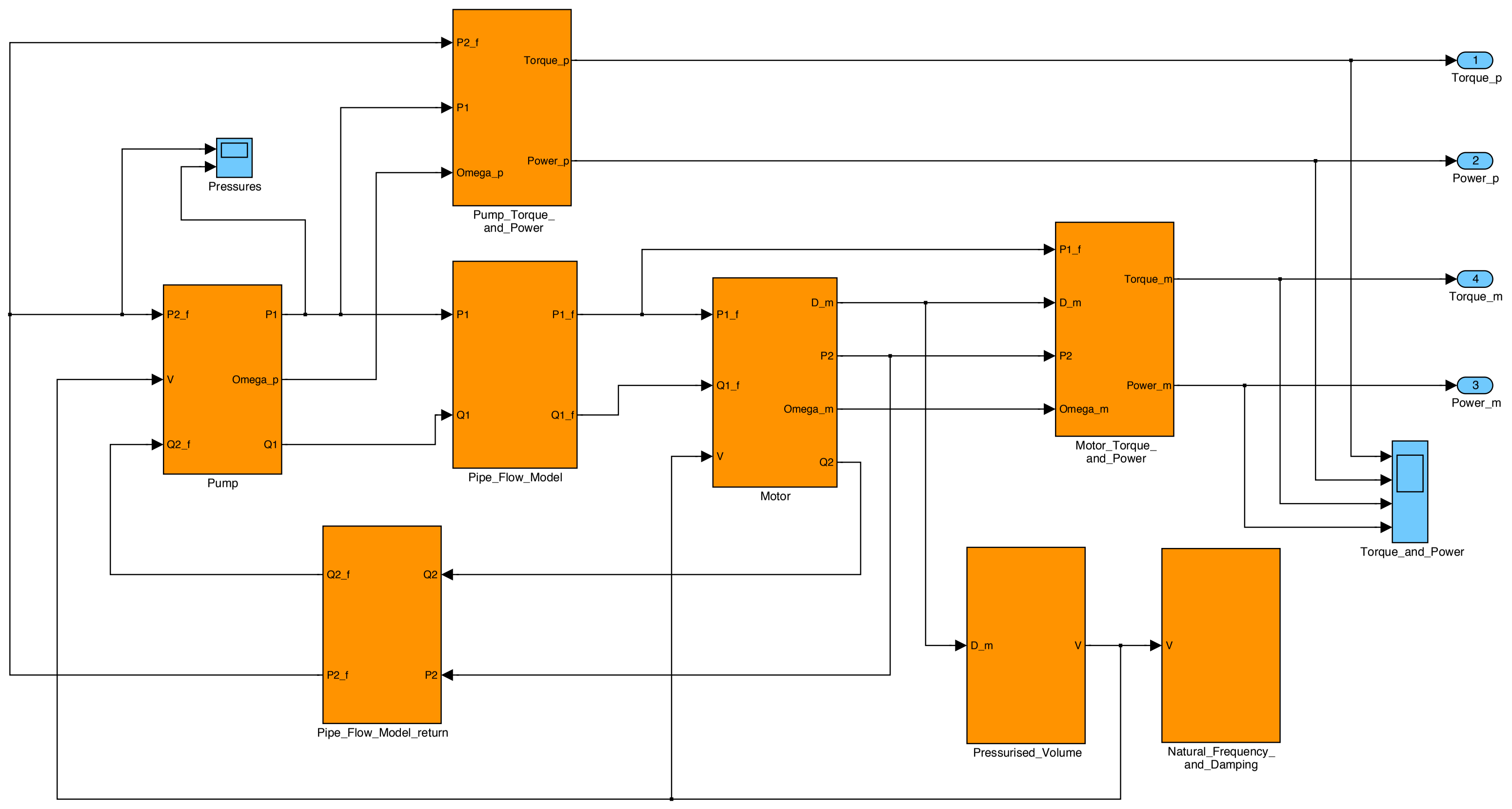
B.1 paramters.m

```
% Pipe

L = 30;
A = 0.05;
B = 1.5e9;


% Rotor

Inertia_rt = 100000;

Torque_rt_max=310e5*1.800679e-3;
Torque_rt_min=40e5*1.800679e-3;

Torque_rt = 20000;
Step_torque_rt = 20000;
Step_time=500;


% Pump

Omega_p_min = 1.7438;
Omega_p_max = 5.7596;
Omega_p_start = 4.1888;
D_p = 1.800679e-3;
Cl_p = 7.5e-12;
Cf_p = 0.01;
Cd_p = 7.5e-12;


% Motor
```

```

Omega_m_start = 159;
Omega_m_set = 157;
Rotary_Inertia_m = 1500;

D_m_min = 2.0e-5;
D_m_max = 8.0e-5;

Cl_m = 7.5e-12;
Cd_m = 7.5e-12;
C_m = .005;


% Generator
Torque_vector = 1e3 * [5 10 15 20 25 30 35 40 45 50 55];
Gain_vector = [2.7 2.2 1.9 1.7 1.5 1.30 1.25 1.15 1.05 .95 .87];


% PID control 40.000
P_gain = 9.7286e-006;
I_gain = 2.5e-007;
D_gain = 4.9985e-5;


 figure1 = figure('PaperSize',[20.98 29.68]);

 axes('Parent',figure1,'XTickLabel',{'5000','30000','55000'},...
     'XTick',[5000 3e+004 5.5e+004]);
 box('on');
 hold('all');

 plot(Torque_vector,Gain_vector);
 xlabel('Rotor torque','Interpreter','latex','FontSize',11);
 ylabel('Gain','Interpreter','latex','FontSize',11);
 title('Generator gains','Interpreter','latex','FontSize',12);
```

B.2 Kvalsund.mdl

Kvalsund

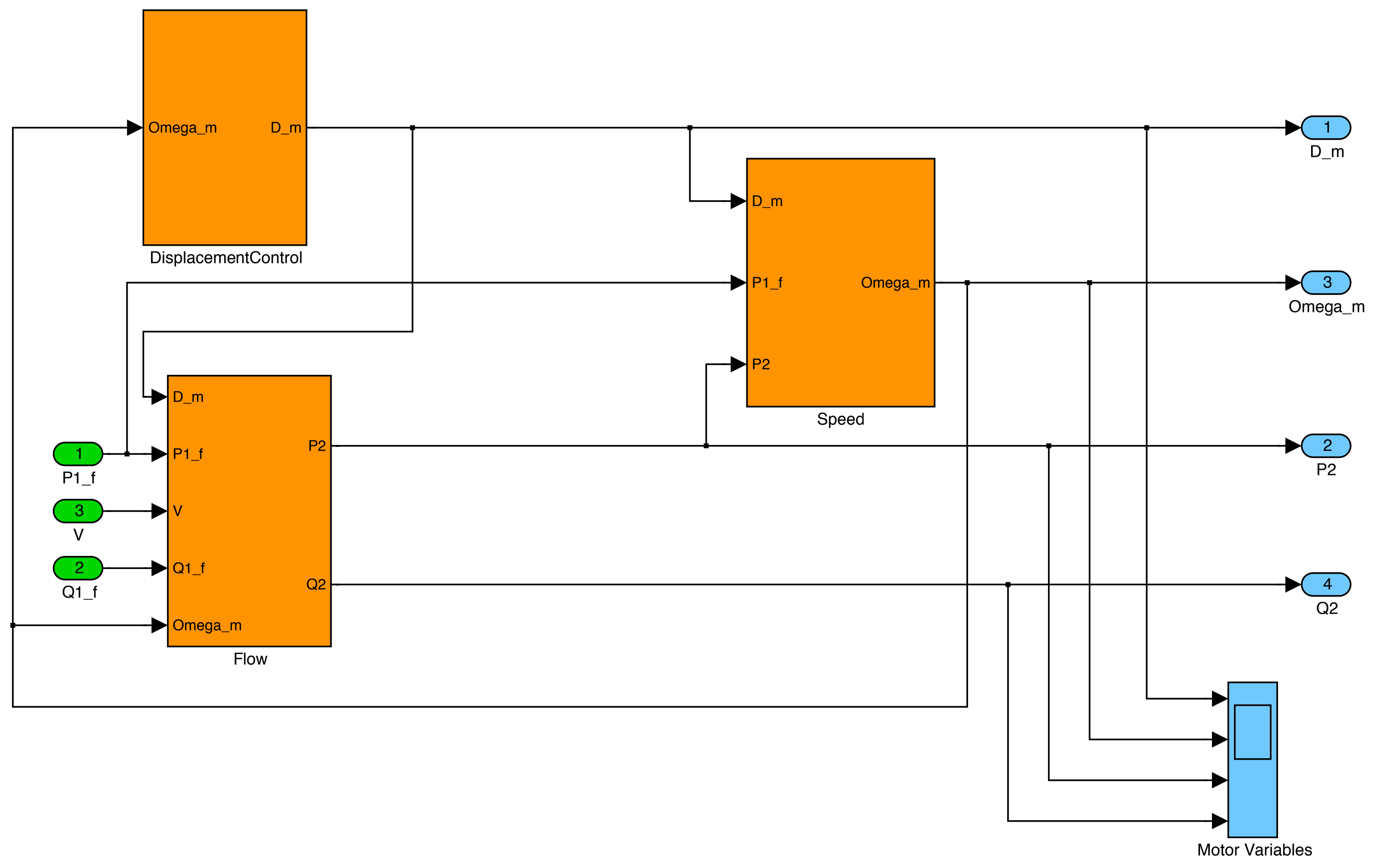


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 1/14

Kvalsund/Motor

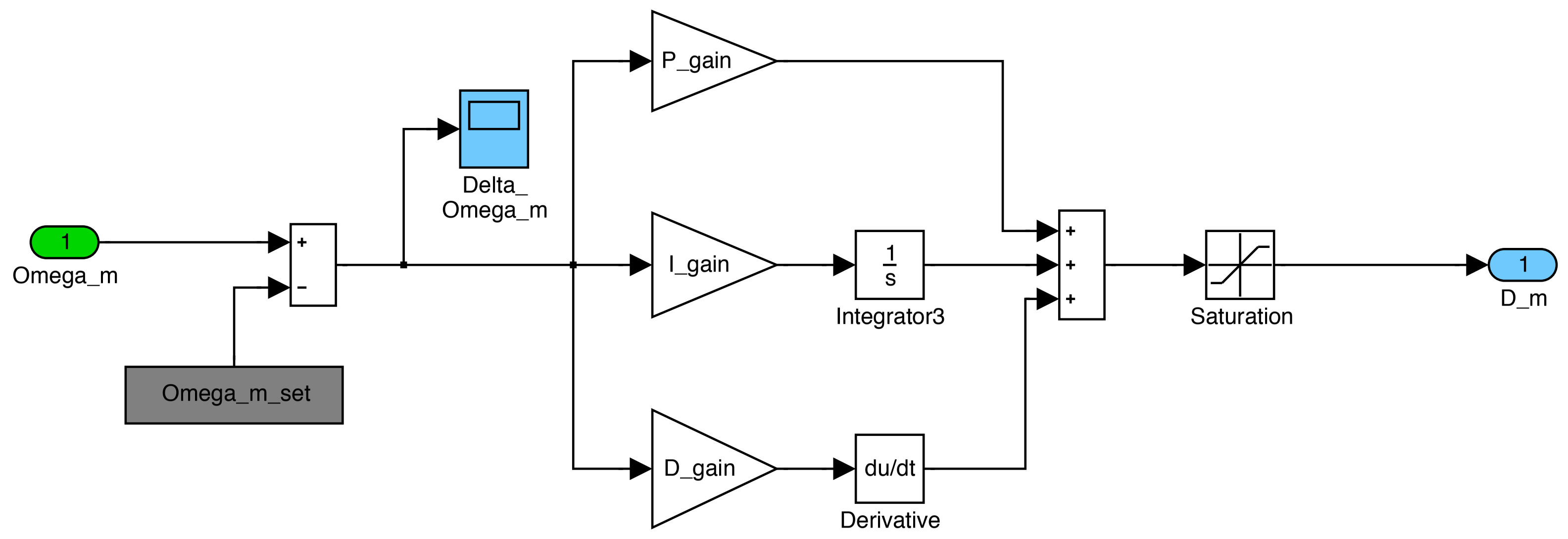


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 2/14

Kvalsund/Motor/DisplacementControl

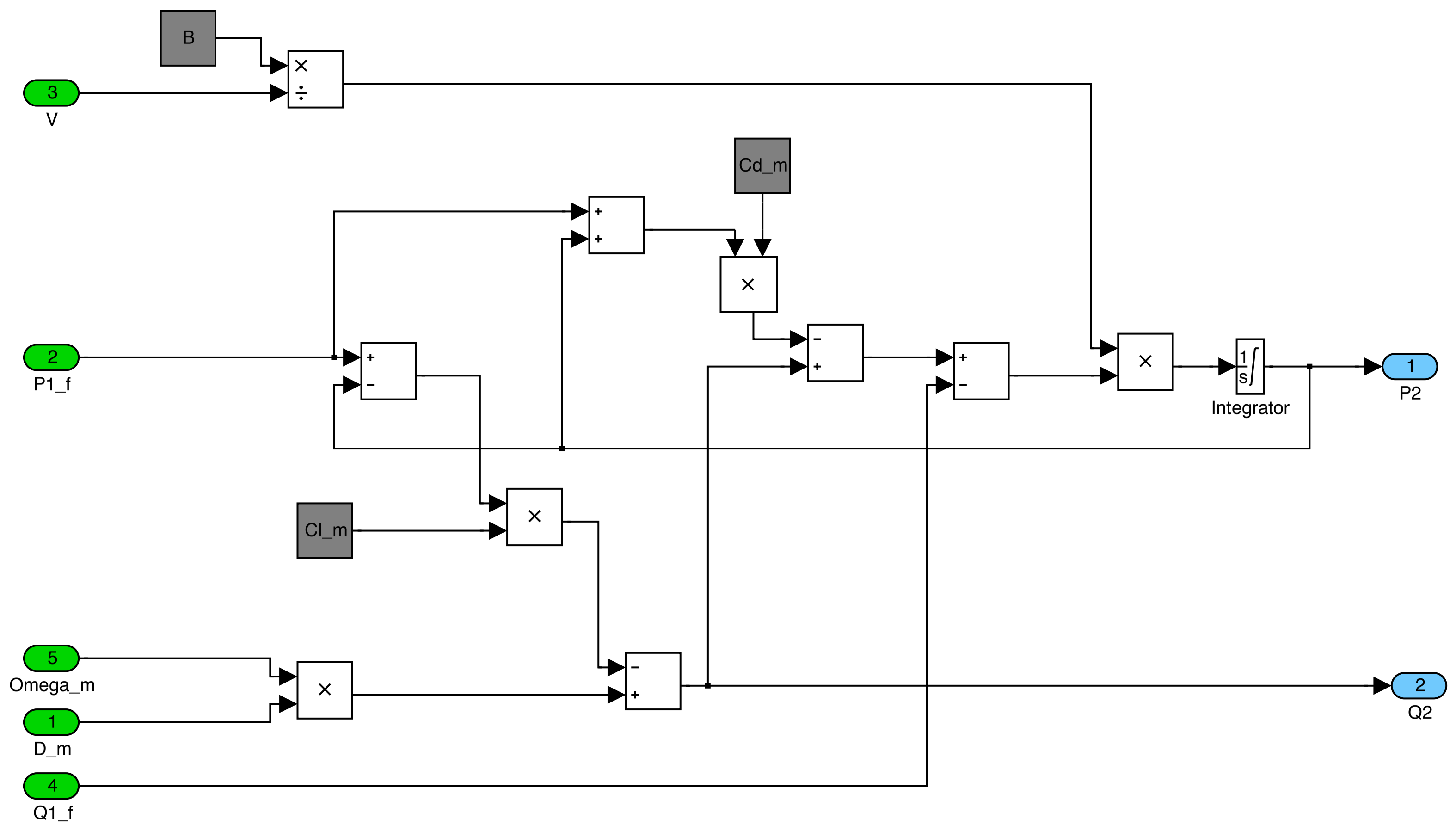


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 3/14

Kvalsund/Motor/Flow

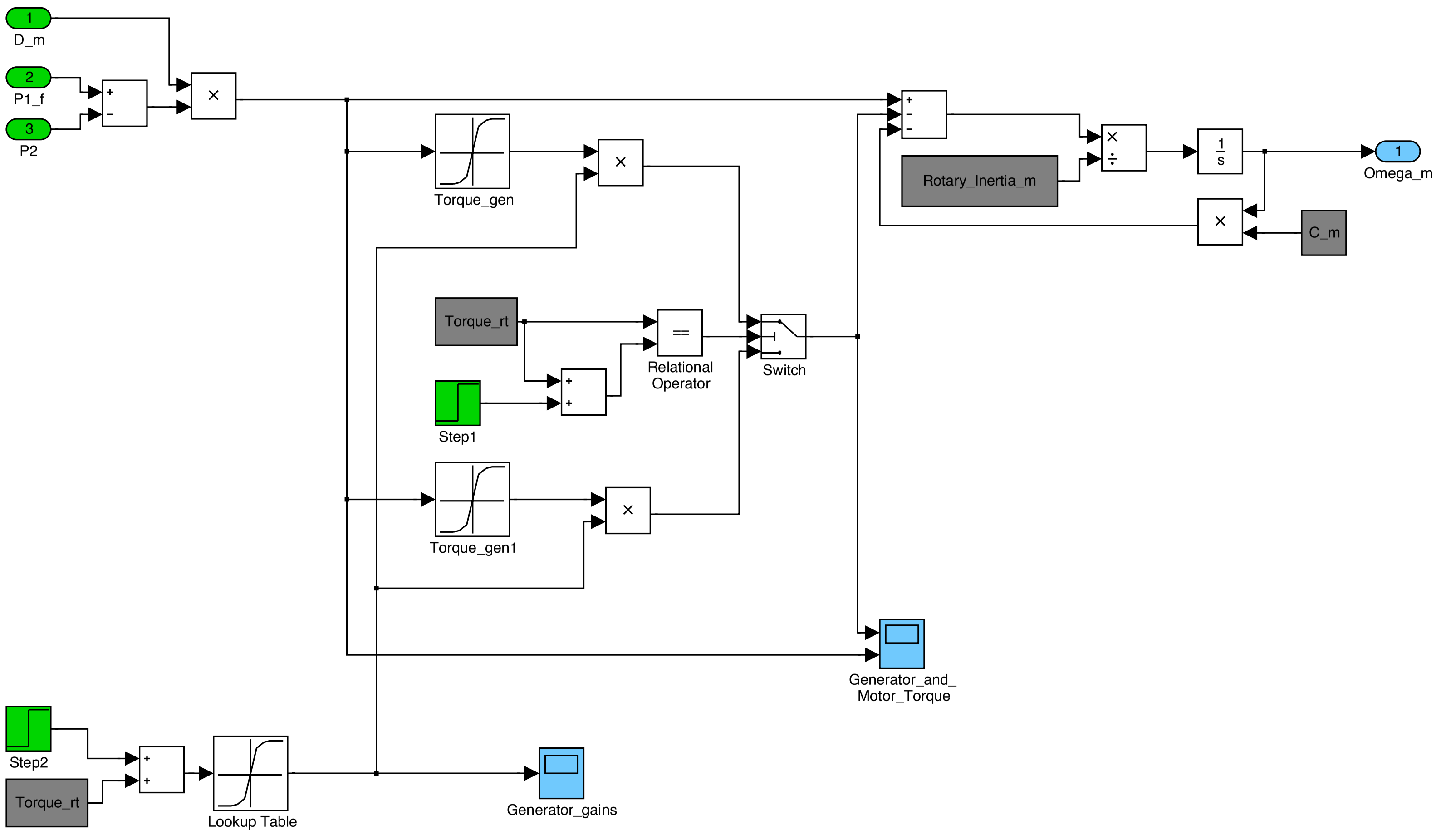


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 4/14

Kvalsund/Motor/Speed

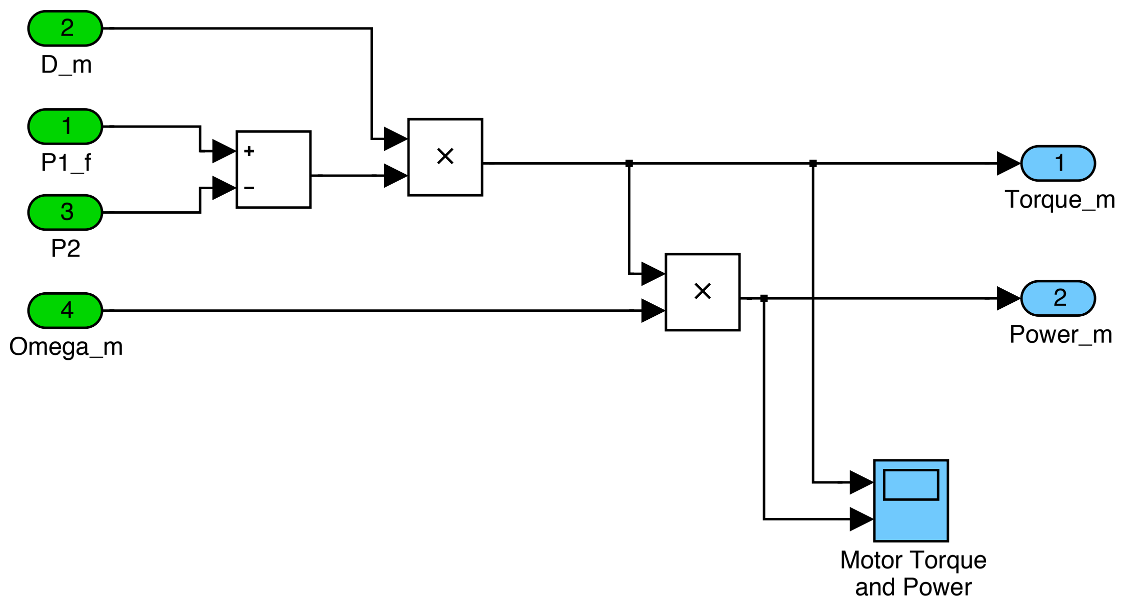


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 5/14

Kvalsund/Motor_Torque_ and_Power

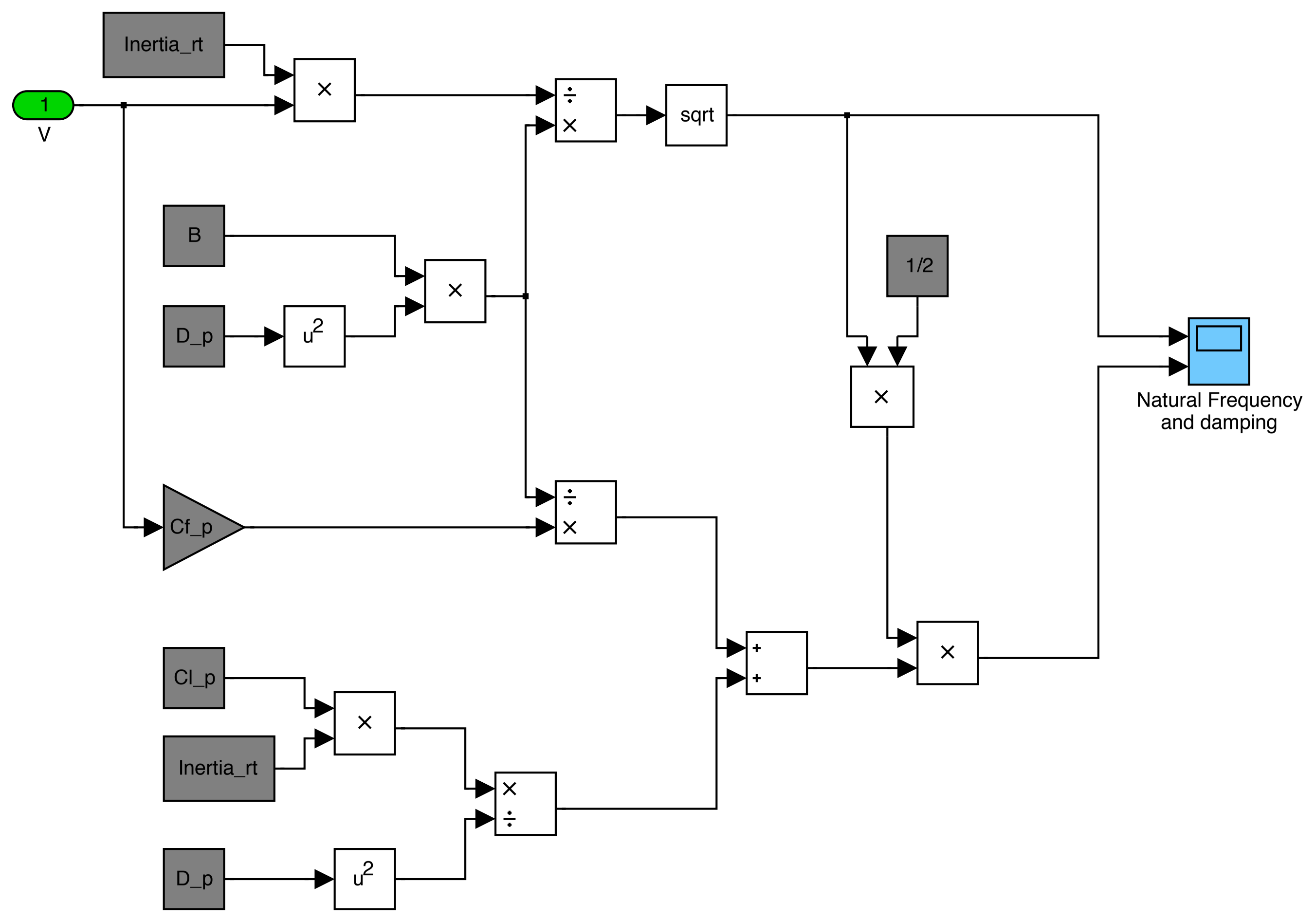


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 6/14

Kvalsund/Natural_Frequency_ and_Damping

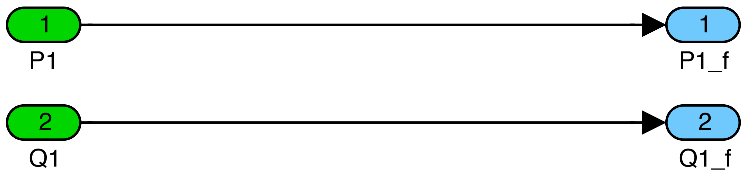


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 7/14

Kvalsund/Pipe_Flow_Model

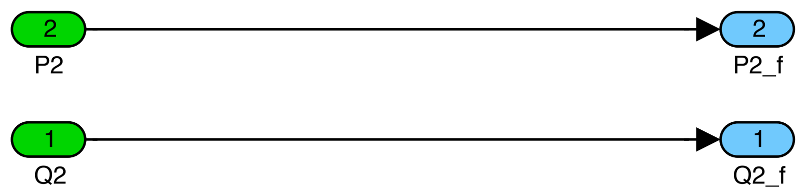


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 8/14

Kvalsund/Pipe_Flow_Model_return

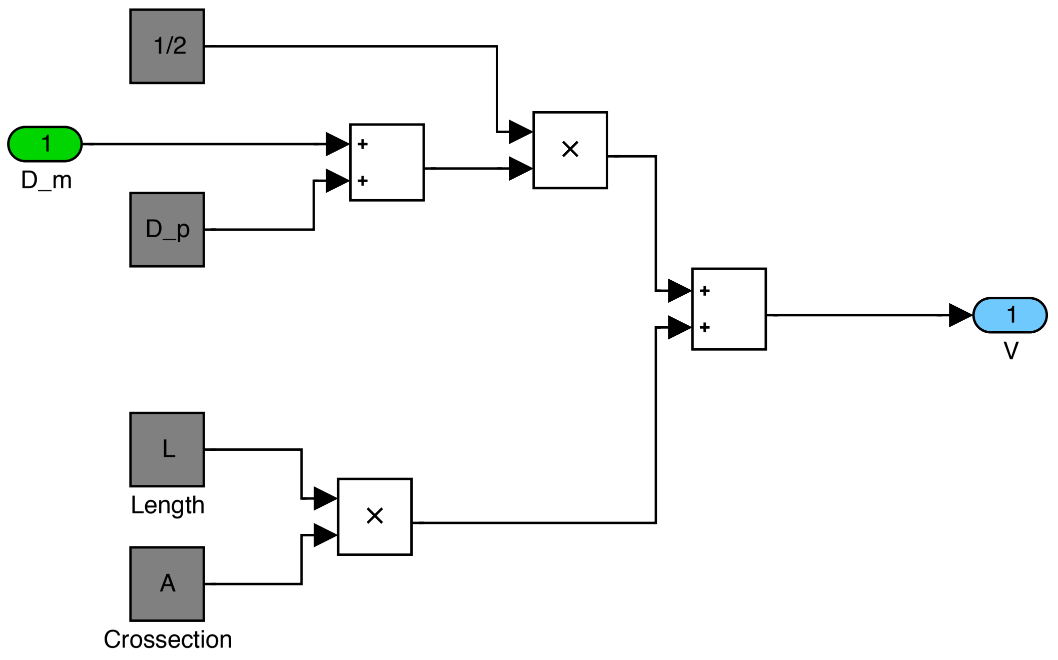


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printted 24–Jun–2007 17:49

page 9/14

Kvalsund/Pressurised_Volume

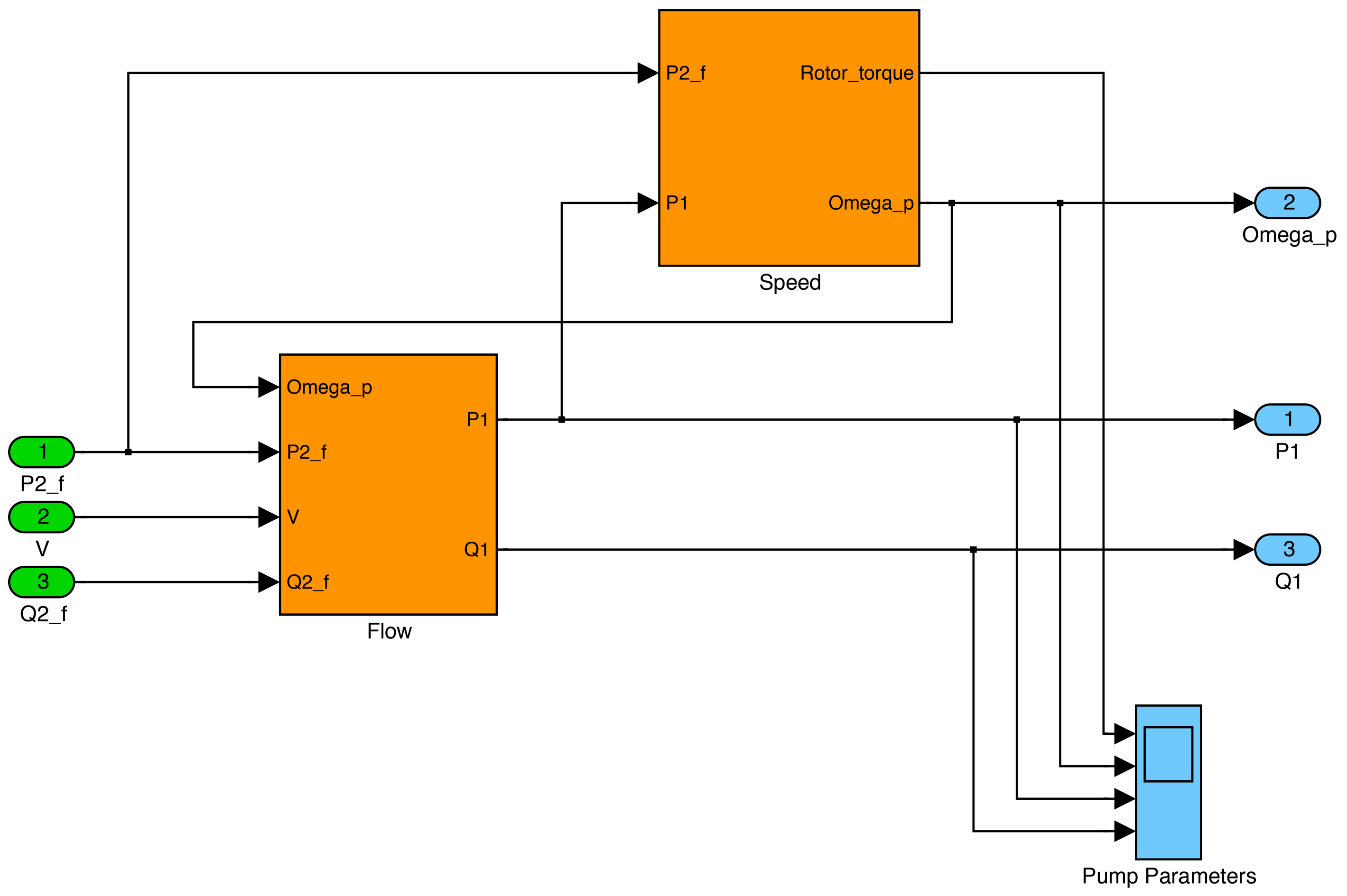


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 10/14

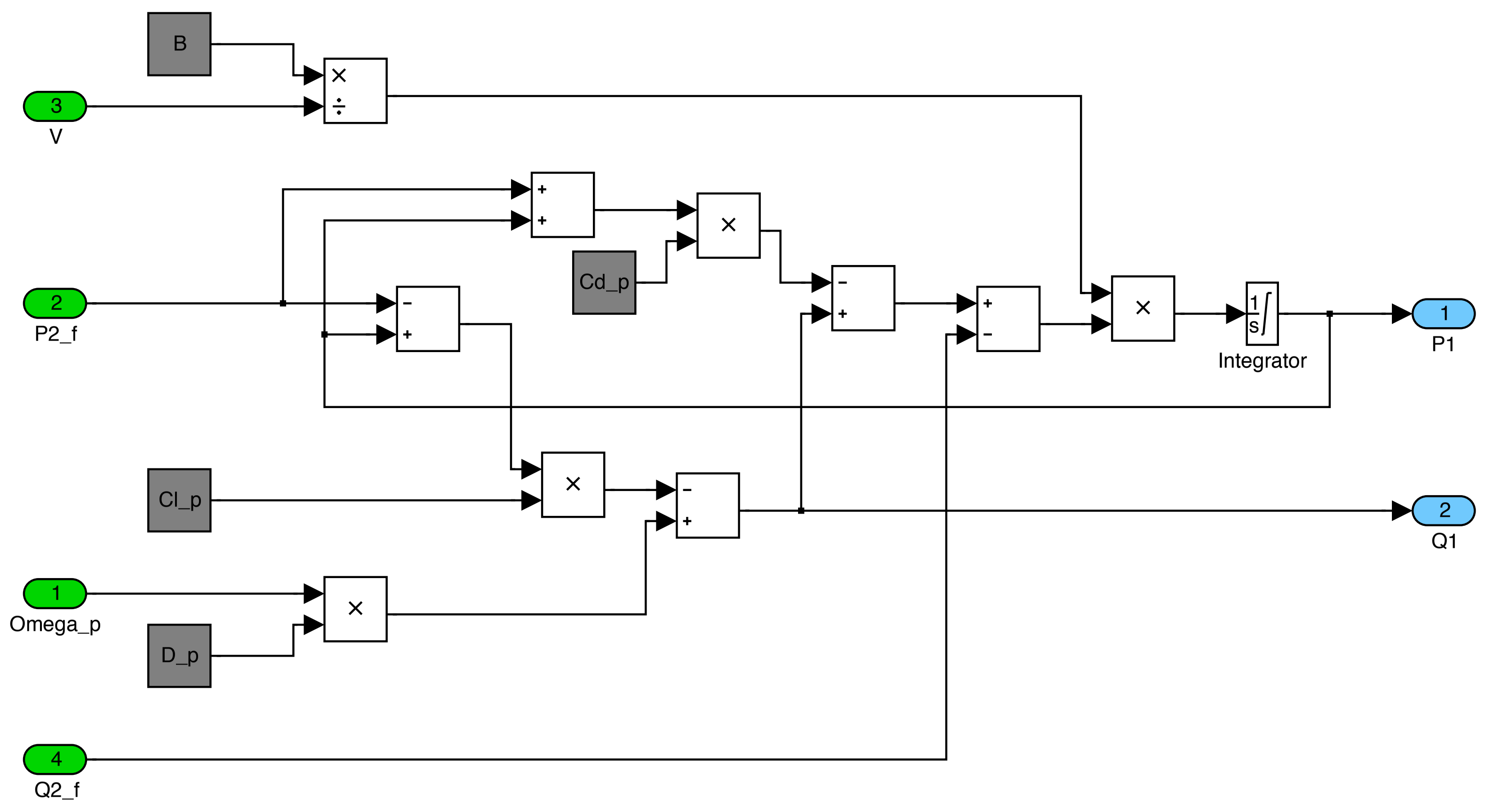
Kvalsund/Pump



/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49 page 11/14

Kvalsund/Pump/Flow

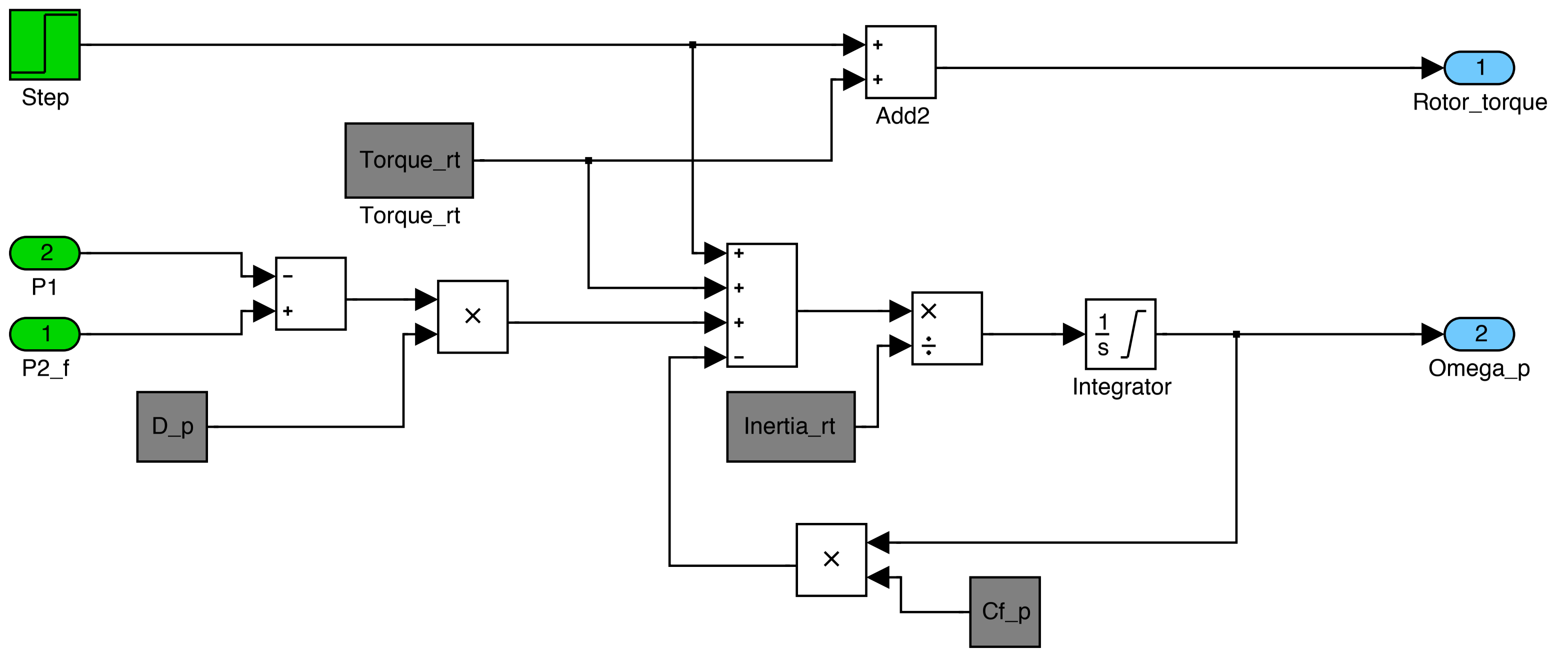


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 12/14

Kvalsund/Pump/Speed

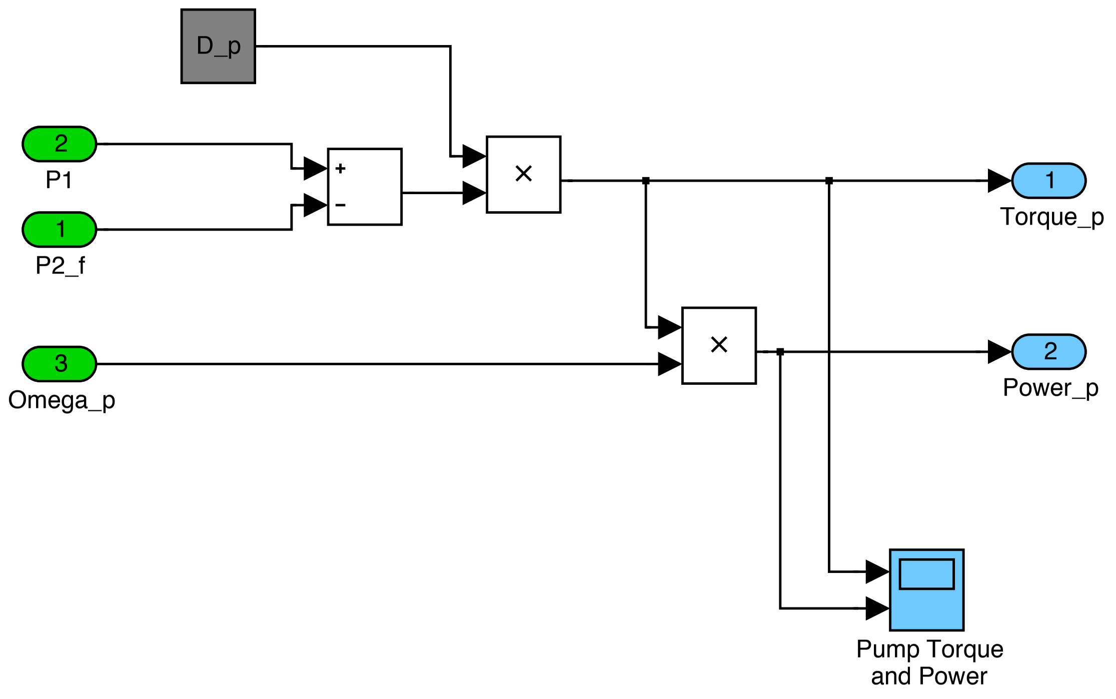


/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 13/14

Kvalsund/Pump_Torque_ and_Power



/Users/magnusgulstad/Desktop/Skole/Diplom/Simulink/pidcontrolled/Kvalsund.mdl

printed 24–Jun–2007 17:49

page 14/14